StratMC Documentation

Release 0.1.1b

Stacey Edmonsond

Aug 14, 2024

CONTENTS

1 Installation **3**
- 1.1 PIP . . . 3
- 1.2 Latest (*unstable*) . . . 3
- 1.3 Installing on Apple Silicon . . . 3

2 Quick start guide **5**

3 Data table format **7**
- 3.1 Stratigraphic proxy data . . . 7
- 3.2 Age constraints . . . 8

4 API Reference **9**
- 4.1 Loading and processing data . . . 9
- 4.2 Statistical model . . . 14
- 4.3 Inference . . . 20
- 4.4 Plotting . . . 26
- 4.5 Tools for generating synthetic data . . . 49
- 4.6 Tests and checks . . . 56
- 4.7 Indices and Tables . . . 58

Python Module Index **59**

Index **61**

StratMC

StratMC is a statistical framework for reconstructing past Earth system change using sediment-hosted proxy data. It is built on the Python probabilistic programming library PyMC, which provides a flexible toolbox for constructing Bayesian models and sampling their posteriors using Markov chain Monte Carlo (MCMC) methods.

Using geochemical proxy observations and geological age constraints from multiple stratigraphic sections, StratMC simultaneously infers the global proxy signal recorded by all sections and builds an age model for each section. For a complete description of the model, see Edmonsond & Dyer (submitted to *Geoscientific Model Development*).

The StratMC Python package can be *installed* from PyPI using the pip package installer. The *API Reference* catalogs built-in functions for processing data, running the inference model, and plotting the results. For example notebooks, refer to the online package documentation (https://stratmc.readthedocs.io/).

CHAPTER

ONE

INSTALLATION

If you're new to Python, we recommend using Anaconda to install Python on your machine. You can then manage packages from the terminal using `conda`.

To create a new conda environment for StratMC, run:

```
conda create --name stratmc_env
```

Before installing StratMC, activate the new environment and install `pip`:

```
conda activate stratmc_env
conda install pip
```

You can then install StratMC and its dependencies using `pip` (note that the `--pre` flag is required to install the current version, which is a pre-release), or by compiling directly from the GitHub repository:

1.1 PIP

```
pip install stratmc --pre
```

1.2 Latest (*unstable*)

```
pip install git+https://github.com/sedmonsond/stratmc
```

1.3 Installing on Apple Silicon

On Apple Silicon machines (M1 chip or later), sampling is significantly faster when the Apple Accelerate BLAS library is used, rather than the default OpenBLAS library. After installing StratMC in a new conda environment, run:

```
conda install -c conda-forge numpy">=1.17.0,<2" "libblas=*=*accelerate"
```

CHAPTER

TWO

QUICK START GUIDE

Here, we provide a basic example of importing data and running the inference model. More complex examples are available on the example notebooks page.

1. Fill out `proxy data` and `age constraint` tables according to the *data formatting* specifications.
2. Pre-process your data for model construction with *load_data()*

```
from stratmc.data import load_data

sample_data, age_data = load_data('path_to_proxy_data.csv', 'path_to_age_
↪data.csv')
```

3. Build a `pymc.model.core.Model` with *build_model(sample_data, age_data, proxies = ['proxy'])*

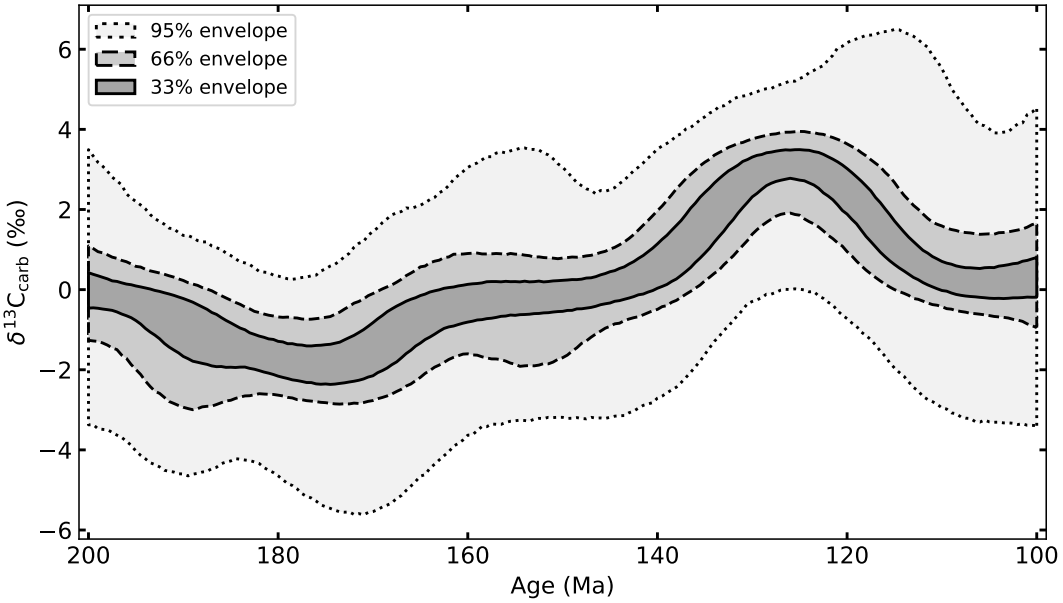
```
from stratmc.model import build_model

model, gp = build_model(sample_data, age_data, proxies = ['d13c'])
```

4. Sample the model posterior using a JAX-assisted MCMC sampling algorithm with `get_trace(model, gp, ages)`.

```
import pymc as pm
from stratmc.inference import get_trace

# array of ages at which to sample the posterior proxy curve
predict_ages = np.linspace(lower_age, upper_age, number_ages)

trace = get_trace(model, gp, predict_ages)
```

5. Plot and analyze the results with the *stratmc.plotting* library.

```
from stratmc import plotting

plotting.proxy_inference(sample_data, age_data, trace)
```


95% envelope
66% envelope
33% envelope
6
4
2
0
−2
−4
−6
200
180
160
140
120
100
Age (Ma)
$\delta^{13}C_{carb}$ (‰)

CHAPTER

THREE

DATA TABLE FORMAT

The inference model requires two inputs: proxy data for multiple stratigraphic sections and age constraints (at least a minimum and maximum age for each section). Proxy data and age constraints should be saved in separate `.csv` files formatted according to the tables below.

3.1 Stratigraphic proxy data

The table below describes the values expected in the data table (`.csv` file) with proxy data for each section. Each entry corresponds to a single sample. Optional parameters that you do not wish to specify should be left blank, and will be replaced with the default value when the data are loaded with *load_data()* in *stratmc.data*.

Parameter	Description
`section`	Name of the **section** this data belongs to (type `string`).
`height`	Height of sample in section (in meters). For sections where stratigraphic position instead is described by **depth**, leave blank.
`depth`	Depth of sample in core (in meters). For sections where stratigraphic position instead is described by **height**, leave blank.
`proxy`	Proxy value for sample (replace column header with name of proxy).
`proxy_std`	1-σ uncertainty of proxy measurement (in column header, **proxy** must match name of the column with the associated proxy value). If not provided, defaults to 0.1.
`offset_group`	Only required if custom groups of samples share an offset term in the model (otherwise, all samples from the same section will share an offset term). Set to the same value (`string` or `float`) for all samples from the same group.
`noise_group`	Only required if custom groups of samples share a noise term in the model (otherwise, all samples from the same section will share a noise term). Set to the same value (`string` or `float`) for all samples from the same group.
`Exclude?`	Whether to exclude sample from the inference (`True` or `False`). Optional; defaults to `False` if not provided.

3.2 Age constraints

The table below describes the values expected in the data table (`.csv` file) with age constraints for each section. Each entry corresponds to a single age constraint from one of the sections included in the `Stratigraphic proxy data` table. Optional parameters that you do not wish to specify should be left blank, and will be replaced with the default value when the data are loaded with *load_data()* in *stratmc.data*.

Parameter	Description
`section`	Name of the **section** this data belongs to (type `string`). Must match the corresponding section in the **Chemostratigraphic Data** table.
`height`	Height of age constraint in section (in meters). For sections where stratigraphic position instead is described by **depth**, leave blank.
`depth`	Depth of age constraint in core (in meters). For sections where stratigraphic position instead is described by **height**, leave blank.
`age`	Age (in Myr).
`age_std`	1-σ age uncertainty (in Myr).
`name`	Name of age constraint (type `string`). Only required if one of `shared?` is `True`.
`shared?`	Whether the age constraint is shared between multiple sections (and should have the same age in all sections where it is present – i.e., a correlative age constraint) (`True` or `False`). Shared constraints must have the same `name`. Optional; defaluts to `False`.
`intermediate detrital?`	Whether the age constraint is an intermediate (i.e., in the middle of a section) detrital age, which provides a maximum age for all overlying samples but does not constrain the age of underlying samples (`True` or `False`. Optional; defaults to `False`.
`intermediate intrusive?`	Whether the age constraint is an intermediate (i.e., in the middle of a section) intrusive age, which provides a minimum age for all underlying samples but does not constrain the age of overlying samples (`True` or `False`). Optional; defaults to `False`.
`distribution_type`	Probability distribution used to model age constraint; must correspond to the name of a continuous `pymc.distributions` object (type `string`). Optional; defaults to *Normal*.
`param_1_name`	Name of parameter for custom distribution; only required if using a distribution other than `Normal`. If the custom distribution has no parameters, leave blank.
`param_1`	Value of parameter for custom distribution; only required if using a distribution other than `Normal`. If the custom distribution has no parameters, leave blank.
`param_2_name`	Name of parameter for custom distribution; only required if using a distribution other than `Normal`. If the custom distribution has 0 or 1 parameters, leave blank.
`param_2`	Value of parameter for custom distribution; only required if using a distribution other than `Normal`. If the custom distribution has 0 or 1 parameters, leave blank.
`Exclude?`	Whether to exclude age constraint from the inference (`True` or `False`). Optional; defaults to `False`.

CHAPTER

FOUR

API REFERENCE

4.1 Loading and processing data

Functions for importing and processing proxy and age constraint data.

load_data	Import and pre-process proxy data and age constraints from .csv files formatted according to the *Data table formatting* guidelines.
combine_data	Helper function for merging `pandas.DataFrame` objects containing proxy observations or age constraints.
load_object	Custom load command for pickle (.pkl) object (variables can be saved as .pkl files with *save_object()*).
load_trace	Custom load command for NetCDF file containing a trace (`arviz.InferenceData` object saved with *save_trace()*).
save_object	Save variable as a pickle (.pkl) object.
save_trace	Save trace (`arviz.InferenceData` object) as a NetCDF file.
combine_traces	Helper function for combining multiple `arviz.InferenceData` objects (saved as NetCDF files) that contain prior and posterior samples for the same inference model (sampled with *get_trace()* in *stratmc.inference*).
drop_chains	Remove a subset of chains from a `arviz.InferenceData` object.
thin_trace	Remove a subset of draws from a `arviz.InferenceData` object.
accumulation_rate	Calculate apparent sediment accumulation rate between successive samples (if `method = 'successive'`) or every possible sample pairing (`method = 'all'`).
clean_data	Helper function for cleaning sample data before running an inversion.
depth_to_height	Helper function for converting depth in core to height in section.
combine_duplicates	Helper function for combining multiple proxy measurements from the same stratigraphic horizon.

`stratmc.data.`**`accumulation_rate`**(*full_trace*, *sample_df*, *ages_df*, *method='all'*, *age_model='posterior'*, *include_age_constraints=True*, ***kwargs*)

Calculate apparent sediment accumulation rate between successive samples (if `method = 'successive'`) or

every possible sample pairing (`method = 'all'`).

Note that if `method = 'all'`, rate is returned in mm/year, and duration is returned in years. If `method = 'successive'`, rate is returned in m/Myr, and duration is returned in Myr. Input data are assumed to have units of meters and millions of years. Used as input to *sadler_plot()* and *accumulation_rate_stratigraphy()* in *stratmc.plotting*.

Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing all proxy data.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints from all sections.

method: str, optional
Whether to calculate accumulation rates between every possible sample pairing (‘all`), or between successive samples (‘successive`); defaults to ‘all`.

age_model: str, optional
Whether to calculate accumulation rates using the the posterior or prior age model for each section; defaults to ‘posterior`.

include_age_constraints: bool, optional
Whether to include radiometric age constraints in accumulation rate calculations; defaults to `True`.

sections: list(str) or numpy.array(str), optional
List of sections to include. Defaults to all sections in `sample_df`.

Returns

rate_df: pandas.DataFrame
`pandas.DataFrame` containing sediment accumulation rates and associated durations.

`stratmc.data.`**`clean_data`**(*sample_df*, *ages_df*, *proxies*, *sections*)

Helper function for cleaning sample data before running an inversion. Sets `Exclude?` to `True` for samples with no relevant proxy observations, removes sections where all samples have been excluded, and drops excluded age constraints.

Parameters

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

proxies: str or list(str)
Tracers to include in the inference.

sections: list(str) or numpy.array(str)
List of sections to include in the inference (as named in `sample_df` and `ages_df`).

Returns

sample_df: pandas.DataFrame
`pandas.DataFrame` containing cleaned proxy data for all sections.

ages_df: pandas.DataFrame
pandas.DataFrame containing cleaned age constraint data for all sections.

stratmc.data.**combine_data**(*dataframes*)

Helper function for merging pandas.DataFrame objects containing proxy observations or age constraints. Data are merged using the section and height columns.

Parameters

dataframes: list(pandas.DataFrame)
List of pandas.DataFrame objects to merge.

Returns

merged_data: pandas.DataFrame
pandas.DataFrame containing merged data.

stratmc.data.**combine_duplicates**(*sample_df*, *proxies*, *proxy_sigma_default=0.1*)

Helper function for combining multiple proxy measurements from the same stratigraphic horizon. For each horizon with multiple proxy values, replaces the proxy value with the mean, and replaces the standard deviation with the combined uncertainty (proxy_std values summed in quadrature) for all measurements. The standard deviation of the population of proxy values for each horizon is stored in the proxy_population_std column of sample_df (in *build_model()*, the uncertainty of each proxy observation is modeled as the proxy_std and proxy_population_std values summed in quadrature).

Parameters

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections.

proxies: list(str)
List of proxies to include in the inference.

proxy_sigma_default: float or dict{float}, optional
Measurement uncertainty (1σ) to use for proxy observations if not specified in proxy_std column of sample_df. To set a different value for each proxy, pass a dictionary with proxy names as keys. Defaults to 0.1.

Returns

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data with duplicates combined.

stratmc.data.**combine_traces**(*trace_list*)

Helper function for combining multiple arviz.InferenceData objects (saved as NetCDF files) that contain prior and posterior samples for the same inference model (sampled with *get_trace()* in *stratmc.inference*). The arviz.InferenceData objects are concatenated along the chain dimension such that if two traces with 8 chains each are concatenated, the new combined trace will have 16 chains.

Parameters

trace_list: list(str)
List of paths to arviz.InferenceData objects (saved as NetCDF files) to be merged.

Returns

combined_trace: arviz.InferenceData
New arviz.InferenceData object containing the prior and posterior draws for all traces in trace_list.

stratmc.data.**depth_to_height**(*sample_df*, *ages_df*)

Helper function for converting depth in core to height in section.

Parameters

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections.

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints for all sections.

Returns

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections, with depth in core converted to height in section.

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints for all sections, with depth in core converted to height in section.

stratmc.data.**drop_chains**(*full_trace*, *chains*)

Remove a subset of chains from a arviz.InferenceData object.

Parameters

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

chains: list or np.array of int
Indices of chains to remove from full_trace.

Returns

full_trace_clean: arviz.InferenceData
Copy of full_trace without the chains specified in chains.

stratmc.data.**load_data**(*sample_file*, *ages_file*, *proxies=['d13c']*, *proxy_sigma_default=0.1*, *drop_excluded_samples=False*, *drop_excluded_ages=True*)

Import and pre-process proxy data and age constraints from .csv files formatted according to the *Data table formatting* guidelines. To combine data from different .csv files, load each file separately and then combine the DataFrames with *combine_data()*.

If sample_file.csv includes multiple proxy observations from the same stratigraphic horizon (for a given proxy), then all measurements marked Exclude? = False will be combined using *combine_duplicates()*.

Parameters

sample_file: str
Path to .csv file containing proxy data for all sections (without '.csv` extension).

ages_file: str
Path to .csv file containing age constraints for all sections (without '.csv` extension).

proxies: str or list(str), optional
Tracer names (must match column headers in sample_file.csv); defaults to 'd13c`.

proxy_sigma_default: float or dict{float}, optional
Measurement uncertainty (1σ) to use for proxy observations if not specified in proxy_std column of sample_df. To set a different value for each proxy, pass a dictionary with proxy names as keys. Defaults to 0.1.

drop_excluded_samples: bool, optional
Whether to remove samples with Exclude? = True from the sample_df; defaults to

`False`. If excluded samples are not dropped, their ages will be passively tracked within the inference model (but they will not be considered during the proxy signal reconstruction).

drop_excluded_ages: bool, optional
Whether to remove ages with `Exclude? = True` from the `ages_df`; defaults to `True`.

Returns

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

`stratmc.data.`**`load_object`**(*path*)

Custom load command for pickle (.pkl) object (variables can be saved as .pkl files with *save_object()*).

Parameters

path: str
Path to saved .pkl file (without the '.pkl` extension).

Returns

var:
Variable saved in `path`.

`stratmc.data.`**`load_trace`**(*path*)

Custom load command for NetCDF file containing a trace (`arviz.InferenceData` object saved with *save_trace()*).

Parameters

path: str
Path to saved NetCDF file (without the '.nc` extension).

Returns

trace: arviz.InferenceData
Trace saved as NetCDF file.

`stratmc.data.`**`save_object`**(*var*, *path*)

Save variable as a pickle (.pkl) object.

Parameters

var:
Variable to be saved.

path: str
Location (including the file name, without '.pkl` extension) to save `var`.

`stratmc.data.`**`save_trace`**(*trace*, *path*)

Save trace (`arviz.InferenceData` object) as a NetCDF file.

Parameters

trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *build_model()* in *stratmc.model* (the output of *get_trace()* in *stratmc.inference*).

path: str
Location (including the file name, without '.nc` extension) to save `trace`.

stratmc.data.**thin_trace**(*full_trace*, *drop_freq=2*)

Remove a subset of draws from a arviz.InferenceData object. Only applies to groups associated with the posterior (the prior draws will not be affected).

Parameters

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

drop_freq: int
Frequency of draw removal. For example, 2 will remove every other draw, while 4 will remove every fourth draw.

Returns

thinned_trace: arviz.InferenceData
Thinned version of full_trace.

4.2 Statistical model

Functions for building a proxy signal inference model.

build_model	Create a proxy signal (i.e., carbon isotope) inference model.
superposition	Helper function for explicitly enforcing stratigraphic superposition (any given sample must be younger than the underlying sample) for a group of radiometric age constraints.
intermediate_detrital_potential	Helper function for enforcing detrital age constraints (maximum age constraint for all overlying samples; no constraint on underlyig samples) in the middle of a section.
intermediate_intrusive_potential	Helper function for enforcing intrusive age constraints (minimum age constraint for all underlying samples; no constraint on overlying samples) in the middle of a section.
transformed_initval	Helper function to retrieve the transformed initial values for a random variable in a pymc.model.core.Model object.

stratmc.model.**build_model**(*sample_df*, *ages_df*, *proxies=['d13c']*, *proxy_sigma_default=0.1*, *approximate=False*, *hsgp_m=15*, *hsgp_c=1.3*, *ls_dist='Wald'*, *ls_min=0*, *ls_mu=20*, *ls_lambda=50*, *ls_sigma=50*, *var_sigma=10*, *white_noise_sigma=0.1*, *gp_mean_mu=None*, *gp_mean_sigma=None*, *offset_type='section'*, *offset_prior='Laplace'*, *offset_alpha=0*, *offset_beta=1*, *offset_sigma=1*, *offset_mu=0*, *offset_b=2*, *noise_type='section'*, *noise_prior='HalfCauchy'*, *noise_beta=1*, *noise_sigma=1*, *noise_nu=1*, *jitter=0.001*, *proxy_observed=True*, ***kwargs*)

Create a proxy signal (i.e., carbon isotope) inference model.

Note that while excluded samples (Exclude? is True in sample_df) do not affect the proxy signal reconstruction, their ages are passively tracked within the inference model (if other samples from the same sections are included).

Parameters

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections. Load from .csv file using *load_data()* in *stratmc.data*.

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints for all sections. Load from .csv file using *load_data()* in *stratmc.data*.

sections:: list(str) or numpy.array(str), optional
List of sections to include in the inference model. Defaults to all sections in `sample_df`.

proxies: str or list(str), optional
Column or columns containing proxy data in `sample_df`. Defaults to 'd13c'.

proxy_sigma_default: float or dict{float}, optional
Measurement uncertainty (1σ) to use for proxy observations if not specified in `proxy_std` column of `sample_df`. To set a different value for each proxy, pass a dictionary with proxy names as keys. Defaults to 0.1.

approximate: bool, optional
Build model with an unapproximated latent GP (`pymc.gp.Latent`) if `False`, or a Hilbert space Gaussian process approximation (`pymc.gp.HSGP`) if `True`; defaults to `False`. If using the HSGP approximation, also pass the `hsgp_m` and `hsgp_c` parameters (the defaults are unlikely to work well for all problems). Appropriate values for `m` and `c` can be estimated using `approx_hsgp_hyperparams()`.

hsgp_m: int or dict{int}, optional
Number of basis vectors to use in the HSGP approximation (see `pymc.gp.HSGP`). Pass as a dictionary with proxy names as keys to specify a different value for each proxy. Defaults to 15.

hsgp_c: float or dict{float}, optional
Proportion extension factor for the HSGP approximation (see `pymc.gp.HSGP`). Pass as a dictionary with proxy names as keys to specify a different value for each proxy. Defaults to 1.3.

ls_dist: str or dict{str}, optional
Prior distribution for the lengthscale hyperparameter of the exponential quadratic covariance kernel (`pymc.gp.cov.ExpQuad`); set to `Wald` (`pymc.Wald`) or `HalfNormal` (`pymc.HalfNormal`). Defaults to `Wald` with `mu = 20` and `lambda = 50`. To change `mu` and `lambda`, pass the `ls_mu` and `ls_lambda` parameters. For `HalfNormal`, the variance defaults to `sigma = 50`; change by passing `ls_sigma`. Pass as a dictionary with proxy names as keys to specify a different prior distribution for each proxy.

ls_min: float or dict{float}, optional
Minimum value for the lengthscale hyperparameter of the `pymc.gp.cov.ExpQuad` covariance kernel; shifts the lengthscale prior by `ls_min`. Pass as a dictionary with proxy names as keys to specify a different value for each proxy. Defaults to 0.

ls_mu: float or dict{float}, optional
Mean (*mu*) of the `pymc.gp.cov.ExpQuad` lengthscale prior if `ls_dist = `Wald``. Pass as a dictionary with proxy names as keys to specify a different value for each proxy. Defaults to 20.

ls_lambda: float or dict{float}, optional
Relative precision (*lam*) of the `pymc.gp.cov.ExpQuad` lengthscale hyperparameter prior if `ls_dist = `Wald``. Pass as a dictionary with proxy names as keys to specify a different value for each proxy. Defaults to 50.

ls_sigma: float or dict{float}, optional
Scale parameter (*sigma*) of the pymc.gp.cov.ExpQuad lengthscale hyperparameter prior if ls_dist = `HalfNormal`. Pass as a dictionary with proxy names as keys to specify a different value for each proxy. Defaults to 50.

var_sigma: float or dict{float}, optional
Scale parameter (*sigma*) of the covariance kernel variance hyperparameter prior, which is a pymc.HalfNormal distribution. Pass as a dictionary with proxy names as keys to specify a different value for each proxy. Defaults to 10.

white_noise_sigma: float or dict{float}, optional
Amplitude of white noise component of GP covariance function. Defaults to 0.1. Should be equal to or less than the proxy measurement uncertainty; use a smaller value for proxies with low-amplitude signals (e.g., Sr isotopes). Pass as a dictionary with proxy names as keys to specify a different value for each proxy.

gp_mean_mu: float or dict{float}, optional
Mean (*mu*) of the GP mean function prior, which is a pymc.Normal distribution. Defaults to the mean of the observations for each proxy. Pass as a dictionary with proxy names as keys to specify a different value for each proxy.

gp_mean_sigma: float or dict{float}, optional
Standard deviation (*sigma*) of the GP mean function prior, which is a pymc.Normal distribution. Defaults to twice the standard deviation of the observations for each proxy. Pass as a dictionary with proxy names as keys to specify a different value for each proxy.

offset_type: str or dict{str}, optional
Parameterize offset such that all samples from the same section have the same offset (set to section), or such that custom sample groups share an offset term (set to groups, and specify the group for each sample and proxy with a offset_group_proxy column in sample_df). To omit the offset term, set to none. Defaults to section. Pass as a dictionary with proxy names as keys to specify a different offset type for each proxy.

offset_prior: str or dict{str}, optional
Type of distribution to use for the offset prior. Pass as a string to use the same prior for all proxies, or as a dict of string (with proxy names as keys) to specify a different prior distribution for each proxy. Defaults to Laplace (pymc.Laplace) with mu = 0 and b = 2. mu and b can be changed by passing offset_mu and offset_b. To use other types of priors, pass the name of a distribution from pymc.distributions, along with parameter values in offset_params. Pass as a dictionary with proxy names as keys to specify a different prior for each proxy.

offset_params: dict{float} or dict{dict{float}}, optional
Only required if using a custom offset_prior. Pass as a dictionary to use the same parameters for all proxies, or as a dictionary of dictionaries (with proxy names as keys) to specify different parameters for each proxy. Keys are param_1_name, param_1, param_2_name, and param_2 (with parameter names corresponding to those required for the specified pymc.distributions object). If only one parameter is required, use np.nan for param_2_name and param_2.

noise_type: str or dict{str}, optional
Parameterize noise as per-section (set to section) or per-group (set to groups, and specify the group for each sample and proxy with a noise_group_proxy column in sample_df). Defaults to section. Pass as a dictionary with proxy names as keys to specify a different noise type for each proxy.

noise_prior: str or dict{str}, optional
Type of distribution to use for the noise prior. Pass as a string to use the same prior for all

proxies, or as a `dict` of `string` (with proxy names as keys) to specify a different prior distribution for each proxy. Defaults to `HalfCauchy` (`pymc.HalfCauchy`) with `beta = 1`. `beta` can be changed by passing `noise_beta`. Other implemented priors (note that noise must be positive-only) are `HalfNormal` (`pymc.HalfNormal`; defaults to `noise_sigma = 1`) and `HalfStudentT` (`pymc.HalfStudentT`; defaults to `noise_nu = 1` and `noise_sigma = 1`).

superposition_dict: dict{list(str)}, optional
Optional; dictionary specifying superposition relationships between different sections. Should only be used when superposition is not implicitly enforced by the age constraints; for example, when sections share the same minimum and maximum age constraints, but are from geological formations with a known stratigraphic relationship. Dictionary keys are section names, and the value for each key is a list of sections that must be older (stratigraphically lower).

jitter: float, optional
Value of `jitter` passed to `pymc.gp.Latent.prior()`. Defaults to 0.001.

proxy_observed: bool, optional
Whether to pass observed values to the likelihood function; defaults to `True`. Only set to `False` to generate synthetic observations from the model prior in *synthetic_observations_from_prior()* in *stratmc.synthetics*.

Returns

model: PyMC model
`pymc.model.core.Model` object.

gp: pymc.gp.Latent or pymc.gp.HSGP
Gaussian process prior for the model. `pymc.gp.Latent` if `approximate = True`, or `pymc.gp.HSGP` if `approximate = False`.

stratmc.model.**intermediate_detrital_potential**(*detrital_age_dist*, *detrital_age_dist_name*, *maximum_age_dist_name*, *minimum_age_dist_name*, *sample_age_dist_sorted*, *sample_age_dist*, *sample_age_dist_name*, *sample_heights*, *detrital_height*, *model*, *section*, *interval*, *sf1_name*, *sf2_name*, *shared_radiometric_age_dist=True*)

Helper function for enforcing detrital age constraints (maximum age constraint for all overlying samples; no constraint on underlyig samples) in the middle of a section.

Parameters

detrital_age_dist: pymc.distributions
Detrital age distribution.

detrital_age_dist_name: str
Name of `detrital_age_dist` in `model`.

maximum_age_dist_name: str
Name of distribution for underlying maximum age constraint in `model`.

minimum_age_dist_name: str
Name of distribution for overlying minimum age constraint in `model`.

sample_age_dist_sorted: pymc.distribtions
Sorted and scaled sample age distribitions.

sample_age_dist: pymc.distributions
Unsorted and unscaled sample age distributions.

sample_age_dist_name: str
Name of `sample_age_dist` in the pymc.Model object.

sample_heights: np.array
Array of heights for samples in the current interval.

detrital_height: float
Height of detrital age constraint.

model: pymc.Model
`pymc.model.core.Model` object associated with the input distributions.

section: str
Name of the current section.

interval: int
Current interval number.

sf1_name: str
Name of the distribution associated with scaling factor 1 in `model`.

sf2_name: str
Name of the distribution associated with scaling factor 2 in `model`.

shared_radiometric_age_dist: bool, optional
Whether the radiometric age distributions are part of a single object (versus initiated as separate distributions). Defaults to `True`.

stratmc.model.**intermediate_intrusive_potential**(*intrusive_age_dist*, *intrusive_age_dist_name*, *maximum_age_dist_name*, *minimum_age_dist_name*, *sample_age_dist_sorted*, *sample_age_dist*, *sample_age_dist_name*, *sample_heights*, *intrusive_height*, *model*, *section*, *interval*, *sf1_name*, *sf2_name*, *shared_radiometric_age_dist=True*, ***kwargs*)

Helper function for enforcing intrusive age constraints (minimum age constraint for all underlying samples; no constraint on overlying samples) in the middle of a section.

Parameters

intrusive_age_dist: pymc.distributions
Intrusive age distribution.

intrusive_age_dist_name: str
Name of `intrusive_age_dist` in `model`.

maximum_age_dist_name: str
Name of distribution for underlying maximum age constraint in `model`.

minimum_age_dist_name: str
Name of distribution for overlying minimum age constraint in `model`.

sample_age_dist_sorted: pymc.distribtions
Sorted and scaled sample age distribitions.

sample_age_dist: pymc.distributions
Unsorted and unscaled sample age distributions.

sample_age_dist_name: str
Name of `sample_age_dist` in the pymc.Model object.

sample_heights: np.array
Array of heights for samples in the current interval.

intrusive_height: float
Height of intrusive age constraint.

model: pymc.Model
`pymc.model.core.Model` object associated with the input distributions.

section: str
Name of the current section.

interval: int
Current interval number.

sf1_name: str
Name of the distribution associated with scaling factor 1 in `model`.

sf2_name: str
Name of the distribution associated with scaling factor 2 in `model`.

shared_radiometric_age_dist: bool, optional
Whether the radiometric age distributions are part of a single object (versus initiated as separate distributions). Defaults to `True`.

`stratmc.model.`**`superposition`**(*age_dist*, *age_dist_names*, *model*, *section_age_df*, *section*)

Helper function for explicitly enforcing stratigraphic superposition (any given sample must be younger than the underlying sample) for a group of radiometric age constraints.

Parameters

age_dist: pymc.distribtions
`pymc.distributions` object for the radiometric ages (must be in stratigraphic order).

age_dist_name: str
Name of `age_dist` in `model`.

model: pymc.Model
`pymc.model.core.Model` object associated with the distributions in `age_dist`.

section_age_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for `section`.

section: str
Name of the section containing the radiometric age constraints.

`stratmc.model.`**`transformed_initval`**(*var_name*, *model*)

Helper function to retrieve the transformed initial values for a random variable in a `pymc.model.core.Model` object.

Parameters

var_name: str
Name of variable to be transformed (as specified in `model`).

model: PyMC model
`pymc.model.core.Model` object that contains the target random variable.

4.3 Inference

Functions for sampling the inference model, checking for convergence, and processing the results.

get_trace	Sample the prior and posterior distributions for a pymc.model.core.Model returned by *build_model()* in *stratmc.model*.
extend_age_model	Extend age models to a different set of proxy observations using posterior sample ages from an existing inference.
interpolate_proxy	Use interpolated sample ages from *extend_age_model()* to calculate proxy values at a given set of ages (e.g., to plot 68 and 95% confidence intervals over time for a new proxy using *interpolated_proxy_inference()* in *stratmc.plotting*).
age_range_to_height	Use the posterior age model for each section to find the stratigraphic interval (with uncertainty) corresponding to a given age range.
map_ages_to_section	Helper function for *section_proxy_signal()* in *stratmc.plotting*.
count_samples	Helper function for *proxy_data_density()* in *stratmc.plotting*.
find_gaps	Helper function for *proxy_data_gaps()* in *stratmc.plotting*.
calculate_lengthscale_stability	Compute the posterior standard deviation of the pymc.gp.cov.ExpQuad covariance kernel lengthscale as additional chains are considered (i.e., for 1 to *N* chains).
calculate_proxy_signal_stability	Compute the residuals between the median inferred proxy signal when all *N* chains are considered compared to when 1 to *N-1* chains are considered.

stratmc.inference.**age_range_to_height**(*full_trace*, *sample_df*, *ages_df*, *lower_age*, *upper_age*, ***kwargs*)

Use the posterior age model for each section to find the stratigraphic interval (with uncertainty) corresponding to a given age range. If `sections` is not provided, returns height estimates for every section that overlaps the target age range. To visualize the stratigraphic intervals, use *section_age_range()* in *stratmc.plotting*.

Parameters

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections.

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints for all sections.

lower_age: float
Lower bound (youngest age) of the target age interval.

upper_age: float
Upper bound (oldest age) of the target age interval.

sections: list(str) or numpy.array(str), optional
List of sections included in the original inference; only required if not all sections in `sample_df` were included.

Returns

height_range_df: pandas.DataFrame
Summary statistics for the base and top height of the target age interval (maximum likelihood estimate, median, and 68% and 95% confidence intervals) for each section.

`stratmc.inference.`**`calculate_lengthscale_stability`**(*full_trace*, ***kwargs*)

Compute the posterior standard deviation of the `pymc.gp.cov.ExpQuad` covariance kernel lengthscale as additional chains are considered (i.e., for 1 to *N* chains). When the posterior has been sufficiently explored, the standard deviation will stabilize; if it has not stabilized, then additional chains should be run. Helper function for *lengthscale_stability()* in *stratmc.plotting*.

To consider chains from multiple traces associated with the same inference model, first combine the traces (saved as NetCDF files) using *combine_traces()* in *stratmc.data*.

Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

proxy: str, optional
Name of the proxy; only required if multiple proxies were included in the inference model.

Returns

gp_ls_std: np.array of float
Posterior standard deviation of the covariance kernel lengthscale posterior; entries correspond to number of chains considered (first entry is 1 chain, last entry is all *N* chains).

`stratmc.inference.`**`calculate_proxy_signal_stability`**(*full_trace*, ***kwargs*)

Compute the residuals between the median inferred proxy signal when all *N* chains are considered compared to when 1 to *N-1* chains are considered. When the posterior has been sufficiently explored, the residuals will stabilize and approach zero; if they have not stabilized, then additional chains should be run. Helper function for *proxy_signal_stability()* in *stratmc.plotting*.

To consider chains from multiple traces associated with the same inference model, first combine the traces (saved as NetCDF files) using *combine_traces()* in *stratmc.data*.

Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

proxy: str, optional
Name of the proxy; only required if multiple proxies were included in the inference model.

Returns

median_residuals: np.array of float
Residuals between the median inferred proxy value (at each time in `ages` passed to *get_trace()*) calculated using 1 to *N-1* chains versus all *N* chains. Shape is (chains x ages).

`stratmc.inference.`**`count_samples`**(*full_trace*, *time_grid=None*)

Helper function for *proxy_data_density()* in *stratmc.plotting*. Counts the number of observations within discrete time bins (based on the posterior sample ages).

Parameters

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

time_grid: np.array, optional
Time bin edges; if not provided, defaults to the ages array passed to *get_trace()*.

Returns

sample_counts: np.array
Number of observations in each time bin, summed over all posterior draws such that the average number of observations is sample_counts/n.

grid_centers: np.array
Time bin centers.

grid_widths: np.array
Time bin widths.

n: int
Number of posterior draws in full_trace.

stratmc.inference.**extend_age_model**(*full_trace*, *sample_df*, *ages_df*, *new_proxies*, *new_proxy_df=None*, ***kwargs*)

Extend age models to a different set of proxy observations using posterior sample ages from an existing inference. For instance, extend an age model built using C isotope data to new S isotope data collected from different stratigraphic horizons in the same sections. Note that the age of stratigraphic horizons that were included in sample_df (but marked Exclude? = True) is already passively tracked within the model; this function is only required to estimate the age of observations that were not in sample_df when the inference was run. To estimate ages for new measurements of the same proxy, place the new data in a different column (e.g., 'd13c_new`).

Parameters

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

sample_df: pandas.DataFrame
pandas.DataFrame with proxy data used during the inference step (as input to *build_model()* in *stratmc.model*).

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints used during the inference step (as input to *build_model()* in *stratmc.model*).

new_proxies: str or list(str)
New proxy(s) to construct age models for.

new_proxy_df: pandas.DataFrame, optional
pandas.DataFrame containing new proxy observations. Optional; if not provided, uses sample_df (assumes that observations for the new proxy are in the same DataFrame as the original proxy observations).

sections: list(str) or numpy.array(str), optional
List of sections included in the inference; only required if not all sections in sample_df were included.

Returns

interpolated_df: pandas.DataFrame
pandas.DataFrame with interpolated age draws and sample age summary statistics (maximum likelihood estimate, median, and 68% and 95% confidence intervals) for each new proxy observation.

stratmc.inference.**find_gaps**(*full_trace*, *time_grid=None*)

Helper function for *proxy_data_gaps()* in *stratmc.plotting*. Counts the number of draws from the posterior where there are no observations within discrete time bins (based on the posterior sample ages).

Parameters

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

time_grid: np.array, optional
Time bin edges; if not provided, defaults to the `ages` array passed to *get_trace()*.

Returns

age_gaps: np.array of int
Number of posterior draws where there are no observations; each entry corresponds to an age bin (corresponding to `grid_centers` and `grid_widths`).

grid_centers: np.array
Time bin centers.

grid_widths: np.array
Time bin widths.

n: int
Number of posterior draws in `full_trace`.

stratmc.inference.**get_trace**(*model*, *gp*, *ages*, *sample_df*, *ages_df*, *proxies=['d13c']*, *approximate=False*, *name=''*, *chains=8*, *draws=1000*, *tune=2000*, *prior_draws=1000*, *target_accept=0.9*, *sampler='numpyro'*, *nuts_kwargs=None*, *jitter=0.001*, *seed=None*, *save=True*, *postprocessing_backend=None*, ***kwargs*)

Sample the prior and posterior distributions for a pymc.model.core.Model returned by *build_model()* in *stratmc.model*. By default, uses pymc.sampling.jax.sample_numpyro_nuts() to sample the posterior; change `sampler` to 'blackjax` to use pymc.sampling.jax.sample_blackjax_nuts().

After the posterior has been sampled, runs *check_inference()* in *stratmc.tests* to check that superposition is never violated in the posterior. Any chains with superposition violations are removed from the trace with *drop_chains()* before it is returned (if `save = True`, both the original and 'cleaned' traces are saved to the `traces` subfolder), and a warning is issued. See *check_inference()* for details; superposition issues are rare, and typically are related to minor violations of detrital or intrusive age constraints.

Problems during sampling, including frequent divergences or minor violations of limiting age constraints, might be resolved by increasing the number of `tune` steps and/or increasing `target_accept` (which decreases the step size).

Parameters

model: PyMC model
pymc.model.core.Model object returned by *build_model()* in *stratmc.model*.

gp: pymc.gp.Latent
Gaussian process prior (pymc.gp.Latent or pymc.gp.HSGP) returned by *build_model()* in *stratmc.model*.

ages: numpy.array(float)
array of ages at which to sample the posterior distribution of the proxy signal.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

sections:: list(str) or numpy.array(str), optional
List of sections to include in the inference model. Defaults to all sections in `sample_df`.

proxies: str or list(str), optional
List of proxies included in the model. Defaults to 'd13c'.

approximate: bool, optional
Set to `True` if the Hilbert space GP approximation (`pymc.gp.HSGP`) was used in *build_model()*; defaults to `False`.

name: str, optional
Prefix for the saved NetCDF file with the inference results (suffix is timestamp for function call).

chains: int, optional
Number of Markov chains to sample in parallel; defaults to 8.

draws: int, optional
Number of samples per chain to draw from the posterior; defaults to 1000.

tune: int, optional
Number of iterations to tune; defaults to 1000.

prior_draws: int, optional
Number of samples to draw from the prior; defaults to 1000.

target_accept: float, optional
Between 0 and 1 (exclusive). During tuning, the sampler adapts the proposals such that the average acceptance probability is equal to `target_accept`; higher values for `target_accept` typically lead to smaller step sizes. Defaults to 0.9.

sampler: str, optional
Which NUTS algorithm to use to sample the posterior (`numpyro` or `blackjax`); defaults to `numpyro`.

nuts_kwargs: dict, optional
Dictionary of keyword arguments passed to NumPyro NUTS sampler (see `pymc.sampling.jax.sample_numpyro_nuts()` and `numpyro.infer.hmc.NUTS`) or blackjax NUTS sampler (see `pymc.sampling.jax.sample_blackjax_nuts()`).

jitter: float, optional
Value of `jitter` passed to `pymc.gp.Latent.conditional()`. Defaults to 0.001. Changing this value may help if a linear algebra error is encountered during posterior predictive sampling.

postprocessing_backend: str, optional
Use the *cpu' or `gpu' for postprocessing. Defaults to ``None`.*

seed: int, optional
Random seed for sampler.

save: bool, optional
Whether to save the trace (to 'traces' subfolder in the current directory); defaults to `True`.

Returns

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples.

`stratmc.inference.`**`interpolate_proxy`**(*interp_df*, *proxy*, *ages*)

Use interpolated sample ages from *`extend_age_model()`* to calculate proxy values at a given set of ages (e.g., to plot 68 and 95% confidence intervals over time for a new proxy using *`interpolated_proxy_inference()`* in *`stratmc.plotting`*).

Parameters

interp_df: pandas.DataFrame
`pandas.DataFrame` with proxy data and interpolated ages from *`extend_age_model()`*.

proxy: str
Tracer to interpolate.

ages: list(float) or numpy.array(float)
Target ages at which to interpolate proxy values.

Returns

interpolated_proxy_df: pandas.DataFrame
`pandas.DataFrame` with interpolated proxy values and summary statistics (maximum likelihood estimate, median, and 68% and 95% confidence intervals) at each age in `ages`.

`stratmc.inference.`**`map_ages_to_section`**(*full_trace*, *sample_df*, *ages_df*, *include_radiometric_ages=False*, ***kwargs*)

Helper function for *`section_proxy_signal()`* in *`stratmc.plotting`*. Maps the `ages` array passed to *`get_trace()`* to height in each section using the most likely posterior age models.

Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

include_radiometric_ages: bool, optional
Whether to consider radiometric ages when calculating the most likely posterior age model for each section; defaults to `False`.

sections: list(str) or numpy.array(str), optional
List of sections included in the inference; only required if not all sections in `sample_df` were included.

Returns

age_model_df: pandas.DataFrame
`pandas.DataFrame` with interpolated heights at each age in the `ages` vector that was passed to *`get_trace()`*.

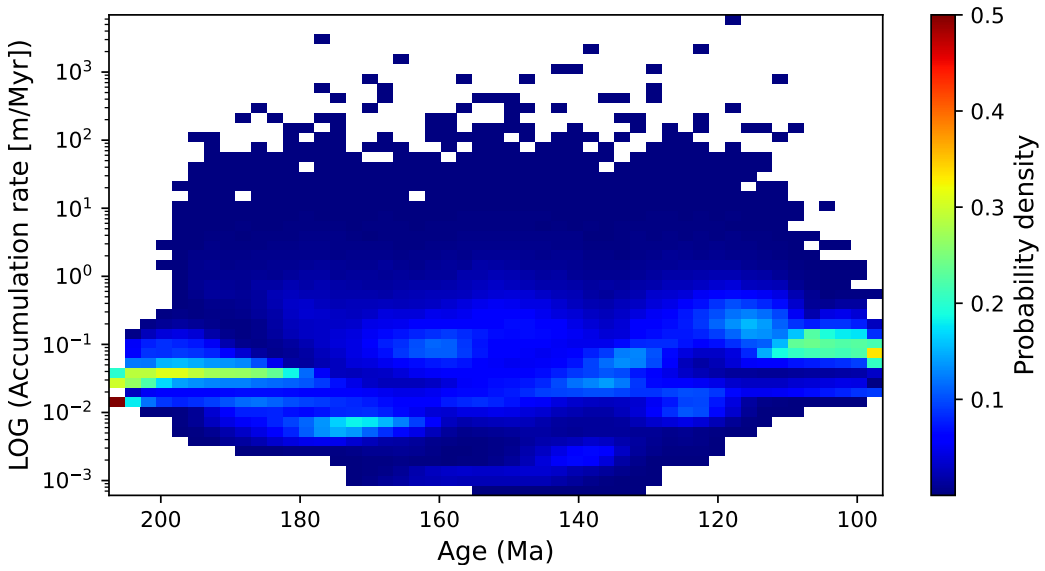
4.4 Plotting

Functions for plotting the input data, inference results, and convergence/stability metrics.

proxy_strat	Plot stratigraphic data (proxy observations and age constraints) for each section.
proxy_inference	Plot the inferred proxy signal over time.
interpolated_proxy_inference	Plot interpolated proxy signal over time (by extending the posterior section age models to a new proxy not included in the inference model) using interpolated age models from *extend_age_model()* and interpolated proxy values from *interpolate_proxy()* in *stratmc.inference*.
age_height_model	Generate a posterior age-height plot for each section.
section_proxy_signal	Map the posterior proxy signal back to height in each section (using its most likely posterior age model), and plot alongside the proxy observations (plotted by most likely posterior age).
covariance_hyperparameters	Plot prior and posterior distributions for the lengthscale (ℓ) and variance (σ) hyperparameters of the pymc.gp.cov.ExpQuad Gaussian process covariance kernel:
section_summary	For a given section, plot posterior estimates of sample age, sedimentation rate, noise, and offset.
noise_summary	Plot posterior noise distributions for each section or group of samples (depending on `noise_type` used in *build_model()*) for a given proxy.
offset_summary	Plot posterior offset distributions for each section or group of samples (depending on `offset_type` used in *build_model()*).
section_proxy_residuals	Plot the residuals between the observed proxy values for each section and the inferred proxy signal (using the posterior section age models to map the signal back to height in section).
sample_ages	Plot sample age prior and posterior distributions for a given section.
sample_ages_per_chain	Plot sample age posterior distributions for a given section, with separate distributions for each chain.
age_constraints	For a given section, plot prior and posterior age distributions for all depositional age constraints (and limiting age constraints that provide a minimum or maximum age for the entire section).
limiting_age_constraints	For a given section, plot prior and posterior age distributions for all intermediate limiting (i.e., detrital and intrusive ages in the middle of a section) age constraints.
sadler_plot	Plot accumulation rate against duration.
accumulation_rate_stratigraphy	Plot the probability density of sediment accumulation rates (calculated between successive samples) through time.
section_age_range	Plot the stratigraphic interval corresponding to a given age range (based on the posterior section age models).
proxy_data_gaps	For a set of discrete time bins, shows the number of draws from the posterior where there are no proxy observations (i.e., where there are temporal *gaps* in the proxy data).
proxy_data_density	Plot the mean number proxy observations in discrete time bins (averaged over all posterior draws).
lengthscale_traceplot	Generate trace plot (parameter value vs.
lengthscale_stability	Plot the posterior standard deviation of the pymc.gp.cov.ExpQuad covariance kernel lengthscale when 1 through N chains are considered.
proxy_signal_stability	Plot the sum (over all time slices) of the residuals between the median inferred proxy signal when all N chains are considered compared to when 1 to *N-1* chains

stratmc.plotting.**accumulation_rate_stratigraphy**(*full_trace*, *sample_df*, *ages_df*, *age_bins=50*, *age_bin_edges=[]*, *rate_bins=50*, *rate_bin_edges=[]*, *rate_scale='log'*, *include_age_constraints=True*, *cmap='jet'*, *figsize=(8, 4)*, ***kwargs*)

Plot the probability density of sediment accumulation rates (calculated between successive samples) through time. Probability density for each age bin sums to 1. Unless a `sections` argument is passed, includes accumulation rates for all sections.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

age_bins: int, optional
Number of bin edges for age data (if `age_bin_edges` not provided). Defaults to 50.

rate_bins: int, optional
Number of bin edges for rate data (if `rate_bin_edges` not provided). Defaults to 50.

age_bin_edges: list(float) or numpy.array(float), optional
List or array of bin edges for the age data.

rate_bin_edges: list(float) numpy.array(float), optional
List or array of bin edges for the rate data.

rate_scale: str, optional
Scaling for rate ('linear' or 'log'). Defaults to 'log'.

include_age_constraints: bool, optional
Include age constraints in sedimentation rate calculations. Defaults to `True`.

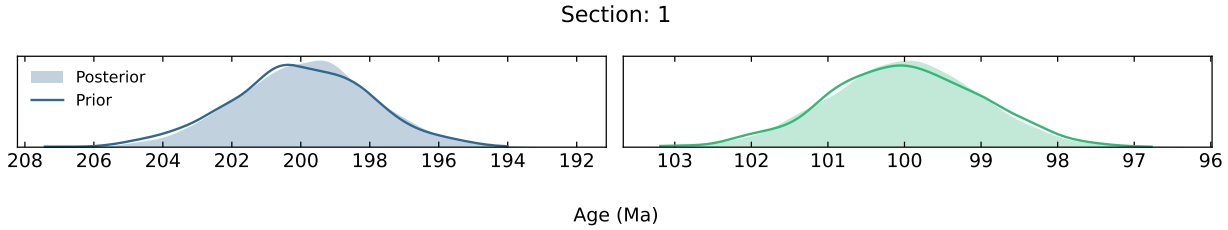
sections: list(str) or numpy.array(str), optional
Section(s) to include. If not provided, combines data from all sections in `sample_df`.

Returns

fig: matplotlib.pyplot.figure
Figure with accumulation rate probability density through time.

`stratmc.plotting.`**`age_constraints`**(*full_trace*, *section*, *cmap='viridis'*, ***kwargs*)

For a given section, plot prior and posterior age distributions for all depositional age constraints (and limiting age constraints that provide a minimum or maximum age for the entire section). To plot intermediate limiting ages (i.e., detrital and intrusive age constraints in the middle of a section), use *`limiting_age_constraints()`*.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

section: str
Name of target section.

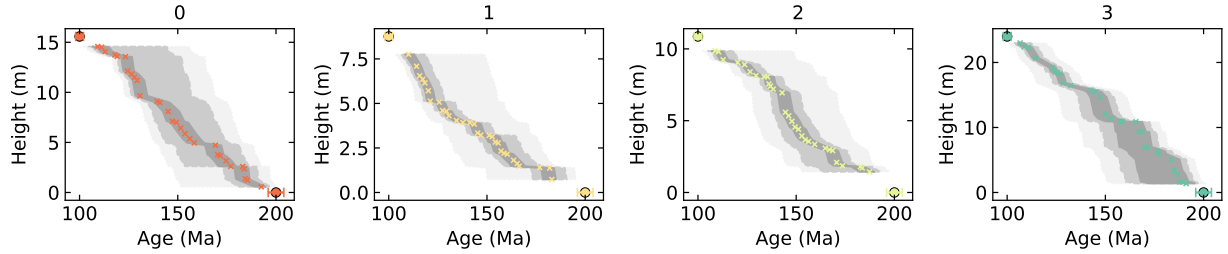
cmap: str, optional
Name of seaborn color palette to use for age distributions. Defaults to 'viridis'.

Returns

fig: matplotlib.pyplot.figure
Figure with prior and posterior depositional age constraint distributions.

`stratmc.plotting.`**`age_height_model`**(*sample_df*, *ages_df*, *full_trace*, *include_excluded_samples=True*, *plot_samples=True*, *cmap='Spectral'*, *legend=False*, ***kwargs*)

Generate a posterior age-height plot for each section.



Parameters

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sections: list(str) or numpy.array(str), optional
List of sections to plot. Defaults to all sections in `sample_df`.

cmap: str, optional
Name of seaborn color palette to use for sections. Defaults to 'Spectral'.

legend: bool, optional
Generate a legend. Defaults to `False`.

include_excluded_samples: bool, optional
Whether to consider excluded samples (`Exclude?` is `True` in `sample_df`) whose ages were passively tracked in the inference model. Defaults to `True`.

plot_samples: bool, optional
Plot proxy observations by most likely posterior age. Defaults to `True`.

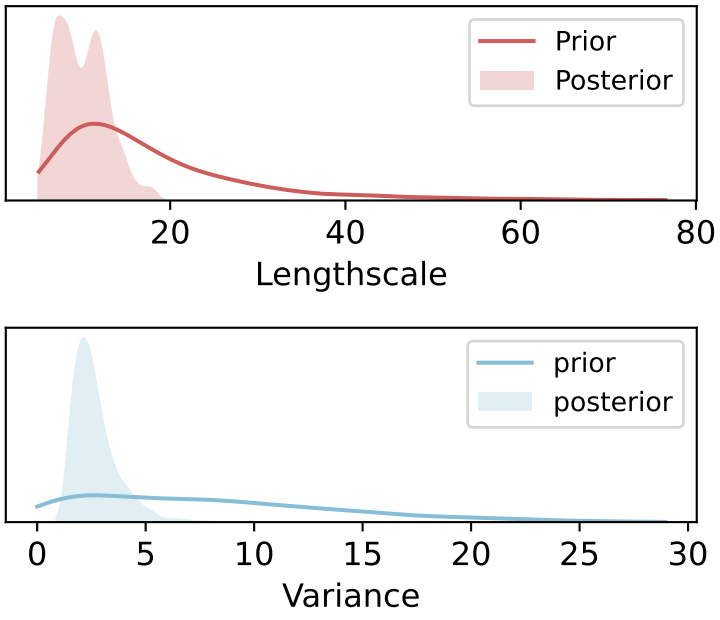
Returns

fig: matplotlib.pyplot.figure
Figure with age-height models for each section.

`stratmc.plotting.`**`covariance_hyperparameters`**(*full_trace*, *figsize=(4, 3.5)*, ***kwargs*)

Plot prior and posterior distributions for the lengthscale (ℓ) and variance (σ) hyperparameters of the `pymc.gp.cov.ExpQuad` Gaussian process covariance kernel:

$$k(x, x') = \sigma^2 \exp\left[-\frac{(x - x')^2}{2\ell^2}\right]$$



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

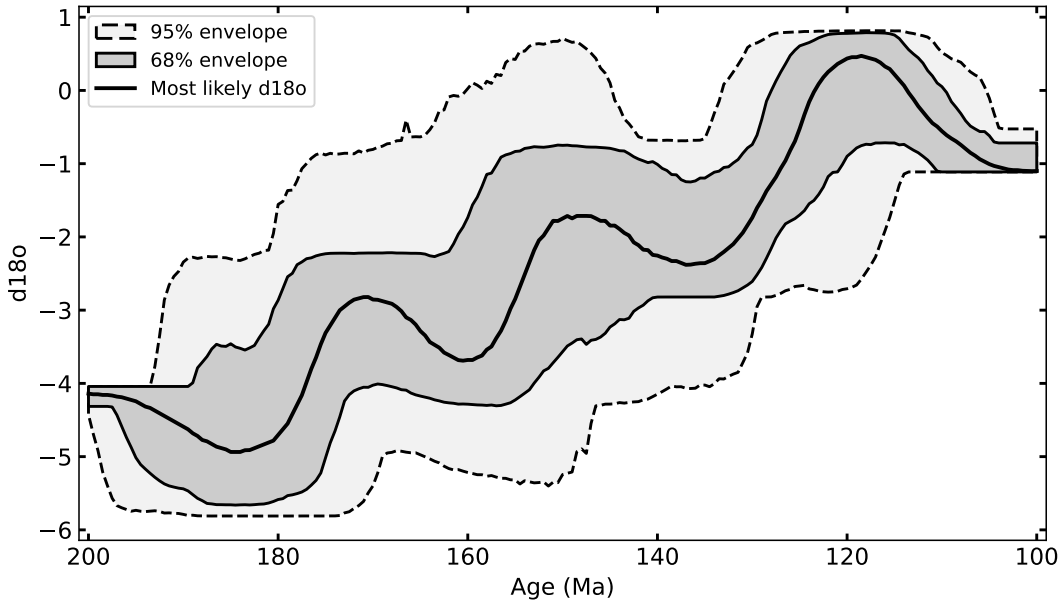
proxy: str, optional
Tracer to plot model parameters for (each proxy has a different covariance kernel); only required if more than one proxy was included in the inference.

Returns

fig: matplotlib.pyplot.figure
Figure with prior and posterior model parameters.

stratmc.plotting.**interpolated_proxy_inference**(*interpolated_df*, *interpolated_proxy_df*, *proxy*, *legend=True*, *plot_samples=False*, *plot_mle=True*, *orientation='horizontal'*, *section_legend=False*, *marker_size=20*, *section_cmap='Spectral'*, ***kwargs*)

Plot interpolated proxy signal over time (by extending the posterior section age models to a new proxy not included in the inference model) using interpolated age models from *`extend_age_model()`* and interpolated proxy values from *`interpolate_proxy()`* in *`stratmc.inference`*.



Parameters

interpolated_df: pandas.DataFrame
`pandas.DataFrame` with interpolated age draws and sample age summary statistics from *`extend_age_model()`* in *`stratmc.inference`*.

interpolated_proxy_df: pandas.DataFrame
`pandas.DataFrame` with interpolated proxy values and summary statistics at target ages from *`interpolate_proxy()`* in *`stratmc.inference`*.

proxy: str
Name of new proxy (must match column name in `interpolated_proxy_df`).

legend: bool, optional
Generate a legend. Defaults to `True`.

plot_samples: bool, optional
Plot proxy observations by most likely posterior age. Defaults to `False`.

plot_mle: bool, optional
Plot the maximum likelihood estimate. Defaults to `True`.

orientation: str, optional
Orientation of figure ('horizontal' or 'vertical'). Defaults to 'horizontal'.

marker_size: int, optional
Size of markers if `plot_samples` is `True`. Defaults to 20.

section_legend: bool, optional
Include section names in the legend (if `plot_samples` is `True`). Defaults to `False`.

section_cmap: str, optional
Name of seaborn color palette to use for sections (if `plot_samples` is `True`). Defaults to 'Spectral'.

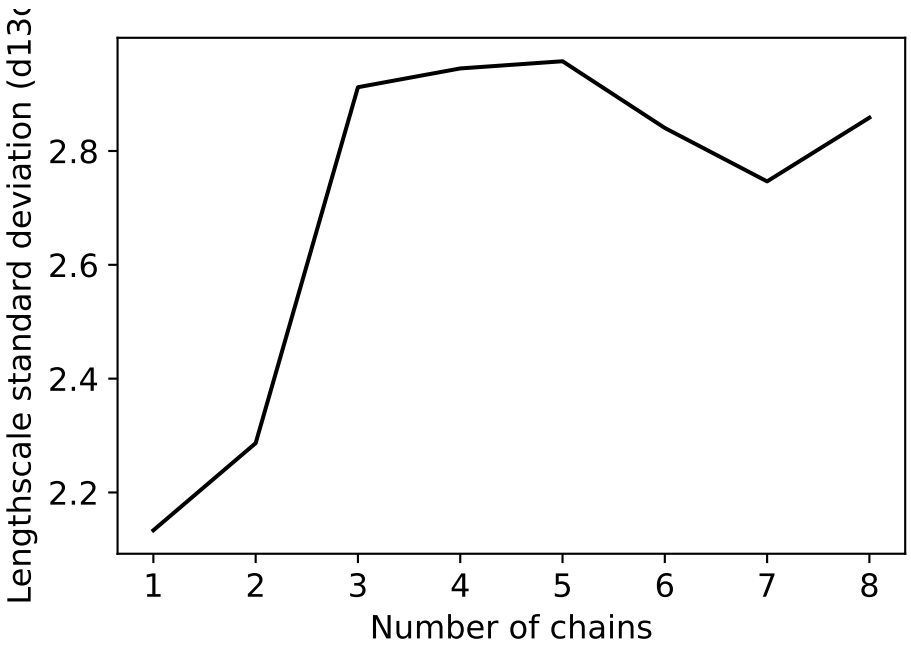
Returns

fig: matplotlib.pyplot.figure
Figure with interpolated proxy signal over time.

`stratmc.plotting.`**`lengthscale_stability`**(*full_trace*, *figsize=(5, 3.5)*, ***kwargs*)

Plot the posterior standard deviation of the `pymc.gp.cov.ExpQuad` covariance kernel lengthscale when 1 through N chains are considered. When the posterior has been sufficiently explored, the standard deviation will stabilize; if it has not stabilized, then additional chains should be run.

To consider chains from multiple traces associated with the same inference model, first combine the traces (saved as NetCDF files) using *`combine_traces()`* in *`stratmc.data`*.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

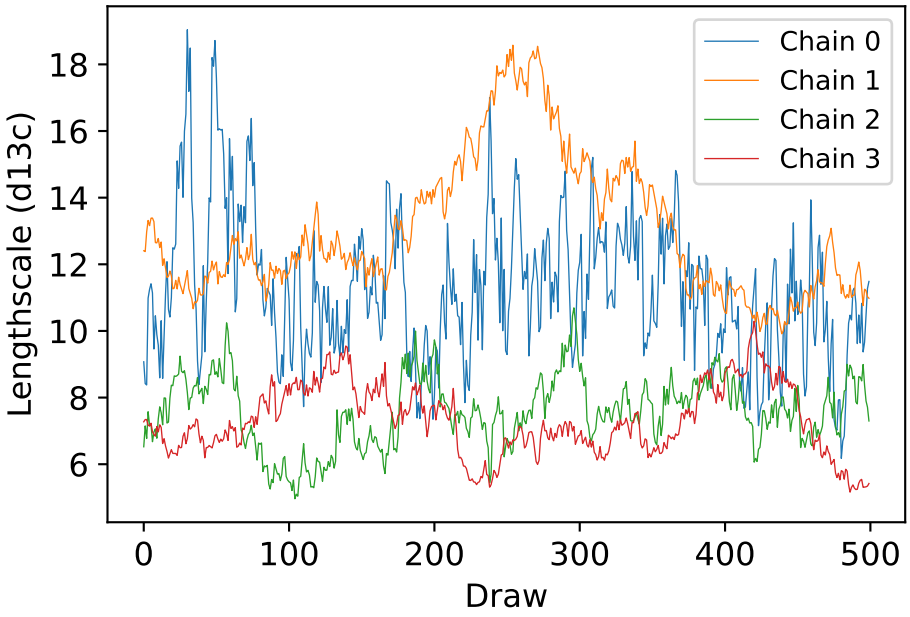
proxy: str, optional
Target proxy; only required if more than one proxy was included in the inference.

Returns

fig: matplotlib.pyplot.figure
Figure showing the standard deviation of the covariance kernel lengthscale hyperparameter posterior for 1 through *N* chains.

`stratmc.plotting.`**`lengthscale_traceplot`**(*full_trace*, *chains=None*, *legend=True*, *figsize=(5, 3.5)*, ***kwargs*)

Generate trace plot (parameter value vs. step in Markov chain) for the `pymc.gp.cov.ExpQuad` covariance kernel lengthscale. By default, includes all chains; to plot the draws for only a subset of chains, past list of chain indices to `chains`. Use to check for posterior multimodality.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

chains: list(int) or numpy.array(int); optional
Indices of chains to plot; optional (defaults to all chains).

proxy: str, optional
Target proxy; only required if more than one proxy was included in the inference.

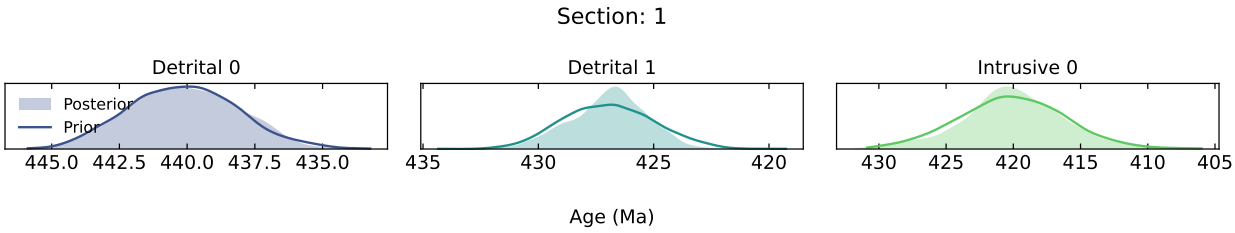
legend: bool, optional
Generate a legend. Defaults to `True`.

Returns

fig: matplotlib.pyplot.figure
Figure with lengthscale trace plot.

stratmc.plotting.**limiting_age_constraints**(*full_trace*, *sample_df*, *ages_df*, *section*, *cmap='viridis'*, ***kwargs*)

For a given section, plot prior and posterior age distributions for all intermediate limiting (i.e., detrital and intrusive ages in the middle of a section) age constraints. To plot depositional age constraints (and limiting age constraints that provide a minimum or maximum age for the entire section), use *age_constraints()*.



Parameters

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections.

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints for all sections.

section: str
Name of target section.

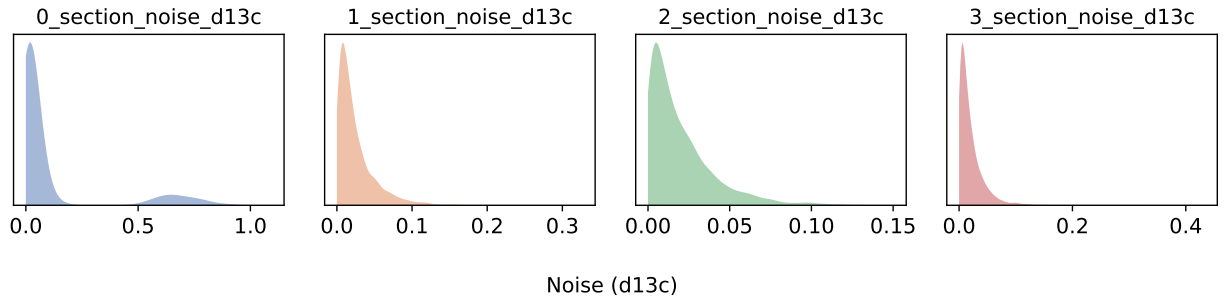
cmap: str, optional
Name of seaborn color palette to use for age distributions. Defaults to 'viridis'.

Returns

fig: matplotlib.pyplot.figure
Figure with prior and posterior limiting age constraint distributions.

stratmc.plotting.**noise_summary**(*full_trace*, ***kwargs*)

Plot posterior noise distributions for each section or group of samples (depending on `noise_type` used in *build_model()*) for a given proxy. If multiple proxies were included in the inference, pass a `proxy` argument to specify which noise terms to plot.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *get_trace* in *stratmc.inference*.

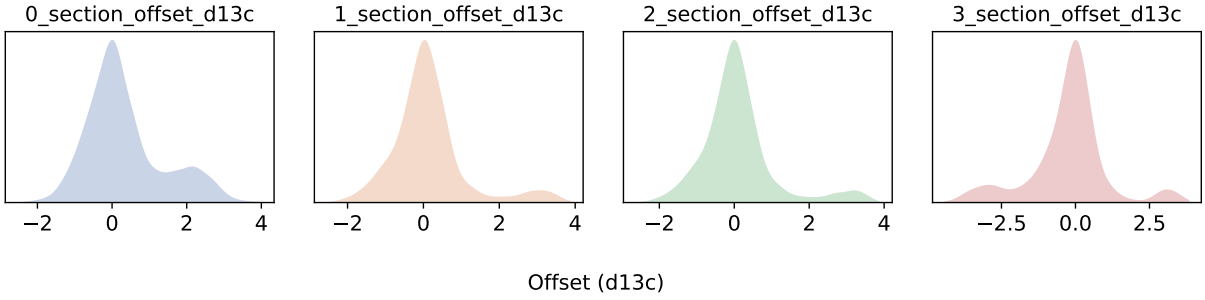
proxy: str, optional
Target proxy; only required if more than one proxy was included in the inference.

Returns

fig: matplotlib.pyplot.figure
Figure with noise distributions for each section or group of samples.

`stratmc.plotting.`**`offset_summary`**(*full_trace*, ***kwargs*)

Plot posterior offset distributions for each section or group of samples (depending on `offset_type` used in *build_model()*). If multiple proxies were included in the inference, pass a `proxy` argument to specify which offset terms to plot.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

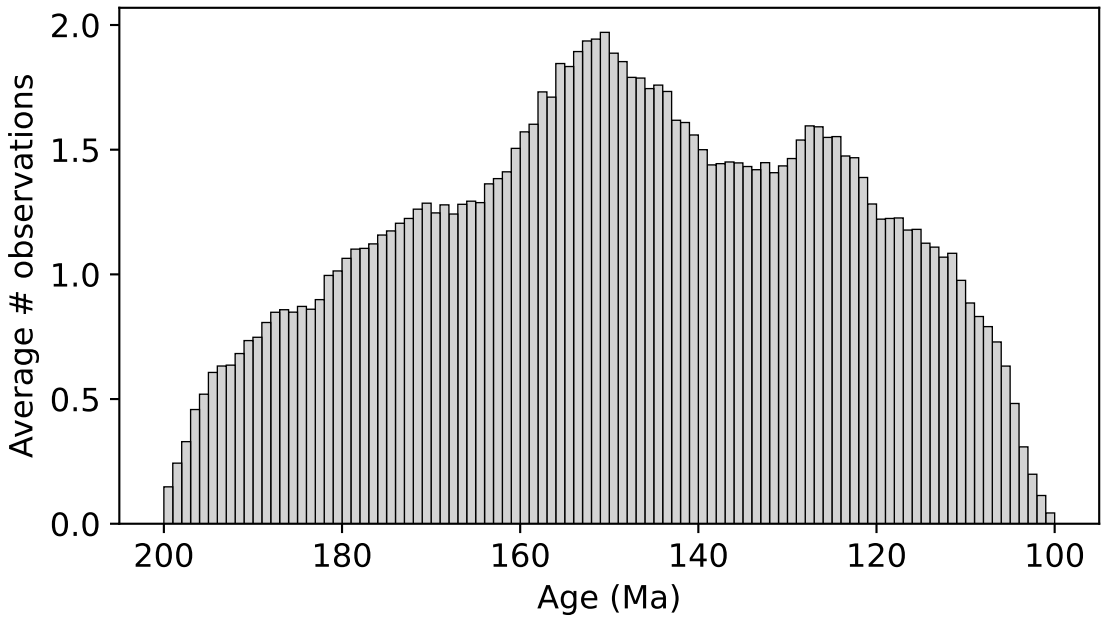
proxy: str, optional
Target proxy; only required if more than one proxy was included in the inference.

Returns

fig: matplotlib.pyplot.figure
Figure with offset distributions for each section or group of samples.

`stratmc.plotting.`**`proxy_data_density`**(*full_trace*, *time_grid=None*, *figsize=(6, 3.5)*, ***kwargs*)

Plot the mean number proxy observations in discrete time bins (averaged over all posterior draws). This plot can be used to determine where proxy observations are relatively sparse, and additional observations may improve the inference.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

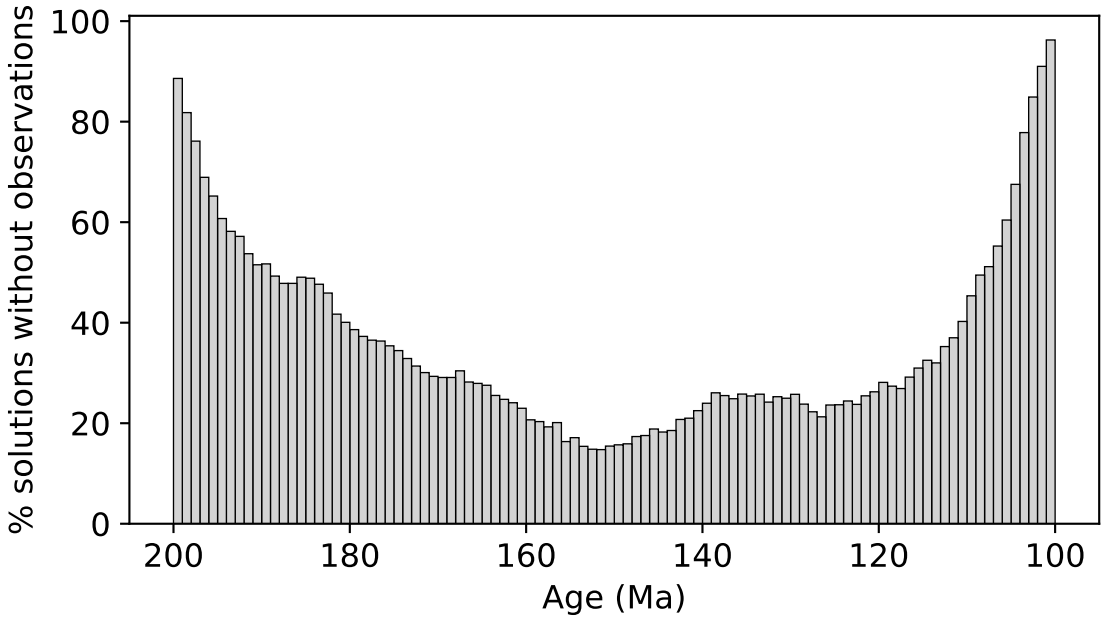
time_grid: np.array, optional
Time bin edges; if not provided, defaults to the `ages` array passed to *`get_trace()`*.

Returns

fig: matplotlib.pyplot.figure
Figure with bar plot of mean number of observations in each time bin.

`stratmc.plotting.`**`proxy_data_gaps`**(*full_trace*, *time_grid=None*, *yaxis='percentage'*, *figsize=(6, 3.5)*, ***kwargs*)

For a set of discrete time bins, shows the number of draws from the posterior where there are no proxy observations (i.e., where there are temporal *gaps* in the proxy data). This plot can be used to determine where additional observations are needed to improve the inference.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

time_grid: np.array, optional
Time bin edges; if not provided, defaults to the `ages` array passed to *`get_trace()`*.

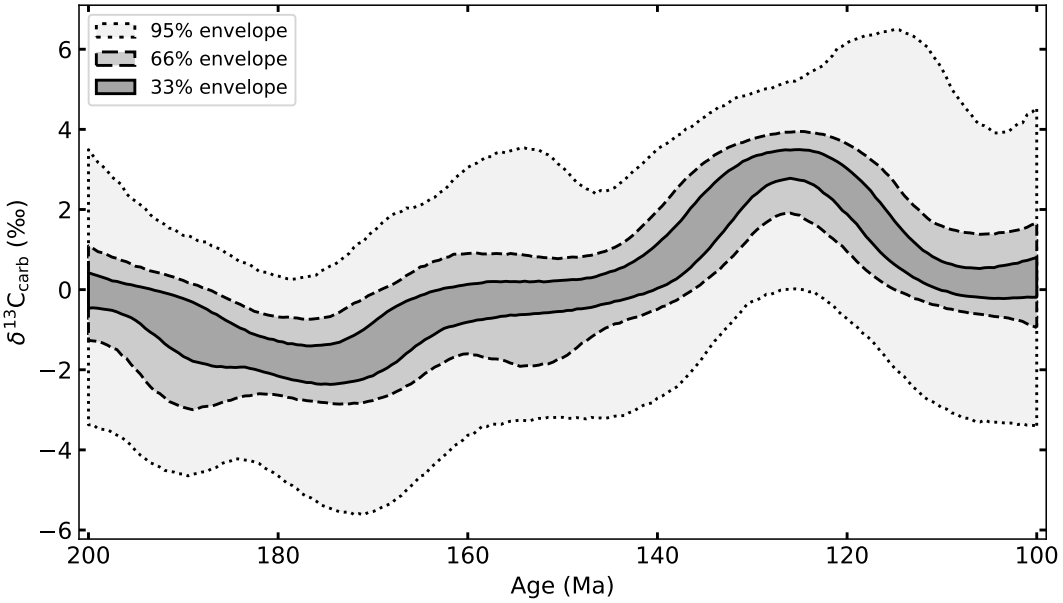
yaxis: str, optional
Set y-axis to percentage of posterior draws without observations ('percentage') or to the number of posterior draws without observations ('count`). Defaults to 'percentage`.

Returns

fig: matplotlib.pyplot.figure
Figure with bar plot of number of posterior draws with gaps for each time bin.

`stratmc.plotting.`**`proxy_inference`**(*sample_df*, *ages_df*, *full_trace*, *legend=True*, *plot_constraints=False*, *plot_samples=False*, *plot_excluded_samples=False*, *plot_mean=False*, *plot_mle=False*, *orientation='horizontal'*, *marker_size=20*, *section_legend=False*, *section_cmap='Spectral'*, ***kwargs*)

Plot the inferred proxy signal over time.



Parameters

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

proxy: str, optional
Tracer to plot; only required if more than one proxy was included in the inference.

legend: bool, optional
Generate a legend. Defaults to `True`.

plot_constraints: bool, optional
Plot age constraints for each section as dashed lines. Defaults to `False`.

plot_samples: bool, optional
Plot proxy observations by most likely posterior age. Defaults to `False`.

plot_excluded_samples: bool, optional
Plot proxy observations that were excluded from the inference (`Exclude?` is `True` in `sample_df`). Defaults to `False`.

plot_mean: bool, optional
Plot the mean as a dashed line. Defaults to `False`.

plot_mle: bool, optional
Plot the maximum likelihood estimate. Defaults to `False`.

orientation: str, optional
Orientation of figure ('horizontal' with age on the x-axis, or 'vertical' with age on the y-axis). Defaults to 'horizontal'.

marker_size: int, optional
Size of markers if `plot_samples` is `True`. Defaults to 20.

section_legend: bool, optional
Include section names in the legend (if `plot_samples` is `True`). Defaults to `False`.

section_cmap: str, optional
Name of seaborn color palette to use for sections (if `plot_samples` is `True`). Defaults to 'Spectral'.

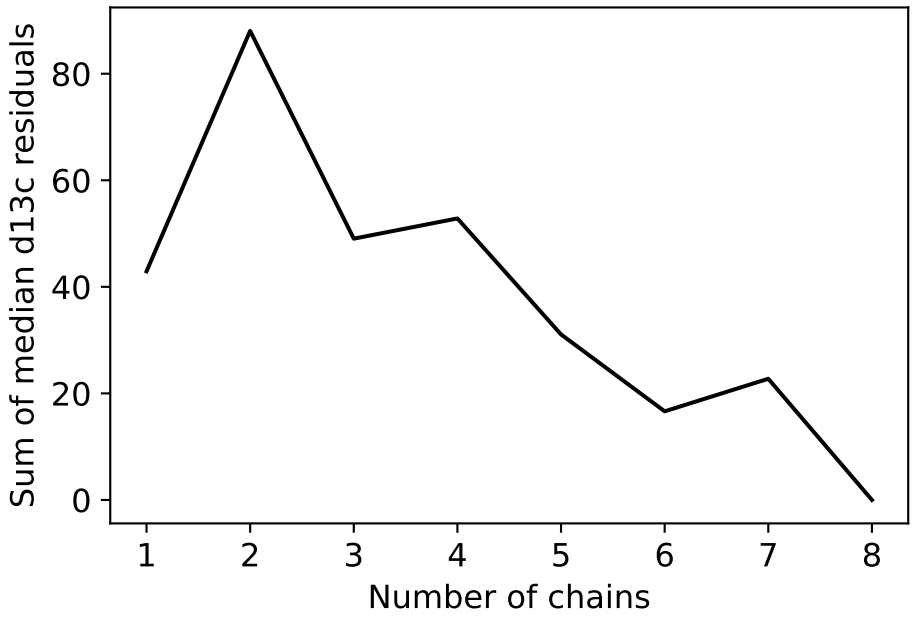
Returns

fig: matplotlib.pyplot.figure
Figure with the proxy signal inference.

`stratmc.plotting.`**`proxy_signal_stability`**(*full_trace*, *figsize=(5, 3.5)*, ***kwargs*)

Plot the sum (over all time slices) of the residuals between the median inferred proxy signal when all *N* chains are considered compared to when 1 to *N-1* chains are considered. When the posterior has been sufficiently explored, the residuals will stabilize and approach zero; if they have not stabilized, then additional chains should be run.

To consider chains from multiple traces associated with the same inference model, first combine the traces (saved as NetCDF files) using *combine_traces()* in *stratmc.data*.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

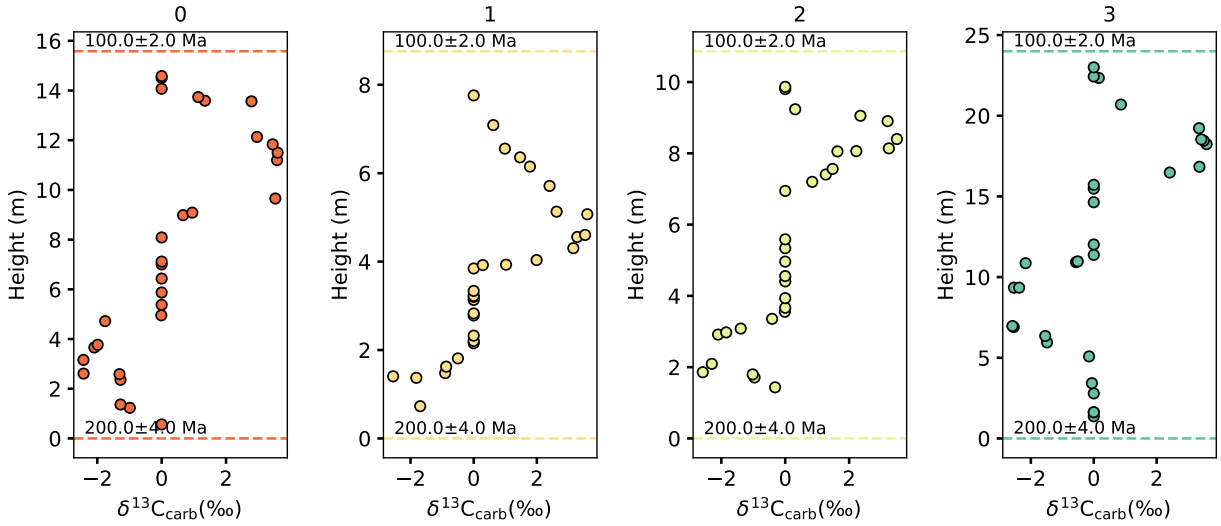
proxy: str, optional
Target proxy; only required if more than one proxy was included in the inference.

Returns

fig: matplotlib.pyplot.figure
Figure showing the stability of the proxy signal inference.

stratmc.plotting.**proxy_strat**(*sample_df*, *ages_df*, *proxy='d13c'*, *plot_constraints=True*, *plot_excluded_samples=False*, *cmap='Spectral'*, *legend=False*, ***kwargs*)

Plot stratigraphic data (proxy observations and age constraints) for each section.



Parameters

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

proxy: str, optional
Name of proxy. Defaults to 'd13c'.

sections: list(str) or numpy.array(str), optional
List of sections to plot. Defaults to all sections in `sample_df`.

plot_constraints: bool, optional
Whether to plot age constraints as dashed lines. Ages are printed above dashed lines by defalut; to turn off age labels, pass `print_ages = False`.

plot_excluded_samples: bool, optional
Whether to plot proxy observations for excluded samples (`Exclude?` is `True` in `sample_df`) as red dots. Defaults to `False`.

cmap: str, optional
Name of seaborn color palette to use for sections. Defaults to 'Spectral'.

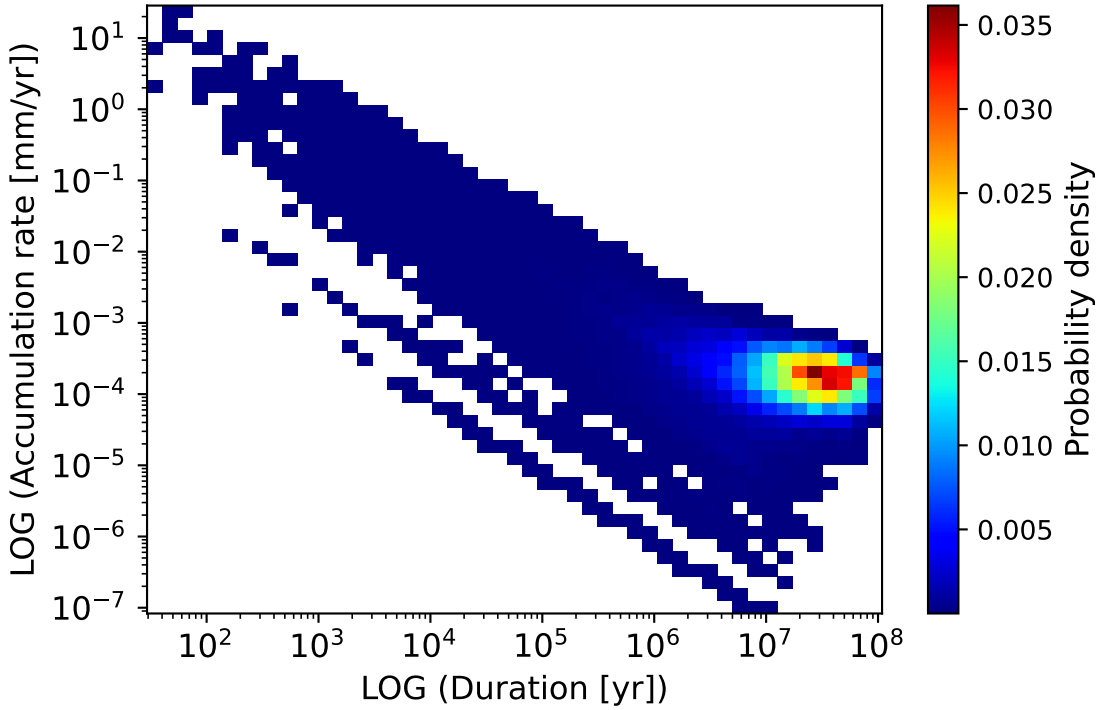
legend: bool, optional
Generate a legend. Defaults to `False`.

Returns

fig: matplotlib.pyplot.figure
Figure with observations for each section.

stratmc.plotting.**sadler_plot**(*full_trace*, *sample_df*, *ages_df*, *method='density'*, *duration_bins=50*, *rate_bins=50*, *scale='log'*, *include_age_constraints=True*, *density_cmap='jet'*, *section_cmap='Spectral'*, *figsize=(6, 4)*, ***kwargs*)

Plot accumulation rate against duration.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

method: str, optional
Plot as a 2D histogram ('density') or as a scatter plot ('scatter'). The density plot will combine data for all sections in `sections`, while a scatter plot will assign a unique color to each section. Defaults to 'density'.

duration_bins: int, optional
Number of bin edges to use for the duration data. Defaults to 50.

rate_bins: int, optional
Number of bin edges to use for the rate data. Defaults to 50.

scale: str, optional
Scaling for x- and y-axes ('linear' or 'log'). Defaults to 'log'.

sections: list(str) numpy.array(str), optional
List of target sections. Defaults to all sections in `sample_df`.

include_age_constraints: bool, optional
Include age constraints in sedimentation rate calculations. Defaults to `False`.

density_cmap: str, optional
Name of matplotlib colormap to use for probability density if `method` is 'density`. Defaults to 'jet'.

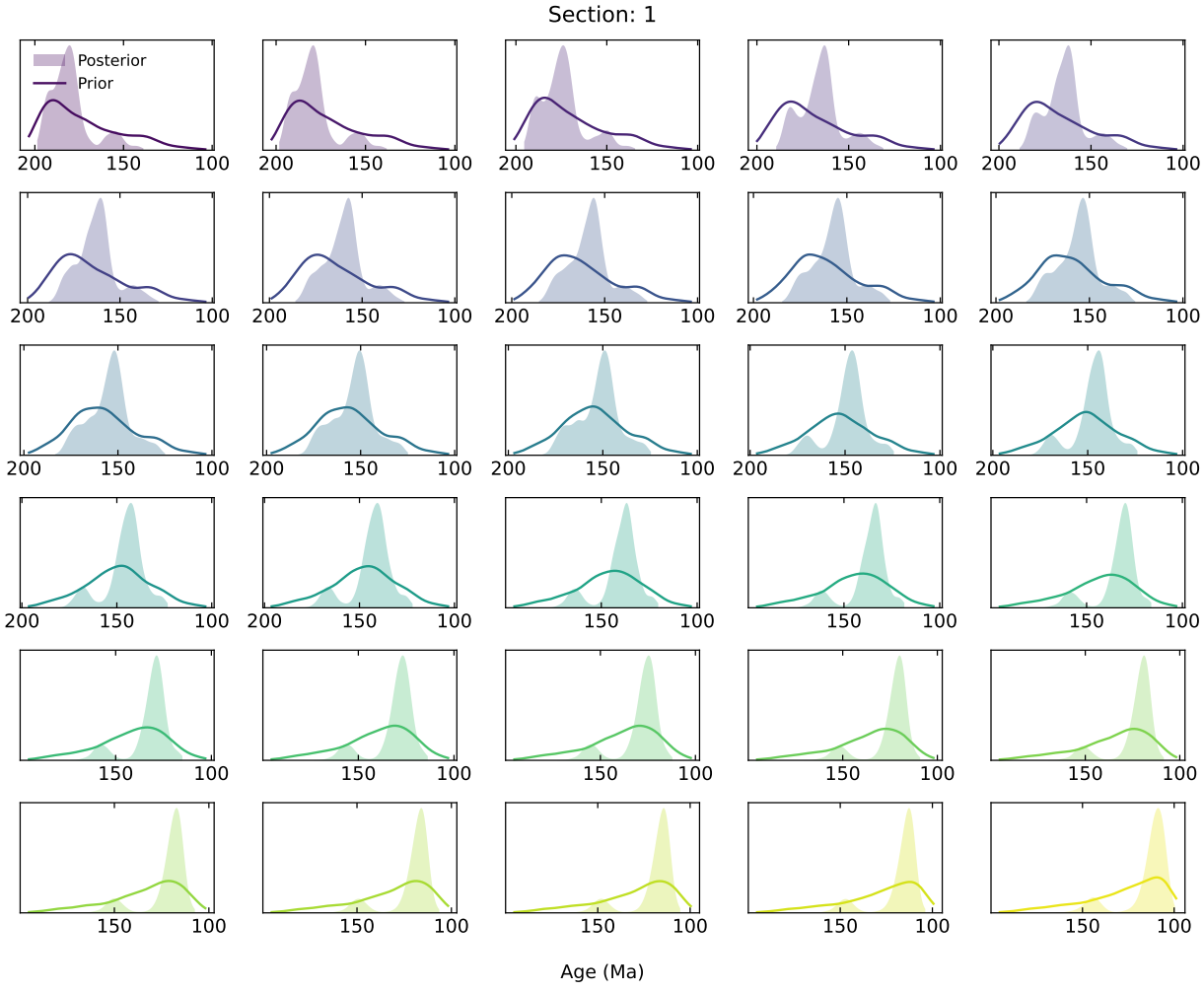
section_cmap: str, optional
Name of seaborn color palette to use for sections if `method` is 'scatter`. Defaults to 'Spectral'.

Returns

fig: matplotlib.pyplot.figure
Figure with sediment accumulation rate plotted against duration.

`stratmc.plotting.`**`sample_ages`**(*full_trace*, *sample_df*, *section*, *plot_excluded_samples=False*, *cmap='viridis'*)

Plot sample age prior and posterior distributions for a given section. Each subplot contains posterior distributions for a different sample.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples

from *get_trace()* in *stratmc.inference*.

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections.

section: str
Name of target section.

plot_excluded_samples: bool, optional
Plot age distributions for proxy observations that were excluded from the inference (`Exclude?` is `True` in `sample_df`). Defaults to `False`.

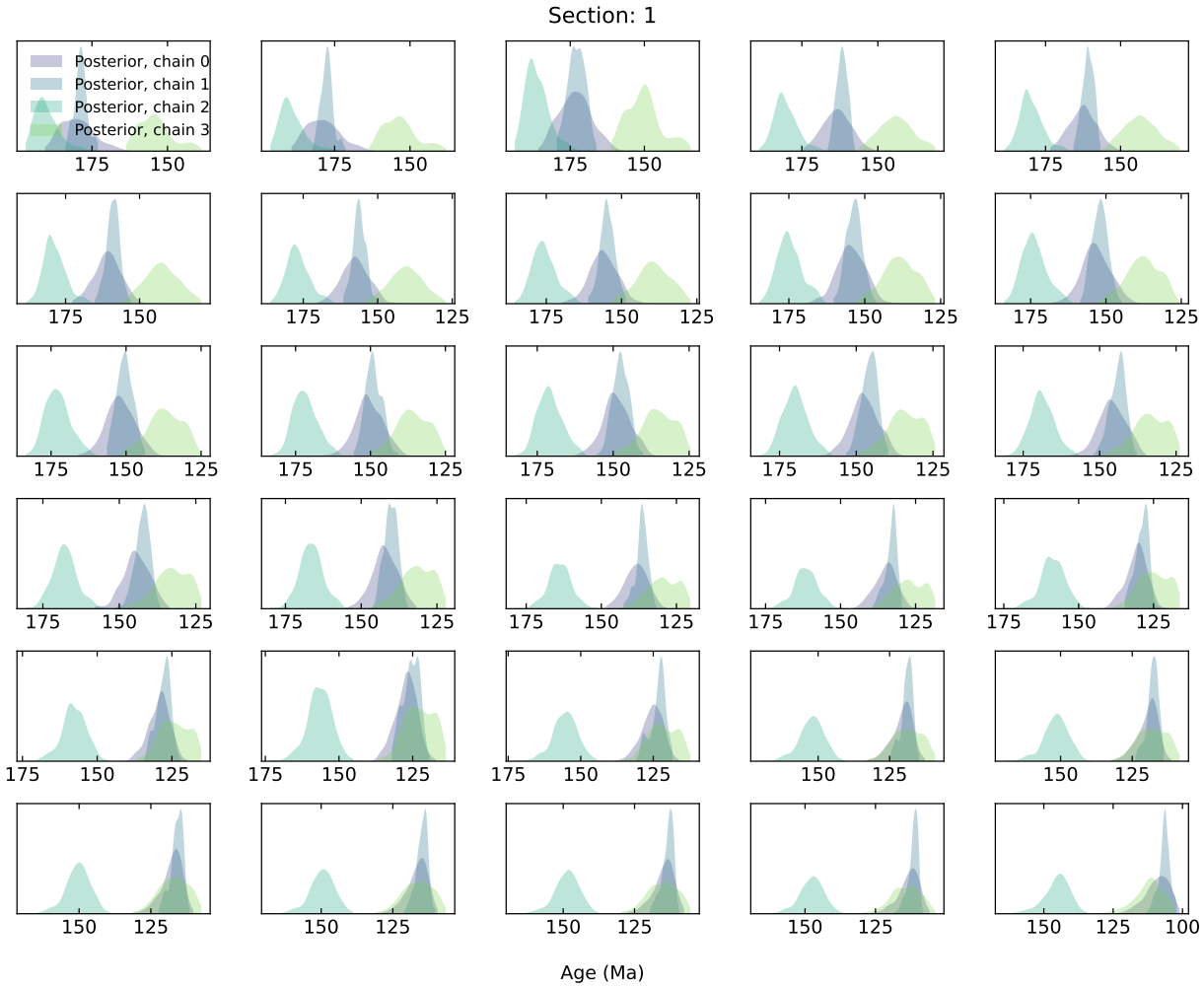
cmap: str, optional
Name of seaborn color palette to use for age distributions. Defaults to 'viridis'.

Returns

fig: matplotlib.pyplot.figure
Figure with prior and posterior sample age distributions.

stratmc.plotting.**sample_ages_per_chain**(*full_trace*, *sample_df*, *section*, *chains=None*, *plot_prior=False*, *plot_excluded_samples=False*, *legend=True*, *cmap='viridis'*)

Plot sample age posterior distributions for a given section, with separate distributions for each chain. Each subplot contains posterior distributions for a different sample. Use to check for posterior multimodality (in this example, each chain has explored a different mode of the posterior age distributions).



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

section: str
Name of target section.

chains: list(int) or numpy.array(int); optional
Indices of chains to include; optional (defaults to all chains).

plot_prior: bool, optional
Plot prior distributions for sample ages. Defaults to `False`.

plot_excluded_samples: bool, optional
Plot age distributions for proxy observations that were excluded from the inference (`Exclude?` is `True` in `sample_df`). Defaults to `False`.

legend: bool, optional
Generate a legend. Defaults to `True`.

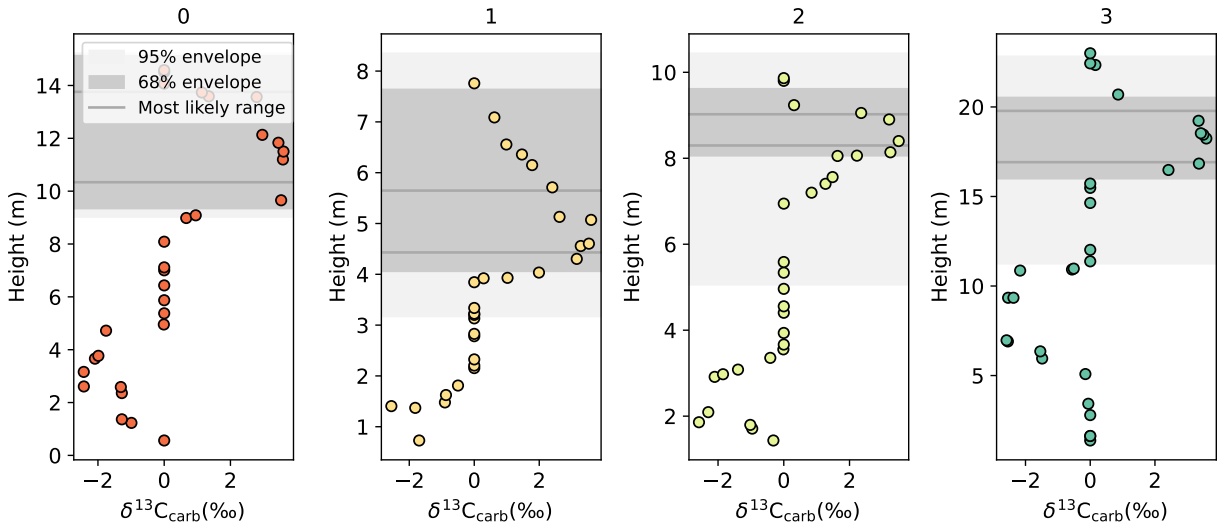
cmap: str, optional
Name of seaborn color palette to use for different chains. Defaults to 'viridis'.

Returns

fig: matplotlib.pyplot.figure
Figure with per-chain posterior sample age distributions.

`stratmc.plotting.`**`section_age_range`**(*full_trace*, *sample_df*, *ages_df*, *lower_age*, *upper_age*, *legend=True*, *section_cmap='Spectral'*, ***kwargs*)

Plot the stratigraphic interval corresponding to a given age range (based on the posterior section age models). If `sections` is not provided, includes each section that overlaps the target age range.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

lower_age: float
Lower bound (youngest) of the target age interval.

upper_age: float
Upper bound (oldest) of the target age interval.

legend: bool, optional
Generate a legend. Defaults to `True`.

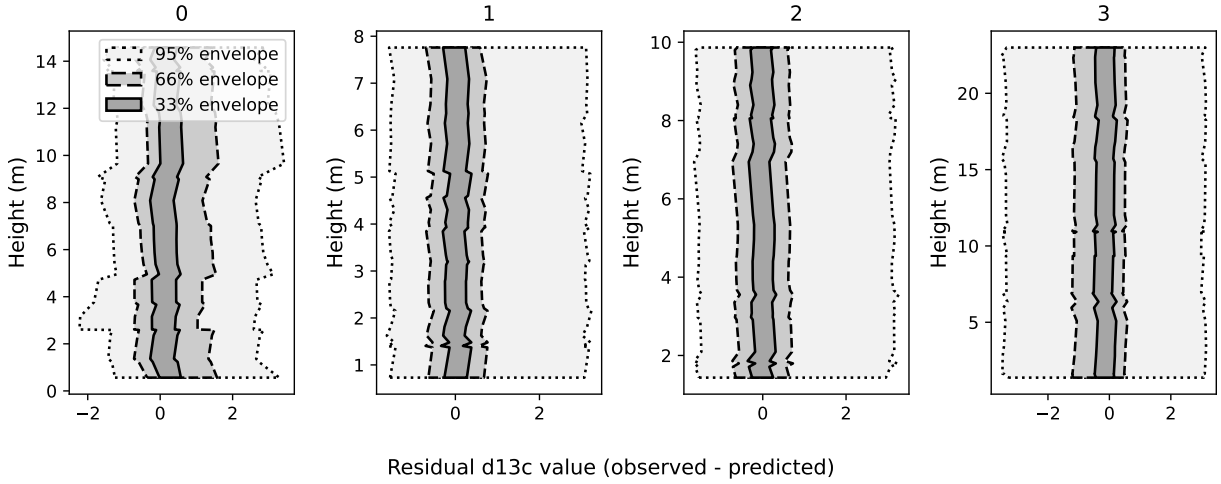
cmap: str, optional
Name of seaborn color palette to use for sections. Defaults to 'Spectral'.

Returns

fig: matplotlib.pyplot.figure
Tracer stratigraphy for each section, with the stratigraphic interval corresponding to the input age range highlighted.

`stratmc.plotting.`**`section_proxy_residuals`**(*full_trace*, *sample_df*, *legend=True*, *cmap='Spectral'*, *plot_mle=False*, *include_excluded_samples=False*, ***kwargs*)

Plot the residuals between the observed proxy values for each section and the inferred proxy signal (using the posterior section age models to map the signal back to height in section). Use to check for stratigraphic trends in the residuals, which may give insight to the processes that cause noisy sections to deviate from the inferred common signal. If multiple proxies were included in the inference, pass a `proxy` argument.



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

legend: bool, optional
Generate a legend. Defaults to `True`.

cmap: str, optional
Name of seaborn color palette to use for sections. Defaults to 'Spectral'.

plot_mle: bool, optional
Plot the maximum likelihood estimate as a line. Defaults to `False`.

proxy: str, optional
Target proxy; only required if more than one proxy was included in the inference.

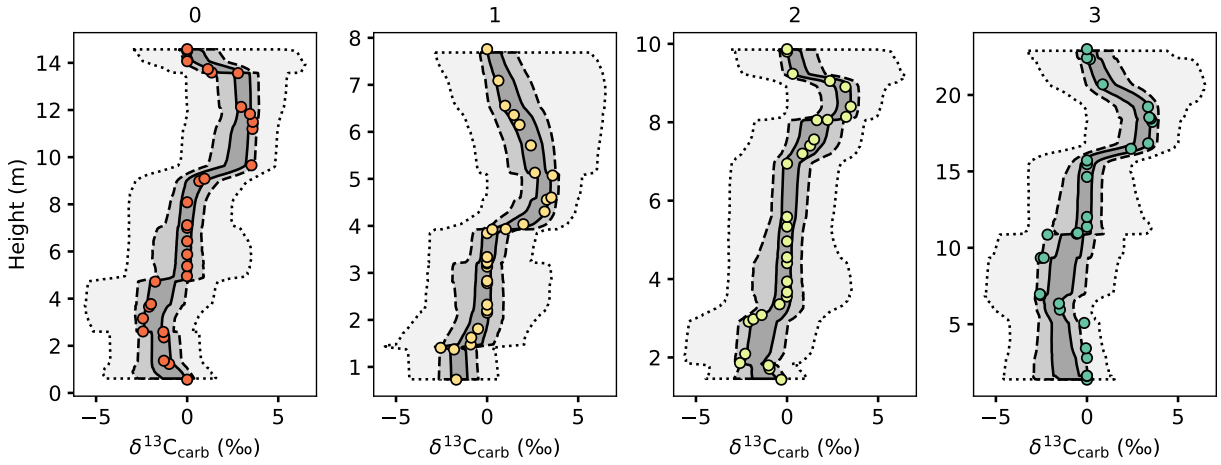
include_excluded_samples: bool, optional
Whether to plot the residuals for samples that were excluded from the inference (`Exclude?` is `True` in `sample_df`). Defaults to `False`.

Returns

fig: matplotlib.pyplot.figure
Figure with residuals for each section.

stratmc.plotting.**section_proxy_signal**(*full_trace*, *sample_df*, *ages_df*, *include_radiometric_ages=False*, *plot_constraints=False*, *plot_mle=False*, *yax='height'*, *legend=False*, *cmap='Spectral'*, ***kwargs*)

Map the posterior proxy signal back to height in each section (using its most likely posterior age model), and plot alongside the proxy observations (plotted by most likely posterior age).



Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

include_radiometric_ages: bool, optional
Whether to consider radiometric ages in the posterior age model for each section. Defaults to `False`.

plot_constraints: bool, optional
Plot age constraints for each section as dashed lines. Defaults to `False`.

plot_mle: bool, optional
Plot the maximum likelihood estimate for the proxy signal as a line. Defaults to `False`.

yax: str, optional
Scale for the y-axis ('height' or 'age'). Defaults to 'height'.

legend: bool, optional
Generate a legend. Defaults to `True`.

cmap: str, optional
Name of seaborn color palette to use for sections. Defaults to 'Spectral'.

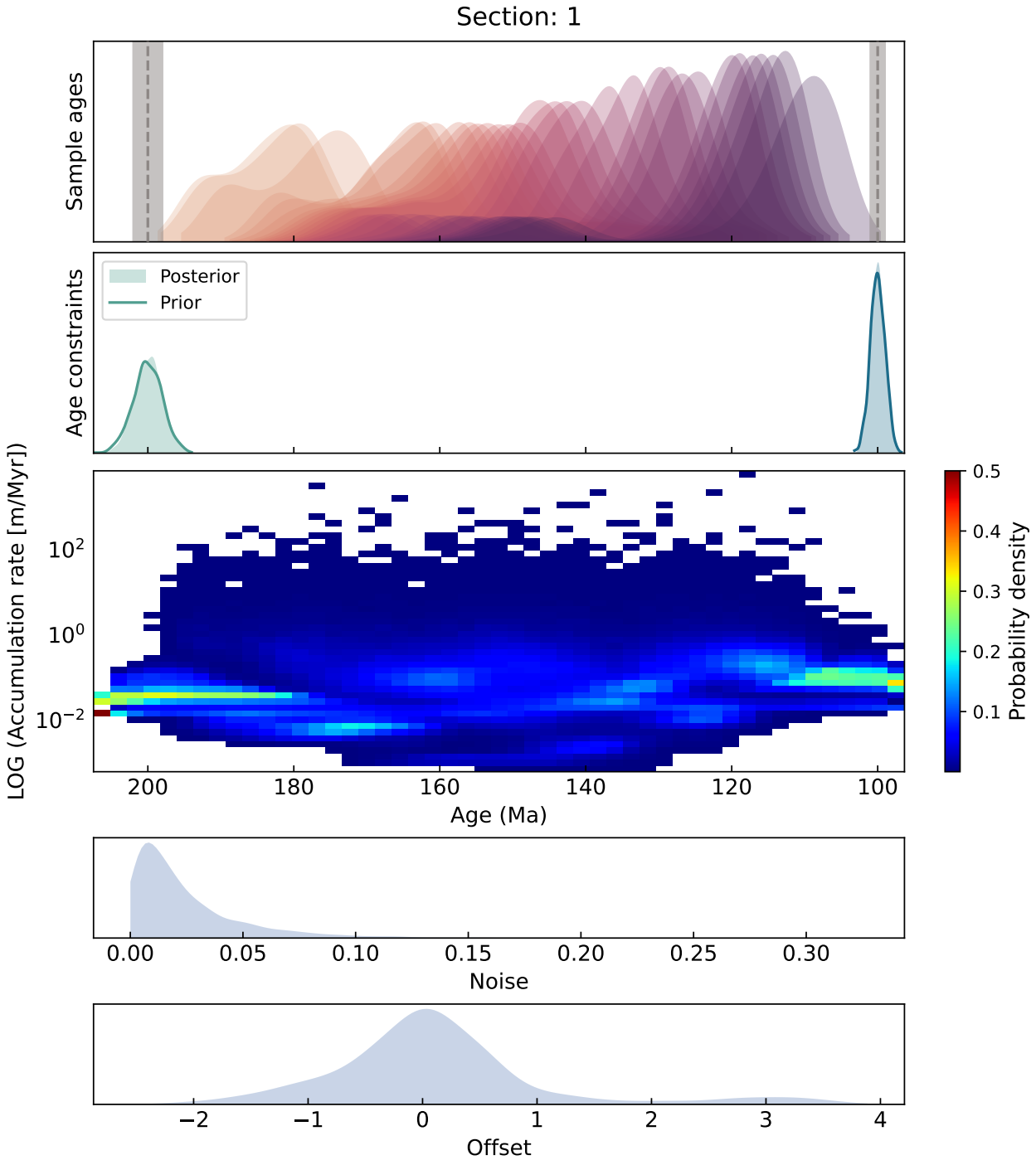
proxy: str, optional
Tracer to plot; only required if more than one proxy was included in the inference.

Returns

fig: matplotlib.pyplot.figure
Figure with inferred proxy signal mapped to height in each section.

`stratmc.plotting.`**`section_summary`**(*sample_df*, *ages_df*, *full_trace*, *section*, *plot_excluded_samples=False*, *plot_noise_prior=False*, *plot_offset_prior=False*, *include_age_constraints_sedrate=True*, *figsize=(8, 9)*)

For a given section, plot posterior estimates of sample age, sedimentation rate, noise, and offset. Noise and offset terms must be either per-section or global; to plot per-sample noise and offset terms, use *noise_summary()* and *offset_summary()*.



Parameters

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections.

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints for all sections.

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

section: str
Name of target section.

plot_excluded_samples: bool, optional
Plot age estimates for proxy observations that were excluded from the inference (`Exclude?` is `True` in `sample_df`). Defaults to `False`.

plot_noise_prior: bool, optional
Plot prior distribution for noise term. Defaults to `False`.

plot_offset_prior: bool, optional
Plot prior distribution for offset term. Defaults to `False`.

include_age_constraints_sedrate: bool, optional
Include age constraints in sedimentation rate calculations. Defaults to `True`.

Returns

fig: matplotlib.pyplot.figure
Figure summarizing posterior sample ages, sedimentation rate, and posterior noise and offset terms for the input section.

4.5 Tools for generating synthetic data

Functions for creating synthetic proxy signals/stratigraphic observations and evaluating model performance for synthetic tests.

make_excursion	Function for generating a synthetic proxy signal that contains a number of user-specified excursions.
synthetic_sections	Function for generating synthetic proxy observations and age constraints using a predefined proxy signal.
synthetic_observations_from_prior	Given age constraints for a set of stratigraphic sections in `ages_df`, generate synthetic proxy observations by sampling the model prior.
synthetic_signal_from_prior	Draws synthetic signals from the model prior, and returns the signal conditioned over the points in `ages`.
quantify_signal_recovery	Calculates the likelihood of the true proxy signal (for synthetic tests, where the true signal is known) conditioned on the posterior (default) or prior proxy signal inference.
sample_age_recovery	Calculates the likelihood of the true sample ages (for synthetic tests, where the true age of each sample is known) given draws from the posterior (default) or prior.
sample_age_residuals	Calculates the residual (for each draw) between the true age and the posterior (default) or prior age of each sample.
synthetic_signal_to_df	Helper function for generating artificial sample and age data using *synthetic_sections()*.

stratmc.synthetics.**make_excursion**(*time*, *amplitude*, *baseline=0*, *rising_time=None*, *rate_offset=True*, *excursion_duration=None*, *min_duration=1*, *smooth=False*, *smoothing_factor=10*, *seed=None*)

Function for generating a synthetic proxy signal that contains a number of user-specified excursions.

Parameters

time: numpy.array(float)
Time vector over which to generate proxy signal.

amplitude: float, list(float), or numpy.array(float)
Amplitude of excursion; pass a list or array to generate multiple excursions.

baseline: float, optional
Baseline proxy value. Defaults to 0.

rising_time: float, list(float), or numpy.array(float), optional
Fraction of excursion duration spent on the rising limb (linear increase/decrease toward peak). Must be between 0 and 1. If not provided, randomly generated if `rate_offset` is `True` and set to 0.5 if `rate_offset` is `False`. Pass a list to specify different rising times for each excursion.

rate_offset: bool, optional
If `False`, rising and falling limbs of excursion have equal duration. If `True`, the fraction of the excursion duration spent on the rising limb is set by `rising_time`. Defaults to `False`.

excursion_duration: float, list(float), or numpy.array(float), optional
Duration of excursion; pass a list or array to generate multiple excursions. Random if not provided.

min_duration: float, optional
Minimum excursion duration if `excursion_duration` is not provided. Defaults to 1.

smooth: bool, optional
Whether to smooth excursion peaks. Defaults to `False`.

smoothing_factor: float, optional
Smoothing factor if `smooth` is `True`; higher values produce smoother signals. Defaults to 10.

seed: int, optional
Random seed used to generate signal.

Returns

interp_proxy: np.array
Tracer signal interpolated to points in the `time` vector

`stratmc.synthetics.`**`quantify_signal_recovery`**(*full_trace*, *true_signal*, *proxy='d13c'*, *mode='posterior'*)

Calculates the likelihood of the true proxy signal (for synthetic tests, where the true signal is known) conditioned on the posterior (default) or prior proxy signal inference. The likelihood is evaluated at each age (the posterior signal and the true signal must be evaluated at the same ages). Provides a measure of signal recovery.

Parameters

full_trace: arviz.InferenceData or list(arviz.InferenceData)
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*. If passed as a list, the posterior draws for all traces will be combined when calculating *posterior_likelihood*.

true_signal: np.array
True values for the proxy signal, evaluated at the same ages as the posterior signal in `full_trace`.

proxy: str, optional
Tracer signal to evaluate. Defaults to 'd13c'.

mode: str, optional
Whether to use the posterior or prior to calculate signal recovery. Defaults to 'posterior'.

Returns

posterior_likelihood: np.array
Array of posterior likelihoods (evaluated at each age).

`stratmc.synthetics.`**`sample_age_recovery`**(*full_trace*, *sample_df*, *sections=None*, *mode='posterior'*)

Calculates the likelihood of the true sample ages (for synthetic tests, where the true age of each sample is known) given draws from the posterior (default) or prior. Provides a measure of age model recovery.

Parameters

full_trace: arviz.InferenceData or list(arviz.InferenceData)
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*. If passed as a list, the posterior draws for all traces will be combined when calculating *posterior_likelihood*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for synthetic sections.

sections: list(str) or numpy.array(str), optional
List of sections to evaluate. Defaults to all sections in sample_df.

mode: str, optional
Whether to use the posterior or prior age models. Defaults to 'posterior'.

Returns

posterior_likelihood: dict{float} or np.array(float)
Posterior likelihoods for the true age of each sample. Returned as an array if only one section is evaluated, or a dictionary of arrays if multiple sections are evaluated.

stratmc.synthetics.**sample_age_residuals**(*full_trace*, *sample_df*, *sections=None*, *mode='posterior'*)

Calculates the residual (for each draw) between the true age and the posterior (default) or prior age of each sample.

Parameters

full_trace: arviz.InferenceData or list(arviz.InferenceData)
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*. If passed as a list, the posterior draws for all traces will be combined when calculating *age_residuals*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for synthetic sections.

sections: list(str) or numpy.array(str), optional
List of sections to evaluate. Defaults to all sections in sample_df.

mode: str, optional
Whether to use the posterior or prior age models. Defaults to 'posterior'.

Returns

age_residuals: np.array or dict{np.array}
Sample age residuals; shape is (number of samples, number of posterior draws). Returned as an array if only one section is evaluated, or a dictionary of arrays if multiple sections are evaluated.

stratmc.synthetics.**synthetic_observations_from_prior**(*age_vector*, *ages_df*, *sample_heights=None*, *uniform_heights=False*, *samples_per_section=20*, *proxies=['d13c']*, *proxy_std=0.1*, *seed=None*, *ls_dist='Wald'*, *ls_min=0*, *ls_mu=20*, *ls_lambda=50*, *ls_sigma=50*, *var_sigma=10*, *white_noise_sigma=0.1*, *gp_mean_mu=0*, *gp_mean_sigma=10*, *approximate=False*, *hsgp_m=15*, *hsgp_c=1.3*, *offset_type='section'*, *offset_prior='Laplace'*, *offset_alpha=0*, *offset_beta=1*, *offset_sigma=1*, *offset_mu=0*, *offset_b=2*, *noise_type='section'*, *noise_prior='HalfCauchy'*, *noise_beta=1*, *noise_sigma=1*, *noise_nu=1*, *jitter=0.001*, ***kwargs*)

Given age constraints for a set of stratigraphic sections in `ages_df`, generate synthetic proxy observations by sampling the model prior. Accepts all arguments that can be passed to *build_model()* in *stratmc.model*.

Parameters

age_vector: np.array(float)
Vector of ages at which to evaluate synthetic proxy signal(s).

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for synthetic sections.

sample_heights: dict{list(float) or numpy.array(float)}, optional
Sample heights for each stratigraphic section in `ages_df`; must be a dictionary with section

names as keys. Defaults to `None`, which results in either uniformly spaced or randomly spaced sample heights (depending on the `uniform_heights` argument).

uniform_heights: bool, optional
Whether to generate uniformly spaced (set to `True`) or randomly spaced (set to `False`) sample heights if dictionary of `sample_heights` not provided. Defaults to `False` (randomly spaced samples).

samples_per_section: int or dict(int), optional
Number of samples per section to generate if `sample_heights` not provided; either an integer (if the same for all sections) or a dictionary with section names as keys. Defaults to 20.

proxies: list(str), optional
List of proxies to generate synthetic observations for. Defaults to *d13c*.

proxy_std: float or dict(float), optional
Measurement uncertainty for each proxy; pass a dictionary of floats with the elements of `proxies` as keys to use a different value for each proxy, or an integer to use the same value for all proxies. Defaults to 0.1.

seed: int, optional
Seed to use while generating synthetic observations.

Returns

signals: dict(float)
Tracers signals drawn from the model prior (evaluated at the points in `age_vector`) used to generate synthetic observations; dictionary keys are `proxies`.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for synthetic stratigraphic sections.

prior: arviz.InferenceData
An `arviz.InferenceData` object containing the prior draw from the model used to generate synthetic observations.

model: pymc.Model
`pymc.model.core.Model` object used to generate synthetic observations.

stratmc.synthetics.**synthetic_sections**(*true_time*, *true_proxy*, *num_sections*, *num_samples*, *max_section_thickness*, *proxies=['d13c']*, *noise=False*, *noise_amp=0.1*, *min_constraints=2*, *max_constraints=3*, *seed=None*, ***kwargs*)

Function for generating synthetic proxy observations and age constraints using a predefined proxy signal.

Parameters

true_time: numpy.array(float)
True time vector for input signal.

true_proxy: numpy.array(float) or dict{numpy.array(float)}
True proxy vector for input signal. If generating synthetic data for multiple proxies, pass as a dictionary with proxy names as keys.

num_sections: int
Number of synthetic sections to generate.

num_samples: int
Number of samples per synthetic section.

max_section_thickness: float
Maximum thickness of synthetic sections.

proxies: str or list(str), optional
Column name(s) for synthetic proxy observations in `sample_df`. Defaults to 'd13c'.

noise: bool, optional
Whether to add white noise to proxy observations. Defaults to `False`.

noise_amp: float or dict{float}, optional
Amplitude of white noise added to proxy observations (if `noise` is `True`). To specify a different noise amplitude for each proxy, pass as a dictionary with proxy names as keys. Defaults to 0.1.

min_constraints: int, optional
Minimum number of age constraints per synthetic section (must be at least 2). Defaults to 2.

max_constraints: int, optional
Maximum number of age constraints per synthetic section. Defaults to 3.

seed: int, optional
Random seed used to generate synthetic sections.

Returns

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for synthetic sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for synthetic sections.

stratmc.synthetics.**synthetic_signal_from_prior**(*ages*, *num_signals=100*, *ls_dist='Wald'*, *ls_min=0*, *ls_mu=20*, *ls_lambda=50*, *ls_sigma=50*, *var_sigma=10*, *gp_mean_mu=0*, *gp_mean_sigma=5*, *seed=None*)

Draws synthetic signals from the model prior, and returns the signal conditioned over the points in `ages`. To generate both signals and synthetic stratigraphic sections, instead use *synthetic_observations_from_prior()*.

Parameters

ages: numpy.array(float)
Array of ages over which to condition the signal.

num_signals: int, optional
Number of signals to draw from prior. Defaults to 100.

ls_dist: str, optional
Prior distribution for the lengthscale hyperparameter of the exponential quadratic covariance kernel (`pymc.gp.cov.ExpQuad`); set to `Wald` (`pymc.Wald`) or `HalfNormal` (`pymc.HalfNormal`). Defaults to `Wald` with `mu = 20` and `lambda = 50`; to change `mu` and `lambda`, pass the `ls_mu` and `ls_lambda` parameters. For `HalfNormal`, the variance defaults to `sigma = 50`; change by passing `ls_sigma`.

ls_min: float, optional
Minimum value for the lengthscale hyperparameter of the `pymc.gp.cov.ExpQuad` covariance kernel; shifts the lengthscale prior by `ls_min`. Defaults to 0.

ls_mu: float, optional
Mean (*mu*) of the `pymc.gp.cov.ExpQuad` lengthscale prior if `ls_dist = `Wald``. Defaults to 20.

ls_lambda: float, optional
Relative precision (*lam*) of the `pymc.gp.cov.ExpQuad` lengthscale hyperparameter prior if `ls_dist = `Wald``. Defaults to 50.

ls_sigma: float, optional
Scale parameter (*sigma*) of the `pymc.gp.cov.ExpQuad` lengthscale hyperparameter prior if `ls_dist = `HalfNormal``. Defaults to 50.

var_sigma: float, optional
Scale parameter (*sigma') of the covariance kernel variance hyperparameter prior, which is a :class:`pymc.HalfNormal* distribution. Defaults to 10.

gp_mean_mu: float, optional
Mean (*mu*) of the GP mean function prior, which is a `pymc.Normal` distribution. Defaults to 0.

gp_mean_sigma: float, optional
Standard deviation (*sigma*) of the GP mean function prior, which is a `pymc.Normal` distribution. Defaults to 5.

seed: int, optional
Random seed used to generate signals.

Returns

signal: numpy.ndarray(float)
Array with shape `ages x number of signals` containing the `n = num_signals` synthetic signals drawn from the prior.

stratmc.synthetics.**synthetic_signal_to_df**(*proxy_vec*, *heights*, *section_ages*, *section_names*, *ages*, *age_std*, *age_heights*, *age_section_names*, *proxies=['d13c']*)

Helper function for generating artificial sample and age data using *synthetic_sections()*.

Parameters

proxy_vec: np.array(float) or dict{np.array(float)}
Array of proxy observations. Pass as a dictionary if more than one proxy.

heights: np.array(float)
Array of heights corresponding to proxy observations in `proxy_vec`.

section_ages: np.array(float)
Array of ages corresponding to proxy observations in `proxy_vec`.

section_names: np.array(str)
Array of section names corresponding to proxy observations in `proxy_vec`.

ages: np.array(float)
Array of age constraints.

age_std: np.array(float)
Array of uncertainties for each age constraint in `ages`.

age_heights: np.array(float)
Array of heights for each age constraint in `ages`.

age_section_names: np.array(str)
Array of section names corresponding to age constraints in `ages`.

proxies: str or list(str), optional
Name(s) of proxies. Defaults to *d13c*.

Returns

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for synthetic sections.

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints for synthetic sections.

4.6 Tests and checks

Functions for checking that the inference model is working correctly.

check_inference	Master function (calls each of the functions in the tests module) for checking that superposition is never violated in the posterior.
check_superposition	Check that stratigraphic superposition between all age constriants and samples is respected in the posterior.
check_detrital_ages	Check that detrital age constraints have been enforced in the posterior.
check_intrusive_ages	Check that intrusive age constraints have been enforced in the posterior.

stratmc.tests.**check_detrital_ages**(*full_trace*, *sample_df*, *ages_df*, *quiet=True*, ***kwargs*)

Check that detrital age constraints have been enforced in the posterior.

Parameters

full_trace: arviz.InferenceData
An arviz.InferenceData object containing the full set of prior and posterior samples from *get_trace()* in *stratmc.inference*.

sample_df: pandas.DataFrame
pandas.DataFrame containing proxy data for all sections.

ages_df: pandas.DataFrame
pandas.DataFrame containing age constraints for all sections.

sections: list(str) or numpy.array(str), optional
List of sections included in the inference. Defaults to all sections in sample_df.

quiet: bool, optional
Whether to print the section name and chain/draw of each superposition violation; defaults to False.

Returns

bad_chains: numpy.array
Array of chain indices where superposition was violated in the posterior.

stratmc.tests.**check_inference**(*full_trace*, *sample_df*, *ages_df*, *quiet=True*, ***kwargs*)

Master function (calls each of the functions in the tests module) for checking that superposition is never violated in the posterior. Returns a list of chain indices where superposition was violated; these chains can be dropped from the trace using *drop_chains()*. Run automatically inside of *get_trace()* in *stratmc.inference*.

Because of the likelihood penalty used to manually enforce detrital and intrusive ages in *intermediate_detrital_potential()* and *intermediate_intrusive_potential()* (called in *build_model()*), rare chains may have minor superposition violations when deterital/intrusive ages are present. These chains can simply be discarded. If superposition is frequently violated in a given section, or if

superposition violations are severe, check that the heights for all age constraints in `ages_df` are correct, and that the reported ages respect superposition. The model can correct for mean ages that are out of superposition, but may fail if the age constraints do not overlap given their 2$sigma$ uncertainties.

Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

sections: list(str) or numpy.array(str), optional
List of sections included in the inference. Defaults to all sections in `sample_df`.

quiet: bool, optional
Whether to print the type, section name, and chain/draw of each superposition violation; defaults to `False`.

Returns

bad_chains: numpy.array
Array of chain indices where superposition was violated in the posterior.

`stratmc.tests.`**`check_intrusive_ages`**(*full_trace*, *sample_df*, *ages_df*, *quiet=True*, ***kwargs*)

Check that intrusive age constraints have been enforced in the posterior.

Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

sections: list(str) or numpy.array(str), optional
List of sections included in the inference. Defaults to all sections in `sample_df`.

quiet: bool, optional
Whether to print the section name and chain/draw of each superposition violation; defaults to `False`.

Returns

bad_chains: numpy.array
Array of chain indices where superposition was violated in the posterior.

`stratmc.tests.`**`check_superposition`**(*full_trace*, *sample_df*, *ages_df*, *quiet=True*, ***kwargs*)

Check that stratigraphic superposition between all age constriants and samples is respected in the posterior.

Parameters

full_trace: arviz.InferenceData
An `arviz.InferenceData` object containing the full set of prior and posterior samples from *`get_trace()`* in *`stratmc.inference`*.

sample_df: pandas.DataFrame
`pandas.DataFrame` containing proxy data for all sections.

ages_df: pandas.DataFrame
`pandas.DataFrame` containing age constraints for all sections.

sections: list(str) or numpy.array(str), optional
List of sections included in the inference. Defaults to all sections in `sample_df`.

quiet: bool, optional
Whether to print the section name and chain/draw of each superposition violation; defaults to `False`.

Returns

bad_chains: numpy.array
Array of chain indices where superposition was violated in the posterior.

4.7 Indices and Tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

stratmc.data, 9
stratmc.inference, 20
stratmc.model, 14
stratmc.plotting, 28
stratmc.synthetics, 50
stratmc.tests, 56

INDEX

A

accumulation_rate() (*in module stratmc.data*), 9
accumulation_rate_stratigraphy() (*in module stratmc.plotting*), 28
age_constraints() (*in module stratmc.plotting*), 29
age_height_model() (*in module stratmc.plotting*), 29
age_range_to_height() (*in module stratmc.inference*), 20

B

build_model() (*in module stratmc.model*), 14

C

calculate_lengthscale_stability() (*in module stratmc.inference*), 21
calculate_proxy_signal_stability() (*in module stratmc.inference*), 21
check_detrital_ages() (*in module stratmc.tests*), 56
check_inference() (*in module stratmc.tests*), 56
check_intrusive_ages() (*in module stratmc.tests*), 57
check_superposition() (*in module stratmc.tests*), 57
clean_data() (*in module stratmc.data*), 10
combine_data() (*in module stratmc.data*), 11
combine_duplicates() (*in module stratmc.data*), 11
combine_traces() (*in module stratmc.data*), 11
count_samples() (*in module stratmc.inference*), 21
covariance_hyperparameters() (*in module stratmc.plotting*), 30

D

depth_to_height() (*in module stratmc.data*), 11
drop_chains() (*in module stratmc.data*), 12

E

extend_age_model() (*in module stratmc.inference*), 22

F

find_gaps() (*in module stratmc.inference*), 23

G

get_trace() (*in module stratmc.inference*), 23

I

intermediate_detrital_potential() (*in module stratmc.model*), 17
intermediate_intrusive_potential() (*in module stratmc.model*), 18
interpolate_proxy() (*in module stratmc.inference*), 25
interpolated_proxy_inference() (*in module stratmc.plotting*), 31

L

lengthscale_stability() (*in module stratmc.plotting*), 32
lengthscale_traceplot() (*in module stratmc.plotting*), 33
limiting_age_constraints() (*in module stratmc.plotting*), 34
load_data() (*in module stratmc.data*), 12
load_object() (*in module stratmc.data*), 13
load_trace() (*in module stratmc.data*), 13

M

make_excursion() (*in module stratmc.synthetics*), 50
map_ages_to_section() (*in module stratmc.inference*), 25
module
    stratmc.data, 9
    stratmc.inference, 20
    stratmc.model, 14
    stratmc.plotting, 28
    stratmc.synthetics, 50
    stratmc.tests, 56

N

noise_summary() (*in module stratmc.plotting*), 34

O

offset_summary() (*in module stratmc.plotting*), 35

P

proxy_data_density() (*in module stratmc.plotting*), 35

proxy_data_gaps() (*in module stratmc.plotting*), 36
proxy_inference() (*in module stratmc.plotting*), 37
proxy_signal_stability() (*in module stratmc.plotting*), 39
proxy_strat() (*in module stratmc.plotting*), 40

Q

quantify_signal_recovery() (*in module stratmc.synthetics*), 51

S

sadler_plot() (*in module stratmc.plotting*), 40
sample_age_recovery() (*in module stratmc.synthetics*), 51
sample_age_residuals() (*in module stratmc.synthetics*), 52
sample_ages() (*in module stratmc.plotting*), 42
sample_ages_per_chain() (*in module stratmc.plotting*), 43
save_object() (*in module stratmc.data*), 13
save_trace() (*in module stratmc.data*), 13
section_age_range() (*in module stratmc.plotting*), 45
section_proxy_residuals() (*in module stratmc.plotting*), 46
section_proxy_signal() (*in module stratmc.plotting*), 46
section_summary() (*in module stratmc.plotting*), 48
stratmc.data
    module, 9
stratmc.inference
    module, 20
stratmc.model
    module, 14
stratmc.plotting
    module, 28
stratmc.synthetics
    module, 50
stratmc.tests
    module, 56
superposition() (*in module stratmc.model*), 19
synthetic_observations_from_prior() (*in module stratmc.synthetics*), 52
synthetic_sections() (*in module stratmc.synthetics*), 53
synthetic_signal_from_prior() (*in module stratmc.synthetics*), 54
synthetic_signal_to_df() (*in module stratmc.synthetics*), 55

T

thin_trace() (*in module stratmc.data*), 13
transformed_initval() (*in module stratmc.model*), 19