

UNIVERSITÉ DE LAUSANNE

DOCTORAL THESIS

---

# Software and Numerical Tools for Paleoclimate Analysis

---

*Author:*

Philipp S. SOMMER

*Supervisor:*

Dr. Basil A. S. Davis



UNIL | Université de Lausanne

*A thesis submitted in fulfillment of the requirements  
for the degree of Doctor of Science*

*in the*

Davis Group  
Institute of Earth Surface Dynamics (IDYST)  
Faculty of Geosciences and Environment (GSE)

*Jury members:*

Prof. Dr. Christian Kull (*president*), Prof. Dr. Grégoire Mariéthoz (*director*),  
Basil A. S. Davis (*co-director*), Prof. Dr. John Robert Tipton (*examiner*),  
Prof. Dr. Christoph Cornelius Raible (*examiner*)

March 9, 2020

*Please cite the original publication by Sommer, 2017*

Sommer, Philipp S. (Aug. 2017). "The psyplot interactive visualization framework". In: *The Journal of Open Source Software* 2.16. DOI: [10 . 21105/joss.00363](https://doi.org/10.21105/joss.00363). URL: <https://doi.org/10.21105/joss.00363>.

*when citing this chapter as Sommer, 2020*

Sommer, Philipp S. (2020). "Psyplot - A flexible framework for interactive data analysis". In: *Software and Numerical Tools for Paleoclimate Analysis*. Ed. by Philipp S. Sommer. University of Lausanne. Chap. 4, pp. 63–79. DOI: [10.5281/zenodo.3757356](https://doi.org/10.5281/zenodo.3757356).

## Chapter 4

# Psyplot

### *A flexible framework for interactive data analysis*

## 4.1 Summary

*From*

Sommer, Philipp S. (Aug. 2017e). “The psyplot interactive visualization framework”. In: *The Journal of Open Source Software* 2.16. DOI: [10 . 21105 / joss . 00363](https://doi.org/10.21105/joss.00363). URL: [https : / / doi . org / 10 . 21105 / joss . 00363](https://doi.org/10.21105/joss.00363).

psyplot (Sommer, 2017e) is a cross-platform open source python project that mainly combines the plotting utilities of matplotlib (Hunter, 2007) and the data management of the xarray (Hoyer and Hamman, 2017) package and integrates them into a software that can be used via command-line and via a graphical user interface (GUI).

The main purpose is to have a framework that allows a fast, attractive, flexible, easily applicable, easily reproducible and especially an interactive visualization of data.

The ultimate goal is to help scientists in their daily work by providing a flexible visualization tool that can be enhanced by their own visualization scripts.

The framework is extended by multiple plugins: psy-simple (Sommer, 2017c) for simple visualization tasks, psy-maps (Sommer, 2017a) for georeferenced data visualization and psy-reg (Sommer, 2017b) for the visualization of fits. It is furthermore extended by the optional graphical user interface psyplot-gui (Sommer, 2017d).

## 4.2 Introduction

The mathematical and statistical processing of climate data is closely related to its visualization and analysis. But in traditional visual analytics literature, these two aspects are commonly treated in separate manners. Keim et al., 2008 for instance, (following Wijk, 2005) distinguish two steps of visual analytics, the initial data processing with statistical or mathematical techniques, and a *sense-making loop* of visualization, exploration and the gain of new knowledge. Böttinger and Röber, 2019 distinguish the *filtering* step (data processing), and *mapping/rendering* step that describes the visualization. Also in the literature there is a clear division between the climate visualization (or visual analytic) papers and the standard statistical or climate literature that describes new methods for data processing. Visualization research focuses mainly on advanced visualization tools such as ParaView (Ayachit,

2015), VAPOR (Clyne et al., 2007) or Avizo<sup>1</sup> (e.g. Böttinger and Röber, 2019; Nocke et al., 2015; Rautenhaus et al., 2018; Wong et al., 2014) whereas statistical or climate literature commonly uses R (R Core Team, 2019), Python (Oliphant, 2006; Perez et al., 2011), Climate Data Operators (CDOs) (Schulzweida, 2019) or other command-line tools.

This separation, however, devalues the interplay between the new knowledge from the visualization step, that commonly raises the need for more statistical and mathematical processing of the initial data. This calls for integrated and flexible tools that tackle both steps: the data processing and the visualization, a requirement that is currently not fulfilled by the visualization tools described above. An example software that integrates data processing and data visualization is provided with the Earth System Model Evaluation Tool (ESMValTool) (Eyring et al., 2016). This framework provides common diagnostics for Earth System Models (ESMs) to enable model intercomparisons. The tool, however, has limited interactivity and a slow learning curve for the implementation of new diagnostics.

This lack leads to large efforts of climate scientists to develop scripts for the data processing and visualization. They usually do not follow a systematic framework and as such need to be adapted every time a new project starts which also make them difficult to share with other researchers. The new *psyplot* framework wants to generalize this data processing and visualization by providing a framework that is highly flexible, interoperates with standard computational data processing tools and enables flexible visualizations and adaptations. The software is written in the programming language Python (Perez et al., 2011) and builds upon the visualization package *matplotlib* (Hunter, 2007) and the N-dimensional array processing package *xarray* (Hoyer and Hamman, 2017), that closely interoperates with the numeric packages *numpy* and *scipy* (Jones et al., 2001; Oliphant, 2006) and the parallel computing library *dask* (Dask Development Team, 2016). Due to the flexibility of Python, it can be used from the command-line, a graphical user interface (GUI) (section 4.3.3) or jupyter notebooks<sup>2</sup> (Kluyver et al., 2016). As such, it supports out-of-core computation (i.e. the processing of data too large to fit into memory), a rich set of visualization methods from *matplotlib*, and can be extended to other visualization packages, such as the 3D-visualization framework VTK (Sommer, 2019b).

The next section 4.3 provide an overview of the framework with its data model, plugins and GUI. Sections 4.4 and 4.5 finally discuss further usage and extensions to the software. For more information, usage and implementation examples I also refer to the online documentation <https://psyplot.readthedocs.io>.

## 4.3 The *psyplot* framework

The *psyplot* framework consists of three parts: The core structure that is built upon *xarray* and provides the general infrastructure (section 4.3.1), the plugins that use the plotting functionalities of *matplotlib* (section 4.3.2), and the GUI (section 4.3.3).

### 4.3.1 Data model

#### *Psyplot* and *xarray*

*psyplot* acts as a high-level interface into the packages *xarray* and *matplotlib*. The first one is a recent package for N-dimensional labeled arrays that adopts Unidata's

<sup>1</sup><https://www.fei.com/software/avizo3d/>

<sup>2</sup><https://jupyter.org/>

self-describing Common Data Model on which the network Common Data Form (netCDF) is built (Brown et al., 1993; Hoyer and Hamman, 2017; Rew and Davis, 1990). The package integrates with standard python from the python environment, such as the computing and analysis packages numpy (Oliphant, 2006), scipy (Jones et al., 2001; Oliphant, 2007), pandas (McKinney, 2010) and statsmodels (Seabold and Perktold, 2010), but also offers intuitive interfaces for other packages, such as a package for empirical orthogonal functions (EOFs, Dawson, 2016), CDOs (Müller, 2019), fourier transforms (Uchida et al., 2019) and many more<sup>3</sup>. This large potential for extension distinguishes psyplot from other high-level visualization software, such as ParaView or Vapor, as such python packages can be implemented as a so-called *formatoption* (without hyphen, see below) or used in a pre-processing step.

### Psyplot core structure

The core structure of psyplot consists of five base classes that interact with each other, the visualization objects *Plotter* and its *Formatoptions*, the data objects *DataArray*, an *InteractiveList* of them, and a collection of all of them, the psyplot *Project*. It is schematically visualized in figure 4.1.

The most high-level Application programming interface (API) object is the psyplot project that consists of multiple data objects that are (or are not) visualized. The main purpose is a parallel handling of multiple plots/arrays that may also interact with each other (e.g. through the sharing of *formatoptions*). It mainly spreads update commands to its contained objects, but also serves as a filter for the data objects. Furthermore, one project may be split up into sub projects which then only control a specific part of the main project, e.g. for a specific formatting of only a small part of the data.

The next level is the *DataArray* from the xarray package (or more explicitly, its accessor, the *InteractiveArray*<sup>3</sup>), that holds the data of one (or more) variables (e.g. temperature) and its corresponding coordinates (e.g. time, latitude, longitude, etc.). It may be one or multidimensional depending on the chosen visualization method. psyplot offers several methods to provide the coordinates for the plotting of different grids to make the visualization easier. The software can interpret CF Conventions<sup>4</sup> and UGRID conventions for unstructured grids (Jagers et al., 2018).

Multiple of these arrays can also be grouped together into an *InteractiveList* that shall be visualized by the same plot method (e.g. multiple lines or a scalar field with overlying vector field).

The visualization part in the framework is managed by the *Plotter* class, a collection of multiple *Formatoptions*. Each plotter subclass is designed to visualize the data in a specific manner (e.g. via line plots, violin plots, or map plots) and is completely defined through its *formatoptions*.

*Formatoptions* are the core of the psyplot structure. The standard functionality of a formatoption is to control the visual appearance of one aspect of the plot (e.g. through the colormap, figure title, etc.). It is, however, completely unlimited and can also do data manipulations or calculations. The psy-reg plugin for example (see section 4.3.2) implements a formatoption that performs a regression through the data

<sup>3</sup> several packages related to xarray are listed in the docs at <http://xarray.pydata.org/en/stable/related-projects.html> and psyplots integration (accessors) in particular is shown at <https://psyplot.readthedocs.io/en/latest/accessors.html>.

<sup>4</sup><http://cfconventions.org>

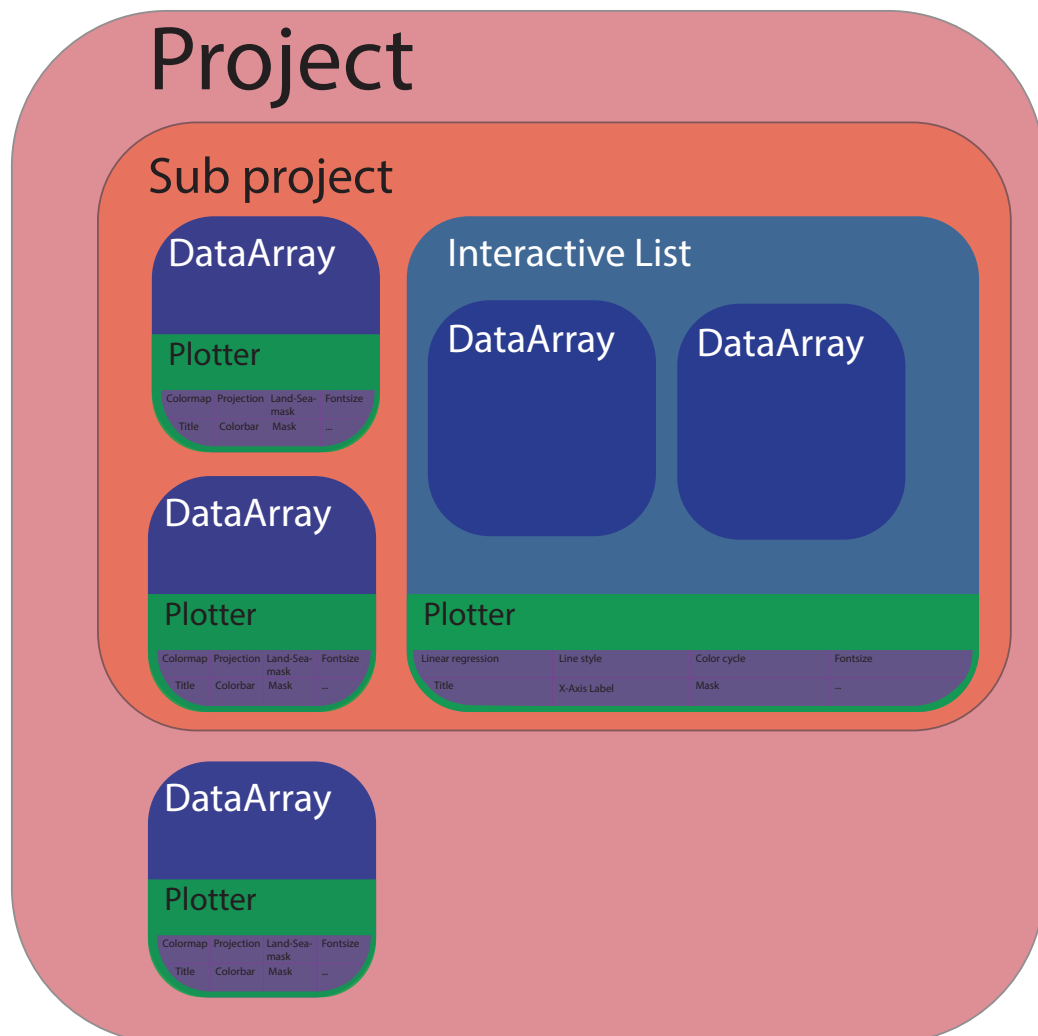


FIGURE 4.1: The psyplot core framework. A (sub) project consists of  $n$ -dimensional data arrays or a list of these that are each visualized by a plotter. Each plotter consists of a set of *formatoptions* that control the appearance of the plot or performs data manipulation.

that is then visualized. As mentioned earlier, each plotter is set up through its *formatoptions* where each *formatoption* has a unique *formatoption* key inside the plotter. This *formatoption* key (e.g. *title* or *cmap*) is what is used for updating the plot, manipulating the data, etc.. *Formatoptions* might also interact with other *formatoptions* inside the plotter or from other plotters. This concept of *formatoptions* allows to use the same *formatoption* with all different kinds of plotters and the interaction of multiple plots with each other. Common plot features, such as the figure title, colormap, etc., therefore do not have to be implemented explicitly for every plotter but can be used from existing implementations. This framework also allows a very easy integration and development of own *formatoptions* with a low or high level of complexity.

### 4.3.2 Psyplot plugins

The psyplot package provides the core of the data management described in the previous section 4.3.1. The real visualization is implemented in external plugins. The advantage of this approach is an increased flexibility of the entire framework (collaborations can evolve through dedicated plugins) and of managing the various dependencies of the packages. As such, the dependencies of psyplot are rather weak (only xarray is needed), but the dependencies of the plugins can be more extensive (e.g. for geo-referencing or advanced statistics).

Each plugin defines new *Plotters* and *Formatoptions* that are specific to the purpose of the visualization/analysis task. The plotters can also be implemented as a plot method (see supplements 4.B to 4.E) and accessed through the psyplot core API (see supplements 4.A for an example).

The current lists of plugins include *psy-simple* for rather simple and standard visualization tasks, *psy-maps* for geo-referenced plots, *psy-reg* for statistical analysis visualization, and *psy-strat* for stratigraphic diagrams.

#### **psy-simple: The psyplot plugin for simple visualizations**

Much of the functionality that is used by other plugins is developed in the psy-simple plugin. This package targets simple visualizations and currently includes plot methods for one-dimensional data: line plots, bar plots and violin plots; for two-dimensional data: scalar plots, vector plots and combined scalar and vector plots; and plots that do not require complex data manipulation: a density plot and a plot of the weighted geographic mean. A full list of examples is provided in the supplementary material, section 4.B.

This package also implements most of the functionality to handle unstructured grids in 2D visualizations and defines most of the commonly used *formatoptions*. The latter include text manipulation (such as plot title, figure title, x- and y-axis labels, etc.), data masking, x- and y-axis tick labeling and positioning, as well as color coding for 2D plots (colormap, colormap sections, etc.).

#### **psy-maps: The psyplot plugin for visualizations on a map**

psy-maps builds on top of the psy-simple plugin and extends its functionality for visualizations on a map using the functionalities of the cartopy package (Met Office, 2010 - 2015) (see supplements 4.C for examples). It simplifies as such the automated generation of maps for climate model data through the flexibility of the psyplot framework.

psy-maps currently implements additional *formatoptions* for choosing the projection of the map, selecting the geographic region, drawing the continents or shaded reliefs of land and ocean, and more. One feature that distinguishes psy-maps from other visualization software, even from pure cartopy, is the ability to visualize unstructured geo-referenced grids on the map. For this purpose, triangles are projected in a pre-processing step to the target projection, prior to the visualization with matplotlib. This drastically increases the performance and makes it possible to visualize even very large data sets. As such, psy-maps visualizes a global scalar field on a hexagonal grid of roughly 4.4 million grid cells ( $\approx 13$  km resolution) in roughly 3.5 minutes. The interactive usage of such a large dataset is however limited by the functionalities of matplotlib to handle such an immense amount of data.

### **psy-reg: The psyplot plugin for visualizing and calculating regression plots**

psy-reg performs regression analysis on 1D variables using the methods of the statsmodels (Seabold and Perktold, 2010) and scipy (Jones et al., 2001; Oliphant, 2007) packages, and visualizes the results with the functionalities of the psy-simple plugin. As such, it implements *formatoptions* for univariate regressions, confidence intervals via bootstrapping, and combined plots of the data density and the fitted model (see also supplements 4.D). The necessity for this package arose from the need to visualize a regression model, compare it (visually) with the original data and to use it afterwards. Other python packages either focus only on the generation of the regressions (such as statsmodels or scipy), or on their visualization (such as seaborn (Waskom et al., 2018)). The psyplot plugin makes it possible to generate the visualization and to access the underlying regression model parameters and uncertainties.

psy-reg has been heavily used for the parameterization of the weather generator in chapter 6 which also gave the initial motivation for the package.

### **psy-strat: A psyplot plugin for stratigraphic plots**

psy-strat (Sommer, 2019a) is the latest plugin for psyplot that has been developed for stratigraphic diagram visualization. It is particularly designed for the stratizite software (Sommer et al., 2019, chapter 3) and was motivated by the need for an automated creation of pollen diagrams. One example of such a diagram is provided in the supplementary material, section 4.E.

As the psy-reg and psy-maps plugins, psy-strat uses the functionalities of the psy-simple plugin for a visualization of multiple variables in separate diagrams that share one common vertical axis (usually age or depth)<sup>5</sup>. Additionally, besides the integration that is common for every psyplot plugin (see next section 4.3.3), psy-strat contains additional functionalities for the psyplot GUI. This implementation allows the user to select and reorder the variables (pollen taxa) that are shown in the stratigraphic diagram.

## **4.3.3 The psyplot Graphical User Interface**

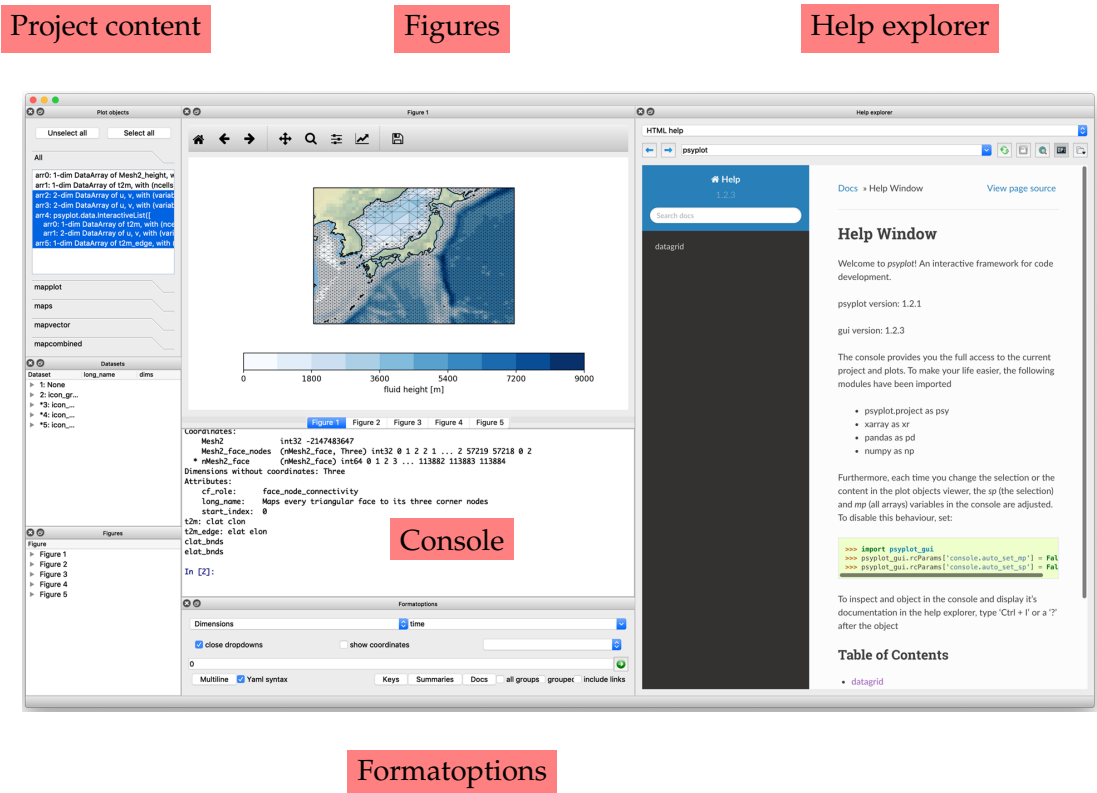
Psyplots objective of providing a platform for flexible and convenient data analysis is further approached with the *psyplot-gui* package. This extension to the framework provides a GUI for simplified access to the plotting features in psyplot.

A strong focus of this interface is, again, the flexibility. psyplot-gui is based on the cross-platform PyQt5 library<sup>6</sup>, a very flexible and frequently used package for

<sup>5</sup>See [psy-strat.readthedocs.io](https://psy-strat.readthedocs.io) for an example of psy-strat.

<sup>6</sup>PyQt5 can be accessed via <https://riverbankcomputing.com/software/pyqt/intro>.





Formatoptions

FIGURE 4.2: Screenshot of the psyplot GUI. The left part shows the content of the psyplot project, the upper center the plots, and the right part contains the help explorer. Below the plots, there is also the IPython console for the usage from the command line and a widget to update the *formatoptions* of the current project.

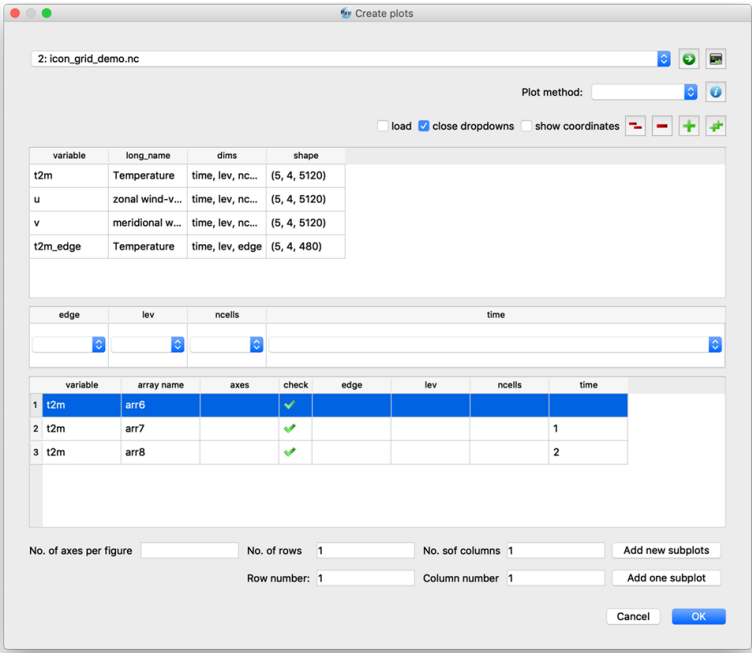


FIGURE 4.3: Plot creation dialog to generate new figures from an xarray dataset.

graphical user interfaces. This enables other software to develop additional features for the package (see `psy-strat` in the previous section 4.3.2, for instance, or `stradtize` in chapter 3) and to flexibly change the layout of the application. The GUI is complemented with an interactive console to provide a fully integrated python environment for data analysis.

The next paragraphs provide an overview on the various widgets, that are also displayed in figure 4.2 and 4.3.

### Console

The central aspects to guarantee flexibility of the application is an in-process IPython console, based on the `qtconsole` package<sup>7</sup> that provides the possibility to communicate with the `psyplot` package via the command line and to load any other module or to run any other script or notebook, or even to run commands in different programming languages, such as R (R Core Team, 2019) or Julia (Bezanson et al., 2017). The console is fully integrated both ways into the GUI. The documentation of every python object in the terminal, for instance, can be viewed in the help explorer of the GUI. And vice versa: a change of the current project through the project content widgets, also changes the corresponding python variable in the shell.

### Help explorer

As a complement to the console, the GUI contains a help explorer to provide immediate and dynamic access to the documentation of python objects in the console, rendered as an HTML webpage<sup>8</sup>. Furthermore, the help explorer is connected to multiple other widgets of the GUI in order to provide a dynamically generated documentation. The documentation of available *formatoptions* in the `psyplot` project, for instance, are rendered as HTML upon request, in order to make the various plot methods more accessible. The same principle works for the plot methods that are accessible in the plot creator.

### Plot creator

The plot creator (figure 4.3) is the starting point of the GUI into the `psyplot` framework (at least, if one does not use the console or a script to generate the plots). It loads data from the disk or the in-process console, and essentially provides a wrapper around the `psyplot` plotting call (see suppl. section 4.A). It additionally displays the documentation of the method and its associated *formatoptions*. This widget creates new plots, that are appended to the `psyplot` project and are accessible through the console and the project content widgets.

### Project content

The `psyplot` project is the most high-level API element in the `psyplot` framework (see section 4.3.1) and is displayed in the project content widgets of the GUI. All other elements, such as the *formatoptions* widget or the plot creator, are interfering with the project, and it is accessible as a variable in the console. The project content widget can be used to see the various items in the project, but it is also used to select

<sup>7</sup><https://github.com/jupyter/qtconsole>

<sup>8</sup>The help explorer widget has been originally motivated by the *Help* widget of the Scientific PYTHON Development Environment, Spyder (<https://www.spyder-ide.org/>) and uses the sphinx package (Hasecke, 2019) to convert restructured Text into HTML.

the specific items for the so-called *current* sub-project. The latter is dynamically set in the console through the *sp* variable and it is used by the *formatoptions* widget to update the plotting parameters of the selected items.

### Formatoptions

As mentioned in section 4.3.1, *formatoptions* are the core elements in psyplot that control the figure aesthetics of the plots and/or perform data manipulations. The generic *formatoptions* widget provides access to these parameters, in order to update them for the selected items in the current project. The *formatoption* itself (i.e. the python object) can in turn generate a widget that is implemented in the *formatoptions* widget, to make the available options more accessible. The *title* *formatoption*, for instance, generates a drop-down menu to select variable attributes (e.g. variable name, variable units, etc.) which is then embedded in the *formatoptions* widget. The modifications of the *formatoptions* via this widgets, updates the figures of the selected items.

### Figures and plots

The plots generated by the plotting methods are displayed in dedicated widgets inside the GUI and can be dynamically adjusted using the *formatoptions* widget or the console. The underlying library of the current implemented psyplot plugins, matplotlib, implements multiple backends to display the data interactively, or to export them as PDF, PNG, etc. The psyplot GUI has implemented a backend on top of the PyQt5 backend of matplotlib, which embeds the figures in the GUI. psyplot can, however, work with any backend of matplotlib and does not depend on the specific implementation.

## 4.4 Conclusions

psyplot (Sommer, 2017e) is a new data visualization framework that integrates rich computational and mathematical software into a flexible framework for visualization. It differs from most of the visual analytic software such that it focuses on extensibility in order to flexibly tackle the different types of analysis questions that arise in pioneering research. The design of the high-level API of the framework enables a simple and standardized usage from the command-line, python scripts or jupyter notebooks. A modular plugin framework enables a flexible development of the framework that can potentially go into many different directions. The additional enhancement with a flexible GUI makes it the only visualization framework that can be handled from the conveniently command-line, and via point-click handling. It also allows to build further desktop applications on top of the existing framework.

The plugins of psyplot currently provide visualization methods that range from simple line plots, to density plots, regression analysis and geo-referenced visualization in two dimensions. The software is currently entirely based on the visualization methods of matplotlib (Hunter, 2007), the most established visualization package in the scientific python community. However, the framework itself is agnostic to the underlying visualization method and can, as such, leverage a variety of existing analytical software.

## 4.5 Outlook

The possibilities for further development of the psyplot framework are numerous, due to its intrinsic generality. The core of the psyplot framework will, in the future, be extended with a standardized algorithm for the generation of animations. Psyplot projects already have the functionality of being saved to a file and reloaded, but they will also be exportable as python scripts for a more flexible reusability and adaptability. The update process within a psyplot project (currently every item in the project is updated in parallel) also has potential for improvement by using a single-threaded scheduler approach that better reflects if one formatoption depends on the *formatoptions* of another plotter.

The GUI has especially high potential for further development, as it still lacks widgets to quickly and intuitively modify the visual appearance of the plots. The only possibility inside the GUI (besides the console) is to use the *formatoptions* widget whose main focus however is on flexibility, rather than usability and has, as such, limited possibilities for adaptation to specific use cases.

Another focus will be the development of new plot methods inside the psyplot framework. The major aspect will be on the development of 3D visualization methods of geo-referenced data, using recently published software that builds on top of the visualization toolkit VTK (Sullivan and Kaszynski, 2019; Sullivan and Trainor-Guitton, 2019), see Sommer, 2019b. psyplot has the unique potential to generate 3D visualizations conveniently from the command line, a distinguishing feature, compared to other visualization software packages, such as ParaView or Vapor. Further potential enhancements for visualizations can involve standard interactive visual analytic tools, e.g. such that the interactive selection of features in one plot affects the visualization in another plot (so-called brushing and linking).

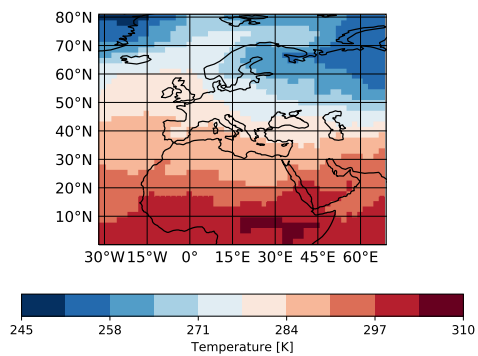
## Supplementary material

### 4.A Example call of a plot method

```
# example call for generating a map
import psyplot.project as psy

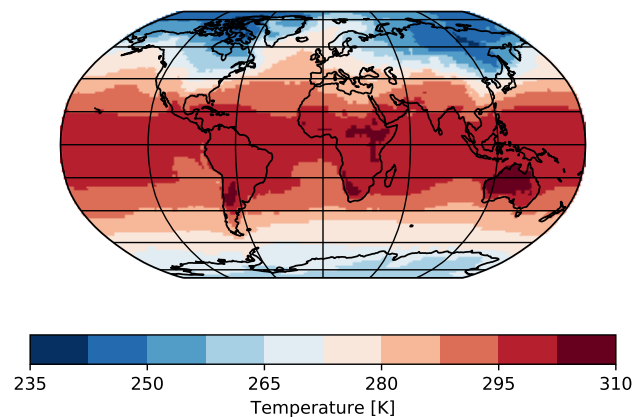
maps = psy.plot.mapplot(
    'psy-maps-demo.nc', # input file name, can also be data in memory
    name='t2m',         # variable to plot (can also be multiples
    ### formatoptions
    # colorbar label uses meta attributes of netCDF variable
    clabel='%(long_name)s [%(units)s]',
    # select colormap
    cmap='RdBu_r',
    # focus on a specific lonlatbox given by [lonmin, lonmax, latmin, latmax]
    lonlatbox=['Europe', 'Europe', 0, 'Europe'])
```

```
maps.show()
```

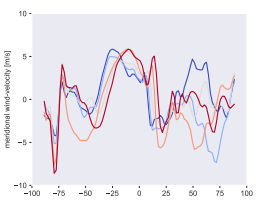
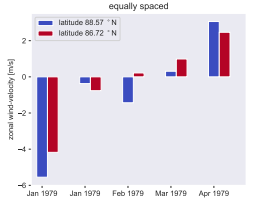
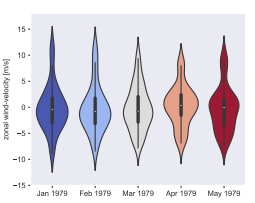
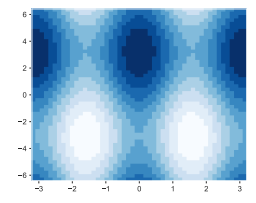
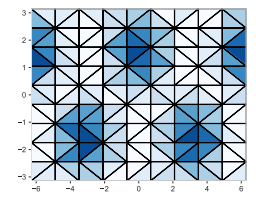
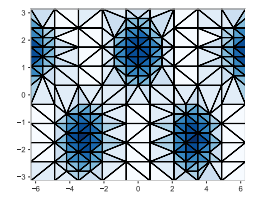
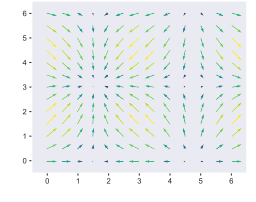
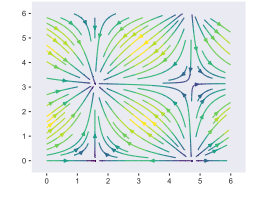
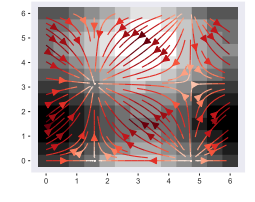

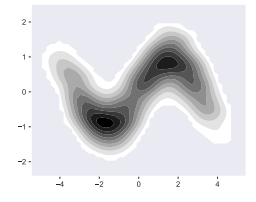
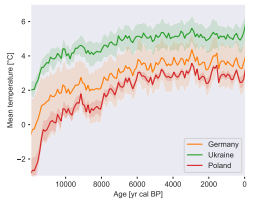


```
# Update the plot, e.g. change projection, plot global
```

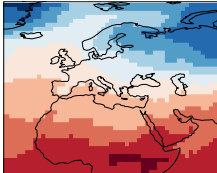
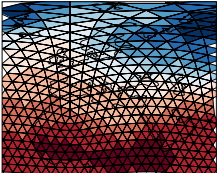

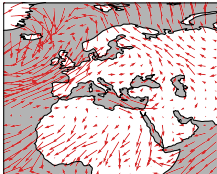
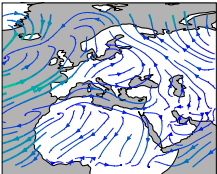
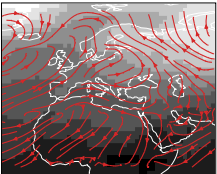
```
maps.update(projection='robin', lonlatbox=None)
```



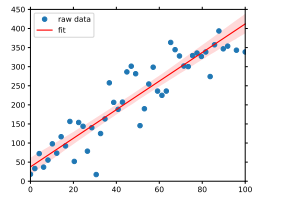
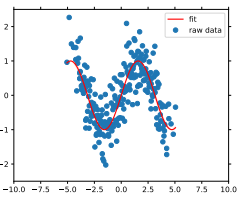
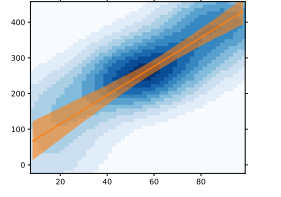
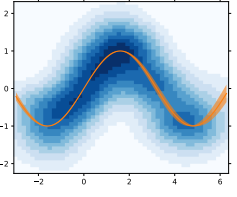
## 4.B psy-simple plot methods

Plot method	lineplot	barplot	violinplot
Example			
Plot method	plot2d		
Grid type	rectilinear	unstructured	
Example			
Plot method	vector		combined
Example			
Plot method	density		fldmean
Example			

4.C psy-maps plot methods

Plot method	mapplot		
Grid type	rectilinear	unstructured	
Example			
Plot method	mapvector	combined	
Example			

4.D psy-reg plot methods

Plot method	linreg	
Example		
Plot method	densityreg	
Example		

## 4.E psy-strat plot methods



## References

- Ayachit, Utkarsh (2015). *The paraview guide: a parallel visualization application*. Kitware, Inc.
- Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B. Shah (Jan. 2017). “Julia: A Fresh Approach to Numerical Computing”. In: *SIAM Review* 59.1, pp. 65–98. DOI: [10.1137/141000671](https://doi.org/10.1137/141000671). eprint: <https://doi.org/10.1137/141000671>. URL: <https://doi.org/10.1137/141000671>.
- Böttinger, Michael and Niklas Röber (2019). “Visualization in Climate Modelling”. In: *International Climate Protection*. Ed. by Michael Palocz-Andresen, Dóra Szalay, András Gosztom, László Sipos, and Tímea Taligàs. Cham: Springer International Publishing, pp. 313–321. ISBN: 978-3-030-03816-8. DOI: [10.1007/978-3-030-03816-8\\_39](https://doi.org/10.1007/978-3-030-03816-8_39). URL: [https://doi.org/10.1007/978-3-030-03816-8\\_39](https://doi.org/10.1007/978-3-030-03816-8_39).
- Brown, Stewart A., Mike Folk, Gregory Goucher, Russ Rew, and Paul F. Dubois (1993). “Software for Portable Scientific Data Management”. In: *Computers in Physics* 7.3, p. 304. DOI: [10.1063/1.4823180](https://doi.org/10.1063/1.4823180).
- Clyne, John, Pablo Mininni, Alan Norton, and Mark Rast (Aug. 2007). “Interactive desktop analysis of high resolution simulations: application to turbulent plume dynamics and current sheet formation”. In: *New Journal of Physics* 9.8, pp. 301–301. DOI: [10.1088/1367-2630/9/8/301](https://doi.org/10.1088/1367-2630/9/8/301).
- Dask Development Team (2016). *Dask: Library for dynamic task scheduling*. URL: <https://dask.org>.
- Dawson, Andrew (Apr. 2016). “eofs: A Library for EOF Analysis of Meteorological, Oceanographic, and Climate Data”. In: *Journal of Open Research Software* 4. DOI: [10.5334/jors.122](https://doi.org/10.5334/jors.122).
- Eyring, Veronika, Mattia Righi, Axel Lauer, Martin Evaldsson, Sabrina Wenzel, Colin Jones, Alessandro Anav, Oliver Andrews, Irene Cionni, Edouard L. Davin, Clara Deser, Carsten Ehbrecht, Pierre Friedlingstein, Peter Gleckler, Klaus-Dirk Gottschaldt, Stefan Hagemann, Martin Juckes, Stephan Kindermann, John Krasting, Dominik Kunert, Richard Levine, Alexander Loew, Jarmo Mäkelä, Gill Martin,



- Erik Mason, Adam S. Phillips, Simon Read, Catherine Rio, Romain Roehrig, Daniel Senftleben, Andreas Sterl, Lambertus H. van Ulft, Jeremy Walton, Shiyu Wang, and Keith D. Williams (May 2016). “ESMValTool (v1.0) – a community diagnostic and performance metrics tool for routine evaluation of Earth system models in CMIP”. In: *Geoscientific Model Development* 9.5, pp. 1747–1802. DOI: [10.5194/gmd-9-1747-2016](https://doi.org/10.5194/gmd-9-1747-2016). URL: <https://www.geosci-model-dev.net/9/1747/2016/>.
- Hasecke, Jan Ulrich (2019). *Software-Dokumentation mit Sphinx: Zweite überarbeitete Auflage (Sphinx 2.0) (German Edition)*. Independently published. ISBN: 1793008779. URL: <https://www.amazon.com/Software-Dokumentation-mit-Sphinx-%C3%BCberarbeitete-Auflage/dp/1793008779?SubscriptionId=AKIAIOBINVZYXZQZ2U3A%5C&tag=chimbori05-20%5C&linkCode=xm2%5C&camp=2025%5C&creative=165953%5C&creativeASIN=1793008779>.
- Hoyer, S. and J. Hamman (2017). “xarray: N-D labeled arrays and datasets in Python”. In: *Journal of Open Research Software* 5.1. DOI: [10.5334/jors.148](https://doi.org/10.5334/jors.148). URL: <http://doi.org/10.5334/jors.148>.
- Hunter, J. D. (May 2007). “Matplotlib: A 2D Graphics Environment”. In: *Computing in Science Engineering* 9.3, pp. 90–95. ISSN: 1521-9615. DOI: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55).
- Jagers, Bert, David Stuebe, Tom Gross, Chris Barker, Brian Zelenke, Rich Signell, Bob Oehmke, Alex Crosby, Karen Schuchardt, David Ham, Brian Blanton, Cristina Forbes, Charles Seaton, Dave Forrest, Bill Howe, Geoff Cowles, and Phil Elson (Aug. 2018). *UGRID Conventions (v1.0)*. URL: <http://ugrid-conventions.github.io/ugrid-conventions> (visited on 09/05/2019).
- Jones, Eric, Travis Oliphant, Pearu Peterson, et al. (2001). *SciPy: Open source scientific tools for Python*. [Online; accessed 2017-02-18]. URL: <http://www.scipy.org/>.
- Keim, Daniel, Gennady Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon (2008). “Visual Analytics: Definition, Process, and Challenges”. In: *Information Visualization: Human-Centered Issues and Perspectives*. Ed. by Andreas Kerren, John T. Stasko, Jean-Daniel Fekete, Chris North, Andreas Kerren, John T. Stasko, Jean-Daniel Fekete, and Chris North. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 154–175. ISBN: 978-3-540-70956-5. DOI: [10.1007/978-3-540-70956-5\\_7](https://doi.org/10.1007/978-3-540-70956-5_7). URL: [https://doi.org/10.1007/978-3-540-70956-5\\_7](https://doi.org/10.1007/978-3-540-70956-5_7).
- Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing (2016). “Jupyter Notebooks – a publishing format for reproducible computational workflows”. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press, pp. 87–90. DOI: [10.3233/978-1-61499-649-1-87](https://doi.org/10.3233/978-1-61499-649-1-87).
- McKinney, Wes (2010). “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman, pp. 51–56.
- Met Office (2010 - 2015). *Cartopy: a cartographic python library with a matplotlib interface*. Exeter, Devon. URL: <http://scitools.org.uk/cartopy>.
- Müller, Ralf (2019). *cdo-bindings: Ruby/Python bindings for CDO*. URL: <https://github.com/Try2Code/cdo-bindings> (visited on 09/05/2019).
- Nocke, T., S. Buschmann, J. F. Donges, N. Marwan, H.-J. Schulz, and C. Tominski (Sept. 2015). “Review: visual analytics of climate networks”. In: *Nonlinear Processes in Geophysics* 22.5, pp. 545–570. DOI: [10.5194/npg-22-545-2015](https://doi.org/10.5194/npg-22-545-2015). URL: <https://www.nonlin-processes-geophys.net/22/545/2015/>.

- Oliphant, Travis E (2006). *A guide to NumPy*. Vol. 1. Trelgol Publishing USA. URL: <http://www.numpy.org/>.
- (2007). “Python for Scientific Computing”. In: *Computing in Science & Engineering* 9.3, pp. 10–20. DOI: [10.1109/mcse.2007.58](https://doi.org/10.1109/mcse.2007.58).
- Perez, Fernando, Brian E. Granger, and John D. Hunter (Mar. 2011). “Python: An Ecosystem for Scientific Computing”. In: *Computing in Science & Engineering* 13.2, pp. 13–21. DOI: [10.1109/mcse.2010.119](https://doi.org/10.1109/mcse.2010.119).
- R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: <https://www.R-project.org/>.
- Rautenhaus, Marc, Michael Böttinger, Stephan Siemen, Robert Hoffman, Robert M. Kirby, Mahsa Mirzargar, Niklas Rober, and Rudiger Westermann (Dec. 2018). “Visualization in Meteorology—A Survey of Techniques and Tools for Data Analysis Tasks”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.12, pp. 3268–3296. DOI: [10.1109/tvcg.2017.2779501](https://doi.org/10.1109/tvcg.2017.2779501).
- Rew, R. and G. Davis (July 1990). “NetCDF: an interface for scientific data access”. In: *IEEE Computer Graphics and Applications* 10.4, pp. 76–82. DOI: [10.1109/38.56302](https://doi.org/10.1109/38.56302).
- Schulzweida, Uwe (Feb. 2019). *CDO User Guide*. DOI: [10.5281/zenodo.2558193](https://doi.org/10.5281/zenodo.2558193). URL: <https://doi.org/10.5281/zenodo.2558193>.
- Seabold, Skipper and Josef Perktold (2010). *Statsmodels: Econometric and statistical modeling with python*.
- Sommer, Philipp S. (Aug. 2017a). “Chilipp/psy-maps: v1.0.0: First official and stable release”. In: DOI: [10.5281/zenodo.845712](https://doi.org/10.5281/zenodo.845712). URL: <https://doi.org/10.5281/zenodo.845712>.
- (Aug. 2017b). “Chilipp/psy-reg: v1.0.0: First official and stable release”. In: DOI: [10.5281/zenodo.845717](https://doi.org/10.5281/zenodo.845717). URL: <https://doi.org/10.5281/zenodo.845717>.
- (Aug. 2017c). “Chilipp/psy-simple: v1.0.0: First official and stable release”. In: DOI: [10.5281/zenodo.845705](https://doi.org/10.5281/zenodo.845705). URL: <https://doi.org/10.5281/zenodo.845705>.
- (Aug. 2017d). “Chilipp/psyplot-gui: v1.0.1: Graphical User Interface for the psyplot package”. In: DOI: [10.5281/zenodo.845726](https://doi.org/10.5281/zenodo.845726). URL: <https://doi.org/10.5281/zenodo.845726>.
- (Aug. 2017e). “The psyplot interactive visualization framework”. In: *The Journal of Open Source Software* 2.16. DOI: [10.21105/joss.00363](https://doi.org/10.21105/joss.00363). URL: <https://doi.org/10.21105/joss.00363>.
- (Aug. 2019a). *psy-strat v0.1.0: A Python package for creating stratigraphic diagrams*. DOI: [10.5281/zenodo.3381753](https://doi.org/10.5281/zenodo.3381753). URL: <https://doi.org/10.5281/zenodo.3381753>.
- (2019b). *psy-vtk: A VTK plugin for psyplot*. Last accessed: 2019-05-27. URL: <https://github.com/Chilipp/psy-vtk> (visited on 05/27/2019).
- Sommer, Philipp S., Dilan Rech, Manuel Chevalier, and Basil A. S. Davis (Feb. 2019). “straditize: Digitizing stratigraphic diagrams”. In: *Journal of Open Source Software* 4.34, p. 1216. DOI: [10.21105/joss.01216](https://doi.org/10.21105/joss.01216). URL: <https://doi.org/10.21105/joss.01216>.
- Sullivan, C. and Alexander Kaszynski (May 2019). “PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)”. In: *Journal of Open Source Software* 4.37, p. 1450. DOI: [10.21105/joss.01450](https://doi.org/10.21105/joss.01450).
- Sullivan, C. and Whitney Trainor-Guitton (June 2019). “PVGeo: an open-source Python package for geoscientific visualization in VTK and ParaView”. In: *Journal of Open Source Software* 4.38, p. 1451. DOI: [10.21105/joss.01451](https://doi.org/10.21105/joss.01451).

- Uchida, Takaya, Ariel Rokem, Tom Nicholas, Ryan Abernathey, Jake Vanderplas, Yaroslav Halchenko, Andreas Mayer, Greg Wilson, Kai Pak, and Aurélien Ponte (2019). *xgcm/xrft v0.2.0*. DOI: [10.5281/zenodo.1402635](https://doi.org/10.5281/zenodo.1402635).
- Waskom, Michael, Olga Botvinnik, Drew O’Kane, Paul Hobson, Joel Ostblom, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmerhoven, Julian De Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Alistair Miles, Yoav Ram, Thomas Brunner, Tal Yarkoni, Mike Lee Williams, Constantine Evans, Clark Fitzgerald, , Brian, and Adel Qalieh (2018). *mwaskom/seaborn: v0.9.0 (July 2018)*. DOI: [10.5281/zenodo.592845](https://doi.org/10.5281/zenodo.592845).
- Wijk, J. J. van (Oct. 2005). “The Value of Visualization”. In: *VIS 05. IEEE Visualization, 2005*. IEEE, pp. 79–86. DOI: [10.1109/visual.2005.1532781](https://doi.org/10.1109/visual.2005.1532781).
- Wong, Pak Chung, Han-Wei Shen, Ruby Leung, Samson Hagos, Teng-Yok Lee, Xin Tong, and Kewei Lu (Nov. 2014). “Visual analytics of large-scale climate model data”. In: *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, pp. 85–92. DOI: [10.1109/ldav.2014.7013208](https://doi.org/10.1109/ldav.2014.7013208).

# List of Abbreviations

**APD** African Pollen Database. 15, 30, 139

**API** Application programming interface. 19, 55, 57, 60, 88

**CDO** Climate Data Operator. 54, 55

**CI** Continuous Integration. 18, 19, 30, 32, 35

**EMPD** Eurasian Modern Pollen Database. 20, 29, 30, 31, 32, 33, 35, 36, 73, 139

**EPD** European Pollen Database. 73

**ESM** Earth System Model. 14, 54

**FOSS** Free and Open-Source Software. 17, 18, 19

**GAM** Generalized Additive Model. 77, 78, 79

**GUI** graphical user interface. 19, 20, 53, 54, 58, 59, 60, 61, 62, 140

**JJA** summer (June, July and August). 72, 75, 84

**LAPD** Latin American Pollen Database. 15, 30, 139

**LGM** Last Glacial Maximum. 14

**MAT** modern analogue technique. 75, 84, 90, 95, 97, 140

**MCMC** Markov chain Monte Carlo. 82, 83

**NAPD** North American Pollen Database. 15

**NCDC** National Climatic Data Center. 16

**NOAA** National Oceanic and Atmospheric Administration. 16

**PMIP** Paleoclimate Modelling Intercomparison Project. 14