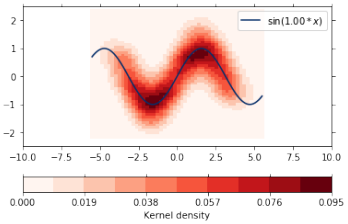
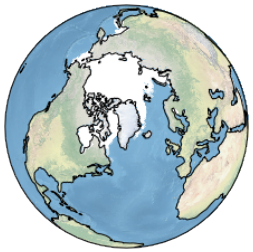
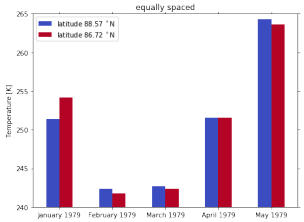
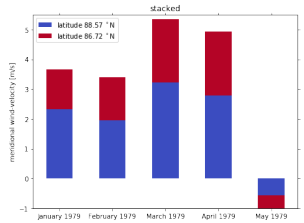
Psyplot

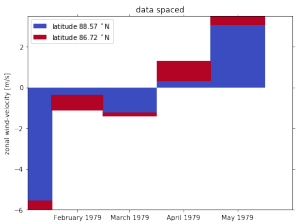
Interactive data analysis and visualization with Python

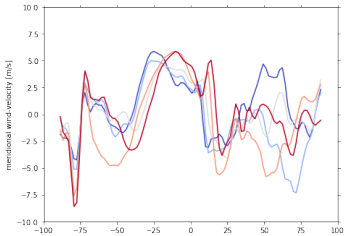












Motivation

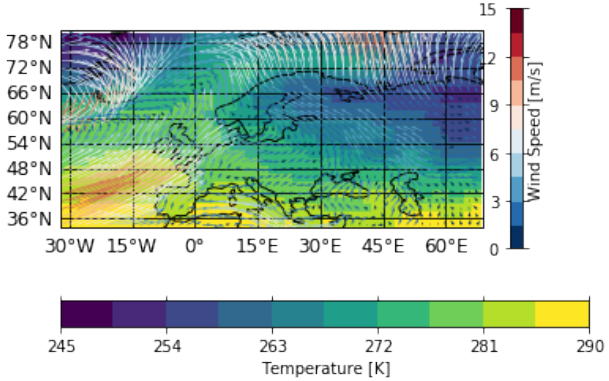
Installation

Framework

Author

Help

GUI



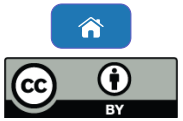
psyplot.readthedocs.io

How to navigate

UNIL | Université de Lausanne

This presentation has been prepared for a PICO presentation at the EGU 2018 in Vienna, Austria. To facilitate the navigation, a lot of hyperlinks are used. Almost every item in this presentation is clickable:

- click buttons like this to be linked to other connected frames
- click the navigation bar above with the sections, Home, Help, About, etc. (including the dots) to navigate in the presentation
- click on navigation buttons like this < 1/2 > to show you more of the current frame.
- click on many of the images to get more information or a close-up
- click the  or the  icon to go back to the menu
- click the buttons at the lower left and lower right that bring you to the next slide



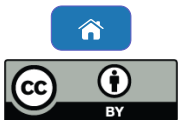
Title page⇒

How to navigate

UNIL | Université de Lausanne

This presentation has been prepared for a PICO presentation at the EGU 2018 in Vienna, Austria. To facilitate the navigation, a lot of hyperlinks are used. Almost every item in this presentation is clickable:

- click buttons like this to be linked to other connected frames
- click the navigation bar above with the sections, Home, Help, About, etc. (including the dots) to navigate in the presentation
- click on navigation buttons like this < 2/2 > to show you more of the current frame. For example this text.
- click on many of the images to get more information or a close-up
- click the  or the  icon to go back to the menu
- click the buttons at the lower left and lower right that bring you to the next slide



Title page⇒

Psyplot

Interactive data analysis and visualization with Python

EGU, Vienna, Austria, April 9th, 2018

Philipp Sommer

Davis Group, Institute of Earth Surface Dynamics (IDYST)
University of Lausanne

UNIL | Université de Lausanne

Author

UNIL | Université de Lausanne

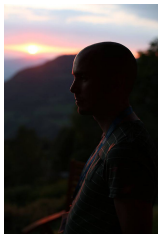
Philipp Sommer

Institute of Surface Dynamics (IDYST)
University of Lausanne (UNIL)
Bâtiment Géopolis - Room 4413
CH - 1015 Lausanne

E-mail philipp.sommer@unil.ch
Telefon +41 21 692 35 40
Github https://github.com/Chilipp/
Webpage https://wp.unil.ch/davisgroup/philipp-sommer/

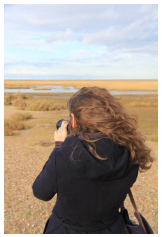
Acknowledgments

Dr. Basil Davis, palaeoclimatologist



Manuel Chevalier, palaeoclimatologist

Andrea Kay, archaeologist



Leanne Phelps, human ecologist

Prof. Jed Kaplan, geographer, climate modeler, ...

Shawn Koppenhöfer, informatician

FNSNF
Fonds national suisse
Schweizerischer Nationalfonds
Fondo nazionale svizzero
Swiss National Science Foundation

UNIL | Université de Lausanne

- Olivier Cartapanis

⇐Author

Abstract⇒

Abstract

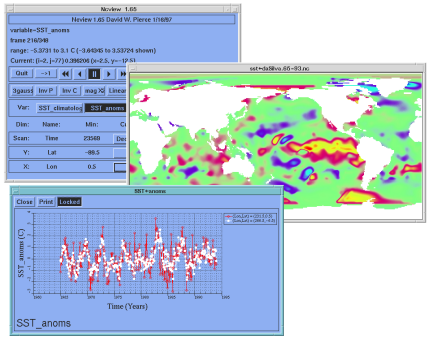
UNIL | Université de Lausanne

Psyplot: Interactive data analysis and visualization with Python

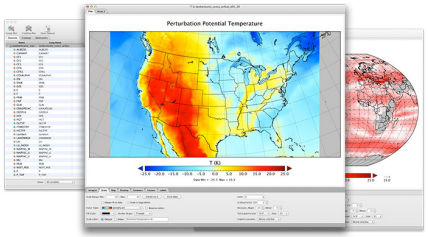
The development, usage and analysis of climate models often requires the visualization of the data. This visualization should ideally be nice looking, simple in application, fast, easy reproducible and flexible. There exist a wide range of software tools to visualize model data which however often lack in their ability of being (easy) scriptable, have low flexibility or simply are far too complex for a quick look into the data. Therefore, we developed the open-source visualization framework psyplot that aims to cover the visualization in the daily work of earth system scientists working with data of the climate system. It is build (mainly) upon the python packages matplotlib, cartopy and xarray and integrates the visualization process into data analysis. This data can either be stored in a NetCDF, GeoTIFF, or any other format that is handled by the xarray package. Due to its interactive nature however, it may also be used with data that is currently processed and not already stored on the hard disk. Visualizations of rastered data on the glob are supported for rectangular grids (following or not following the CF Conventions) or on a triangular grid (following the CF Conventions (like the earth system model ICON) or the unstructured grid conventions (UGRID)). Furthermore, the package visualizes scalar and vector fields, enables to easily manage and format multiple plots at the same time. Psyplot can either be used with only a few lines of code from the command line in an interactive python session, via python scripts or from through a graphical user interface (GUI). Finally, the framework developed in this package enables a very flexible configuration, an easy integration into other scripts using matplotlib.

Data analysis tools

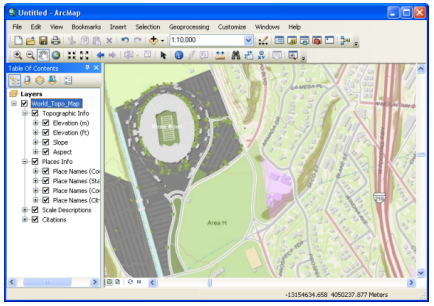
UNIL | Université de Lausanne



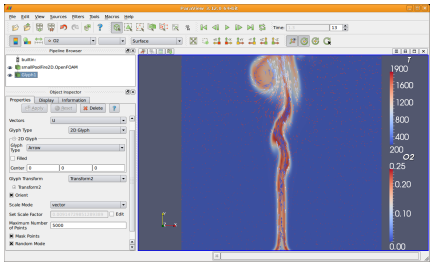
ncview



panoply

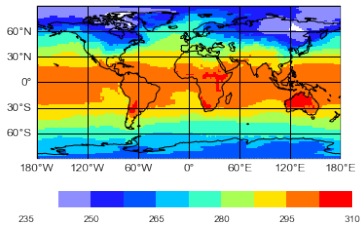


ArcGIS



Paraview

- Analysis tools mainly focus on visualization
  $\Rightarrow$ No flexible analysis for differing data types
- Requires long scripts in R, Python, Matlab, etc.




```
        xvert, yvert = nodes
        xvert = xvert.values
        yvert = yvert.values
        loc = var.attrs.get('location', 'face')
        if loc == 'face':
            triangles = get_coord(
                mesh.attrs.get('face_node_connectivity', '')).values
            if triangles is None:
                raise ValueError(
                    "Could not find the connectivity information!")
        elif loc == 'node':
            triangles = None
        else:
            raise ValueError(
                "Could not interprete location attribute (%s) of mesh "
                "variable %s!" % (loc, mesh.name))

        if convert_radian:
            for coord in nodes:
                if coord.attrs.get('units') == 'radian':
                    coord = coord * 180. / np.pi
        if src_crs is not None and src_crs != target_crs:
            if target_crs is None:
                raise ValueError(
                    "Found %s for the source crs but got None for the "
                    "target_crs!" % (src_crs, ))
            xvert = xvert[triangles].ravel()
            yvert = yvert[triangles].ravel()
            arr = target_crs.transform_points(src_crs, xvert, yvert)
            xvert = arr[:, 0]
            yvert = arr[:, 1]
            if loc == 'face':
                triangles = np.reshape(range(len(xvert)), (len(xvert) // 3,
                                                           3))

        return Triangulation(xvert, yvert, triangles)

    @staticmethod
    @docstrings.dedent
    def decode_coords(ds, gridfile=None, inplace=True):
        """
        Reimplemented to set the mesh variables as coordinates

        Parameters
        ----------
        %(CFDecoder.decode_coords.parameters)s

        Returns
        -------
        %(CFDecoder.decode_coords.returns)s"""
        extra_coords = set(ds.coords)
        for var in six.itervalues(ds.variables):
            if 'mesh' in var.attrs:
```

⇐Abstract

Psyplot⇒

Psyplot: Interactive analysis

UNIL | Université de Lausanne

Current data analysis:

1. ncview, Panoply, etc. to look into the data
2. R or Python for analysis and calculations

3. Make figures with R or Python for publications

Psyplot: Interactive analysis

UNIL | Université de Lausanne

Current data analysis:

1. ncview, Panoply, etc. to look into the data
2. R or Python for analysis and calculations
3. Make figures with R or Python for publications

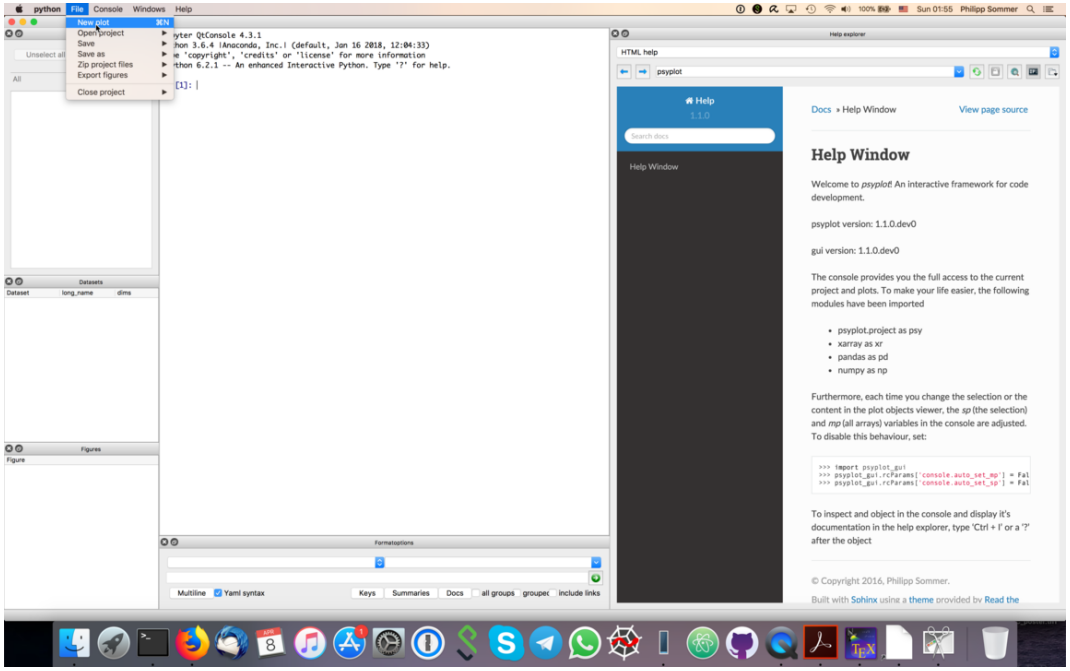
Combined strategy: psyplot

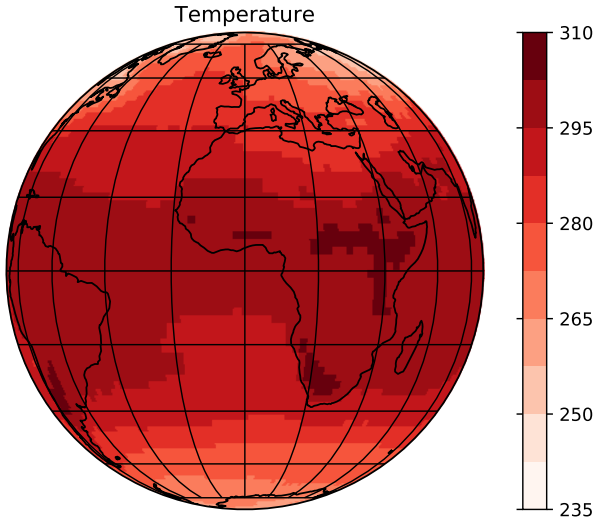
1. graphical user interface (like ncview or Panoply)
2. built-in python command line for calculations, etc., and can be easily implemented in python scripts
3. Uses matplotlib to make publication-ready and reproducible figures

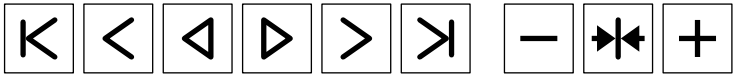
Demo

UNIL | Université de Lausanne

```
In [1]: psy.plot.mapplot(
            'demo.nc', name='t2m', projection='ortho',
            cmap='Reds', cbar='r', title='%(long_name)s')
```







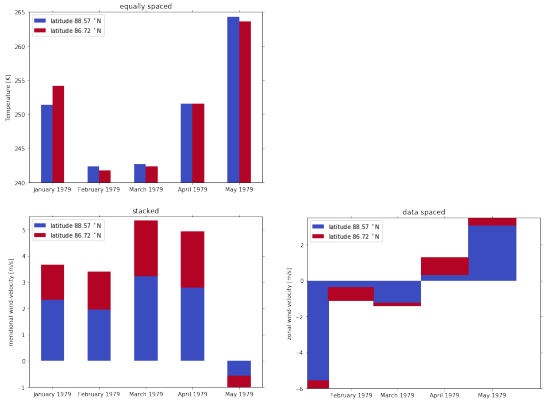
⇐Psyplot
Plot methods⇒

Plot methods

UNIL | Université de Lausanne

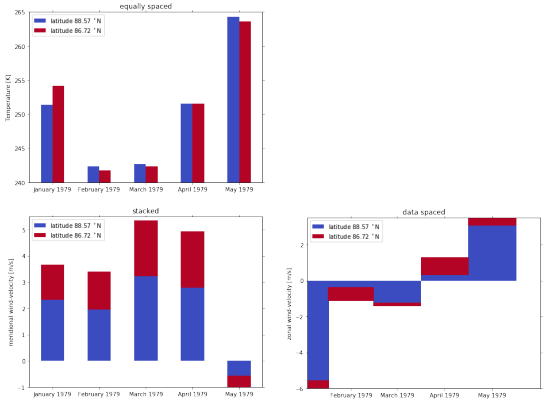
Plot methods

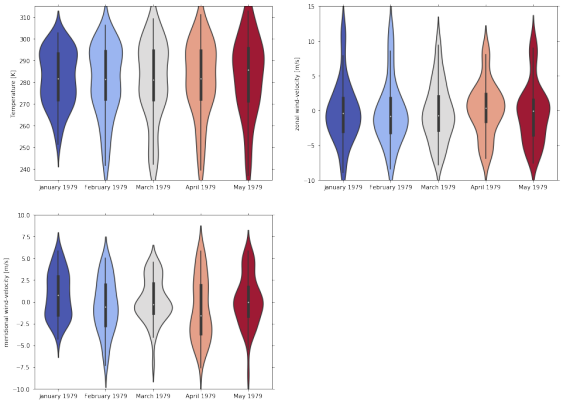
UNIL | Université de Lausanne



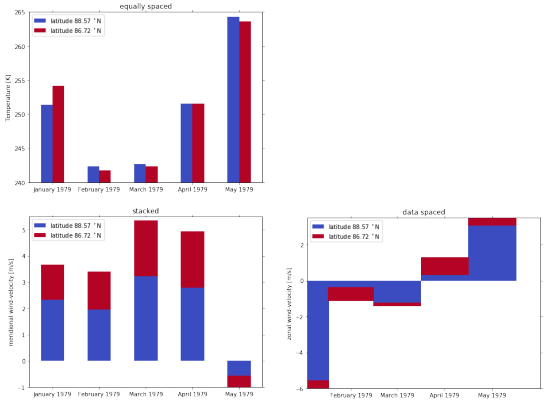
Plot methods

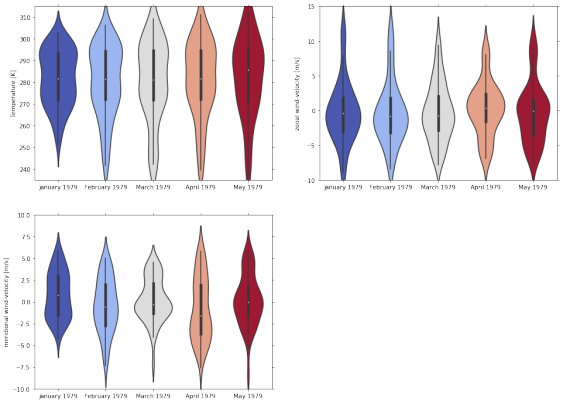
UNIL | Université de Lausanne

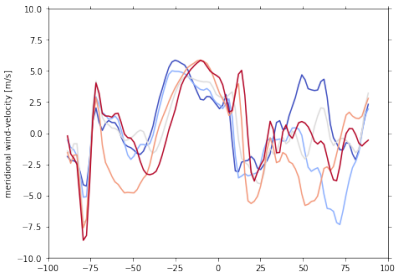




Plot methods







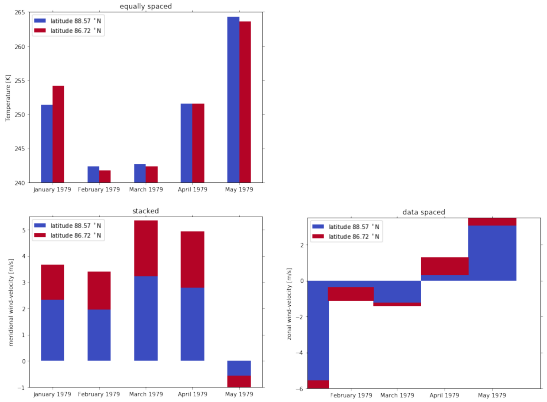
4/10

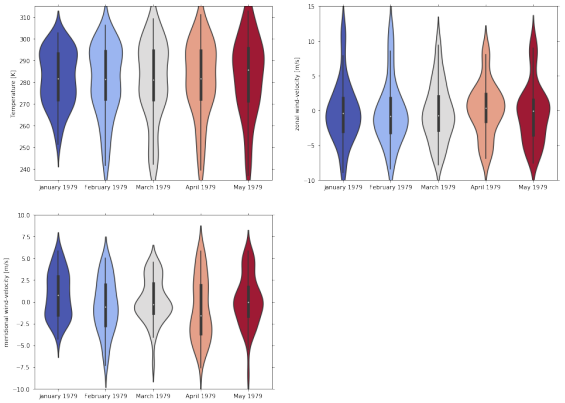
⇐Demo

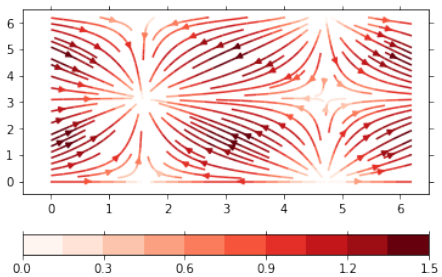
Installation⇒

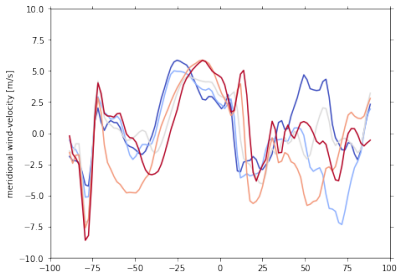
Plot methods

UNIL | Université de Lausanne



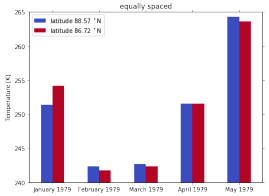


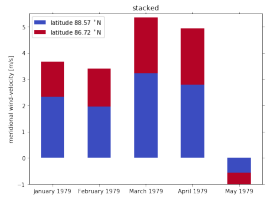


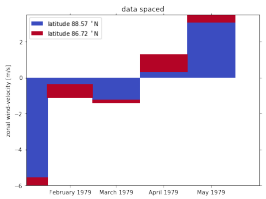


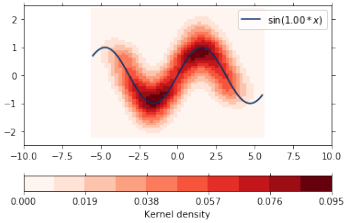
Plot methods

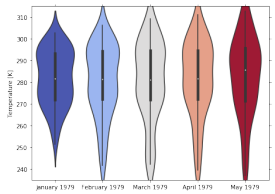
UNIL | Université de Lausanne

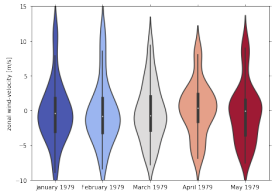


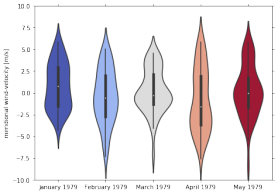


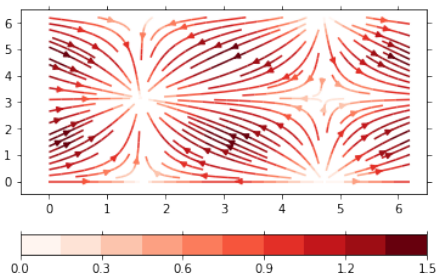


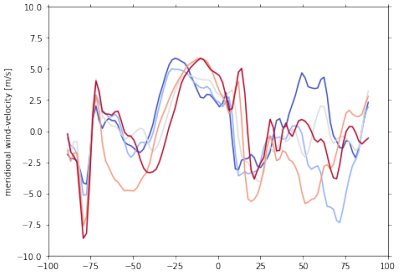












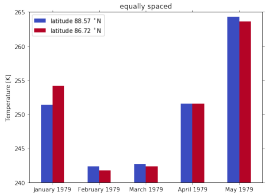
6/10

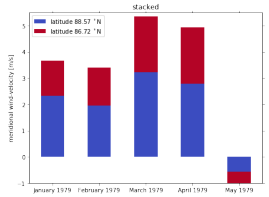
⇐Demo

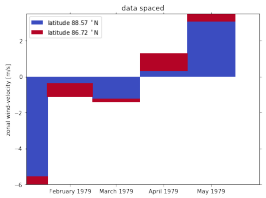
Installation⇒

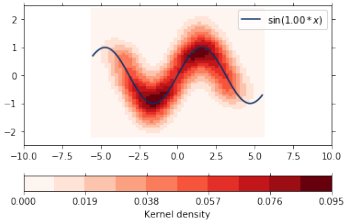
Plot methods

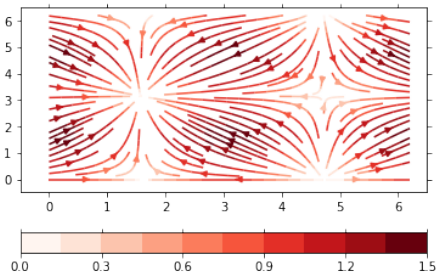
UNIL | Université de Lausanne

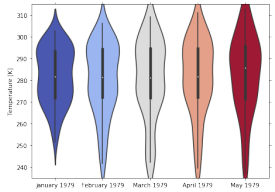


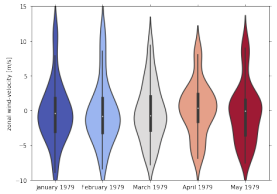


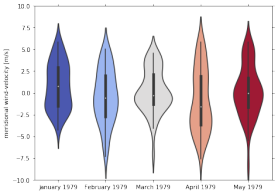


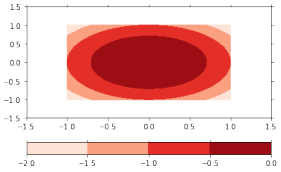


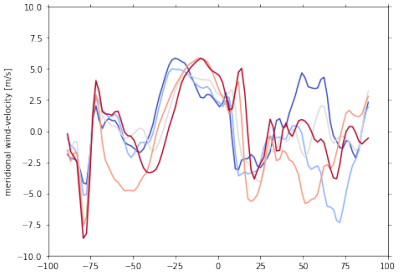












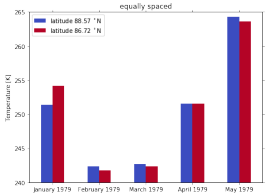
7/10

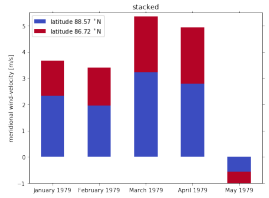
⇐Demo

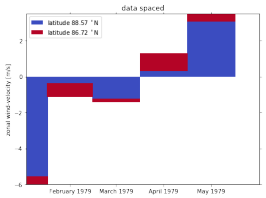
Installation⇒

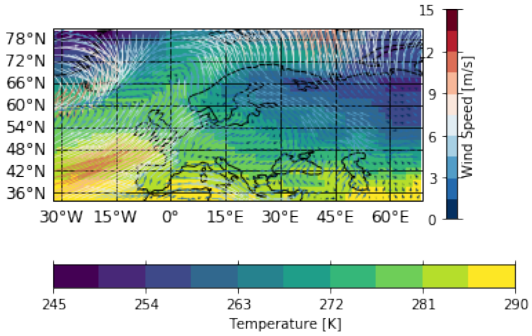
Plot methods

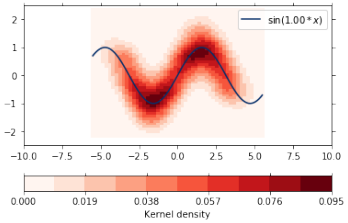
UNIL | Université de Lausanne

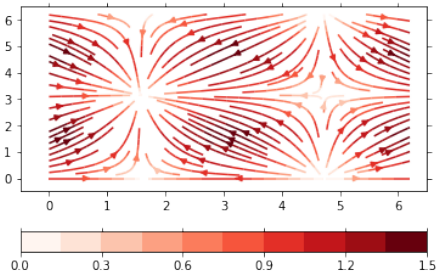


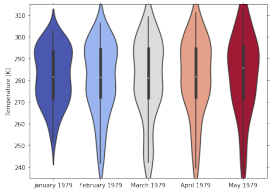


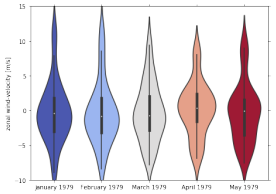


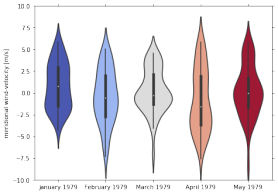


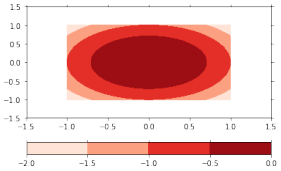


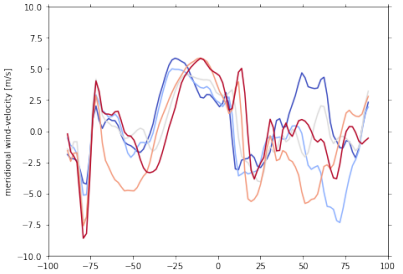








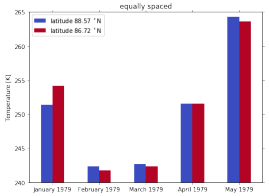


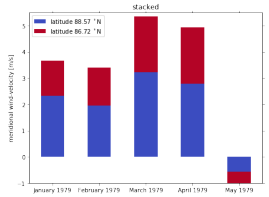


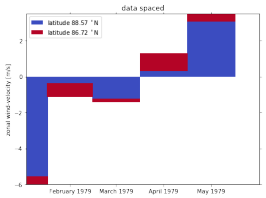
⇐Demo
Installation⇒

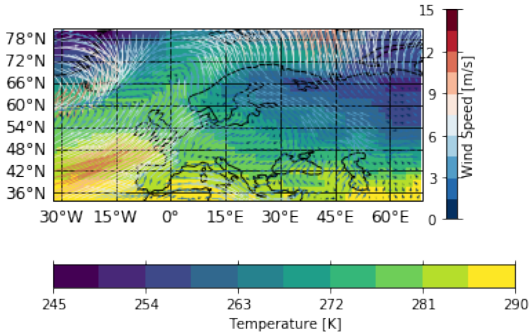
Plot methods

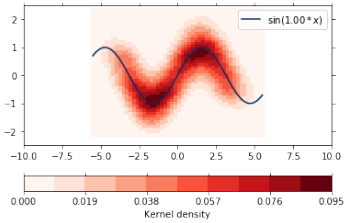
UNIL | Université de Lausanne

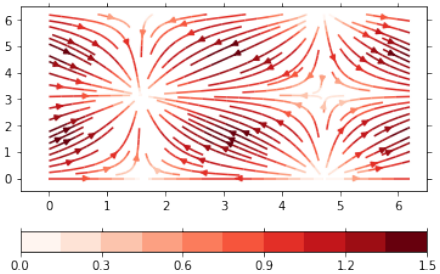


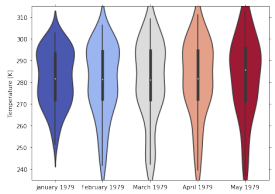


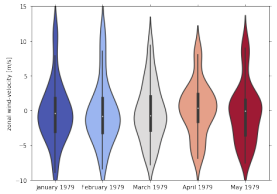


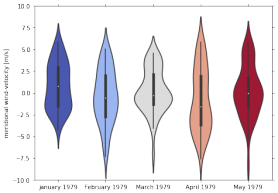


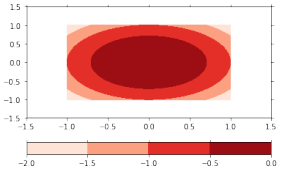


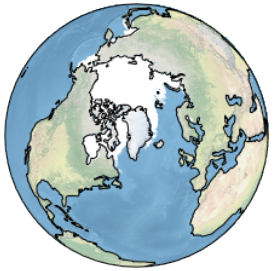


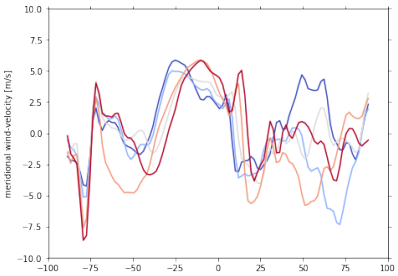






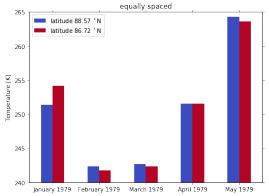


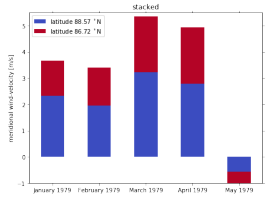


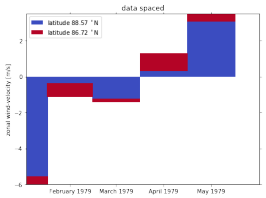


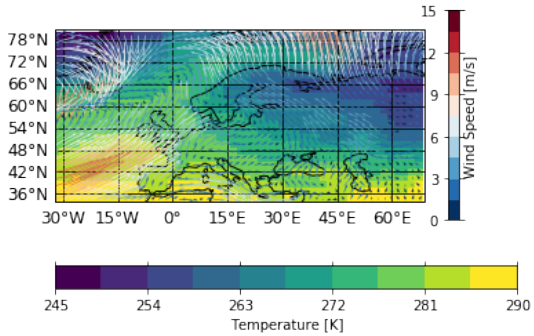
Plot methods

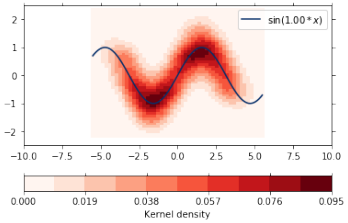
UNIL | Université de Lausanne

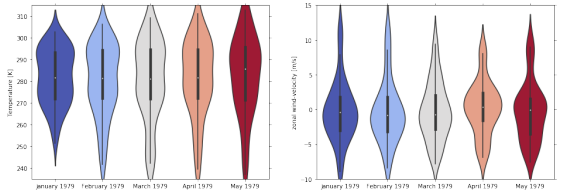


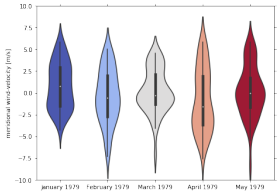


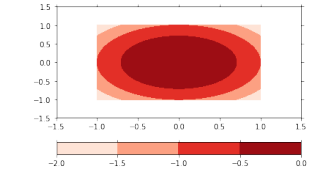


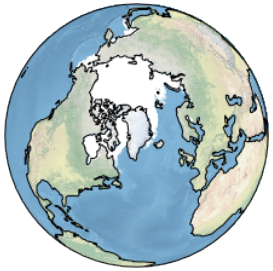


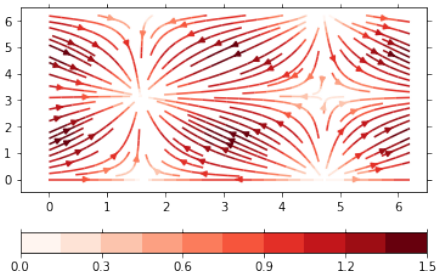


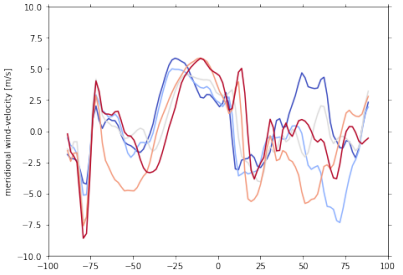












more on
psyplot.readthedocs.io

⇐Demo Installation⇒

Installation

UNIL | Université de Lausanne

The package is written in Python and hosted on Github:
https://github.com/Chilipp/psyplot
and can be installed

1. from source:

```
git clone https://github.com/Chilipp/psyplot.git
cd psyplot
python setup.py install
```

2. using pip (https://pypi.org/)

```
pip install psyplot
```

3. or using anaconda (https://anaconda.org/conda-forge)

```
conda install -c conda-forge psyplot psyplot-gui psy-maps
```

or via stand-alone installers (see
https://psyplot.readthedocs.io/en/latest/install.html).

Dependencies

UNIL | Université de Lausanne

The python package (Sommer, 2017a), depends mainly on

numpy and scipy: numeric python libraries (Jones et al., 2001)

matplotlib: python visualization package (Hunter, 2007)

xarray: for the data management (Hoyer and Hamman, 2017)

The graphical user interface is based on the psyplot-gui package (Sommer, 2017b) which is programmed using the Qt bindings of PyQt (Summerfield, 2007). psy-maps, the psyplot plugin for visualizing geo-referenced data, is based on cartopy (Met Office, 2010 - 2015).

The Psyplot Framework

UNIL | Université de Lausanne

1. The basis forms a variable in a (netCDF) dataset (e.g. temperature).

The Psyplot Framework

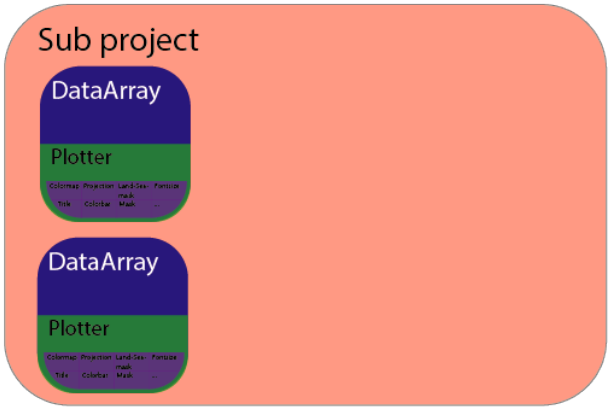
UNIL | Université de Lausanne

1. The basis forms a variable in a (netCDF) dataset (e.g. temperature).
2. It is plotted by a plotter to your choice (e.g. using `psy.plot.mapplot`.

The Psyplot Framework

UNIL | Université de Lausanne

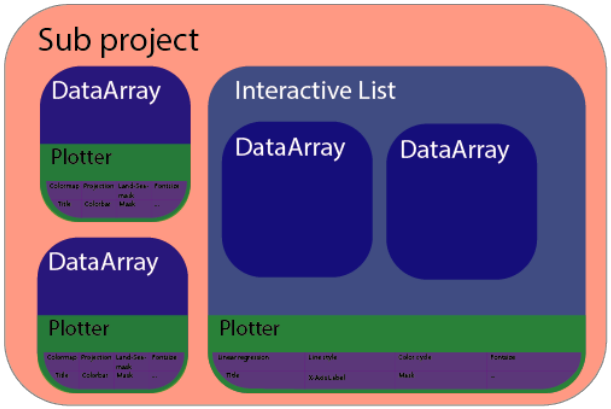
1. The basis forms a variable in a (netCDF) dataset (e.g. temperature).
2. It is plotted by a plotter to your choice (e.g. using `psy.plot.mapplot`.
3. Multiple data arrays (and plotters) form a project.



The Psyplot Framework

UNIL | Université de Lausanne

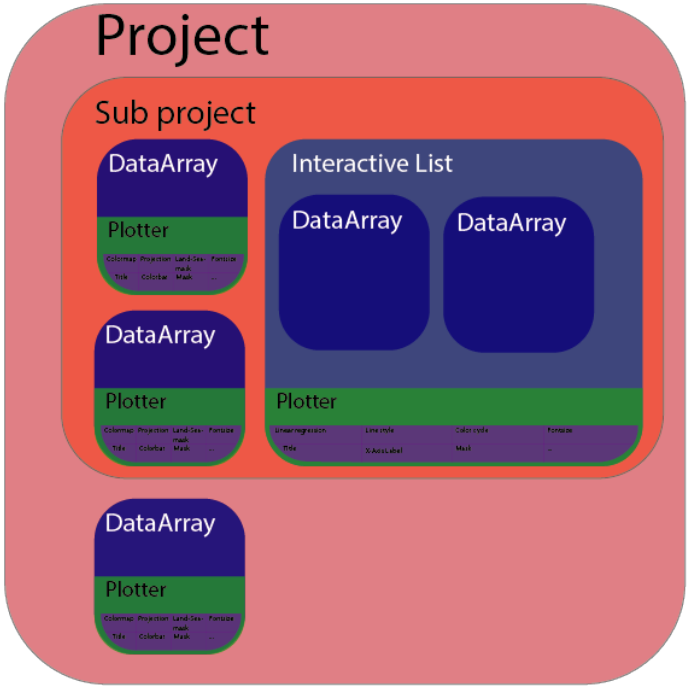
1. The basis forms a variable in a (netCDF) dataset (e.g. temperature).
2. It is plotted by a plotter to your choice (e.g. using `psy.plot.mapplot`.
3. Multiple data arrays (and plotters) form a project.
4. Some plotters also visualize multiple data arrays at once (e.g. line plots), that are then concatenated as an `InteractiveList`.



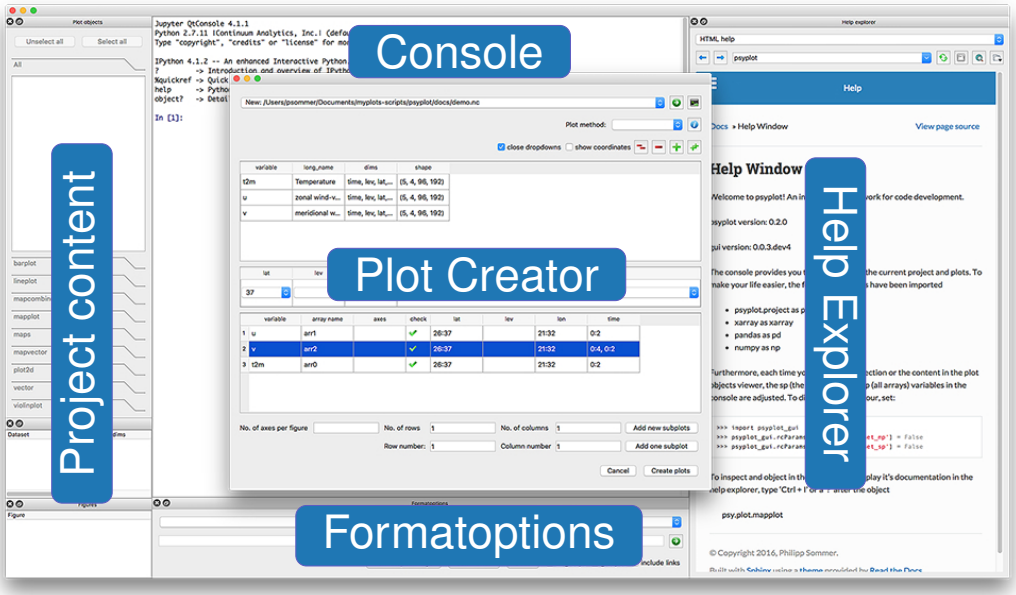
The Psyplot Framework

UNIL | Université de Lausanne

1. The basis forms a variable in a (netCDF) dataset (e.g. temperature).
2. It is plotted by a plotter to your choice (e.g. using `psy.plot.mapplot`.
3. Multiple data arrays (and plotters) form a project.
4. Some plotters also visualize multiple data arrays at once (e.g. line plots), that are then concatenated as an `InteractiveList`.
5. The sub project represents the selected data and plots. The main project represents all.



The Graphical User Interface (GUI)



- Console: An IPython console
- Help Explorer: A browser to display help and browse in the internet
- Plot Creator: A widget to create new plots and open datasets
- Project content: A widget to interact with the psyplot project
- Formatoptions widget: A widget to update and change formatoptions

⇐Framework

Console⇒

The IPython console

UNIL | Université de Lausanne

The central widget in the GUI is an in-process IPython console that provides the possibility to communicate with the psyplot package via the command line and to load any other modules or to run any other script.
It is based on the qtconsole and is connected to the help explorer. If you type, for example, `np.sum(` or `np.sum?` it will show you the documentation of the numpy.sum module in the help explorer.
This feature is motivated from the Spyder editor.

```
Jupyter QtConsole 4.3.1
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 12:04:33)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: ds = psy.open_dataset('demo.nc')

In [2]: ds
Out[2]:
<xarray.Dataset>
Dimensions:  (lat: 96, lev: 4, lon: 192, time: 5)
Coordinates:
  * lon      (lon) float64 0.0 1.875 3.75 5.625 7.5 9.375 11.25 13.1
  * lat      (lat) float64 88.57 86.72 84.86 83.0 81.13 79.27 77.41
  * lev      (lev) float64 1e+05 8.5e+04 5e+04 2e+04
  * time     (time) datetime64[ns] 1979-01-31T18:00:00 1979-02-28T18
Data variables:
    t2m      (time, lev, lat, lon) float32 ...
    u        (time, lev, lat, lon) float32 ...
    v        (time, lev, lat, lon) float32 ...
Attributes:
    CDI:          Climate Data Interface version 1.6.8 (http://mpime
    Conventions:  CF-1.4
    history:      Mon Aug 17 22:51:40 2015: cdo -r copy test-t2m-u-v
    title:        Test file
    CDO:          Climate Data Operators version 1.6.8rc2 (http://mp

In [3]: ds.psy.plot.mapplot(name='t2m', cmap='Reds')
Out[3]: psyplot.project.Project([    arr0: 2-dim DataArray of t2m, v
lev=100000.0, time=1979-01-31T18:00:00])

In [4]:
```

1/2

Back to overview

⇐GUI

Help explorer⇒

The IPython console

UNIL | Université de Lausanne

Furthermore, this widget is connected to the current psyplot project (see `psyplot.project.scp` and `psyplot.project.gcp`). They can be access via

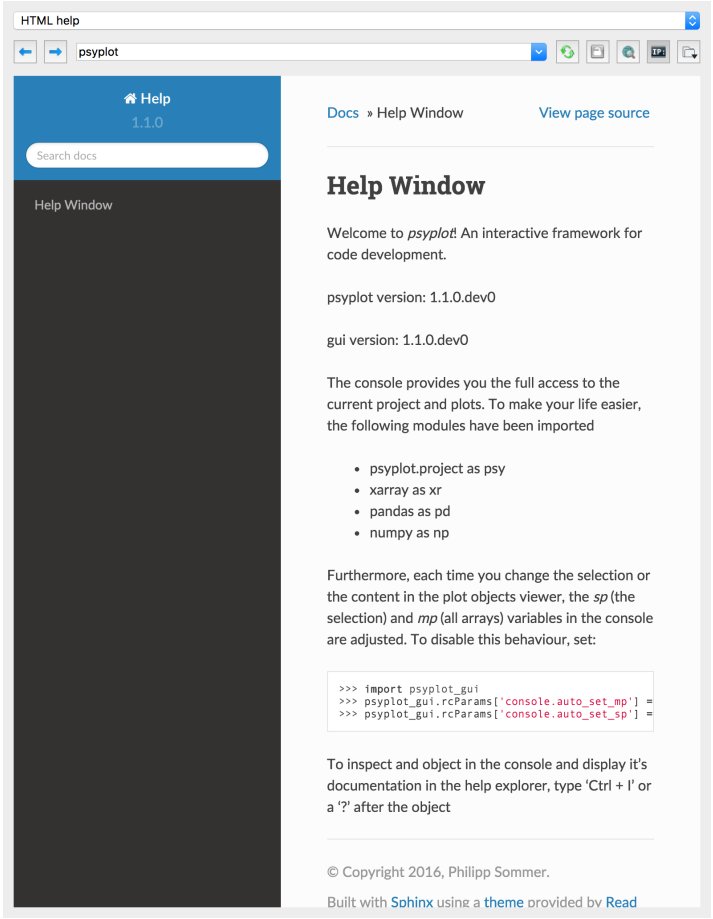
- **sp** This variable links to the current subproject (`psy.gcp()`)
- **mp** This variable links to the current main project (`psy.gcp(True)`)

```
Jupyter QtConsole 4.3.1
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 12:04:33)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: ds = psy.open_dataset('demo.nc')

In [2]: ds
Out[2]:
<xarray.Dataset>
Dimensions:  (lat: 96, lev: 4, lon: 192, time: 5)
Coordinates:
  * lon      (lon) float64 0.0 1.875 3.75 5.625 7.5 9.375 11.25 13.1
  * lat      (lat) float64 88.57 86.72 84.86 83.0 81.13 79.27 77.41
  * lev      (lev) float64 1e+05 8.5e+04 5e+04 2e+04
  * time     (time) datetime64[ns] 1979-01-31T18:00:00 1979-02-28T18
Data variables:
    t2m      (time, lev, lat, lon) float32 ...
    u        (time, lev, lat, lon) float32 ...
    v        (time, lev, lat, lon) float32 ...
Attributes:
    CDI:          Climate Data Interface version 1.6.8 (http://mpime
    Conventions:  CF-1.4
    history:      Mon Aug 17 22:51:40 2015: cdo -r copy test-t2m-u-v
    title:        Test file
    CDO:          Climate Data Operators version 1.6.8rc2 (http://mp

In [3]: ds.psy.plot.mapplot(name='t2m', cmap='Reds')
Out[3]: psyplot.project.Project([    arr0: 2-dim DataArray of t2m, v
lev=100000.0, time=1979-01-31T18:00:00])

In [4]:
```

Interactive Help and Documentation

UNIL | Université de Lausanne

The help explorer provides you access to python object documentations, online information and other help. Plus, it can be used as a webbrowser.
This widget is motivated by the Help of the Scientific PYthon Development EnviRonment (Spyder) editor and uses Sphinx to automatically render python documentation written in restructured Text.
The explorer is also connected to the information functions of psyplot.
`psy.plot.lineplot.keys()`, for example, would be converted to HTML and shown in the help explorer.



Back to overview

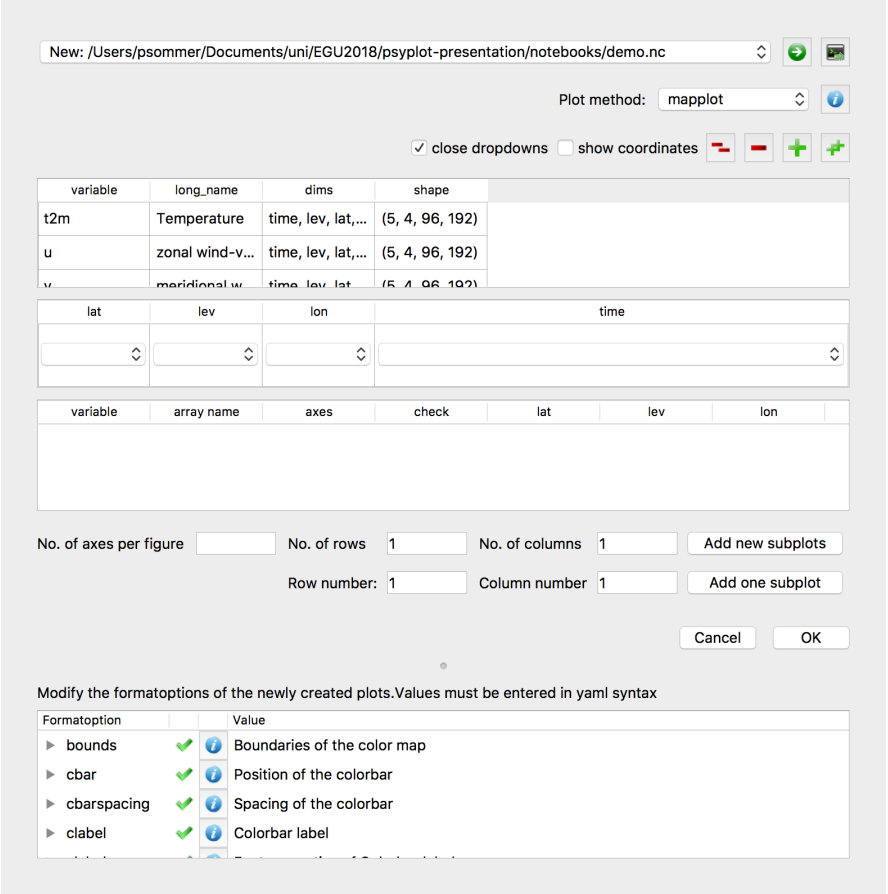
⇐Console

Plotting⇒

Creating new plots

UNIL | Université de Lausanne

The plot creator is used to create new plots from a `xarray.Dataset`.
It can load an in-memory dataset from the console or open a dataset from a file.
It can be used to simply access the data, or by setting up a new plot using the various available plot methods.



Back to overview

⇐Help explorer

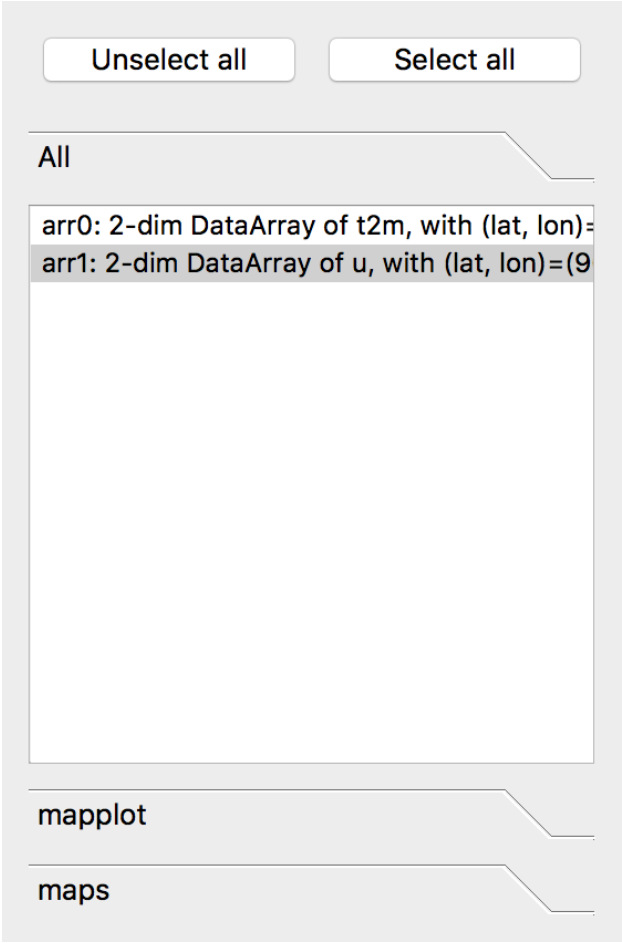
Project Content⇒

The Project Content Widget

UNIL | Université de Lausanne

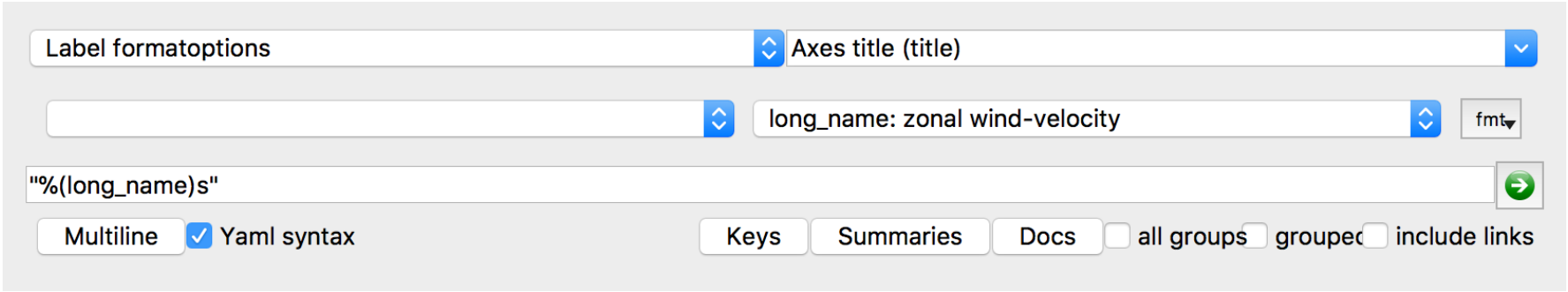
The project content shows you the current project (see `psyplot.project.gcp`). The selected arrays are the current subproject. With this widget, in conjunction with the formatoptions widget, you can quickly change and update the current project.

Back to overview



Updating formatoptions in the GUI

UNIL | Université de Lausanne



The formatoption widget can be used to update the formatoptions of the current subproject or to show their help.
It is build up by one drop down list for the Formatoption group (here *Label formatoptions*) and the formatoption (here *title*). Additionally it contains a line editor (second last row) that shows you the current value and let's you modify it.
Between this editor and the dropdown menus, formatoptions can show different utilities to help you formatting your plots (i.e. the current subproject). In the case of text labels (e.g. title, xlabel, ylabel, etc.), this means the netCDF attributes of the variables or coordinates.

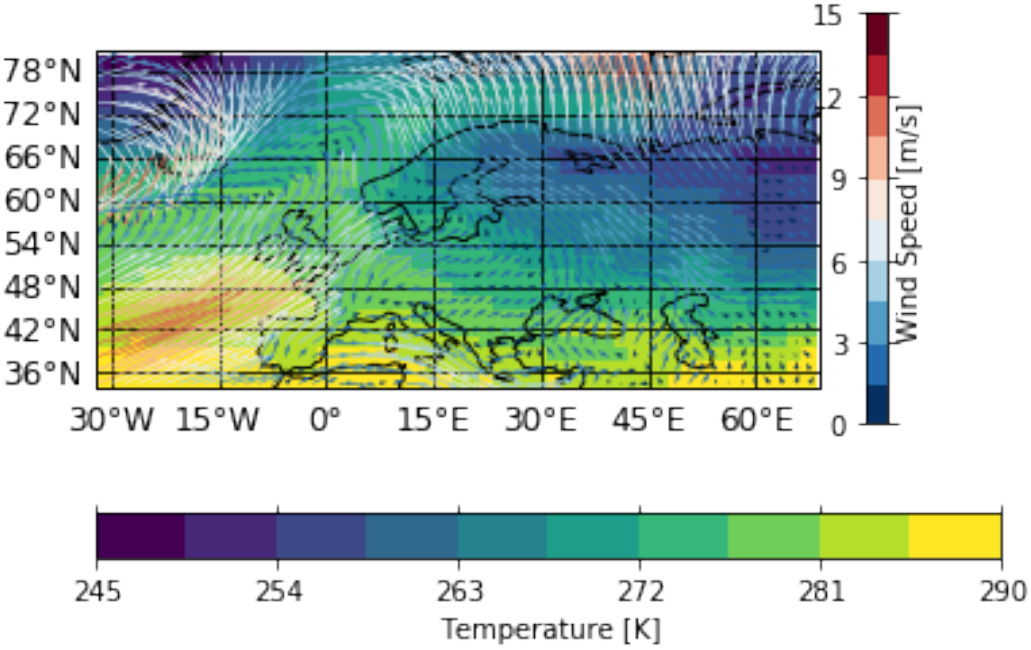
Back to overview

⇐Project Content

Map example⇒

Visualization on a map

UNIL | Université de Lausanne



This example demonstrates the plotting of geo-referenced data. It uses the psy-maps plugin and a dataset with temperature, zonal and meridional wind direction.

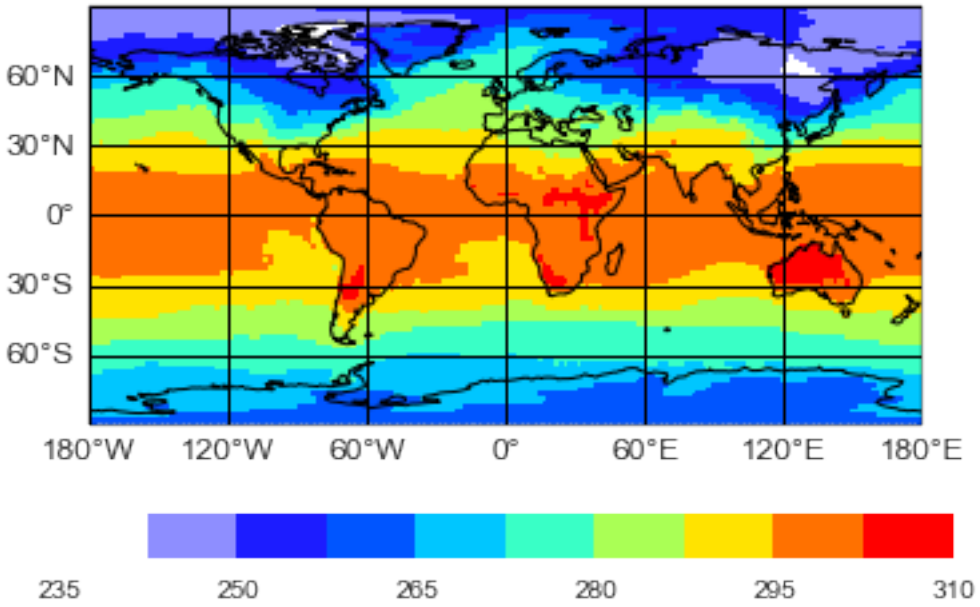
⇐Fmt widget

Scalar data⇒

Visualizing scalar fields I

The *mapplot* method visualizes scalar data on a map.

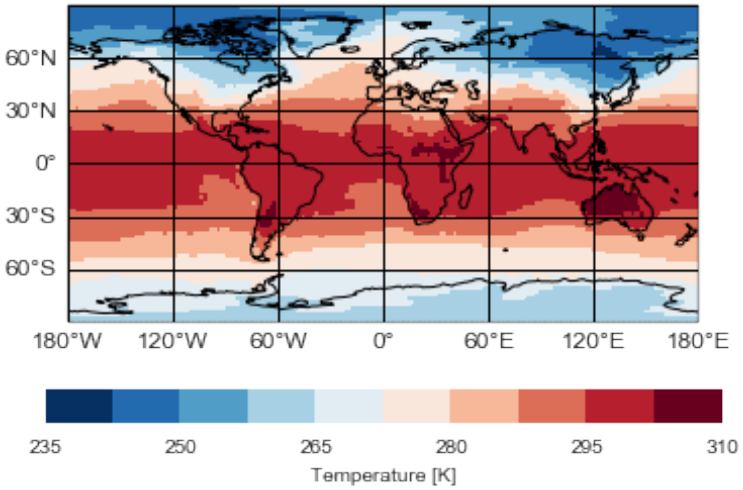
```
In [2]: maps = psy.plot.mapplot('demo.nc', name='t2m')
```



Visualizing scalar fields II

To show the colorbar label we can use the *clabel* formatoption keyword and use one of the predefined labels. Furthermore we can use the *cmap* formatoption to see one of the many available colormaps

```
In [3]: maps.update(clabel='{desc}', cmap='RdBu_r')
```



Formatoptions

Formatoptions are used to update the plot. For maps, especially useful formatoptions are

- projection: To modify the projection on which we draw +
- lonlatbox: To select only a specific slice +
- xgrid and ygrid: to disable, enable or modify the latitude-longitude grid

There are many more formatoption keys that you can explore in the GUI, the online-documentation or via

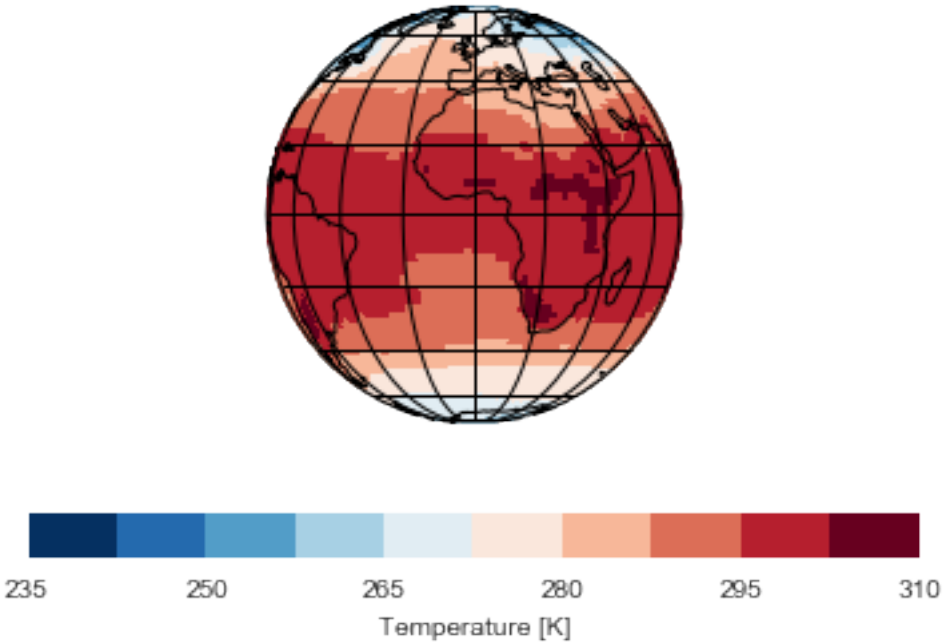
```
In [6]: psy.plot.mapplot.keys(grouped=True)
**************************
Color coding formatoptions
**************************
+------------+------------+------------+------------+
| bounds     | cbar       | cbarspacing | cmap       |
+------------+------------+------------+------------+
| ctickprops | cticksize  | ctickweight | extend     |
+------------+------------+------------+------------+
| miss_color |            |            |            |
+------------+------------+------------+------------+
...
```

Updating the projection

UNIL | Université de Lausanne

To use an orthogonal projection, we change the projection keyword to

```
In [4]: maps.update(projection='ortho')
```

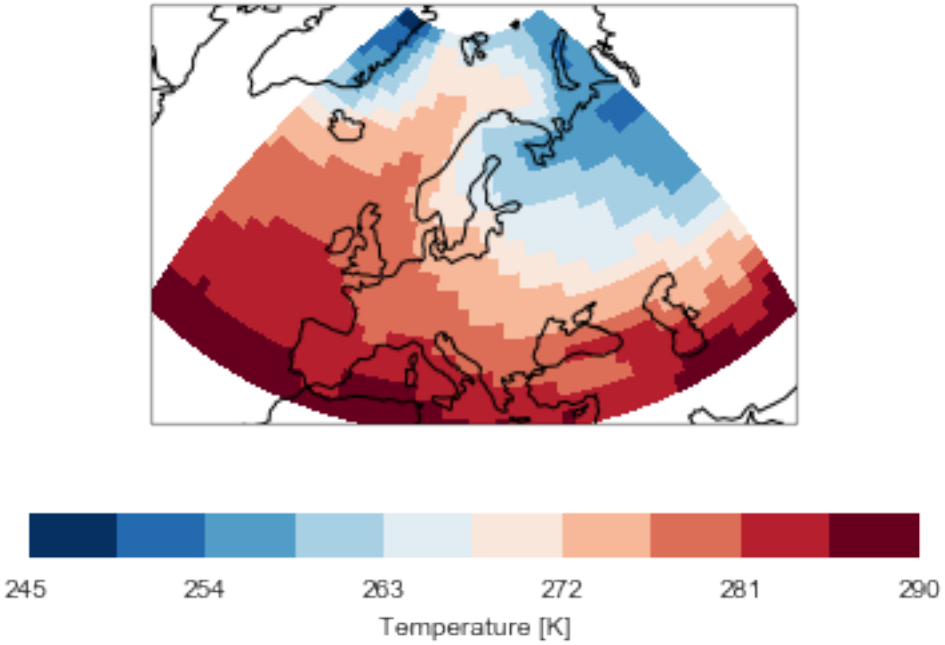


Choosing the region

UNIL | Université de Lausanne

To focus on Europe and disable the latitude-longitude grid, we can set

```
In [5]: maps.update(lonlatbox='Europe', xgrid=False, ygrid=False)
```

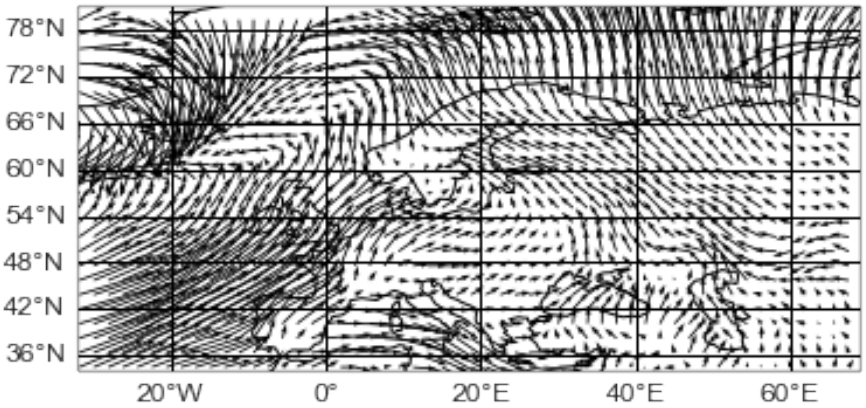


Visualizing vector data I

UNIL | Université de Lausanne

The *mapvector* method can visualize vectorized data on a map. It requires the the wind component in x- (here `'u'`) and y-direction (here `'v'`).

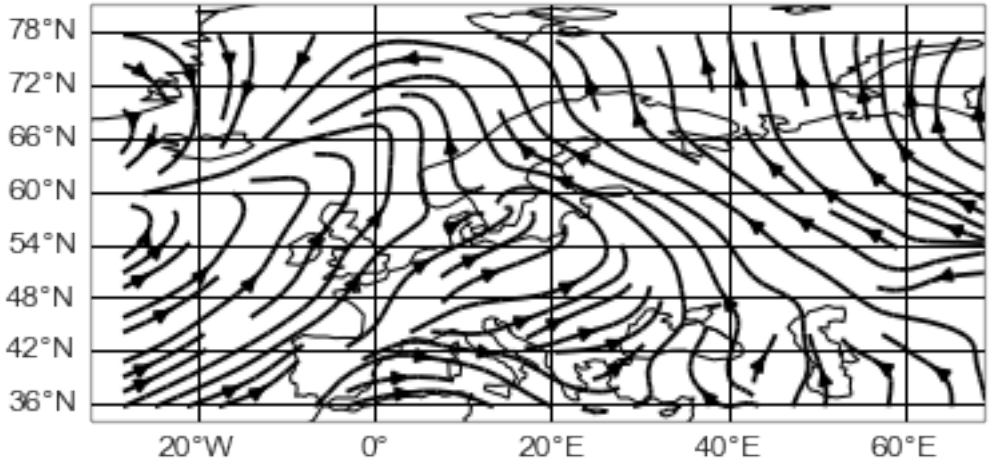
```
In [8]: mapvectors = psy.plot.mapvector(
            'demo.nc', name=[['u', 'v']], lonlatbox='Europe',
            arrowsize=100)
```



Visualizing vector data II

The plotter supports all formatoptions that the *mapplot* method supports (lonlatbox, projection, etc.). The *plot* formatoption furthermore supplies the `'stream'` value in order to make a streamplot

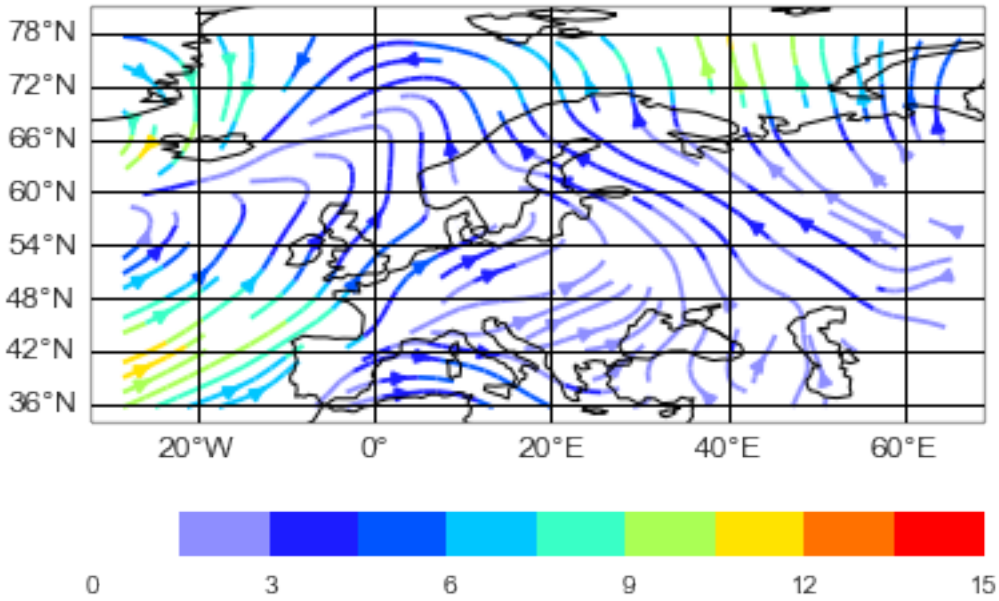
```
In [9]: mapvectors.update(plot='stream', arrowsize=None)
```



Visualizing vector data III

and we have two possibities to visualize the strength of the wind, either via the color coding

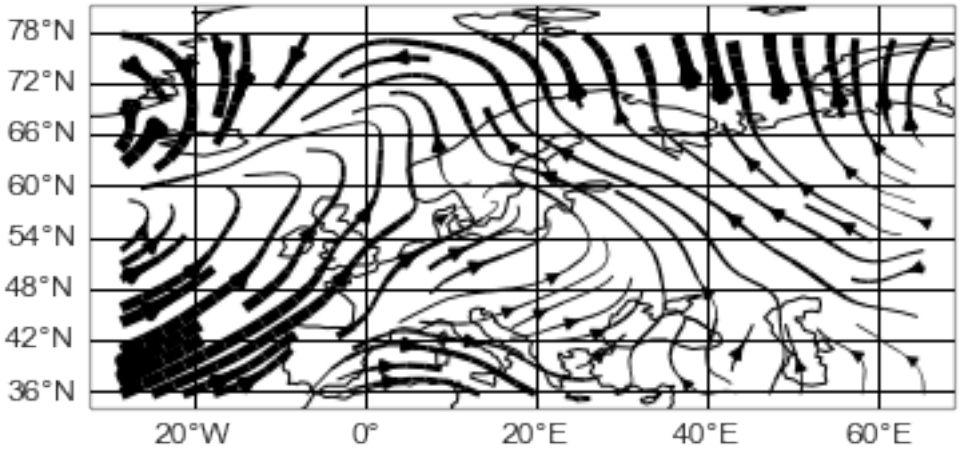
```
In [10]: mapvectors.update(color='absolute')
```



Visualizing vector data IV

or via the linewidth

```
In [11]: mapvectors.update(
             color='k', linewidth=['absolute', 0.5])
```



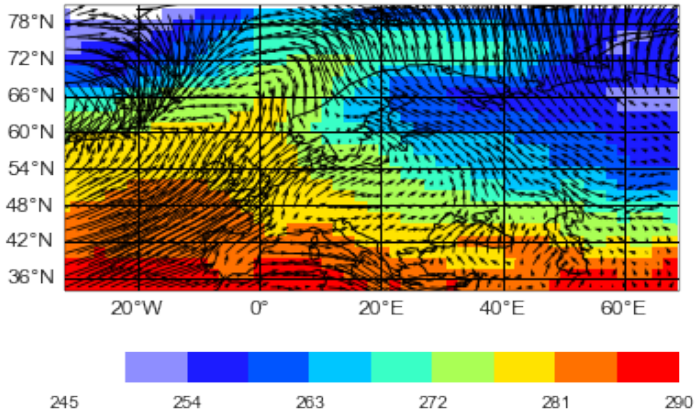
where 0.5 is a scaling factor.

Visualizing combined scalar and vector I

UNIL | Université de Lausanne

The *mapcombined* method can visualize a scalar field (here temperature) with overlayed vector field. This method needs 3 variables: one for the scalar field and two for the wind fields.

```
In [13]: maps = psy.plot.mapcombined(
             'demo.nc', name=[['t2m', ['u', 'v']]],
             lonlatbox='Europe', arrowsize=100)
```



Visualizing combined scalar and vector II

UNIL | Université de Lausanne

We can also modify the color coding etc. here, but all the formatoptions that affect the vector color coding start with `'v'`

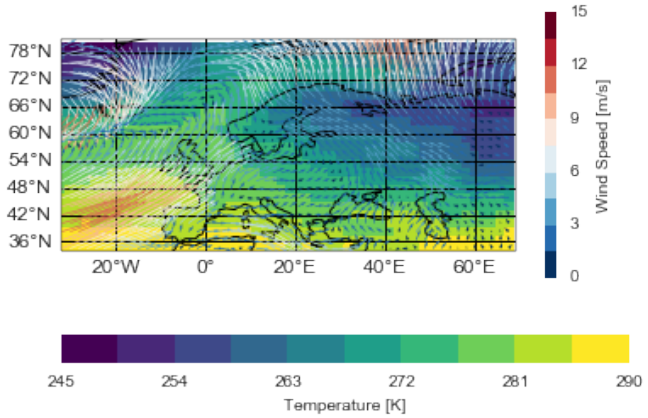
```
In [14]: psy.plot.mapcombined.keys('colors')
```

```
+-------------+-------------+-------------+-------------+
| vcmap       | vcticksize  | color       | vcbar       |
+-------------+-------------+-------------+-------------+
| vctickprops | vctickweight | vbounds     | vcbarspacing |
+-------------+-------------+-------------+-------------+
| bounds      | cbar        | miss_color  | extend      |
+-------------+-------------+-------------+-------------+
| ctickweight | ctickprops  | cbarspacing | cmap        |
+-------------+-------------+-------------+-------------+
| cticksize   |             |             |             |
+-------------+-------------+-------------+-------------+
```

Visualizing combined scalar and vector III

For example, let's modify the wind vector plots color coding and place a colorbar on the right side

```
In [15]: maps.update(
             color='absolute', cmap='viridis', vcmap='RdBu_r',
             vcbar='r', clabel='{desc}',
             vclabel='Wind Speed [%(units)s]')
```



Bar plot demo I

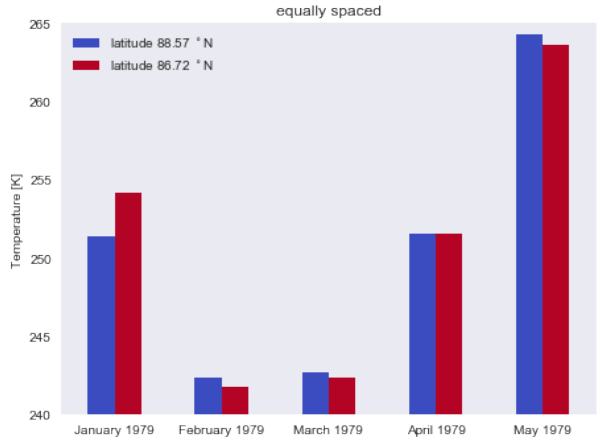
UNIL | Université de Lausanne

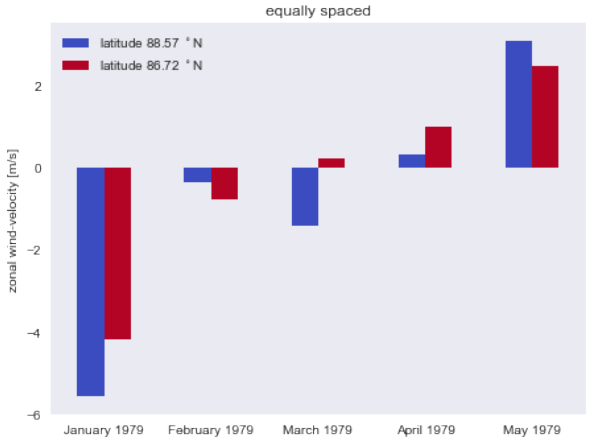
This example shows you how to make a bar plot using the `psyplot.project.ProjectPlotter.barplot` method.

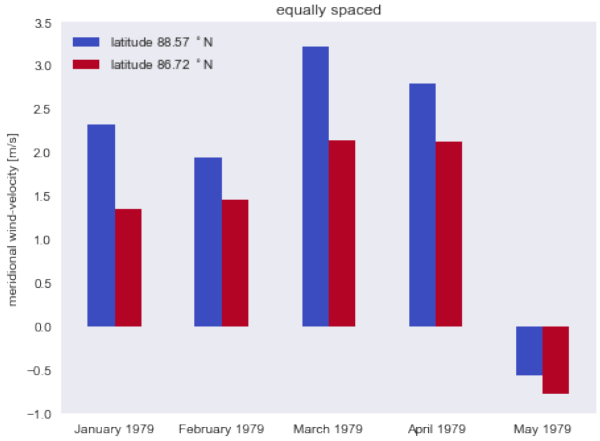
```
In [2]: axes = sy.multiple_subplots(2, 2, n=3)
        for var,ax in zip(['t2m', 'u', 'v'], axes):
            psy.plot.barplot(
                'demo.nc',          # netCDF file storing the data
                name=var,           # one plot for each variable
                y=[0, 1],           # two bars in total at different latitudes
                z=0, x=0,           # height (z) and longitude (x) as dimensions
                ax=ax,
                ### Formatoptions
                ylabel="{desc}",  # use the longname and units on the y-axis
                color='coolwarm', xticklabels='%B %Y',
                legendlabels='latitude %(y)1.2f $^\circ$N',
                legend='upper left',
                title='equally spaced')
        bars = psy.gcp(True)
        bars.show()
```

Bar plot demo II

UNIL | Université de Lausanne







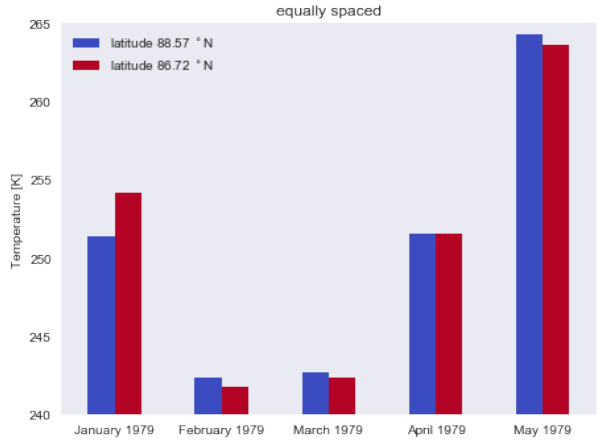
Changing the widths I

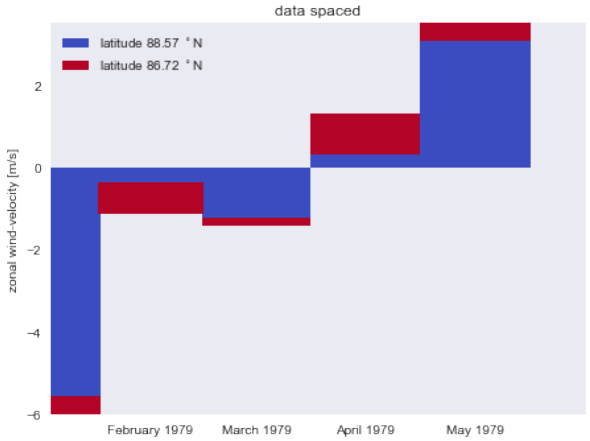
The default is that all bars have the same width. You can however change that by setting the `widths` keyword to `data`

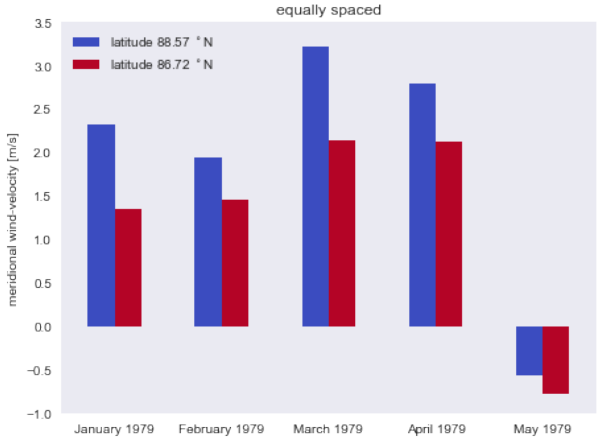
```
In [3]: bars(name='u').update(
            widths='data', xticks='month', title='data spaced')
        bars.show()
```

Changing the widths II

UNIL | Université de Lausanne







Stacked bars I

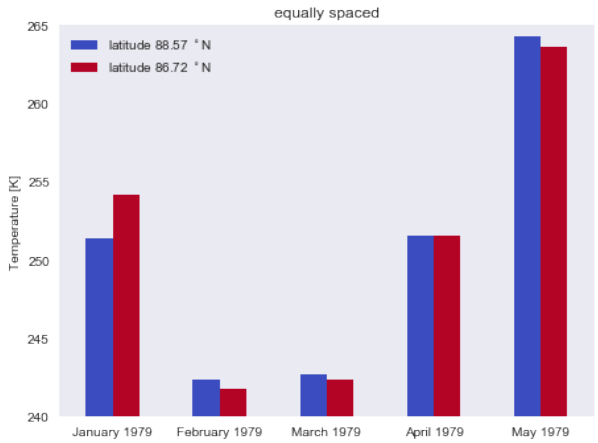
UNIL | Université de Lausanne

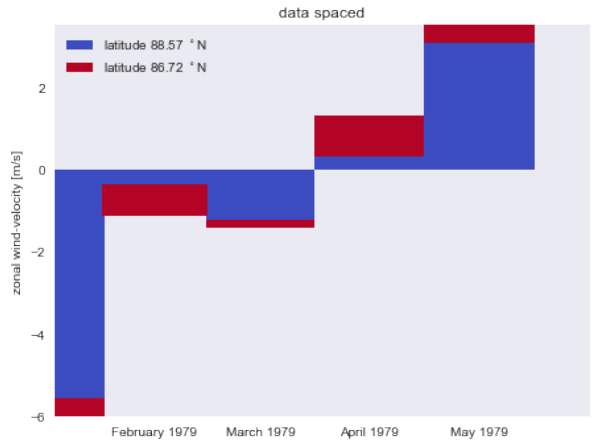
Or you make a stacked plot

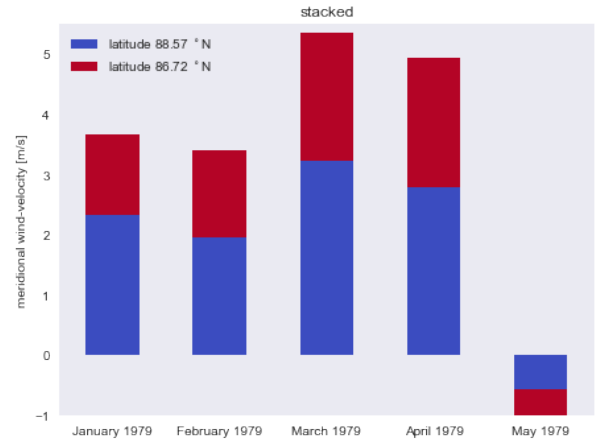
```
In [4]: bars(name='v').update(plot='stacked', title='stacked')
        bars.show()
```

Stacked bars II

UNIL | Université de Lausanne

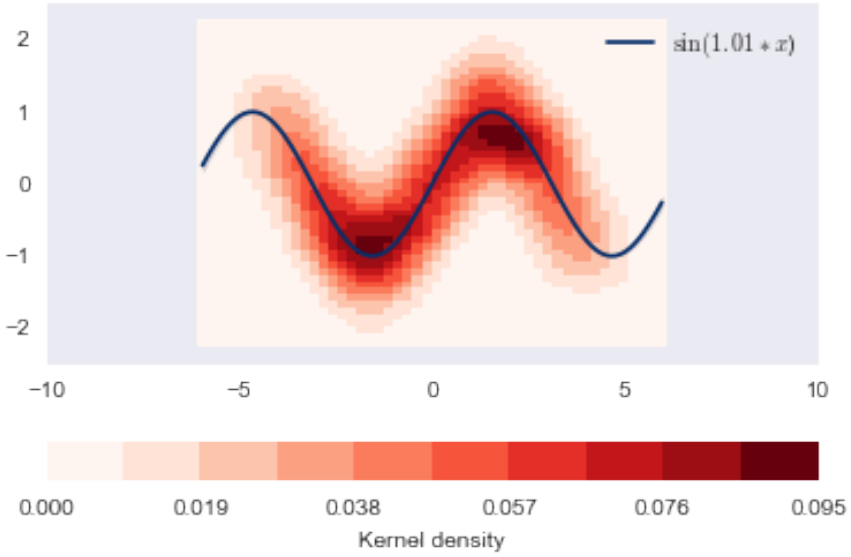






Combining data and regression I

UNIL | Université de Lausanne



This example uses artifical data to show you the capabilities of the `densityreg` plot method. This method combines fits and their raw data to provide an overview on the spread and the regression.

,

⇐Stacked bars

Sample data⇒

Sample data I

UNIL | Université de Lausanne

First we define our data which comes from multiple realizations of the underlying equation `sin(x)`

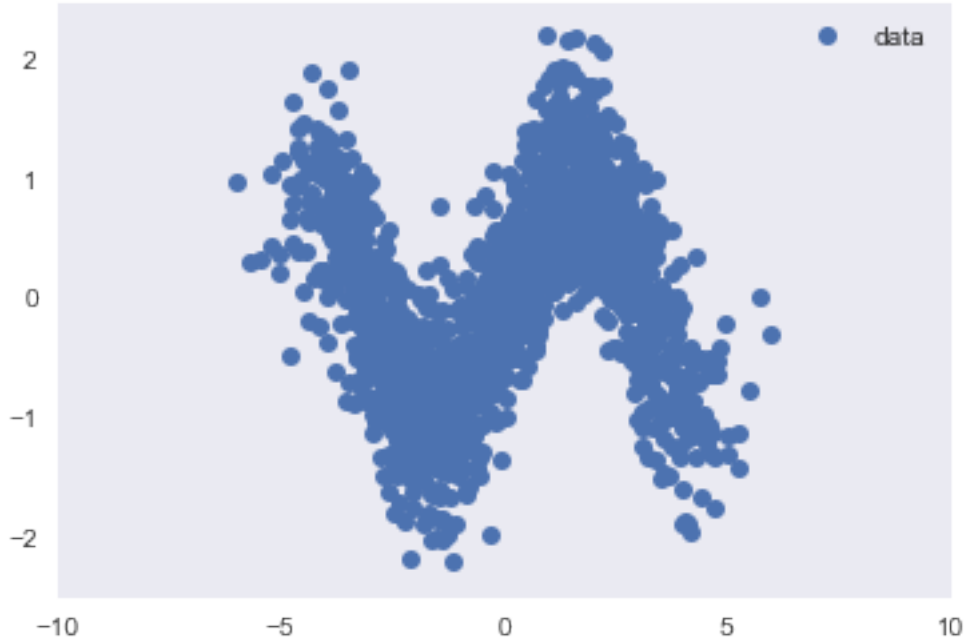
```
In [2]: all_x = []
        all_y = []
        for i in range(30):
            deviation = np.abs(np.random.normal())
            all_x.append(np.linspace(-np.pi - deviation, np.pi + deviation))
            all_y.append(np.sin(all_x[-1]) + np.random.normal(
                scale=0.5, size=all_x[-1].size))
        x = np.concatenate(all_x)
        y = np.concatenate(all_y)
        ds = xr.Dataset({'x': xr.Variable(('experiment', ), x),
                         'y': xr.Variable(('experiment', ), y)})
        ds

Out[2]: <xarray.Dataset>
Dimensions:  (experiment: 1500)
Dimensions without coordinates: experiment
Data variables:
y            (experiment) float64 0.375 -0.474 0.4116 1.099 1.278 0.2544 ...
x            (experiment) float64 -5.01 -4.805 -4.601 -4.396 -4.192 -3.987 ...
```

Sample data II

UNIL | Université de Lausanne

This dataset now contains the two variables `x` and `y`. A scatter plot of the data looks pretty messy

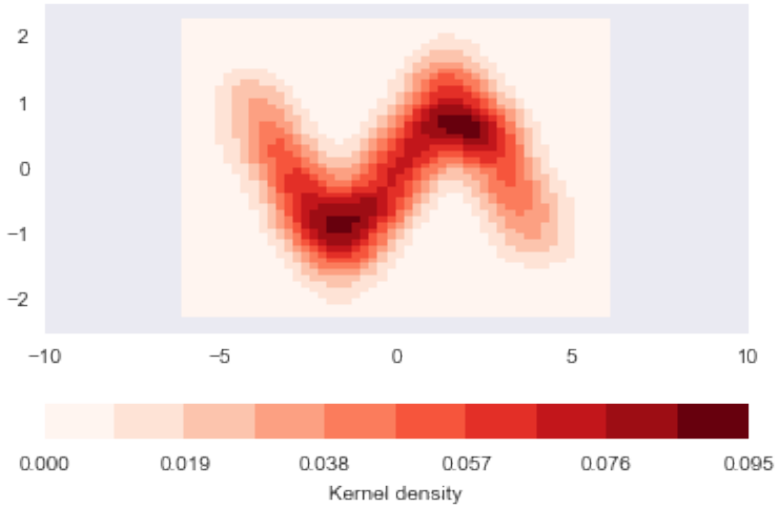
```
In [3]: psy.plot.lineplot(ds, name='y', coord='x', marker='o', li
```



Density plot

Due to the high number of points, a scatter plot is not very informative. Instead, we can use the `density` plot

```
In [4]: psy.plot.density(
            ds, name='y', coord='x', cmap='Reds', bins=50,
            density='kde', clabel='Kernel density')
```

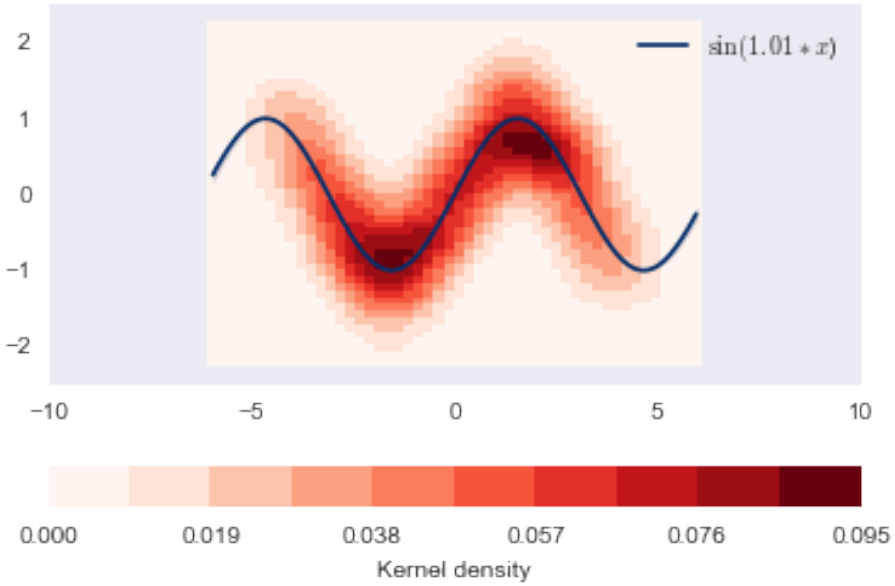


Regression plot

UNIL | Université de Lausanne

The `densityreg` plot method combines this plot with a fit through the data

```
In [5]: psy.close('all')
        psy.plot.densityreg(
            ds, name='y', coord='x', cmap='Reds', bins=50, density='kde',
            clabel='Kernel density', color='Blues_r',
            fit=lambda x, a: np.sin(a * x),
            legendlabels='$\sin (%(a)1.2f * %(xname)s$)')
```



⇐Density plot

Polar data⇒

Visualizing circumpolar data I

UNIL | Université de Lausanne

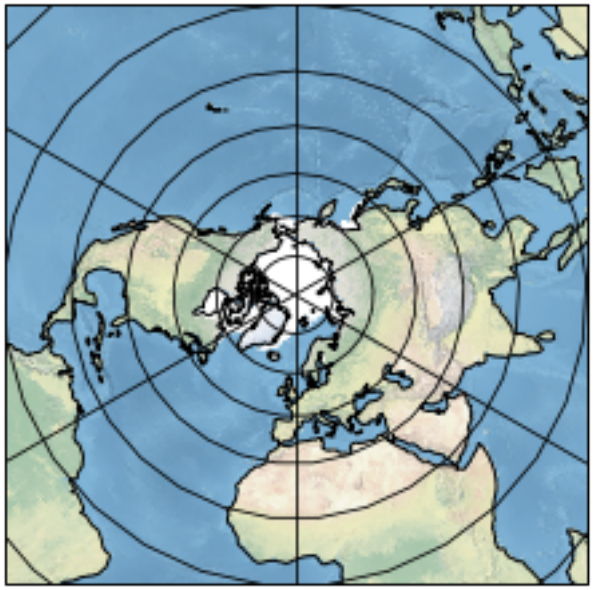
This example uses the psy-maps plugin and data from Walsh et al. (2015) that has been remapped to a circumpolar grid using Climate Data Operators (Max-Planck-Institute for Meteorology, 2018).
Usually, netCDF files contain one-dimensional coordinates, one for the longitude and one for the latitude. Circumpolar grids, however, are defined using 2D coordinates. The visualization using `psyplot` is however straight forward.
The file we are plotting here contains a variable for the sea ice concentration (0 - the grid cell contains no ice, 1 - fully ice covered). Therefore we use a colormap that reflects this behaviour. It is white but it's transparency (the alpha value) increases for larger concentration.

Visualizing circumpolar data II

UNIL | Université de Lausanne

First, we use a `'northpole'` projection to display it

```
In [2]: colors = np.ones((100, 4))    # all white
        # increase the alpha values from 0 to 1
        colors[50:, -1] = np.linspace(0, 1, 50)
        colors[:50, -1] = 0
        cmap = mcol.LinearSegmentedColormap.from_list(
            'white', colors, 100)
        sp = psy.plot.mapplot(
            'G10010_SIBT1850_v1.1._2013-01-15_circumpolar.nc',
            projection='northpole', cmap=cmap,
            # mask all values below 0
            maskless=0.0,
            # do not show the colorbar
            cbar=False,
            # plot a Natural Earth shaded relief raster
            stock_img=True)
```

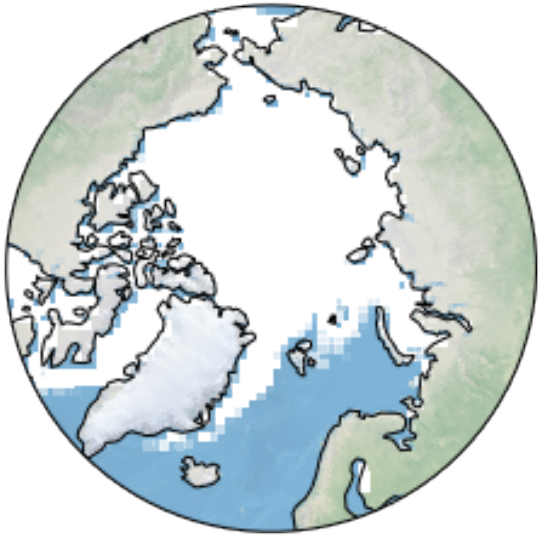


Zoom in

UNIL | Université de Lausanne

The previous plot did show the entire northern hemisphere. We are however only interested in the arctic, so we adapt our lonlatbox

```
In [3]: sp.update(
            # lonmin, lonmax, latmin, latmax
            lonlatbox=[-180, 180, 60, 90],
            xgrid=False, ygrid=False)  # disable the grid
```



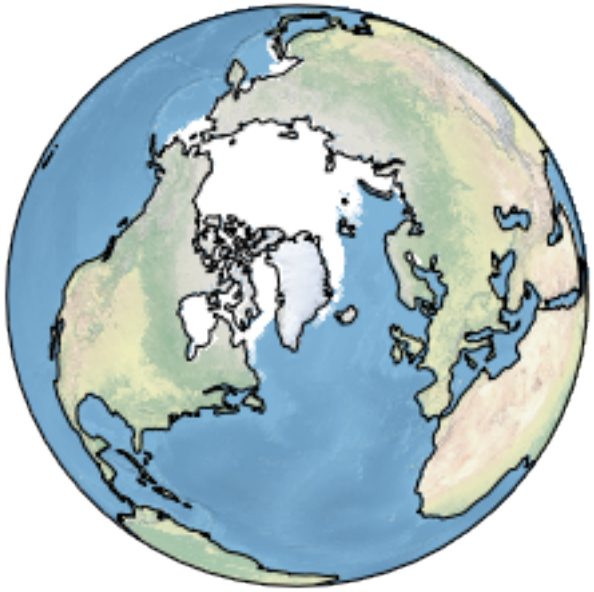
⇐Polar data

Setting the focus⇒

Setting the focus

We can also use the `clon` and `clat` formatoptions to focus on Greenland. Here, we might also want to change the projection since the *northpole* projection implies `clat=0`

```
In [4]: sp.update(clon='Greenland', clat='Greenland',
                  projection='ortho', lonlatbox=None)
```

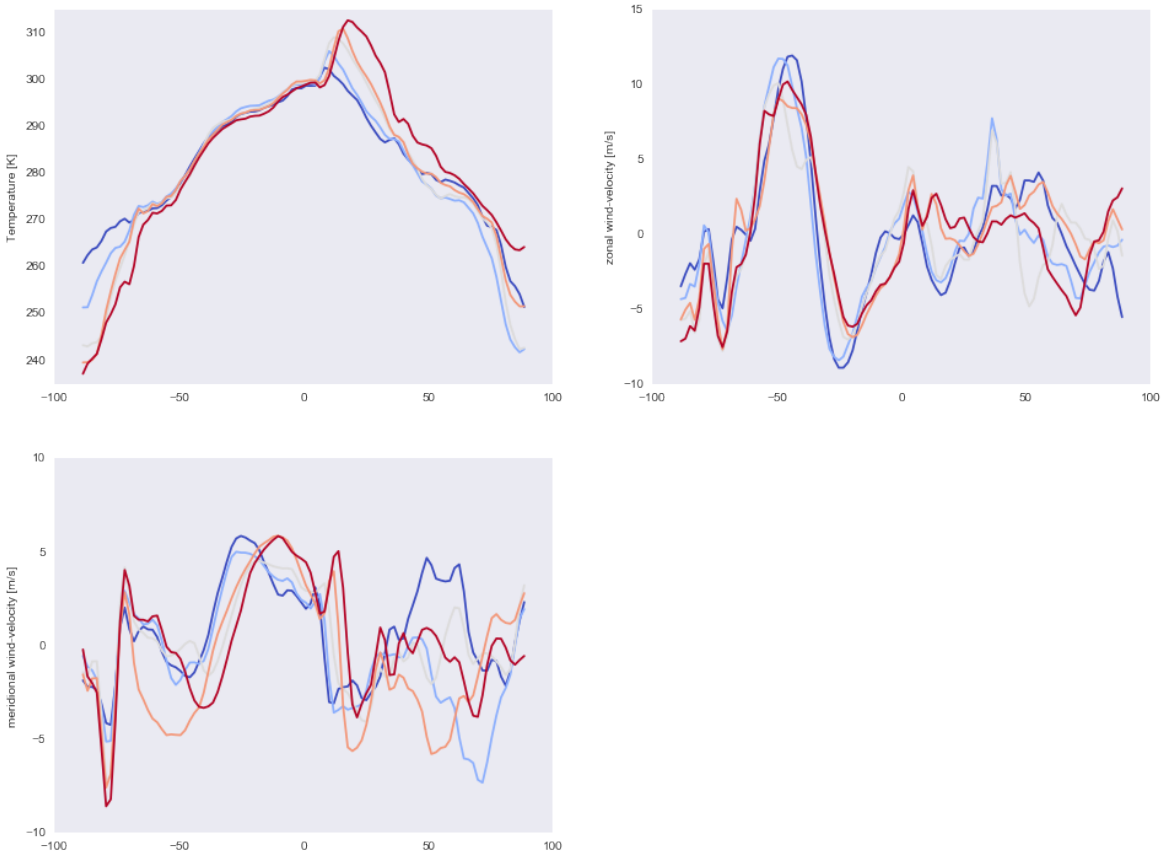


Line plot demo I

This example shows you how to make a line plot using the `psyplot.project.ProjectPlotter.lineplot` method.

```
In [2]: axes = iter(psy.multiple_subplots(2, 2, n=3))
        for var in ['t2m', 'u', 'v']:
            psy.plot.lineplot(
                'demo.nc',  # netCDF file storing the data
                name=var, # one plot for each variable
                t=range(5),  # one violin plot for each time step
                z=0, x=0,       # choose latitude and longitude as dimensions
                ylabel="{desc}",  # use the longname and units on the y-axis
                ax=next(axes),
                color='coolwarm', legend=False )
        lines = psy.gcp(True)
        lines.show()
```

Line plot demo II



References I

S. Hoyer and J. Hamman. xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, 5(1), 2017. doi: 10.5334/jors.148. URL http://doi.org/10.5334/jors.148.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, May 2007. ISSN 1521-9615. doi: 10.1109/MCSE.2007.55.

E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL http://www.scipy.org/. [Online; accessed 2017-02-18].

Max-Planck-Institute for Meteorology. CDO: Climate Data Operators, 2018. URL http://www.mpimet.mpg.de/cdo.

Met Office. *Cartopy: a cartographic python library with a matplotlib interface*. Exeter, Devon, 2010 - 2015. URL http://scitools.org.uk/cartopy.

P. S. Sommer. The psyplot interactive visualization framework. *The Journal of Open Source Software*, 2(16), aug 2017a. doi: 10.21105/joss.00363. URL https://doi.org/10.21105/joss.00363.

P. S. Sommer. Chilipp/psyplot-gui: v1.0.1: Graphical user interface for the psyplot package. Aug 2017b. doi: 10.5281/zenodo.845726. URL https://doi.org/10.5281/zenodo.845726.

M. Summerfield. *Rapid GUI Programming with Python and Qt (Prentice Hall Open Source Software Development)*. Prentice Hall, 2007. ISBN 978-0132354189. URL https://www.amazon.com/Programming-Python-Prentice-Software-Development/dp/0132354187?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0132354187.

J. E. Walsh, W. L. Chapman, and F. Fetterer. Gridded monthly sea ice extent and concentration, 1850 onwards, 2015.

,

mapplot formatoptions I

UNIL | Université de Lausanne

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			

mapplot formatoptions II

UNIL | Université de Lausanne

Miscallaneous formatoptions

clat	clip	clon	datagrid
grid_color	grid_labels	grid_labelsize	grid_settings
interp_bounds	lonlatbox	lsm	map_extent
projection	stock_img	transform	xgrid
ygrid			

Axis tick formatoptions

cticklabels	cticks

Masking formatoptions

maskbetween	maskgeq	maskgreater	maskleq
maskless			

mapplot formatoptions III

UNIL | Université de Lausanne

Plot formatoptions

plot

Post processing formatoptions

post | post_timing

Axes formatoptions

tight

Formatoption summaries I

UNIL | Université de Lausanne

Color coding formatoptions

Boundaries of the color map (bounds) Specify the boundaries of the colorbar

Position of the colorbar (cbar) Specify the position of the colorbars

Spacing of the colorbar (cbarspacing) Specify the spacing of the bounds in the colorbar

Colormap (cmap) Specify the color map

Font properties of the colorbar ticklabels (ctickprops) Specify the font properties of the colorbar ticklabels

Font size of the colorbar ticklabels (cticksize) Specify the font size of the colorbar ticklabels

Formatoption summaries II

Font weight of the colorbar ticklabels (ctickweight) Specify the fontweight of the colorbar ticklabels

Ends of the colorbar (extend) Draw arrows at the side of the colorbar

Levels for the filled contour plot (levels) The levels for the contour plot

Color of missing values (miss_color) Set the color for missing values

Label formatoptions

Colorbar label (clabel) Show the colorbar label

Font properties of Colorbar label (clabelprops) Properties of the Colorbar label

Font size of Colorbar label (clabelsize) Set the size of the Colorbar label

Formatoption summaries III

Font weight of Colorbar label (clabelweight) Set the fontweight of the Colorbar label

Figure title (figtitle) Plot a figure title

Font properties of Figure title (figtitleprops) Properties of the figure title

Font size of Figure title (figtitlesize) Set the size of the figure title

Font weight of Figure title (figtitleweight) Set the fontweight of the figure title

Arbitrary text on the plot (text) Add text anywhere on the plot

Axes title (title) Show the title

Font properties of Axes title (titleprops) Properties of the title

Font size of Axes title (titlesize) Set the size of the title

Formatoption summaries IV

Font weight of Axes title (titleweight) Set the fontweight of the title

Miscallaneous formatoptions

Latitude of the center of the plot (clat) Set the center latitude of the plot

clip Clip the part outside the latitudes of the map extent

Longitude of the center of the plot (clon) Set the center longitude of the plot

Grid of the data (datagrid) Show the grid of the data

Color of the latitude-longitude grid (grid_color) Set the color of the grid

Labels of the latitude-longitude grid (grid_labels) Display the labels of the grid

Formatoption summaries V

Label size of the latitude-longitude grid (grid_labelsize) Modify the size of the grid tick labels

Line properties of the latitude-longitude grid (grid_settings) Modify the settings of the grid explicitly

interp_bounds Interpolate grid cell boundaries for 2D plots

Longitude-Latitude box of the data (lonlatbox) Set the longitude-latitude box of the data shown

Land-Sea mask (lsm) Draw the continents

Longitude-Latitude box of the plot (map_extent) Set the extent of the map

Projection of the plot (projection) Specify the projection for the plot

Formatoption summaries VI

Display Natural Earth shaded relief raster (stock_img) Display a stock image on the map

Coordinate system of the data (transform) Specify the coordinate system of the data

Meridians (xgrid) Draw vertical grid lines (meridians)

Parallels (ygrid) Draw horizontal grid lines (parallels)

Axis tick formatoptions

Colorbar ticklabels (cticklabels) Specify the colorbar ticklabels

Colorbar ticks (cticks) Specify the tick locations of the colorbar

Formatoption summaries VII

UNIL | Université de Lausanne

Masking formatoptions

Mask between two values (maskbetween) Mask data points between two numbers

Mask greater than or equal (maskgeq) Mask data points greater than or equal to a number

Mask greater (maskgreater) Mask data points greater than a number

Mask lesser than or equal (maskleq) Mask data points smaller than or equal to a number

Mask less (maskless) Mask data points smaller than a number

Plot formatoptions

Formatoption summaries VIII

2D plot type (plot) Choose how to visualize a 2-dimensional scalar data field

Post processing formatoptions

Custom post processing script (post) Apply your own postprocessing script

Timing of the post processing (post_timing) Determine when to run the :attr:‘post‘ formatoption

Axes formatoptions

Tight layout (tight) Automatically adjust the plots.

Géopolis, Lausanne, Switzerland

UNIL | Université de Lausanne

Back to author page

Institute of Surface Dynamics (IDYST) - University of Lausanne (UNIL)
Bâtiment Géopolis - CH - 1015 Lausanne

⇐Summaries