pyBacktrack Documentation

Release 1.5.0.dev8

John Cannon

Nov 20, 2023

CONTENTS

A tool for reconstructing paleobathymetry on oceanic and continental crust.

PyBacktrack is a Python package that backtracks the paleo-water depth of ocean drill sites through time by combining a model of tectonic subsidence with decompaction of the site stratigraphic lithologies. PyBacktrack can also include the effects of mantle-convection driven dynamic topography on paleo-water depth, as well as sea-level variations. PyBacktrack provides a model of tectonic subsidence on both oceanic and continental crust. Ocean crust subsidence is based on a user-selected lithospheric age-depth model and the present-day unloaded basement depth. Continental crust subsidence is based on syn-rift and post-rift subsidence that is modelled using the total sediment thickness at the site and the timing of the transition from rifting to thermal subsidence. At drill sites that did not penetrate to basement, the age-coded stratigraphy is supplemented with a synthetic stratigraphic section that represents the undrilled section, whose thickness is estimated using a global sediment thickness map. This is essential for estimating the decompacted thickness of the total sedimentary section, and thus bathymetry, through time. At drill sites on stretched continental crust where the paleo-water depth is known from benthic fossil assemblages, tectonic subsidence can be computed via backstripping. The workflow is similar to backtracking, but paleo-water depths and their uncertainties need to be supplied as part of the input. In addition to individual 1D drill sites, all submerged present-day crust (assigned a single lithology) can be backtracked and reconstructed to generate 2D paleobathymetry grids through time.

CHAPTER ONE

REFERENCE

The following paper covers the theory and algorithms of pyBacktrack:

- Muller, R. D., Cannon, J., Williams, S. and Dutkiewicz, A., 2018, PyBacktrack 1.0: A Tool for Reconstructing Paleobathymetry on Oceanic and Continental Crust, **Geochemistry, Geophysics, Geosystems,** 19, 1898-1909, doi: 10.1029/2017GC007313

Note: The paper can be downloaded either at Geochemistry, Geophysics, Geosystems or ResearchGate.

CHAPTER

TWO

CONTENTS

2.1 Getting Started

2.1.1 Installation

Install pybacktrack

You can install `pybacktrack` using:

1. *conda*, or
2. *pip*, or
3. *Docker*.

We recommend using *conda* since it installs all the dependencies of `pybacktrack` (using *pip* currently only installs some of the dependencies, the rest must be install manually). Using *Docker* is also more straightforward than `pip` since all the dependencies have been pre-installed.

Using conda

We recommend installing pyBacktrack using conda.

To install the latest stable version of pyBacktrack type the following in a terminal or command window (on macOS and Ubuntu this is a *Terminal* window, and on Windows you'll need to open an *Anaconda prompt* from the Start menu):

```
conda install -c conda-forge pybacktrack
```

We recommend installing pyBacktrack into a new conda environment. For example, the following creates and activates a Python 3.10 environment named `pybacktrack_py310` containing pyBacktrack and all its dependencies:

```
conda create -n pybacktrack_py310 -c conda-forge python=3.10 pybacktrack
conda activate pybacktrack_py310
```

You can then use pyBacktrack. For example, to see the pyBacktrack version:

```
python -c "import pybacktrack; print(pybacktrack.__version__)"
```

Using pip

Python packages installed using pip will typically also have their dependency packages automatically installed also. However `pybacktrack` requires manual installation of some of its dependencies.

Requirements

PyBacktrack depends on:

- NumPy
- SciPy
- Generic Mapping Tools (GMT) (>=5.0.0)
- PyGPlates

NumPy and *SciPy* are automatically installed by *pip* when *pybacktrack is installed*, however *GMT* (version 5 or above) and *pyGPlates* need to be manually installed.

GMT is called via the command-line (shell) and so just needs to be in the PATH in order for *pyBacktrack* to find it. Also ensure that version 5 or above (supports NetCDF version 4) is installed since the *bundled grid files in pyBacktrack* are in NetCDF4 format.

PyGPlates is not currently installable as a package and so needs to be in the python path (sys.path or PYTHONPATH). Installation instructions are available here.

PyGPlates supports Python 3 (in addition to Python 2.7) so you can now use pyBacktrack with either. The *Macports install example* below shows one approach to selecting the default Python using `sudo port select`. Another approach is using Python virtual environments where each environment has its own `python`, `pip` and installed packages. However, currently pyGPlates does not yet work in virtual environments (at least on Mac systems).

Install Python, Pip, GMT and pyGPlates on Ubuntu

This is an example demonstrating how to install GMT and pyGPlates on Ubuntu 18.04 (Bionic).

Note: The main difference for other Ubuntu versions will be the pyGPlates install package (you'll need to select the package appropriate for your Ubuntu version).

First install GMT 5:

```
sudo apt install gmt
```

Then install Python 3 (and Pip):

```
sudo apt update

sudo apt install python3 python3-pip
sudo pip3 install --upgrade pip
```

Then download the pyGPlates Python 3 debian package pygplates_0.36.0_py36_ubuntu-18.04-amd64.deb, and install it:

```
sudo apt install ./pygplates_0.36.0_py36_ubuntu-18.04-amd64.deb
```

Then add the installed location of pyGPlates to the PYTHONPATH environment variable:

```
export PYTHONPATH=$PYTHONPATH:/usr/lib
```

Install Python, Pip, GMT and pyGPlates on Mac using Macports

This is an example demonstrating how to install GMT and pyGPlates on a Mac system using Macports.

First install GMT 5:

```
sudo port install gmt5
```

Note: You will likely need to add `/opt/local/lib/gmt5/bin/` to your `PATH` environment variable, for example in your `~/.bashrc`, `~/.bash_profile` or `~/.zprofile` file so that PATH is set each time you open a new terminal window. After doing this, typing `gmt` should find GMT and show some help options.

Then install Python 3 (and Pip):

```
sudo port install python38
sudo port install py38-pip
```

Set your default `python` to Python 3.8:

```
sudo port select --set python python38
sudo port select --set pip pip38
```

Note: If you already have `python` referencing Python 2 then you can instead use `python3` to reference Python 3:

```
sudo port select --set python3 python38
sudo port select --set pip3 pip38
```

. . . but this will require using `python3` on the command-line to run *pybacktrack* (instead of just `python`).

Then download a pyGPlates Mac zip file, such as pygplates_0.36.0_py38_Darwin-x86_64.zip for Python 3.8 on an Intel Mac, and extract it to your home directory.

Then add the unzipped location of pyGPlates to the PYTHONPATH environment variable, such as:

```
export PYTHONPATH=~/pygplates_0.36.0_py38_Darwin-x86_64:$PYTHONPATH
```

Note: The above line can be added to your `~/.bashrc`, `~/.bash_profile` or `~/.zprofile` file so that PYTHONPATH is set each time you open a new terminal window.

Install pybacktrack

To install the latest stable version, run:

```
python -m pip install pybacktrack
```

Warning:

On Mac systems, when using Macports, it might be better to install to the local user install directory with `python -m pip install --user pybacktrack` to avoid confusing Macports (which installs to the system install directory).

And on linux systems, if you have admin privileges, you can install to the system install directory with `sudo python -m pip install pybacktrack`.

Note: We generally recommend using `python -m pip install pybacktrack` instead of `pip install pybacktrack` to ensure `pybacktrack` is installed into the `python` you are actually using. For example, when using Conda Python it might be that `python` executes the Conda Python interpreter but `pip` installs into the system Python (eg, because the base Conda environment is not activated).

If you already have `pybacktrack` installed and would like to upgrade to the latest version then use the `--upgrade` flag:

```
python -m pip install --upgrade pybacktrack
```

To install the latest development version (requires Git on local system), run:

```
python -m pip install "git+https://github.com/EarthByte/pyBacktrack.git#egg=pybacktrack"
```

Note:

You may need to update your *Git* if you receive an error ending with `tlsv1 alert protocol version`.
This is apparently due to an update on GitHub.

…or download the pyBacktrack source code, extract to a local directory and run:

```
python -m pip install <path-to-local-directory>
```

Note: Installing *pyBacktrack* will automatically install the *NumPy* and *SciPy* *requirements*. However, as mentioned in *requirements*, *GMT* and *pyGPlates* still need to be manually installed.

Using Docker

This method of running `pybacktrack` relies on Docker, so before installing the `pybacktrack` docker image, ensure you have installed Docker.

Note:

On Windows platforms you can install Docker Desktop for Windows. Note that Docker Toolbox has been deprecated (and now *Docker Desktop for Windows* is recommended).
A similar situation applies on Mac platforms where you can install Docker Desktop for Mac (with *Docker Toolbox* being deprecated).

Once Docker is installed, open a terminal (command-line interface).

Note:

For Docker Desktop for Windows and Docker Desktop for Mac this a regular command-line terminal.
Also on Linux systems this a regular command-line terminal.

To install the `pybacktrack` docker image, type:

```
docker pull earthbyte/pybacktrack
```

To run the docker image:

```
docker run -it --rm -p 18888:8888 -w /usr/src/pybacktrack earthbyte/pybacktrack
```

This should bring up a command prompt inside the running docker container.
The current working directory should be `/usr/src/pybacktrack/`.
It should have a `pybacktrack_examples` sub-directory containing test data.

Note: On Linux systems you may have to use *sudo* when running *docker* commands. For example:

```
sudo docker pull earthbyte/pybacktrack
sudo docker run -it --rm -p 18888:8888 -w /usr/src/pybacktrack earthbyte/pybacktrack
```

From the current working directory you can run the *backtracking example* below, or any *other examples* in this documentation. For example, you could run:

```
python3 -m pybacktrack.backtrack_cli -w pybacktrack_examples/example_data/ODP-114-699-
↪Lithology.txt -d age water_depth -- ODP-114-699_backtrack_decompacted.txt
```

If you wish to run the example notebooks then there is a `notebook.sh` script to start a Jupyter notebook server in the running docker container:

```
./notebook.sh
```

Then you can start a web browser on your local machine and type the following in the URL field:

```
http://localhost:18888/tree
```

This will display the current working directory in the docker container.
In the web browser, navigate to `pybacktrack_examples` and then `notebooks`.
Then click on a notebook (such as backtrack.ipynb).
You should be able to run the notebook, or modify it and then run it.

Install the examples

Before running the example below, or any *other examples*, you'll also need to install the example data (from the pybacktrack package itself). This assumes you've already *installed pybacktrack*.

The following command installs the examples (example data and notebooks) to a new sub-directory of your *current working directory* called `pybacktrack_examples`:

```
python -c "import pybacktrack; pybacktrack.install_examples()"
```

Note: The *current working directory* is whatever directory you are in when you run the above command.

Note:

Alternatively you can choose a different sub-directory by providing an argument to the `install_examples()` function above.
For example, `python -c "import pybacktrack; pybacktrack.install_examples('pybacktrack/examples')"` creates a new sub-directory of your *current working directory* called `pybacktrack/examples`.
However the example below assumes the default directory (`pybacktrack_examples`).

Install supplementary scripts

You can optionally install supplementary scripts. These are not necessary for running the `pybacktrack` module. They are various pre/post processing, conversion and test scripts that have only been included for reference (for those interested).

The following command installs the supplementary scripts to a new sub-directory of your *current working directory* called `pybacktrack_supplementary`:

```
python -c "import pybacktrack; pybacktrack.install_supplementary()"
```

Note: Like *the examples* you can specify your own sub-directory.

2.1.2 A Backtracking Example

Once *installed*, `pybacktrack` is available to:

1. run built-in scripts (inside `pybacktrack`), or
2. `import pybacktrack` into your own script.

The following example is used to demonstrate both approaches. It backtracks an ocean drill site and saves the output to a text file by:

- reading the ocean drill site file `pybacktrack_examples/example_data/ODP-114-699-Lithology.txt`,

  Note:

  This file is part of the *example data*.
  However if you have your own ocean drill site file then you can substitute it in the example below if you want.

- backtracking it using:
  - the `M2` dynamic topography model, and
  - the `Haq87_SealevelCurve_Longterm` sea-level model,
- writing the amended drill site to `ODP-114-699_backtrack_amended.txt`, and
- writing the following columns to `ODP-114-699_backtrack_decompacted.txt`:
  - age
  - compacted_depth
  - compacted_thickness
  - decompacted_thickness
  - decompacted_density
  - decompacted_sediment_rate
  - decompacted_depth
  - dynamic_topography
  - water_depth
  - tectonic_subsidence

  - lithology

Use a built-in module script

Since there is a backtrack module inside pybacktrack that can be run as a script, we can invoke it on the command-line using `python -m pybacktrack.backtrack_cli` followed by command line options that are specific to that module. This is the easiest way to run backtracking.

To see its command-line options, run:

```
python -m pybacktrack.backtrack_cli --help
```

The backtracking example can now be demonstrated by running the script as:

```
python -m pybacktrack.backtrack_cli \
    -w pybacktrack_examples/example_data/ODP-114-699-Lithology.txt \
    -d age compacted_depth compacted_thickness decompacted_thickness decompacted_density␣
↪decompacted_sediment_rate decompacted_depth dynamic_topography water_depth tectonic_
↪subsidence lithology \
    -ym M2 \
    -slm Haq87_SealevelCurve_Longterm \
    -o ODP-114-699_backtrack_amended.txt \
    -- \
    ODP-114-699_backtrack_decompacted.txt
```

Import into your own script

An alternative to running a built-in script is to write your own script (using a text editor) that imports pybacktrack and calls its functions. You might do this if you want to combine pyBacktrack functionality with other research functionality into a single script.

The following Python code does the same as the *built-in script* by calling the *pybacktrack.backtrack_and_write_well()* function:

```
import pybacktrack

# Input and output filenames.
input_well_filename = 'pybacktrack_examples/example_data/ODP-114-699-Lithology.txt'
amended_well_output_filename = 'ODP-114-699_backtrack_amended.txt'
decompacted_output_filename = 'ODP-114-699_backtrack_decompacted.txt'

# Read input well file, and write amended well and decompacted results to output files.
pybacktrack.backtrack_and_write_well(
    decompacted_output_filename,
    input_well_filename,
    dynamic_topography_model='M2',
    sea_level_model='Haq87_SealevelCurve_Longterm',
    # The columns in decompacted output file...
    decompacted_columns=[pybacktrack.BACKTRACK_COLUMN_AGE,
                         pybacktrack.BACKTRACK_COLUMN_COMPACTED_DEPTH,
                         pybacktrack.BACKTRACK_COLUMN_COMPACTED_THICKNESS,
                         pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_THICKNESS,
                         pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DENSITY,
```

(continues on next page)

(continued from previous page)

```
                         pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_SEDIMENT_RATE,
                         pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DEPTH,
                         pybacktrack.BACKTRACK_COLUMN_DYNAMIC_TOPOGRAPHY,
                         pybacktrack.BACKTRACK_COLUMN_WATER_DEPTH,
                         pybacktrack.BACKTRACK_COLUMN_TECTONIC_SUBSIDENCE,
                         pybacktrack.BACKTRACK_COLUMN_LITHOLOGY],
    # Might be an extra stratigraphic well layer added from well bottom to ocean␣
↪basement...
    ammended_well_output_filename=amended_well_output_filename)
```

If you save the above code to a file called `my_backtrack_script.py` then you can run it as:

```
python my_backtrack_script.py
```

2.2 Overview

This document gives a brief overview of the scripts inside the `pybacktrack` package.

2.2.1 Running pyBacktrack

Once *installed*, the `pybacktrack` Python package is available to:

1. *run built-in scripts* (inside `pybacktrack`), or
2. `import pybacktrack` *into your own script*.

It is generally easier to run the built-in scripts since you only need to specify parameters on the command-line.

However you may need to create your own script if you want to combine `pybacktrack` functionality with other research functionality. In this case it is generally better to `import pybacktrack`, along with the other modules, into your own script. This also gives a finer granularity of control compared to the command-line.

The following two sections give an overview of both approaches.

Note:

The input files used in the examples below (except *interpolate*) are available in the example data.
Please ensure you have *installed the example data* before running any of these examples.

2.2.2 Running the scripts built into pyBacktrack

PyBacktrack is a Python package containing modules. And each module can be run as a script using `python -m pybacktrack.<module>_cli` followed by command line options that are specific to that module. For example, the `backtrack` module can be run as `python -m pybacktrack.backtrack_cli ...`, or the `backstrip` module can be run as `python -m pybacktrack.backstrip_cli ...`, with `...` replaced by command-line options.

The following sections give an introduction to each module.

Note: In each module you can use the `--help` option to see all available command-line options for that specific module. For example, `python -m pybacktrack.backtrack_cli --help` describes all options available to the `backtrack` module.

backtrack

The `backtrack` module is used to find paleo water depths from a tectonic subsidence model (such as an age-to-depth curve in ocean basins, or rifting near continental passive margins) and sediment decompaction over time.

This example takes an ocean drill site as input and outputs a file containing a backtracked water depth for each age in the drill site:

```
python -m pybacktrack.backtrack_cli -w pybacktrack_examples/example_data/ODP-114-699-
↪Lithology.txt -d age water_depth -- ODP-114-699_backtrack_decompacted.txt
```

...where the `-w` option specifies the input drill site file `pybacktrack_examples/example_data/ODP-114-699-Lithology.txt`, the `-d` option specifies the desired columns (`age` and `water_depth`) of the output file, and `ODP-114-699_backtrack_decompacted.txt` is the output file.

There are other command-line options available to the `backtrack` module (use the `--help` option to list them) but they all have default values and hence only need to be specified if the default does not suit.

See also:

Backtrack

backstrip

The backstrip module is used to find tectonic subsidence (typically due to lithospheric stretching) from paleo water depths and sediment decompaction over time.

This example takes a passive margin site as input and outputs a file containing a backstripped tectonic subsidence for each age in the drill site:

```
python -m pybacktrack.backstrip_cli -w pybacktrack_examples/example_data/sunrise_
↪lithology.txt -l primary extended -d age average_tectonic_subsidence -- sunrise_
↪backstrip_decompacted.txt
```

…where the `-w` option specifies the input drill site file `pybacktrack_examples/example_data/sunrise_lithology.txt`, the `-l` option specifies the lithology definitions, the `-d` option specifies the desired columns (`age` and `average_tectonic_subsidence`) of the output file, and `sunrise_backstrip_decompacted.txt` is the output file.

Note: It is necessary to specify the bundled `primary` and `extended` lithology definitions, with `-l primary extended`, because the input drill site references lithologies in both lithology definition files. See *Bundled lithology definitions*. This is unlike the *backtracking example* above that only references the `primary` lithologies, and hence does not need to specify lithology definitions because `primary` is the default (when `-l` is not specified).

Note: `average_tectonic_subsidence` is an *average* of the minimum and maximum tectonic subsidences, that are in turn a result of the minimum and maximum water depths specified in the drill site file.

There are other command-line options available to the backstrip module (use the `--help` option to list them) but they all have default values and hence only need to be specified if the default does not suit.

See also:

Backstrip

paleo_bathymetry

The paleo_bathymetry module is used to generate paleo bathymetry grids by reconstructing and backtracking present-day sediment-covered crust through time.

This example generates paleobathymetry grids at 12 minute resolution from 0Ma to 240Ma in 1Myr increments using the M7 *dynamic topography model* and the GDH1 *oceanic subsidence model*:

```
python -m pybacktrack.paleo_bathymetry_cli -gm 12 -ym M7 -m GDH1 --use_all_cpus -- 240␣
↪paleo_bathymetry_12m_M7_GDH1
```

…where the `-gm` option specifies the grid spacing (in minutes), the `-ym` specifies the dynamic topography model, the `-m` option specifies the oceanic subsidence model, the `--use_all_cpus` option uses all CPUs (it also accepts an optional number of CPUs) and the generated paleobathymetry grid files are named `paleo_bathymetry_12m_M7_GDH1_<time>.nc`.

There are other command-line options available to the paleo_bathymetry module (use the `--help` option to list them) but they all have default values and hence only need to be specified if the default does not suit.

See also:

Paleobathymetry

age_to_depth

The age_to_depth module is used to convert ocean floor age to ocean basement depth (in ocean basins).

This example takes an input file containing a column of ages, and outputs a file containing two columns (age and depth):

```
python -m pybacktrack.age_to_depth_cli -- pybacktrack_examples/example_data/ages.txt␣
↪ages_and_depths.txt
```

Here the input file pybacktrack_examples/example_data/ages.txt contains ages in the first (and only) column. If they had been in another column, for example if there were other unused columns, then we would need to specify the age column with the -a option.

The output file ages_and_depths.txt contains ages in the first column and depths in the second column. To reverse this order you can use the -r option.

There are three built-in age-to-depth ocean models:

- RHCW18 - Richards et al. (2020) Structure and dynamics of the oceanic lithosphere-asthenosphere system
- CROSBY_2007 - Crosby, A.G., (2007) Aspects of the relationship between topography and gravity on the Earth and Moon, PhD thesis
- GDH1 - Stein and Stein (1992) Model for the global variation in oceanic depth and heat flow with lithospheric age

Here the conversion was performed using the *default* model RHCW18 since the -m command-line option was not specified. However you can specify the alternate CROSBY_2007 model using -m CROSBY_2007 (or GDH1 using -m GDH1).

Note: The default age-to-depth model was updated in pyBacktrack version 1.4. It is now RHCW18. Previously it was GDH1.

Or you can use your own age-to-depth model by specifying a file containing an age column and a depth column followed by two integers representing the age and depth column indices. For example, if you have your own age-to-depth file called age-depth-model.txt where age is in the first column and depth is in the second column then you can specify this using -w age-depth-model.txt 0 1.

Note: Use python -m pybacktrack.age_to_depth_cli --help to see a description of all command-line options.

stratigraphic_depth_to_age

The stratigraphic_depth_to_age module is used to convert stratigraphic depths to ages using a depth-to-age model.

Here the depth-to-age model is specified as a file containing a column of ages and a column of depths that forms a piecewise linear function of age with depth (a model where age is a function of depth age=function(depth)). Then another file specifies the input stratigraphic depths that you wish to convert to ages. Finally a third file is created containing the input depths and output ages, where each interpolated output age is a result of querying the piecewise linear function using the input depth:

```
python -m pybacktrack.stratigraphic_depth_to_age_cli -m pybacktrack_examples/example_
↪data/Site1089B_age_depth.txt -- pybacktrack_examples/example_data/Site1089B_strat_
↪depth.txt Site1089B_age_strat_depth.txt
```

Here the `age=function(depth)` model is specified with the `-m` option, where the `pybacktrack_examples/example_data/Site1089B_age_depth.txt` file contains a column of ages and a column of depths. By default, age is the first column and depth the second but you can optionally choose any column by specifying two integers representing the age and depth column indices in the `-m` option. For example, you can change `-m pybacktrack_examples/example_data/Site1089B_age_depth.txt` to `-m pybacktrack_examples/example_data/Site1089B_age_depth.txt 1 0` to select the second column (index `1`) for age and the first column (index `0`) for depth.

The input stratigraphic depths are in `pybacktrack_examples/example_data/Site1089B_strat_depth.txt` and must be in the *first* column. Any text after the depth value in a row (eg, lithologies) is copied to the output file. Also any metadata at the top of the file is copied to the output file.

The interpolated ages and associated depths are written to the output file `Site1089B_age_strat_depth.txt`. The first column contains (interpolated) age and the second column contains depth. To reverse this order you can use the `-r` option.

Note: The output file `Site1089B_age_strat_depth.txt` does *not* contain rows for depths that are *outside* the depth range of the model `Site1089B_age_depth.txt`. This is the default behaviour. You can change this using the `-m` option which, in addition to specifying optional age and depth column indices, allows you to optionally specify how to handle out-of-bounds depth values with `exclude` (to exclude rows outside depth range), `clamp` (to use boundary age values) or `extrapolate` (to extrapolate age from boundary).

Note: Use `python -m pybacktrack.stratigraphic_depth_to_age_cli --help` to see a description of all command-line options.

interpolate

The `interpolate` module can perform linear interpolation of any piecewise linear function `y=f(x)`. As such it can be used for any type of data.

Here the `y=f(x)` model is specified as a file containing a column of *x* values and a column of *y* values that forms a piecewise linear function of *y* with *x*. Then another file specifies the input *x* values. Finally a third file is created containing the input *x* values and the output *y* values, where each interpolated output *y* value is a result of querying the piecewise linear function using an input *x* value:

```
python -m pybacktrack.util.interpolate_cli -cx 1 -cy 0 -c function_y_of_x.txt -- input_x_
↪values.txt output_x_y_values.txt
```

Note: These files, specifically `function_y_of_x.txt` and `input_x_values.txt`, do not exist in the *example data*. They are just placeholders for your own data that you would like to interpolate.

Here the `y=f(x)` model is specified with the `-c`, `-cx` and `-cy` options. The `-c` option specifies the file `function_y_of_x.txt` containing a column of `y` values followed by a column of `x` values. The `-cx` and `-cy` options specify the *x* and *y* columns of the model function `y=f(x)`. These default to `0` and `1` respectively. However if *y* happens to be in the first column (`0`) and *x* in the second column (`1`) then you can swap the default order of column indices using `-cx 1 -cy 0`.

The input `x` values are in `input_x_values.txt` in the first column (by default). If they had been in another column, for example if there were other unused columns, then we would need to specify the *x* column with the `-ix` option.

The output (interpolated) *y* values (and associated *x* values) are written to the output file `output_x_y_values.txt`. The first column contains the *x* values and the second column contains the (interpolated) *y* values. To reverse this order you can use the `-r` option.

Note: Use `python -m pybacktrack.util.interpolate_cli --help` to see a description of all command-line options.

2.2.3 Running your own script that imports pyBacktrack

An alternative to *running the built-in scripts* is to write your own script (using a text editor) that imports `pybacktrack` and calls its *functions*. You might do this if you want to combine pyBacktrack functionality with other research functionality into a single script.

The following shows Python source code that is equivalent to the above *examples running built-in scripts*.

If you save any of the code examples below to a file called `my_script.py` then you can run that example as:

```
python my_script.py
```

backtrack

The following Python source code (using *these functions*):

```
import pybacktrack

pybacktrack.backtrack_and_write_well(
    'ODP-114-699_backtrack_decompacted.txt',
    'pybacktrack_examples/example_data/ODP-114-699-Lithology.txt',
    decompacted_columns=[pybacktrack.BACKTRACK_COLUMN_AGE,
                         pybacktrack.BACKTRACK_COLUMN_WATER_DEPTH])
```

…is equivalent to *running the backtrack script example*:

```
python -m pybacktrack.backtrack_cli -w pybacktrack_examples/example_data/ODP-114-699-
↪Lithology.txt -d age water_depth -- ODP-114-699_backtrack_decompacted.txt
```

Note: The `backtrack` module is covered in more detail *here*.

backstrip

The following Python source code (using *these functions*):

```
import pybacktrack

pybacktrack.backstrip_and_write_well(
    'sunrise_backstrip_decompacted.txt',
    'pybacktrack_examples/example_data/sunrise_lithology.txt',
    lithology_filenames=[pybacktrack.PRIMARY_BUNDLE_LITHOLOGY_FILENAME,
                         pybacktrack.EXTENDED_BUNDLE_LITHOLOGY_FILENAME],
```

(continues on next page)

(continued from previous page)

```
    decompacted_columns=[pybacktrack.BACKSTRIP_COLUMN_AGE,
                         pybacktrack.BACKSTRIP_COLUMN_AVERAGE_TECTONIC_SUBSIDENCE])
```

…is equivalent to *running the backstrip script example*:

```
python -m pybacktrack.backstrip_cli -w pybacktrack_examples/example_data/sunrise_
↪lithology.txt -l primary extended -d age average_tectonic_subsidence -- sunrise_
↪backstrip_decompacted.txt
```

Note: The `backstrip` module is covered in more detail *here*.

paleo_bathymetry

The following Python source code (using *these functions*):

```
import pybacktrack

pybacktrack.reconstruct_paleo_bathymetry_grids(
    'paleo_bathymetry_12m_M7_GDH1',
    0.2,  # degrees (same as 12 minutes)
    240,
    dynamic_topography_model='M7',
    ocean_age_to_depth_model=pybacktrack.AGE_TO_DEPTH_MODEL_GDH1,
    use_all_cpus=True)  # can also be an integer (the number of CPUs to use)
```

…is equivalent to *running the paleobathymetry script example*:

```
python -m pybacktrack.paleo_bathymetry_cli -gm 12 -ym M7 -m GDH1 --use_all_cpus -- 240␣
↪paleo_bathymetry_12m_M7_GDH1
```

Note: The `paleo_bathymetry` module is covered in more detail *here*.

age_to_depth

The following Python source code (using *these functions*):

```
import pybacktrack

pybacktrack.convert_age_to_depth_files(
    'pybacktrack_examples/example_data/ages.txt',
    'ages_and_depths.txt')
```

…is equivalent to *running the age-to-depth script example*:

```
python -m pybacktrack.age_to_depth_cli -- pybacktrack_examples/example_data/ages.txt␣
↪ages_and_depths.txt
```

stratigraphic_depth_to_age

The following Python source code (using *these functions*):

```
import pybacktrack

# Read the age=f(depth) function, where 'x' is depth and 'y' is age (in the returned␣
↪function y=f(x)).
age_column_index = 0    # age is in the first column
depth_column_index = 1  # depth is in the second column
# This determines the age values for depth values outside the depth range of the depth-
↪to-model model.
# It can be 'exclude' to exclude age values outside range, or 'clamp' to use boundary age␣
↪values, or 'extrapolate' to extrapolate age from boundary.
# Here we use 'exclude' (instead of the default 'clamp') to avoid getting the same age␣
↪value for different depth values (outside depth range).
out_of_bounds = 'exclude'
# Ignore the x (depth) and y (age) values read from file by using '_'.
depth_to_age_model, _, _ = pybacktrack.read_interpolate_function('pybacktrack_examples/
↪example_data/Site1089B_age_depth.txt', depth_column_index, age_column_index, out_of_
↪bounds)

# Convert depth values in input file to age and depth values in output file.
pybacktrack.convert_stratigraphic_depth_to_age_files(
    'pybacktrack_examples/example_data/Site1089B_strat_depth.txt',
    'Site1089B_age_strat_depth.txt',
    depth_to_age_model)
```

…is equivalent to *running the stratigraphic depth-to-age script example*:

```
python -m pybacktrack.stratigraphic_depth_to_age_cli -m pybacktrack_examples/example_
↪data/Site1089B_age_depth.txt -- pybacktrack_examples/example_data/Site1089B_strat_
↪depth.txt Site1089B_age_strat_depth.txt
```

interpolate

The following Python source code (using *these functions*):

```
import pybacktrack

# Read the y=f(x) function from a 2-column file.
# Ignore the x and y values read from file by using '_'.
function_y_of_x, _, _ = pybacktrack.read_interpolate_function('function_y_of_x.txt', 1,␣
↪0)

# Convert x values in a 1-column input file to x and y values in a 2-column output file.
pybacktrack.interpolate_file(
    function_y_of_x,
    'input_x_values.txt',
    'output_x_y_values.txt')
```

…is equivalent to *running the interpolate script example*:

```
python -m pybacktrack.util.interpolate_cli -cx 1 -cy 0 -c function_y_of_x.txt -- input_x_
↪values.txt output_x_y_values.txt
```

2.3 Stratigraphy

This document covers drill site stratigraphy, and lithology names that reference lithology definitions of density, surface porosity and porosity decay.

2.3.1 Drill site

Both backtracking and backstripping involve sediment decompaction over time. So the main input file for backtracking and backstripping is a drill site. It provides a record of the present-day litho-stratigraphy of the sediment sitting on top of the submerged oceanic or continental crust.

The difference between backtracking and backstripping is whether recorded paleo-water depths are recorded in the drill site file. When there are no recorded paleo-water depths, *backtracking* uses a known model of tectonic subsidence (oceanic or continental) to determine the unknown paleo-water depths. Conversely, when there is a record of paleo-water depths, *backstripping* uses these known paleo-water depths to determine the unknown history of tectonic subsidence.

Backtracking versus backstripping sites

ODP drill site 699 is located on deep *ocean* crust and has no recorded paleo-water depths:

```
# SiteLongitude = -30.677
# SiteLatitude = -51.542
# SurfaceAge = 0

## bottom_age bottom_depth lithology
   18.7       85.7         Diatomite 0.7 Clay 0.3
   25.0       142.0        Coccolith_ooze 0.3 Diatomite 0.5 Mud 0.2
   31.3       233.6        Coccolith_ooze 0.3 Diatomite 0.7
   31.9       243.1        Sand 1
```

(continues on next page)

(continued from previous page)

```
   36.7        335.4        Coccolith_ooze 0.8 Diatomite 0.2
   40.8        382.6        Chalk 1
   54.5        496.6        Chalk 1
   55.3        516.3        Chalk 0.5 Clay 0.5
```

So it is suitable for *backtracking*, to find the unknown paleo-water depths.

In contrast, the sunrise drill site is located on shallower *continental* crust and has a record of paleo-water depths:

```
# SiteLatitude = -9.5901
# SiteLongitude = 128.1538
# SurfaceAge = 0.0000
#
## bottom_age bottom_depth min_water_depth max_water_depth lithology
   2.000       462.000      0.000           100.000         Shale            0.20       ␣
↪Limestone         0.75       Dolostone         0.05
   10.000      525.000      0.000           100.000         Shale            0.20       ␣
↪Limestone         0.75       Dolostone         0.05
   24.000      822.000      0.000           100.000         Shale            0.10       ␣
↪Limestone         0.80       Sand              0.10
   30.000      1062.000     0.000           100.000         Shale            0.30       ␣
↪Limestone         0.55       Dolostone         0.05       Sand            0.10
   34.000      1086.000     0.000           100.000         Shale            0.20       ␣
↪Limestone         0.10       Sand              0.70
   45.000      1366.000     0.000           100.000         Shale            0.10       ␣
↪Limestone         0.75       Dolostone         0.05       Sand            0.10
   58.000      1442.000     0.000           100.000         Shale            0.15       ␣
↪Limestone         0.15       Sand              0.70
   68.000      1494.000     50.000          200.000         Shale            0.45       ␣
↪Limestone         0.50       Sand              0.05
   83.000      1521.000     20.000          200.000         Shale            0.30       ␣
↪Limestone         0.65       Sand              0.05
   86.000      1545.000     20.000          200.000         Shale            0.55       ␣
↪Limestone         0.35       Sand              0.10
   88.000      1582.000     20.000          200.000         Shale            0.35       ␣
↪Limestone         0.65
   90.000      1620.000     20.000          200.000         Shale            0.70       ␣
↪Limestone         0.15       Sand              0.15
   95.000      1890.000     20.000          200.000         Shale            0.70       ␣
↪Limestone         0.15       Sand              0.15
   100.000     2036.000     20.000          200.000         Shale            0.70       ␣
↪Limestone         0.15       Sand              0.15
   107.000     2062.000     20.000          200.000         Shale            0.64       ␣
↪Limestone         0.18       Sand              0.18
   125.000     2066.000     0.000           100.000         Shale            0.40       ␣
↪Chalk             0.10       Sand              0.50
   160.000     2068.000     0.000           100.000         Shale            0.40       ␣
↪Limestone         0.30       Sand              0.30
   165.000     2130.000     0.000           100.000         Shale            0.40       ␣
↪Limestone         0.30       Sand              0.30
   170.000     2176.000     0.000           100.000         Shale            0.50       ␣
↪Sand              0.50
   177.000     2187.000     -10.000         25.000          Shale            0.30       ␣
```

(continues on next page)

(continued from previous page)

```
↪Sand          0.70
  180.000    2237.000     -10.000         25.000          Shale          0.30      ␣
↪Sand          0.70
  190.000    2311.000     -10.000         20.000          Shale          0.30      ␣
↪Sand          0.70
```

So it is suitable for *backstripping*, to find the unknown history of tectonic subsidence. Note that this site records the paleo-water depths as two extra columns, for the minimum and maximum water depths. Backstripping will then use these paleo-water depths, along with sediment decompaction, to reveal the complex tectonic subsidence of rift stretching at the site location.

Note: It is possible, although perhaps not desirable, to backtrack (instead of backstrip) the sunrise drill site to provide simulated paleo-water depths via a built-in model of continental rift stretching. This would involve ignoring the recorded paleo-water depth columns (using the `-c` option of *backtrack*) and supplying the start and end times of rifting (using the `-rs` and `-re` options of *backtrack*).

Drill site file format

As seen in the *Backtracking versus backstripping sites*, the file format of drill sites consist of two main sections. The top section specifies the *attributes* of the drill site, and the bottom section specifies the *stratigraphic layers*.

The attributes `SiteLongitude` and `SiteLatitude` specify the drill site location (in degrees).

Note: If `SiteLongitude` and `SiteLatitude` are not specified then they must be specified directly in the *backtrack* or *backstrip* module using the `-w` command-line option, or the *well_location* argument of the `pybacktrack.backtrack_and_write_well()` or `pybacktrack.backstrip_and_write_well()` function.

For each stratigraphic layer in the drill site there is a mixture of lithologies representing the stratigraphic composition of that layer. Each lithology (in a layer) is identified by a lithology name and the fraction it contributes to the layer (where all the fractions must add up to `1.0`). Each lithology name is used to look up a list of *lithology definitions* to obtain lithology density, surface porosity and porosity decay.

For each stratigraphic layer in the drill site there is also an age (Ma) and a depth (m) representing the bottom of that layer. The top age and depth of each layer is the bottom age and depth of the layer above. Since the surface (top) layer has no layer above it, the top age and depth of the surface layer are 0Ma and 0m respectively. However, if the `SurfaceAge` attribute is specified then it replaces the top age of the surface layer. A non-zero value of `SurfaceAge` implies that sediment deposition ended prior to present day. In other words, it represents the age of the total sediment surface.

Note: The `SurfaceAge` attribute is optional, and defaults to 0Ma if not specified.

Base sediment layer

It is also possible that the sediment thickness recorded at the drill site is less than the total sediment thickness. This happens when the drill site does not penetrate all the way to the basement depth of oceanic or continental crust. In this situation a base stratigraphic layer is automatically added during backtracking and backstripping to represent sediment from the bottom of the drill site down to the basement depth of oceanic or continental crust.

For backtracking, the bottom age of this new base layer is the age of oceanic crust if the drill site is on ocean crust, or the age that rifting starts if the drill site is on continental crust (since it is assumed that deposition began when continental stretching started) - see *backtrack* for more details.

For backstripping, the bottom age of this new base layer is simply duplicated from the age at the bottom of the drill site (ie, bottom age of deepest stratigraphic layer). This is because, unlike backtracking, we don't know the age of the crust. But this is fine since the decompacted output only uses the top age of each layer. And the decompacted sediment thickness/density (and hence the tectonic subsidence) still takes into account the base sediment layer and hence the total sediment thickness. Also since backstripping requires min/max recorded paleo-water depths for each layer, these are simply duplicated from the bottom layer of the drill site to the new base layer.

By default the lithology of the base layer is `Shale`, but can be changed using the `-b` command-line option in the *backtrack* and *backstrip* modules. To determine the total sediment thickness, a grid is sampled at the drill site location. The default grid is *bundled* inside `pybacktrack`. However, you can override this with your own grid by using the `-s` command-line option in the *backtrack* and *backstrip* modules.

The default total sediment thickness grid is:

- Straume, E.O., Gaina, C., Medvedev, S., Hochmuth, K., Gohl, K., Whittaker, J. M., et al. (2019). GlobSed: Updated total sediment thickness in the world's oceans. Geochemistry, Geophysics, Geosystems, 20. DOI: 10.1029/2018GC008115

Note: The default total sediment thickness grid was updated in pyBacktrack version 1.4.

Warning: If the drill site thickness happens to exceed the total sediment thickness then no base layer is added, and a warning is emitted to `standard error` on the console. This can happen as a result of uncertainties in the sediment thickness grid.

You can optionally write out an amended drill site file that adds this base sediment layer. This is useful when you want to know the basement depth at the drill site location.

For example, backtracking the ODP drill site 699 (located on *ocean* crust):

```
# SiteLongitude = -30.677
# SiteLatitude = -51.542
# SurfaceAge = 0

## bottom_age bottom_depth lithology
   18.7       85.7         Diatomite 0.7 Clay 0.3
   25.0       142.0        Coccolith_ooze 0.3 Diatomite 0.5 Mud 0.2
   31.3       233.6        Coccolith_ooze 0.3 Diatomite 0.7
   31.9       243.1        Sand 1
   36.7       335.4        Coccolith_ooze 0.8 Diatomite 0.2
   40.8       382.6        Chalk 1
   54.5       496.6        Chalk 1
   55.3       516.3        Chalk 0.5 Clay 0.5
```

…generates the following amended drill site file:

```
# SiteLatitude = -51.5420
# SiteLongitude = -30.6770
# SurfaceAge = 0.0000
#
## bottom_age bottom_depth lithology
   18.700     85.700       Diatomite       0.70       Clay            0.30
   25.000     142.000      Coccolith_ooze  0.30       Diatomite       0.50       Mud    ␣
↪        0.20
   31.300     233.600      Coccolith_ooze  0.30       Diatomite       0.70
   31.900     243.100      Sand            1.00
   36.700     335.400      Coccolith_ooze  0.80       Diatomite       0.20
   40.800     382.600      Chalk           1.00
   54.500     496.600      Chalk           1.00
   55.300     516.300      Chalk           0.50       Clay            0.50
   79.133     601.000      Shale           1.00
```

…containing the extra base shale layer with a bottom age equal to the age grid sampled at the drill site and a bottom depth equal to the total sediment thickness.

Note: To output an amended drill site file, specify the amended output filename using the `-o` command-line option in the *backtrack* or *backstrip* module.

Geohistory analysis

The Decompacting Stratigraphic Layers notebook shows how to visualize the decompaction of stratigraphic layers at a drill site.

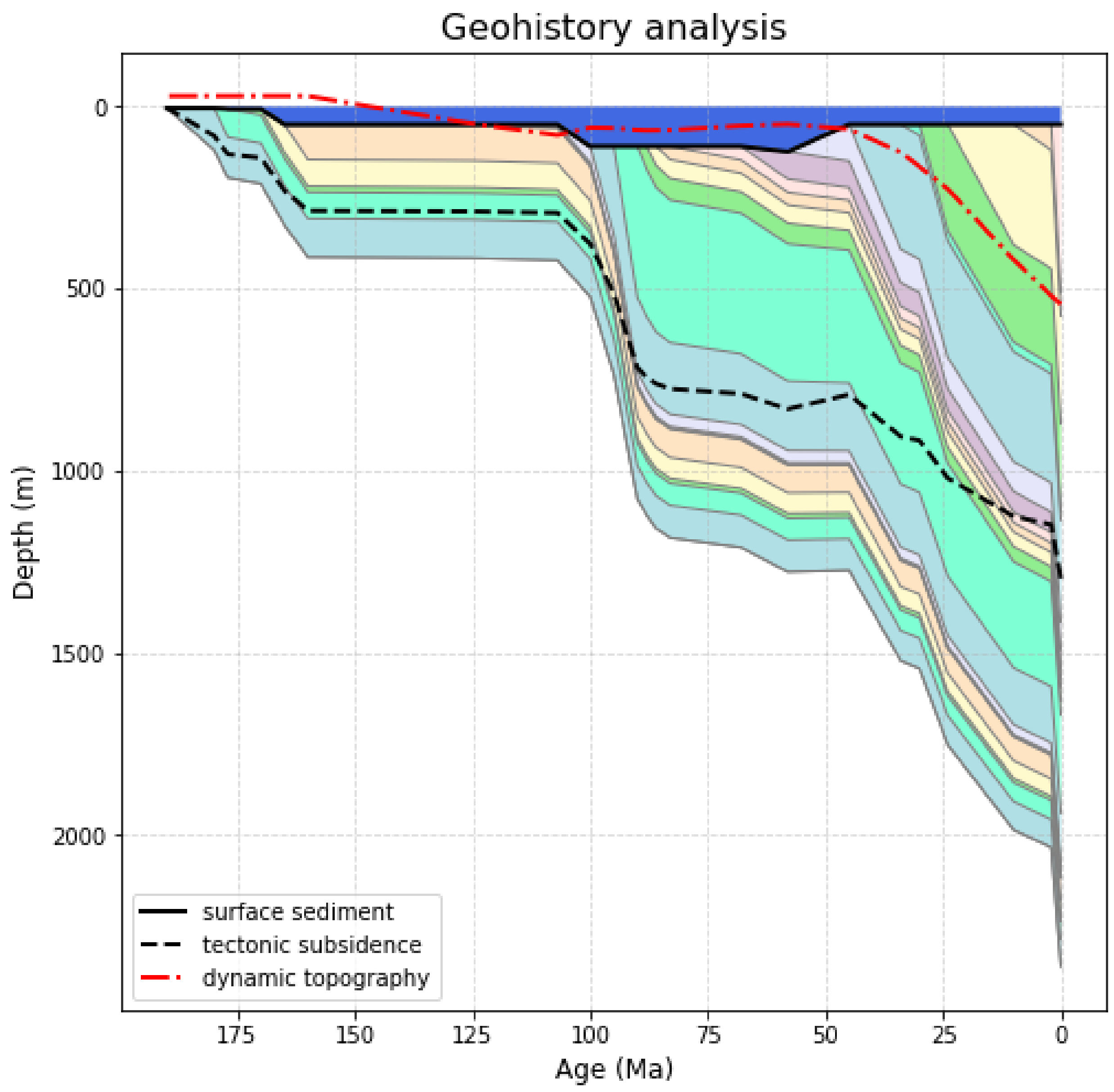
Note: The example notebooks are installed as part of the example data which can be installed by following *these instructions*.

That notebook decompacts drill sites in the context of *backtracking* and *backstripping* (covered in later sections), but regardless of whether we're backstripping or backtracking we are still decompacting the sediment layers in the same way. The following image (from that notebook) shows the decompaction of a shallow continental drill site over time.

2.3.2 Lithology Definitions

The stratigraphy layers in a *drill site* contain lithology *names* that reference lithology *definitions*. Each lithology definition contains a density, a surface porosity and a porosity decay.

These definitions are stored in lithology files.


Geohistory analysis
Depth (m)
0
500
1000
1500
2000
Age (Ma)
175
150
125
100
75
50
25
0
surface sediment
tectonic subsidence
dynamic topography

Bundled lithology definitions

There are two lithology files currently *bundled* inside `pybacktrack`, one containing *primary* lithologies and the other *extended* lithologies.

The *primary* lithologies (inside `pybacktrack`) contains the deep-sea lithologies listed in Table 1 in the pyBacktrack paper:

- Muller, R. D., Cannon, J., Williams, S. and Dutkiewicz, A., 2018, PyBacktrack 1.0: A Tool for Reconstructing Paleobathymetry on Oceanic and Continental Crust, **Geochemistry, Geophysics, Geosystems,** 19, 1898-1909, doi: 10.1029/2017GC007313.

```
# name                            density   porosity     decay
#                                   kg/m3    (%/100)         m
#
Average_ocean_floor_sediment         2647       0.66      1333    # Kominz et al. [2011]
Basalt                               2700        0.2      5000    # Turer and Maynard␣
↪[2003]
Biogenic_sand                        2710       0.89      1338    # Kominz et al. [2011]
Carbonate_sand                       2710       0.48      3990    # Goldhammer [1997]
Chalk                                2710        0.7      1408    # Sclater and␣
↪Christie [1980]
Clay                                 2735       0.76      1252    # Kominz et al. [2011]
Coccolith_ooze                       2710       0.59      1660    # Kominz et al. [2011]
Diatomite                            2457       0.84      436     # Kominz et al. [2011]
Dolomite                             2870       0.38      1986    # Schmoker and Halley␣
↪[1982]
Limestone                            2850       0.51      4545    # Turer and Maynard␣
↪[2003]
Micrite                              2710       0.69      1135    # Kominz et al. [2011]
Mud                                  2438       0.36      2015    # Van Sickel et al.␣
↪[2004]
Sand                                 2650       0.49      3704    # Sclater and␣
↪Christie [1980]
Shale                                2700       0.63      1960    # Sclater and␣
↪Christie [1980]
Shaley_sand                          2680       0.56      2564    # Sclater and␣
↪Christie [1980]
Silt                                 2661       0.76      1091    # Kominz et al. [2011]
```

And the *extended* lithologies (inside `pybacktrack`) mostly contain shallow-water lithologies:

- Baldwin, S., 1999, Quantifying the development of a deep sedimentary basin: the Bonaparte Basin, NW Australia, PhD Thesis, Univ. of Cambridge.

```
# name                            density   porosity     decay
#                                   kg/m3    (%/100)         m
#
Anhydrite                            2960       0.40       500
Chert                                1929       0.65      2850
Conglomerate                         3500       0.48      2700
Dolostone                            2700       0.48      3500
Grainstone                           2700       0.48      3500
Reef                                 2700       0.10      3500
Rhyolite                             2820       0.20      2700
```

(continues on next page)

(continued from previous page)

```
Salt                                    2160       0.20        750
```

Lithology file format

As seen in the *bundled lithology definitions*, the first column is the lithology name. The second column is the lithology's sediment density (kg/m3). The third column is the surface porosity as a fraction, and fourth column is porosity decay (m).

Note: You can also use your own lithology files provided they use this format.

Porosity is the contribution of water to the sediment volume and decays exponentially with depth according to the decay constant (since sediment compaction increases with depth and squeezes out more water from between the sediment grains).

Specifying lithology definitions

Any number of lithology files can be specified. In the *backtrack* and *backstrip* modules these are specified using the `-l` command-line option. With this option you can specify one or more lithologies files including the *bundled* lithologies. To specify the bundled *primary* and *extended* lithologies you specify `primary` and `extended`. And to specify your own lithology files you provide the entire filename as usual. If you don't specify the `-l` option then it defaults to using only the *primary* lithologies (*extended* lithologies are not included by default).

Note:

If you don't use the `-l` option then *only* the `primary` lithologies will be included (they are the default).

However if you use the `-l` option but do not specify `primary` then the primary lithologies will **not** be included.

Conflicting lithology definitions

When specifying more than one lithology file it is possible to have conflicting definitions. This occurs when two or more lithology files contain the same lithology *name* but have different values for its density, surface porosity or porosity decay. When there is a conflict, the lithology *definition* is taken from the last conflicting lithology file specified. For example, if you specify `-l primary my_conflicting_lithologies.txt` then conflicting lithologies in `my_conflicting_lithologies.txt` override those in `primary`. However, specifying the reverse order with `-l my_conflicting_lithologies.txt primary` will result in `primary` overriding those in `my_conflicting_lithologies.txt`.

2.4 Backtrack

2.4.1 Overview

The `backtrack` module is used to find paleo water depths from a tectonic subsidence model, and sediment decompaction over time. The tectonic subsidence model is either an age-to-depth curve (in ocean basins) or rifting (near continental passive margins).

2.4.2 Running backtrack

You can either run `backtrack` as a built-in script, specifying parameters as command-line options (`...`):

```
python -m pybacktrack.backtrack_cli ...
```

…or `import pybacktrack` into your own script, calling its functions and specifying parameters as function arguments (`...`):

```
import pybacktrack

pybacktrack.backtrack_and_write_well(...)
```

Note: You can run `python -m pybacktrack.backtrack_cli --help` to see a description of all command-line options available, or see the *backtracking reference section* for documentation on the function parameters.

Example

For example, revisiting our *backtracking example*, we can run it from the command-line as:

```
python -m pybacktrack.backtrack_cli \
    -w pybacktrack_examples/example_data/ODP-114-699-Lithology.txt \
    -d age compacted_depth compacted_thickness decompacted_thickness decompacted_density␣
↪decompacted_sediment_rate decompacted_depth dynamic_topography water_depth tectonic_
↪subsidence lithology \
    -ym M2 \
    -slm Haq87_SealevelCurve_Longterm \
    -o ODP-114-699_backtrack_amended.txt \
    -- \
    ODP-114-699_backtrack_decompacted.txt
```

. . . or write some Python code to do the same thing:

```
import pybacktrack

# Input and output filenames.
input_well_filename = 'pybacktrack_examples/example_data/ODP-114-699-Lithology.txt'
amended_well_output_filename = 'ODP-114-699_backtrack_amended.txt'
decompacted_output_filename = 'ODP-114-699_backtrack_decompacted.txt'

# Read input well file, and write amended well and decompacted results to output files.
pybacktrack.backtrack_and_write_well(
    decompacted_output_filename,
    input_well_filename,
    dynamic_topography_model='M2',
    sea_level_model='Haq87_SealevelCurve_Longterm',
    # The columns in decompacted output file...
    decompacted_columns=[pybacktrack.BACKTRACK_COLUMN_AGE,
                         pybacktrack.BACKTRACK_COLUMN_COMPACTED_DEPTH,
                         pybacktrack.BACKTRACK_COLUMN_COMPACTED_THICKNESS,
                         pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_THICKNESS,
                         pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DENSITY,
                         pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_SEDIMENT_RATE,
                         pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DEPTH,
                         pybacktrack.BACKTRACK_COLUMN_DYNAMIC_TOPOGRAPHY,
                         pybacktrack.BACKTRACK_COLUMN_WATER_DEPTH,
                         pybacktrack.BACKTRACK_COLUMN_TECTONIC_SUBSIDENCE,
                         pybacktrack.BACKTRACK_COLUMN_LITHOLOGY],
    # Might be an extra stratigraphic well layer added from well bottom to ocean␣
↪basement...
    ammended_well_output_filename=amended_well_output_filename)
```

Note: The drill site file `pybacktrack_examples/example_data/ODP-114-699-Lithology.txt` is part of the *example data*.

2.4.3 Backtrack output

For each stratigraphic layer in the input drill site file, `backtrack` can write one or more parameters to an output file.

Running the *above example* on ODP drill site 699:

```
# SiteLongitude = -30.677
# SiteLatitude = -51.542
# SurfaceAge = 0

## bottom_age bottom_depth lithology
   18.7       85.7         Diatomite 0.7 Clay 0.3
   25.0       142.0        Coccolith_ooze 0.3 Diatomite 0.5 Mud 0.2
   31.3       233.6        Coccolith_ooze 0.3 Diatomite 0.7
   31.9       243.1        Sand 1
   36.7       335.4        Coccolith_ooze 0.8 Diatomite 0.2
   40.8       382.6        Chalk 1
   54.5       496.6        Chalk 1
   55.3       516.3        Chalk 0.5 Clay 0.5
```

…produces an *amended drill site output file* containing an extra base sediment layer, and a *decompacted output file* containing the decompacted output parameters like sediment thickness and water depth.

Amended drill site output

The amended drill site output file:

```
# SiteLatitude = -51.5420
# SiteLongitude = -30.6770
# SurfaceAge = 0.0000
#
## bottom_age bottom_depth lithology
   18.700     85.700       Diatomite      0.70       Clay           0.30
   25.000     142.000      Coccolith_ooze 0.30       Diatomite      0.50       Mud     ␣
↪          0.20
   31.300     233.600      Coccolith_ooze 0.30       Diatomite      0.70
   31.900     243.100      Sand           1.00
   36.700     335.400      Coccolith_ooze 0.80       Diatomite      0.20
   40.800     382.600      Chalk          1.00
   54.500     496.600      Chalk          1.00
   55.300     516.300      Chalk          0.50       Clay           0.50
   79.133     601.000      Shale          1.00
```

There is an extra *base sediment layer* that extends from the bottom of the drill site (516.3 metres) to the total sediment thickness (601 metres). The bottom age of this new base layer (86.79 Ma) is the age of oceanic crust that ODP drill site 699 is on. If it had been on continental crust (near a passive margin such as DSDP drill site 327) then the bottom age of this new base layer would have been when rifting started (since we would have assumed deposition began when continental stretching began).

See also:

Base sediment layer and *Oceanic versus continental drill sites*

Decompacted output

The decompacted output file:

```
# SiteLatitude = -51.5420
# SiteLongitude = -30.6770
# SurfaceAge = 0.0000
#
# age       compacted_depth compacted_thickness decompacted_thickness decompacted_
↪density decompacted_sediment_rate decompacted_depth dynamic_topography water_depth␣
↪tectonic_subsidence lithology
 0.000     0.000           601.000             601.000               1726.994        ␣
↪ 5.810                     0.000           0.000               3798.317    4134.284  ␣
↪          Diatomite       0.70       Clay             0.30
 18.700    85.700          515.300             552.679               1733.298        ␣
↪ 10.682                    108.648         75.174              3604.761    3872.267  ␣
↪          Coccolith_ooze  0.30       Diatomite        0.50       Mud             0.20
 25.000    142.000         459.000             518.231               1715.612        ␣
↪ 24.388                    175.945         88.541              3543.441    3770.680  ␣
↪          Coccolith_ooze  0.30       Diatomite        0.70
 31.300    233.600         367.400             431.443               1727.755        ␣
↪ 16.781                    329.587         102.254             3536.973    3651.861  ␣
↪          Sand            1.00
 31.900    243.100         357.900             424.770               1719.132        ␣
↪ 25.543                    339.656         104.005             3544.482    3639.106  ␣
↪          Coccolith_ooze  0.80       Diatomite        0.20
 36.700    335.400         265.600             342.298               1675.058        ␣
↪ 17.557                    462.265         128.465             3467.220    3519.684  ␣
↪          Chalk           1.00
 40.800    382.600         218.400             291.817               1662.325        ␣
↪ 13.524                    534.247         133.268             3439.494    3423.104  ␣
↪          Chalk           1.00
 54.500    496.600         104.400             147.135               1649.440        ␣
↪ 45.709                    719.529         161.651             3103.703    2999.804  ␣
↪          Chalk           0.50       Clay             0.50
 55.300    516.300         84.700              114.840               1678.129        ␣
↪ 5.054                     756.096         162.953             3116.404    2970.460  ␣
↪          Shale           1.00
```

The *age*, *compacted_depth* and *lithology* columns are the same as the *bottom_age*, *bottom_depth* and *lithology* columns in the input drill site (except there is also a row associated with the surface age).

The *compacted_thickness* column is the total sediment thickness (601 metres - see base sediment layer of *amended drill site* above) minus *compacted_depth*. The *decompacted_thickness* column is the thickness of all sediment at the associated age. In other words, at each consecutive age another stratigraphic layer is essentially removed, allowing the underlying layers to expand (due to their porosity). At present day (or the surface age) the decompacted thickness is just the compacted thickness. The *decompacted_density* column is the average density integrated over the decompacted thickness of the drill site (each stratigraphic layer contains a mixture of water and sediment according to its porosity at the decompacted depth of the layer). The *decompacted_sediment_rate* column is the rate of sediment deposition in units of metres/Ma. At each time it is calculated as the fully decompacted thickness (ie, using surface porosity only) of the surface stratigraphic layer (whose deposition ends at the specified time) divided by the layer's deposition time interval. The *decompacted_depth* column is similar to *decompacted_sediment_rate* in that the stratigraphic layers are fully decompacted (using surface porosity only) as if no portion of any layer had ever been buried. It is also similar to *compacted_depth* except all effects of compaction have been removed. The *dynamic_topography* column is the dynamic topography elevation relative to present day (or zero if no dynamic topography model was specified).

Finally, *tectonic_subsidence* is the output of the underlying *tectonic subsidence model*, and *water_depth* is obtained from tectonic subsidence by subtracting an isostatic correction of the decompacted sediment thickness.

Note: The output columns are specified using the `-d` command-line option (run `python -m pybacktrack.backtrack_cli --help` to see all options), or using the *decompacted_columns* argument of the `pybacktrack.backtrack_and_write_well()` function. By default, only *age* and *decompacted_thickness* are output.

2.4.4 Sea level variation

A model of the variation of sea level relative to present day can optionally be used when backtracking. This adjusts the isostatic correction of the decompacted sediment thickness to take into account sea-level variations.

There are two built-in sea level models *bundled* inside `backtrack`:

- `Haq87_SealevelCurve` - The Phanerozoic Record of Global Sea-Level Change
- `Haq87_SealevelCurve_Longterm` - Normalised to start at zero at present-day.

A sea-level model is optional. If one is not specified then sea-level variation is assumed to be zero.

Note: A built-in sea-level model can be specified using the `-slm` command-line option (run `python -m pybacktrack.backtrack_cli --help` to see all options), or using the *sea_level_model* argument of the `pybacktrack.backtrack_and_write_well()` function.

Note: It is also possible to specify your own sea-level model. This can be done by providing your own text file containing a column of ages (Ma) and a corresponding column of sea levels (m), and specifying the name of this file to the `-sl` command-line option or to the *sea_level_model* argument of the `pybacktrack.backtrack_and_write_well()` function.

2.4.5 Oceanic and continental tectonic subsidence

Tectonic subsidence is modelled separately for ocean basins and continental passive margins. The subsidence model chosen by the `backtrack` module depends on whether the drill site is on oceanic or continental crust. This is determined by an oceanic age grid. Since the age grid captures only oceanic crust, a drill site inside this region will automatically use the oceanic subsidence model whereas a drill site outside this region uses the continental subsidence model.

The default present-day age grid *bundled* inside `backtrack` is a 6-minute resolution grid of the age of the world's ocean crust that uses the timescale of Gee and Kent (2007):

- Seton, M., Müller, R. D., Zahirovic, S., Williams, S., Wright, N., Cannon, J., Whittaker, J., Matthews, K., McGirr, R., (2020), A global dataset of present-day oceanic crustal age and seafloor spreading parameters, Geochemistry, Geophysics, Geosystems, doi: 10.1029/2020GC009214

Note: The default present-day age grid was updated in pyBacktrack version 1.4.

Note: You can optionally specify your own age grid using the `-a` command-line option (run `python -m pybacktrack.backtrack_cli --help` to see all options), or using the *age_grid_filename* argument of the `pybacktrack.backtrack_and_write_well()` function.

Oceanic versus continental drill sites

ODP drill site 699 is located on deeper *ocean* crust (as opposed to shallower continental crust):

```
# SiteLongitude = -30.677
# SiteLatitude = -51.542
# SurfaceAge = 0

## bottom_age bottom_depth lithology
   18.7       85.7         Diatomite 0.7 Clay 0.3
   25.0       142.0        Coccolith_ooze 0.3 Diatomite 0.5 Mud 0.2
   31.3       233.6        Coccolith_ooze 0.3 Diatomite 0.7
   31.9       243.1        Sand 1
   36.7       335.4        Coccolith_ooze 0.8 Diatomite 0.2
   40.8       382.6        Chalk 1
   54.5       496.6        Chalk 1
   55.3       516.3        Chalk 0.5 Clay 0.5
```

So it will use the *oceanic* subsidence model.

See also:

Oceanic subsidence

In contrast, DSDP drill site 327 is located on shallower *continental* crust (as opposed to deeper ocean crust):

```
# SiteLongitude = -46.7837
# SiteLatitude = -50.8713
# RiftStartAge = 160
# RiftEndAge = 120
# SurfaceAge = 0

## bottom_age bottom_depth lithology
   1.5        10.0         Shaley_sand 1
   55.8       30.0         Clay 1
   59.9       68.0         Diatomite 0.7 Clay 0.3
   62.2       90.0         Clay 1
   77.4       142.0        Coccolith_ooze 0.7 Biogenic_sand 0.3
   86.4       154.0        Clay 1
   113.1      324.0        Coccolith_ooze 0.3 Clay 0.7
   122.3      469.5        Clay 1
```

So it will use the *continental* subsidence model. Since continental subsidence involves rifting, it requires a rift start and end time. These extra rift parameters can be specified at the top of the drill site file as `RiftStartAge` and `RiftEndAge` attributes (see *Continental subsidence*).

See also:

Continental subsidence

If you are not sure whether your drill site lies on oceanic or continental crust then first prepare your drill site assuming it's on oceanic crust (since this does not need rift start and end ages). If an error message is generated when *running backtrack* then you'll need to determine the rift start and end age, then add these to your drill site file as `RiftStartAge` and `RiftEndAge` attributes, and then run backtrack again.

Note: In pyBacktrack version 1.4 if the `RiftStartAge` and `RiftEndAge` attributes are not specified in your drill site file then they are obtained implicitly from the builtin rift start/end time grids (see *Continental subsidence*), so an error message is unlikely to be generated when your drill site file is on *continental* crust.

Present-day tectonic subsidence

The tectonic subsidence at present day is used in both the oceanic and continental subsidence models. Tectonic subsidence is unloaded water depth, that is with sediment removed. So to obtain an accurate value, `backtrack` starts with a bathymetry grid to obtain the present-day water depth (the depth of the sediment surface). Then an isostatic correction of the present-day sediment thickness (at the drill site) takes into account the removal of sediment to reveal the present-day tectonic subsidence. The isostatic correction uses the average sediment density of the drill site stratigraphy.

The default present-day bathymetry grid *bundled* inside `backtrack` is a 6-minute resolution global grid of the land topography and ocean bathymetry (although only the ocean bathymetry is actually needed):

- Amante, C. and B. W. Eakins, ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. NOAA Technical Memorandum NESDIS NGDC-24, 19 pp, March 2009

Note: You can optionally specify your own bathymetry grid using the `-t` command-line option (run `python -m pybacktrack.backtrack_cli --help` to see all options), or using the *topography_filename* argument of the `pybacktrack.backtrack_and_write_well()` function.

Note: If you specify your own bathymetry grid, ensure that its ocean water depths are negative. It is assumed that elevations in the grid above/below sea level are positive/negative.

2.4.6 Oceanic subsidence

Oceanic subsidence is somewhat simpler and more accurately modelled than continental subsidence (due to *no* lithospheric stretching).

The age of oceanic crust at the drill site (sampled from the oceanic age grid) can be converted to tectonic subsidence (depth with sediment removed) by using an age-to-depth model. There are three models built into `backtrack`:

- `RHCW18` - Richards et al. (2020) Structure and dynamics of the oceanic lithosphere-asthenosphere system
- `CROSBY_2007` - Crosby, A.G., (2007). *Aspects of the relationship between topography and gravity on the Earth and Moon, PhD thesis, University of Cambridge*

  The Python source code that implements this age-depth relationship can be found here. And note that additional background information on this model can be found in: Crosby, A.G. and McKenzie, D., 2009. An analysis of young ocean depth, gravity and global residual topography.
- `GDH1` - Stein and Stein (1992) Model for the global variation in oceanic depth and heat flow with lithospheric age

The default model is `RHCW18`.

Note: The default age-to-depth model was updated in pyBacktrack version 1.4. It is now `RHCW18`. Previously it was `GDH1`.

Note: These oceanic subsidence models can be specified using the `-m` command-line option (run `python -m pybacktrack.backtrack_cli --help` to see all options), or using the *ocean_age_to_depth_model* argument of the `pybacktrack.backtrack_and_write_well()` function.

Note: It is also possible to specify your own age-to-depth model. This can be done by providing your own text file containing a column of ages and a corresponding column of depths, and specifying the name of this file along with two integers representing the age and depth column indices to the `-m` command-line option. Or you can pass your own function as the *ocean_age_to_depth_model* argument of the `pybacktrack.backtrack_and_write_well()` function, where your function should accept a single age (Ma) argument and return the corresponding depth (m).

Since the drill site might be located on anomalously thick or thin ocean crust, a constant offset is added to the age-to-depth model to ensure the model subsidence matches the *actual subsidence* at present day.

2.4.7 Continental subsidence

Continental subsidence is somewhat more complex and less accurately modelled than oceanic subsidence (due to lithospheric stretching).

The continental subsidence model has two components of rifting as described in PyBacktrack 1.0: A Tool for Reconstructing Paleobathymetry on Oceanic and Continental Crust. The first contribution is *initial* subsidence due to lithospheric thinning where low-density crust is thinned and hot asthenosphere rises underneath. In our model the crust and lithospheric mantle are identically stretched (uniform extension). The second contribution is thermal subsidence where the lithosphere thickens as it cools due to conductive heat loss. In our model thermal subsidence only takes place once the stretching stage has ended. In this way, there is instantaneous stretching from a thermal perspective (in the sense that, although stretching happens over a finite period of time, the model assumes no cooling during the stretching stage).

Note: The tectonic subsidence at the start of rifting is zero. This is because it is assumed that rifting begins at sea level, and begins with a sediment thickness of zero (since sediments are yet to be deposited on newly forming ocean crust).

For drill sites on continental crust, the rift *end* time must be provided. However the rift *start* time is optional. If it is not specified then it is assumed to be equal to the rift *end* time. In other words, lithospheric stretching is assumed to happen immediately at the rift *end* time (as opposed to happening over a period of time). This is fine for stratigraphic layers deposited after rifting has ended, since the subsidence will be the same regardless of whether a rift *start* time was specified or not.

Note: The rift start and end times can be specified in the drill site file using the `RiftStartAge` and `RiftEndAge` attributes. Or they can be specified directly on the `backtrack` command-line using the `-rs` and `-re` options respectively (run `python -m pybacktrack.backtrack_cli --help` to see all options). Or using the *rifting_period* argument of the `pybacktrack.backtrack_and_write_well()` function.

Note: If the rift end time (and optional start time) is not explicitly specified in the drill site file or explicitly on the `backtrack` command-line (or explicitly via the `pybacktrack.backtrack_and_write_well()` function) then both the rift start and end times are obtained implicitly from the builtin rift start/end time grids. If the well location is outside valid regions of the rift start/end time grids then an error is generated and you must then explicitly provide the rift end time (and optionally the rift start time). However currently the rift grids cover all submerged continental crust (ie, where the total sediment thickness grid contains valid values but the age grid does not) and not just the areas that are rifting - see *rift gridding* - so an error is unlikely to be generated.

If a rift *start* time is specified, then the stretching factor varies exponentially between the rift *start* and *end* times (assuming a constant strain rate). The stretching factor at the rift *start* time is `1.0` (since the lithosphere has not yet stretched). The stretching factor at the rift *end* time is estimated such that our model produces a subsidence matching the *actual subsidence* at present day, while also thinning the crust to match the actual crustal thickness at present day.

Note: The crustal thickness at the end of rifting and at present day are assumed to be the same.

Warning: If the estimated rift stretching factor (at the rift *end* time) results in a tectonic subsidence inaccuracy (at present day) of more than 100 metres, then a warning is emitted to `standard error` on the console. This can happen if the actual present-day subsidence is quite deep and the stretching factor required to achieve this subsidence would be unrealistically large and result in a pre-rift crustal thickness (equal to the stretching factor multiplied by the actual present-day crustal thickness) that exceeds typical lithospheric thicknesses (125km). In this case the stretching factor is clamped to avoid this but, as a result, the modeled subsidence is not as deep as the actual subsidence.

The default present-day crustal thickness grid *bundled* inside `backtrack` is a 1-degree resolution grid of the thickness of the crustal part of the lithosphere:

- Laske, G., Masters., G., Ma, Z. and Pasyanos, M., Update on CRUST1.0 - A 1-degree Global Model of Earth's Crust, Geophys. Res. Abstracts, 15, Abstract EGU2013-2658, 2013

Note: You can optionally specify your own crustal thickness grid using the `-k` command-line option (run `python -m pybacktrack.backtrack_cli --help` to see all options), or using the *crustal_thickness_filename* argument of the `pybacktrack.backtrack_and_write_well()` function.

2.4.8 Dynamic topography

The effects of dynamic topography can be included in the models of tectonic subsidence (both oceanic and continental).

A dynamic topography model is optional. If one is not specified then dynamic topography is assumed to be zero.

All dynamic topography models consist of a sequence of time-dependent global grids (where each grid is associated with a past geological time). The grids are in the *mantle* reference frame (instead of the *plate* reference frame) and hence the drill site location must be reconstructed (back in time) before sampling these grids. To enable this, a dynamic topography model also includes an associated static-polygons file to assign a reconstruction plate ID to the drill site, and associated rotation file(s) to reconstruct the drill site location.

Note: The dynamic topography grids are interpolated at times not coinciding with the grid times. The method of interpolation changed in pyBacktrack version 1.4 (as described in the notes of `pybacktrack.DynamicTopography.sample()`) - however this change has no effect *at* the grid times (only between grid times).

Warning: If the drill site is reconstructed to a time that is older than supported by the dynamic topography model then the oldest dynamic topography grid is used. Also note that the drill site can be reconstructed to a time that is older than the age of the crust it is located on if the bottom age in the drill site file is older than the basement age.

Dynamic topography is included in the oceanic subsidence model by adjusting the subsidence to account for the change in dynamic topography at the drill site since present day.

See also:

Oceanic subsidence

Dynamic topography is included in the continental subsidence model by first removing the effects of dynamic topography (between the start of rifting and present day) prior to estimating the rift stretching factor. This is because estimation of the stretching factor only considers subsidence due to lithospheric thinning (stretching) and subsequent thickening (thermal cooling). Once the optimal stretching factor has been estimated, the continental subsidence is adjusted to account for the change in dynamic topography since the start of rifting.

See also:

Continental subsidence

These are the built-in dynamic topography models *bundled* inside `backtrack`:

- *Young et al., 2022* - Long-term Phanerozoic sea level change from solid Earth processes
  - gld428
- *Braz et al., 2021* - Modelling the role of dynamic topography and eustasy in the evolution of the Great Artesian Basin
  - D10_gmcm9
- *Cao et al., 2019* - The interplay of dynamic topography and eustasy on continental flooding in the late Paleozoic
  - AY18
  - KM16
- *Müller et al., 2017* - Dynamic topography of passive continental margins and their hinterlands since the Cretaceous
  - M1
  - M2
  - M3
  - M4
  - M5
  - M6
  - M7
- *Rubey et al., 2017* - Global patterns of Earth's dynamic topography since the Jurassic
  - terra
- *Müller et al., 2008* - Long-term sea-level fluctuations driven by ocean basin dynamics
  - ngrand
  - s20rts
  - smean

Note: The above model links reference dynamic topography models that can be visualized in the GPlates Web Portal.

The `M1` model is a combined forward/reverse geodynamic model, while models `M2-M7` are forward models. Models `ngrand`, `s20rts` and `smean` are backward-advection models. The backward-advection models are generally good for the recent geological past (up to last 70 million years). While the `M1-M7` models are most useful when it is necessary to look at times older than 70 Ma because their oceanic paleo-depths lack the regional detail at more recent times that the backward-advection models capture (because of their assimilation of seismic tomography). `M1` also assimilates seismic tomography but suffers from other shortcomings.

Note: A built-in dynamic topography model can be specified using the `-ym` command-line option (run `python -m pybacktrack.backtrack_cli --help` to see all options), or using the *dynamic_topography_model* argument of the `pybacktrack.backtrack_and_write_well()` function.

Note: It is also possible to specify your own dynamic topography model. This can be done by providing your own grid list text file with the first column containing a list of the dynamic topography grid filenames (where each filename should be relative to the directory on the list file) and the second column containing the associated grid times (in Ma). You'll also need the associated static-polygons file, and one or more associated rotation files. The grid list filename, static-polygons filename and one or more rotation filenames are then specified using the `-y` command-line option (run `python -m pybacktrack.backtrack_cli --help` to see all options), or to the *dynamic_topography_model* argument of the `pybacktrack.backtrack_and_write_well()` function.

2.4.9 Geohistory analysis

The Decompacting Stratigraphic Layers notebook shows how to visualize the decompaction of stratigraphic layers at a drill site.

Note: The example notebooks are installed as part of the example data which can be installed by following *these instructions*.

Continental subsidence

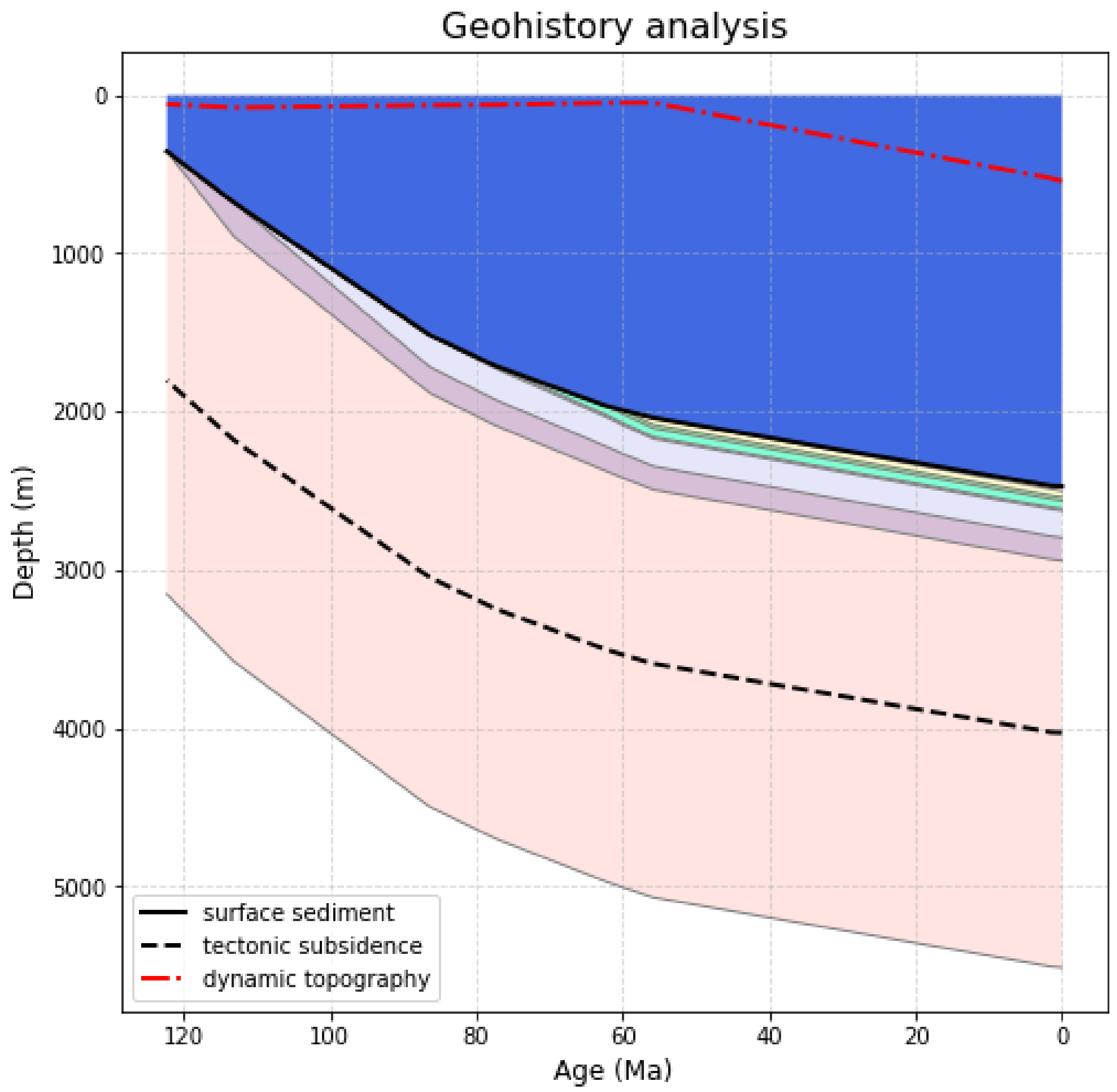
One of the examples in that notebook demonstrates decompaction of a shallow continental drill site using backtracking. The tectonic subsidence (black dashed line) is from our *model of continental subsidence* and the paleo water depths (blue fill) are backtracked using tectonic subsidence and sediment decompaction. Note that, unlike backstripping, *dynamic topography* *does* affect tectonic subsidence (because its effects are included in the model of tectonic subsidence).

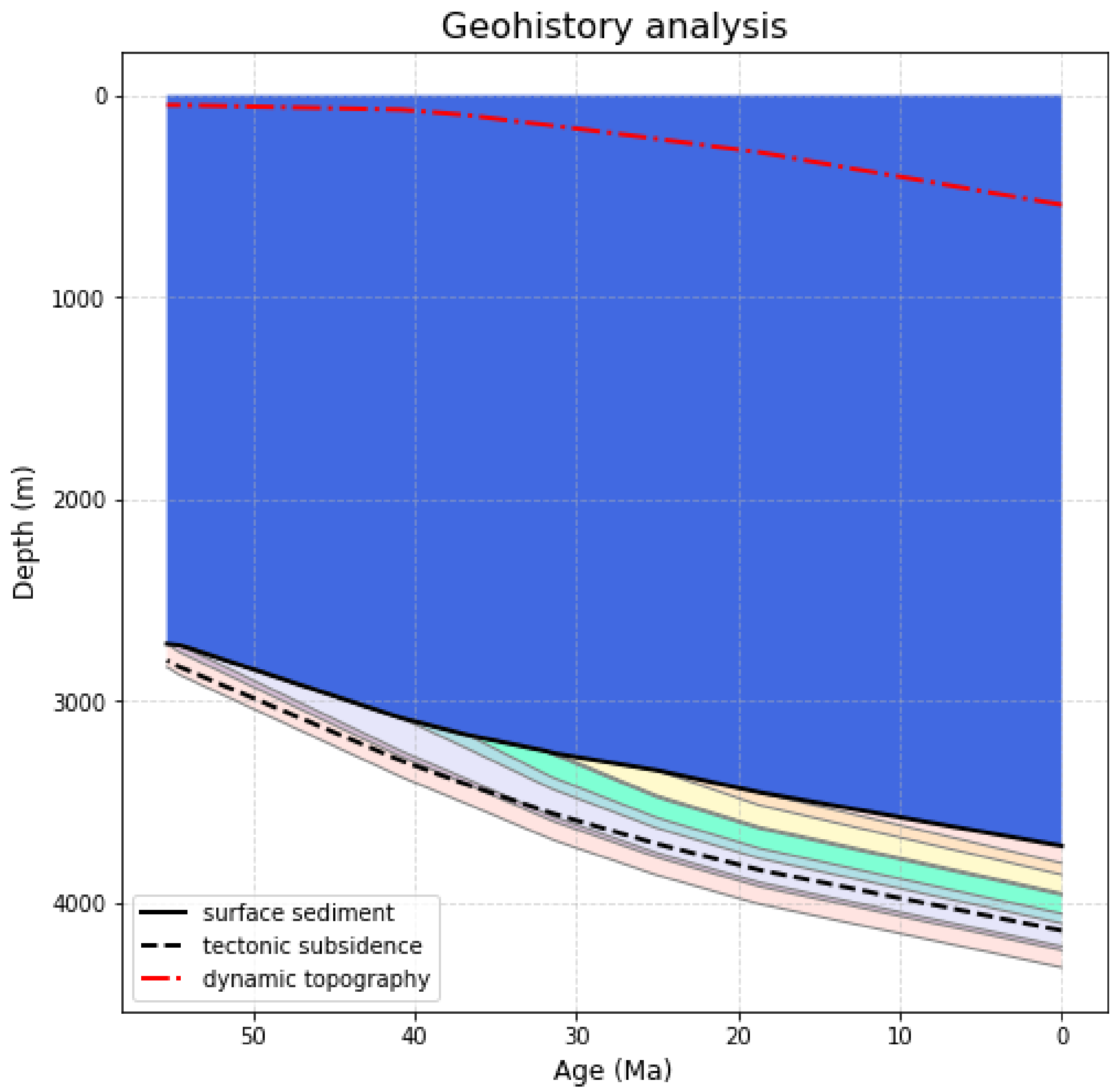
Note: There is a base sediment layer below the drill site (from the bottom of drill site to basement depth) since the drill site does not reach basement depth. And for this drill site the base sediment layer is quite thick because the default total sediment thickness grid is not as accurate near continental margins (compared to deeper ocean basins).

Oceanic subsidence

Another example in that notebook demonstrates decompaction of an oceanic drill site using backtracking. The tectonic subsidence (black dashed line) is from our *model of oceanic subsidence* and the paleo water depths (blue fill) are back-tracked using tectonic subsidence and sediment decompaction. Note that, unlike backstripping, *dynamic topography* *does* affect tectonic subsidence (because its effects are included in the model of tectonic subsidence).

Note: There is a base sediment layer below the drill site (from the bottom of drill site to basement depth) since the drill site does not reach basement depth.


Geohistory analysis
Depth (m)
0
1000
2000
3000
4000
5000
Age (Ma)
120
100
80
60
40
20
0
surface sediment
tectonic subsidence
dynamic topography


Geohistory analysis
Depth (m)
0
1000
2000
3000
4000
Age (Ma)
50
40
30
20
10
0
surface sediment
tectonic subsidence
dynamic topography

2.5 Backstrip

2.5.1 Overview

The `backstrip` module is used to find tectonic subsidence from paleo water depths, and sediment decompaction over time.

2.5.2 Running backstrip

You can either run `backstrip` as a built-in script, specifying parameters as command-line options (`...`):

```
python -m pybacktrack.backstrip_cli ...
```

…or `import pybacktrack` into your own script, calling its functions and specifying parameters as function arguments (`...`):

```
import pybacktrack

pybacktrack.backstrip_and_write_well(...)
```

Note: You can run `python -m pybacktrack.backstrip_cli --help` to see a description of all command-line options available, or see the *backstripping reference section* for documentation on the function parameters.

Example

To backstrip the sunrise drill site (located on shallower *continental* crust), and output all available parameters (via the `-d` option), we can run it from the command-line as:

```
python -m pybacktrack.backstrip_cli \
    -w pybacktrack_examples/example_data/sunrise_lithology.txt \
    -l primary extended \
    -d age compacted_depth compacted_thickness decompacted_thickness decompacted_density␣
↪decompacted_sediment_rate decompacted_depth min_tectonic_subsidence max_tectonic_
↪subsidence average_tectonic_subsidence min_water_depth max_water_depth average_water_
```

(continues on next page)

(continued from previous page)

```
↪depth lithology \
    -slm Haq87_SealevelCurve_Longterm \
    -o sunrise_backstrip_amended.txt \
    -- \
    sunrise_backstrip_decompacted.txt
```

…or write some Python code to do the same thing:

```
import pybacktrack

# Input and output filenames.
input_well_filename = 'pybacktrack_examples/example_data/sunrise_lithology.txt'
amended_well_output_filename = 'sunrise_backstrip_amended.txt'
decompacted_output_filename = 'sunrise_backstrip_decompacted.txt'

# Read input well file, and write amended well and decompacted results to output files.
pybacktrack.backstrip_and_write_well(
    decompacted_output_filename,
    input_well_filename,
    lithology_filenames=[pybacktrack.PRIMARY_BUNDLE_LITHOLOGY_FILENAME,
                         pybacktrack.EXTENDED_BUNDLE_LITHOLOGY_FILENAME],
    sea_level_model=pybacktrack.BUNDLE_SEA_LEVEL_MODELS['Haq87_SealevelCurve_Longterm'],
    decompacted_columns=[pybacktrack.BACKSTRIP_COLUMN_AGE,
                         pybacktrack.BACKSTRIP_COLUMN_COMPACTED_DEPTH,
                         pybacktrack.BACKSTRIP_COLUMN_COMPACTED_THICKNESS,
                         pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_THICKNESS,
                         pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_DENSITY,
                         pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_SEDIMENT_RATE,
                         pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_DEPTH,
                         pybacktrack.BACKSTRIP_COLUMN_MIN_TECTONIC_SUBSIDENCE,
                         pybacktrack.BACKSTRIP_COLUMN_MAX_TECTONIC_SUBSIDENCE,
                         pybacktrack.BACKSTRIP_COLUMN_AVERAGE_TECTONIC_SUBSIDENCE,
                         pybacktrack.BACKSTRIP_COLUMN_MIN_WATER_DEPTH,
                         pybacktrack.BACKSTRIP_COLUMN_MAX_WATER_DEPTH,
                         pybacktrack.BACKSTRIP_COLUMN_AVERAGE_WATER_DEPTH,
                         pybacktrack.BACKSTRIP_COLUMN_LITHOLOGY],
    # Might be an extra stratigraphic well layer added from well bottom to ocean␣
↪basement...
    ammended_well_output_filename=amended_well_output_filename)
```

Note: The drill site file `pybacktrack_examples/example_data/sunrise_lithology.txt` is part of the *example data*.

2.5.3 Backstrip output

For each stratigraphic layer in the input drill site file, `backstrip` can write one or more parameters to an output file.

Running the *above example* on the sunrise drill site:

```
# SiteLatitude = -9.5901
# SiteLongitude = 128.1538
# SurfaceAge = 0.0000
#
## bottom_age bottom_depth min_water_depth max_water_depth lithology
   2.000      462.000      0.000            100.000         Shale           0.20       ␣
↪Limestone         0.75      Dolostone        0.05
   10.000     525.000      0.000            100.000         Shale           0.20       ␣
↪Limestone         0.75      Dolostone        0.05
   24.000     822.000      0.000            100.000         Shale           0.10       ␣
↪Limestone         0.80      Sand             0.10
   30.000     1062.000     0.000            100.000         Shale           0.30       ␣
↪Limestone         0.55      Dolostone        0.05       Sand            0.10
   34.000     1086.000     0.000            100.000         Shale           0.20       ␣
↪Limestone         0.10      Sand             0.70
   45.000     1366.000     0.000            100.000         Shale           0.10       ␣
↪Limestone         0.75      Dolostone        0.05       Sand            0.10
   58.000     1442.000     0.000            100.000         Shale           0.15       ␣
↪Limestone         0.15      Sand             0.70
   68.000     1494.000     50.000           200.000         Shale           0.45       ␣
↪Limestone         0.50      Sand             0.05
   83.000     1521.000     20.000           200.000         Shale           0.30       ␣
↪Limestone         0.65      Sand             0.05
   86.000     1545.000     20.000           200.000         Shale           0.55       ␣
↪Limestone         0.35      Sand             0.10
   88.000     1582.000     20.000           200.000         Shale           0.35       ␣
↪Limestone         0.65
   90.000     1620.000     20.000           200.000         Shale           0.70       ␣
↪Limestone         0.15      Sand             0.15
   95.000     1890.000     20.000           200.000         Shale           0.70       ␣
↪Limestone         0.15      Sand             0.15
   100.000    2036.000     20.000           200.000         Shale           0.70       ␣
↪Limestone         0.15      Sand             0.15
   107.000    2062.000     20.000           200.000         Shale           0.64       ␣
↪Limestone         0.18      Sand             0.18
   125.000    2066.000     0.000            100.000         Shale           0.40       ␣
↪Chalk             0.10      Sand             0.50
   160.000    2068.000     0.000            100.000         Shale           0.40       ␣
↪Limestone         0.30      Sand             0.30
   165.000    2130.000     0.000            100.000         Shale           0.40       ␣
↪Limestone         0.30      Sand             0.30
   170.000    2176.000     0.000            100.000         Shale           0.50       ␣
↪Sand              0.50
   177.000    2187.000     -10.000          25.000          Shale           0.30       ␣
↪Sand              0.70
   180.000    2237.000     -10.000          25.000          Shale           0.30       ␣
↪Sand              0.70
   190.000    2311.000     -10.000          20.000          Shale           0.30       ␣
```

(continues on next page)

(continued from previous page)

```
↪Sand          0.70
```

…produces an *amended drill site output file*, and a *decompacted output file* containing the decompacted output parameters like sediment thickness and tectonic subsidence.

Amended drill site output

The amended drill site output file:

```
# SiteLatitude = -9.5901
# SiteLongitude = 128.1538
# SurfaceAge = 0.0000
#
## bottom_age bottom_depth min_water_depth max_water_depth lithology
   2.000      462.000      0.000           100.000          Shale           0.20      ␣
↪Limestone        0.75      Dolostone       0.05
   10.000     525.000      0.000           100.000          Shale           0.20      ␣
↪Limestone        0.75      Dolostone       0.05
   24.000     822.000      0.000           100.000          Shale           0.10      ␣
↪Limestone        0.80      Sand            0.10
   30.000     1062.000     0.000           100.000          Shale           0.30      ␣
↪Limestone        0.55      Dolostone       0.05      Sand            0.10
   34.000     1086.000     0.000           100.000          Shale           0.20      ␣
↪Limestone        0.10      Sand            0.70
   45.000     1366.000     0.000           100.000          Shale           0.10      ␣
↪Limestone        0.75      Dolostone       0.05      Sand            0.10
   58.000     1442.000     0.000           100.000          Shale           0.15      ␣
↪Limestone        0.15      Sand            0.70
   68.000     1494.000     50.000          200.000          Shale           0.45      ␣
↪Limestone        0.50      Sand            0.05
   83.000     1521.000     20.000          200.000          Shale           0.30      ␣
↪Limestone        0.65      Sand            0.05
   86.000     1545.000     20.000          200.000          Shale           0.55      ␣
↪Limestone        0.35      Sand            0.10
   88.000     1582.000     20.000          200.000          Shale           0.35      ␣
↪Limestone        0.65
   90.000     1620.000     20.000          200.000          Shale           0.70      ␣
↪Limestone        0.15      Sand            0.15
   95.000     1890.000     20.000          200.000          Shale           0.70      ␣
↪Limestone        0.15      Sand            0.15
   100.000    2036.000     20.000          200.000          Shale           0.70      ␣
↪Limestone        0.15      Sand            0.15
   107.000    2062.000     20.000          200.000          Shale           0.64      ␣
↪Limestone        0.18      Sand            0.18
   125.000    2066.000     0.000           100.000          Shale           0.40      ␣
↪Chalk            0.10      Sand            0.50
   160.000    2068.000     0.000           100.000          Shale           0.40      ␣
↪Limestone        0.30      Sand            0.30
   165.000    2130.000     0.000           100.000          Shale           0.40      ␣
↪Limestone        0.30      Sand            0.30
   170.000    2176.000     0.000           100.000          Shale           0.50      ␣
```

(continues on next page)

(continued from previous page)

```
↪Sand           0.50
  177.000    2187.000      -10.000         25.000          Shale           0.30      ␣
↪Sand           0.70
  180.000    2237.000      -10.000         25.000          Shale           0.30      ␣
↪Sand           0.70
  190.000    2311.000      -10.000         20.000          Shale           0.30      ␣
↪Sand           0.70
```

Note: No extra *base sediment layer* is added from the bottom of the drill site (2311 metres) to the total sediment thickness at the drill site (1298.15 metres), because the former (bottom of drill site) is already deeper than the latter (total sediment thickness). This happens because the *default total sediment thickness grid* is not as accurate near continental margins (compared to deeper ocean basins).

Decompacted output

The decompacted output file:

```
# SiteLatitude = -9.5901
# SiteLongitude = 128.1538
# SurfaceAge = 0.0000
#
# age       compacted_depth compacted_thickness decompacted_thickness decompacted_
↪density decompacted_sediment_rate decompacted_depth min_tectonic_subsidence max_
↪tectonic_subsidence average_tectonic_subsidence min_water_depth max_water_depth␣
↪average_water_depth lithology
  0.000     0.000           2311.000            2311.000              2089.479        ␣
↪ 245.712                    0.000              1236.930               1336.930       ␣
↪      1286.930                    0.000            100.000         50.000           ␣
↪Shale          0.20        Limestone       0.75        Dolostone       0.05
  2.000     462.000         1849.000            1984.750              2057.304        ␣
↪ 8.922                     491.425             1038.610               1138.610       ␣
↪      1088.610                    0.000            100.000         50.000           ␣
↪Shale          0.20        Limestone       0.75        Dolostone       0.05
  10.000    525.000         1786.000            1936.640              2052.112        ␣
↪ 24.613                    562.801             956.290                1056.290       ␣
↪      1006.290                    0.000            100.000         50.000           ␣
↪Shale          0.10        Limestone       0.80        Sand            0.10
  24.000    822.000         1489.000            1703.149              2018.885        ␣
↪ 50.826                    907.389             846.332                946.332        ␣
↪      896.332                     0.000            100.000         50.000           ␣
↪Shale          0.30        Limestone       0.55        Dolostone       0.05      Sand ␣
↪           0.10
  30.000    1062.000        1249.000            1493.707              1994.320        ␣
↪ 7.743                     1212.348            678.295                778.295        ␣
↪      728.295                     0.000            100.000         50.000           ␣
↪Shale          0.20        Limestone       0.10        Sand            0.70
  34.000    1086.000        1225.000            1472.172              1991.762        ␣
↪ 32.462                    1243.321            653.467                753.467        ␣
↪      703.467                     0.000            100.000         50.000           ␣
```

(continues on next page)

(continued from previous page)

```
↪Shale          0.10        Limestone       0.75       Dolostone       0.05        Sand ␣
↪          0.10
 45.000     1366.000      945.000          1223.986             1937.093             ␣
↪ 7.857                   1600.399         515.129              615.129              ␣
↪       565.129                  0.000          100.000       50.000                ␣
↪Shale          0.15        Limestone       0.15       Sand            0.70
 58.000     1442.000      869.000          1153.349             1921.307             ␣
↪ 7.583                   1702.542         546.589              696.589              ␣
↪       621.589                  50.000         200.000       125.000               ␣
↪Shale          0.45        Limestone       0.50       Sand            0.05
 68.000     1494.000      817.000          1100.849             1911.363             ␣
↪ 2.517                   1778.370         443.659              623.659              ␣
↪       533.659                  20.000         200.000       110.000               ␣
↪Shale          0.30        Limestone       0.65       Sand            0.05
 83.000     1521.000      790.000          1074.658             1904.627             ␣
↪ 12.270                  1816.128         439.905              619.905              ␣
↪       529.905                  20.000         200.000       110.000               ␣
↪Shale          0.55        Limestone       0.35       Sand            0.10
 86.000     1545.000      766.000          1049.084             1900.276             ␣
↪ 26.459                  1852.938         423.355              603.355              ␣
↪       513.355                  20.000         200.000       110.000               ␣
↪Shale          0.35        Limestone       0.65
 88.000     1582.000      729.000          1012.260             1890.539             ␣
↪ 31.590                  1905.856         388.195              568.195              ␣
↪       478.195                  20.000         200.000       110.000               ␣
↪Shale          0.70        Limestone       0.15       Sand            0.15
 90.000     1620.000      691.000          968.032              1884.877             ␣
↪ 92.186                  1969.037         343.109              523.109              ␣
↪       433.109                  20.000         200.000       110.000               ␣
↪Shale          0.70        Limestone       0.15       Sand            0.15
 95.000     1890.000      421.000          625.158              1845.618             ␣
↪ 51.506                  2429.968         160.243              340.243              ␣
↪       250.243                  20.000         200.000       110.000               ␣
↪Shale          0.70        Limestone       0.15       Sand            0.15
 100.000    2036.000      275.000          412.490              1835.863             ␣
↪ 6.452                   2687.497         111.343              291.343              ␣
↪       201.343                  20.000         200.000       110.000               ␣
↪Shale          0.64        Limestone       0.18       Sand            0.18
 107.000    2062.000      249.000          373.048              1835.892             ␣
↪ 0.375                   2732.658         109.605              209.605              ␣
↪       159.605                  0.000          100.000       50.000                ␣
↪Shale          0.40        Chalk           0.10       Sand            0.50
 125.000    2066.000      245.000          367.097              1835.856             ␣
↪ 0.090                   2739.405         145.550              245.550              ␣
↪       195.550                  0.000          100.000       50.000                ␣
↪Shale          0.40        Limestone       0.30       Sand            0.30
 160.000    2068.000      243.000          364.308              1835.425             ␣
↪ 19.655                  2742.563         198.479              298.479              ␣
↪       248.479                  0.000          100.000       50.000                ␣
↪Shale          0.40        Limestone       0.30       Sand            0.30
 165.000    2130.000      181.000          276.308              1821.056             ␣
↪ 15.434                  2840.839         117.435              217.435              ␣
```

(continues on next page)

(continued from previous page)

```
↪      167.435                        0.000          100.000          50.000          ␣
↪Shale             0.50       Sand             0.50
 170.000   2176.000        135.000           205.283             1822.654          ␣
↪ 2.458                    2918.011         116.777               151.777          ␣
↪      134.277                      -10.000          25.000          7.500           ␣
↪Shale             0.30       Sand             0.70
 177.000   2187.000        124.000           189.178             1820.541          ␣
↪ 26.161                   2935.218         133.251               168.251          ␣
↪      150.751                      -10.000          25.000          7.500           ␣
↪Shale             0.30       Sand             0.70
 180.000   2237.000        74.000            114.644             1810.669          ␣
↪ 11.696                   3013.701         74.454                104.454          ␣
↪      89.454                       -10.000          20.000          5.000           ␣
↪Shale             0.30       Sand             0.70
 190.000   2311.000        0.000             0.000               0.000             ␣
↪ 0.000                    3130.665         -10.000               20.000           ␣
↪      5.000                        -10.000          20.000          5.000           ␣
↪Shale             1.00
```

The *age*, *compacted_depth*, *min_water_depth*, *max_water_depth* and *lithology* columns are the same as the *bottom_age*, *bottom_depth*, *min_water_depth*, *max_water_depth* and *lithology* columns in the input drill site (except there is also a row associated with the surface age).

The *compacted_thickness* column is the bottom depth of the drill site (2311 metres - noting that there is no base sediment layer in the *amended drill site* above) minus *compacted_depth*. The *decompacted_thickness* column is the thickness of all sediment at the associated age. In other words, at each consecutive age another stratigraphic layer is essentially removed, allowing the underlying layers to expand (due to their porosity). At present day (or the surface age) the decompacted thickness is just the compacted thickness. And note that because no extra *base sediment layer* was added to the bottom of the drill site (2311 metres) the thickness and density is zero there. The *decompacted_density* column is the average density integrated over the decompacted thickness of the drill site (each stratigraphic layer contains a mixture of water and sediment according to its porosity at the decompacted depth of the layer). The *decompacted_sediment_rate* column is the rate of sediment deposition in units of metres/Ma. At each time it is calculated as the fully decompacted thickness (ie, using surface porosity only) of the surface stratigraphic layer (whose deposition ends at the specified time) divided by the layer's deposition time interval. The *decompacted_depth* column is similar to *decompacted_sediment_rate* in that the stratigraphic layers are fully decompacted (using surface porosity only) as if no portion of any layer had ever been buried. It is also similar to *compacted_depth* except all effects of compaction have been removed.

Finally, *average_water_depth* is just the average *min_water_depth* and *max_water_depth*. And *min_tectonic_subsidence*, *max_tectonic_subsidence* and *average_tectonic_subsidence* are obtained from *min_water_depth* and *max_water_depth* and *average_water_depth* by adding an isostatic correction of the decompacted sediment thickness (to obtain the deeper isostatically compensated, sediment-free water depth also known as tectonic subsidence).

Note: The output columns are specified using the `-d` command-line option (run `python -m pybacktrack.backstrip_cli --help` to see all options), or using the *decompacted_columns* argument of the `pybacktrack.backstrip_and_write_well()` function. By default, only *age* and *decompacted_thickness* are output.

2.5.4 Sea level variation

A model of the variation of sea level relative to present day can optionally be used when backstripping. This adjusts the isostatic correction of the decompacted sediment thickness to take into account sea-level variations.

There are two built-in sea level models *bundled* inside `backstrip`:

- `Haq87_SealevelCurve` - The Phanerozoic Record of Global Sea-Level Change
- `Haq87_SealevelCurve_Longterm` - Normalised to start at zero at present-day.

A sea-level model is optional. If one is not specified then sea-level variation is assumed to be zero.

Note: A built-in sea-level model can be specified using the `-slm` command-line option (run `python -m pybacktrack.backstrip_cli --help` to see all options), or using the *sea_level_model* argument of the `pybacktrack.backstrip_and_write_well()` function.

Note: It is also possible to specify your own sea-level model. This can be done by providing your own text file containing a column of ages (Ma) and a corresponding column of sea levels (m), and specifying the name of this file to the `-sl` command-line option or to the *sea_level_model* argument of the `pybacktrack.backstrip_and_write_well()` function.

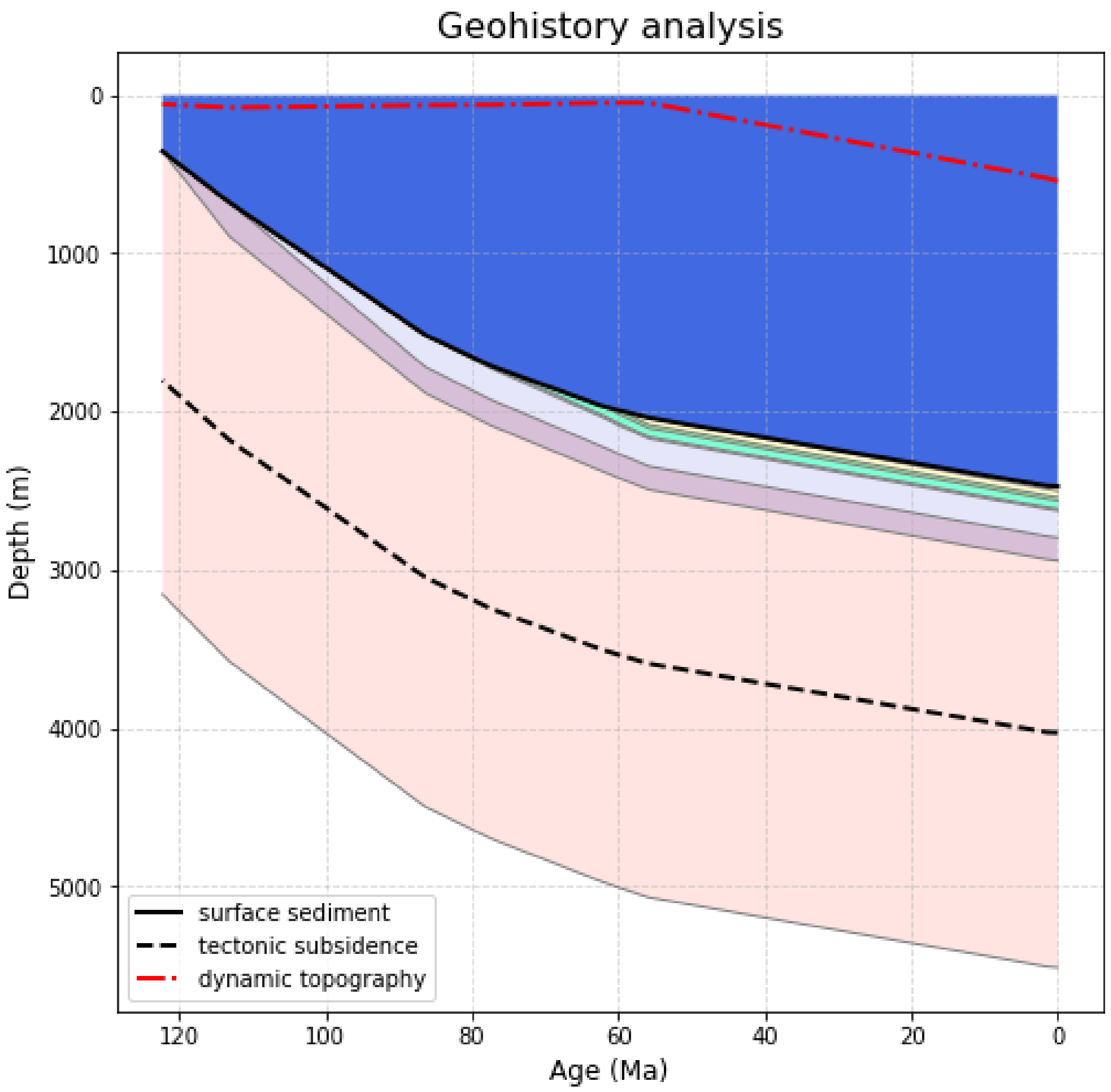
2.5.5 Geohistory analysis

The Decompacting Stratigraphic Layers notebook shows how to visualize the decompaction of stratigraphic layers at a drill site.

Note: The example notebooks are installed as part of the example data which can be installed by following *these instructions*.

One of the examples in that notebook demonstrates decompaction of a shallow continental drill site using backstripping. The paleo water depths (blue fill) are recorded in the drill site file and the tectonic subsidence (black dashed line) is backstripped using the paleo water depths and sediment decompaction. Note that, unlike backtracking, dynamic topography does *not* affect tectonic subsidence (because backstripping does *not* have a model of tectonic subsidence). So the image below is simply plotting dynamic topography alongside backstripped tectonic subsidence.

2.6 Paleobathymetry


Geohistory analysis
Depth (m)
Age (Ma)
0
1000
2000
3000
4000
5000
120
100
80
60
40
20
0
surface sediment
tectonic subsidence
dynamic topography

2.6.1 Overview

The `paleo_bathymetry` module is used to generate paleo bathymetry grids by reconstructing and backtracking present-day sediment-covered crust through time.

2.6.2 Running paleobathymetry

You can either run `paleo_bathymetry` as a built-in script, specifying parameters as command-line options (`...`):

```
python -m pybacktrack.paleo_bathymetry_cli ...
```

…or `import pybacktrack` into your own script, calling its functions and specifying parameters as function arguments (`...`):

```
import pybacktrack

pybacktrack.reconstruct_paleo_bathymetry_grids(...)
```

Note: You can run `python -m pybacktrack.paleo_bathymetry_cli --help` to see a description of all command-line options available, or see the *paleobathymetry reference section* for documentation on the function parameters.

Example

To generate paleobathymetry NetCDF grids at 12 minute resolution from 0Ma to 240Ma in 1Myr increments, we can run it from the command-line as:

```
python -m pybacktrack.paleo_bathymetry_cli \
    -gm 12 \
    -ym M7 \
    -m GDH1 \
    --use_all_cpus \
    -- \
    240 paleo_bathymetry_12m_M7_GDH1
```

…where the `-gm` option specifies the grid spacing (12 minutes), the `-ym` specifies the `M7` *dynamic topography model*, the `-m` option specifies the `GDH1` *oceanic subsidence model*, the `--use_all_cpus` option uses all CPUs (it also accepts an optional number of CPUs) and the generated paleobathymetry grid files are named `paleo_bathymetry_12m_M7_GDH1_<time>.nc`.

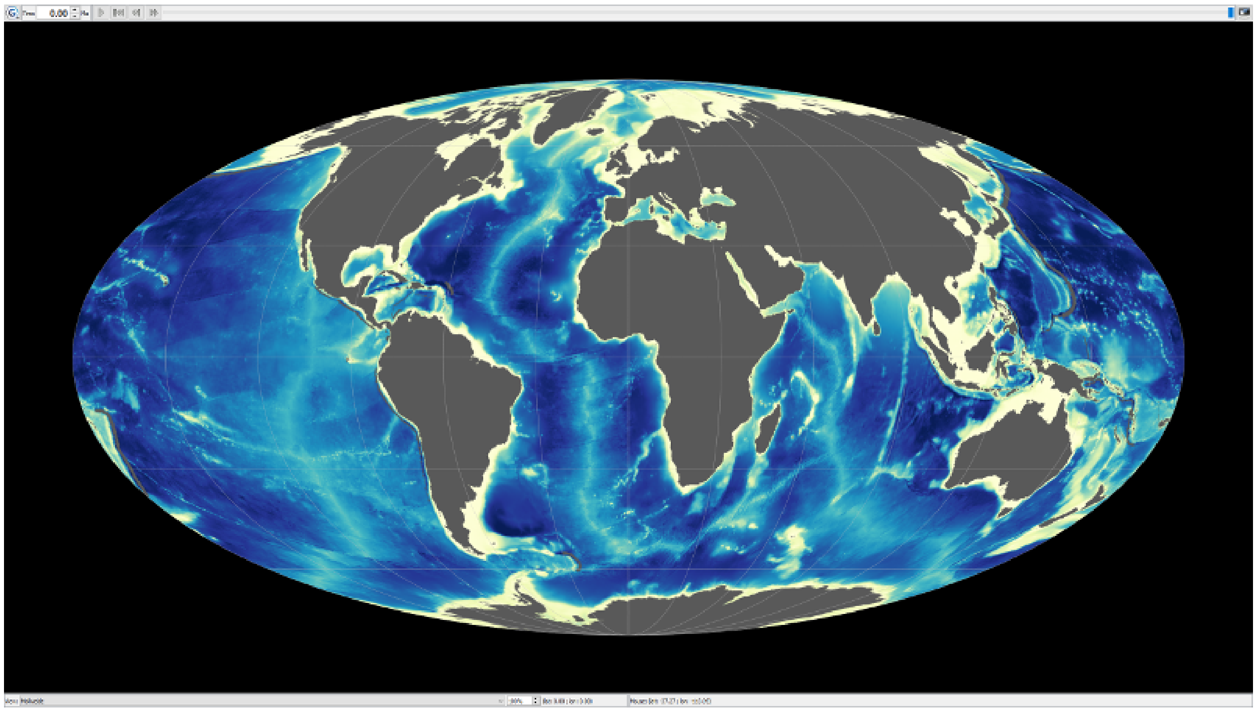
…or write some Python code to do the same thing:

```
import pybacktrack

pybacktrack.reconstruct_paleo_bathymetry_grids(
    'paleo_bathymetry_12m_M7_GDH1',
    0.2,  # degrees (same as 12 minutes)
    240,
    dynamic_topography_model='M7',
    ocean_age_to_depth_model=pybacktrack.AGE_TO_DEPTH_MODEL_GDH1,
    use_all_cpus=True)  # can also be an integer (the number of CPUs to use)
```

2.6.3 Paleobathymetry output

The following shows two of the 241 paleobathymetry NetCDF grids generated by the example above. They're both visualised in GPlates, the first at present day and the second at 60 Ma.



Also the Paleobathymetry notebook has a similar example.

Note: The example notebooks are installed as part of the example data which can be installed by following *these instructions*.

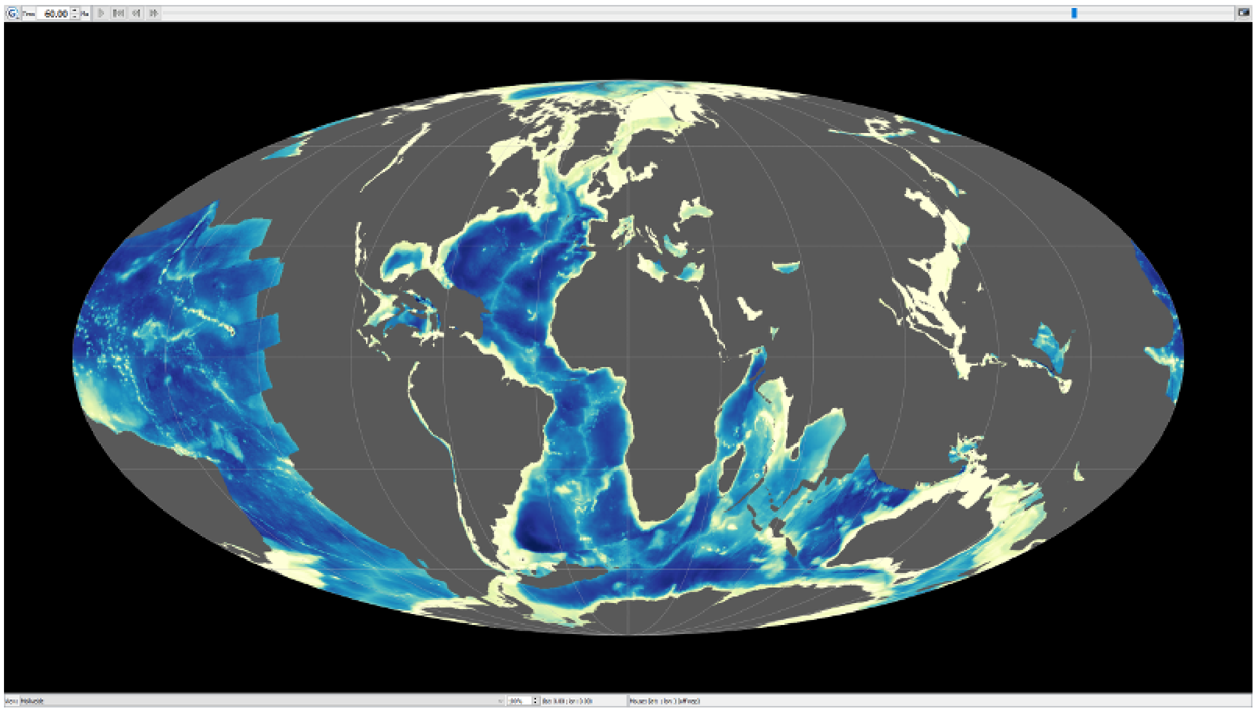
2.6.4 Paleobathymetry gridding procedure

Paleobathymetry gridding uses the *builtin rift start/end age grids* along with the existing subsidence models (continental rifting and oceanic) and the sediment decompaction functionality in pyBacktrack to generate paleo bathymetry grids (typically in 1 Myr intervals).

The `paleo_bathymetry` module has similar options to the `backtrack` module. Such as options for the present day grids containing age, bathymetry, crustal thickness and sediment thickness. And options for the dynamic topography and sea level models. And the defaults for those options are the same as the `backtrack` module (except for the default *lithology* - see below). For example, the `paleo_bathymetry` module defaults to the same *present-day ETOPO1 bathymetry grid*.

However the `paleo_bathymetry` module differs from the `backtrack` module in that, instead of a single point location for a drill site, a uniform grid of points containing sediment (inside valid regions of the total sediment thickness grid) are backtracked to obtain gridded paleo water depths through time.

Note: Sediment grid points near trenches are excluded by default to avoid deep bathymetry areas near trenches appearing in the reconstructed grids. Each trench has an exclusion distance on the subducting plate side (typically 60 kms) and an exlusion distance on the overriding plate side (typically 0 kms). And these per-trench distances are all built into



pyBacktrack. Any sediment grid points within these per-trench distances are excluded. However this masking near trenches can be removed by specifying `--exclude_distances_to_trenches_kms 0 0` (for example, in the *paleo bathymetry example above*).

As with regular backtracking, those sediment grid points lying inside the age grid (valid regions) use an oceanic subsidence model and those outside use a continental rifting model. However, in lieu of explicitly providing the rift start and end ages (as for a 1D drill site) each 2D grid point samples the builtin rift start/end age grids. Each grid point is also assigned a plate ID (using static polygons) and reconstructed back through time.

Each grid point has a single lithology, with an initial compacted thickness sampled from the total sediment thickness grid at present day that is progressively decompacted back through geological time.

Note: The single lithology defaults to `Average_ocean_floor_sediment` which is the average of the ocean floor sediment. This differs from the *base lithology of drill sites* where the undrilled portions of drill sites are usually below the Carbonate Compensation Depth (CCD) where shale dominates. Note that you can override the default lithology by specifying the `-b` command-line option.

The decompaction progresses incrementally (eg, in 1 Myr intervals) assuming a constant (average) decompacted sedimentation rate over the entire sedimentation period calculated as the fully decompacted initial thickness (ie, using surface porosity only) divided by the sedimentation period (from start of rifting for continental crust, and from crustal age for oceanic crust, to present day). Loading each reconstructed point's decompacted thicknesses onto its modelled tectonic subsidence (oceanic or continental) back through time, along with the effects of dynamic topography and sea level models, reveals its history of water depths. Finally, the reconstructed locations of all grid points and their reconstructed bathymetries are combined, at each reconstruction time, to create a history of paleo bathymetry grids.

Note: The supplementary script `pybacktrack/supplementary/merge_paleo_bathymetry_grids.py` can preferentially merge paleobathymetry grids produced by `pybacktrack` with externally produced paleobathymetry grids. This script first adds a user-specified dynamic topography to the external grids and then inserts only at grid locations not covered by the `pybacktrack` grids (eg, the external grids may contain paleobathymetry on subducted crust that is not covered by the reconstructed present-day sediment-deposited crust generated by `pybacktrack`). This script can be obtained by *installing the supplementary scripts*.

2.6.5 Builtin rift gridding procedure

PyBacktrack comes with two builtin grids containing rift start and end ages on submerged continental crust at 5 minute resolution. This is used during paleobathymetry gridding to obtain the rift periods of gridded points on continental crust. It is also used during regular backtracking to obtain the rift period of a drill site on continental crust (when it is not specified in the drill site file or on the command-line).

The rift grids cover all submerged continental crust, not just those areas that have undergone rifting. Submerged continental crust is where the total sediment thickness grid contains valid values but the age grid does not (ie, submerged crust that is non oceanic).

The rift grids were generated with `pybacktrack/supplementary/generate_rift_grids.py` using the Müller 2019 deforming plate model:

- Müller, R. D., Zahirovic, S., Williams, S. E., Cannon, J., Seton, M., Bower, D. J., Tetley, M. G., Heine, C., Le Breton, E., Liu, S., Russell, S. H. J., Yang, T., Leonard, J., and Gurnis, M. (2019), A global plate model including lithospheric deformation along major rifts and orogens since the Triassic. Tectonics, vol. 38,.

Note: The rift generation script `pybacktrack/supplementary/generate_rift_grids.py` can be obtained by *installing the supplementary scripts*.

This paragraph gives a brief overview of rift gridding… First, grid points on continental crust that have undergone *extensional* deformation (rifting) during their most recent deformation period have their rift start and end ages assigned as the start and end of that most recent deformation period (for each grid point). Next, grid points on continental crust that have undergone *contractional* deformation during their most recent deformation period have their rift periods set to default values (currently 200 to 0 Ma) to model these complex areas with simple rifting (despite a rifting model no longer strictly applying). So that covers the *deforming* grid points on continental crust. Next, the *non-deforming* grid points on continental crust obtain their rift period from the nearest deforming grid points. This ensures that all continental crust contains a rift period and hence can be used to generate paleobathymetry grids from all present day continental crust. Finally, only those continental grid points that are submerged are stored in the final rift grids since we only need to backtrack submerged crust.

This paragraph gives a more detailed explanation of how deformation in particular is used in `pybacktrack/supplementary/generate_rift_grids.py`… The script allows one to specify a total sediment thickness grid and an age grid (defaulting to those included with pyBacktrack). Grid points are uniformly generated in longitude/latitude space on continental crust. Next pyGPlates is used to load the Müller 2019 topological plate model (containing rigid plate polygons and deforming networks) and reconstruct these continental grid points on back through geological time. Note that plate IDs do not need to be explicitly assigned in order to be able to reconstruct because recent functionality in pyGPlates, known as *reconstructing by topologies*, essentially continually assigns plate IDs using the topological plate polygons and deforming networks while each grid point is reconstructed back through time. This ensures the path of each grid point is correctly reconstructed through deforming regions so that we can correctly determine when it enters and exits a deforming region. During this reconstruction each grid point is queried (at 1Myr intervals) whether it passes through a deforming network. The time at which a reconstructed grid point first encounters a deforming network (going backward in time) becomes its potential rift end time. Following that point further back in time we find when it first exits a deforming network (again going backward in time), which becomes its potential rift start time. We also keep track of a crustal stretching factor through time for each grid point so we can distinguish between extensional and contractional deformation.

2.7 Reference

This section documents the Python functions and classes that make up the public interface of the *pybacktrack* package.

The `pybacktrack` package has the `__version__` attribute:

```
import pybacktrack

pybacktrack.__version__
```

2.7.1 Backtracking

Find decompacted total sediment thickness and water depth through time.

Summary

`pybacktrack.backtrack_well()` finds decompacted total sediment thickness and water depth for each age in a well.

`pybacktrack.write_backtrack_well()` writes decompacted parameters as columns in a text file.

`pybacktrack.backtrack_and_write_well()` both backtracks well and writes decompacted data.

Detail

`pybacktrack.`**`backtrack_well`**(*well_filename*, *lithology_filenames=[pybacktrack.DEFAULT_BUNDLE_LITHOLOGY_FILENAME]*, *age_grid_filename=pybacktrack.BUNDLE_AGE_GRID_FILENAME*, *topography_filename=pybacktrack.BUNDLE_TOPOGRAPHY_FILENAME*, *total_sediment_thickness_filename=pybacktrack.BUNDLE_TOTAL_SEDIMENT_THICKNESS_FILE*
crustal_thickness_filename=pybacktrack.BUNDLE_CRUSTAL_THICKNESS_FILENAME, *dynamic_topography_model=None*, *sea_level_model=None*, *base_lithology_name=pybacktrack.DEFAULT_BASE_LITHOLOGY_NAME*, *ocean_age_to_depth_model=pybacktrack.AGE_TO_DEPTH_DEFAULT_MODEL*, *rifting_period=None*, *well_location=None*, *well_bottom_age_column=0*, *well_bottom_depth_column=1*, *well_lithology_column=2*)

Finds decompacted total sediment thickness and water depth for each age in a well.

Parameters

- **`well_filename`** (*`string`*) – Name of well text file.
- **`lithology_filenames`** (*`list of string, optional`*) – One or more text files containing lithologies.
- **`age_grid_filename`** (*`string, optional`*) – Age grid filename. Used to obtain age of seafloor at well location. Can be explicitly set to None if well site is known to be on continental crust (and hence age grid should be ignored). Note that this is different than not specifying a filename (since that will use the default bundled age grid).
- **`topography_filename`** (*`string, optional`*) – Topography filename. Used to obtain water depth at well location.
- **`total_sediment_thickness_filename`** (*`string, optional`*) – Total sediment thickness filename. Used to obtain total sediment thickness at well location. Can be explicitly set to None if well site is known to be drilled to basement depth (and hence total sediment thickness grid should be ignored). Note that this is different than not specifying a filename (since that will use the default bundled total sediment thickness grid).
- **`crustal_thickness_filename`** (*`string, optional`*) – Crustal thickness filename. Used to obtain crustal thickness at well location.
- **`dynamic_topography_model`** (*`string or tuple, optional`*) – Represents a time-dependent dynamic topography raster grid (in *mantle* frame).

  Can be either:

  – A string containing the name of a bundled dynamic topography model.

    Choices include `terra`, `M1`, `M2`, `M3`, `M4`, `M5`, `M6`, `M7`, `ngrand`, `s20rts`, `smean`, `AY18`, `KM16`, `D10_gmcm9` and `gld428`.

  – A tuple containing the three elements (dynamic topography list filename, static polygon filename, rotation filenames).

    The first tuple element is the filename of file containing list of dynamic topography grids (and associated times). Each row in this list file should contain two columns. First column containing filename (relative to list file) of a dynamic topography grid at a particular time. Second column containing associated time (in Ma). The second tuple element is the filename of file containing static polygons associated with dynamic topography model. This is used to assign plate ID to well location so it can be reconstructed. The third tuple element is the filename of the rotation file associated with model. Only the rotation file for static continents/oceans is needed (ie, deformation rotations not needed).

- **sea_level_model** (*string, optional*) – Used to obtain sea levels relative to present day. Can be either the name of a bundled sea level model, or a sea level filename. Bundled sea level models include `Haq87_SealevelCurve` and `Haq87_SealevelCurve_Longterm`.
- **base_lithology_name** (*string, optional*) – Lithology name of the stratigraphic unit at the base of the well (must be present in lithologies file). The stratigraphic units in the well might not record the full depth of sedimentation. The base unit covers the remaining depth from bottom of well to the total sediment thickness. Defaults to `Shale`.
- **ocean_age_to_depth_model** (*{pybacktrack.AGE_TO_DEPTH_MODEL_RHCW18, pybacktrack.AGE_TO_DEPTH_MODEL_CROSBY_2007, pybacktrack.AGE_TO_DEPTH_MODEL_GDH1} or function, optional*) – The model to use when converting ocean age to depth at well location (if on ocean floor - not used for continental passive margin). It can be one of the enumerated values, or a callable function accepting a single non-negative age parameter and returning depth (in metres).
- **rifting_period** (*tuple, optional*) – Optional time period of rifting (if on continental passive margin - not used for oceanic floor). If specified then should be a 2-tuple (rift_start_age, rift_end_age) where rift_start_age can be None (in which case rifting is considered instantaneous from a stretching point-of-view, not thermal). If specified then overrides value in well file (and value from builtin rift start/end grids). If well is on continental passive margin then at least rift end age should be specified either here or in well file, or well location must be inside rifting region of builtin rift start/end grids, otherwise a ValueError exception will be raised.
- **well_location** (*tuple, optional*) – Optional location of well. If not provided then is extracted from the `well_filename` file. If specified then overrides value in well file. If specified then must be a 2-tuple (longitude, latitude) in degrees.
- **well_bottom_age_column** (*int, optional*) – The column of well file containing bottom age. Defaults to 0.
- **well_bottom_depth_column** (*int, optional*) – The column of well file containing bottom depth. Defaults to 1.
- **well_lithology_column** (*int, optional*) – The column of well file containing lithology(s). Defaults to 2.

Returns

- *pybacktrack.Well* – The well read from `well_filename`. It may also be amended with a base stratigraphic unit from the bottom of the well to basement.
- list of *pybacktrack.DecompactedWell* – The decompacted wells associated with the well. There is one decompacted well per age, in same order (and ages) as the well units (youngest to oldest).

Raises

- **ValueError** – If `lithology_column` is not the largest column number (must be last column).
- **ValueError** – If `well_location` is not specified *and* the well location was not extracted from the well file.
- **ValueError** – If well is on continental passive margin but rift end age was not specified by user and was not extracted from well file, and well location was not inside rifting region of builtin rift start/end grids.

Notes

Each attribute to read from well file (eg, bottom_age, bottom_depth, etc) has a column index to direct which column it should be read from.

The tectonic subsidence at each age (of decompacted wells) is added as a *tectonic_subsidence* attribute to each decompacted well returned.

`pybacktrack.`**`write_backtrack_well`**(*decompacted_wells*, *decompacted_wells_filename*, *well*, *well_attributes=None*, *decompacted_columns=[0, 1]*)

write_backtrack_well(decompacted_wells, decompacted_wells_filename, well, well_attributes=None, decompacted_columns=pybacktrack.BACKTRACK_DEFAULT_DECOMPACTED_COLUMNS): Write decompacted parameters as columns in a text file.

Parameters

- **`decompacted_wells`** (sequence of *`pybacktrack.DecompactedWell`*) – The decompacted wells returned by *`pybacktrack.backtrack_well()`*.
- **`decompacted_wells_filename`** (*`string`*) – Name of output text file.
- **`well`** (*`pybacktrack.Well`*) – The well to extract metadata from.
- **`well_attributes`** (*`dict, optional`*) – Optional attributes in *`pybacktrack.Well`* object to write to well file metadata. If specified then must be a dictionary mapping each attribute name to a metadata name. For example, `{'longitude' : 'SiteLongitude', 'latitude' : 'SiteLatitude'}`. will write `well.longitude` (if not None) to metadata 'SiteLongitude', etc. Not that the attributes must exist in `well` (but can be set to None).
- **`decompacted_columns`** (*`list of columns, optional`*) – The decompacted columns (and their order) to output to `decompacted_wells_filename`.

  Available columns are:

  – pybacktrack.BACKTRACK_COLUMN_AGE

  – pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_THICKNESS

  – pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DENSITY

  – pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_SEDIMENT_RATE

  – pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DEPTH

  – pybacktrack.BACKTRACK_COLUMN_DYNAMIC_TOPOGRAPHY

  – pybacktrack.BACKTRACK_COLUMN_TECTONIC_SUBSIDENCE

  – pybacktrack.BACKTRACK_COLUMN_WATER_DEPTH

  – pybacktrack.BACKTRACK_COLUMN_COMPACTED_THICKNESS

  – pybacktrack.BACKTRACK_COLUMN_LITHOLOGY

  – pybacktrack.BACKTRACK_COLUMN_COMPACTED_DEPTH

Raises

- **`ValueError`** – If an unrecognised value is encountered in `decompacted_columns`.
- **`ValueError`** – If `pybacktrack.BACKTRACK_COLUMN_LITHOLOGY` is specified in `decompacted_columns` but is not the last column.

`pybacktrack.`**`backtrack_and_write_well`**(*decompacted_output_filename*, *well_filename*, *lithology_filenames=[pybacktrack.DEFAULT_BUNDLE_LITHOLOGY_FILENAME]*, *age_grid_filename=pybacktrack.BUNDLE_AGE_GRID_FILENAME*, *topography_filename=pybacktrack.BUNDLE_TOPOGRAPHY_FILENAME*, *total_sediment_thickness_filename=pybacktrack.BUNDLE_TOTAL_SEDIMENT_THICK* *crustal_thickness_filename=pybacktrack.BUNDLE_CRUSTAL_THICKNESS_FILENA* *dynamic_topography_model=None*, *sea_level_model=None*, *base_lithology_name=pybacktrack.DEFAULT_BASE_LITHOLOGY_NAME*, *ocean_age_to_depth_model=pybacktrack.AGE_TO_DEPTH_DEFAULT_MODEL*, *rifting_period=None*, *decompacted_columns=pybacktrack.BACKTRACK_DEFAULT_DECOMPACTED_COLUMN* *well_location=None*, *well_bottom_age_column=0*, *well_bottom_depth_column=1*, *well_lithology_column=2*, *ammended_well_output_filename=None*)

Same as *`pybacktrack.backtrack_well()`* but also writes decompacted results to a text file.

Also optionally write amended well data (ie, including extra stratigraphic base unit from well bottom to ocean basement) to `ammended_well_output_filename` if specified.

Parameters

- **`decompacted_output_filename`** (*`string`*) – Name of text file to write decompacted results to.
- **`well_filename`** (*`string`*) – Name of well text file.
- **`lithology_filenames`** (*`list of string, optional`*) – One or more text files containing lithologies.
- **`age_grid_filename`** (*`string, optional`*) – Age grid filename. Used to obtain age of seafloor at well location. Can be explicitly set to None if well site is known to be on continental crust (and hence age grid should be ignored). Note that this is different than not specifying a filename (since that will use the default bundled age grid).
- **`topography_filename`** (*`string, optional`*) – Topography filename. Used to obtain water depth at well location.
- **`total_sediment_thickness_filename`** (*`string, optional`*) – Total sediment thickness filename. Used to obtain total sediment thickness at well location. Can be explicitly set to None if well site is known to be drilled to basement depth (and hence total sediment thickness grid should be ignored). Note that this is different than not specifying a filename (since that will use the default bundled total sediment thickness grid).
- **`crustal_thickness_filename`** (*`string, optional`*) – Crustal thickness filename. Used to obtain crustal thickness at well location.
- **`dynamic_topography_model`** (*`string or tuple, optional`*) – Represents a time-dependent dynamic topography raster grid. Currently only used for oceanic floor (ie, well location inside age grid) it is not used if well is on continental crust (passive margin).

  Can be either:

  – A string containing the name of a bundled dynamic topography model.

    Choices include `terra`, `M1`, `M2`, `M3`, `M4`, `M5`, `M6`, `M7`, `ngrand`, `s20rts`, `smean`, `AY18`, `KM16`, `D10_gmcm9` and `gld428`.

  – A tuple containing the three elements (dynamic topography list filename, static polygon filename, rotation filenames).

The first tuple element is the filename of file containing list of dynamic topography grids (and associated times). Each row in this list file should contain two columns. First column containing filename (relative to list file) of a dynamic topography grid at a particular time. Second column containing associated time (in Ma). The second tuple element is the filename of file containing static polygons associated with dynamic topography model. This is used to assign plate ID to well location so it can be reconstructed. The third tuple element is the filename of the rotation file associated with model. Only the rotation file for static continents/oceans is needed (ie, deformation rotations not needed).

- **sea_level_model** (*string, optional*) – Used to obtain sea levels relative to present day. Can be either the name of a bundled sea level model, or a sea level filename. Bundled sea level models include `Haq87_SealevelCurve` and `Haq87_SealevelCurve_Longterm`.
- **base_lithology_name** (*string, optional*) – Lithology name of the stratigraphic unit at the base of the well (must be present in lithologies file). The stratigraphic units in the well might not record the full depth of sedimentation. The base unit covers the remaining depth from bottom of well to the total sediment thickness. Defaults to `Shale`.
- **ocean_age_to_depth_model** (*{pybacktrack.AGE_TO_DEPTH_MODEL_RHCW18, pybacktrack.AGE_TO_DEPTH_MODEL_CROSBY_2007, pybacktrack.AGE_TO_DEPTH_MODEL_GDH1} or function, optional*) – The model to use when converting ocean age to depth at well location (if on ocean floor - not used for continental passive margin). It can be one of the enumerated values, or a callable function accepting a single non-negative age parameter and returning depth (in metres).
- **rifting_period** (*tuple, optional*) – Optional time period of rifting (if on continental passive margin - not used for oceanic floor). If specified then should be a 2-tuple (rift_start_age, rift_end_age) where rift_start_age can be None (in which case rifting is considered instantaneous from a stretching point-of-view, not thermal). If specified then overrides value in well file (and value from builtin rift start/end grids). If well is on continental passive margin then at least rift end age should be specified either here or in well file, or well location must be inside rifting region of builtin rift start/end grids, otherwise a ValueError exception will be raised.
- **decompacted_columns** (*list of columns, optional*) – The decompacted columns (and their order) to output to `decompacted_wells_filename`.

  Available columns are:

  – pybacktrack.BACKTRACK_COLUMN_AGE

  – pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_THICKNESS

  – pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DENSITY

  – pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_SEDIMENT_RATE

  – pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DEPTH

  – pybacktrack.BACKTRACK_COLUMN_DYNAMIC_TOPOGRAPHY

  – pybacktrack.BACKTRACK_COLUMN_TECTONIC_SUBSIDENCE

  – pybacktrack.BACKTRACK_COLUMN_WATER_DEPTH

  – pybacktrack.BACKTRACK_COLUMN_COMPACTED_THICKNESS

  – pybacktrack.BACKTRACK_COLUMN_LITHOLOGY

  – pybacktrack.BACKTRACK_COLUMN_COMPACTED_DEPTH

- **well_location** (*tuple, optional*) – Optional location of well. If not provided then is extracted from the `well_filename` file. If specified then overrides value in well file. If specified then must be a 2-tuple (longitude, latitude) in degrees.
- **well_bottom_age_column** (*int, optional*) – The column of well file containing bottom age. Defaults to 0.
- **well_bottom_depth_column** (*int, optional*) – The column of well file containing bottom depth. Defaults to 1.
- **well_lithology_column** (*int, optional*) – The column of well file containing lithology(s). Defaults to 2.
- **ammended_well_output_filename** (*string, optional*) – Amended well data filename. Useful if an extra stratigraphic base unit is added from well bottom to ocean basement.

Raises

- **ValueError** – If `lithology_column` is not the largest column number (must be last column).
- **ValueError** – If `well_location` is not specified *and* the well location was not extracted from the well file.
- **ValueError** – If well is on continental passive margin but rift end age was not specified by user and was not extracted from well file, and well location was not inside rifting region of builtin rift start/end grids.

Notes

Each attribute to read from well file (eg, bottom_age, bottom_depth, etc) has a column index to direct which column it should be read from.

2.7.2 Backstripping

Find decompacted total sediment thickness and tectonic subsidence through time.

Summary

pybacktrack.backstrip_well() finds decompacted total sediment thickness and tectonic subsidence for each age in a well.

pybacktrack.write_backstrip_well() writes decompacted parameters as columns in a text file.

Detail

pybacktrack.**backstrip_well**(*well_filename, lithology_filenames=[pybacktrack.DEFAULT_BUNDLE_LITHOLOGY_FILENAME], total_sediment_thickness_filename=pybacktrack.BUNDLE_TOTAL_SEDIMENT_THICKNESS_FILE*
sea_level_model=None, base_lithology_name=pybacktrack.DEFAULT_BASE_LITHOLOGY_NAME, well_location=None, well_bottom_age_column=0, well_bottom_depth_column=1, well_min_water_depth_column=2, well_max_water_depth_column=3, well_lithology_column=4)

Finds decompacted total sediment thickness and tectonic subsidence for each age in well.

Parameters

- **`well_filename`** (*`str`*) – Name of well text file.
- **`lithology_filenames`** (*`list of string, optional`*) – One or more text files containing lithologies.
- **`total_sediment_thickness_filename`** (*`str, optional`*) – Total sediment thickness filename. Used to obtain total sediment thickness at well location. Can be explicitly set to None if well site is known to be drilled to basement depth (and hence total sediment thickness grid should be ignored). Note that this is different than not specifying a filename (since that will use the default bundled total sediment thickness grid).
- **`sea_level_model`** (*`string, optional`*) – Used to obtain sea levels relative to present day. Can be either the name of a bundled sea level model, or a sea level filename. Bundled sea level models include `Haq87_SealevelCurve` and `Haq87_SealevelCurve_Longterm`.
- **`base_lithology_name`** (*`string, optional`*) – Lithology name of the stratigraphic unit at the base of the well (must be present in lithologies file). The stratigraphic units in the well might not record the full depth of sedimentation. The base unit covers the remaining depth from bottom of well to the total sediment thickness. Defaults to `Shale`.
- **`well_location`** (*`tuple, optional`*) – Optional location of well. If not provided then is extracted from the `well_filename` file. If specified then overrides value in well file. If specified then must be a 2-tuple (longitude, latitude) in degrees.
- **`well_bottom_age_column`** (*`int, optional`*) – The column of well file containing bottom age. Defaults to 0.
- **`well_bottom_depth_column`** (*`int, optional`*) – The column of well file containing bottom depth. Defaults to 1.
- **`well_min_water_depth_column`** (*`int, optional`*) – The column of well file containing minimum water depth. Defaults to 2.
- **`well_max_water_depth_column`** (*`int, optional`*) – The column of well file containing maximum water depth. Defaults to 3.
- **`well_lithology_column`** (*`int, optional`*) – The column of well file containing lithology(s). Defaults to 4.

Returns

- *`pybacktrack.Well`* – The well read from `well_filename`. It may also be amended with a base stratigraphic unit from the bottom of the well to basement.
- list of *`pybacktrack.DecompactedWell`* – The decompacted wells associated with the well. There is one decompacted well per age, in same order (and ages) as the well units (youngest to oldest).

Raises

- **`ValueError`** – If `well_lithology_column` is not the largest column number (must be last column).
- **`ValueError`** – If `well_location` is not specified *and* the well location was not extracted from the well file.

Notes

Each attribute to read from well file (eg, *bottom_age*, *bottom_depth*, etc) has a column index to direct which column it should be read from.

The min/max paleo water depths at each age (of decompacted wells) are added as *min_water_depth* and *max_water_depth* attributes to each decompacted well returned.

`pybacktrack.`**`write_backstrip_well`**`(`*decompacted_wells*, *decompacted_wells_filename*, *well*, *well_attributes=None*, *decompacted_columns=[0, 1]*`)`

write_backstrip_well(decompacted_wells, decompacted_wells_filename, well, well_attributes=None, decompacted_columns=pybacktrack.BACKTRACK_DEFAULT_DECOMPACTED_COLUMNS): Write decompacted parameters as columns in a text file.

Parameters

- **decompacted_wells** (sequence of *`pybacktrack.DecompactedWell`*) – The decompacted wells returned by *`pybacktrack.backstrip_well()`*.
- **decompacted_wells_filename** (*`string`*) – Name of output text file.
- **well** (*`pybacktrack.Well`*) – The well to extract metadata from.
- **well_attributes** (*`dict, optional`*) – Optional attributes in *`pybacktrack.Well`* object to write to well file metadata. If specified then must be a dictionary mapping each attribute name to a metadata name. For example, `{'longitude' : 'SiteLongitude', 'latitude' : 'SiteLatitude'}`. will write `well.longitude` (if not None) to metadata 'SiteLongitude', etc. Not that the attributes must exist in `well` (but can be set to None).
- **decompacted_columns** (*`list of columns, optional`*) – The decompacted columns (and their order) to output to `decompacted_wells_filename`.

  Available columns are:

  – pybacktrack.BACKSTRIP_COLUMN_AGE

  – pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_THICKNESS

  – pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_DENSITY

  – pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_SEDIMENT_RATE

  – pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_DEPTH

  – pybacktrack.BACKSTRIP_COLUMN_AVERAGE_TECTONIC_SUBSIDENCE

  – pybacktrack.BACKSTRIP_COLUMN_MIN_TECTONIC_SUBSIDENCE

  – pybacktrack.BACKSTRIP_COLUMN_MAX_TECTONIC_SUBSIDENCE

  – pybacktrack.BACKSTRIP_COLUMN_AVERAGE_WATER_DEPTH

  – pybacktrack.BACKSTRIP_COLUMN_MIN_WATER_DEPTH

  – pybacktrack.BACKSTRIP_COLUMN_MAX_WATER_DEPTH

  – pybacktrack.BACKSTRIP_COLUMN_COMPACTED_THICKNESS

  – pybacktrack.BACKSTRIP_COLUMN_LITHOLOGY

  – pybacktrack.BACKSTRIP_COLUMN_COMPACTED_DEPTH

Raises

- **ValueError** – If an unrecognised value is encountered in `decompacted_columns`.

- **ValueError** – If `pybacktrack.BACKSTRIP_COLUMN_LITHOLOGY` is specified in `decompacted_columns` but is not the last column.

`pybacktrack.`**`backstrip_and_write_well`**(*decompacted_output_filename*, *well_filename*, *lithology_filenames=[pybacktrack.DEFAULT_BUNDLE_LITHOLOGY_FILENAME]*, *total_sediment_thickness_filename=pybacktrack.BUNDLE_TOTAL_SEDIMENT_THICK* *sea_level_model=None*, *base_lithology_name=pybacktrack.DEFAULT_BASE_LITHOLOGY_NAME*, *decompacted_columns=DEFAULT_DECOMPACTED_COLUMNS*, *well_location=None*, *well_bottom_age_column=0*, *well_bottom_depth_column=1*, *well_min_water_depth_column=2*, *well_max_water_depth_column=3*, *well_lithology_column=4*, *ammended_well_output_filename=None*)

Same as *`pybacktrack.backstrip_well()`* but also writes decompacted results to a text file.

Also optionally write amended well data (ie, including extra stratigraphic base unit from well bottom to ocean basement) to `ammended_well_output_filename` if specified.

Parameters

- **decompacted_output_filename** (*string*) – Name of text file to write decompacted results to.
- **well_filename** (*string*) – Name of well text file.
- **lithology_filenames** (*list of string, optional*) – One or more text files containing lithologies.
- **total_sediment_thickness_filename** (*string, optional*) – Total sediment thickness filename. Used to obtain total sediment thickness at well location.
- **sea_level_model** (*string, optional*) – Used to obtain sea levels relative to present day. Can be either the name of a bundled sea level model, or a sea level filename. Bundled sea level models include `Haq87_SealevelCurve` and `Haq87_SealevelCurve_Longterm`.
- **base_lithology_name** (*string, optional*) – Lithology name of the stratigraphic unit at the base of the well (must be present in lithologies file). The stratigraphic units in the well might not record the full depth of sedimentation. The base unit covers the remaining depth from bottom of well to the total sediment thickness. Defaults to `Shale`.
- **decompacted_columns** (*list of columns, optional*) – The decompacted columns (and their order) to output to `decompacted_wells_filename`.

  Available columns are:

  – pybacktrack.BACKSTRIP_COLUMN_AGE
  – pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_THICKNESS
  – pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_DENSITY
  – pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_SEDIMENT_RATE
  – pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_DEPTH
  – pybacktrack.BACKSTRIP_COLUMN_AVERAGE_TECTONIC_SUBSIDENCE
  – pybacktrack.BACKSTRIP_COLUMN_MIN_TECTONIC_SUBSIDENCE
  – pybacktrack.BACKSTRIP_COLUMN_MAX_TECTONIC_SUBSIDENCE
  – pybacktrack.BACKSTRIP_COLUMN_AVERAGE_WATER_DEPTH

  - pybacktrack.BACKSTRIP_COLUMN_MIN_WATER_DEPTH
  - pybacktrack.BACKSTRIP_COLUMN_MAX_WATER_DEPTH
  - pybacktrack.BACKSTRIP_COLUMN_COMPACTED_THICKNESS
  - pybacktrack.BACKSTRIP_COLUMN_LITHOLOGY
  - pybacktrack.BACKSTRIP_COLUMN_COMPACTED_DEPTH
- **well_location** (*tuple, optional*) – Optional location of well. If not provided then is extracted from the `well_filename` file. If specified then overrides value in well file. If specified then must be a 2-tuple (longitude, latitude) in degrees.
- **well_bottom_age_column** (*int, optional*) – The column of well file containing bottom age. Defaults to 0.
- **well_bottom_depth_column** (*int, optional*) – The column of well file containing bottom depth. Defaults to 1.
- **well_min_water_depth_column** (*int, optional*) – The column of well file containing minimum water depth. Defaults to 2.
- **well_max_water_depth_column** (*int, optional*) – The column of well file containing maximum water depth. Defaults to 3.
- **well_lithology_column** (*int, optional*) – The column of well file containing lithology(s). Defaults to 4.
- **ammended_well_output_filename** (*string, optional*) – Amended well data filename. Useful if an extra stratigraphic base unit is added from well bottom to ocean basement.

Raises

- **ValueError** – If `well_lithology_column` is not the largest column number (must be last column).
- **ValueError** – If `well_location` is not specified *and* the well location was not extracted from the well file.

Notes

Each attribute to read from well file (eg, *bottom_age*, *bottom_depth*, etc) has a column index to direct which column it should be read from.

The min/max paleo water depths at each age (of decompacted wells) are added as *min_water_depth* and *max_water_depth* attributes to each decompacted well returned.

2.7.3 Paleobathymetry

Generate paleo bathymetry grids through time.

Summary

pybacktrack.generate_lon_lat_points() generates a global grid of points uniformly spaced in longitude and latitude.

pybacktrack.reconstruct_paleo_bathymetry() reconstructs and backtracks sediment-covered crust through time to get paleo bathymetry.

pybacktrack.write_paleo_bathymetry_grids() grid paleo bathymetry into NetCDF grids files.

pybacktrack.reconstruct_paleo_bathymetry_grids() generates a global grid of points, reconstructs/backtracks their bathymetry and writes paleo bathymetry grids.

Detail

pybacktrack.**generate_lon_lat_points**(*grid_spacing_degrees*)

Generates a global grid of points uniformly spaced in longitude and latitude.

Parameters
grid_spacing_degrees (*float*) – Spacing between points (in degrees).

Return type
list of (longitude, latitude) tuples

Raises
ValueError – If `grid_spacing_degrees` is negative or zero.

Notes

Longitudes start at -180 (dateline) and latitudes start at -90. If 180 is an integer multiple of `grid_spacing_degrees` then the final longitude is also on the dateline (+180).

New in version 1.4.

pybacktrack.**reconstruct_paleo_bathymetry**(*input_points*, *oldest_time=None*, *time_increment=1*, *lithology_filenames=[pybacktrack.DEFAULT_BUNDLE_LITHOLOGY_FILENAME]*, *age_grid_filename=pybacktrack.BUNDLE_AGE_GRID_FILENAME*, *topography_filename=pybacktrack.BUNDLE_TOPOGRAPHY_FILENAME*, *total_sediment_thickness_filename=pybacktrack.BUNDLE_TOTAL_SEDIMENT_T*
crustal_thickness_filename=pybacktrack.BUNDLE_CRUSTAL_THICKNESS_FIL
rotation_filenames=pybacktrack.bundle_data.BUNDLE_RECONSTRUCTION_ROTA
static_polygon_filename=pybacktrack.bundle_data.BUNDLE_RECONSTRUCTIO
dynamic_topography_model=None, *sea_level_model=None*, *lithology_name=pybacktrack.DEFAULT_PALEO_BATHYMETRY_LITHOLOGY_NAM*
ocean_age_to_depth_model=pybacktrack.AGE_TO_DEPTH_DEFAULT_MODEL
exclude_distances_to_trenches_kms=None, *region_plate_ids=None*, *anchor_plate_id=0*, *output_positive_bathymetry_below_sea_level=False*, *use_all_cpus=False*)

Reconstructs and backtracks sediment-covered crust through time to get paleo bathymetry.

Parameters

- **input_points** (*sequence of (longitude, latitude) tuples*) – The point locations to sample bathymetry at present day. Note that any samples outside the masked region of the total sediment thickness grid are ignored.
- **oldest_time** (*float, optional*) – The oldest time (in Ma) that output is generated back to (from present day). Value must not be negative. If not specified then the oldest of oceanic crustal ages (for those input points on oceanic crust) and rift start ages (for those input points on continental crust) is used instead.
- **time_increment** (*float*) – The time increment (in My) that output is generated (from present day back to oldest time). Value must be positive.
- **lithology_filenames** (*list of string, optional*) – One or more text files containing lithologies.
- **age_grid_filename** (*string, optional*) – Age grid filename. Used to obtain age of oceanic crust at present day. Crust is oceanic at locations inside masked age grid region, and continental outside.
- **topography_filename** (*string, optional*) – Topography filename. Used to obtain bathymetry at present day.
- **total_sediment_thickness_filename** (*string, optional*) – Total sediment thickness filename. Used to obtain total sediment thickness at present day.
- **crustal_thickness_filename** (*string, optional*) – Crustal thickness filename. Used to obtain crustal thickness at present day.
- **rotation_filenames** (*list of string, optional*) – List of filenames containing rotation features (to reconstruct sediment-deposited crust). If not specified then defaults to the built-in global rotations associated with the topological model used to generate the built-in rift start/end time grids.
- **static_polygon_filename** (*string, optional*) – Filename containing static polygon features (to assign plate IDs to points on sediment-deposited crust). If not specified then defaults to the built-in static polygons associated with the topological model used to generate the built-in rift start/end time grids.
- **dynamic_topography_model** (*string or tuple, optional*) – Represents a time-dependent dynamic topography raster grid (in *mantle* frame).

  Can be either:

  – A string containing the name of a bundled dynamic topography model.

    Choices include `terra`, `M1`, `M2`, `M3`, `M4`, `M5`, `M6`, `M7`, `ngrand`, `s20rts`, `smean`, `AY18`, `KM16`, `D10_gmcm9` and `gld428`.

  – A tuple containing the three elements (dynamic topography list filename, static polygon filename, rotation filenames).

    The first tuple element is the filename of file containing list of dynamic topography grids (and associated times). Each row in this list file should contain two columns. First column containing filename (relative to list file) of a dynamic topography grid at a particular time. Second column containing associated time (in Ma). The second tuple element is the filename of file containing static polygons associated with dynamic topography model. This is used to assign plate ID to a location so it can be reconstructed. The third tuple element is the filename of the rotation file associated with model. Only the rotation file for static continents/oceans is needed (ie, deformation rotations not needed).

- **sea_level_model** (*string, optional*) – Used to obtain sea levels relative to present day. Can be either the name of a bundled sea level model, or a sea level filename. Bundled sea level models include `Haq87_SealevelCurve` and `Haq87_SealevelCurve_Longterm`.
- **lithology_name** (*string, optional*) – Lithology name of the all sediment (must be present in lithologies file). The total sediment thickness at all sediment locations consists of a single lithology. Defaults to `Average_ocean_floor_sediment`.
- **ocean_age_to_depth_model** (*{pybacktrack.AGE_TO_DEPTH_MODEL_RHCW18, pybacktrack.AGE_TO_DEPTH_MODEL_CROSBY_2007, pybacktrack.AGE_TO_DEPTH_MODEL_GDH1} or function, optional*) – The model to use when converting ocean age to depth at a location (if on ocean floor - not used for continental passive margin). It can be one of the enumerated values, or a callable function accepting a single non-negative age parameter and returning depth (in metres).
- **exclude_distances_to_trenches_kms** (*2-tuple of float, optional*) – The two distances to present-day trenches (on subducting and overriding sides, in that order) to exclude bathymetry grid points (in kms), or None to use built-in per-trench defaults. Default is None.
- **region_plate_ids** (*list of int, optional*) – Plate IDs of one or more plates to restrict paleobathymetry reconstruction to. Defaults to global.
- **anchor_plate_id** (*int, optional*) – The anchor plate id used when reconstructing paleobathymetry grid points. Defaults to zero.
- **output_positive_bathymetry_below_sea_level** (*bool, optional*) – Whether to output positive bathymetry values below sea level (the same as backtracked water depths at a drill site). However topography/bathymetry grids typically have negative values below sea level (and positive above). So the default (`False`) matches typical topography/bathymetry grids (ie, outputs negative bathymetry values below sea level).
- **use_all_cpus** (*bool or int, optional*) – If `False` (or zero) then use a single CPU. If `True` then distribute CPU processing across all CPUs (cores). If a positive integer then use that many CPUs (cores). Defaults to `False` (single CPU).

Returns

The reconstructed paleo bathymetry points from present day to the oldest time (see `oldest_time`) in increments of `time_increment`. Each key in the returned dict is one of those times and each value in the dict is a list of reconstructed paleo bathymetries represented as a 3-tuple containing reconstructed longitude, reconstructed latitude and paleo bathmetry.

Return type

dict mapping each time to a list of 3-tuple (longitude, latitude, bathymetry)

Raises

ValueError – If `oldest_time` is negative (if specified) or if `time_increment` is not positive.

Notes

The output paleo bathymetry values are negative below sea level by default. Note that this is the inverse of water depth (which is positive below sea level).

Any input points outside the masked region of the total sediment thickness grid are ignored (since bathymetry relies on sediment decompaction over time).

New in version 1.4.

Changed in version 1.5: `oldest_time` no longer needs to be specified (defaults to oldest of ocean crust ages and continental rift start ages of input points).

`pybacktrack.`**`write_paleo_bathymetry_grids`**(*paleo_bathymetry*, *grid_spacing_degrees*, *output_file_prefix*, *output_xyz=False*, *use_all_cpus=False*)

Grid paleo bathymetry into a NetCDF grid for each time step.

Parameters

- **`paleo_bathymetry`** (*`dict`*) – A dict mapping each reconstructed time to a list of 3-tuple (longitude, latitude, bathymetry) The reconstructed paleo bathymetry points over a sequence of reconstructed times. Each key in the returned dict is one of those times and each value in the dict is a list of reconstructed paleo bathymetries represented as a 3-tuple containing reconstructed longitude, reconstructed latitude and paleo bathmetry.
- **`grid_spacing_degrees`** (*`float`*) – Lat/lon grid spacing (in degrees). Ideally this should match the spacing of the input points used to generate the paleo bathymetries.
- **`output_file_prefix`** (*`string`*) – The prefix of the output paleo bathymetry grid filenames over time, with "_<time>.nc" appended.
- **`output_xyz`** (*`bool, optional`*) – Whether to also create a GMT xyz file (with ".xyz" extension) for each output paleo bathymetry grid. Each row of each xyz file contains "longitude latitude bathymetry". Default is to only create grid files (no xyz).
- **`use_all_cpus`** (*`bool or int, optional`*) – If `False` (or zero) then use a single CPU. If `True` then distribute CPU processing across all CPUs (cores). If a positive integer then use that many CPUs (cores). Defaults to `False` (single CPU).

Notes

New in version 1.4.

`pybacktrack.`**`reconstruct_paleo_bathymetry_grids`**(*output_file_prefix*, *grid_spacing_degrees*, *oldest_time=None*, *time_increment=1*, *lithology_filenames=[pybacktrack.DEFAULT_BUNDLE_LITHOLOGY_FILEN*
age_grid_filename=pybacktrack.BUNDLE_AGE_GRID_FILENAME,
topography_filename=pybacktrack.BUNDLE_TOPOGRAPHY_FILENAME,
total_sediment_thickness_filename=pybacktrack.BUNDLE_TOTAL_SEDIM
crustal_thickness_filename=pybacktrack.BUNDLE_CRUSTAL_THICKNE
rotation_filenames=pybacktrack.bundle_data.BUNDLE_RECONSTRUCTION
static_polygon_filename=pybacktrack.bundle_data.BUNDLE_RECONST
dynamic_topography_model=None,
sea_level_model=None, *lithology_name=pybacktrack.DEFAULT_PALEO_BATHYMETRY_LITHOLOG*
*ocean_age_to_depth_model=pybacktrack.AGE_TO_DEPTH_DEFAULT_*
exclude_distances_to_trenches_kms=None,
region_plate_ids=None, *anchor_plate_id=0*,
output_positive_bathymetry_below_sea_level=False,
output_xyz=False, *use_all_cpus=False*)

Same as *`pybacktrack.reconstruct_paleo_bathymetry()`* but also generates present day input points on a lat/lon grid and outputs paleobathymetry as a NetCDF grid for each time step.

Parameters

- **`output_file_prefix`** (*`string`*) – The prefix of the output paleo bathymetry grid filenames over time, with "_<time>.nc" appended.

- **grid_spacing_degrees** (*float*) – Spacing between lat/lon points (in degrees) to sample bathymetry at present day. Note that any samples outside the masked region of the total sediment thickness grid are ignored.
- **oldest_time** (*float, optional*) – The oldest time (in Ma) that output is generated back to (from present day). Value must not be negative. If not specified then the oldest of oceanic crustal ages (for those grid points on oceanic crust) and rift start ages (for those grid points on continental crust) is used instead.
- **time_increment** (*float*) – The time increment (in My) that output is generated (from present day back to oldest time). Value must be positive.
- **lithology_filenames** (*list of string, optional*) – One or more text files containing lithologies.
- **age_grid_filename** (*string, optional*) – Age grid filename. Used to obtain age of oceanic crust at present day. Crust is oceanic at locations inside masked age grid region, and continental outside.
- **topography_filename** (*string, optional*) – Topography filename. Used to obtain bathymetry at present day.
- **total_sediment_thickness_filename** (*string, optional*) – Total sediment thickness filename. Used to obtain total sediment thickness at present day.
- **crustal_thickness_filename** (*string, optional*) – Crustal thickness filename. Used to obtain crustal thickness at present day.
- **rotation_filenames** (*list of string, optional*) – List of filenames containing rotation features (to reconstruct sediment-deposited crust). If not specified then defaults to the built-in global rotations associated with the topological model used to generate the built-in rift start/end time grids.
- **static_polygon_filename** (*string, optional*) – Filename containing static polygon features (to assign plate IDs to points on sediment-deposited crust). If not specified then defaults to the built-in static polygons associated with the topological model used to generate the built-in rift start/end time grids.
- **dynamic_topography_model** (*string or tuple, optional*) – Represents a time-dependent dynamic topography raster grid (in *mantle* frame).

  Can be either:

  - A string containing the name of a bundled dynamic topography model.

    Choices include `terra`, `M1`, `M2`, `M3`, `M4`, `M5`, `M6`, `M7`, `ngrand`, `s20rts`, `smean`, `AY18`, `KM16`, `D10_gmcm9` and `gld428`.

  - A tuple containing the three elements (dynamic topography list filename, static polygon filename, rotation filenames).

    The first tuple element is the filename of file containing list of dynamic topography grids (and associated times). Each row in this list file should contain two columns. First column containing filename (relative to list file) of a dynamic topography grid at a particular time. Second column containing associated time (in Ma). The second tuple element is the filename of file containing static polygons associated with dynamic topography model. This is used to assign plate ID to a location so it can be reconstructed. The third tuple element is the filename of the rotation file associated with model. Only the rotation file for static continents/oceans is needed (ie, deformation rotations not needed).

- **sea_level_model** (*string, optional*) – Used to obtain sea levels relative to present day. Can be either the name of a bundled sea level model, or a sea level filename. Bundled sea level models include `Haq87_SealevelCurve` and `Haq87_SealevelCurve_Longterm`.
- **lithology_name** (*string, optional*) – Lithology name of the all sediment (must be present in lithologies file). The total sediment thickness at all sediment locations consists of a single lithology. Defaults to `Average_ocean_floor_sediment`.
- **ocean_age_to_depth_model** (*{pybacktrack.AGE_TO_DEPTH_MODEL_RHCW18, pybacktrack.AGE_TO_DEPTH_MODEL_CROSBY_2007, pybacktrack.AGE_TO_DEPTH_MODEL_GDH1} or function, optional*) – The model to use when converting ocean age to depth at a location (if on ocean floor - not used for continental passive margin). It can be one of the enumerated values, or a callable function accepting a single non-negative age parameter and returning depth (in metres).
- **exclude_distances_to_trenches_kms** (*2-tuple of float, optional*) – The two distances to present-day trenches (on subducting and overriding sides, in that order) to exclude bathymetry grid points (in kms), or None to use built-in per-trench defaults. Default is None.
- **region_plate_ids** (*list of int, optional*) – Plate IDs of one or more plates to restrict paleobathymetry reconstruction to. Defaults to global.
- **anchor_plate_id** (*int, optional*) – The anchor plate id used when reconstructing paleobathymetry grid points. Defaults to zero.
- **output_positive_bathymetry_below_sea_level** (*bool, optional*) – Whether to output positive bathymetry values below sea level (the same as backtracked water depths at a drill site). However topography/bathymetry grids typically have negative values below sea level (and positive above). So the default (`False`) matches typical topography/bathymetry grids (ie, outputs negative bathymetry values below sea level).
- **output_xyz** (*bool, optional*) – Whether to also create a GMT xyz file (with ".xyz" extension) for each output paleo bathymetry grid. Each row of each xyz file contains "longitude latitude bathymetry". Default is to only create grid files (no xyz).
- **use_all_cpus** (*bool or int, optional*) – If `False` (or zero) then use a single CPU. If `True` then distribute CPU processing across all CPUs (cores). If a positive integer then use that many CPUs (cores). Defaults to `False` (single CPU).

Raises

`ValueError` – If `oldest_time` is negative (if specified) or if `time_increment` is not positive.

Notes

The output paleo bathymetry grids have negative values below sea level by default. Note that this is the inverse of water depth (which is positive below sea level).

Any input points outside the masked region of the total sediment thickness grid are ignored (since bathymetry relies on sediment decompaction over time).

New in version 1.4.

Changed in version 1.5: `oldest_time` no longer needs to be specified (defaults to oldest of ocean crust ages and continental rift start ages of grid points).

2.7.4 Creating lithologies

Create lithologies or read them from file(s).

Summary

pybacktrack.Lithology is a class containing data for a lithology.

pybacktrack.read_lithologies_file() reads lithologies from a text file.

pybacktrack.read_lithologies_files() reads and merges lithologies from one or more text files.

pybacktrack.create_lithology() creates a lithology by looking up a name in a dictionary of lithologies.

pybacktrack.create_lithology_from_components() creates a lithology by combining multiple lithologies using different weights.

Detail

class pybacktrack.**Lithology**(*density*, *surface_porosity*, *porosity_decay*)

Class containing lithology data.

__init__(*density*, *surface_porosity*, *porosity_decay*)

Create a lithology from density, surface porosity and porosity decay.

Parameters

- **density** (*float*) – Density (in kg/m3).
- **surface_porosity** (*float*) – Surface porosity (unit-less).
- **porosity_decay** (*float*) – Porosity decay (in metres).

pybacktrack.**read_lithologies_file**(*lithologies_filename*)

Reads a text file with each row representing lithology parameters.

Parameters

lithologies_filename (*str*) – Filename of the lithologies text file.

Returns

Dictionary mapping lithology names to *pybacktrack.Lithology* objects.

Return type

dict

Notes

The four parameter columns in the lithologies text file should contain:

1. name
2. density
3. surface_porosity
4. porosity_decay

`pybacktrack.`**`read_lithologies_files`**(*lithologies_filenames*)

Reads each lithologies text file in the sequence and merges their lithologies.

Parameters
`lithologies_filenames` (*`sequence of str`*) – Filenames of the lithologies text files.

Returns
Dictionary mapping lithology names to *`pybacktrack.Lithology`* objects.

Return type
dict

Notes

The four parameter columns in each lithologies text file should contain:

1. name
2. density
3. surface_porosity
4. porosity_decay

The order of filenames is important. If a lithology name exists in multiple files but has different definitions (values for density, surface porosity and porosity decay) then the definition in the last file containing the lithology name is used.

New in version 1.2.

`pybacktrack.`**`create_lithology`**(*lithology_name*, *lithologies*)

Looks up a lithology using a name.

Parameters

- **`lithology_name`** (*`str`*) – The name of the lithology to look up.
- **`lithologies`** (*`dict`*) – A dictionary mapping lithology names to *`pybacktrack.Lithology`* objects.

Returns
The lithology matching `lithology_name`.

Return type
`pybacktrack.Lithology`

Raises
`KeyError` – If `lithology_name` is not found in `lithologies`.

`pybacktrack.`**`create_lithology_from_components`**(*components*, *lithologies*)

Creates a combined lithology (if necessary) from multiple weighted lithologies.

Parameters

- **`components`** (*`sequence of tuples`*) – A sequence (eg, `list`) of tuples (str, float) containing a lithology name and its fraction of contribution.
- **`lithologies`** (*`dict`*) – A dictionary mapping lithology names to *`pybacktrack.Lithology`* objects.

Returns
The combined lithology.

Return type
pybacktrack.Lithology

Raises

- **ValueError** – If all fractions do not add up to 1.0.
- **KeyError** – If a lithology name is not found in `lithologies`.

2.7.5 Decompacting well sites

- *Read/write well site files*,
- *query a well and its stratigraphic layers*, and
- *query decompacted sections at past times*.

Reading and writing well files

Read/write well site files.

Summary

pybacktrack.read_well_file() reads a text file with each row representing a stratigraphic unit.

pybacktrack.write_well_file() writes a text file with each row representing a stratigraphic unit.

pybacktrack.write_well_metadata() writes well metadata to a text file.

Detail

pybacktrack.**read_well_file**(*well_filename*, *lithologies*, *bottom_age_column=0*, *bottom_depth_column=1*, *lithology_column=2*, *other_columns=None*, *well_attributes=None*)

Reads a text file with each row representing a stratigraphic unit.

Parameters

- **well_filename** (*str*) – Name of well text file.
- **lithologies** (*dict*) – Dictionary mapping lithology names to *pybacktrack.Lithology* objects.
- **well_bottom_age_column** (*int, optional*) – The column of well file containing bottom age. Defaults to 0.
- **well_bottom_depth_column** (*int, optional*) – The column of well file containing bottom depth. Defaults to 1.
- **well_lithology_column** (*int, optional*) – The column of well file containing lithology(s). Defaults to 2.
- **other_columns** (*dict, optional*) – Dictionary of extra columns (besides age, depth and lithology(s)). Each dict value should be a column index (to read from file), and each associated dict key should be a string that will be the name of an attribute (added to each *pybacktrack.StratigraphicUnit* object in the returned *pybacktrack.Well*) containing the value read. For example, backstripping

will add `min_water_depth` and `max_water_depth` attributes (when *pybacktrack.backstrip_well()* or *pybacktrack.backstrip_and_write_well()* has been called, which in turn calls this function).

- **well_attributes** (*dict, optional*) – Attributes to read from well file metadata and store in returned *pybacktrack.Well* object. If specified then must be a dictionary mapping each metadata name to a 2-tuple containing attribute name and a function to convert attribute string to attribute value. For example, {'SiteLongitude' : ('longitude', float), 'SiteLatitude' : ('latitude', float)} will look for metadata name 'SiteLongitude' and store a float value in Well.longitude (or None if not found), etc. Each metadata not found in well file will store None in the associated attribute of *pybacktrack.Well* object.

Returns
Well read from file.

Return type
pybacktrack.Well

Raises
ValueError – If `lithology_column` is not the largest column number (must be last column).

Notes

Each attribute to read (eg, bottom_age, bottom_depth, etc) has a column index to direct which column it should be read from.

If file contains `SurfaceAge = <age>` in commented (#) lines then the top age of the youngest stratigraphic unit will have that age, otherwise it defaults to 0Ma (present day).

pybacktrack.**write_well_file**(*well*, *well_filename*, *other_column_attribute_names=None*, *well_attributes=None*)

Writes a text file with each row representing a stratigraphic unit.

Parameters

- **well** (*pybacktrack.Well*) – The well to write.
- **well_filename** (*str*) – Name of well text file.
- **other_column_attribute_names** (*sequence of str*) – Names of any extra column attributes to write as column before the lithology(s) column. For example, backstripping will add `min_water_depth` and `max_water_depth` attributes (when *pybacktrack.backstrip_well()* or *pybacktrack.backstrip_and_write_well()* has been called, which in turn calls this function).
- **well_attributes** (*dict, optional*) – Attributes in *pybacktrack.Well* object to write to well file metadata. If specified then must be a dictionary mapping each attribute name to a metadata name. For example, {'longitude' : 'SiteLongitude', 'latitude' : 'SiteLatitude'} will write well.longitude (if not None) to metadata 'SiteLongitude', etc. Not that the attributes must exist in `well` (but can be set to None).

pybacktrack.**write_well_metadata**(*well_file*, *well*, *well_attributes=None*)

Writes well metadata to file object `well_file`.

Parameters

- **well_file** (*file object*) – Well file object to write to.
- **well** (*pybacktrack.Well*) – Well to extract metadata from.

- **`well_attributes`** (*`dict, optional`*) – Attributes in *`pybacktrack.Well`* object to write to well file metadata. If specified then must be a dictionary mapping each attribute name to a metadata name. For example, {'longitude' : 'SiteLongitude', 'latitude' : 'SiteLatitude'} will write well.longitude (if not None) to metadata 'SiteLongitude', etc. Not that the attributes must exist in `well` (but can be set to None).

Compacted well

Query a well and its stratigraphic layers.

Summary

`pybacktrack.Well` is a class containing all stratigraphic units in a well.

`pybacktrack.StratigraphicUnit` is a class containing data for a stratigraphic unit.

Detail

class `pybacktrack.`**`Well`**(*attributes=None*, *stratigraphic_units=None*)

Class containing all the stratigraphic units in a well sorted by age (from youngest to oldest).

`longitude`

Longitude of the well location.

Note: This attribute is available provided *`pybacktrack.backtrack_well()`* or *`pybacktrack.backstrip_well()`* (or any function calling them) have been called.

Type
float, optional

`latitude`

Latitude of the well location.

Note: This attribute is available provided *`pybacktrack.backtrack_well()`* or *`pybacktrack.backstrip_well()`* (or any function calling them) have been called.

Type
float, optional

`stratigraphic_units`

List of stratigraphic units in this well sorted by age (from youngest to oldest).

Type
list of *`pybacktrack.StratigraphicUnit`*

`__init__`(*attributes=None*, *stratigraphic_units=None*)

Create a well from optional stratigraphic units.

Parameters

- **attributes** (*dict, optional*) – Attributes to store on this well object. If specified then must be a dictionary mapping attribute names to values.
- **stratigraphic_units** (sequence of *pybacktrack.StratigraphicUnit*, optional) – Sequence of StratigraphicUnit objects. They can be unsorted (by age) but will be added in sorted order.

Raises

`ValueError` – If:

#. Youngest unit does not have zero depth, or #. adjacent units do not have matching top and bottom ages and depths.

. . . this ensures the units are contiguous in depth from the surface (ie, no gaps).

Notes

Stratigraphic units can also be added using *pybacktrack.Well.add_compacted_unit()*

add_compacted_unit(*top_age*, *bottom_age*, *top_depth*, *bottom_depth*, *lithology_components*, *lithologies*, *other_attributes=None*)

Add the next deeper stratigraphic unit.

Units must be added in order of age.

Parameters

- **top_age** (*float*) – Age of top of stratigraphic unit (in Ma).
- **bottom_age** (*float*) – Age of bottom of stratigraphic unit (in Ma).
- **top_depth** (*float*) – Depth of top of stratigraphic unit (in metres).
- **bottom_depth** (*float*) – Depth of bottom of stratigraphic unit (in metres).
- **lithology_components** (*sequence of tuples (str, float)*) – Sequence of tuples (name, fraction) containing a lithology name and its fraction of contribution.
- **lithologies** (*dict*) – A dictionary mapping lithology names to *pybacktrack.Lithology* objects.
- **other_attributes** (*dict, optional*) – A dictionary of attribute name/value pairs to set on stratigraphic unit object (using `setattr`). For example, backstripping will add the `min_water_depth` and `max_water_depth` attributes (when *pybacktrack.backstrip_well()* or *pybacktrack.backstrip_and_write_well()* has been called).

Raises

`ValueError` – If:

#. Youngest unit does not have zero depth, or #. adjacent units do not have matching top and bottom ages and depths.

. . . this ensures the units are contiguous in depth from the surface (ie, no gaps).

decompact(*age=None*)

Finds decompacted total sediment thickness at 'age' (if specified), otherwise at each (top) age in all stratigraphic units.

Returns

If 'age' is specified then returns the decompacted well at that age (or None if 'age' is not younger than bottom age of well), otherwise a list of decompacted wells with one per age in same order (and ages) as the well units (youngest to oldest).

Return type
`pybacktrack.DecompactedWell`, or list of `pybacktrack.DecompactedWell`

Notes

Changed in version 1.4: Added the 'age' parameter.

class `pybacktrack.`**`StratigraphicUnit`**(*top_age*, *bottom_age*, *top_depth*, *bottom_depth*, *lithology_components*, *lithologies*, *other_attributes=None*)

Class to hold data for a stratigraphic unit.

`top_age`

Age of top of stratigraphic unit (in Ma).

Type
float

`bottom_age`

Age of bottom of stratigraphic unit (in Ma).

Type
float

`top_depth`

Depth of top of stratigraphic unit (in metres).

Type
float

`bottom_depth`

Depth of bottom of stratigraphic unit (in metres).

Type
float

`decompacted_top_depth`

Fully decompacted depth of top of stratigraphic unit (in metres) as if no portion of any layer had ever been buried (ie, using surface porosities only).

Type
float

`decompacted_bottom_depth`

Fully decompacted depth of bottom of stratigraphic unit (in metres) as if no portion of any layer had ever been buried (ie, using surface porosities only).

Type
float

`min_water_depth`

Minimum paleo-water depth of stratigraphic unit (in metres).

Note: This attribute is only available when backstripping (not backtracking). For example, it is available if `pybacktrack.backstrip_well()` or `pybacktrack.backstrip_and_write_well()` has been called.

Type
float, optional

`max_water_depth`

Maximum paleo-water depth of stratigraphic unit (in metres).

Note: This attribute is only available when backstripping (not backtracking). For example, it is available if *`pybacktrack.backstrip_well()`* or *`pybacktrack.backstrip_and_write_well()`* has been called.

Type
float, optional

`lithology_components`

Sequence of tuples (name, fraction) containing a lithology name and its fraction of contribution.

Type
sequence of tuples (str, float)

`__init__`(*top_age*, *bottom_age*, *top_depth*, *bottom_depth*, *lithology_components*, *lithologies*, *other_attributes=None*)

Create a stratigraphic unit from top and bottom age, top and bottom depth and lithology components.

Parameters

- **`top_age`** (*`float`*) – Age of top of stratigraphic unit (in Ma).
- **`bottom_age`** (*`float`*) – Age of bottom of stratigraphic unit (in Ma).
- **`top_depth`** (*`float`*) – Depth of top of stratigraphic unit (in metres).
- **`bottom_depth`** (*`float`*) – Depth of bottom of stratigraphic unit (in metres).
- **`lithology_components`** (*`sequence of tuples (str, float)`*) – Sequence of tuples (name, fraction) containing a lithology name and its fraction of contribution.
- **`lithologies`** (*`dict`*) – A dictionary mapping lithology names to *`pybacktrack.Lithology`* objects.
- **`other_attributes`** (*`dict, optional`*) – A dictionary of attribute name/value pairs to set on stratigraphic unit object (using `setattr`). For example, backstripping will add the `min_water_depth` and `max_water_depth` attributes (when *`pybacktrack.backstrip_well()`* or *`pybacktrack.backstrip_and_write_well()`* has been called).

`calc_decompacted_density`(*decompacted_thickness*, *decompacted_depth_to_top*)

Calculate average decompacted density when top of this stratigraphic unit is at a decompacted depth.

Parameters

- **`decompacted_thickness`** (*`float`*) – Decompacted thickness of this stratigraphic unit as returned by *`pybacktrack.StratigraphicUnit.calc_decompacted_thickness()`*.
- **`decompacted_depth_to_top`** (*`float`*) – Decompacted depth of the top of this stratigraphic unit.

Returns
Decompacted density.

Return type
float

`calc_decompacted_thickness`(*decompacted_depth_to_top*)

Calculate decompacted thickness when top of this stratigraphic unit is at a decompacted depth.

Parameters
`decompacted_depth_to_top` (*float*) – Decompacted depth of the top of this stratigraphic unit.

Returns
Decompacted thickness.

Return type
float

`static create_partial_unit`(*unit*, *top_age*)

Create a new stratigraphic unit equivalent to 'unit' but with the top part stripped off according to 'top_age'.

Essentially sediment deposited from 'top_age' to the top age of 'unit' is stripped off (assuming a constant sediment deposition rate for 'unit'). And so 'top_age' is expected to be older/earlier than the top age of 'unit' (and younger than the bottom age of 'unit').

Parameters
`top_age` (*float*) – Top age of new stratigraphic unit.

Raises
`ValueError` – If 'top_age' is outside the top/bottom age range of 'unit'.

Notes

This does *not* partially decompact 'unit'. It is simply adjusting the top depth of new unit to correspond to its new top age. Then when the returned partial stratigraphic unit is subsequently decompacted it'll have the correct volume of grains (assuming a constant sediment deposition rate) and hence be decompacted correctly at its new top age.

New in version 1.4.

`get_decompacted_sediment_rate`()

Return fully decompacted sediment rate.

This is the fully decompacted thickness of this unit divided by its (deposition) time interval.

Returns
Decompacted sediment rate (in units of metres/Ma).

Return type
float

Notes

Fully decompacted is equivalent to assuming this unit is at the surface (ie, no units on top of it) and porosity decay within the unit is not considered (in other words the weight of the upper part of the unit does not compact the lower part of the unit).

`get_fully_decompacted_thickness()`

Get fully decompacted thickness. It is calculated on first call.

Returns
Fully decompacted thickness.

Return type
float

Notes

Fully decompacted is equivalent to assuming this unit is at the surface (ie, no units on top of it) and porosity decay within the unit is not considered (in other words the weight of the upper part of the unit does not compact the lower part of the unit).

New in version 1.4.

Decompacted well

Query decompacted sections at past times.

Summary

`pybacktrack.DecompactedWell` is a class containing the decompacted well data at a specific age.

`pybacktrack.DecompactedStratigraphicUnit` is a class to hold data for a *decompacted* stratigraphic unit.

Detail

`class pybacktrack.DecompactedWell(`*surface_unit*`)`

Class containing the decompacted well data at a specific age.

`surface_unit`

Top stratigraphic unit in this decompacted well.

Type
`pybacktrack.StratigraphicUnit`

`total_compacted_thickness`

Total compacted thickness of all stratigraphic units.

Type
float

`total_decompacted_thickness`

Total decompacted thickness of all decompacted stratigraphic units.

Type
float

`tectonic_subsidence`

Tectonic subsidence (in metres).

Note: This attribute is only available when backtracking (not backstripping). For example, it is available if *pybacktrack.backtrack_well()* or *pybacktrack.backtrack_and_write_well()* has been called.

Type
float, optional

`min_water_depth`

Minimum water depth (in metres).

Note: This attribute is only available when backstripping (not backtracking). For example, it is available if *pybacktrack.backstrip_well()* or *pybacktrack.backstrip_and_write_well()* has been called.

New in version 1.2.

Type
float, optional

`max_water_depth`

Maximum water depth (in metres).

Note: This attribute is only available when backstripping (not backtracking). For example, it is available if *pybacktrack.backstrip_well()* or *pybacktrack.backstrip_and_write_well()* has been called.

New in version 1.2.

Type
float, optional

`sea_level`

Sea level in metres (positive for a sea-level rise and negative for a sea-level fall, relative to present day).

Note: This attribute is only available if a sea model was specified when backtracking or backstripping (for example, if `sea_level_model` was specified in *pybacktrack.backtrack_well()* or *pybacktrack.backstrip_well()*).

See also:

pybacktrack.DecompactedWell.get_sea_level()

Type
float, optional

`dynamic_topography`

Dynamic topography elevation *relative to present day* (in metres).

Note: This attribute contains dynamic topography **relative to present day**.

Note: This attribute is only available when backtracking (not backstripping) **and** if a dynamic topography model was specified. For example, it is available if `dynamic_topography_model` was specified in *`pybacktrack.backtrack_well()`* or *`pybacktrack.backtrack_and_write_well()`*

Note: Dynamic topography is elevation which is opposite to tectonic subsidence in that an increase in dynamic topography results in a decrease in tectonic subsidence.

See also:

`pybacktrack.DecompactedWell.get_dynamic_topography()`

New in version 1.2.

Type
float, optional

`decompacted_stratigraphic_units`

Decompacted stratigraphic units. They are sorted from top to bottom (in depth) which is the same as youngest to oldest.

Type
list of *`pybacktrack.DecompactedStratigraphicUnit`*

`__init__`(*surface_unit*)

Create a decompacted well whose top stratigraphic unit is `surface_unit`.

Parameters
`surface_unit` (*`pybacktrack.StratigraphicUnit`*) – Top stratigraphic unit in this decompacted well.

Notes

You still need to add the decompacted units with *`pybacktrack.DecompactedWell.add_decompacted_unit()`*.

See also:

`pybacktrack.Well.decompact()`

`add_decompacted_unit`(*stratigraphic_unit*, *decompacted_thickness*, *decompacted_density*)

Add a decompacted stratigraphic unit.

Parameters

- **`stratigraphic_unit`** (*`pybacktrack.StratigraphicUnit`*) – Stratigraphic unit referenced by decompacted stratigraphic unit.
- **`decompacted_thickness`** (*`float`*) – Decompacted thickness.
- **`decompacted_density`** (*`float`*) – Decompacted density.

Notes

Stratigraphic units should be decompacted from top of well column to bottom.

`get_age()`

Returns
Age of the surface of the decompacted column of the well.

Return type
float

`get_average_decompacted_density()`

Returns
Average density of the entire decompacted column of the well.

Return type
float

`get_dynamic_topography`(*default_dynamic_topography=0.0*)

Returns the dynamic topography elevation *relative to present day*, or `default_dynamic_topography` if a dynamic topography model was not specified (when backtracking).

Returns
Dynamic topography elevation *relative to present day*.

Return type
float

Notes

Note: Returns dynamic topography **relative to present day**.

Returns the `dynamic_topography` attribute if a `dynamic_topography_model` was specified to *`pybacktrack.backtrack_well()`* or *`pybacktrack.backtrack_and_write_well()`*, otherwise returns `default_dynamic_topography`.

Note: Dynamic topography is elevation which is opposite to tectonic subsidence in that an increase in dynamic topography results in a decrease in tectonic subsidence.

Note: `default_dynamic_topography` can be set to `None`

New in version 1.2.

`get_min_max_tectonic_subsidence()`

Returns the minimum and maximum tectonic subsidence obtained directly from subsidence model (if backtracking) or indirectly from minimum and maximum water depth and sea level (if backstripping).

Returns

- **min_tectonic_subsidence** (*float*) – Minimum tectonic subsidence (unloaded water depth) of this decompacted well.

- **max_tectonic_subsidence** (*float*) – Maximum tectonic subsidence (unloaded water depth) of this decompacted well.

Notes

When backtracking, the tectonic subsidence is obtained directly from the `tectonic_subsidence` attribute. In this case the minimum and maximum tectonic subsidence are the same.

When backstripping, the tectonic subsidence is obtained indirectly from the `min_water_depth` and `max_water_depth` attributes and optional `sea_level` attribute (if a sea level model was specified).

New in version 1.2.

`get_min_max_tectonic_subsidence_from_water_depth`(*min_water_depth*, *max_water_depth*, *sea_level=None*)

Returns the minimum and maximum tectonic subsidence obtained from specified minimum and maximum water depths (and optional sea level).

Parameters

- **`min_water_depth`** (*`float`*) – Minimum water depth.
- **`max_water_depth`** (*`float`*) – Maximum water depth.
- **`sea_level`** (*`float, optional`*) – Sea level relative to present day (positive to sea-level rise and negative for sea-level fall).

Returns

- **min_tectonic_subsidence** (*float*) – Minimum tectonic subsidence (unloaded water depth) of this decompacted well from its minimum water depth.
- **max_tectonic_subsidence** (*float*) – Maximum tectonic subsidence (unloaded water depth) of this decompacted well from its maximum water depth.

Notes

Optional sea level fluctuation is included if specified.

`get_min_max_water_depth`()

Returns the minimum and maximum water depth obtained directly from minimum and maximum water depth (if backstripping) or indirectly from tectonic subsidence model and sea level (if backtracking).

Returns

- **min_water_depth** (*float*) – Minimum water depth of this decompacted well.
- **max_water_depth** (*float*) – Maximum water depth of this decompacted well.

Notes

When backstripping, the minimum and maximum water depths are obtained directly from the `min_water_depth` and `max_water_depth` attributes.

When backtracking, the water depth is obtained indirectly from the `tectonic_subsidence` attribute and optional `sea_level` attribute (if a sea level model was specified). In this case the minimum and maximum water depths are the same.

New in version 1.2.

get_sea_level(*default_sea_level=0.0*)

Returns the sea level relative to present day, or `default_sea_level` if a sea level model was not specified (when either backtracking or backstripping).

Returns
Sea level relative to present day (positive to sea-level rise and negative for sea-level fall).

Return type
float

Notes

Returns the `sea_level` attribute if a `sea_level_model` was specified to *pybacktrack.backtrack_well()* or *pybacktrack.backstrip_well()*, otherwise returns `default_sea_level`.

Note: `default_sea_level` can be set to `None`

New in version 1.2.

get_sediment_isostatic_correction()

Returns
Isostatic correction of this decompacted well.

Return type
float

Notes

The returned correction can be added to a known water depth to obtain the deeper isostatically compensated, sediment-free water depth (tectonic subsidence). Or the correction could be subtracted from a known tectonic subsidence (unloaded water depth) to get the depth at sediment/water interface.

get_tectonic_subsidence()

Returns the tectonic subsidence obtained directly from subsidence model (if backtracking) or indirectly from average of minimum and maximum water depth and sea level (if backstripping).

Returns
Tectonic subsidence (unloaded water depth) of this decompacted well.

Return type
float

Notes

When backtracking, the tectonic subsidence is obtained directly from the `tectonic_subsidence` attribute.

When backstripping, the tectonic subsidence is obtained indirectly from the `min_water_depth` and `max_water_depth` attributes and optional `sea_level` attribute (if a sea level model was specified).

New in version 1.2.

`get_water_depth()`

Returns the water depth obtained directly from average of minimum and maximum water depth (if backstripping) or indirectly from tectonic subsidence model and sea level (if backtracking).

Returns
Water depth of this decompacted well.

Return type
float

Notes

When backstripping, the water depth is obtained directly as an average of the `min_water_depth` and `max_water_depth` attributes.

When backtracking, the water depth is obtained indirectly from the `tectonic_subsidence` attribute and optional `sea_level` attribute (if a sea level model was specified).

New in version 1.2.

`get_water_depth_from_tectonic_subsidence`(*tectonic_subsidence*, *sea_level=None*)

Returns the water depth of this decompacted well from the specified tectonic subsidence (and optional sea level).

Parameters

- **`tectonic_subsidence`** (*`float`*) – Tectonic subsidence.
- **`sea_level`** (*`float, optional`*) – Sea level relative to present day (positive to sea-level rise and negative for sea-level fall).

Returns
Water depth of this decompacted well from its tectonic subsidence (unloaded water depth).

Return type
float

Notes

Optional sea level fluctuation (relative to present day) is included if specified.

`class pybacktrack.`**`DecompactedStratigraphicUnit`**(*stratigraphic_unit*, *decompacted_thickness*, *decompacted_density*)

Class to hold data for a *decompacted* stratigraphic unit (decompacted at a specific age).

`stratigraphic_unit`

Stratigraphic unit referenced by this decompacted stratigraphic unit.

Type
`pybacktrack.StratigraphicUnit`

decompacted_thickness

Decompacted thickness.

Type
float

decompacted_density

Decompacted density.

Type
float

__init__(*stratigraphic_unit*, *decompacted_thickness*, *decompacted_density*)

Create a decompacted stratigraphic unit from a stratigraphic unit, decompacted thickness and decompacted density.

Parameters

- **stratigraphic_unit** (*pybacktrack.StratigraphicUnit*) – Stratigraphic unit referenced by this decompacted stratigraphic unit.
- **decompacted_thickness** (*float*) – Decompacted thickness.
- **decompacted_density** (*float*) – Decompacted density.

2.7.6 Converting oceanic age to depth

Convert ocean basin ages (Ma) to basement depth (metres) using different age/depth models.

Summary

pybacktrack.convert_age_to_depth() converts a single ocean basin age to basement depth.

pybacktrack.convert_age_to_depth_files() converts a sequence of ages (read from an input file) to depths (and writes both ages and depths to an output file).

Detail

pybacktrack.**convert_age_to_depth**(*age*, *model=pybacktrack.AGE_TO_DEPTH_DEFAULT_MODEL*)

Convert ocean basin age to basement depth using a specified age/depth model.

Parameters

- **age** (*float*) – The age in Ma.
- **model** (*{pybacktrack.AGE_TO_DEPTH_MODEL_RHCW18, pybacktrack.AGE_TO_DEPTH_MODEL_CROSBY_2007, pybacktrack.AGE_TO_DEPTH_MODEL_GDH1} or function, optional*) – The model to use when converting ocean age to basement depth. It can be one of the enumerated values, or a callable function accepting a single non-negative age parameter and returning depth (in metres).

Returns
Depth (in metres) as a positive number.

Return type
float

Raises

- `ValueError` – If *age* is negative.
- `TypeError` – If *model* is not a recognised model, or a function accepting a single parameter.

`pybacktrack.`**`convert_age_to_depth_files`**(*input_filename*, *output_filename*, *model=pybacktrack.AGE_TO_DEPTH_DEFAULT_MODEL*, *age_column_index=0*, *reverse_output_columns=False*)

Converts age to depth by reading *age* rows from input file and writing rows containing both *age* and *depth* to output file.

Parameters

- **`input_filename`** (*`string`*) – Name of input text file containing the *age* values. A single *age* value is obtained from each row by indexing the *age_column_index* column (zero-based index).
- **`output_filename`** (*`string`*) – Name of output text file containing *age* and *depth* values. Each row of output file contains an *age* value and its associated *depth* value (with order depending on *reverse_output_columns*).
- **`model`** (*`{pybacktrack.AGE_TO_DEPTH_MODEL_RHCW18, pybacktrack.AGE_TO_DEPTH_MODEL_CROSBY_2007, pybacktrack.AGE_TO_DEPTH_MODEL_GDH1} or function, optional`*) – The model to use when converting ocean age to basement depth. It can be one of the enumerated values, or a callable function accepting a single non-negative age parameter and returning depth (in metres).
- **`age_column_index`** (*`int, optional`*) – Determines which column of input file to read *age* values from.
- **`reverse_output_columns`** (*`bool, optional`*) – Determines order of *age* and *depth* columns in output file. If *True* then output *depth age*, otherwise output *age depth*.

Raises

`ValueError` – If cannot read *age* value, as a floating-point number, from input file at column index *age_column_index*.

2.7.7 Continental rifting

Continental passive margin initial rifting subsidence and subsequent thermal subsidence. Rifting is assumed instantaneous in that thermal contraction only happens after rifting has ended.

Summary

`pybacktrack.estimate_rift_beta()` estimates the stretching factor (beta).

`pybacktrack.total_rift_subsidence()` calcultaes the total subsidence as syn-rift plus post-rift.

`pybacktrack.syn_rift_subsidence()` calculates the initial subsidence due to continental stretching.

`pybacktrack.post_rift_subsidence()` calculates the thermal subsidence as a function of time.

Detail

`pybacktrack.`**`estimate_rift_beta`**(*present_day_subsidence*, *present_day_crustal_thickness*, *rift_end_time*)

Estimate the stretching factor (beta).

Parameters

- **`present_day_subsidence`** (*`float`*) – The (sediment-free) subsidence at present day (in metres).
- **`present_day_crustal_thickness`** (*`float`*) – The crustal thickness at present day (in metres).
- **`rift_end_time`** (*`float`*) – The time that rifting ended (in My).

Returns

- **beta** (*float*) – The estimated stretching factor.
- **residual** (*float*) – The inaccuracy between present day subsidence and subsidence calculated using the estimated stretching factor (beta).

Notes

Stretching factor (beta) is calculated by minimizing difference between actual subsidence and subsidence calculated from beta (both at present day).

`pybacktrack.`**`total_rift_subsidence`**(*beta*, *pre_rift_crustal_thickness*, *time*, *rift_end_time*, *rift_start_time=None*)

Total subsidence as syn-rift plus post-rift.

Parameters

- **`beta`** (*`float`*) – Stretching factor.
- **`pre_rift_crustal_thickness`** (*`float`*) – Initial crustal thickness prior to rifting (in metres).
- **`time`** (*`float`*) – Time to calculate subsidence (in My).
- **`rift_end_time`** (*`float`*) – Time at which rifting ended (in My).
- **`rift_start_time`** (*`float, optional`*) – Time at which rifting started (in My). If not specified then assumes initial (non-thermal) subsidence happens instantaneously at `rift_end_time`. Defaults to `rift_end_time`.

Returns

Total subsidence (in metres).

Return type

float

`pybacktrack.`**`syn_rift_subsidence`**(*beta*, *pre_rift_crustal_thickness*)

Initial subsidence (in metres) due to continental stretching.

Parameters

- **`beta`** (*`float`*) – Stretching factor.
- **`pre_rift_crustal_thickness`** (*`float`*) – Initial crustal thickness prior to rifting (in metres).

Returns
Initial subsidence (in metres) due to continental stretching.

Return type
float

`pybacktrack.`**`post_rift_subsidence`**(*beta*, *time*)

Thermal subsidence (in metres) as a function of time.

Parameters

- **`beta`** (*`float`*) – Stretching factor.
- **`time`** (*`float`*) – The amount of time that has passed after rifting/stretching has ended.

Returns
Thermal subsidence (in metres).

Return type
float

2.7.8 Dynamic topography

Summary

`pybacktrack.DynamicTopography` is a class that reconstructs point location(s) and samples (and interpolates) time-dependent dynamic topography *mantle* frame grids.

`pybacktrack.InterpolateDynamicTopography` is a class that just samples and interpolates time-dependent dynamic topography *mantle* frame grid files.

Detail

`class pybacktrack.`**`DynamicTopography`**(*grid_list_filename*, *static_polygon_filename*, *rotation_filenames*, *longitude*, *latitude*, *age=None*)

Class that reconstructs point location(s) and samples (and interpolates) time-dependent dynamic topography *mantle* frame grid files.

`longitude`

Longitude of the point location, or list of longitudes (if multiple point locations).

Type
float or list of float

`latitude`

Latitude of the point location, or list of latitudes (if multiple point locations).

Type
float or list of float

`age`

The age of the crust that the point location is on, or list of ages (if multiple point locations).

Note: If no age(s) was supplied then the age(s) of the static polygon(s) containing location(s) is used (or zero when no polygon contains a location).

Type
float or list of float

Notes

Changed in version 1.4: Can have multiple point locations (version 1.3 allowed only one location). So `longitude`, `latitude` and `age` can all have either a single value or multiple values (same number for each).

`__init__`(*grid_list_filename*, *static_polygon_filename*, *rotation_filenames*, *longitude*, *latitude*, *age=None*)

Load dynamic topography grid filenames and associated ages from grid list file 'grid_list_filename'.

Parameters

- **`grid_list_filename`** (*`str`*) – The filename of the grid list file.
- **`static_polygon_filename`** (*`str`*) – The filename of the static polygons file.
- **`rotation_filenames`** (*`list of str`*) – The list of rotation filenames.
- **`longitude`** (*`float or list of float`*) – Longitude of the point location, or list of longitudes (if multiple point locations).
- **`latitude`** (*`float or list of float`*) – Latitude of the point location, or list of latitudes (if multiple point locations).
- **`age`** (*`float or list of float, optional`*) – The age of the crust that the point location is on, or list of ages (if multiple point locations). If not specified then the appearance age(s) of the static polygon(s) containing the point(s) is used.

Raises

- **`ValueError`** – If any `age` is negative (if specified).
- **`ValueError`** – If `longitude` and `latitude` (and `age` if specified) are all not a single value or all not a sequence (of same length).
- **`ValueError`** – If `grid_list_filename` does not contain a grid at present day, or `grid_list_filename` contains fewer than two grids, or not all rows in `grid_list_filename` contain a grid filename followed by an age, or there are two ages in `grid_list_filename` with same age.

Notes

Each dynamic topography grid should be in the *mantle* reference frame (not *plate* reference frame) and should have global coverage (such that no sample location will return NaN).

Each row in the grid list file should contain two columns. First column containing filename (relative to directory of list file) of a dynamic topography grid at a particular time. Second column containing associated time (in Ma).

Each present day location is also assigned a plate ID using the static polygons, and the rotations are used to reconstruct each location when sampling the grids at a reconstructed time.

Changed in version 1.4: The following changes were made:

- Added ability to specify a list of point locations (as an alternative to specifying a single location).
- Raises `ValueError` if there's no present day grid or if any age is negative.

static create_from_bundled_model(*dynamic_topography_model_name*, *longitude*, *latitude*, *age=None*)

Create a DynamicTopography instance from a bundled dynamic topography model name.

Parameters

- **dynamic_topography_model_name** (`str`) – Name of a bundled dynamic topography model. Choices include `terra`, `M1`, `M2`, `M3`, `M4`, `M5`, `M6`, `M7`, `ngrand`, `s20rts`, `smean`, `AY18`, `KM16`, `D10_gmcm9` and `gld428`.
- **longitude** (`float or list of float`) – Longitude of the point location, or list of longitudes (if multiple point locations).
- **latitude** (`float or list of float`) – Latitude of the point location, or list of latitudes (if multiple point locations).
- **age** (`float or list of float, optional`) – The age of the crust that the point location is on, or list of ages (if multiple point locations). If not specified then the appearance age(s) of the static polygon(s) containing the point(s) is used.

Returns

The bundled dynamic topography model.

Return type

`pybacktrack.DynamicTopography`

Raises

`ValueError` – If `dynamic_topography_model_name` is not the name of a bundled dynamic topography model.

Notes

New in version 1.2.

Changed in version 1.4: Added ability to specify a list of point locations (as an alternative to specifying a single location).

static create_from_model_or_bundled_model_name(*dynamic_topography_model_or_bundled_model_name*, *longitude*, *latitude*, *age=None*)

Create a DynamicTopography instance from a user-provided model or from a bundled model.

Parameters

- **dynamic_topography_model_or_bundled_model_name** (`str or 3-tuple (str, str, list of str)`) – Either the name of a bundled dynamic topography model (see `pybacktrack.DynamicTopography.create_from_bundled_model()`), or a user-provided model specified as a 3-tuple (filename of the grid list file, filename of the static polygons file, list of rotation filenames) (see first three parameters of `pybacktrack.DynamicTopography.__init__()`).
- **longitude** (`float or list of float`) – Longitude of the point location, or list of longitudes (if multiple point locations).
- **latitude** (`float or list of float`) – Latitude of the point location, or list of latitudes (if multiple point locations).
- **age** (`float or list of float, optional`) – The age of the crust that the point location is on, or list of ages (if multiple point locations). If not specified then the appearance age(s) of the static polygon(s) containing the point(s) is used.

Returns

The dynamic topography model loaded from a user-provided model or from a bundled model.

Return type
pybacktrack.DynamicTopography

Notes

New in version 1.4.

static **get_bundled_model**(*dynamic_topography_model_name*)

Get the bundled model files for the specified dynamic topography model name.

Parameters
dynamic_topography_model_name (*str*) – Name of a bundled dynamic topography model. Choices include `terra`, `M1`, `M2`, `M3`, `M4`, `M5`, `M6`, `M7`, `ngrand`, `s20rts`, `smean`, `AY18`, `KM16`, `D10_gmcm9` and `gld428`.

Returns
The bundled dynamic topography model files (see first three parameters of *pybacktrack.DynamicTopography.__init__()*). This consists of a 3-tuple of:

- Filename of the grid list file.
- Filename of the static polygons file.
- List of rotation filenames.

Return type
3-tuple of (grid_list_filename, static_polygon_filename, rotation_filenames)

Raises
ValueError – If `dynamic_topography_model_name` is not the name of a bundled dynamic topography model.

Notes

The returned model information is obtained from `pybacktrack.BUNDLE_DYNAMIC_TOPOGRAPHY_MODELS` (see *Bundle data*).

New in version 1.5.

sample(*time*, *fallback=True*)

Samples and interpolates the two time-dependent dynamic topography grids surrounding `time` at point location(s) reconstructed to `time`, but optionally falls back to a non-optimal sampling if necessary (depending on `time`)

Parameters

- **time** (*float*) – Time to sample dynamic topography.
- **fallback** (*bool*) – Whether to fall back to a non-optimal sampling if neccessary (see notes below). Defaults to `True`.

Returns
The sampled dynamic topography value or list of values. If constructed with a single location then returns a single value, otherwise returns a list of values (one per location).

When `fallback` is `True` then `float('NaN`)` will never be returned (see notes below). When `fallback` is `False` then `float('NaN`)` will be returned:

- for all points when the oldest dynamic topography grid is younger than `time`, or

- for each point location whose age is younger than `time` (ie, has not yet appeared).

Return type
float or list of float

Notes

Each point location is first reconstructed to `time` before sampling the two grids surrounding `time` at the reconstructed location and interpolating between them.

For each point location, if `time` is older than its appearance age then it is still reconstructed to `time` when `fallback` is `True`, otherwise `float('NaN`)` is returned (for that location) when `fallback` is `False`.

If `time` is older than the oldest grid then the oldest grid is sampled when `fallback` is `True`, otherwise `float('NaN`)` is returned for all locations when `fallback` is `False`.

Changed in version 1.2: Previously this method was called *sample_interpolated* and did not fall back to non-optimal sampling when necessary.

Changed in version 1.4: The following changes were made:

- Merged *sample*, *sample_interpolated* and *sample_oldest* methods into one method (this method).
- Added *fallback* parameter (where `False` behaves like removed *sample_interpolated* method).
- Added ability to specify a list of point locations (as an alternative to specifying a single location).
- Changed how grids are interpolated:
  - Version 1.3 (and earlier) reconstructed each location to two times (of the two grids surrounding `time`) to get *two* reconstructed locations. Then each reconstructed location sampled its respective grid (ie, each grid was sampled at a *different* reconstructed location). Then these two samples were interpolated (based on `time`).
  - Version 1.4 reconstructs each location to the single `time` to get a *single* reconstructed location. Then that single reconstructed location samples both grids surrounding `time` (ie, each grid is sampled at the *same* reconstructed location). Then these two samples are interpolated (based on `time`).

…note that there is no difference *at* grid times (only between grid times).

class pybacktrack.**InterpolateDynamicTopography**(*grid_list_filename*)

Class that just samples and interpolates time-dependent dynamic topography *mantle* frame grid files.

This class accepts locations that have already been reconstructed whereas *pybacktrack.DynamicTopography* accepts present day locations and reconstructs them prior to sampling the dynamic topography grids.

Notes

New in version 1.4.

__init__(*grid_list_filename*)

Load dynamic topography grid filenames and associated ages from grid list file 'grid_list_filename'.

Parameters
grid_list_filename (*str*) – The filename of the grid list file.

Raises
ValueError – If `grid_list_filename` does not contain a grid at present day, or `grid_list_filename` contains fewer than two grids, or not all rows in

`grid_list_filename` contain a grid filename followed by an age, or there are two ages in `grid_list_filename` with same age.

Notes

Each dynamic topography grid should be in the *mantle* reference frame (not *plate* reference frame) and should have global coverage (such that no sample location will return NaN).

Each row in the grid list file should contain two columns. First column containing filename (relative to directory of list file) of a dynamic topography grid at a particular time. Second column containing associated time (in Ma).

New in version 1.4.

static create_from_bundled_model(*dynamic_topography_model_name*)

Create a InterpolateDynamicTopography instance from a bundled dynamic topography model name.

Parameters
dynamic_topography_model_name (*str*) – Name of a bundled dynamic topography model. Choices include `terra`, `M1`, `M2`, `M3`, `M4`, `M5`, `M6`, `M7`, `ngrand`, `s20rts`, `smean`, `AY18`, `KM16`, `D10_gmcm9` and `gld428`.

Returns
The bundled dynamic topography model.

Return type
pybacktrack.InterpolateDynamicTopography

Raises
ValueError – If `dynamic_topography_model_name` is not the name of a bundled dynamic topography model.

Notes

New in version 1.4.

static create_from_model_or_bundled_model_name(*dynamic_topography_model_or_bundled_model_name*)

Create a InterpolateDynamicTopography instance from a user-provided model or from a bundled model.

Parameters
dynamic_topography_model_or_bundled_model_name (*str*) – Either the name of a bundled dynamic topography model (see *pybacktrack.InterpolateDynamicTopography.create_from_bundled_model()*), or a user-provided model specified as the filename of the grid list file (see parameter of *pybacktrack.InterpolateDynamicTopography.__init__()*).

Raises
ValueError – If `dynamic_topography_model_or_bundled_model_name` is not the name of a bundled dynamic topography model or the filename of an existing grid list file.

Returns
The dynamic topography model loaded from a user-provided model or from a bundled model.

Return type
pybacktrack.InterpolateDynamicTopography

Notes

New in version 1.4.

`sample`(*time*, *locations*, *fallback_to_oldest=True*)

Samples and interpolates the two time-dependent dynamic topography grids surrounding `time` at the specified point location(s), but optionally falls back to sampling oldest grid (if `time` is too old).

Parameters

- **`time`** (*`float`*) – Time to sample dynamic topography.
- **`locations`** (*`sequence of 2-tuple (float, float)`*) – A sequence of (longitude, latitude) point locations.
- **`fallback_to_oldest`** (*`bool`*) – Whether to fall back to sampling oldest grid (if `time` is too old) rather than interpolating the two grids surrounding `time`. Defaults to `True`.

Returns

The sampled dynamic topography values (one per location).

When `time` is older than the oldest dynamic topography grid:

- if `fallback_to_oldest` is `True` then the oldest dynamic topography grid is sampled, or
- if `fallback_to_oldest` is `False` then `None` is returned.

Return type

list of float, or None

Notes

The point location(s) sample the two grids with ages bounding `time` and then interpolate between them.

However if `time` is older than the oldest grid then the oldest grid is sampled (if `fallback_to_oldest` is `True`).

All returned sample values are non-NaN.

New in version 1.4.

2.7.9 Average sea level variations

Read a sea level file and compute average sea level variations during time periods.

Summary

`pybacktrack.SeaLevel` is a class that calculates integrated sea levels (relative to present day) over a time period.

Detail

class `pybacktrack.SeaLevel`(*sea_level_filename*)

Class to calculate integrated sea levels (relative to present day) over a time period.

`__init__`(*sea_level_filename*)

Load sea level curve (linear segments) from file.

Parameters

`sea_level_filename` (*`str`*) – Text file with first column containing ages (Ma) and a corresponding second column of sea levels (m).

static `create_from_bundled_model`(*sea_level_model_name*)

Create a SeaLevel instance from a bundled sea level model name.

Parameters

`sea_level_model_name` (*`string`*) – Name of a bundled sea level model. Bundled sea level models include `Haq87_SealevelCurve` and `Haq87_SealevelCurve_Longterm`.

Returns

The bundled sea level model.

Return type

`pybacktrack.SeaLevel`

Raises

`ValueError` – If `sea_level_model_name` is not the name of a bundled sea level model.

Notes

New in version 1.2.

static `create_from_model_or_bundled_model_name`(*sea_level_model_or_bundled_model_name*)

Create a SeaLevel instance from a user-provided model or from a bundled model.

Parameters

`sea_level_model_or_bundled_model_name` (*`string`*) – Either a user-provided model specified as a text filename containing sea level curve (see *`pybacktrack.SeaLevel.__init__()`*), or name of a bundled model (see *`pybacktrack.SeaLevel.create_from_bundled_model()`*), .

Returns

The sea level model loaded from a user-provided model or from a bundled model.

Return type

`pybacktrack.SeaLevel`

Notes

New in version 1.4.

`get_average_level`(*begin_time*, *end_time*)

Return the average sea level over the specified time period.

Parameters

- **`begin_time`** (*`float`*) – The begin time (in Ma). Should be larger than *end_time*.
- **`end_time`** (*`float`*) – The end time (in Ma). Should be smaller than *begin_time*.

Returns
Average sea level (in metres).

Return type
float

Notes

The average sea level is obtained by integrating sea level curve over the specified time period and then dividing by time period.

2.7.10 Converting stratigraphic depth to age

Convert stratigraphic depths (metres) to age (Ma) using an depth-to-age model.

Summary

`pybacktrack.convert_stratigraphic_depth_to_age()` converts a single stratigraphic depth to an age.

`pybacktrack.convert_stratigraphic_depth_to_age_files()` converts a sequence of stratigraphic depths (read from an input file) to ages (and writes both ages and depths, and any lithologies in the input file, to an output file).

Detail

pybacktrack.**convert_stratigraphic_depth_to_age**(*age*, *depth_to_age_model*)

Convert stratigraphic depth to age using a specified depth-to-age model.

Parameters

- **depth** (*float*) – The stratigraphic depth in metres.
- **depth_to_age_model** (*function*) – The model to use when converting stratigraphic depth to age. A callable function accepting a single non-negative depth parameter (in metres) and returning age (in Ma).

Returns
Age (in Ma) as a positive number.

Return type
float

Raises

- **ValueError** – If *depth* is negative.
- **TypeError** – If *depth_to_age_model* is not a function accepting a single parameter.

Notes

New in version 1.5.

`pybacktrack.`**`convert_stratigraphic_depth_to_age_files`**(*input_filename*, *output_filename*, *depth_to_age_model*, *reverse_output_columns=False*)

Converts stratigraphic depth to age by reading *depth* rows (in first column) from input file and writing rows containing both *age* and *depth* to output file.

Parameters

- **`input_filename`** (*`string`*) – Name of input text file containing the *depth* values. A single *depth* value is obtained from each row by indexing the first column.
- **`output_filename`** (*`string`*) – Name of output text file containing *age* and *depth* values. Each row of output file contains an *age* value and its associated *depth* value (with order depending on *reverse_output_columns*).
- **`depth_to_age_model`** (*`function`*) – The model to use when converting stratigraphic depth to age. A callable function accepting a single non-negative depth parameter (in metres), and returning age (in Ma) or *None* to exclude from output.
- **`reverse_output_columns`** (*`bool, optional`*) – Determines order of *age* and *depth* columns in output file. If *True* then output *depth age*, otherwise output *age depth*.

Raises

- **`ValueError`** – If cannot read *depth* value, as a floating-point number, from input file in the first column.
- **`ValueError`** – If stratigraphic depths are not monotonically increasing.

Notes

New in version 1.5.

2.7.11 Utilities

Interpolate a sequence of linear segments read from a 2-column file at the values read from a 1-column file.

Summary

`pybacktrack.read_interpolate_function()` reads x and y columns from a curve file and returns a function y(x) that linearly interpolates.

`pybacktrack.interpolate_file()` interpolates a curve function at *x* positions, read from input file, and stores both *x* and interpolated *y* values to output file.

Detail

pybacktrack.**read_interpolate_function**(*curve_filename*, *x_column_index=0*, *y_column_index=1*, *out_of_bounds='clamp'*)

Read x and y columns from a curve file and return a function y(x) that linearly interpolates.

Parameters

- **curve_filename** (*string*) – Name of input text file containing the *x* and *y* data from which to create the returned curve function.
- **x_column_index** (*int, optional*) – Determines which column of input text file to read *x* values from.
- **y_column_index** (*int, optional*) – Determines which column of input text file to read *y* values from.
- **out_of_bounds** (*string, optional*) – Determines the *y* value returned by curve function when *x* is outside the range of *x* values in curve file. This can be:
  - *clamp* to return the boundary *y* value, or
  - *exclude* to return *None* (eg, to indicate that there's no *y* value), or
  - *extrapolate* to return an extrapolated value.

Returns

- **curve_function** (*function*) – A callable function *y=f(x)* accepting a single *x* argument, and returning a *y* value or *None* (if no *y* value).
- **x_column** (*list of float*) – The *x* values read from the curve file.
- **y_column** (*list of float*) – The *y* values read from the curve file.

Raises

- **ValueError** – If cannot read x and y columns, as floating-point numbers, from the curve file at column indices *x_column_index* and *y_column_index*.
- **ValueError** – If curve file contains no data.
- **ValueError** – If *out_of_bounds* is not *clamp*, *exclude* or *extrapolate*.

Notes

The returned *x* and *y* columns are useful if integrating the curve function with `scipy.integrate.quad` (since can pass x column to its *points* argument and *len(x)* to its *limit*).

Changed in version 1.5: Added *out_of_bounds* argument. If *out_of_bounds* is *exclude* then returned curve function will return *None* for any input *x* outside the range of *x* values in curve file.

pybacktrack.**interpolate_file**(*curve_function*, *input_filename*, *output_filename*, *input_x_column_index=0*, *reverse_output_columns=False*)

Interpolate *curve_function* at *x* positions, read from input file, and store both *x* and interpolated *y* values to output file.

Parameters

- **curve_function** (*function*) – A callable function *y=f(x)* accepting a single *x* argument and returning a *y* value (or *None* to exclude from output).

- **input_filename** (*string*) – Name of input text file containing the *x* positions at which to sample *curve_function*. A single *x* value is obtained from each row by indexing the *input_x_column_index* column (zero-based index).
- **output_filename** (*string*) – Name of output text file containing *x* and *y* values. Each row of output file contains an *x* value and its associated *y* value (with order depending on *reverse_output_columns*).
- **input_x_column_index** (*int, optional*) – Determines which column of input file to read *x* values from.
- **reverse_output_columns** (*bool, optional*) – Determines order of *x* and *y* columns in output file. If *True* then output *y x*, otherwise output *x y*.

Raises

`ValueError` – If cannot read an *x* value, as a floating-point number, from input file at column index *input_x_column_index*.

Notes

Changed in version 1.5: *curve_function* can return *None*, in which case there is no output row for the input *x*.

2.7.12 Constants

This section covers the various pre-defined constants that can be passed to the above functions and classes.

Bundle data

The following bundled data comes included with the `pybacktrack` package:

- a lithologies text file
- an age grid
- a sediment thickness grid
- a crustal thickness grid
- a topography grid
- a collection of common dynamic topography models
- a couple of sea level curves

The following attributes are available to access the bundled data:

pybacktrack.BUNDLE_PATH

Base directory of the bundled data.

This is an absolute path so that scripts outside the `pybacktrack` package can also reference the bundled data. All bundle data paths are derived from this base path.

pybacktrack.BUNDLE_LITHOLOGY_FILENAMES

A list of bundled lithology filenames.

pybacktrack.DEFAULT_BUNDLE_LITHOLOGY_FILENAME

Same as `pybacktrack.PRIMARY_BUNDLE_LITHOLOGY_FILENAME`.

pybacktrack.PRIMARY_BUNDLE_LITHOLOGY_FILENAME

The primary lithology filename contains the lithologies covered in Table 1 in the pyBacktrack paper:

- Muller, R. D., Cannon, J., Williams, S. and Dutkiewicz, A., 2018, PyBacktrack 1.0: A Tool for Reconstructing Paleobathymetry on Oceanic and Continental Crust, **Geochemistry, Geophysics, Geosystems,** 19, 1898-1909, doi: 10.1029/2017GC007313.

pybacktrack.EXTENDED_BUNDLE_LITHOLOGY_FILENAME
The optional extended lithology filename extends the primary lithologies, and mostly contains lithologies in shallow water.

pybacktrack.BUNDLE_AGE_GRID_FILENAME
Bundled age grid file.

pybacktrack.BUNDLE_TOPOGRAPHY_FILENAME
Bundled topography/bathymetry grid file.

pybacktrack.BUNDLE_TOTAL_SEDIMENT_THICKNESS_FILENAME
Bundled total sediment thickness grid file.

pybacktrack.BUNDLE_CRUSTAL_THICKNESS_FILENAME
Bundled crustal thickness grid file.

pybacktrack.BUNDLE_DYNAMIC_TOPOGRAPHY_MODELS
Bundled dynamic topography models.

This is a dict mapping dynamic topography model name to model information 3-tuple of (grid list filenames, static polygon filename and rotation filenames). Each *key* or *value* in the dict can be passed to the `dynamic_topography_model` argument of *pybacktrack.backtrack_well()* and *pybacktrack.backtrack_and_write_well()*.

pybacktrack.BUNDLE_DYNAMIC_TOPOGRAPHY_MODEL_NAMES
A list of bundled dynamic topography model *names* (keys in *BUNDLE_DYNAMIC_TOPOGRAPHY_MODELS*).

Choices include `terra`, `M1`, `M2`, `M3`, `M4`, `M5`, `M6`, `M7`, `ngrand`, `s20rts`, `smean`, `AY18`, `KM16`, `D10_gmcm9` and `gld428`.

pybacktrack.BUNDLE_SEA_LEVEL_MODELS
Bundled sea level models.

This is a dict mapping sea level model name to sea level file. Each *key* or *value* in the dict can be passed to the `sea_level_model` argument of *pybacktrack.backtrack_well()* and *pybacktrack.backtrack_and_write_well()*.

pybacktrack.BUNDLE_SEA_LEVEL_MODEL_NAMES
A list of bundled sea level model *names* (keys in *BUNDLE_SEA_LEVEL_MODELS*).

Choices include `Haq87_SealevelCurve` and `Haq87_SealevelCurve_Longterm`.

pybacktrack.BUNDLE_RECONSTRUCTION_ROTATION_FILENAMES
Rotation files of the reconstruction model used to reconstruct sediment-deposited crust for paleobathymetry gridding.

pybacktrack.BUNDLE_RECONSTRUCTION_STATIC_POLYGON_FILENAME
Static polygon file of the reconstruction model used to assign plate IDs to points on sediment-deposited crust for paleobathymetry gridding.

Backtracking

pybacktrack.BACKTRACK_DEFAULT_DECOMPACTED_COLUMNS

Default list of decompacted columns used for `decompacted_columns` argument of *pybacktrack.backtrack_well()* and *pybacktrack.backtrack_and_write_well()*.

List of column types available for the `decompacted_columns` argument of *pybacktrack.backtrack_well()* and *pybacktrack.backtrack_and_write_well()*:

- `pybacktrack.BACKTRACK_COLUMN_AGE`
- `pybacktrack.BACKTRACK_COLUMN_COMPACTED_DEPTH`
- `pybacktrack.BACKTRACK_COLUMN_COMPACTED_THICKNESS`
- `pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_THICKNESS`
- `pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DENSITY`
- `pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_SEDIMENT_RATE`
- `pybacktrack.BACKTRACK_COLUMN_DECOMPACTED_DEPTH`
- `pybacktrack.BACKTRACK_COLUMN_DYNAMIC_TOPOGRAPHY`
- `pybacktrack.BACKTRACK_COLUMN_TECTONIC_SUBSIDENCE`
- `pybacktrack.BACKTRACK_COLUMN_WATER_DEPTH`
- `pybacktrack.BACKTRACK_COLUMN_LITHOLOGY`

Backstripping

pybacktrack.BACKSTRIP_DEFAULT_DECOMPACTED_COLUMNS

Default list of decompacted columns used for `decompacted_columns` argument of *pybacktrack.backstrip_well()* and *pybacktrack.backstrip_and_write_well()*.

List of column types available for the `decompacted_columns` argument of *pybacktrack.backstrip_well()* and *pybacktrack.backstrip_and_write_well()*:

- `pybacktrack.BACKSTRIP_COLUMN_AGE`
- `pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_THICKNESS`
- `pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_DENSITY`
- `pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_SEDIMENT_RATE`
- `pybacktrack.BACKSTRIP_COLUMN_DECOMPACTED_DEPTH`
- `pybacktrack.BACKSTRIP_COLUMN_AVERAGE_TECTONIC_SUBSIDENCE`
- `pybacktrack.BACKSTRIP_COLUMN_MIN_TECTONIC_SUBSIDENCE`
- `pybacktrack.BACKSTRIP_COLUMN_MAX_TECTONIC_SUBSIDENCE`
- `pybacktrack.BACKSTRIP_COLUMN_AVERAGE_WATER_DEPTH`
- `pybacktrack.BACKSTRIP_COLUMN_MIN_WATER_DEPTH`
- `pybacktrack.BACKSTRIP_COLUMN_MAX_WATER_DEPTH`
- `pybacktrack.BACKSTRIP_COLUMN_COMPACTED_THICKNESS`
- `pybacktrack.BACKSTRIP_COLUMN_LITHOLOGY`
- `pybacktrack.BACKSTRIP_COLUMN_COMPACTED_DEPTH`

Paleobathymetry

`pybacktrack.DEFAULT_PALEO_BATHYMETRY_LITHOLOGY_NAME`

Default name of the lithology of all sediment (for paleo bathymetry gridding the total sediment thickness at all sediment locations consists of a single lithology). This lithology is the average of the ocean floor sediment. This differs from the base lithology of drill sites where the undrilled portions are usually below the Carbonate Compensation Depth (CCD) where shale dominates.

Lithology

`pybacktrack.DEFAULT_BASE_LITHOLOGY_NAME`

Default name of the lithology of the stratigraphic unit at the base of a drill site (the undrilled portion). This lithology is shale since the undrilled portions are usually below the Carbonate Compensation Depth (CCD) where shale dominates.

Oceanic subsidence

`pybacktrack.AGE_TO_DEPTH_MODEL_RHCW18`

Richards et al. (2020) `Structure and dynamics of the oceanic lithosphere-asthenosphere system`.

`pybacktrack.AGE_TO_DEPTH_MODEL_CROSBY_2007`

Crosby, A.G., (2007) `Aspects of the relationship between topography and gravity on the Earth and Moon, PhD thesis`.

`pybacktrack.AGE_TO_DEPTH_MODEL_GDH1`

Stein and Stein (1992) `Model for the global variation in oceanic depth and heat flow with lithospheric age`.

`pybacktrack.AGE_TO_DEPTH_DEFAULT_MODEL`

The age-to-depth model to use by default.

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX