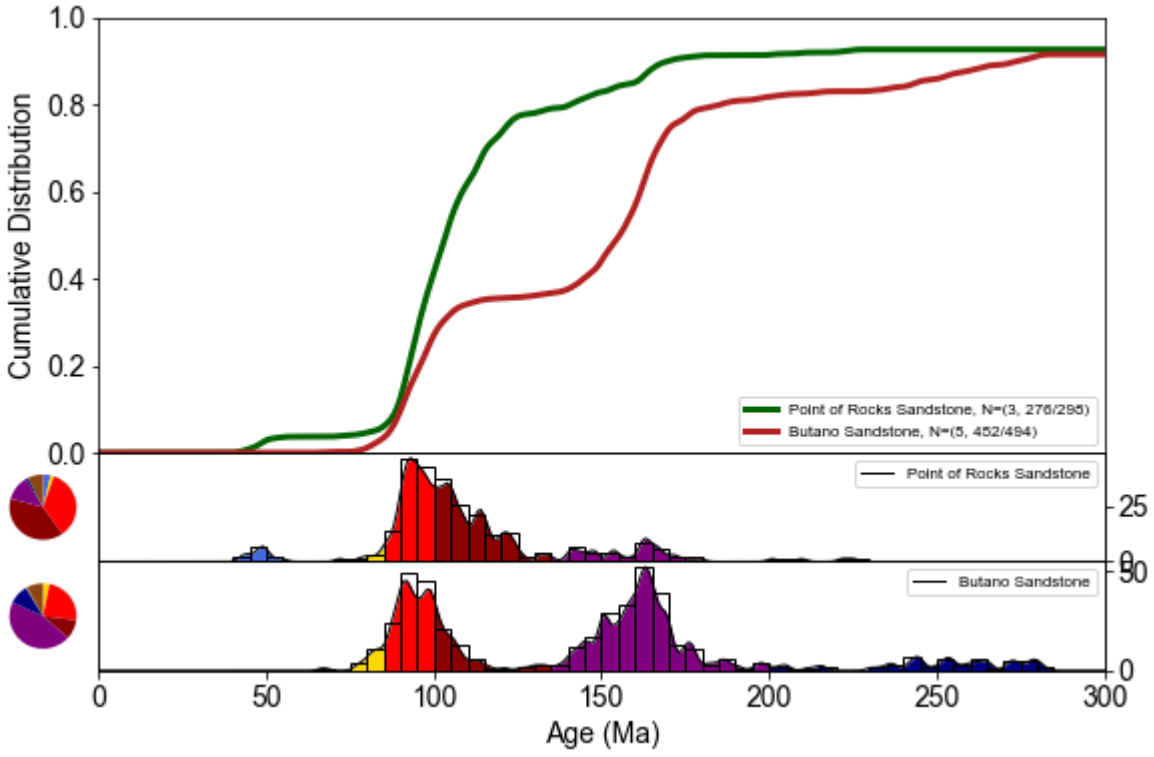
detritalPy

A Python-based Toolset for Visualizing and Analyzing Detrital Geo-Thermochronologic Data

Version 1.3.18

Updated June 5, 2020

Glenn R. Sharman, Jonathan P. Sharman, and Zoltan Sylvester



Cover page: Example visualization of detrital zircon U-Pb age distributions from two sample groups. The top plot displays a cumulative kernel density estimate (CKDE) of all three samples using a bandwidth of 1.5 Myr. The bottom panels display (1) kernel density estimates (KDE) constructed using a bandwidth of 1.5 Myr and shaded according to user-defined age populations, (2) a histogram (bin size of 5 Myr), and (3) pie diagrams showing the proportions of user-defined age populations. Data are from Sharman *et al*. (2013, 2015).

TABLE OF CONTENTS

INTRODUCTION

This manual provides an overview of detritalPy, a Python-based toolset designed to promote efficient organization, visualization, and analysis of detrital geochronologic and thermochronologic datasets. For additional information, please refer to Sharman *et al*. 2018 (https://doi.org/10.1002/dep2.45) and commented lines within detritalPy code (https://github.com/grsharman/detritalPy). Dr. Jeff Amato (New Mexico State University) has put together a beginner's guide to detritalPy (geared towards MacOS) that can be downloaded at https://geology.nmsu.edu/files/2020/04/DetritalPyForMacManual.pdf.

SELECTED UPDATES

v.1.3.18 (June 5, 2020)

- MDS stress values are now only reported from the sklearn module. 'Stress-1' is no longer offered as an output (see Multi-dimensional scaling section for more information).
- Metric MDS can now be specified as a starting configuration for the initialization of non-metric MDS (see Multi-dimensional scaling section for more information).
- Similarity metrics (S; i.e., Cross-correlation, similarity, likeness) are now transformed into dissimilarity via the square root of (1-S), following Sundell and Saylor (2017). Previously (v.1.3.17), similarity metrics were transformed by (1-S).
- Minimum cluster size ('min_cluster_size') can now be specified for the Youngest Statistical Population (YSP) function.

v.1.3.17 (May 8, 2020)

- Module installation via **pip install detritalpy**
- Implementation of a revised and updated **multidimensional scaling** routine that includes new plotting functions, dissimilarity metrics (similarity and likeness added), and control over algorithm parameters (e.g., number of initializations and iterations)
- New algorithms for calculating the **maximum depositional age** (see Sharman and Malkowski, 2020: Earth-Science Reviews, doi:10.1016/j.earscirev.2020.103109)
- Improvements to **Plot detrital age distributions**, including the ability to split the x-axis in (see tutorial_splitAxis.ipynb)

v.1.2.1. (June 5, 2019)

- New plotting options in **Plot detrital age distributions**
    - Plot age distributions as heatmaps
    - Plot depositional age as a vertical line
- New **multidimensional scaling** plots and options
    - The stress parameter is now reported using the 'Stress-1' formulation by default (Kruskal, 1964; Vermeesch, 2013). The stress formulation used by the sklearn manifold library ("sum of squared distance of the disparities and the distances for all constrained points") that was used in previous detritalPy versions can also be output, optionally.
    - The number of times the algorithm is run with different initializations (the '**n_init**' parameter) has been increased by default to help minimize stress in large, complex datasets.
    - A number of new plots can now be generated, including a "QQ" plot, a dissimilarity matrix plotted as a heatmap, a stress vs number of dimensions plot, and a Shepard's plot.
    - MDS points can be colored by a variable (e.g., by unit or basin).
    - MDS points can be plotted as a pie diagram where the units represent any sum-to-one data.

PYTHON INSTALLATION

If using Python for the first time, we suggest installing the open data science platform Anaconda by Continuum Analytics that includes the most commonly used Python packages. detritalPy is compatible with Python 3.7 and is run through Jupyter Notebook or Jupyter Lab.

DETRITALPY INSTALLATION

If using Windows, launch the Anaconda Prompt. If using a Macintosh, launch a Terminal window. Enter the following command:

pip install detritalpy

We recommend frequently checking for updates to detritalPy, which can be done in a similar way as the initial install:

pip install detritalpy --upgrade

(note the double dash ('--') before upgrade)

Although the basic functions of detritalPy can be accessed via the packages that are installed by default with Anaconda, additional modules are required to utilize the full functionality of detritalPy. These modules are automatically installed upon installing detritalPy, if not already installed.

- numpy
- matplotlib
- pandas
- xlrd
- folium
- vincent
- simplekml
- scipy
- sklearn
- statsmodels
- peakutils

DATA FORMATING

detritalPy requires that input data be organized using a specific format. The default format is a Microsoft Excel file that contains two worksheets: (1) a worksheet that contains a row for each unique sample in the dataset (default worksheet name is **"Samples"**), and (2) a data worksheet that contains a row for each unique detrital analysis in the dataset (default worksheet name is **"ZrUPb"** for zircon U-Pb and/or (U-Th)/He data) (Fig. 1). This data structure places sample-level information within the **Samples** worksheet and zircon U-Pb and/or (U-Th)/He analysis-level data within the **ZrUPb** worksheet. The two worksheets are linked together by a unique sample identifier ("**Sample_ID"** column) that must be assigned to each individual

sample. Each sample identifier must be distinct (unique) from all others in the database. There should be no empty rows within data in either the **Samples** or **ZrUPb** worksheets.

More than one Excel file can be loaded into detritalPy, provided that there is no duplication of data, no duplicated sample identifiers, and the column headings are all the same. Two example datasets are provided to illustrate how input data should be formatted.

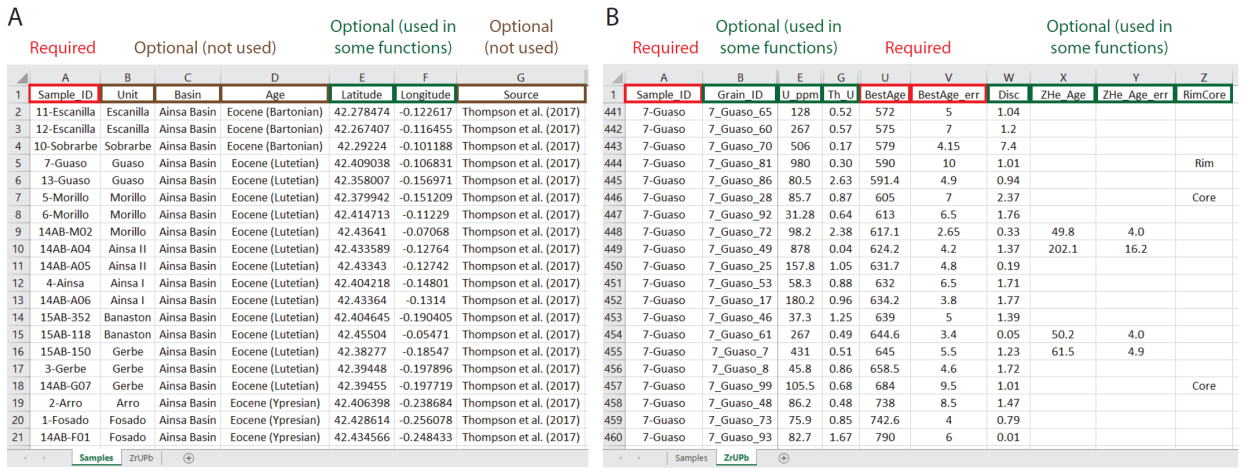


Figure 1. Illustration of default Excel data format. (A) Samples worksheet that contains a row for each individual sample. (B) Data worksheet ("ZrUPb") that contains a row for each individual analysis.

Samples Worksheet

The **Samples** worksheet contains information related to individual samples. The first row of the **Samples** worksheet must contain column names and at a minimum must have a "**Sample_ID**" column. Each unique sample in the database must have its own row and must be labeled with a unique sample identifier that can be any combination of numbers, letters, and symbols. We recommend that sample identifiers not be comprised of only numbers or contain a backslash. "**Latitude**" and "**Longitude**" fields are required for plotting the locations of samples on a map, and must be entered in decimal degrees format (e.g., XX.XXXX°) using a WGS84 datum. Other column headings ("**Age**", "**Basin**", etc.) can also be included but are not currently used in detritalPy functions.

Required column headings:

- **Sample_ID**: An alpha-numeric identifier that is unique for each individual sample

Optional column headings, but used for some detritalPy functions:

- **Latitude**: Sample coordinates in decimal degrees format (e.g., XX.XXXX°) using a WGS84 datum. Negative indicates southern hemisphere.
- **Longitude**: Sample coordinates in decimal degrees format (e.g., XX.XXXX°) using a WGS84 datum. Negative indicates western hemisphere.

ZrUPb Worksheet

The **ZrUPb** worksheet contains information related to each individual detrital analysis. Note that a single mineral grain may have multiple analyses. A minimum of three columns are required: "**Sample_ID**", "**BestAge**", and "**BestAge_err**". The **Sample_ID** column must contain the exact, unique sample identifier that is used to identify the sample within the Samples worksheet. The **BestAge** column represents the preferred U-Pb age of the grain analysis (typically the $^{206}Pb/^{238}U$ age for young grains and the $^{207}Pb/^{206}Pb$ for older grains). The **BestAge_err** column must contain the analytical uncertainty, either reported at 1σ or 2σ confidence level. ***The choice of 1σ or 2σ confidence level must be used consistently throughout the entire database***, and can be specified when loading data using the **sampleToData()** function. All detritalPy functions assume 1σ analytical uncertainties, and 2σ analytical uncertainties will be converted to 1σ during data loading. In addition, both the **BestAge** and **BestAge_err** columns must not be blank and must be a number (e.g., "DISC", "N/A", or other non-numeric values will cause an error).

If additional numeric data types are included, such as the concentration of Uranium ("**U_ppm**") or Th:U ("**Th_U**"), these data can be plotted. Rim and core relationships can be plotted if each analysis has a unique grain identifier ("**Grain_ID**") and a column that identifies rims and cores (default is "**RimCore**"). Other data columns (e.g., **"Analysis_ID"**) may be useful but will not be used for data analysis by detritalPy.

Required column headings:

- **Sample_ID**: Unique sample identifier that matches the **Sample_ID** used in the Samples worksheet.
- **BestAge**: The preferred geochronologic or thermochronologic age. Default is the U-Pb crystallization age.

- **BestAge_err**: The 1σ or 2σ uncertainty associated with the **BestAge**

Optional column headings, but used for some detritalPy functions:

- Any numerical attribute associated with a detrital analysis (e.g., U_ppm, Th_U, or trace element abundance).
- **Grain_ID**: Unique grain identifier, used for plotting rim versus core ages
- **RimCore**: Column that identifies rims and cores
- **ZHeAge**: The (U-Th)/He cooling age of the detrital analysis
- **ZHeAge_err**: The 1σ or 2σ uncertainty associated with the **ZHeAge**

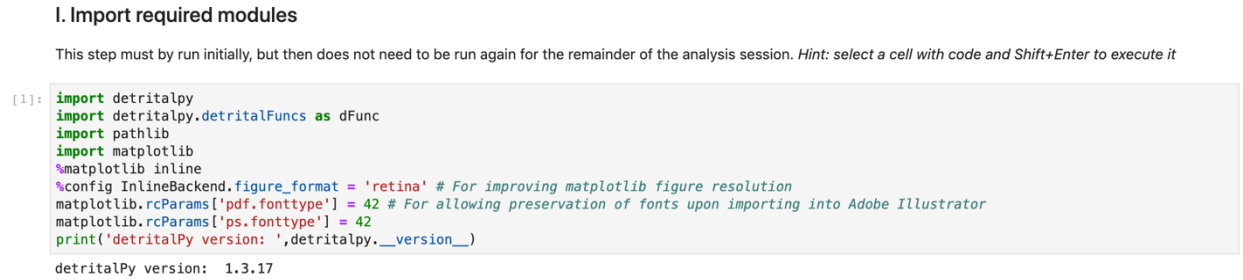
OPENING JUPYTER NOTEBOOK

detritalPy is run by opening the detritalPy.ipynb file in Jupyter Notebook, a browser-based Python application. Launch Jupyter Notebook and open the latest detritalPy notebook (e.g., detritalPy_v.1.3.ipynb). Launch the detritalPy notebook by clicking on the detritalPy.ipynb file from the Home screen.

STEPS I-III: Database import and sample selection

I. Import required modules

The first cell of detritalPy imports the modules that are required for the code to function.



I. Import required modules

This step must by run initially, but then does not need to be run again for the remainder of the analysis session. *Hint: select a cell with code and Shift+Enter to execute it*

[1]:
```
import detritalpy
import detritalpy.detritalFuncs as dFunc
import pathlib
import matplotlib
%matplotlib inline
%config InlineBackend.figure_format = 'retina' # For improving matplotlib figure resolution
matplotlib.rcParams['pdf.fonttype'] = 42 # For allowing preservation of fonts upon importing into Adobe Illustrator
matplotlib.rcParams['ps.fonttype'] = 42
print('detritalPy version: ',detritalpy.__version__)
```

```
detritalPy version:  1.3.17
```

II. Import the dataset as an Excel file

The Excel file(s) that contains the detrital age dataset(s) is/are imported in Step II. Relative file paths can be used if the files are in the same folder directory as the detritalPy.ipynb.

More examples of different methods of importing data into detritalPy are provided in tutorial_dataLoading.ipynb.

II. Import the dataset as an Excel file

This step must by run initially, and should be repeated if any changes are made to the dataset in Excel

In [ ]:
```
# Import relative file pathway(s)
from pathlib import Path

# Specify file paths to data input file(s)
dataToLoad = [Path("example-data/") / "ExampleDataset_1.xlsx",
              Path("example-data/") / "ExampleDataset_2.xlsx"]

main_df, main_byid_df, samples_df, analyses_df = dFunc.loadDataExcel(dataToLoad)
```

Alternatively, file paths can be hard coded. If using Windows, backslashes are acceptable provided the file path name is preceded by 'r' as shown below:

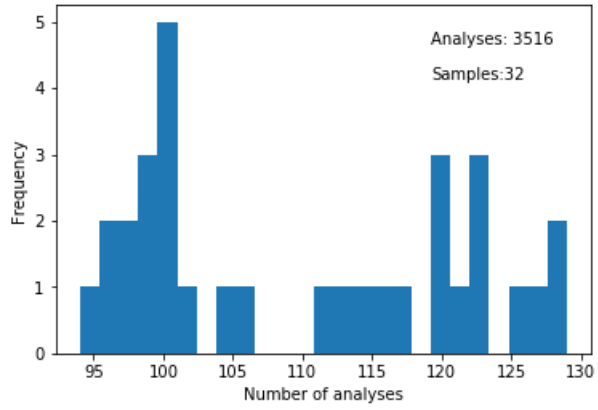
In [ ]:
```
# Specify file paths to data input file(s)
dataToLoad = [r'C:\Users\gsharman\Documents\GitHub\detritalPy\example-data\ExampleDataset_1.xlsx',
              r'C:\Users\gsharman\Documents\GitHub\detritalPy\example-data\ExampleDataset_1.xlsx']

main_df, main_byid_df, samples_df, analyses_df = dFunc.loadDataExcel(dataToLoad)
```

In either case, multiple Excel files can be loaded by listing the file paths in the **'dataToLoad'** array. Data should not be duplicated if loading multiple Excel files, however.

To create a plot of sample size distribution (number of grains per sample), execute the following cell. The number of grain ages and samples are shown in the upper right portion of the plot.

Optional: run the cell below to plot a distribution of sample size (number of analyses per sample) in the dataset

```
In [3]: dFunc.plotSampleDist(main_byid_df, numBins=25)
```



Step II should be repeated if any changes are made to the Excel file.

III. Select samples

detritalPy supports two methods of selecting samples for plotting and analysis. (1) Any number of individual samples can be selected for plotting by entering a list of sample identifiers in the 'sampleList' array, in the format shown below.

III. Select samples

Individual or groups of samples can be selected by entering their unique Sample ID's in an array or tuple (see example below for the correct syntax). This sample list will be used for all subsequent plotting and analysis functions.

```
In [4]: sampleList = ['POR-1','POR-2','POR-3','BUT-5','BUT-4','BUT-3','BUT-2','BUT-1']

        ages, errors, numGrains, labels = dFunc.sampleToData(sampleList, main_byid_df, sigma = '1sigma');
```

(2) Any number of groups of samples can also be selected by entering a list of sample identifiers within a tuple data structure, using the format shown below. For each group, a label must be provided.

```
In [5]: sampleList = [(['POR-1','POR-2','POR-3'],'Point of Rocks Sandstone'),
                      (['BUT-5','BUT-4','BUT-3','BUT-2','BUT-1'],'Butano Sandstone')]

        ages, errors, numGrains, labels = dFunc.sampleToData(sampleList, main_byid_df, sigma = '1sigma');
```

Step III should be repeated to select a different set of samples or groups of samples for plotting or analysis. Also, samples or groups of samples will be plotted in the order (top to bottom) that they are entered into the array or tuple, respectively.

DETRITALPY FUNCTIONS

The primary visualization and analysis functions included with detritalPy are described below. These functions can be executed in any order.

Plot detrital age distributions

detritalPy provides flexibility in plotting detrital age distributions using a variety of the most common data visualization approaches. The plotting function is divided into a cumulative distribution plot (upper panel) and one or more relative distribution plots (lower panel). The **'whatToPlot'** variable can be set to equal **'cumulative'**, **'relative'**, or **'both'**. When set to **'both'**, the cumulative distribution is shown on top and the relative distribution for each sample or group of samples is shown below.

Setting **'separateSubplots'** variable equal to **True** results in each relative age distribution being plotted in a separate subplot. This option allows a histogram and pie diagram to also be plotted with relative age distributions. If **'separateSubplots'** does not equal **True**, relative age distributions will be plotted on top of each other, rather than in separate subplots.

```
# Enter plot options below
whatToPlot = 'both' # Options: cumulative, relative, or both
separateSubplots = True # Set to True to plot each relative age distribution in a separate subplot (allows histogram and pie)
```

The age range of the plot in Myr can be selected by assigning the variables **x1** and **x2** that specify the beginning and age of the plotted age range, respectively. To plot the x-axis on a log scale, set '**plotLog'** to **True**. Note that if **x1** = 0, then it will be automatically assigned to equal 0.1.

```
# Specify the age range (Myr) that you want to plot
x1 = 0
x2 = 300
plotLog = False # Set to True to plot the x-axis as a log scale
```

Plot dimensions can be specified using the variables **'w'**, **'c'**, and **'h'**.

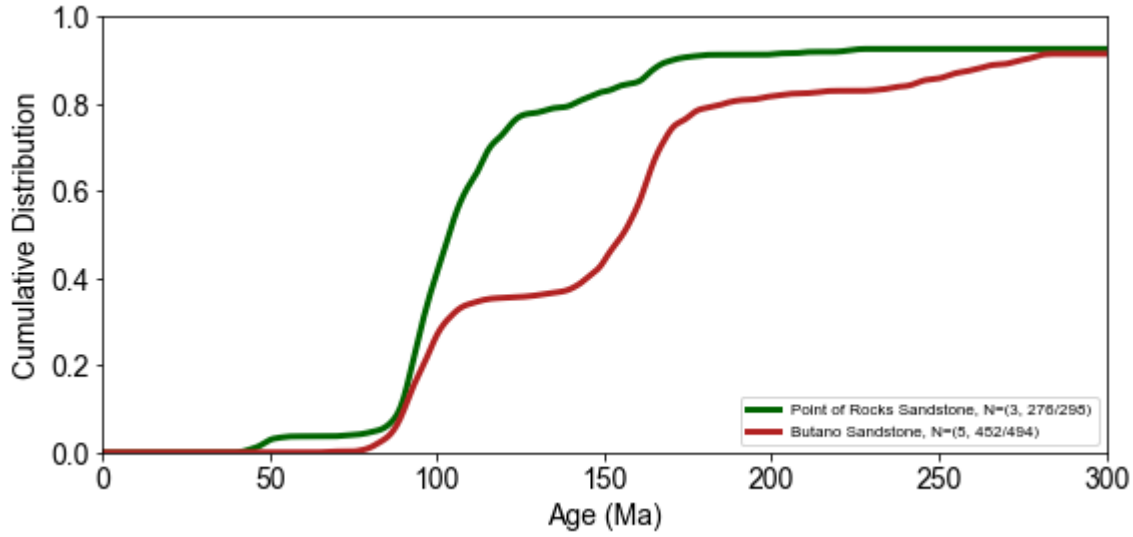
```
# Specify the plot dimensions
w = 10 # width of the plot
c = 4 # height of CDF panel
h = 5 # height of the relative panel (only required if separateSubplots is False). Options: 'auto' or an integer
```

The variable '**xdif'** specifies the discretization interval (in Myr) at which age distributions are calculated. The default (*and recommended value*) is 1 Myr. Changing **'xdif'** to a large number can result in algorithm instability.

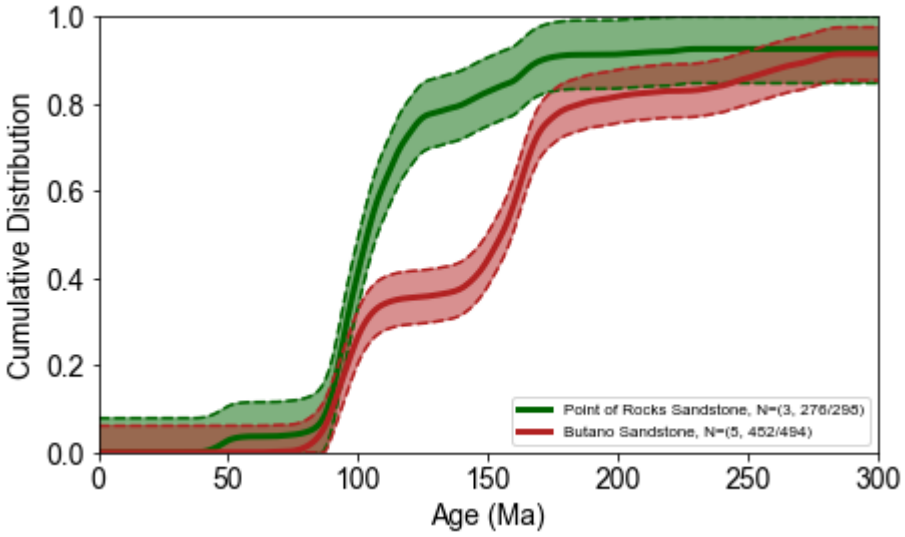
```
# Specify the interval (Myr) over which distributions are calculated
xdif = 1
```

There are three options for plotting a cumulative distribution. Set the variable **'plotCDF'** to **True** to plot a standard cumulative distribution function that is discretized at **xdif** interval. Set **'plotCPDP'** to **True** to plot a cumulative (summed) probability density plot (CPDP). Set **'plotCKDE'** = to **True** to plot a cumulative (summed) kernel density estimate (CKDE). Both the CPDP and CKDE produce a smoothed cumulative distribution relative to the unsmoothed CDF.

Cumulative distribution functions are automatically colored by sample or sample group. The colors that are selected can be customized by modifying the **colorMe()** function within the detritalFuncs.py file. Note that the figure legend includes the number of samples (X), the number of analyses shown in the plotted age range (Y), and the total number of analyses in the sample or sample group (Z) in the following notation: N=(X, Y/Z).



A 95% confidence interval about the cumulative distribution function can be estimated based on the Dvoretsky-Kiefer-Wolfowitz inequality (Anderson *et al*., 2018). This option can be enabled by setting the variable variable **'plotDFW'** to **True**. By default, the 95% confidence interval is denoted by shaded lines and transparent shading that are colored the same as the corresponding CDF.



If relative distributions are selected to be plotted (i.e., **'whatToPlot'** = **'relative'** or **'both'**), then there are a number of options can be selected.

- **'normPlots'**: setting to **True** will result in relative distributions (i.e., PDP, KDE, and histogram) having the same y-axis scale. This option also applies if **'separateSubplots'** equals **True**; relative distributions are always normalized if **'separateSubplots'** does not equal **True**.

```
# Relative distribution options
normPlots = False # Will normalize the PDP/KDE if yes to True (if separateSubplots is True)
```

- **'plotKDE'**: setting to **True** will plot a KDE as a black line.
- **'colorKDE'**: setting to **True** will fill each KDE plot with a solid color. The color will match the CDF, CKDE, and/or CPDP, if selected to be plotted. This will only be plotted if **'plotKDE'** is set to **True**.
- **'colorKDEbyAge'**: setting to **True** will fill each KDE plot with a color spectrum that corresponds to user-specified age boundaries. The age boundaries and color choices are specified in the **'agebins'** and **'agebinsc'** variables. This will only be plotted if **'plotKDE'** is set to **True**.
- **'bw'**: sets the KDE bandwidth. There are three options:
    - Setting **'bw'** equal to a number assigns a bandwidth in Myr.
    - Setting **bw** equal to **'optimizedFixed'** selects an optimized bandwidth for each sample or sample group, based on Shimazaki and Shinomoto (2010). This option

requires the 'adaptiveKDE.py' file to present in the default folder (i.e., same folder as the 'detritalFuncs.py' file.

- Setting **bw** equal to **'optimizedVariable'** assigns a bandwidth that varies depending on the dataset characteristics for each sample or sample group, based on Shimazaki and Shinomoto (2010). Note that this option can result in very long processing times, particularly for large numbers of samples or sample groups. This option requires the 'adaptiveKDE.py' file to present in the default folder (i.e., same folder as the 'detritalFuncs.py' file.

```
plotKDE = True # Set to True if want to plot KDE
colorKDE = False # Will color KDE according to same coloration as used in CDF plotting
colorKDEbyAge = True # Will color KDE according to age populations if set to True
bw = 1.5 # Specify the KDE bandwidth. Options are 'optimizedFixed', 'optimizedVariable', or a number (bandwidth in Myr)
```

- **'plotPDP'**: setting to **True** will plot a PDP as a black line.
- **'colorPDP'**: setting to **True** will fill each PDP plot with a solid color. The color will match the CDF, CKDE, and/or CPDP, if selected to be plotted. This will only be plotted if **'plotPDP'** is set to **True**.
- **'colorPDPbyAge'**: setting to **True** will fill each PDP plot with a color spectrum that corresponds to user-specified age boundaries. The age boundaries and color choices are selected in the **'agebins'** and **'agebinsc'** variables (see below). This will only be plotted if **'plotPDP'** is set to **True**.

```
plotPDP = False # Set to True if want to plot PDP
colorPDP = False # Will color PDP according to same coloration as used in CDF plotting
colorPDPbyAge = True # Will color PDP according to age populations if set to True
```

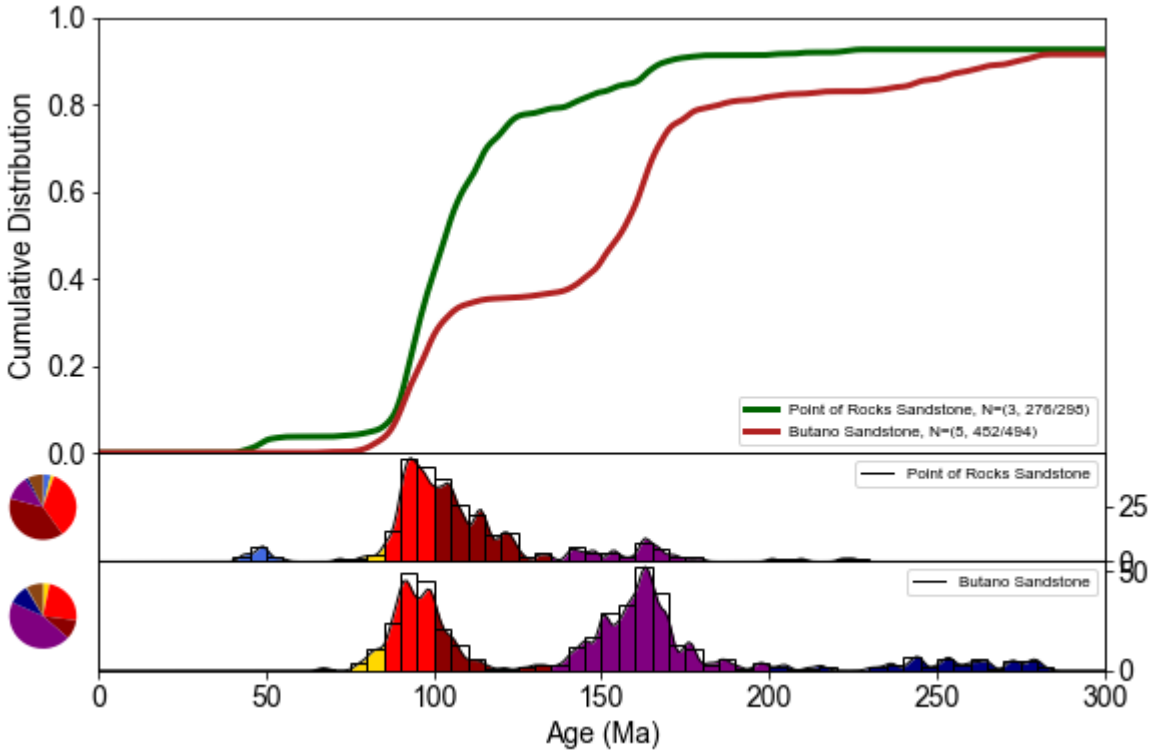
- **'plotColorBar'**: setting to **True** will result in a vertical color bar being shown on the cumulative and/or relative plots. The age boundaries and color choices are selected in the **'agebins'** and **'agebinsc'** variables (see below).
- **'plotHist'**: setting to **True** will result in a histogram being plotted.
- **'b'**: specifies the histogram bin size in Myr
- **'plotPIE'**: setting to **True** will result in a pie diagram being plotted to the left of each age distribution. The age boundaries and color choices are selected in the **'agebins'** and **'agebinsc'** variables (see below).
- **'agebins'**: an array that contains the age bin boundaries.
- **'agebinsc'**: an array of colors that correspond to the specified age bin boundaries.

```
plotColorBar = False # Color age categories as vertical bars, can add white bars to create blank space between other colored bars

plotHist = True # Set to True to plot a histogram (only available when separateSubplots is True)
b = 5 # Specify the histogram bin size (Myr)

plotPIE = True # Will plot a pie diagram (only available when separateSubplots is True)

# Specify  age categories for colored KDE, PDP, and/or pie plots
# Sharman et al. 2015 scheme
agebins = [0, 23, 65, 85, 100, 135, 200, 300, 500, 4500]
agebinsc = ['slategray','royalblue','gold','red','darkred','purple','navy','gray','saddlebrown']
```

- **'plotAgePeaks'**: setting to **True** will result in identification of age peaks, based on the peakutils library that must be installed separately (https://pypi.python.org/pypi/PeakUtils).
- **'agePeakOptions'**: specifies the parameters used to identify age peaks within an array. 'distType' may be set to either 'KDE' or 'PDP' and specifies which relative distribution to select age peaks from. 'threshold' refers to a y-axis cutoff from which to exclude potential age peaks. 'minDist' refers to an x-axis cutoff which relates to the proximity of adjacent age peaks. 'minPeakSize' refers to the minimum peak height threshold for an age peak. The age of the age peak (Ma) will be plotted if 'labels' is set to **True**. See https://pypi.python.org/pypi/PeakUtils and the agePeak() in detritalFuncs.py for more information.

```
plotAgePeaks = False # Will identify and plot age peaks
agePeakOptions = ['KDE', 0.05, 5, 2, True] # [distType, threshold, minDist, minPeakSize, labels]
```

Executing the cell results in generation of a figure, such as the one below (see also Cover Image and caption).



The resulting figure can be exported as a PDF file by executing the following cell.

Optional: Run the cell below to save the figure as a pdf file

```
In [ ]: pathlib.Path('Output').mkdir(parents=True, exist_ok=True) # Recursively creates
        fig.savefig('Output/DZageDistributions.pdf')
```

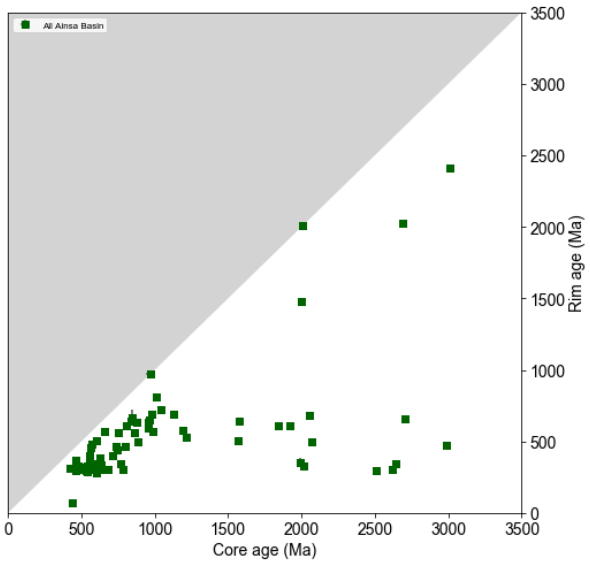
For instructions on how to split the x-axis, see tutorial_splitAxes.ipynb.

Plot rim age versus core age

Rim ages can be plotted against core ages in a scatterplot. This function requires that the **'ZrUPb'** worksheet contains 1) a column with a unique grain identifier (default is "**Grain_ID**"), and 2) a column that identifies rims and core (default is "**RimCore**" with "**Rim**" and "**Core**" designations).

- **'x1', 'x2', 'y1', 'y2'**: These variables specify the minimum and maximum x-axis and y-axis extents, in Ma.
- **'plotLog'**: Setting to **True** will plot the x-axis as a log scale.
- **'plotError'**: Setting to **True** will plot error bars.
- **'w', 'c'**: These variables specify the width and height of the plot.

```
In [10]: # Specify the age range (Myr) that you want to plot
x1 = 0
x2 = 3500
y1 = 0
y2 = 3500
plotLog = False # Set to True to plot the x-axis as a log scale
plotError = True # Select whether to plot error bars

# Specify the plot dimensions
w = 8 # width of the plot
c = 8 # height of the plot

rimsVsCores = dFunc.plotRimsVsCores(main_byid_df, sampleList, ages, errors, labels, x1, x2, y1, y2, plotLog, plotError, w, c)
```



Plot detrital age distributions in comparison to another variable (e.g., Th/U)

This function allows any numeric variable in the **'ZrUPb'** worksheet to be plotted adjacent (above) detrital age distributions for each sample or sample group. Examples could include the uranium concentration (**U_ppm**), the thorium to uranium ratio (**Th_U**), the epsilon hafnium value (**eHf**), or the concentration of a trace element.

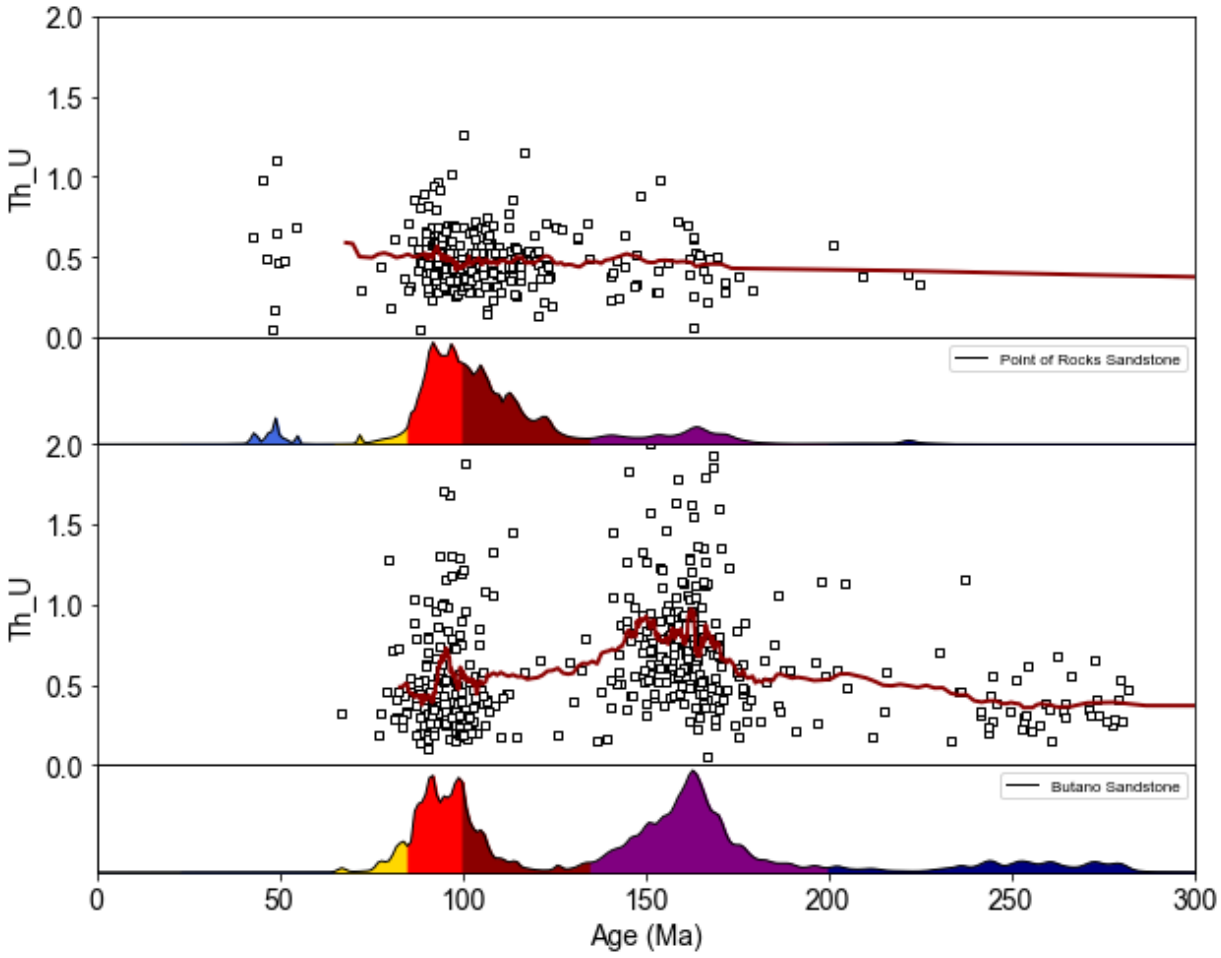
The variable is selected by assigning the column heading of the desired variable to '**variableName**'. Error bars can be plotted if **'plotError'** is set to **True**. **'variableError'** must be specified if **'plotError'** equals **True** and can be set to a percentage (e.g., 5% error would be entered as 0.05) or as a column heading that contains the error data (e.g., **"Th_U_err"**).

```
variableName = 'Th_U'
plotError = False # Select True to plot error bars
variableError = 0.05 # Required if plotError = True: Select the variable name or specify the error as a percentage (e.g., 0.05)
```

Many of the same options for plotting relative age distributions are described above in the **"Plot detrital age distributions"** section. In addition, the y-axis scale of the plotted variable can be selected automatically by setting **'autoScaleY'** to **True**. Otherwise, **'y1'** and **'y2'** will specify

the y-axis extents. Plot dimensions can be specified via the **'w'**, **'t'**, and **'l'** variables. A moving average can be plotted by setting '**plotMovingAverage'** to **True** with the '**windowSize'** variable indicating the number of analyses to include in the moving average.

```
plotMovingAverage = True
windowSize = 25
```

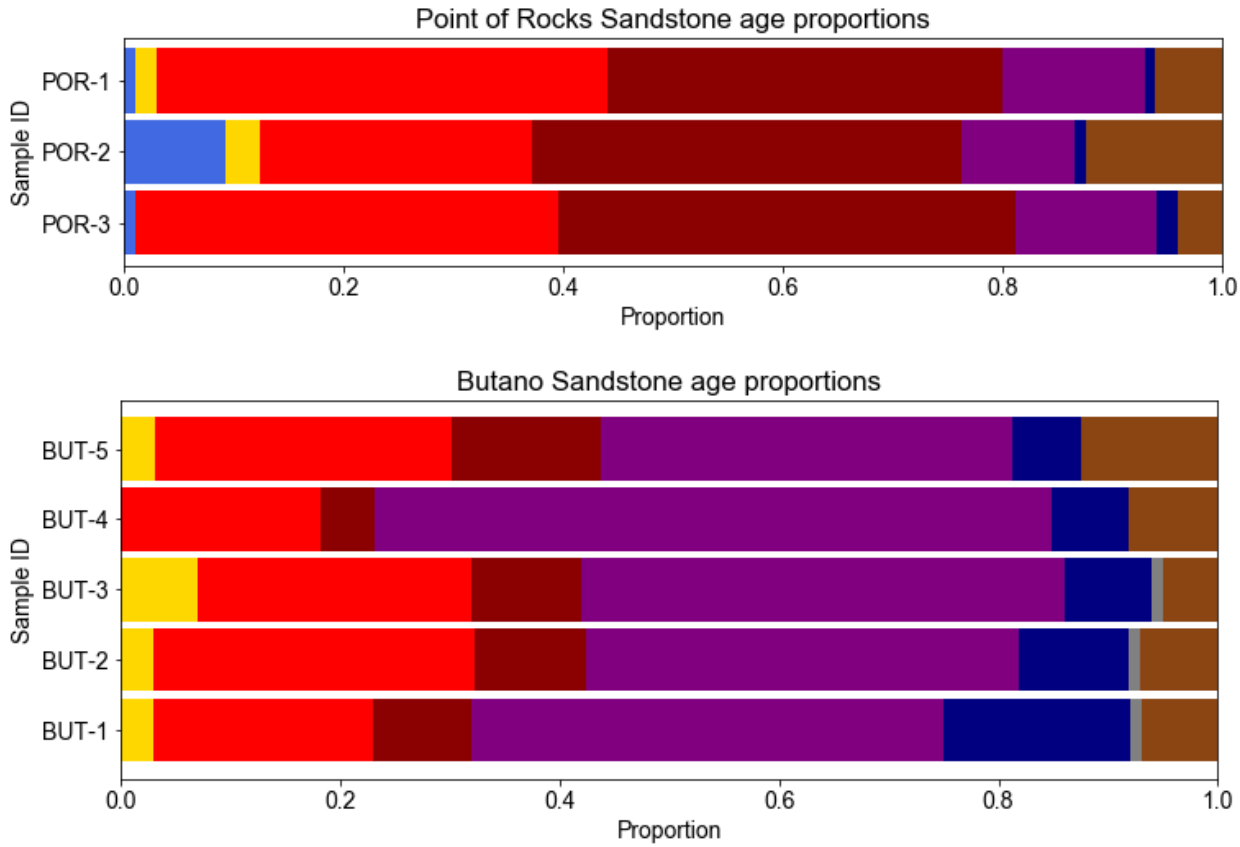


Example plot of Th/U versus PDP plots for the Point of Rocks Sandstone and Butano Sandstone groups (see example data).

Plot detrital age populations as a bar graph

This function allows detrital age distributions to be plotted as a bar graph, using user-specified age categories and colors. The dimensions of the plot, bar height, and bar overlap can

be specified using the **'overlap'**, **'width'**, and **'height'** variables. Setting **'separateGroups'** equal to **True** will divide sample groups into individual samples, rather than displaying each group as a single bar graph. A file name can be assigned to the variable **'fileName'**, and a CSV file will be automatically saved to the default project folder that contains population abundance information. Setting **'savePlot'** to **True** will automatically save each output plot as a PDF file within an Output folder in the default project directory. If a folder named "Output" does not already exist, one will be created.



Example output for the Point of Rocks Sandstone and Butano Sandstone groups (see example data).

Plot sample locations on an interactive map

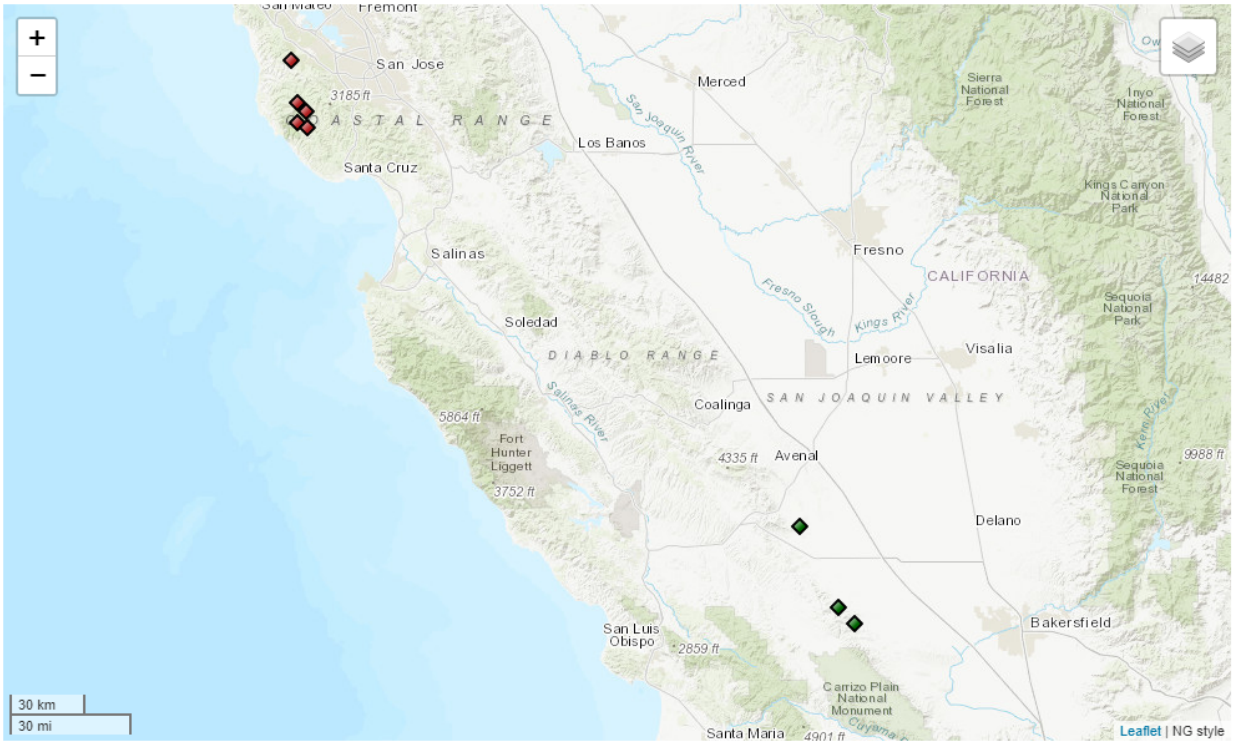
Samples with latitude and longitude coordinates (decimal degrees format; WGS84 datum) can be plotted on an interactive map (requires an internet connection). Sample age distributions can be optionally plotted by clicking on a sample by setting any of the following variables equal to **True**: **'plotMapKDE'**, **'plotMapPDP'**, and/or **'plotCumulative'**. If

'plotCumulative' equals **True**, then the KDE or PDP plot will be plotted as a CKDE or CPDP plot (provided that either **'plotMapKDE'** or **'plotMapPDP'** is also set to **True**). The upper age limit of the plot can be specified by setting **'x2'** equal to a number (Ma).

A number of different map options are available, and can be specified via the variable **'mapType'**. If **'exportKML'** is set to **True**, a Google Earth KML file will be automatically exported to the default project folder. A description can be included with each KML data point, and can be specified via the variable **'descrpt'** (default is **'Unit'**).

If plotting sample groups, each group will be colored according to the colorMe() function. We recommend not selecting plotting age distributions if plotting a large number of samples.

```
# Specify whether age distributions should be enabled (can be viewed by clicking on samples)
plotMapKDE = True # Choose True to enable KDEs when samples are selected
plotMapPDP = True # Choose True to enable PDPs when samples are selected
plotCumulative = True # Choose True to plot either a cumulative KDE or PDP. A discretized CDF will be plotted if KDE and PDP are

# Specify the upper age limit that you want to plot
x2 = 300
# Specify the KDE bandwidth
bw = 3 # Specify the KDE bandwidth. Options are 'optimizedFixed', 'optimizedVariable', or a number (bandwidth in Myr)

mapType = 'World_Topo_Map' # Options: 'NatGeo_World_Map', 'World_Street_Map', 'World_Topo_Map', 'World_Light_Gray',
                           # 'World_Shaded_Relief', 'World_Terrain_Base', 'World_Hillshade', 'World_Physical_Map'

exportKML = True
descrpt = 'Unit' # Description to be included with each sample
```



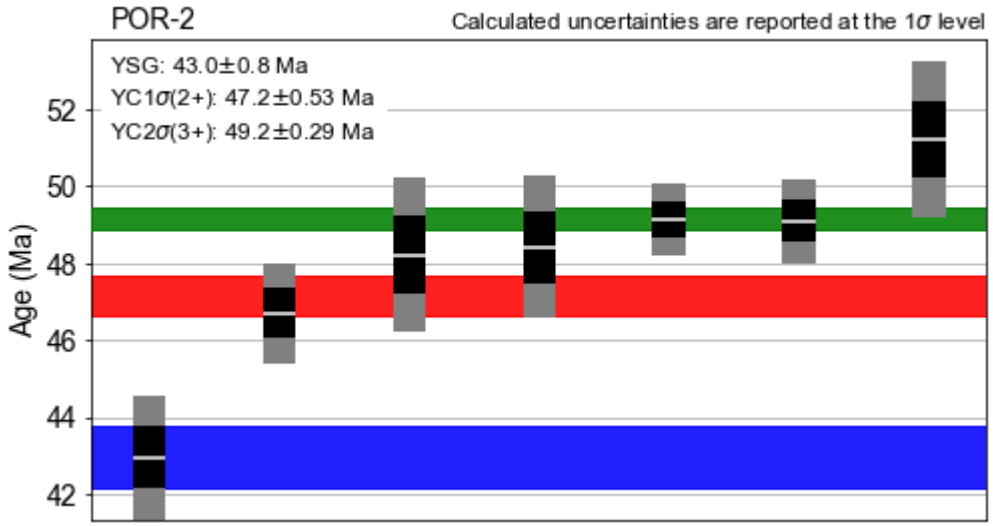
Example plot using the 'World_Topo_Map' option for the **'mapType'** variable.

Example Google Earth KML file generated from detritalPy.

Plot and export maximum depositional age (MDA) calculations

Estimates of maximum depositional age for a sample or sample group can be calculated and plotted. We use three metrics outlined in Dickinson and Geherls (2009): the youngest single grain (YSG), the youngest cluster of 2 or more ages with overlapping 1σ uncertainties (YC1σ(2+)), and the youngest cluster of 3 or more ages with overlapping 2σ uncertainties (YC2σ(3+)) (see Sharman *et al*., 2018, for additional details).

A CSV file will be automatically generated in the default project folder. It's name can be specified via the **'fileName'** variable. Setting **'makePlot'** to **True** will result in a plot of the youngest detrital ages and each maximum depositional age estimate to be generated for each sample or sample group. The grain analyses can be sorted by their mean age, mean age plus 1σ uncertainty, or mean age plus 2σ uncertainty via the variable **'sortBy'**. A number of plot parameters can be specified via the following variables: **'plotWidth'**, **'plotHeight'**, **'barWidth'**, **'ageColors'**, **'fillMDACalc'**, and **'alpha'**.



Example output showing the youngest detrital analyses and MDA calculations for a single sample.

Multi-dimensional scaling

Multi-dimensional scaling (MDS) plots are a popular approach to comparing sample similarity and dissimilarity (Vermeesch, 2013; Saylor *et al*., 2017). detritalPy allows plotting samples or groups of samples using metric or non-metric MDS. The MDS algorithm and plotting

routines were revised in versions 1.3.17 and 1.3.18. MDS is implemented via the sklearn module. Additional information on MDS as implemented by sklearn can be found here and in this useful tutorial.

The MDS is implemented via a class. The MDS model is first run and then various plots can be made (and executed in any order, provided the model has first been run).

```
[85]: model = dFunc.MDS_class(ages, errors, labels, sampleList, metric=False, criteria='Vmax', bw='optimizedFixed', n_init='metric',
                              max_iter=1000, x1=0, x2=4500, xdif=1, min_dim=1, max_dim=3, dim=2)
```

Setting the variable **'metric'** to **True** results in metric MDS, whereas non-metric MDS will be used if **'metric'** is **False**. The dissimilarity **'criteria'** variable can equal any of the following:

- **'Vmax'** (the sum of the maximum positive and negative differences between the two CDFs; **the default value used in detritalPy**)
- **'Dmax'** (the maximum difference between the CDFs)
- **'R2-PDP'** (the Cross-correlation coefficient between two PDP plots)
- **'R2-KDE'** (the Cross-correlation coefficient between two KDE plots)
- **'similarity-PDP'** (the similarity value between two PDP plots)
- **'similarity-KDE'** (the similarity value between two KDE plots)
- **'likeness-PDP'** (the likeness value between two PDP plots)
- **'likeness-KDE'** (the likeness value between two PDP plots)

Note that the MDS algorithm uses the square root of the complement of the Cross-correlation, similarity, and likeness values, such that the value (between 0 and 1) is an indication of dissimilarity, rather than similarity (e.g., Sundell and Saylor, 2017).

A number of additional optional keyword arguments allow you to specify how the algorithm operates (see the documentation for sklearn.manifold.MDS for more information).

- **'max_iter'** (the maximum number of iterations that the algorithm performs for a single run; default value = 1000)
- **'n_init'** (the number of algorithm initializations; the final result will be the best of these initializations, as determined by the output with the lowest final stress value)

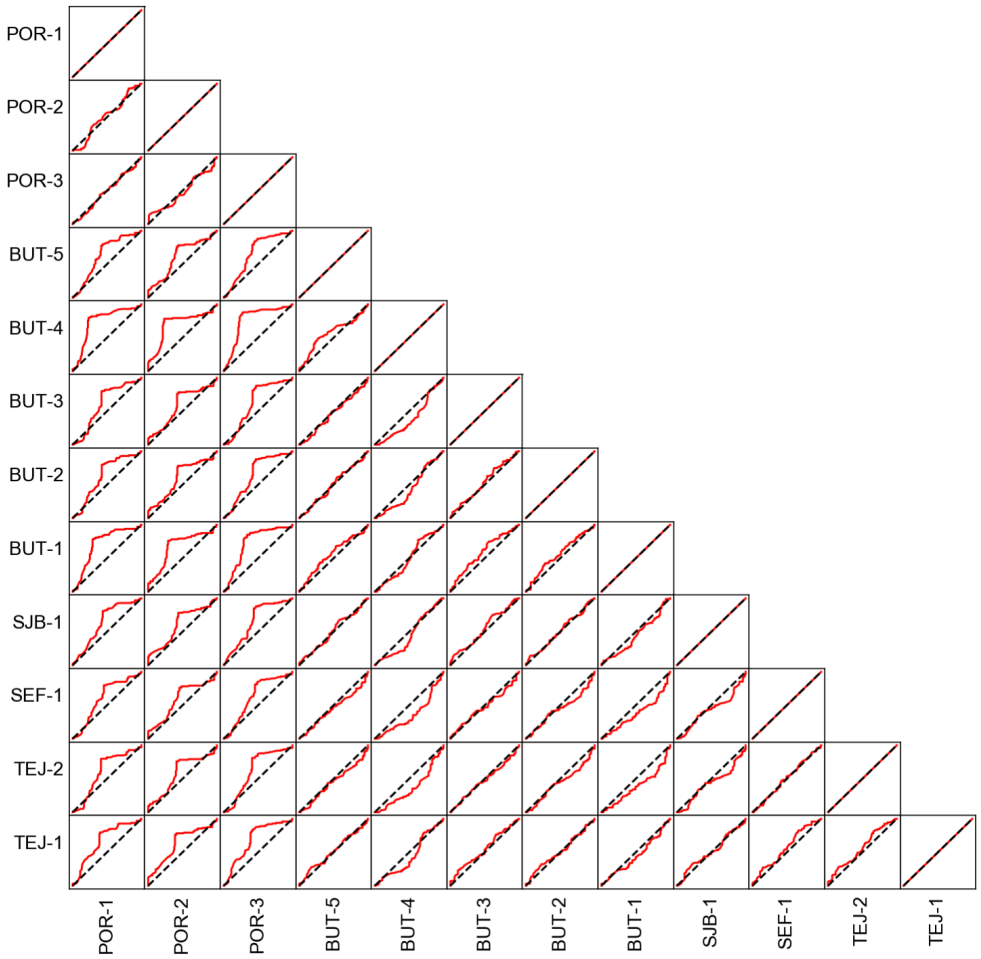
    - **Note:** if 'n_init' = 'metric', the configuration from metric MDS is used as a starting configuration for initialization of the algorithm, as demonstrated in this tutorial (new in v.1.3.18).
    - **Note:** non-metric MDS with 'n_init' = 'metric' were set to the default MDS_class paramters in v.1.3.18. Previously (v.1.3.17), 'n_init' = 300 was the default value.
- **'x1'** (the lower limit (Ma) of the age distributions to conduct MDS analysis on; default value = 0)
- **'x2'** (the upper limit (Ma) of the age distributions to conduct MDS analysis on; default value = 4500)
- **'min_dim'** (the minimum number of dimensions over which to calculate MDS; default value = 1)
- **'max_dim'** (the maximum number of dimensions over which to calculate MDS; default value = 3)
- **'dim'** (the chosen number of dimensions to plot; default value = 2)

Plot a QQ matrix

The QQ matrix plots each sample CDF against the others. A perfect match falls along the dashed link. *Note that this plotting function is not recommended for large datasets (e.g., >20 samples or sample groups).*

- **'figsize'** (specify the width and height of the figure, inches; default value = (12, 12))
- **'savePlot'** (set to True to export a PDF of the plot; default value = True)
- **'fileName'** (the name of the file used when exporting; only relevant if savePlot = True; default value = 'QQplot.pdf')
- **'halfMatrix'** (set to True to only plot half of the matrix; default value = True)

```
[21]: model.QQplot(figsize=(12,12), savePlot=False, fileName='QQplot.pdf', halfMatrix=True)
```



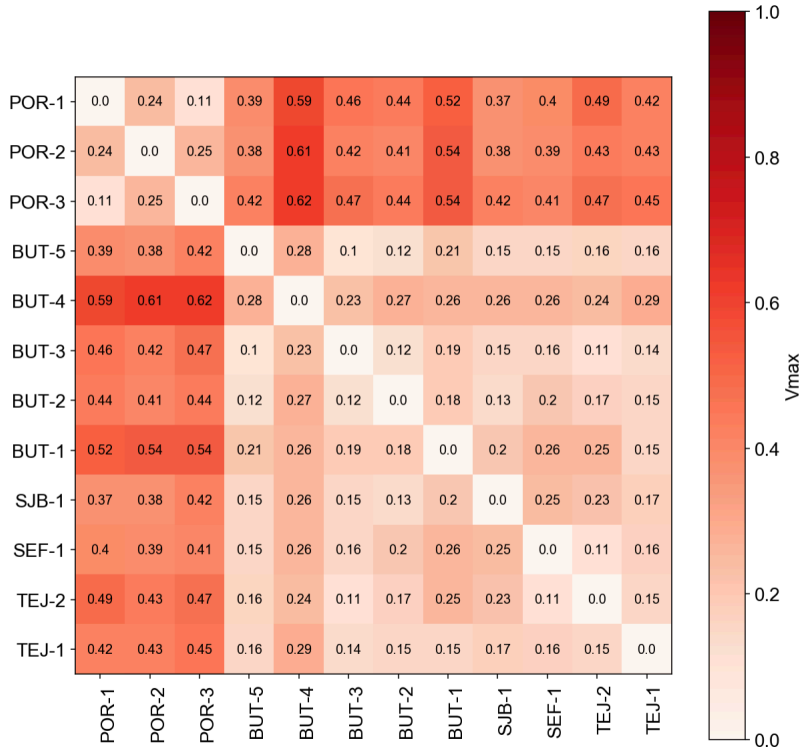
Plot a heat map

A heat map plot can be used to visualize the dissimilarity metric being used, or the Euclidean distance between sample pairs in MDS space.

- **'figsize'** (specify the width and height of the figure, inches; default value = (10, 10))
- **'savePlot'** (set to True to export a PDF of the plot; default value = True)
- **'fileName'** (the name of the file used when exporting; only relevant if savePlot = True; default value = 'HeatMapPlot.pdf')
- **'plotValues'** (set to True to plot values on the heat map; default value = True)
- **'plotType'** (set to 'dissimilarity' to plot the dissimilarity values; set to 'distance' to plot the Euclidean distance values in MDS space; default value = 'dissimilarity')
- **'fontsize'** (specifies the size of the font used if plotValues = True; default value = 10)

- **'halfMatrix'** (set to True to only plot half of the matrix; default value = True)

```
[22]: model.heatMap(figsize=(10,10), savePlot=False, fileName='HeatMapPlot.pdf', plotValues=True,
                    plotType='dissimilarity', fontsize=10)
```

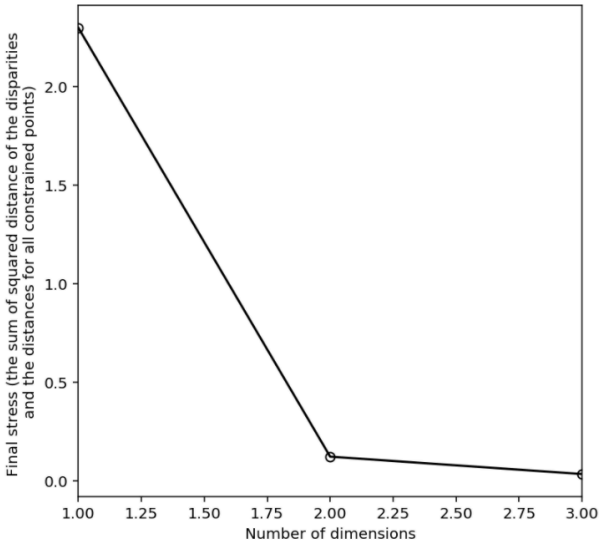


Stress plot

The stress plot displays the final model stress (i.e., goodness-of-fit) versus the number of dimensions modeled (specified by **'min_dim'** and **'max_dim'** variables). Stress typically decreases with increasing number of dimensions modeled.

- **'figsize'** (specify the width and height of the figure, inches; default value = (6, 6))
- **'savePlot'** (set to True to export a PDF of the plot; default value = True)
- **'fileName'** (the name of the file used when exporting; only relevant if savePlot = True; default value = 'stressPlot.pdf')
- **'stressType'** (Set to 'sklearn' to use the stress as calculated by the sklearn.Manifold.MDS module: "The final value of the stress (sum of squared distance of the disparities and the distances for all constrained points)". This article has additional information.
    - As of v.1.3.18, the 'Stress-1' option is no longer enabled. A warning will be returned if 'stressType = 'Stress-1'.

```
[89]: model.stressPlot(figsize=(6,6), savePlot=False, fileName='stressPlot.pdf', stressType='sklearn')
```

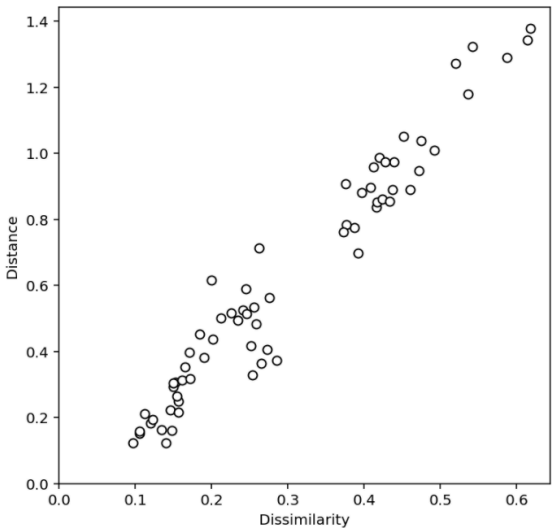


Shepard plot

Create a plot of Euclidean distance (i.e., sample-to-sample distance on the MDS plot) versus dissimilarity value. Samples that are similar should plot closer together than samples that are more different.

- **'figsize'** (specify the width and height of the figure, inches; default value = (6, 6))
- **'savePlot'** (set to True to export a PDF of the plot; default value = True)
- **'fileName'** (the name of the file used when exporting; only relevant if savePlot = True; default value = 'shepardPlot.pdf')
- **'plotOneToOneLine'** (set to True to plot a 1:1 line; default value = False)
- **'equalAspect'** (set to True to display y- and x-axes at the same scale; default value = False)

```
[90]: model.shepardPlot(figsize=(6,6), savePlot=False, fileName='shepardPlot.pdf', plotOneToOneLine=False)
```



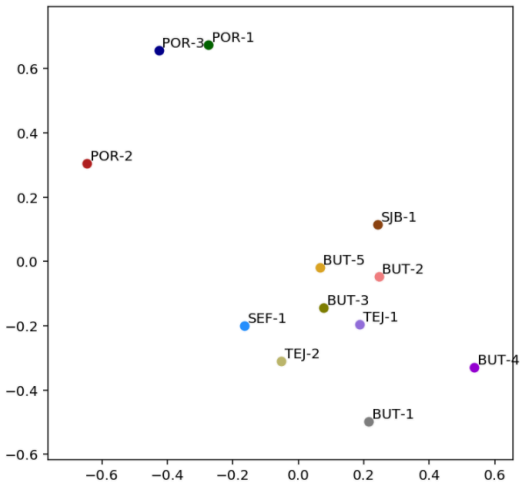
MDS plot

The MDS plot is a depiction of sample similarity and dissimilarity (refer to Vermeesch, 2013: Chemical Geology for a more complete description).

- **'figsize'** (specify the width and height of the figure, inches; default value = (6, 6))
- **'savePlot'** (set to True to export a PDF of the plot; default value = True)
- **'fileName'** (the name of the file used when exporting; only relevant if savePlot = True; default value = 'MDSplot.pdf')
- **'plotLabels'** (set to True to plot the Sample_ID next to each point; default value = True)
- **'equalAspect'** (set to True to display y- and x-axes at the same scale; default value = False)
- **'stressType'** (select which type of stress value to display above the MDS plot; see description of 'stressType' variable in the Stress plot section above for an explanation; default value = 'sklearn')

```
[91]: model.MDSplot(figsize=(6,6), savePlot=False, fileName='MDSplot.pdf', plotLabels=True, equalAspect=False,
                    stressType='sklearn')
```

```
Final stress:  0.12225806546055401
```

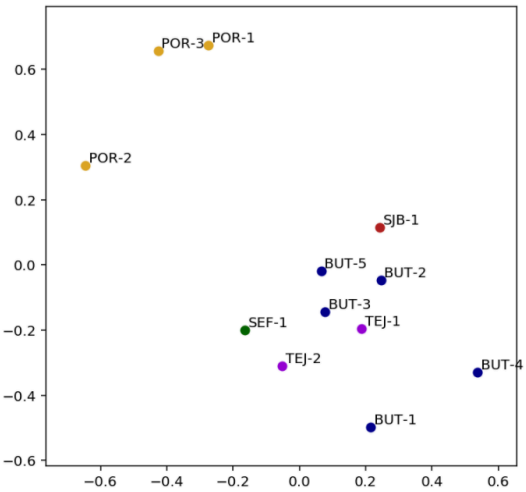


The MDS plot can be colored according to a categorical variable (e.g., 'Unit' or 'Basin').

- **'colorBy'** (specify the column name to use in coloring data points)
- **'df'** (specify the Pandas DataFrame that contains the categorical data)

```
[92]: model.MDSplot(figsize=(6,6), savePlot=False, fileName='MDSplot.pdf', plotLabels=True, colorBy='Unit',
                    df=main_byid_df, equalAspect=False, stressType='sklearn')
```

```
Final stress:  0.12225806546055401
```



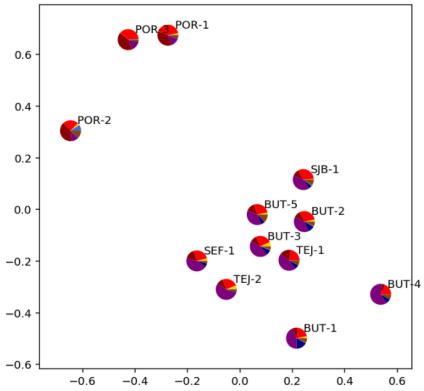
Samples can be plotted as pie diagram where bins correspond to different age fractions.

- **'plotPie'** (set to True to plot samples as pie diagrams)
- **'pieType'** (set to 'Age' to allow pie binsn to correspond to different age fractions)

- **'agebins'** (specify the age categories for pie plots)
- **'agebinsc'** (specify the colors of each age category)

```
[94]: # Sharman et al. 2015 scheme
agebins = [0, 23, 65, 85, 100, 135, 200, 300, 500, 4500]
agebinsc = ['slategray','royalblue','gold','red','darkred','purple','navy','gray','saddlebrown']

model.MDSplot(figsize=(6,6), savePlot=False, fileName='MDSplot.pdf', plotLabels=True,
              plotPie=True, pieType='Age', pieSize=0.04, agebins=agebins, agebinsc=agebinsc, equalAspect=False, stressType='sklearn')
```

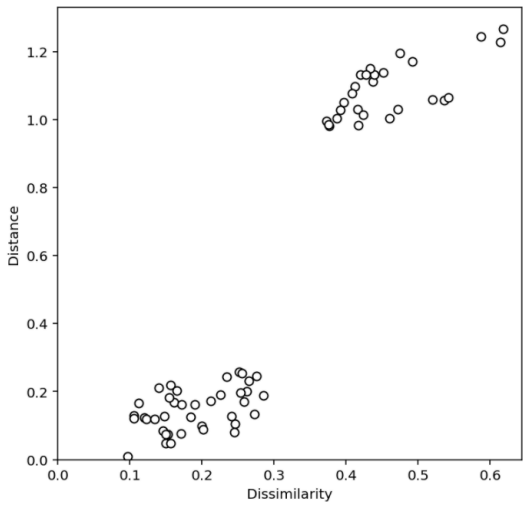
```
Final stress:  0.12225806546055401
```



The preceding outputs were based on non-metric MDS that was initialized with metric MDS as a starting configuration (the default option in v.1.3.18). Non-metric MDS can also be performed by repeated random initializations (the default option in v.1.3.17). For some datasets, a lower final stress may result using this procedure, at a price of increased computational time.

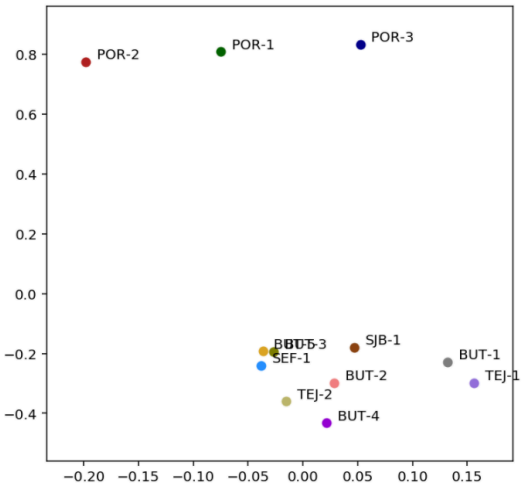
```
[95]: model = dFunc.MDS_class(ages, errors, labels, sampleList, metric=False, criteria='Vmax', bw='optimizedFixed', n_init=300,
                              max_iter=1000, x1=0, x2=4500, xdif=1, min_dim=1, max_dim=3, dim=2)
```

```
[97]: model.shepardPlot(figsize=(6,6), savePlot=False, fileName='shepardPlot.pdf', plotOneToOneLine=False)
```



```
[98]: model.MDSplot(figsize=(6,6), savePlot=False, fileName='MDSplot.pdf', plotLabels=True, equalAspect=False,
                    stressType='sklearn')
```

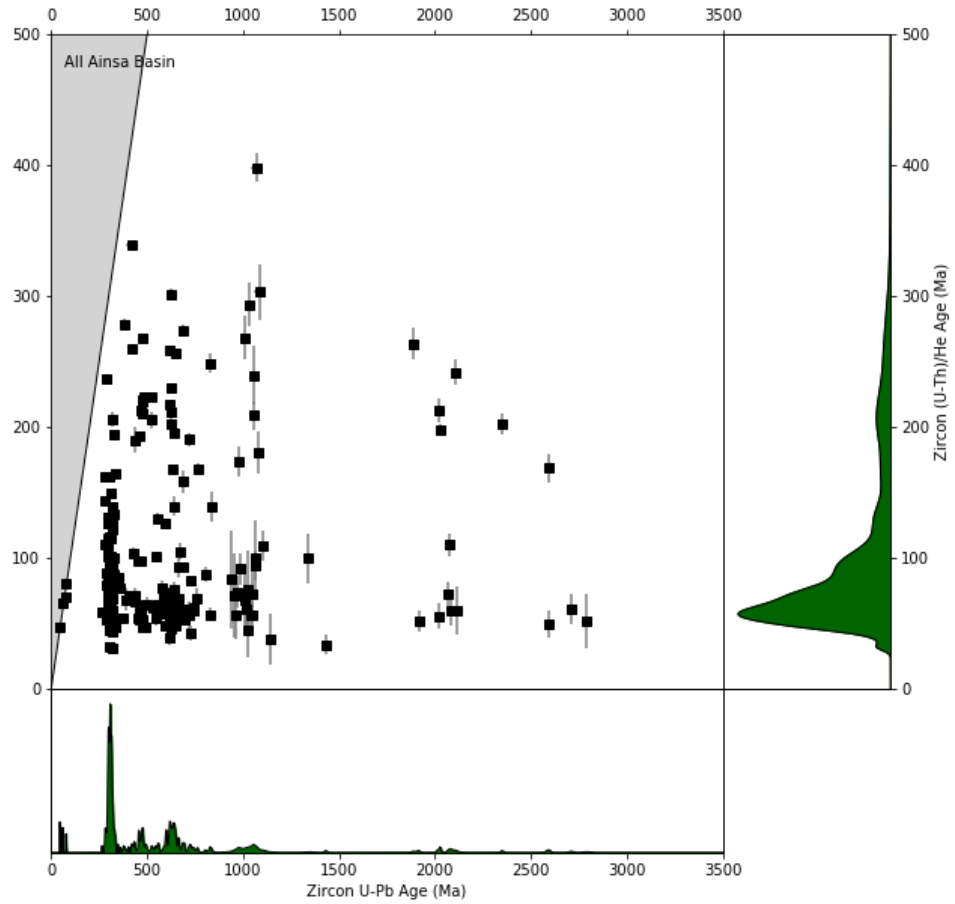
```
Final stress:  0.0728379036122038
```



(U-Th)/He vs U-Pb age "double dating" plot

For grains that have been "double dated" (e.g., a detrital zircon having both a U-Pb crystallization age and a (U-Th)/He cooling age), cooling ages can be plotting against grain crystallization ages. The default column names in the **"ZrUPb"** worksheet for the U-Pb crystallization age and analytical uncertainty are **"BestAge"** and **"BestAge_err"**, respectively. The default column names for the (U-Th)/He cooling age and analytical uncertainty are **"ZHe_Age"** and **"ZHe_Age_err"**, respectively.

The plotting extents of the x-axis and y-axis can be specified via the variables **'x1'**, **'x2'**, **'y1'**, and **'y2'**. Other plotting options are the same as described above in the **"Plot detrital age distributions"** section. Set the variable **'savePlot'** equal to **True** to save output plots as PDF files in the Output folder. A separate PDF file will be created for each sample or sample group plotted.



Export sample comparison matrices as a CSV file

A number of metrics have been proposed to assess the similarity and/or dissimilarity of detrital age distributions (see Saylor and Sundell, 2016). detritalPy current allows computation of five different metrics for a given list of samples or sample groups: similarity, likeness, the Kolgomorov-Smirnov statistic, the Kuiper statistic, and the cross-correlation (r^2) of either the KDE or PDP. In addition, the name of the output CSV file can be specified via the **'fileName'** variable.

The similarity, likeness, and cross-correlation (r^2) metrics requires selection of the type of relative distribution (either KDE or PDP), specified via the variable **'distType'**. By default, all sample comparison metrics are calculated over the entire age distribution from 0 to 4500 Ma, in 1 Myr bins. Note that the cross-correlation (r^2) metric is not independent of the age range it is computed over.

	A	B	C	D	E
1					
2	Similarity				
3	Label	n	POR-1	POR-2	POR-3
4	POR-1	100	1	0.915357	0.965365
5	POR-2	97	0.915357	1	0.899814
6	POR-3	101	0.965365	0.899814	1
7					
8	Likeness				
9	Label	n	POR-1	POR-2	POR-3
10	POR-1	100	1	0.802846	0.919585
11	POR-2	97	0.802846	1	0.774469
12	POR-3	101	0.919585	0.774469	1
13					
14	Kolgomorov-Smirnov Dmax				
15	Label	n	POR-1	POR-2	POR-3
16	POR-1	100	0	0.125258	0.070792
17	POR-2	97	0.125258	0	0.134837
18	POR-3	101	0.070792	0.134837	0
19					
20	Kolgomorov-Smirnov p-value				
21	Label	n	POR-1	POR-2	POR-3
22	POR-1	100	1	0.398374	0.956172
23	POR-2	97	0.398374	1	0.307348
24	POR-3	101	0.956172	0.307348	1
25					
26	Kuiper Vmax				
27	Label	n	POR-1	POR-2	POR-3
28	POR-1	100	0	0.240825	0.105149
29	POR-2	97	0.240825	0	0.251914
30	POR-3	101	0.105149	0.251914	0
31					
32	Cross-correlation of kernal density estimation				
33	Label	n	POR-1	POR-2	POR-3
34	POR-1	100	1	0.952206	0.989297
35	POR-2	97	0.952206	1	0.939741
36	POR-3	101	0.989297	0.939741	1

Export detrital age distributions as a CSV file

Raw detrital age distributions can be exported as a CSV file. The **'exportType'** variable selects the distribution type to export: a CDF, PDP, or KDE. If **'cumulative'** equals **True**, then the a CPDP or CKDE will be exported, if **'exportType'** equals **'PDP'** or **'KDE'**, respectively.

The **'normalize'** variable specifies whether to require the exported distribution to sum-to-1. Other variables are the same as described above in the **"Plot detrital age distributions"** section.

Export ages and error in tabular format as a CSV file

Some toolsets for analyzing detrital geochronologic data require ages and analytical uncertainties to be arranged adjacent to each other (e.g., Arizona LaserChron Center Excel worksheets). detritalPy provides a function for creating a CSV file that contains U-Pb ages and analytical uncertainties that are automatically sorted from youngest to oldest and listed in

adjacent columns. This script can be used to quickly export sample or sample group data for use in other software that require this data format.

	A	B	C	D	E	F
1	POR-1		POR-2		POR-3	
2	Age	Error	Age	Error	Age	Error
3	43.43501	0.810038	42.9692	0.800852	45.13831	6.879499
4	81.55798	10.03819	46.70842	0.651555	85.16718	3.147989
5	84.24094	3.583873	48.22694	0.999377	86.87153	1.769631
6	87.67574	4.462487	48.4173	0.921306	87.89	1.499487
7	88.45866	1.037609	49.10836	0.547924	88.45253	2.774341
8	90.11458	1.904244	49.14462	0.465099	89.01756	3.89697
9	90.14936	13.35084	49.48531	3.919021	89.33417	4.37469
10	90.41347	2.368385	51.22086	1.010155	89.87122	2.064821
11	91.04729	1.894384	54.65589	0.310222	90.01312	3.391591
12	91.04854	2.177847	72.05458	0.508739	90.27415	1.983233
13	91.58732	2.714647	77.91566	2.253699	90.2928	1.928168
14	91.60503	2.575328	80.16965	3.378144	90.66179	1.944734
15	91.89105	2.741065	85.20392	3.368808	91.04282	3.387784
16	91.91394	2.177761	85.78571	2.922958	91.45137	0.832607
17	91.96467	2.047263	86.19213	0.445216	91.85895	1.883917
18	92.4411	1.006607	88.27203	0.894238	91.96271	1.305136
19	93.17971	3.428009	89.76161	1.720394	91.9941	2.372452
20	93.22836	4.535769	90.52889	0.611294	92.04241	1.815038
21	93.23345	5.769852	91.12967	1.257791	92.86932	2.73986
22	93.3184	4.578497	91.60579	0.509365	92.87973	3.364401
23	93.55453	4.7254	92.2196	1.822106	92.91784	6.760237

An example of the first 21 detrital analyses exported for samples POR-1, POR-2, and POR-3 (see example data).

REFERENCES

Anderson, T., Kristoffersen, M. and **Elburg, M.A.** (2018) Visualizing, interpreting and comparing detrital zircon age and Hf isotope data in basin analysis – a graphical approach: *Basin Research*, **30**, 132-147.

Dickinson, W.R. and **Gehrels, G.E.** (2009) Use of U-Pb ages of detrital zircons to infer maximum depositional ages of strata: A test against a Colorado Plateau Mesozoic database: *Earth and Planetary Science Letters*, **288**, 115-125.

Sharman, G.R., Graham, S.A., Grove, M. and **Hourigan, J.K.** (2013) A reappraisal of the early slip history of the San Andreas fault, central California, USA: *Geology*, **41**, 727-730.

Sharman, G.R., Graham, S.A., Grove, M., Kimbrough, D.L. and **Wright, J.E.** (2015) Detrital Zircon Provenance of the Late Cretaceous-Eocene California Forearc: Influence

of Laramide Low-Angle Subduction on Sediment Dispersal and Paleogeography: *Geological Society of America Bulletin*, **127**, 38-60.

Sharman, G.R., Sharman, J.P. and **Sylvester, Z.** (2018) A Python-based Toolset for Visualizing and Analyzing Detrital Geo-Thermochronologic Data: *The Depositional Record*, v. 4, p. 202-215, https://doi.org/10.1002/dep2.45.

Saylor, J.E. and **Sundell, K.E.** (2016) Quantifying comparison of large detrital geochronology data sets: *Geosphere*, **12**, 203-220.