Pyleoclim Documentation

Release 0.4.0

Deborah Khider, Julien Emile-Geay, Feng Zhu

Mar 08, 2018

CONTENTS

1 Pyleoclim **3**
- 1.1 What is it? . . . 3
- 1.2 Version Information . . . 3
- 1.3 Installation . . . 4
- 1.4 Quickstart guide . . . 4
- 1.5 Requirements . . . 4
- 1.6 Further information . . . 5
- 1.7 Contact . . . 5
- 1.8 License . . . 5
- 1.9 Disclaimer . . . 5

2 Main Functions **7**
- 2.1 Using Pyleoclim with a LiPD file . . . 7
  - 2.1.1 Getting started . . . 7
  - 2.1.2 Mapping . . . 7
  - 2.1.3 Plotting . . . 9
  - 2.1.4 Statistics . . . 11
  - 2.1.5 Timeseries . . . 12
  - 2.1.6 Analysis in the frequency domain . . . 13
  - 2.1.7 Age modelling . . . 15
- 2.2 Using Pyleoclim without a LiPD file . . . 17

3 Mapping Functions **19**

4 Plotting Functions **21**

5 Statistics Functions **23**

6 Timeseries Functions **25**

7 LiPD Utilities **27**
- 7.1 Creating Directories and saving . . . 27
- 7.2 LiPD files . . . 27
- 7.3 Handling Variables . . . 28
- 7.4 Handling timeseries objects . . . 28
- 7.5 Linking LiPDs to the LinkedEarth Ontology . . . 29
- 7.6 Dealing with models . . . 29
- 7.7 Extracting tables . . . 29
- 7.8 Dealing with ensembles . . . 30

8 Summary Plots **31**

9 Spectral Functions 33

10 RBchron 39

11 Indices and tables 43

Index 45

Contents:

CHAPTER ONE

PYLEOCLIM

1.1 What is it?

Pyleoclim is a Python package primarily geared towards the analysis and visualization of paleoclimate data. Such data often come in the form of timeseries with missing values and age uncertainties, and the package includes several low-level methods to deal with these issues, as well as high-level methods that re-use those to perform scientific workflows.

The package assumes that the data are stored in the Linked Paleo Data (LiPD) format and makes extensive use of the LiPD utilities. The package is aware of age ensembles stored via LiPD and uses them for time-uncertain analyses very much like GeoChronR.

Current Capabilities:

- binning
- interpolation
- plotting maps, timeseries, and basic age model information
- paleo-aware correlation analysis (isopersistent, isospectral, and classical t-test)
- weighted wavelet Z transform (WWZ)
- age modeling through Bchron

Future capabilities:

- paleo-aware singular spectrum analysis (AR(1) null eigenvalue identification, missing data)
- spectral analysis (Multi-Taper Method, Lomb-Scargle)
- cross-wavelet analysis
- index reconstruction
- climate reconstruction
- ensemble methods for most of the above

1.2 Version Information

0.4.0: New functionalities: map nearest records by archive type, plot ensemble time series, age modelling through Bchron.

0.3.1: New functionalities: segment a timeseries using a gap detection criteria, update to summary plot to perform spectral analysis

0.3.0: Compatibility with LiPD 1.3 and Spectral module added

0.2.5: Fix error on loading (Looking for Spectral Module)
0.2.4: Fix load error from init
0.2.3: Freeze LiPD version to 1.2 to avoid conflicts with 1.3
0.2.2: Change progressbar to tqdm and add standardization function
0.2.1: Update package requirements
0.2.0: Restructure the package so that the main functions can be called without the use of a LiPD files and associated timeseries objects.
0.1.4: Rename functions using camel case convention and consistency with LiPD utilities version 0.1.8.5
0.1.3: Compatible with LiPD utilities version 0.1.8.5
    Function openLiPD() renamed openLiPDs()
0.1.2: Compatible with LiPD utilities version 0.1.8.3
    Uses Basemap instead of cartopy
0.1.1: Freezes the package prior to version 0.1.8.2 of LiPD utilities
0.1.0: First release

1.3 Installation

Python v3.4+ is required. Tested with Python v3.5

Will not run on a Windows system.

Pyleoclim is published through Pypi and easily installed via pip:

```
pip install pyleoclim
```

1.4 Quickstart guide

1. Open your command line application (Terminal or Command Prompt)
2. Install with command:

   pip install pyleoclim
3. Wait for installation to complete, then:

1. Import the package into your favorite Python environment (we recommend the use of Spyder, which comes standard with the Anaconda build)
2. Use Jupyter Notebook to go through the tutorial contained in the PyleolimQuickstart.ipynb

1.5 Requirements

- LiPD v0.2.5+
- pandas v0.22+
- numpy v1.14+
- matplotlib v2.0+
- Basemap v1.0.7+
- scipy v0.19.0+

- statsmodel v0.8.0+
- seaborn v0.7.0+
- scikit-learn v0.17.1+
- tqdm v4.14.0+
- pathos v0.2.0+
- tqdm 4.14+
- rpy2 2.8.4+

The installer will automatically check for the needed updates.

1.6 Further information

GitHub: https://github.com/LinkedEarth/Pyleoclim_util
LinkedEarth: http://linked.earth
Python and Anaconda: http://conda.pydata.org/docs/test-drive.html
Jupyter Notebook: http://jupyter.org/

1.7 Contact

Please report issues to linkedearth@gmail.com

1.8 License

The project is licensed under the GNU Public License .

1.9 Disclaimer

This material is based upon work supported by the U.S. National Science Foundation under Grant Number ICER-1541029. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the investigators and do not necessarily reflect the views of the National Science Foundation.

CHAPTER

TWO

MAIN FUNCTIONS

2.1 Using Pyleoclim with a LiPD file

2.1.1 Getting started

Pyleoclim relies heavily on the concept of timeseries objects introduced in LiPD and implemented in the LiPD utilities.

Briefly, timeseries objects are dictionaries containing the ChronData values and PaleoData values as well as the metadata associated with the record. If one record has three ProxyObservations (e.g., Mg/Ca, d18O, d13C) then it will have three timeseries objects, one for each of the observations.

The LiPD utilities function lipd.extractTs() returns a list of dictionaries for the selected LiPD files, which need to be passed to Pyleoclim along with the path to the directory containing the LiPD files.

This is done through the functions pyleoclim.openLiPD() and pyleoclim.fetchTs(), which are lightweight versions of their counterparts in the LiPD utilities:

`pyleoclim.`**`openLipd`**`(`*usr_path=”*`)`

Read Lipd files into a dictionary

Sets the dictionary as global variable so that it doesn’t have to be provided as an argument for every function.

Args: usr_path (str): The path to a directory or a single file. (Optional argument)

Returns: lipd_dict - a dictionary containing the LiPD library

`pyleoclim.`**`fetchTs`**`(`*lipds=None*`)`

Extract timeseries dictionary

This function is based on the function of the same name in the LiPD utilities. Set the dictionary as a global variable so that it doesn’t have to be provided as an argument for every function.

Args: lipds (dict): A dictionary of LiPD files obtained through the readLipd function

Returns: ts_list - A list of timeseries object

2.1.2 Mapping

`pyleoclim.`**`mapAllArchive`**`(`*lipds=”, markersize=50, background=’shadedrelief’, figsize=[10, 4], saveFig=False, dir=”, format=’eps’*`)`

Map all the available records loaded into the workspace by archiveType.

Map of all the records into the workspace by archiveType. Uses the default color palette. Enter pyleoclim.plot_default for detail.

Args: lipds (dict): A list of LiPD files. (Optional) markersize (int): The size of the markers. Default is 50 background (str): Plots one of the following images on the map:

bluemarble, etopo, shadedrelief, or none (filled continents). Default is shadedrelief.

figsize (list): the size for the figure ax: Return as axis instead of figure (useful to integrate plot into a subplot) saveFig (bool): Default is to not save the figure dir (str): The absolute path of the directory in which to save the

figure. If not provided, creates a default folder called 'figures' in the LiPD working directory (lipd.path).

format (str): One of the file extensions supported by the active backend. Default is "eps". Most backend support png, pdf, ps, eps, and svg.

Returns: The figure

`pyleoclim.`**`mapLipd`**(*timeseries=", countries=True, counties=False, rivers=False, states=False, background='shadedrelief', scale=0.5, markersize=50, marker='default', figsize=[4, 4], saveFig=False, dir=", format='eps'*)

Create a Map for a single record

Orthographic projection map of a single record.

Args: timeseries: a LiPD timeseries object. Will prompt for one if not given countries (bool): Draws the country borders. Default is on (True). counties (bool): Draws the USA counties. Default is off (False). rivers (bool): Draws the rivers. Default is off (False). states (bool): Draws the American and Australian states borders.

Default is off (False)

background (str): Plots one of the following images on the map: bluemarble, etopo, shadedrelief, or none (filled continents). Default is shadedrelief

scale (float): useful to downgrade the original image resolution to speed up the process. Default is 0.5.

markersize (int): default is 50 marker (str): a string (or list) containing the color and shape of the

marker. Default is by archiveType. Type pyleo.plot_default to see the default palette.

figsize (list): the size for the figure saveFig (bool): default is to not save the figure dir (str): the full path of the directory in which to save the figure.

If not provided, creates a default folder called 'figures' in the LiPD working directory (lipd.path).

format (str): One of the file extensions supported by the active backend. Default is "eps". Most backend support png, pdf, ps, eps, and svg.

Returns: The figure

`pyleoclim.`**`mapNearRecords`**(*timeseries=", lipds=", n=5, radius=None, sameArchive=False, projection='ortho', lat_0=", lon_0=", llcrnrlat=-90, urcrnrlat=90, llcrnrlon=-180, urcrnrlon=180, countries=True, counties=False, rivers=False, states=False, background='shadedrelief', scale=0.5, markersize=200, markersize_adjust=True, marker_r='ko', marker_c='default', cmap='Reds', colorbar=True, location='right', label='Distance in km', figsize=[4, 4], ax=None, saveFig=False, dir=", format='eps'*)

Map the nearest records from the record of interest

Args: timeseries (dict): A timeseries object. If none given, will prompt for one lipds (list): A list of LiPD files. (Optional) n (int): the number of records to match radius (float): The distance (in km) to search for nearby records.

Default is to search the entire globe

sameArchive (bool): Returns only records with the same archiveType. Default is not to do so.

projection (string): the map projection. Refers to the Basemap documentation for a list of available projections. Only projections supporting setting the map center with a single lat/lon or with the coordinates of the rectangle are currently supported. Default is to use a Robinson projection.

lat_0, lon_0 (float): the center coordinates for the map. Default is mean latitude/longitude in the list. If the chosen projection doesn't support it, Basemap will ignore the given values.

llcrnrlat, urcrnrlat, llcrnrlon, urcrnrlon (float): The coordinates of the two opposite corners of the rectangle.

countries (bool): Draws the countries border. Defaults is off (False). counties (bool): Draws the USA counties. Default is off (False). rivers (bool): Draws the rivers. Default is off (False). states (bool): Draws the American and Australian states borders.

Default is off (False).

background (string): Plots one of the following images on the map: bluemarble, etopo, shadedrelief, or none (filled continents). Default is none.

scale (float): Useful to downgrade the original image resolution to speed up the process. Default is 0.5.

markersize (int): the size of the marker markersize_adjust (bool): If True, will proportionaly adjust the size of

the marker according to distance.

marker_r (list or str): The color and shape of the marker for the reference record.

marker_c (list or str): The color and shape of the marker for the other records. Default is to use the color palette by archiveType. If set to None then the color of the marker will represent the distance from the reference records.

cmap (str): The colormap to use to represent the distance from the reference record if no marker is selected.

colorbar (bool): Create a colorbar. Default is True location (str): Location of the colorbar label (str): Label for the colorbar. figsize (list): the size for the figure ax: Return as axis instead of figure (useful to integrate plot into a subplot) saveFig (bool): default is to not save the figure dir (str): the full path of the directory in which to save the figure.

If not provided, creates a default folder called 'figures' in the LiPD working directory (lipd.path).

format (str): One of the file extensions supported by the active backend. Default is "eps". Most backend support png, pdf, ps, eps, and svg.

Returns: ax - The figure

2.1.3 Plotting

`pyleoclim.`**`plotTs`**`(`*timeseries=", x_axis=", markersize=50, marker='default', figsize=[10, 4], saveFig=False, dir=", format='eps'*`)`

Plot a single time series.

Args: A timeseries: By default, will prompt the user for one. x_axis (str): The representation against which to plot the paleo-data.

Options are “age”, “year”, and “depth”. Default is to let the system choose if only one available or prompt the user.

markersize (int): default is 50. marker (str): a string (or list) containing the color and shape of the

marker. Default is by archiveType. Type pyleo.plot_default to see the default palette.

figsize (list): the size for the figure saveFig (bool): default is to not save the figure dir (str): the full path of the directory in which to save the figure.

If not provided, creates a default folder called ‘figures’ in the LiPD working directory (lipd.path).

format (str): One of the file extensions supported by the active backend. Default is “eps”. Most backend support png, pdf, ps, eps, and svg.

Returns: The figure.

`pyleoclim.`**`histTs`**(*timeseries=”, bins=None, hist=True, kde=True, rug=False, fit=None, hist_kws={’label’: ’Histogram’}, kde_kws={’label’: ’KDE fit’}, rug_kws={’label’: ’Rug’}, fit_kws={’label’: ’Fit’}, color=’default’, vertical=False, norm_hist=True, figsize=[5, 5], saveFig=False, format=’eps’, dir=”*)

Plot a univariate distribution of the PaleoData values

This function is based on the seaborn displot function, which is itself a combination of the matplotlib hist function with the seaborn kdeplot() and rugplot() functions. It can also fit scipy.stats distributions and plot the estimated PDF over the data.

Args: timeseries: A timeseries. By default, will prompt the user for one. bins (int): Specification of hist bins following matplotlib(hist),

or None to use Freedman-Diaconis rule

hist (bool): Whether to plot a (normed) histogram kde (bool): Whether to plot a gaussian kernel density estimate rug (bool): Whether to draw a rugplot on the support axis fit: Random variable object. An object with fit method, returning

a tuple that can be passed to a pdf method of positional arguments following a grid of values to evaluate the pdf on.

{hist, kde, rug, fit}_kws: Dictionaries. Keyword arguments for underlying plotting functions. If modifying the dictionary, make sure the labels “hist”, “kde”, “rug” and “fit” are still passed.

color (str): matplotlib color. Color to plot everything but the fitted curve in. Default is to use the default paletter for each archive type.

vertical (bool): if True, oberved values are on y-axis. norm_hist (bool): If True (default), the histrogram height shows

a density rather than a count. This is implied if a KDE or fitted density is plotted

figsize (list): the size for the figure saveFig (bool): default is to not save the figure dir (str): the full path of the directory in which to save the figure.

If not provided, creates a default folder called ‘figures’ in the LiPD working directory (lipd.path).

format (str): One of the file extensions supported by the active backend. Default is “eps”. Most backend support png, pdf, ps, eps, and svg.

Returns fig - The figure

Summary Plots

Summary plots are a special categories of plots enabled by Pyleoclim. They allow to plot specific information about a timeseries but are not customizable.

`pyleoclim.`**`summaryTs`**`(`*timeseries=''*, *x_axis=''*, *saveFig=False*, *dir=''*, *format='eps'*`)`

Basic summary plot

Plots the following information: the time series, a histogram of the PaleoData_values, location map, spectral density using the wwz method, and metadata about the record.

Args: timeseries: a timeseries object. By default, will prompt for one x_axis (str): The representation against which to plot the paleo-data.

Options are "age", "year", and "depth". Default is to let the system choose if only one available or prompt the user.

saveFig (bool): default is to not save the figure dir (str): the full path of the directory in which to save the figure.

If not provided, creates a default folder called 'figures' in the LiPD working directory (lipd.path).

format (str): One of the file extensions supported by the active backend. Default is "eps". Most backend support png, pdf, ps, eps, and svg.

Returns: The figure

2.1.4 Statistics

`pyleoclim.`**`statsTs`**`(`*timeseries=''*`)`

Calculate simple statistics of a timeseries

Args: timeseries: sytem will prompt for one if not given

Returns: the mean, median, min, max, standard deviation and the inter-quartile range (IQR) of a timeseries.

Examples:

```
>>> mean, median, min_, max_, std, IQR = pyleo.statsTs(timeseries)
```

`pyleoclim.`**`corrSigTs`**`(`*timeseries1=''*, *timeseries2=''*, *x_axis=''*, *interp_step=''*, *start=''*, *end=''*, *nsim=1000*, *method='isospectral'*, *alpha=0.5*`)`

Estimates the significance of correlations between non IID timeseries.

Function written by. F. Zhu.

Args: timeseries1, timeseries2: timeseries object. Default is blank. x-axis (str): The representation against which to express the

paleo-data. Options are "age", "year", and "depth". Default is to let the system choose if only one available or prompt the user.

interp_step (float): the step size. By default, will prompt the user. start (float): Start time/age/depth. Default is the maximum of

the minima of the two timeseries

end (float): End time/age/depth. Default is the minimum of the maxima of the two timeseries

nsim (int): the number of simulations. Default is 1000 method (str): method use to estimate the correlation and significance.

Available methods include:

- 'ttest': T-test where the degrees of freedom are corrected for

the effect of serial correlation

- 'isopersistant': AR(1) modeling of the two timeseries
- 'isospectral' (default): phase randomization of original

inputs.

The T-test is parametric test, hence cheap but usually wrong except in idyllic circumstances. The others are non-parametric, but their computational requirements scales with nsim.

alpha (float): significance level for critical value estimation. Default is 0.05

Returns: r (float) - correlation between the two timeseries

sig (bool) - Returns True if significant, False otherwise

p (real) - the p-value

2.1.5 Timeseries

Basic manipulations of the timeseries objects.

`pyleoclim.`**`binTs`**(*timeseries=", x_axis=", bin_size=", start=", end="*)

Bin the paleoData values of the timeseries

Args: timeseries. By default, will prompt the user for one. x-axis (str): The representation against which to plot the paleo-data.

Options are "age", "year", and "depth". Default is to let the system choose if only one available or prompt the user.

bin_size (float): the size of the bins to be used. By default, will prompt for one

start (float): Start time/age/depth. Default is the minimum end (float): End time/age/depth. Default is the maximum

Returns: binned_values- the binned output,

bins- the bins (centered on the median, i.e. the 100-200 bin is 150),

n- number of data points in each bin,

error- the standard error on the mean in each bin

`pyleoclim.`**`interpTs`**(*timeseries=", x_axis=", interp_step=", start=", end="*)

Simple linear interpolation

Simple linear interpolation of the data using the numpy.interp method

Args: timeseries. Default is blank, will prompt for it x-axis (str): The representation against which to plot the paleo-data.

Options are "age", "year", and "depth". Default is to let the system choose if only one available or prompt the user.

interp_step (float): the step size. By default, will prompt the user. start (float): Start year/age/depth. Default is the minimum end (float): End year/age/depth. Default is the maximum

Returns: interp_age - the interpolated age/year/depth according to the end/start and time step,

interp_values - the interpolated values

`pyleoclim.`**`standardizeTs`**(*timeseries=”*, *scale=1*, *ddof=0*, *eps=0.001*)

Centers and normalizes the paleoData values of a given time series.

Constant or nearly constant time series not rescaled.

Args: x (array): vector of (real) numbers as a time series, NaNs allowed scale (real): a scale factor used to scale a record to a match a given variance axis (int or None): axis along which to operate, if None, compute over the whole array ddof (int): degress of freedom correction in the calculation of the standard deviation eps (real): a threshold to determine if the standard deviation is too close to zero

Returns:

- z (array): the standardized time series (z-score), Z = (X - E[X])/std(X)*scale, NaNs allowed
- mu (real): the mean of the original time series, E[X]
- sig (real): the standard deviation of the original time series, std[X]

References:

1. Tapio Schneider’s MATLAB code: http://www.clidyn.ethz.ch/imputation/standardize.m
2. The zscore function in SciPy: https://github.com/scipy/scipy/blob/master/scipy/stats/stats.py

@author: fzhu

`pyleoclim.`**`segmentTs`**(*timeseries=”*, *factor=2*)

Divides a time series into several segments using a gap detection algorithm

Gap detection rule: If the time interval between some two data points is larger than some factor times the mean resolution of the timeseries, then a brak point is applied and the timseries is divided.

Args: timeseries: a LiPD timeseries object factor (float): factor to adjust the threshold. threshold = factor*dt_mean.

Default is 2.

Returns: seg_y (list) - a list of several segments with potentially different length seg_t (list) - A list of the time values for each y segment. n_segs (int) - the number of segments

2.1.6 Analysis in the frequency domain

`pyleoclim.`**`wwzTs`**(*timeseries=”*, *lim=None*, *wwz=False*, *psd=True*, *wwz_default=True*, *psd_default=True*, *wwaplot_default=True*, *psdplot_default=True*, *fig=True*, *saveFig=False*, *dir=”*, *format=’eps’*)

Weigthed wavelet Z-transform analysis

Wavelet analysis for unevenly spaced data adapted from Foster et al. (1996)

Args: timeseries (dict): A LiPD timeseries object (Optional, will prompt for one.) lim (list): Truncate the timeseries between min/max time (e.g., [0,10000]) wwz (bool): If True, will perform wavelet analysis psd (bool): If True, will inform the power spectral density of the timeseries wwz_default: If True, will use the following default parameters:

wwz_default = {‘tau’:None, ‘freqs’:None, ‘c’:1/(8*np.pi**2), ‘Neff’:3, ‘Neff_coi’:3, ‘nMC’:200, ‘nproc’:8, ‘detrend’:’no’, ‘params’ : [“default”,4,0,1], ‘gaussianize’: False, ‘standardize’:True, ‘method’:’Kirchner_f2py’, ‘bc_mode’:’reflect’, ‘reflect_type’:’odd’, ‘len_bd’:0}

Modify the values for specific keys to change the default behavior.

psd_default: If True, will use the following default parameters:

> **psd_default = {'tau':None,** 'freqs': None, 'c':1e-3, 'nproc':8, 'nMC':200, 'detrend':'no', 'params' : ["default",4,0,1], 'gaussianize': False, 'standardize':True, 'Neff':3, 'anti_alias':False, 'avgs':1, 'method':'Kirchner_f2py', }
>
> Modify the values for specific keys to change the default behavior.

wwaplot_default: If True, will use the following default parameters:

> **wwaplot_default={'AR1_q':AR1_q,** 'coi':coi, 'levels':None, 'tick_range':None, 'yticks':None, 'yticks_label': None, 'ylim':None, 'xticks':None, 'xlabels':None, 'figsize':[20,8], 'clr_map':'OrRd', 'cbar_drawedges':False, 'cone_alpha':0.5, 'plot_signif':True, 'signif_style':'contour', 'plot_cone':True, 'title':None, 'ax':None, 'xlabel': label.upper()[0]+label[1:]+'('+s+')', 'ylabel': 'Period ('+ageunits+')', 'cbar_orientation':'vertical', 'cbar_pad':0.05, 'cbar_frac':0.15, 'cbar_labelsize':None}
>
> Modify the values for specific keys to change the default behavior.

psdplot_default: If True, will use the following default parameters:

> **psdplot_default={'lmstyle':'-',** 'linewidth':None, 'color': sns.xkcd_rgb["denim blue"], 'ar1_lmstyle':'-', 'ar1_linewidth':1, 'period_ticks':None, 'period_tickslabel':None, 'psd_lim':None, 'period_lim':None, 'figsize':[20,8], 'label':'PSD', 'plot_ar1':True, 'psd_ar1_q95':psd_ar1_q95, 'title': None, 'psd_ar1_color':sns.xkcd_rgb["pale red"], 'ax':None, 'vertical':False, 'plot_gridlines':True, 'period_label':'Period ('+ageunits+')', 'psd_label':'Spectral Density', 'zorder' : None}
>
> Modify the values for specific keys to change the default behavior.

fig (bool): If True, plots the figure saveFig (bool): default is to not save the figure dir (str): the full path of the directory in which to save the figure.

> If not provided, creates a default folder called 'figures' in the LiPD working directory (lipd.path).

format (str): One of the file extensions supported by the active backend. Default is "eps". Most backend support png, pdf, ps, eps, and svg.

Returns: dict_out (dict): A dictionary of outputs.

For wwz:

- wwa (array): The weights wavelet amplitude
- AR1_q (array): AR1 simulations
- coi (array): cone of influence
- freqs (array): vector for frequencies
- tau (array): the evenly-spaced time points, namely the time

shift for wavelet analysis.

- Neffs (array): The matrix of effective number of points in the

time-scale coordinates.

- coeff (array): The wavelet transform coefficients

For psd:

- psd (array): power spectral density
- freqs (array): vector of frequency

- psd_ar1_q95 (array): the 95% quantile of the psds of AR1 processes

fig: The figure

References: Foster, G. (1996). Wavelets for period analysis of unevenly sampled time series. The Astronomical Journal, 112(4), 1709-1729.

Examples: To run both wwz and psd:

```
>>> dict_out, fig = pyleoclim.wwzTs(wwz=True)
```

Note: This will return a single figure with wwa and psd

To change a default behavior:

```
>>> dict_out, fig = pyleoclim.wwzTs(psd_default = {'nMC':1000})
```

2.1.7 Age modelling

`pyleoclim.`**`Bchron`**(*lipd, modelNum=None, objectName=None, rejectAges=None, calCurves=None, reservoirAgeCorr=None, predictPositions='paleo', positionsThickness=None, outlierProbs=None, iterations=1000, burn=2000, thin=8, extractDate=-68, maxExtrap=500, thetaMhSd=0.5, muMhSd=0.1, psiMhSd=0.1, ageScaleVal=1000, positionScaleVal=100, saveLipd=True, plot=True, figsize=[4, 8], flipCoor=False, xlabel=None, ylabel=None, xlim=None, ylim=None, violinColor='#8B008B', medianLineColor='black', medianLineWidth=2.0, CIFillColor='Silver', samplePaths=True, samplePathNumber=10, alpha=0.5, saveFig=False, dir='', format='eps'*)

Runs Bchron and plot if asked

Fits a non-parametric chronology model to age/position data according to the Compound Poisson-Gamma model defined by Haslett and Parnell (2008). This version used a slightly modified Markov chain Monte-Carlo fitting algorithm which aims to converge quicker and requires fewer iterations. It also a slightly modified procedure for identifying outliers.

The Bchronology functions fits a compounf Poisson-Gamma distribution to the incrememnts between the dated levels. This involves a stochastic linear interpolation step where the age gaps are Gamma distributed, and the position gaps are Exponential. Radiocarbon and non-radiocarbon dates (including outliers) are updated within the fucntion also by MCMC.

This function also allows to save the ensemble, distributions, and probability tables as well as the parameters with which the model was run into the LiPD file.

Finally allows to make a plot.

Args:

lipd (dict): A dictionary containing the entry of a LiPD file. Can be obtained from lipd.readLipd() or pyleoclim.openLipd(). Please note that the Bchron function currently only allows for a single LiPD file (i.e., not the entire directory).

modelNum (int): The model number in which to place the Bchron output. If unknown, the function will try to make a guess and/or prompt based on the number of already available models.

objectName (str): The name of the chron object in which to store the new model (e.g. "chron0")

rejectAges (vector): A vector of 1/0 where 1 include the dates to be rejected. Default it None.

calCurves (list): (Optional) A vector of values containing either 'intcal13', 'marine13', 'shcal13', or 'normal'. If none is provided, will prompt the user. Should be either of length =1 if using the same calibration for each age or the same length as the vector of ages.

reservoirAgeCorr (array): (Optional) A list (matrix) of two floats that correspond to the DeltaR and DeltaR uncertainty. If already added to the ages and ages standard deviation, then enter [0,0] to bypass the prompt. Will only be applied if CalCurves is set to 'marine13'. Otherwise, leave to none.

predictPositions (array): (Optional) a vector of positions (e.g. depths) at which predicted age values are required. Defaults to a sequence of length 100 from the top position to the bottom position.

positionsThickness (array): (Optional) Thickness values for each of the positions. The thickness values should be the full thickness value of the slice. By default set to zero.

outlierProbs (array): (Optional) A vector of prior outlier probabilities, one for each age. Defaults to 0.01

iterations (int): (Optional) The number of iterations to start the procedure. Default and minimum should be 10000.

burn (int): (Optional) The number of starting iterations to discard. Default is 200

thin (int): (Optional) The step size for every iteration to keep beyond the burnin. Default is 8.

extractDate (float): (Optional) The top age of the core. Used for extrapolation purposes so that no extrapolated ages go beyond the top age of the core. Defaults to the current year.

maxExtrap (int): (Optional) The maximum number of extrapolations to perform before giving up and setting the predicted ages to NA. Useful for when large amounts of extrapolation are required, i.e. some of the predictPositions are a long way from the dated positions. Defaults to 500.

thetaMhSd (float): (Optional) The Metropolis-Hastings standard deviation for the age parameters. Defaults to 0.5.

muMhSd (float): (Optional) The Metropolis-Hastings standard deviation for the compound Poisson-Gamma Scale. Defaults to 0.1

psiMhSd (float): (Optional) The Metropolis-Hastings standard deviation for the Compound Poisson-Gamma Scale.

ageScaleVal (int): (Optional) A scale value for the ages. Bchronology works best when the ages are scaled to be approximately between 0 and 100. The default value is thus 1000 for ages given in years.

positionScaleVal (int): (Optional) A scale value for the positions. Bchronology works best when the positions are scaled to be approximately between 0 and 100. The default value is thus 100 for positions given in cm.

saveLipd (bool): If True, saves the ensemble, distribution, and probability tables along with the parameters used to run the model in the LiPD file.

plot (bool): If True, makes a plot for the chronology figsize (list): The figure size. Default is [4,8] flipCoor (bool): If True, plots depth on the y-axis. xlabel (str): The label for the x-axis ylabel (str): The label for the y-axis xlim (list): Limits for the x-axis. Default corresponds to the min/max

of the depth vector.

ylim (list): Limits for the y-axis. Default set by matplotlib violinColor (str): The color for the violins. Default is purple medianLineColor (str): The color for the median line. Default is black. medianLineWidth (float): The width for the median line CIFillColor (str): Fill color in between the 95% confidence interval.

Default is silver.

samplePaths (bool): If True, draws sample paths from the distribution. Use the same color as the violins.

samplePathNumber (int): The number of sample paths to draw. Default is 10. Note: samplePaths need to be set to True.

alpha (float): The violins' transparency. Number between 0 and 1 saveFig (bool): default is to not save the figure dir (str): the full path of the directory in which to save the figure.

If not provided, creates a default folder called 'figures' in the LiPD working directory (lipd.path).

format (str): One of the file extensions supported by the active backend. Default is "eps". Most backend support png, pdf, ps, eps, and svg.

Returns: depth - the predicted positions (either same as the user or the default)

chron - a numpy array of possible chronologies in each column. The number of rows is the same as the length of depth

ageDist - the distribution of ages around each dates. fig - the figure

Warnings:

This function requires R and the Bchron package and all its dependencies to be installed on the same machine.

Reference:

- **Haslett, J., and Parnell, A. C. (2008). A simple monotone** process with application to radiocarbon-dated depth chronologies. Journal of the Royal Statistical Society, Series C, 57, 399-418. DOI:10.1111/j.1467-9876.2008.00623.x
- **Parnell, A. C., Haslett, J., Allen, J. R. M., Buck, C. E.,** and Huntley, B. (2008). A flexible approach to assessing synchroneity of past events using Bayesian reconstructions of sedimentation history. Quaternary Science Reviews, 27(19-20), 1872-1885. DOI:10.1016/j.quascirev.2008.07.009

2.2 Using Pyleoclim without a LiPD file

The Pyleoclim modules can be called separately so the main functions can be used without a timeseries objects. The following modules are available:

- pyleoclim.Map: mapping functions
- pyleoclim.Plot: plotting functions
- pyleoclim.Stats: statistics (including correlation)
- pyleoclim.Timeseries: binning, interpolating
- pyleoclim.Spectral: analysis in the frequency domain
- pyleoclim.RBchron: Bchron age model analysis

In addition, the pyleoclim.LipdUtils module allows the basic manipulation of LiPD files and the pyleoclim.SummaryPlots module contains some functions to extract information about a timeseries object.

CHAPTER THREE

MAPPING FUNCTIONS

This module uses the Basemap package for mapping.

`pyleoclim.Map.`**`mapAll`**`(`*lat, lon, criteria, projection='robin', lat_0='', lon_0='', llcrnrlat=-90, urcrnrlat=90, llcrnrlon=-180, urcrnrlon=180, countries=False, counties=False, rivers=False, states=False, figsize=[10, 4], ax=None, background='none', scale=0.5, palette='', markersize=50*`)`

Map the location of all lat/lon according to some criteria

Map the location of all lat/lon according to some criteria. The choice of plotting color/marker is passed through palette according to unique criteria (e.g., record name, archive type, proxy observation type).

Args: lat (list): a list of latitude. lon (list): a list of longitude. criteria (list): a list of criteria for plotting purposes. For instance,

a map by the types of archive present in the dataset or proxy observations.

projection (string): the map projection. Refers to the Basemap documentation for a list of available projections. Only projections supporting setting the map center with a single lat/lon or with the coordinates of the rectangle are currently supported. Default is to use a Robinson projection.

lat_0, lon_0 (float): the center coordinates for the map. Default is mean latitude/longitude in the list. If the chosen projection doesn't support it, Basemap will ignore the given values.

llcrnrlat, urcrnrlat, llcrnrlon, urcrnrlon (float): The coordinates of the two opposite corners of the rectangle.

countries (bool): Draws the countries border. Defaults is off (False). counties (bool): Draws the USA counties. Default is off (False). rivers (bool): Draws the rivers. Default is off (False). states (bool): Draws the American and Australian states borders.

Default is off (False).

background (string): Plots one of the following images on the map: bluemarble, etopo, shadedrelief, or none (filled continents). Default is none.

scale (float): Useful to downgrade the original image resolution to speed up the process. Default is 0.5.

palette (dict): A dictionary of plotting color/marker by criteria. The keys should correspond to ***unique*** criteria with a list of associated values. The list should be in the format ['color', 'marker'].

markersize (int): The size of the marker. figsize (list): the size for the figure ax: Return as axis instead of figure (useful to integrate plot into a subplot)

Returns: The figure

`pyleoclim.Map.`**`mapOne`**(*lat, lon, projection='ortho', lat_0='', lon_0='', llcrnrlat=-90, urcrnrlat=90, llcrnrlon=-180, urcrnrlon=180, countries=True, counties=False, rivers=False, states=False, background='shadedrelief', scale=0.5, markersize=50, marker='ro', figsize=[4, 4], ax=None*)

Map one location on the globe

Args: lat (float): a float number representing latitude lon (float): a float number representing longitude projection (string): the map projection. Refers to the Basemap

documentation for a list of available projections. Only projections supporting setting the map center with a single lat/lon or with the coordinates of the rectangle are currently supported. Default is to use a Robinson projection.

lat_0, lon_0 (float): the center coordinates for the map. Default is mean latitude/longitude in the list. If the chosen projection doesn't support it, Basemap will ignore the given values.

llcrnrlat, urcrnrlat, llcrnrlon, urcrnrlon (float): The coordinates of the two opposite corners of the rectangle.

countries (bool): Draws the countries border. Defaults is off (False). counties (bool): Draws the USA counties. Default is off (False). rivers (bool): Draws the rivers. Default is off (False). states (bool): Draws the American and Australian states borders.

Default is off (False).

background (string): Plots one of the following images on the map: bluemarble, etopo, shadedrelief, or none (filled continents). Default is none.

scale (float): Useful to downgrade the original image resolution to speed up the process. Default is 0.5.

markersize (int): The size of the marker. marker (str or list): color and type of marker. figsize (list): the size for the figure ax: Return as axis instead of figure (useful to integrate plot into a subplot)

CHAPTER

FOUR

PLOTTING FUNCTIONS

`pyleoclim.Plot.`**`plot`**(*x, y, markersize=50, marker='ro', x_label='', y_label='', title='', figsize=[10, 4], ax=None*)

Make a 2-D plot

Args: x (numpy array): a 1xn numpy array of values for the x-axis y (numpy array): a 1xn numpy array for the y-axis markersize (int): the size of the marker marker (string or list): color and shape of the marker x_axis_label (str): the label for the x-axis y_axis_label (str): the label for the y-axis title (str): the title for the plot figsize (list): the size of the figure ax: Return as axis instead of figure (useful to integrate plot into a subplot)

Return: The figure

`pyleoclim.Plot.`**`plot_hist`**(*y, bins=None, hist=True, label='', kde=True, rug=False, fit=None, hist_kws={'label': 'Histogram'}, kde_kws={'label': 'KDE fit'}, rug_kws={'label': 'rug'}, fit_kws={'label': 'fit'}, color='0.7', vertical=False, norm_hist=True, figsize=[5, 5], ax=None*)

Plot a univariate distribution of the PaleoData values

This function is based on the seaborn displot function, which is itself a combination of the matplotlib hist function with the seaborn kdeplot() and rugplot() functions. It can also fit scipy.stats distributions and plot the estimated PDF over the data.

Args: y (array): nx1 numpy array. No missing values allowed bins (int): Specification of hist bins following matplotlib(hist),

or None to use Freedman-Diaconis rule

hist (bool): Whether to plot a (normed) histogram label (str): The label for the axis kde (bool): Whether to plot a gaussian kernel density estimate rug (bool): Whether to draw a rugplot on the support axis fit: Random variable object. An object with fit method, returning

a tuple that can be passed to a pdf method of positional arguments following a grid of values to evaluate the pdf on.

{hist, kde, rug, fit}_kws: Dictionaries. Keyword arguments for underlying plotting functions. If modifying the dictionary, make sure the labels "hist", "kde", "rug" and "fit" are still passed.

color (str): matplotlib color. Color to plot everything but the fitted curve in.

vertical (bool): if True, oberved values are on y-axis. norm_hist (bool): If True (default), the histrogram height shows

a density rather than a count. This is implied if a KDE or fitted density is plotted

figsize (list): the size of the figure ax: Return as axis instead of figure (useful to integrate plot into a subplot)

Returns fig - The figure

CHAPTER

FIVE

STATISTICS FUNCTIONS

`pyleoclim.Stats.`**`simpleStats`**(*y*, *axis=None*)

Computes simple statistics

Computes the mean, median, min, max, standard deviation, and interquartile range of a numpy array y.

Args: y (array): A Numpy array axis (int, typle of ints): Optional. Axis or Axes along which the means

are computed, the default is to compute the mean of the flattened array. If a tuple of ints, performed over multiple axes

Returns: The mean, median, min, max, standard deviation and IQR by columns

`pyleoclim.Stats.`**`corrsig`**(*y1*, *y2*, *nsim=1000*, *method='isospectral'*, *alpha=0.5*)

Estimates the significance of correlations between non IID time series by 3 independent methods:

1. 'ttest': T-test where d.o.f are corrected for the effect of serial correlation
2. 'isopersistent': AR(1) modeling of x and y.

3) 'isospectral': phase randomization of original inputs. (default) The T-test is parametric test, hence cheap but usually wrong except in idyllic circumstances. The others are non-parametric, but their computational requirements scales with nsim.

Args: y1, y2 (array)- vector of (real) numbers of identical length, no NaNs allowed nsim (int)- the number of simulations [1000] method (str)- methods 1-3 above ['isospectral'] alpha (float)- significance level for critical value estimation [0.05]

Returns: r (real): correlation between x and y

signif (int): true if significant; false otherwise

p (real): Fraction of time series with higher correlation coefficents than observed (approximates the p-value).

Note that signif = True if and only if p <= alpha.

CHAPTER

SIX

TIMESERIES FUNCTIONS

This module allows the manipulation of timeseries.

pyleoclim.Timeseries.**bin**(*x*, *y*, *bin_size=''*, *start=''*, *end=''*)

Bin the values

Args: x (array): the x-axis series. y (array): the y-axis series. bin_size (float): The size of the bins. Default is the average resolution start (float): Where/when to start binning. Default is the minimum end (float): When/where to stop binning. Defulat is the maximum

Returns: binned_values - the binned output

bins - the bins (centered on the median, i.e., the 100-200 bin is 150)

n - number of data points in each bin

error - the standard error on the mean in each bin

pyleoclim.Timeseries.**interp**(*x*, *y*, *interp_step=''*, *start=''*, *end=''*)

Linear interpolation onto a new x-axis

Args: x (array): the x-axis y (array): the y-axis interp_step (float): the interpolation step. Default is mean resolution. start (float): where/when to start the interpolation. Default is min.. end (float): where/when to stop the interpolation. Default is max.

Returns: xi - the interpolated x-axis

interp_values - the interpolated values

pyleoclim.Timeseries.**onCommonAxis**(*x1*, *y1*, *x2*, *y2*, *interp_step=''*, *start=''*, *end=''*)

Places two timeseries on a common axis

Args: x1 (array): x-axis values of the first timeseries y1 (array): y-axis values of the first timeseries x2 (array): x-axis values of the second timeseries y2 (array): y-axis values of the second timeseries interp_step (float): The interpolation step. Default is mean resolution of lowest resolution series start (float): where/when to start. Default is the maximum of the minima of the two timeseries end (float): Where/when to end. Default is the minimum of the maxima of the two timeseries

Returns: xi - the interpolated x-axis

interp_values1 - the interpolated y-values for the first timeseries interp_values2 - the intespolated y-values for the second timeseries

pyleoclim.Timeseries.**standardize**(*x*, *scale=1*, *axis=0*, *ddof=0*, *eps=0.001*)

Centers and normalizes a given time series. Constant or nearly constant time series not rescaled.

Args: x (array): vector of (real) numbers as a time series, NaNs allowed scale (real): a scale factor used to scale a record to a match a given variance axis (int or None): axis along which to operate, if None, compute over the whole array ddof (int): degress of freedom correction in the calculation of the standard deviation eps (real): a threshold to determine if the standard deviation is too close to zero

Returns: z (array): the standardized time series (z-score), Z = (X - E[X])/std(X)*scale, NaNs allowed mu (real): the mean of the original time series, E[X] sig (real): the standard deviation of the original time series, std[X]

References:

1. Tapio Schneider's MATLAB code: http://www.clidyn.ethz.ch/imputation/standardize.m
2. The zscore function in SciPy: https://github.com/scipy/scipy/blob/master/scipy/stats/stats.py

@author: fzhu

`pyleoclim.Timeseries.`**`ts2segments`**(*ys*, *ts*, *factor=10*)

Chop a time series into several segments based on gap detection.

The rule of gap detection is very simple: we define the intervals between time points as dts, then if dts[i] is larger than factor * dts[i-1], we think that the change of dts (or the gradient) is too large, and we regard it as a breaking point and chop the time series into two segments here

Args: ys (array): a time series, NaNs allowed ts (array): the time points factor (float): the factor that adjusts the threshold for gap detection

Returns: seg_ys (list): a list of several segments with potentially different lengths seg_ts (list): a list of the time axis of the several segments n_segs (int): the number of segments

@author: fzhu

`pyleoclim.Timeseries.`**`clean_ts`**(*ys*, *ts*)

Delete the NaNs in the time series and sort it with time axis ascending

Args: ys (array): a time series, NaNs allowed ts (array): the time axis of the time series, NaNs allowed

Returns: ys (array): the time series without nans ts (array): the time axis of the time series without nans

`pyleoclim.Timeseries.`**`gaussianize`**(*X*)

Transforms a (proxy) timeseries to Gaussian distribution.

Originator: Michael Erb, Univ. of Southern California - April 2017

`pyleoclim.Timeseries.`**`gaussianize_single`**(*X_single*)

Transforms a single (proxy) timeseries to Gaussian distribution.

Originator: Michael Erb, Univ. of Southern California - April 2017

`pyleoclim.Timeseries.`**`detrend`**(*y, x=None, method='linear', params=['default', 4, 0, 1]*)

Detrend a timeseries according to three methods

Detrending methods include, "linear" (default), "constant", and using a low-pass Savitzky-Golay filters.

Args: y (array): The series to be detrended. x (array): The time axis for the timeseries. Necessary for use with

the Savitzky-Golay filters method since the series should be evenly spaced.

method (str): The type of detrending. If linear (default), the result of a linear least-squares fit to y is subtracted from y. If constant, only the mean of data is subtrated. If "savitzy-golay", y is filtered using the Savitzky-Golay filters and the resulting filtered series is subtracted from y.

params (list): The paramters for the Savitzky-Golay filters. The first parameter corresponds to the window size (default it set to half of the data) while the second parameter correspond to the order of the filter (default is 4). The third parameter is the order of the derivative (the default is zero, which means only smoothing.)

Returns: ys (array) - the detrended timeseries.

CHAPTER

SEVEN

LIPD UTILITIES

This modules allow basic manipulation of LiPD files

7.1 Creating Directories and saving

`pyleoclim.LipdUtils.`**`createDir`**`(`*path*, *foldername*`)`

Create a new folder in a working directory

Create a new folder in a working directory to save outputs from Pyleoclim.

Args: path(str): the path to the new folder. foldername(str): the name of the folder to be created

Returns: newdir - the full path to the new directory

`pyleoclim.LipdUtils.`**`saveFigure`**`(`*name*, *format='eps'*, *dir=''*`)`

Save a figure

Save the figure in the directory. If not given, creates a folder in the current working directory.

Args: name (str): name of the file format (str): One of the file extensions supported by the active

backend. Default is "eps". Most backend support png, pdf, ps, eps, and svg.

dir (str): the name of the folder in the LiPD working directory. If not provided, creates a default folder called 'figures'.

7.2 LiPD files

`pyleoclim.LipdUtils.`**`enumerateLipds`**`(`*lipds*`)`

Enumerate the LiPD files loaded in the workspace

Args:

lipds (dict): A dictionary of LiPD files. Can be obtained from pyleoclim.readLipd()

`pyleoclim.LipdUtils.`**`getLipd`**`(`*lipds*`)`

Prompt for a LiPD file

Ask the user to select a LiPD file from a list Use this function in conjunction with enumerateLipds()

Args:

lipds (dict): A dictionary of LiPD files. Can be obtained from pyleoclim.readLipd()

Returns: The index of the LiPD file

7.3 Handling Variables

`pyleoclim.LipdUtils.`**`promptForVariable`**`()`

Prompt for a specific variable

Ask the user to select the variable they are interested in. Use this function in conjunction with readHeaders() or getTSO()

Returns: The index of the variable

`pyleoclim.LipdUtils.`**`xAxisTs`**`(`*timeseries*`)`

Prompt the user to choose a x-axis representation for the timeseries.

Args: timeseries: a timeseries object

Returns: x_axis - the values for the x-axis representation,

label - returns either "age", "year", or "depth"

`pyleoclim.LipdUtils.`**`checkXaxis`**`(`*timeseries*, *x_axis=''*`)`

Check that a x-axis is present for the timeseries

Args: timeseries : a timeseries x_axis (str) : the x-axis representation, either depth, age or year

Returns: x - the values for the x-axis representation,

label - returns either "age", "year", or "depth"

`pyleoclim.LipdUtils.`**`searchVar`**`(`*timeseries_list*, *key*, *exact=True*, *override=True*`)`

This function search for key words (exact match) for a variable

Args: timeseries_list (list): A list of available series key (list): A list of keys to search exact (bool): if True, looks for an exact match. override (bool): if True, override the exact match if no match is found

Returns:

match (list)- A list of keys for the timeseries that match the selection criteria.

7.4 Handling timeseries objects

`pyleoclim.LipdUtils.`**`enumerateTs`**`(`*timeseries_list*`)`

Enumerate the available time series objects

Args:

timeseries_list: a list of available timeseries objects. To use the timeseries loaded upon initiation of the pyleoclim package, use pyleo.time_series.

`pyleoclim.LipdUtils.`**`getTs`**`(`*timeseries_list*, *option=''*`)`

Get a specific timeseries object from a dictionary of timeseries

Args:

timeseries_list: a list of available timeseries objects. To use the timeseries loaded upon initiation of the pyleoclim package, use pyleo.time_series.

option: An expression to filter the datasets. Uses lipd.filterTs()

Returns: A single timeseries object if not optional filter selected or a filtered list if optional arguments given

7.5 Linking LiPDs to the LinkedEarth Ontology

`pyleoclim.LipdUtils.`**`LipdToOntology`**`(`*archiveType*`)`

standardize archiveType

Transform the archiveType from their LiPD name to their ontology counterpart

Args: archiveType (STR): name of the archiveType from the LiPD file

Returns: archiveType according to the ontology

7.6 Dealing with models

`pyleoclim.LipdUtils.`**`isModel`**`(`*csvName*, *lipd*`)`

Check for the presence of a model in the same object than the measurement table

Args: csvName (str): The name of the csv file corresponding to the measurement table lipd (dict): A LiPD object

Returns: model (list): List of models already available

dataObject (str): The name of the paleoData or ChronData object in which the model(s) are stored

`pyleoclim.LipdUtils.`**`modelNumber`**`(`*model*`)`

Assign a new or existing model number

Args: model (list): List of possible model number. Obtained from isModel

Returns: modelNum (int): The number of the model

7.7 Extracting tables

`pyleoclim.LipdUtils.`**`isMeasurement`**`(`*csv_dict*`)`

Check whether measurement tables are available

Args: csv_dict (dict): Dictionary of available csv

Returns: paleoMeasurementTables - List of available paleoMeasurementTables chronMeasurementTables - List of available chronMeasurementTables

`pyleoclim.LipdUtils.`**`whichMeasurement`**`(`*measurementTableList*, *csv_dict*`)`

Select a measurement table from a list

Use in conjunction with the function isMeasurement

Args:

measurementTableList (list): List of measurement tables contained in the LiPD file. Output from the isMeasurement function

csv_list (list): Dictionary of available csv

Returns: csvName (str) - the name of the csv file

`pyleoclim.LipdUtils.`**`getMeasurement`**`(`*csvName*, *lipd*`)`

Extract the dictionary corresponding to the measurement table

Args: csvName (str): The name of the csv file lipd (dict): The LiPD object from which to extract the data

Returns:

ts_list - A dictionary containing data and metadata for each column in the csv file.

7.8 Dealing with ensembles

`pyleoclim.LipdUtils.`**`isEnsemble`**(*csv_dict*)

Check whether ensembles are available

Args: csv_dict (dict): Dictionary of available csv

Returns: paleoEnsembleTables - List of available paleoEnsembleTables

chronEnsembleTables - List of availale chronEnsemble Tables

`pyleoclim.LipdUtils.`**`mapAgeEnsembleToPaleoData`**(*ensembleValues*, *depthEnsemble*, *depthPaleo*)

Map the depth for the ensemble age values to the paleo depth

Args:

ensembleValues (array): A matrix of possible age models. Realizations should be stored in columns

depthEnsemble (array): A vector of depth. The vector should have the same length as the number of rows in the ensembleValues

depthPaleo (array): A vector corresponding to the depth at which there are paleodata information

Returns: ensembleValuesToPaleo - A matrix of age ensemble on the PaleoData scale

CHAPTER

EIGHT

SUMMARY PLOTS

This module handles some basic return for the summary plots.

Requires a LiPD file

`pyleoclim.SummaryPlots.`**`getMetadata`**`(`*timeseries*`)`

Get the necessary metadata to be printed out automatically

Args: timeseries: a specific timeseries object.

Returns: A dictionary containing the following metadata:

archiveType

Authors (if more than 2, replace by et al.

PublicationYear

Publication DOI

Variable Name

Units

Climate Interpretation

Calibration Equation

Calibration References

Calibration Notes

`pyleoclim.SummaryPlots.`**`TsData`**`(`*timeseries*, *x_axis=''*`)`

Get the PaleoData with age/depth information

Get the necessary information for the TS plots/necessary to allow for axes specification

Args:

timeseries: a single timeseries object. By default, will prompt the user

x-axis (str): The representation against which to plot the paleo-data. Options are “age”, “year”, and “depth”. Default is to let the system choose if only one available or prompt the user.

Returns: x - the x-valus

y - the y-values

archiveType - the archiveType (for plot settings)

x_label - the label for the x-axis

y_label - the label for the y-axis

label - the results of the x-axis query. Either depth, year, or age

`pyleoclim.SummaryPlots.`**`agemodelData`**`(`*timeseries*`)`

Get the necessary information for the agemodel plot

Args:

timeseries: a single timeseries object. By default, will prompt the user

Returns: depth - the depth values

age - the age values

x_label - the label for the x-axis

y_label - the label for the y-axis

archiveType - the archiveType (for default plot settings)

CHAPTER

NINE

SPECTRAL FUNCTIONS

This module allows analysis in the frequency domain

`pyleoclim.Spectral.`**`ar1_fit`**(*ys, ts=None, detrend='no', params=['default', 4, 0, 1]*)

Returns the lag-1 autocorrelation from ar1 fit OR persistence from tauest.

Args: ys (array): the time series ts (array): the time axis of that series detrend (str): 'no' - the original time series is assumed to have no trend;

'linear' - a linear least-squares fit to *ys* is subtracted; 'constant' - the mean of *ys* is subtracted 'savitzy-golay' - ys is filtered using the Savitzky-Golay

filters and the resulting filtered series is subtracted from y.

params (list): The paramters for the Savitzky-Golay filters. The first parameter corresponds to the window size (default it set to half of the data) while the second parameter correspond to the order of the filter (default is 4). The third parameter is the order of the derivative (the default is zero, which means only smoothing.)

Returns: g (float): lag-1 autocorrelation coefficient (for evenly-spaced time series) OR estimated persistence (for unevenly-spaced time series)

`pyleoclim.Spectral.`**`ar1_sim`**(*ys, n, p, ts=None, detrend='no', params=['default', 4, 0, 1]*)

Produce p realizations of an AR1 process of length n with lag-1 autocorrelation g calculated from *ys* and *ts*

Args: ys (array): a time series n, p (int): dimensions as n rows by p columns ts (array): the time axis of that series detrend (str): 'no' - the original time series is assumed to have no trend;

'linear' - a linear least-squares fit to *ys* is subtracted; 'constant' - the mean of *ys* is subtracted 'savitzy-golay' - ys is filtered using the Savitzky-Golay

filters and the resulting filtered series is subtracted from y.

params (list): The paramters for the Savitzky-Golay filters. The first parameter corresponds to the window size (default it set to half of the data) while the second parameter correspond to the order of the filter (default is 4). The third parameter is the order of the derivative (the default is zero, which means only smoothing.)

Returns: red (matrix): n rows by p columns matrix of an AR1 process

`pyleoclim.Spectral.`**`wwz`**(*ys, ts, tau=None, freqs=None, c=0.012665147955292222, Neff=3, Neff_coi=3, nMC=200, nproc=8, detrend='no', params=['default', 4, 0, 1], gaussianize=False, standardize=True, method='Kirchner_f2py', len_bd=0, bc_mode='reflect', reflect_type='odd'*)

Return the weighted wavelet amplitude (WWA) with phase, AR1_q, and cone of influence, as well as WT coeeficients

Args: ys (array): a time series, NaNs will be deleted automatically ts (array): the time points, if *ys* contains any NaNs, some of the time points will be deleted accordingly tau (array): the evenly-spaced time points freqs (array): vector of frequency c (float): the decay constant, the default value 1/(8*np.pi**2) is good for most of the cases Neff (int): effective number of points nMC (int): the number of Monte-Carlo simulations nproc (int): the number of processes for multiprocessing detrend (str): 'no' - the original time series is assumed to have no trend;

'linear' - a linear least-squares fit to *ys* is subtracted; 'constant' - the mean of *ys* is subtracted 'savitzy-golay' - ys is filtered using the Savitzky-Golay

filters and the resulting filtered series is subtracted from y.

params (list): The paramters for the Savitzky-Golay filters. The first parameter corresponds to the window size (default it set to half of the data) while the second parameter correspond to the order of the filter (default is 4). The third parameter is the order of the derivative (the default is zero, which means only smoothing.)

method (str): 'Foster' - the original WWZ method; 'Kirchner' - the method Kirchner adapted from Foster; 'Kirchner_f2py' - the method Kirchner adapted from Foster with f2py

len_bd (int): the number of the ghost grids want to creat on each boundary bc_mode (str): see np.lib.pad() reflect_type (str): see np.lib.pad()

Returns: wwa (array): the weighted wavelet amplitude. AR1_q (array): AR1 simulations coi (array): cone of influence freqs (array): vector of frequency tau (array): the evenly-spaced time points, namely the time shift for wavelet analysis Neffs (array): the matrix of effective number of points in the time-scale coordinates coeff (array): the wavelet transform coefficents

`pyleoclim.Spectral.`**`wwz_psd`**(*ys, ts, freqs=None, tau=None, c=0.001, nproc=8, nMC=200, detrend='no', params=['default', 4, 0, 1], gaussianize=False, standardize=True, Neff=3, anti_alias=False, avgs=1, method='Kirchner_f2py'*)

Return the psd of a timeseries directly using wwz method.

Args: ys (array): a time series, NaNs will be deleted automatically ts (array): the time points, if *ys* contains any NaNs, some of the time points will be deleted accordingly freqs (array): vector of frequency tau (array): the evenly-spaced time points, namely the time shift for wavelet analysis c (float): the decay constant, the default value 1e-3 is good for most of the cases nproc (int): the number of processes for multiprocessing nMC (int): the number of Monte-Carlo simulations detrend (str): 'no' - the original time series is assumed to have no trend;

'linear' - a linear least-squares fit to *ys* is subtracted; 'constant' - the mean of *ys* is subtracted 'savitzy-golay' - ys is filtered using the Savitzky-Golay

filters and the resulting filtered series is subtracted from y.

params (list): The paramters for the Savitzky-Golay filters. The first parameter corresponds to the window size (default it set to half of the data) while the second parameter correspond to the order of the filter (default is 4). The third parameter is the order of the derivative (the default is zero, which means only smoothing.)

gaussionize (bool): If True, gaussianizes the timeseries standardize (bool): If True, standardizes the timeseries method (str): 'Foster' - the original WWZ method;

'Kirchner' - the method Kirchner adapted from Foster; 'Kirchner_f2py' - the method Kirchner adapted from Foster with f2py

Returns: psd (array): power spectral density freqs (array): vector of frequency psd_ar1_q95 (array): the 95% quantile of the psds of AR1 processes psd_ar1 (array): the psds of AR1 processes pyleoclim.Spectral.**xwc**(*ys1, ts1, ys2, ts2, tau=None, freqs=None, c=0.012665147955292222, Neff=3, nproc=8, detrend='no', nMC=200, params=['default', 4, 0, 1], gaussianize=False, standardize=True, method='Kirchner_f2py'*)

Return the crosse wavelet coherence of two time series.

Args: ys1, ys2 (array): the two time series ts1, ts2 (array): the time axis of the two time series tau (array): the evenly-spaced time points freqs (array): vector of frequency c (float): the decay constant, the default value 1/(8*np.pi**2) is good for most of the cases Neff (int): effective number of points nproc (int): the number of processes for multiprocessing nMC (int): the number of Monte-Carlo simulations detrend (str): 'no' - the original time series is assumed to have no trend;

'linear' - a linear least-squares fit to *ys* is subtracted; 'constant' - the mean of *ys* is subtracted 'savitzy-golay' - ys is filtered using the Savitzky-Golay

filters and the resulting filtered series is subtracted from y.

params (list): The paramters for the Savitzky-Golay filters. The first parameter corresponds to the window size (default it set to half of the data) while the second parameter correspond to the order of the filter (default is 4). The third parameter is the order of the derivative (the default is zero, which means only smoothing.)

gaussionize (bool): If True, gaussianizes the timeseries standardize (bool): If True, standardizes the timeseries method (str): 'Foster' - the original WWZ method;

'Kirchner' - the method Kirchner adapted from Foster; 'Kirchner_f2py' - the method Kirchner adapted from Foster with f2py

Returns: xw_coherence (array): the cross wavelet coherence xw_phase (array): the cross wavelet phase freqs (array): vector of frequency tau (array): the evenly-spaced time points AR1_q (array): AR1 simulations coi (array): cone of influence

pyleoclim.Spectral.**plot_wwa**(*wwa, freqs, tau, AR1_q=None, coi=None, levels=None, tick_range=None, yticks=None, yticks_label=None, ylim=None, xticks=None, xlabels=None, figsize=[20, 8], clr_map='OrRd', cbar_drawedges=False, cone_alpha=0.5, plot_signif=False, signif_style='contour', title=None, plot_cone=False, ax=None, xlabel='Year', ylabel='Period', cbar_orientation='vertical', cbar_pad=0.05, cbar_frac=0.15, cbar_labelsize=None*)

Plot the wavelet amplitude

Args: wwa (array): the weighted wavelet amplitude. freqs (array): vector of frequency tau (array): the evenly-spaced time points, namely the time shift for wavelet analysis AR1_q (array): AR1 simulations coi (array): cone of influence levels (array): levels of values to plot tick_range (array): levels of ticks to show on the colorbar yticks (list): ticks on y-axis ylim (list): limitations for y-axis xticks (list): ticks on x-axis figsize (list): the size for the figure clr_map (str): the name of the colormap cbar_drawedges (bool): whether to draw edges on the colorbar or not cone_alpha (float): the alpha value for the area covered by cone of influence plot_signif (bool): plot 95% significant area or not signif_style (str): plot 95% significant area with *contour* or *shade* plot_cone (bool): plot cone of influence or not ax: Return as axis instead of figure (useful to integrate plot into a subplot) xlabel (str): The x-axis label ylabel (str): The y-axis label cbar_pad (float): the pad for the colorbar cbar_frac (float): the frac for the colorbar cbar_labelsize (float): the font size of the colorbar label

Returns: fig (figure): the 2-D plot of wavelet analysis `pyleoclim.Spectral.`**`plot_coherence`**(*xw_coherence, xw_phase, freqs, tau, AR1_q=None, coi=None, levels=None, tick_range=None, basey=2, yticks=None, ylim=None, xticks=None, xlabels=None, figsize=[20, 8], clr_map='OrRd', exg=5, scale=30, width=0.004, cbar_drawedges=False, cone_alpha=0.5, plot_signif=False, signif_style='contour', title=None, plot_cone=False, ax=None, xlabel='Year', ylabel='Period', cbar_orientation='vertical', cbar_pad=0.05, cbar_frac=0.15, cbar_labelsize=None*)

Plot the wavelet amplitude

Args: xw_coherence (array): the wavelet cohernce xw_phase (array): the wavelet cohernce phase freqs (array): vector of frequency tau (array): the evenly-spaced time points, namely the time shift for wavelet analysis AR1_q (array): AR1 simulations coi (array): cone of influence levels (array): levels of values to plot tick_range (array): levels of ticks to show on the colorbar yticks (list): ticks on y-axis ylim (list): limitations for y-axis xticks (list): ticks on x-axis figsize (list): the size for the figure clr_map (str): the name of the colormap cbar_drawedges (bool): whether to draw edges on the colorbar or not cone_alpha (float): the alpha value for the area covered by cone of influence plot_signif (bool): plot 95% significant area or not signif_style (str): plot 95% significant area with *contour* or *shade* plot_cone (bool): plot cone of influence or not ax: Return as axis instead of figure (useful to integrate plot into a subplot) xlabel (str): The x-axis label ylabel (str): The y-axis label cbar_pad (float): the pad for the colorbar c)bar_frac (float): the frac for the colorbar cbar_labelsize (float): the font size of the colorbar label

Returns: fig (figure): the 2-D plot of wavelet analysis

`pyleoclim.Spectral.`**`plot_wwadist`**(*wwa*, *ylim=None*)

Plot the distribution of wwa with the 95% quantile line.

Args: wwa (array): the weighted wavelet amplitude. ylim (list): limitations for y-axis

Returns: fig (figure): the 2-D plot of wavelet analysis

`pyleoclim.Spectral.`**`plot_psd`**(*psd, freqs, lmstyle='-', linewidth=None, color='#3b5b92', ar1_lmstyle='-', ar1_linewidth=None, period_ticks=None, period_tickslabel=None, psd_lim=None, period_lim=None, figsize=[20, 8], label='PSD', plot_ar1=False, psd_ar1_q95=None, title=None, legend=True, psd_ar1_color='#d9544d', ax=None, vertical=False, plot_gridlines=True, period_label='Period (years)', psd_label='Spectral Density', zorder=None*)

Plot the wavelet amplitude

Args: psd (array): power spectral density freqs (array): vector of frequency period_ticks (list): ticks for period psd_lim (list): limits for spectral density axis label (str): the label for the PSD plot_ar1 (bool): plot the ar1 curve or not psd_ar1_q95 (array): the 95% quantile of the AR1 PSD psd_ar1_color (str): the color for the 95% quantile of the AR1 PSD title (str): the title for the figure period_lim (list): limits for period axis figsize (list): the size for the figure ax (axis): Return as axis instead of figure (useful to integrate plot into a subplot) vertical (bool): plot in vertical layout or not legend (bool): plot legend or not lmstyle (str): the line style linewidth (float): the line width period_label (str): the label for period psd_label (str): the label for psd zorder (int): the order of the layer

Returns: ax (figure): the 2-D plot of wavelet analysis `pyleoclim.Spectral.`**`plot_summary`**(*ys, ts, freqs=None, tau=None, c1=0.012665147955292222, c2=0.001, nMC=200, nproc=8, detrend='no', gaussianize=False, standardize=True, levels=None, method='Kirchner_f2py', anti_alias=False, period_ticks=None, ts_color=None, title=None, ts_ylabel=None, wwa_xlabel=None, wwa_ylabel=None, psd_lmstyle='-', psd_lim=None, period_I=[0.125, 0.5], period_D=[0.005, 0.05]*)

Plot the time series with the wavelet analysis and psd

Args: ys (array): a time series ts (array): time axis of the time series freqs (array): vector of frequency tau (array): the evenly-spaced time points, namely the time shift for wavelet analysis c (float): the decay constant Neff (int): the threshold of the number of effective degree of freedom nproc (int): fake argument, just for convenience detrend (str): 'no' - the original time series is assumed to have no trend;

'linear' - a linear least-squares fit to *ys* is subtracted; 'constant' - the mean of *ys* is subtracted

ts_color (str): the color for the time series curve title (str): the title for the time series plot ts_ylabel (str): label for y-axis in the time series plot wwa_xlabel (str): label for x-axis in the wwa plot wwa_ylabel (str): label for y-axis in the wwa plot psd_lmstyle (str): the line style in the psd plot psd_lim (list): the limits for psd period_I, period_D (list): the ranges for beta estimation

Returns: fig (figure): the summary plot

CHAPTER

TEN

RBCHRON

This module helps in the preparation of a Bchron run.

When using this module, please cite: * Haslett, J., and Parnell, A.C. (2008). A simple monotone process with application to radiocarbon-dated depth chronologies. Journal of the Royal Statistical Society, Series C, 57, 399-418, DOI:10.111/j.1467-9876.2008.00623.x * Parnell, A. C., Haslett, J., Allen, J. R. M., Buck, C. E., and Huntley, B. (2008). A flexible approach to assessing synchroneity of past events using Bayesian reconstructions of sedimentation history. Quaternary Science Reviews, 27(19-20), 1872-1885. DOI:10.1016/j.quascirev.2008.07.009

`pyleoclim.RBchron.`**`chooseCalCurves`**`()`

Prompt for a calibration curve if not given by the user.

Prompt the user for the name of a calibration curve used to run the Bchron software package. The user can enter either enter only one name that will be applied to each age or a list of names of different ages. To enter a list, separate each name with a comma. No quotation marks needed.

Returns: A list of calibration curves to be applied

`pyleoclim.RBchron.`**`reservoirAgeCorrection`**`()`

Estimate reservoir age correction

Assists in estimating the reservoir age correction for marine records. If unknown, will direct the user to copy and paste the table available on the 14Chrono Marine Reservoir database: http://intcal.qub.ac.uk/marine/

Returns: ageCorr - the DeltaR for the site.

ageCorrStd - The error on DeltaR estimated as the standard error on the mean if using the 14Chrono Marine Reservoir database.

`pyleoclim.RBchron.`**`runBchron`**`(`*ages*, *agesStd*, *positions*, *rejectAges=None*, *positionsThickness=None*, *calCurves=None*, *reservoirAgeCorr=None*, *outlierProbs=None*, *predictPositions=None*, *iterations=10000*, *burn=2000*, *thin=8*, *extractDate=-68*, *maxExtrap=500*, *thetaMhSd=0.5*, *muMhSd=0.1*, *psiMhSd=0.1*, *ageScaleVal=1000*, *positionScaleVal=100*`)`

Age model for Tie-Point chronologies

Fits a non-parametric chronology model to age/position data according to the Compound Poisson-Gamma model defined by Haslett and Parnell (2008). This version used a slightly modified Markov chain Monte-Carlo fitting algorithm which aims to converge quicker and requires fewer iterations. It also a slightly modified procedure for identifying outliers.

The Bchronology functions fits a compounf Poisson-Gamma distribution to the incrememnts between the dated levels. This involves a stochastic linear interpolation step where the age gaps are Gamma distributed, and the position gaps are Exponential. Radiocarbon and non-radiocarbon dates (including outliers) are updated within the fucntion also by MCMC.

Args: ages (array): A vector of ages (most likely 14C) ageSds (array): A vector of 1-sigma values for the ages given above positions (array): Position values (e.g. depths) for each age rejectAges (vector): A vector of 1/0 where 1 include the dates to be rejected.

Default it None.

positionsThickness (array): (Optional) Thickness values for each of the positions. The thickness values should be the full thickness value of the slice. By default set to zero.

calCurves (list): (Optional) A vector of values containing either 'intcal13', 'marine13', 'shcal13', or 'normal'. If none is provided, will prompt the user. Should be either of length =1 if using the same calibration for each age or the same length as the vector of ages.

reservoirAgeCorr (array): (Optional) A list (matrix) of two floats that correspond to the DeltaR and DeltaR uncertainty. If already added to the ages and ages standard deviation, then enter [0,0] to bypass the prompt. Will only be applied if CalCurves is set to 'marine13'. Otherwise, leave to none.

outlierProbs (array): (Optional) A vector of prior outlier probabilities, one for each age. Defaults to 0.01

predictPositions (array): (Optional) a vector of positions (e.g. depths) at which predicted age values are required. Defaults to a sequence of length 100 from the top position to the bottom position.

iterations (int): (Optional) The number of iterations to start the procedure. Default and minimum should be 10000.

burn (int): (Optional) The number of starting iterations to discard. Default is 200

thin (int): (Optional) The step size for every iteration to keep beyond the burnin. Default is 8.

extractDate (float): (Optional) The top age of the core. Used for extrapolation purposes so that no extrapolated ages go beyond the top age of the core. Defaults to the current year.

maxExtrap (int): (Optional) The maximum number of extrapolations to perform before giving up and setting the predicted ages to NA. Useful for when large amounts of extrapolation are required, i.e. some of the predictPositions are a long way from the dated positions. Defaults to 500.

thetaMhSd (float): (Optional) The Metropolis-Hastings standard deviation for the age parameters. Defaults to 0.5.

muMhSd (float): (Optional) The Metropolis-Hastings standard deviation for the compound Poisson-Gamma Scale. Defaults to 0.1

psiMhSd (float): (Optional) The Metropolis-Hastings standard deviation for the Compound Poisson-Gamma Scale.

ageScaleVal (int): (Optional) A scale value for the ages. Bchronology works best when the ages are scaled to be approximately between 0 and 100. The default value is thus 1000 for ages given in years.

positionScaleVal (int): (Optional) A scale value for the positions. Bchronology works best when the positions are scaled to be approximately between 0 and 100. The default value is thus 100 for positions given in cm.

Returns: depth - the predicted positions (either same as the user or the default)

chron - a numpy array of possible chronologies in each column. The number of rows is the same as the length of depth

ageDist - the distribution of ages around each dates. run - the full R object containing the outputs of the Bchron run

Warnings:

This function requires R and the Bchron package and all its dependencies to be installed on the same machine.

Reference:

- **Haslett, J., and Parnell, A. C. (2008). A simple monotone** process with application to radiocarbon-dated depth chronologies. Journal of the Royal Statistical Society, Series C, 57, 399-418. DOI:10.1111/j.1467-9876.2008.00623.x
- **Parnell, A. C., Haslett, J., Allen, J. R. M., Buck, C. E.,** and Huntley, B. (2008). A flexible approach to assessing synchroneity of past events using Bayesian reconstructions of sedimentation history. Quaternary Science Reviews, 27(19-20), 1872-1885. DOI:10.1016/j.quascirev.2008.07.009

`pyleoclim.RBchron.`**`plotBchron`**(*depth, chron, positions, ageDist, flipCoor=False, xlabel='Depth', ylabel='Age', xlim=None, ylim=None, violinColor='#8B008B', medianLineColor='black', medianLineWidth=2.0, CIFillColor='Silver', samplePaths=True, samplePathNumber=10, alpha=0.5, figsize=[4, 8], ax=None*)

Plot a Bchron output

This function creates a plot showing the calibrated calendar ages and associated 95% confidence interval as error bars, the 95% ensemble from the produced age model as well as randomly drawn members of the ensemble.

Args:

depth (array): the positions in the archive (often referred to as depth) where the age model was interpolated to. Should be a vector

chron (array): The possible age models returned by BChron. The number of rows should be the same length as the depth vector, with each possible realization stored in the columns.

positions (array): The depth on the archive at which chronological measurements have been made. Should be a vector

agesDist (array): The distribution of ages for each chronological tie points. The number of columns should correspond to the number of chronological tie points available.

flipCoor (bool): If True, plots depth on the y-axis. xlabel (str): The label for the x-axis ylabel (str): The label for the y-axis xlim (list): Limits for the x-axis. Default corresponds to the min/max

of the depth vector.

ylim (list): Limits for the y-axis. Default set by matplotlib violinColor (str): The color for the violins. Default is purple medianLineColor (str): The color for the median line. Default is black. medianLineWidth (float): The width for the median line CIFillColor (str): Fill color in between the 95% confidence interval.

Default is silver.

samplePaths (bool): If True, draws sample paths from the distribution. Use the same color as the violins.

samplePathNumber (int): The number of sample paths to draw. Default is 10. Note: samplePaths need to be set to True.

alpha (float): The violins' transparency. Number between 0 and 1 figsize (list): The figure size. Default is [4,8] ax: Default is None. Useful to set for subplots.

Returns:

- fig: the figure.

CHAPTER ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

A

agemodelData() (in module pyleoclim.SummaryPlots), 32
ar1_fit() (in module pyleoclim.Spectral), 33
ar1_sim() (in module pyleoclim.Spectral), 33

B

Bchron() (in module pyleoclim), 15
bin() (in module pyleoclim.Timeseries), 25
binTs() (in module pyleoclim), 12

C

checkXaxis() (in module pyleoclim.LipdUtils), 28
chooseCalCurves() (in module pyleoclim.RBchron), 39
clean_ts() (in module pyleoclim.Timeseries), 26
corrsig() (in module pyleoclim.Stats), 23
corrSigTs() (in module pyleoclim), 11
createDir() (in module pyleoclim.LipdUtils), 27

D

detrend() (in module pyleoclim.Timeseries), 26

E

enumerateLipds() (in module pyleoclim.LipdUtils), 27
enumerateTs() (in module pyleoclim.LipdUtils), 28

F

fetchTs() (in module pyleoclim), 7

G

gaussianize() (in module pyleoclim.Timeseries), 26
gaussianize_single() (in module pyleoclim.Timeseries), 26
getLipd() (in module pyleoclim.LipdUtils), 27
getMeasurement() (in module pyleoclim.LipdUtils), 29
getMetadata() (in module pyleoclim.SummaryPlots), 31
getTs() (in module pyleoclim.LipdUtils), 28

H

histTs() (in module pyleoclim), 10

I

interp() (in module pyleoclim.Timeseries), 25
interpTs() (in module pyleoclim), 12
isEnsemble() (in module pyleoclim.LipdUtils), 30
isMeasurement() (in module pyleoclim.LipdUtils), 29
isModel() (in module pyleoclim.LipdUtils), 29

L

LipdToOntology() (in module pyleoclim.LipdUtils), 29

M

mapAgeEnsembleToPaleoData() (in module pyleoclim.LipdUtils), 30
mapAll() (in module pyleoclim.Map), 19
mapAllArchive() (in module pyleoclim), 7
mapLipd() (in module pyleoclim), 8
mapNearRecords() (in module pyleoclim), 8
mapOne() (in module pyleoclim.Map), 19
modelNumber() (in module pyleoclim.LipdUtils), 29

O

onCommonAxis() (in module pyleoclim.Timeseries), 25
openLipd() (in module pyleoclim), 7

P

plot() (in module pyleoclim.Plot), 21
plot_coherence() (in module pyleoclim.Spectral), 35
plot_hist() (in module pyleoclim.Plot), 21
plot_psd() (in module pyleoclim.Spectral), 36
plot_summary() (in module pyleoclim.Spectral), 36
plot_wwa() (in module pyleoclim.Spectral), 35
plot_wwadist() (in module pyleoclim.Spectral), 36
plotBchron() (in module pyleoclim.RBchron), 41
plotTs() (in module pyleoclim), 9
promptForVariable() (in module pyleoclim.LipdUtils), 28

R

reservoirAgeCorrection() (in module pyleoclim.RBchron), 39
runBchron() (in module pyleoclim.RBchron), 39

S

saveFigure() (in module pyleoclim.LipdUtils), 27
searchVar() (in module pyleoclim.LipdUtils), 28
segmentTs() (in module pyleoclim), 13
simpleStats() (in module pyleoclim.Stats), 23
standardize() (in module pyleoclim.Timeseries), 25
standardizeTs() (in module pyleoclim), 13
statsTs() (in module pyleoclim), 11
summaryTs() (in module pyleoclim), 11

T

ts2segments() (in module pyleoclim.Timeseries), 26
TsData() (in module pyleoclim.SummaryPlots), 31

W

whichMeasurement() (in module pyleoclim.LipdUtils), 29
wwz() (in module pyleoclim.Spectral), 33
wwz_psd() (in module pyleoclim.Spectral), 34
wwzTs() (in module pyleoclim), 13

X

xAxisTs() (in module pyleoclim.LipdUtils), 28
xwc() (in module pyleoclim.Spectral), 34