

---

# **openest Documentation**

***Release***

**Author**

**Jan 30, 2018**



---

## Contents

---

<b>1</b>	<b>Intro and Setup</b>	<b>3</b>
1.1	Installation . . . . .	3
<b>2</b>	<b>The API Documentation / Guide</b>	<b>5</b>
2.1	openest package . . . . .	5
<b>3</b>	<b>The Developers Guide</b>	<b>31</b>
3.1	Contributor's Guide . . . . .	31
<b>4</b>	<b>Indices and tables</b>	<b>39</b>
	<b>Python Module Index</b>	<b>41</b>



OpenEst is a library created by the [Climate Impact Lab](#) team.

This code is open source and available on [github](#).

We can add any additional information about the library here. Please make suggestions.



This is where the introduction to our libraries will go. Ideally, this will be a short overview of the library, with one or two very simple use cases. It will, hopefully, answer the question: ‘Why should I use this tool?’

## 1.1 Installation

This part of the documentation covers the installation of OpenEst.

This library depends on `numpy` and `scipy`. In addition, to use Mean-Size hierarchical sampling, the `emcee` library must be installed.

```
pip install numpy
```

```
pip install scipy
```

```
pip install emcee
```

Install the package by calling `python setup.py install` (or use *develop* rather than *install* if you’ll be editing the code).





If you are looking for information on a specific function, class, or method, this part of the documentation is for you.

## 2.1 openest package

### 2.1.1 Subpackages

**openest.generate**

**Submodules**

**openest.generate.calculation module**

**class** openest.generate.calculation.**Application**(*region*)

Bases: object

**done**()

**push**(*ds*)

Returns an iterator of (yyyy, value, ...).

**class** openest.generate.calculation.**ApplicationByChunks**(*region*)

Bases: *openest.generate.calculation.Application*

**push**(*ds*)

**push\_saved**(*ds*)

Returns an iterator of (yyyy, value, ...). Removes used daily values from saved.

**class** openest.generate.calculation.**ApplicationByIrregular**(*region*, *func*, *\*args*,  
*\*\*kwargs*)

Bases: *openest.generate.calculation.Application*

**push**(*ds*)

```
class openest.generate.calculation.ApplicationByYear (region, func, *args, **kwargs)
    Bases: openest.generate.calculation.ApplicationByChunks

    push_saved (ds)
        Returns an iterator of (yyyy, value, ...). Removes used daily values from saved.

class openest.generate.calculation.ApplicationEach (region, func, finishfunc=<function
                                                    <lambda>>, *args, **kwargs)
    Bases: openest.generate.calculation.Application
    Pass every set of values to the calculation for a value.

    done ()

    push (ds)

class openest.generate.calculation.ApplicationPassCall (region, subapp, handler,
                                                         *handler_args, **han-
                                                         dler_kw)
    Bases: openest.generate.calculation.Application
    Apply a non-enumerator to all elements of a function. if unshift, tack on the result to the front of a sequence of
    results. Calls func with each year and value; returns the newly computed value

    push (ds)
        Returns an iterator of (yyyy, value, ...).

class openest.generate.calculation.Calculation (unitses)
    Bases: object

    apply (region, *args, **kwargs)

    cleanup ()

    column_info ()
        Returns an array of dictionaries, with 'name', 'title', and 'description'.

    static describe ()
        Returns dictionary containing: - input_timerate: expected time rate of data, day, month, year, or any -
        output_timerate: expected time rate of data, day, month, year, or same - arguments: a list of subclasses of
        arguments.ArgumentType, describing each constructor argument - description: text description

    format (lang, *args, **kwargs)
        Returns a dictionary of FormatElements. Only keys in the tree of dependencies will be output.

    test ()

class openest.generate.calculation.CustomFunctionalCalculation (subcalc,
                                                                from_units,
                                                                to_units, unshift,
                                                                *handler_args,
                                                                **handler_kw)
    Bases: openest.generate.calculation.FunctionalCalculation, openest.generate.
calculation.Application
    Calculation that creates a copy of itself for an application.

    apply (region, *args, **kwargs)

    done ()

    donehandler (*allargs, **allkwargs)

    init_apply ()

    push (ds)
```

```

    pushhandler (ds, *allargs, **allkwargs)

class openest.generate.calculation.FunctionalCalculation (subcalc, from_units,
                                                         to_units, unshift, *han-
                                                         dler_args, **han-
                                                         dler_kw)

    Bases: openest.generate.calculation.Calculation

    Calculation that calls a handler when it's applied.

    apply (region, *args, **kwargs)

    cleanup ()

    format (lang, *args, **kwargs)

    format_handler (substr, lang, *handler_args, **handler_kw)

    handler (year, result, *handler_args, **handler_kw)

```

## openest.generate.curvegen module

```

class openest.generate.curvegen.ConstantCurveGenerator (indepunits, depenunit,
                                                         curve)

    Bases: openest.generate.curvegen.CurveGenerator

    format_call (lang, *args)

    get_curve (region, year, *args, **kw)

class openest.generate.curvegen.CurveGenerator (indepunits, depenunit)
    Bases: object

    format_call (lang, *args)

    get_curve (region, year, *args, **kw)
        Returns an object of type Curve.

class openest.generate.curvegen.DelayedCurveGenerator (curvegen)
    Bases: openest.generate.curvegen.CurveGenerator

    format_call (lang, *args)

    get_curve (region, year, *args, **kwargs)

    get_next_curve (region, year, *args, **kwargs)

class openest.generate.curvegen.TransformCurveGenerator (transform, description,
                                                         *curvegens)

    Bases: openest.generate.curvegen.CurveGenerator

    format_call (lang, *args)

    get_curve (region, year, *args, **kw)

```

## openest.generate.daily module

## openest.generate.functions module

## openest.generate.latextools module

`openest.generate.latextools.call(func, units, description=None, *args)`

Return a representation of this call. Any elements in args can be given their own FormatElements in the final dictionary.

`openest.generate.latextools.english_function(func, *args)`

`openest.generate.latextools.latex_function(func, *args)`

## openest.generate.retrieve module

`openest.generate.retrieve.any_from_url(url)`

Returns a model retrieved from the argument *url*

*any\_from\_url* is a wrapper around *from\_url()*. It returns a model chosen by *choose\_model()*. Therefore, the file reader returned by *from\_url()* must have one of the allowed model types as the first four characters in the document.

**Parameters** *url* (*str*) – URL of file to retrieve

**Returns** Model chosen by *choose\_model()*

**Return type** object

`openest.generate.retrieve.choose_model(fp, source=None)`

Reads a file object and returns a model based on file header

The file is converted into a *BinModel*, *DDPModel*, or *SplineModel* depending on the first four characters of the file.

To use *choose\_model*, the first four characters of the file reader object must be one of the following:

- *bin1*, in which case a *BinModel* will be returned,
- *ddp1* or *ddp2*, in which case a *DDPModel* will be returned, or
- *spp1*, in which case a *SplineModel* will be returned.

If the model type is not one of the types listed above, a *BaseException* will be raised.

---

### Todo:

- **Change exception type - subclassing *BaseException* is not PEP compliant.** Custom exceptions should inherit from *Exception* or other built-in exceptions. This is so that `except Exception` will catch all exceptions except for *KeyboardInterrupt* and *SystemExit*, which are not errors, but user-triggered events. See [PEP-352](#).
- 

### Parameters

- **fp** (*file reader object*) – file reader object to be converted into a model.
- **source** (*str*) – Meta-information about url the file was recovered from

**Returns** Model of class *BinModel*, *DDPModel*, or *SplineModel*.

**Return type** object

`openest.generate.retrieve.ddp_from_url(url)`

Returns a *DDPModel* from the argument *url*

`openest.generate.retrieve.from_url(url, create_func)`

Returns a `StringIO.StringIO` buffer with the contents of the response from *url*

---

**Todo:**

- response from `urllib2.urlopen(req)` is already a buffer. Is writing to a new buffer necessary?
- 

`openest.generate.retrieve.spline_from_url(url)`

Returns a *SplineModel* from the argument *url*

## openest.generate.shortterm module

```
class openest.generate.shortterm.InstaZScoreApply(units,                                curve,
                                                curve_description,          lasttime,
                                                weather_change=<function
                                                <lambda>>>)
```

Bases: `openest.generate.calculation.Calculation`, `openest.generate.calculation.Application`

**apply** (*region, \*args, \*\*kwargs*)

**column\_info** ()

**static describe** ()

**push** (*time, weather*)

```
class openest.generate.shortterm.MonthlyClimateApply(units, curve, curve_description,
                                                    monthmeans,          regions,
                                                    weather_change=<function
                                                    <lambda>>>)
```

Bases: `openest.generate.calculation.Calculation`

**apply** (*region, \*args*)

**column\_info** ()

**static describe** ()

```
class openest.generate.shortterm.MonthlyZScoreApply(units, curve, curve_description,
                                                    monthmeans,          months-
                                                    devs,              regions,
                                                    weather_change=<function
                                                    <lambda>>>)
```

Bases: `openest.generate.calculation.Calculation`, `openest.generate.calculation.Application`

**apply** (*region, \*args, \*\*kwargs*)

**column\_info** ()

**static describe** ()

**push** (*time, weather*)

```
class openest.generate.shortterm.SingleWeatherApply(units, curve, curve_description,  
                                                    weather_change=<function  
                                                    <lambda>>())  
    Bases: openest.generate.calculation.Calculation  
    apply (region, *args)  
    column_info ()  
    static describe ()  
  
class openest.generate.shortterm.SplitByMonth(subcalc)  
    Bases: openest.generate.calculation.Calculation  
    apply (region, *args, **kwargs)  
    column_info ()  
    static describe ()
```

## openest.generate.stdlib module

## openest.generate.weathertools module

```
openest.generate.weathertools.combo_effects(effect_dicts, scale_gens)  
openest.generate.weathertools.date_to_datestr(date)  
openest.generate.weathertools.get_crop_calendar(cropfile)  
openest.generate.weathertools.growing_seasons_daily_ncdf(yyyyddd, weather, plant-  
                                                         day, harvestday)  
openest.generate.weathertools.growing_seasons_mean_ncdf(yyyyddd, weather, plant-  
                                                         day, harvestday)  
openest.generate.weathertools.growing_seasons_mean_reader(reader, plantday, har-  
                                                         vestday)  
openest.generate.weathertools.read_scale_file(filepath, factor)  
openest.generate.weathertools.xmap_apply_model(xmap, model, pval)  
openest.generate.weathertools.yearly_daily_ncdf(yyyyddd, weather)
```

## openest.lincombo package

### Submodules

## openest.lincombo.continuous\_sampled module

```
class openest.lincombo.continuous_sampled.ContinuousSampled(func)  
    Bases: scipy.stats._distn_infrastructure.rv_continuous  
    guess_ranges (mini, maxi, count=10000)  
    guess_ranges_gridded (mini, maxi, count=10000)  
    pdf (xxs)  
    prepare_draws (mini, maxi, count=10000)
```

```
prepare_draws_gridded (mini, maxi, count=10000)
rvs (size=1, random_state=None)
```

## openest.lincombo.helpers module

### Helper functions

```
openest.lincombo.helpers.check_arguments (betas, stderrs, portions)
    Ensure that the parameters have the right dimensions for calculation.
openest.lincombo.helpers.issparse (portions)
    Check if an array is sparse.
```

## openest.lincombo.hiernorm module

```
openest.lincombo.hiernorm.alpha_given_taus (betas, stdvars, portions, obstaus)
openest.lincombo.hiernorm.betahat_given_taus (betas, stdvars, portions, obstaus)
openest.lincombo.hiernorm.get_sampled_column (allvals, col)
openest.lincombo.hiernorm.lincombo_hiernorm_taubyalpha (betas, stderrs, portions, maxtau=None, guess_range=False, draws=100)
openest.lincombo.hiernorm.lincombo_hiernorm_taubybeta (betas, stderrs, portions, maxtaus=None, guess_range=False, draws=100)
openest.lincombo.hiernorm.probability_tau (alphas, taus, obstaus, betas, stdvars, portions, probability_prior_taus)
openest.lincombo.hiernorm.sample_posterior (betas, stderrs, portions, taudist, taus2obstaus, draws=100)
```

## openest.lincombo.hierregress module

```
openest.lincombo.hierregress.betahat_given_tau (yy, stdvars, XX, tau)
openest.lincombo.hierregress.get_sampled_column (allvals, col)
openest.lincombo.hierregress.lincombo_hierregress (yy, stderrs, XX, maxtau=None, guess_range=False, draws=100)
openest.lincombo.hierregress.lincombo_hierregress_taubybeta (yy, stderrs, XX, maxtau=None, guess_range=False, draws=100)
openest.lincombo.hierregress.lincombo_hierregress_taubymu (yy, stderrs, XX, maxtau=None, guess_range=False, draws=100)
openest.lincombo.hierregress.mu_given_tau (yy, stdvars, XX, tau)
```

```
openest.lincombo.hierregress.probability_tau(mus, tau, yy, stdvars, XX, probability_prior_tau)
openest.lincombo.hierregress.sample_posterior(yy, stderrs, XX, taudist, draws=100)
```

### openest.lincombo.montecarlo module

```
openest.lincombo.montecarlo.regress_distribution(means, serrs, XX, count=1000)
openest.lincombo.montecarlo.regress_draws(means, serrs, XX, count=1000)
openest.lincombo.montecarlo.regress_summary(means, serrs, XX, count=1000)
```

### openest.lincombo.multi\_delta module

```
class openest.lincombo.multi_delta.MultivariateDelta(vals)
    Bases: scipy.stats._multivariate.multi_rv_frozen

    pdf(xxs)

    rvs(size=1, random_state=None)

    vals()
```

### openest.lincombo.multi\_draws module

```
class openest.lincombo.multi_draws.MultivariateDraws(draws)
    Bases: scipy.stats._multivariate.multi_rv_frozen

    mean()

    rvs(size=1, random_state=None)

    std()
```

### openest.lincombo.multi\_normal module

```
class openest.lincombo.multi_normal.MultivariateNormal(means, big_sigma)
    Bases: scipy.stats._multivariate.multi_rv_frozen

    logpdf(xxs)

    pdf(xxs)

    rvs(size=1)
```

### openest.lincombo.multi\_sampled module

```
class openest.lincombo.multi_sampled.MultivariateSampled(func, dims)
    Bases: scipy.stats._multivariate.multi_rv_frozen

    guess_ranges(mins, maxs, count=10000)

    guess_ranges_gridded(mins, maxs, count=10000)

    pdf(xxs)
```



```

prepare_draws (mins, maxs, count=10000)
prepare_draws_gridded (mins, maxs, lens)
rvs (size=1, random_state=None)

```

### openest.lincombo.multi\_uniform module

```

class openest.lincombo.multi_uniform.MultivariateUniform (mins, maxs)
    Bases: scipy.stats._multivariate.multi_rv_frozen

    maxs ()
    mins ()
    pdf (xxs)
    rvs (size=1, random_state=None)

```

### openest.lincombo.pooling module

Create a pooled estimates.

The main function is *lincombo\_pooled*.

```

openest.lincombo.pooling.estimated_maxlintaus (betas, stderrs, portions)
    For use with hiernorm by-beta.

openest.lincombo.pooling.estimated_maxtau (betas, stderrs, portions)
    For use with hiernorm by-alpha.

openest.lincombo.pooling.lincombo_pooled (betas, stderrs, portions)

openest.lincombo.pooling.sum_multiply (sparsecol, densevec)

openest.lincombo.pooling.sum_multiply2 (sparse, col1, col2, densevec)

```

## openest.models package

### Submodules

#### openest.models.bin\_model module

```

class openest.models.bin_model.BinModel (xx=None, model=None)
    Bases: openest.models.univariate_model.UnivariateModel, openest.models.memoizable.MemoizableUnivariate

```

Bin Model

A bin model represents bins of different spans, where the distribution is constant over each bin. It is a combination of information describing the bins and an underlying categorical model of one of the other types.

The underlying model is always categorical, with categories starting at 1. 0 is reserved for a future version that allows an out-of-sample distribution

The format is:

```
bin1
<x0>,<x1>,<x2>, ...
<underlying model>
```

### Parameters

- **xx** (*list-like*) – List-like array of bin edges. *len(xx)* should be one more than the number of bins.
- **model** (*object*) – Statistical model used in each bin

**cdf** (*x, y*)

**static combine** (*one, two*)  
Both models are BinModels

**static consistent\_bins** (*models*)  
All models are BinModels

**copy** ()  
copy data and return BinModel with the same data

**draw\_sample** (*x=None*)

**eval\_pval** (*x, p, threshold=0.001*)

**eval\_pval\_index** (*ii, p, threshold=0.001*)

**filter\_x** (*xx*)  
Returns new *BinModel*

**get\_bin\_at** (*x*)  
Returns bin containing value *x*

**Parameters** *x* (*numeric*) – Value to search for in binned axis

**Returns** Returns index of bin containing *x*. If bin is not contained in the bin range, returns -1.

**Return type** int

**get\_edges** ()  
Returns bin edges (duplicate of *get\_xx()*)

**get\_mean** (*x=None, index=None*)

**get\_sdev** (*x=None, index=None*)

**get\_xx** ()  
returns x axis index

**get\_xx\_centers** ()  
returns x axis index

**init\_from\_bin\_file** (*file, delimiter, status\_callback=None, init\_submodel=<function <lambda>>>*)

**interpolate\_x** (*newxx*)  
Returns a copy of the model. *Does not interpolate.*

**kind** ()  
returns model type (“bin\_model”)

**static merge** (*models*)  
All models are BinModels

**scale\_p** (*a*)  
 Scales p-values of underlying bin models (in log\_p format)  
 Interface to *self.model.scale\_p*.

**scale\_y** (*a*)  
 Scales y-axes of underlying bin models  
 Interface to *self.model.scale\_y(a)*

**to\_ddp** (*ys=None*)

**to\_points\_at** (*x, ys*)

**write** (*file, delimiter*)  
 Write model as delimited document to file-like object  
 Prepends model type (bin1) and bin borders (xx) to document written by *self.model.write*.

#### Parameters

- **file** (*object*) – file-like object
- **delimiter** (*str*) – Delimiter to use in file (e.g. ‘ ‘, ‘;’)

**write\_file** (*filename, delimiter*)  
 Write model as delimited document to filepath  
 Wrapper around *write()* method.

#### Parameters

- **filename** (*str*) – Path to file to be written
- **delimiter** (*str*) – Delimiter to use in file (e.g. ‘ ‘, ‘;’)

## openest.models.curve module

**class** openest.models.curve.ClippedCurve (*curve, cliplow=True*)  
 Bases: openest.models.curve.UnivariateCurve

**class** openest.models.curve.CoefficientsCurve (*coeffs, curve, xtrans*)  
 Bases: openest.models.curve.UnivariateCurve  
 A curve represented by the sum of multiple predictors, each multiplied by a coefficient.

**class** openest.models.curve.CubicSplineCurve (*knots, coeffs*)  
 Bases: openest.models.curve.UnivariateCurve

**get\_terms** (*x*)  
 Get the set of knots-1 terms representing temperature x.

**class** openest.models.curve.CurveCurve (*xx, curve*)  
 Bases: openest.models.curve.UnivariateCurve

**static make\_linear\_spline\_curve** (*xx, yy, limits*)

**class** openest.models.curve.FlatCurve (*yy*)  
 Bases: openest.models.curve.CurveCurve

**class** openest.models.curve.LinearCurve (*yy*)  
 Bases: openest.models.curve.CurveCurve

**class** openest.models.curve.MinimumCurve (*curve1, curve2*)  
 Bases: openest.models.curve.UnivariateCurve

```

class openest.models.curve.OtherClippedCurve (clipping_curve, value_curve)
    Bases: openest.models.curve.ClippedCurve

class openest.models.curve.PiecewiseCurve (curves, knots, xtrans=<function <lambda>>)
    Bases: openest.models.curve.UnivariateCurve

class openest.models.curve.ProductCurve (curve1, curve2)
    Bases: openest.models.curve.UnivariateCurve

class openest.models.curve.SelectiveInputCurve (curve, indices)
    Bases: openest.models.curve.UnivariateCurve

    Assumes input is a matrix, and only pass selected input columns to child curve.

class openest.models.curve.ShiftedCurve (curve, offset)
    Bases: openest.models.curve.UnivariateCurve

class openest.models.curve.StepCurve (xxlimits, yy, xtrans)
    Bases: openest.models.curve.CurveCurve

class openest.models.curve.UnivariateCurve (xx)
    Bases: openest.models.univariate_model.UnivariateModel

    eval_pval (x, p, threshold=0.001)
    eval_pvals (x, p, threshold=0.001)
    get_xx ()

class openest.models.curve.ZeroInterceptPolynomialCurve (xx, ccs)
    Bases: openest.models.curve.UnivariateCurve

openest.models.curve.pos (x)

```

## openest.models.ddp\_model module

```

class openest.models.ddp_model.DDPModel (p_format=None, source=None,
                                          xx_is_categorical=False, xx=None,
                                          yy_is_categorical=False, yy=None, pp=None,
                                          unaccounted=None, scaled=True)

    Bases: openest.models.univariate_model.UnivariateModel, openest.models.
    memoizable.MemoizableUnivariate

```

Discrete-Discrete-Probability (DDP) Format

A DDP file describes a dose-response relationship with a limited collection of response outcomes. The dose and response values may be either categorical or sampled at a collection of numerical levels.

<y-value-1>, ..., <y-value-N> and <x-value-1>, ..., <x-value-N> are either strings (for named categories) or numerical values.

The format of a DDP file is:

```

<format>, <y-value-1>, <y-value-2>, ...
<x-value-1>, p(y1|x1), p(y2|x1), ...
<x-value-2>, p(y1|x2), p(y2|x2), ...

```

Below is a sample categorical DDP file:

```

ddp1, live, dead
control, .5, .5
treated, .9, .1

```

Below is a sample numerical DDP file:

```
ddp1, -10.0, -.333333333333, 3.33333333333, 10.0
0.0, 0.5, 0.5, 0.0, 0.0
13.3333333333, 0.0, 0.5, 0.5, 0.0
26.6666666667, 0.0, 0.0, 0.5, 0.5
40.0, 0.0, 0.0, 0.0, 0.5
```

### Parameters

- **p\_format** (*str*) – Probability format. May be one of the following values:
  - ddp1 - the p(.) values are simple probabilities ( $0 < p(.) < 1$  and  $\sum p(y|x) = 1$ )
  - ddp2 - the p(.) values are log probabilities
- **source** (*str*) – Metadata attribute. Name of file this object was read in from.
- **xx\_is\_categorical** (*bool*) – Indicates whether xx is categorical. False indicates numeric data.
- **xx** (*list-like*) – X axis index
- **yy\_is\_categorical** (*bool*) – Indicates whether yy is categorical. False indicates numeric data.
- **yy** (*list-like*) – Y axis index
- **pp** (*array-like*) – underlying numpy(?) data array
- **unaccounted** (*numpy.array*) – column of remaining probability. `unaccounted = 1 - sum(pp, axis=1)`.
- **scaled** (*bool*) – Indicates whether data has been scaled. If scaled, re-scale so `pp.sum(axis=1) == 1`.

**add\_to\_y** (*a*)

add value a to each element of index y (numeric only)

**static combine** (*one, two*)

**copy** ()

copy data and return DDPMModel with the same data

**static create\_lin** (*yy, xxs*)

Create a DDP model by supplying y index and dictionary of p-values

### Parameters

- **yy** (*list-like*) – y-index labels
- **xxs** (*dict*) – dictionary keyed with x-index values with p-values for vals

**draw\_sample** (*x=None*)

Randomly sample label from y-index using p values in row x

If x is None (default), use first row. Uses `self.get_closest(x)` to find matching nearest match for x-index label x

**eval\_pval** (*x, p, threshold=0.001*)

**eval\_pval\_index** (*ii, p, threshold=0.001*)

**filter\_x** (*xx*)

Slice DDPMModel data such that the values of the x index == xx

**static from\_file** (*filename, delimiter*)

read DDP file from file path

**get\_closest** (*x=None*)

return closest index on x axis

If x index is categorical, coerce x to string and find first matching index. If numeric, find the closest value.

If x is None (default), return 0

**get\_mean** (*x=None*)

Returns the mean of the y-index labels weighted by p values in row x

If x is None (default), use first row. Uses self.get\_closest(x) to find matching nearest match for x-index label x

**get\_sdev** (*x=None*)

Returns the std dev of the y-index labels weighted by p values in row x

If x is None (default), use first row. Uses self.get\_closest(x) to find matching nearest match for x-index label x

**get\_xx** ()

returns x axis index

**get\_yy** ()

returns x axis index

**init\_from** (*file, delimiter, status\_callback=None, source=None*)

Read DDP data set from file

**init\_from\_other** (*ddp*)

copy attributes of other DDP dataset to this one

**interpolate\_x** (*newxx, kind='quadratic'*)

custom interpolation method. wrapper around scipy.interp1d.

#### Parameters

- **newxx** (*list-like*) – new x axis
- **kind** (*str*) – interpolation method, passed to scipy.interp1d

**interpolate\_y** (*newyy, kind='quadratic'*)

custom interpolation method. wrapper around scipy.interp1d.

#### Parameters

- **newyy** (*list-like*) – new y axis
- **kind** (*str*) – interpolation method, passed to scipy.interp1d

**kind** ()

returns model type (“ddp\_model”)

**lin\_p** ()

convert any DDPMModel to ddp1 (linear probability) format

**log\_p** ()

convert any DDPMModel to ddp2 (log probability) format

**static merge** (*models*)

**recategorize\_x** (*oldxx, newxx*)

```

rescale (as_ddp=True)
    Can rescale non-ddp (that is, as sampling of continuous distribution)

scale_p (a)
    coerce to ddp2 (log probability) format and scale by a

scale_y (a)
    multiply index y (numeric only) by scale factor a

to_ddp (ys=None)
    coerce to DDP, interpolating along y axis if necessary

transpose ()
    transpose data structure

write (file, delimiter)
    write CSV to file object

write_file (filename, delimiter)
    write CSV to file path

```

### openest.models.delta\_model module

```

class openest.models.delta_model.DeltaModel (xx_is_categorical=False, xx=None, loca-
                                             tions=None, scale=1)
    Bases: openest.models.univariate_model.UnivariateModel

    cdf (xx, yy)

    static combine (one, two)

    copy ()

    draw_sample (x=None)

    filter_x (xx)

    init_from_delta_file (file, delimiter, status_callback=None)

    interpolate_x (newxx, kind='quadratic')

    kind ()

    static merge (models)

    scale_p (a)
        Raising a delta function to a power makes no difference.

    scale_y (a)

    to_points_at (x, ys)

    write (file, delimiter)

    write_file (filename, delimiter)

    static zero_delta (model)

```

**openest.models.distribution\_model module**

```
class openest.models.distribution_model.DistributionModel (p_format=None,
                                                    source=None,
                                                    xx_is_categorical=False,
                                                    xx=None,
                                                    yy_is_categorical=False,
                                                    yy=None, pp=None,
                                                    unaccounted=None,
                                                    scaled=True)

Bases: openest.models.ddp_model.DDPModel

apply_as_distribution (model)
```

**openest.models.features\_interpreter module**

## Probability Features File

The probability features file has the following format:

```
dpcl, <p-header-1>, <p-header-2>, ...
<x-value-1>, g_1(y | x_1), g_2(y | x_1), ...
<x-value-2>, g_1(y | x_2), g_2(y | x_2), ...
...
```

<p-header> headers can be any of the following, with the corresponding values in their rows (<p-value-ij>).

- mean:  $E(y|x_i)$
- var:  $E(y|x_i - E(y|x_i))^2$
- sdev:  $\sqrt{E(y|x_i - E(y|x_i))^2}$
- skew:  $E((y|x_i - E(y|x_i)) / \sqrt{E(y|x_i - E(y|x_i))^2})^3$
- mode:  $\max f(y | x_i)$
- numeric (0 - 1):  $F^{-1}\{p_j|x_i\}$

The row headers (<x-value>) can be numeric, in which case a continuous spline bridges them, or categorical strings.

Below is a sample features file:

```
dpcl, mean, var
treated, 0, 1
control, 4, 4
```

```
class openest.models.features_interpreter.FeaturesInterpreter
```

```
static best_knot (knots, newknots)
    Find the knot furthest from existing knots

static best_spline (header, row, limits)

static evaluate_spline (header, row, spline, limits)

static features_to_exponential (header, row, limits)

static features_to_gaussian (header, row, limits)

static features_to_uniform (header, row, limits)
```



```

static init_from_feature_file (spline, file, delimiter, limits, status_callback=None)
static make_conditional (header, row, limits)
static make_conditional_respecting (header, row, limits)
static skew_gaussian_construct (ys, lps, low_segment, high_segment)
static skew_gaussian_evaluate (ys, lps, low_segment, high_segment, mean, lowp, highp)

```

### openest.models.generate module

```

openest.models.generate.polynomial (lowbound, highbound, betas, covas, num=40)
openest.models.generate.uniform_constant (xx, yy, min, max)
openest.models.generate.uniform_doseless (start, end, height=None)

```

### openest.models.hierarchical\_normal module

```

openest.models.hierarchical_normal.draw_from_counts (x_counts, x_range, pval=None)
openest.models.hierarchical_normal.generate_thetas (mu_counts, mu_range,
                                                    tau_counts, tau_range, count)
openest.models.hierarchical_normal.get_random (taus, F_tau, count)
openest.models.hierarchical_normal.helper_params (means, varis, tau)
openest.models.hierarchical_normal.p_tau_given_y (tau, means, varis)
openest.models.hierarchical_normal.simulate_normal_model (means, serrs,
                                                            count, taus=None,
                                                            do_thetas=False)

```

### openest.models.integral\_model module

Integral model

The integral over x of another model.

```

class openest.models.integral_model.IntegralModel (model=None)
    Bases: openest.models.univariate_model.UnivariateModel

    copy ()
    eval_pval (x, p, threshold=0.001)
    get_xx ()
    interpolate_x (newxx)
    kind ()
    scale_p (a)
    scale_y (a)
    write (file, delimiter)
    write_file (filename, delimiter)

```

## openest.models.mean\_size\_model module

### Mean-Size Model

In Mean-Size models, each point is characterized only by a value and the population size that went into estimating that value. As such, it does not have enough information to generate a full distribution. It can be safely combined with other mean-size models, or approximated with a Gaussian (with a variance which is equal to the absolute value of the mean for size = 1, and a variance that decreases with the square root of the size, according to the Central Limit Theorem).

The format is:

```
msx1,mean,size
<x0>,<mean0>,<size0>
<x1>,<mean1>,<size1>
...
```

```
class openest.models.mean_size_model.MeanSizeModel (xx_is_categorical=False,
                                                    xx=None,          means=None,
                                                    sizes=None)
    Bases: openest.models.univariate_model.UnivariateModel
    attribute_list ()
    static combine (one,two)
    copy ()
    filter_x (xx)
    get_attribute (title)
    get_mean (x=None)
    get_sdev (x=None)
    init_from_mean_size_file (file,delimiter,status_callback=None)
    interpolate_x (newxx,kind='quadratic')
    kind ()
    static merge (models,treatment='default')
    scale_p (a)
    scale_y (a)
    write (file,delimiter)
    write_file (filename,delimiter)
```

## openest.models.memoizable module

```
class openest.models.memoizable.MemoizableUnivariate
    Bases: object
    eval_pval_index (ii,p,threshold=0.001)
    get_edges ()
class openest.models.memoizable.MemoizedUnivariate (model)
    Bases: openest.models.univariate_model.UnivariateModel
```

```

copy()
eval_pval(x, p, threshold=0.001)
eval_pvals(xs, p, threshold=0.001)
get_eval_pval_spline(p, limits, threshold=0.001, linextrap=False)
get_index(x)
get_indexes(xs)
get_xx()
interpolate_x(newxx)
kind()
reset_cache()
scale_p(a)
scale_y(a)
set_x_cache_decimals(decimals)
write(file, delimiter)
write_file(filename, delimiter)

```

## openest.models.model module

```

class openest.models.model.Attribute(title, description, reference, subtitle, value, comments,
                                     source)

```

Bases: object

An attribute is an arbitrary piece of information about a model, available from the attribute functions on Model.

```

class openest.models.model.Model(scaled=True)

```

Bases: object

Model class

Top level Model class, from which all specific model derive. All models should implement most of these functions (with the notable exceptions of merge and combine).

```

attribute_list()

```

```

static combine(models, factors)

```

Construct a weighted sum over the shared values of x

Each form provides methods for constructing the distribution of the sum of multiple parameters, which is generally constructed by performing the convolution:  $p(y + z | x) = p_y(y | x) * p_z(z | x)$ .

```

combiners = {'delta_model+ddp_model': <function combine>, 'delta_model+delta_model':

```

```

copy()

```

```

draw_sample(x=None)

```

Produce a sample value of y from the conditional distribution.

```

eval_pval(x, p, threshold=0.001)

```

Inverse CDF Evaluation

Returns the value of  $y$  that corresponds to a given p-value:  $F^{-1}(p | x)$ .

```

get_attribute(title)

```

**get\_mean** (*x=None*)  
 $E[Y | X]$

**get\_sdev** (*x=None*)  
 $\sqrt{\text{Var}[Y | X]}$

**kind** ()

**static merge** (*models*)  
 Pooling Merging

Each form provides methods for producing a pooled parameter estimate from multiple parameter estimates. These could all be parameter estimates with the same form, or with two different forms:  $p_1(y | x)$   $p_2(y | x)$ .

**mergers** = {'delta\_model': <function merge>, 'mean\_size\_model': <function merge>, 'de

**scale\_p** (*a*)  
 Raise the distribution to the power 'a' and rescales.

**Returns** modifies this model and returns it

**Return type** self

**scale\_y** (*a*)  
 Rescaling of the Parameter Dimension

Produces a new conditional PDF with the  $y$  dimension scaled by a constant:  $p(z | x) = p(\text{rac}\{y\}\{a\} | x)$ .

**to\_points\_at** (*x, ys*)  
 Conditional Probability Density Evaluation

Returns unscaled probability density values for given values of  $x$  and  $y$ :  $f(y | x)$ .

## openest.models.multivariate\_model module

**class** openest.models.multivariate\_model.**MultivariateModel** (*xx\_is\_categoricals*,  
*scaled=True*)

Bases: openest.models.model.Model

**condition** (*conditions*)

**default\_condition** ()

**numvars** ()

## openest.models.outer\_multi\_model module

**class** openest.models.outer\_multi\_model.**OuterMultiModel** (*xxs*, *xx\_is\_categoricals*,  
*union*, *scaled=True*)

Bases: openest.models.multivariate\_model.MultivariateModel

**condition** (*conditions*)

**default\_condition** ()

**dims** ()

**float\_condition** (*conditions*)

**init\_from\_union** (*union*)

```

kind()
static re_condition(condition)
re_numeric = <_sre.SRE_Pattern object>
scale_p(a)
write(file, delimiter)
write_file(filename, delimiter)

```

## openest.models.parameter module

```

class openest.models.parameter.ParameterBase(title, units)
    Bases: object
    derive(subtitle)

```

## openest.models.spline\_model module

```

class openest.models.spline_model.SplineModel(xx_is_categorical=False, xx=None, con-
                                              ditionals=None, scaled=True)
    Bases: openest.models.univariate_model.UnivariateModel, openest.models.
memoizable.MemoizableUnivariate

```

### Model Spline File

Each line in a model spline file represents a polynomial segment in log-probability space. The format is as follows:

```

spp1
<x>, <y0>, <y1>, <a0>, <a1>, <a2>
...

```

Each line describes a segment of a probability distribution of  $y$ , conditional on  $x = \langle x \rangle$ . The segment spans from  $\langle y0 \rangle$  to  $\langle y1 \rangle$ , where the lowest value of  $\langle y0 \rangle$  may be  $-\text{inf}$ , and the highest value of  $\langle y1 \rangle$  may be  $\text{inf}$ . The  $\langle x \rangle$  values may also be categorical or numerical. If they are numerical, it is assumed that these values represent samples of a smoothly varying function (a cubic spline in every  $y$ ).

The values  $\langle a0 \rangle$ ,  $\langle a1 \rangle$  and  $\langle a2 \rangle$  are the polynomial coefficients in  $y$  (with quadratic coefficients, only normal or exponential tails are possible). The final segment of the probability function is:

```
exp(a0 + a1 y + a2 y2)
```

### Parameters

- **xx\_is\_categorical** (*bool*) –
- **xx** (*list-like*) –
- **conditionals** –
- **scaled** (*bool*) –

```
add_conditional(x, conditional)
```

```
cdf(xx, yy)
```

```
static combine(one, two)
```

```
copy ()

static create_gaussian (xxs, order=None, xx_is_categorical=True)
    xxs should be a dictionary of the form {x: (mean, variance)}.

static create_single (xxs, y0s, y1s, coeffss, order=None, xx_is_categorical=True)

draw_sample (x=None)

eval_pval (x, p, threshold=0.001)

eval_pval_index (ii, p, threshold=0.001)

filter_x (xx)

static from_ddp (ddp_model, limits)

get_conditional (x)

get_mean (x=None)

get_sdev (x=None)

get_xx ()

init_from_spline_file (file, delimiter, status_callback=None)

interpolate_x (newxx)
    Determines whether argument newxx a subset of index xx.

is_gaussian (x=None)

kind ()

static merge (models)

neginf = -inf

posinf = inf

recategorize_x (oldxx, newxx)
    Construct a new model with categorical x values 'newxx', using the conditionals currently assigned to
    categorical x values 'oldxx'.

samples = 1000

scale_p (a)

scale_y (a)

to_ddp (ys=None)

to_points_at (x, ys)

write (file, delimiter)

write_file (filename, delimiter)

write_gaussian (file, delimiter)

write_gaussian_plus (file, delimiter)

class openest.models.spline_model.SplineModelConditional (y0s=None, y1s=None,
                                                         coeffss=None)

    add_segment (y0, y1, coeffss)

    approximate_mean (limits)

    static approximate_sum (conditionals)
```

```

static ascinv (y, func, minx, maxx, threshold)
cdf (yy)
convolve (other)
copy ()
draw_sample ()
evaluate (ii, y)
find_mode ()
static find_nearest (array, value, within)
gaussian_mean (ii)
gaussian_sdev (ii)
get_pval (p, threshold=0.001)
is_gaussian ()
static make_conditional_from_spline (spline, limits)
static make_gaussian (y0, y1, mean, var)
static make_single (y0, y1, coeffs)
nongaussian_x2px (ii)
nongaussian_xpx (ii)
partial_cdf (ii, y1)
static propose_grid (conditionals)
rescale ()
rough_limits ()
rough_span ()
scale (factor)
scale_p (a)
scale_y (a)
segment_max (jj)
size ()
to_points (ys)

```

### openest.models.sum\_multi\_model module

```

class openest.models.sum_multi_model.SumMultiModel (unis)
    Bases: openest.models.multivariate_model.MultivariateModel

```

## openest.models.univariate\_model module

```
class openest.models.univariate_model.UnivariateModel (xx_is_categorical=False,  
                                                    xx=None, scaled=True)  
  
    Bases: openest.models.model.Model  
  
    filter_x (xx)  
  
    get_xx ()  
        Listing conditional values  
        Provide a list of all sampled conditional values.  
  
    interpolate_x (xx)  
  
    static intersect_get_model (model, xx)  
  
    static intersect_get_x (xx_is_categorical, one_xx, two_xx)  
  
    static intersect_x (one, two)  
  
    static intersect_x_all (models)  
  
    recategorize_x (oldxx, newxx)
```

## openest.swapbin package

### Submodules

## openest.swapbin.swapmodel module

```
openest.swapbin.swapmodel.find_bins (means, sdevs, beta, vcv)  
openest.swapbin.swapmodel.swap_any (model, beta, vcv, dropbin, totals)  
openest.swapbin.swapmodel.swap_bin (model, beta, vcv, dropbin, totals)  
openest.swapbin.swapmodel.swap_spline (model, beta, vcv, dropbin, totals)  
openest.swapbin.swapmodel.swap_values (means, sdevs, beta, vcv, dropbin, totals)
```

## openest.swapbin.transform module

```
openest.swapbin.transform.swap_beta (beta, T)  
openest.swapbin.transform.swap_vcv (V, T)  
openest.swapbin.transform.transform (predcount, bins, dropbin, totals)  
    Construct a transform matrix from an old set of predictors to a bin swapped set.
```

The intercept is assumed to be the first predictor.

### Parameters

- **predcount** (*int*) – the number predictors, including the intercept.
- **bins** (*list[int]*) – the indices of the bins amongst the predictors.
- **dropbin** (*int*) – an index into *bins*
- **totals** (*float*) – the value that all bin values would sum to, e.g. *1* if the bins are indicators e.g., 365 if the bins are daily over a year



## Module contents



---

## The Developers Guide

---

If you want to contribute to the project, this part of the documentation is for you.

### 3.1 Contributor's Guide

This document lays out guidelines and advice for contributing to this project. If you're thinking of contributing, please start by reading this document and getting a feel for how contributing to this project works. If you have any questions, feel free to reach out to either [James Rising](#), [Mike Delgado](#), or [Justin Simcock](#).

When contributing code, you'll want to follow this checklist:

1. Fork the repository on GitHub.
2. Run the tests to confirm they all pass on your system. If they don't, you'll need to investigate why they fail. If you're unable to diagnose this yourself, raise it as a bug report by following the guidelines in this document: [Bug Reports](#).
3. Write tests that demonstrate your bug or feature. Ensure that they fail.
4. Make your change.
5. Run the entire test suite again, confirming that all tests pass *including the ones you just added*.
6. Send a GitHub Pull Request to the main repository's `master` branch. GitHub Pull Requests are the expected method of code collaboration on this project.

#### 3.1.1 Documentation Links

The documentation files live in the `openest_docs/` directory of the codebase. They're written in `reStructuredText`, and use `Sphinx` to generate the full suite of documentation.

When writing documentation, please do your best to follow the style of the documentation files.

Here is an reference example Python Module with `reStructuredText` in the docstring.

```

"""Example NumPy style docstrings.

This module demonstrates documentation as specified by the `NumPy
Documentation HOWTO`. Docstrings may extend over multiple lines. Sections
are created with a section header followed by an underline of equal length.

Example
-----
Examples can be given using either the ``Example`` or ``Examples``
sections. Sections support any reStructuredText formatting, including
literal blocks::

    $ python example_numpy.py

Section breaks are created with two blank lines. Section breaks are also
implicitly created anytime a new section starts. Section bodies *may* be
indented:

Notes
-----
    This is an example of an indented section. It's like any other section,
    but the body is indented to help it stand out from surrounding text.

If a section is indented, then a section break is created by
resuming unindented text.

Attributes
-----
module_level_variable1 : int
    Module level variables may be documented in either the ``Attributes``
    section of the module docstring, or in an inline docstring immediately
    following the variable.

    Either form is acceptable, but the two should not be mixed. Choose
    one convention to document module level variables and be consistent
    with it.

.. _NumPy Documentation HOWTO:
    https://github.com/numpy/numpy/blob/master/doc/HOWTO\_DOCUMENT.rst.txt

"""

module_level_variable1 = 12345

module_level_variable2 = 98765
"""int: Module level variable documented inline.

The docstring may span multiple lines. The type may optionally be specified
on the first line, separated by a colon.
"""

def function_with_types_in_docstring(param1, param2):
    """Example function with types documented in the docstring.

    `PEP 484`_ type annotations are supported. If attribute, parameter, and

```

```

return types are annotated according to `PEP 484`, they do not need to be
included in the docstring:

Parameters
-----
param1 : int
    The first parameter.
param2 : str
    The second parameter.

Returns
-----
bool
    True if successful, False otherwise.

.. _PEP 484:
    https://www.python.org/dev/peps/pep-0484/

"""

def function_with_pep484_type_annotations(param1: int, param2: str) -> bool:
    """Example function with PEP 484 type annotations.

    The return type must be duplicated in the docstring to comply
    with the NumPy docstring style.

    Parameters
    -----
    param1
        The first parameter.
    param2
        The second parameter.

    Returns
    -----
    bool
        True if successful, False otherwise.

    """

def module_level_function(param1, param2=None, *args, **kwargs):
    """This is an example of a module level function.

    Function parameters should be documented in the ``Parameters`` section.
    The name of each parameter is required. The type and description of each
    parameter is optional, but should be included if not obvious.

    If *args or *kwargs are accepted,
    they should be listed as *args and *kwargs.

    The format for a parameter is::

        name : type
            description

        The description may span multiple lines. Following lines

```

*should be indented to match the first line of the description.  
The ": type" is optional.*

*Multiple paragraphs are supported in parameter  
descriptions.*

*Parameters*

-----

*param1 : int*

*The first parameter.*

*param2 : :obj:`str`, optional*

*The second parameter.*

*\*args*

*Variable length argument list.*

*\*\*kwargs*

*Arbitrary keyword arguments.*

*Returns*

-----

*bool*

*True if successful, False otherwise.*

*The return type is not optional. The ``Returns`` section may span  
multiple lines and paragraphs. Following lines should be indented to  
match the first line of the description.*

*The ``Returns`` section supports any reStructuredText formatting,  
including literal blocks::*

```
{
    'param1': param1,
    'param2': param2
}
```

*Raises*

-----

*AttributeError*

*The ``Raises`` section is a list of all exceptions  
that are relevant to the interface.*

*ValueError*

*If `param2` is equal to `param1`.*

*"""*

**if** param1 == param2:

**raise** ValueError('param1 may not be equal to param2')

**return True**

**def** example\_generator(n):

*"""Generators have a ``Yields`` section instead of a ``Returns`` section.*

*Parameters*

-----

*n : int*

*The upper limit of the range to generate, from 0 to `n` - 1.*

*Yields*

-----

```

int
    The next number in the range of 0 to `n` - 1.

Examples
-----
Examples should be written in doctest format, and should illustrate how
to use the function.

>>> print([i for i in example_generator(4)])
[0, 1, 2, 3]

"""
for i in range(n):
    yield i

class ExampleError(Exception):
    """Exceptions are documented in the same way as classes.

    The __init__ method may be documented in either the class level
    docstring, or as a docstring on the __init__ method itself.

    Either form is acceptable, but the two should not be mixed. Choose one
    convention to document the __init__ method and be consistent with it.

    Note
    ----
    Do not include the `self` parameter in the ``Parameters`` section.

    Parameters
    -----
    msg : str
        Human readable string describing the exception.
    code : :obj:`int`, optional
        Numeric error code.

    Attributes
    -----
    msg : str
        Human readable string describing the exception.
    code : int
        Numeric error code.

    """

    def __init__(self, msg, code):
        self.msg = msg
        self.code = code

class ExampleClass(object):
    """The summary line for a class docstring should fit on one line.

    If the class has public attributes, they may be documented here
    in an ``Attributes`` section and follow the same formatting as a
    function's ``Args`` section. Alternatively, attributes may be documented
    inline with the attribute's declaration (see __init__ method below).

```

*Properties created with the ``@property`` decorator should be documented in the property's getter method.*

*Attributes*

-----

```
attr1 : str
    Description of `attr1`.
attr2 : :obj:`int`, optional
    Description of `attr2`.
```

"""

```
def __init__(self, param1, param2, param3):
    """Example of docstring on the __init__ method.
```

*The \_\_init\_\_ method may be documented in either the class level docstring, or as a docstring on the \_\_init\_\_ method itself.*

*Either form is acceptable, but the two should not be mixed. Choose one convention to document the \_\_init\_\_ method and be consistent with it.*

*Note*

----

*Do not include the `self` parameter in the ``Parameters`` section.*

*Parameters*

-----

```
param1 : str
    Description of `param1`.
param2 : :obj:`list` of :obj:`str`
    Description of `param2`. Multiple
    lines are supported.
param3 : :obj:`int`, optional
    Description of `param3`.
```

"""

```
self.attr1 = param1
self.attr2 = param2
self.attr3 = param3 #: Doc comment *inline* with attribute
```

```
#: list of str: Doc comment *before* attribute, with type specified
self.attr4 = ["attr4"]
```

```
self.attr5 = None
"""str: Docstring *after* attribute, with type specified."""
```

**@property**

```
def readonly_property(self):
    """str: Properties should be documented in their getter method."""
    return "readonly_property"
```

**@property**

```
def readwrite_property(self):
    """obj:`list` of :obj:`str`: Properties with both a getter and setter
    should only be documented in their getter method.
```

*If the setter method contains notable behavior, it should be mentioned here.*



```

    """
    return ["readwrite_property"]

@readwrite_property.setter
def readwrite_property(self, value):
    value

def example_method(self, param1, param2):
    """Class methods are similar to regular functions.

    Note
    ----
    Do not include the `self` parameter in the ``Parameters`` section.

    Parameters
    -----
    param1
        The first parameter.
    param2
        The second parameter.

    Returns
    -----
    bool
        True if successful, False otherwise.

    """
    return True

def __special__(self):
    """By default special members with docstrings are not included.

    Special members are any methods or attributes that start with and
    end with a double underscore. Any special member with a docstring
    will be included in the output, if
    ``napoleon_include_special_with_doc`` is set to True.

    This behavior can be enabled by changing the following setting in
    Sphinx's conf.py::

        napoleon_include_special_with_doc = True

    """
    pass

def __special_without_docstring__(self):
    pass

def _private(self):
    """By default private members are not included.

    Private members are any methods or attributes that start with an
    underscore and are not special. By default they are not included
    in the output.

    This behavior can be changed such that private members are included
    by changing the following setting in Sphinx's conf.py::

```

```
        napoleon_include_private_with_doc = True

        """
        pass

    def _private_without_docstring(self):
        pass
```

### 3.1.2 Bug Reports

Bug reports are hugely important! Before you raise one, though, please check through the [GitHub issues](#), **both open and closed**, to confirm that the bug hasn't been reported before. Duplicate bug reports are a huge drain on the time of other contributors, and should be avoided as much as possible.

## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### O

- `openest.generate.calculation`, 5
- `openest.generate.curvegen`, 7
- `openest.generate.latextools`, 8
- `openest.generate.retrieve`, 8
- `openest.generate.shortterm`, 9
- `openest.generate.weathertools`, 10
- `openest.lincombo.continuous_sampled`, 10
- `openest.lincombo.helpers`, 11
- `openest.lincombo.hiernorm`, 11
- `openest.lincombo.hierregress`, 11
- `openest.lincombo.montecarlo`, 12
- `openest.lincombo.multi_delta`, 12
- `openest.lincombo.multi_draws`, 12
- `openest.lincombo.multi_normal`, 12
- `openest.lincombo.multi_sampled`, 12
- `openest.lincombo.multi_uniform`, 13
- `openest.lincombo.pooling`, 13
- `openest.models.bin_model`, 13
- `openest.models.curve`, 15
- `openest.models.ddp_model`, 16
- `openest.models.delta_model`, 19
- `openest.models.distribution_model`, 20
- `openest.models.features_interpreter`, 20
- `openest.models.generate`, 21
- `openest.models.hierarchical_normal`, 21
- `openest.models.integral_model`, 21
- `openest.models.mean_size_model`, 22
- `openest.models.memoizable`, 22
- `openest.models.model`, 23
- `openest.models.multivariate_model`, 24
- `openest.models.outer_multi_model`, 24
- `openest.models.parameter`, 25
- `openest.models.spline_model`, 25
- `openest.models.sum_multi_model`, 27
- `openest.models.univariate_model`, 28
- `openest.swapbin`, 29
  - `openest.swapbin.swapmodel`, 28
  - `openest.swapbin.transform`, 28



## A

**add\_conditional()** (openest.models.spline\_model.SplineModelConditional method), 25  
**add\_segment()** (openest.models.spline\_model.SplineModelConditional method), 26  
**add\_to\_y()** (openest.models.ddp\_model.DDPModel method), 17  
**alpha\_given\_taus()** (in module openest.lincombo.hiernorm), 11  
**any\_from\_url()** (in module openest.generate.retrieve), 8  
**Application** (class in openest.generate.calculation), 5  
**ApplicationByChunks** (class in openest.generate.calculation), 5  
**ApplicationByIrregular** (class in openest.generate.calculation), 5  
**ApplicationByYear** (class in openest.generate.calculation), 5  
**ApplicationEach** (class in openest.generate.calculation), 6  
**ApplicationPassCall** (class in openest.generate.calculation), 6  
**apply()** (openest.generate.calculation.Calculation method), 6  
**apply()** (openest.generate.calculation.CustomFunctionalCalculation method), 6  
**apply()** (openest.generate.calculation.FunctionalCalculation method), 7  
**apply()** (openest.generate.shortterm.InstaZScoreApply method), 9  
**apply()** (openest.generate.shortterm.MonthlyClimateApply method), 9  
**apply()** (openest.generate.shortterm.MonthlyZScoreApply method), 9  
**apply()** (openest.generate.shortterm.SingleWeatherApply method), 10  
**apply()** (openest.generate.shortterm.SplitByMonth method), 10  
**apply\_as\_distribution()** (openest.models.distribution\_model.DistributionModel method), 20  
**approximate\_mean()** (openest.models.spline\_model.SplineModelConditional static method), 26  
**approximate\_sum()** (openest.models.spline\_model.SplineModelConditional static method), 26  
**ascinv()** (openest.models.spline\_model.SplineModelConditional static method), 26  
**Attribute** (class in openest.models.model), 23  
**attribute\_list()** (openest.models.mean\_size\_model.MeanSizeModel method), 22  
**attribute\_list()** (openest.models.model.Model method), 23

## B

**best\_knot()** (openest.models.features\_interpreter.FeaturesInterpreter static method), 20  
**best\_spline()** (openest.models.features\_interpreter.FeaturesInterpreter static method), 20  
**betahat\_given\_tau()** (in module openest.lincombo.hierregress), 11  
**betahat\_given\_taus()** (in module openest.lincombo.hiernorm), 11  
**BinModel** (class in openest.models.bin\_model), 13

## C

**Calculation** (class in openest.generate.calculation), 6  
**call()** (in module openest.generate.latextools), 8  
**cdf()** (openest.models.bin\_model.BinModel method), 14  
**cdf()** (openest.models.delta\_model.DeltaModel method), 19  
**cdf()** (openest.models.spline\_model.SplineModel method), 25  
**cdf()** (openest.models.spline\_model.SplineModelConditional method), 27  
**check\_arguments()** (in module openest.lincombo.helpers), 11  
**choose\_model()** (in module openest.generate.retrieve), 8

[cleanup\(\)](#) (openest.generate.calculation.Calculation method), 22  
[cleanup\(\)](#) (openest.generate.calculation.FunctionalCalculation method), 22  
[ClippedCurve](#) (class in openest.models.curve), 15  
[CoefficientsCurve](#) (class in openest.models.curve), 15  
[column\\_info\(\)](#) (openest.generate.calculation.Calculation method), 6  
[column\\_info\(\)](#) (openest.generate.shortterm.InstaZScoreApply method), 9  
[column\\_info\(\)](#) (openest.generate.shortterm.MonthlyClimateApply method), 9  
[column\\_info\(\)](#) (openest.generate.shortterm.MonthlyZScoreApply method), 9  
[column\\_info\(\)](#) (openest.generate.shortterm.SingleWeatherApply method), 10  
[column\\_info\(\)](#) (openest.generate.shortterm.SplitByMonth method), 10  
[combine\(\)](#) (openest.models.bin\_model.BinModel static method), 14  
[combine\(\)](#) (openest.models.ddp\_model.DDPModel static method), 17  
[combine\(\)](#) (openest.models.delta\_model.DeltaModel static method), 19  
[combine\(\)](#) (openest.models.mean\_size\_model.MeanSizeModel static method), 22  
[combine\(\)](#) (openest.models.model.Model static method), 23  
[combine\(\)](#) (openest.models.spline\_model.SplineModel static method), 25  
[combiners](#) (openest.models.model.Model attribute), 23  
[combo\\_effects\(\)](#) (in module openest.generate.weathertools), 10  
[condition\(\)](#) (openest.models.multivariate\_model.MultivariateModel method), 24  
[condition\(\)](#) (openest.models.outer\_multi\_model.OuterMultiModel method), 24  
[consistent\\_bins\(\)](#) (openest.models.bin\_model.BinModel static method), 14  
[ConstantCurveGenerator](#) (class in openest.generate.curvegen), 7  
[ContinuousSampled](#) (class in openest.lincombo.continuous\_sampled), 10  
[convolve\(\)](#) (openest.models.spline\_model.SplineModelConditional method), 27  
[copy\(\)](#) (openest.models.bin\_model.BinModel method), 14  
[copy\(\)](#) (openest.models.ddp\_model.DDPModel method), 17  
[copy\(\)](#) (openest.models.delta\_model.DeltaModel method), 19  
[copy\(\)](#) (openest.models.integral\_model.IntegralModel method), 21  
[copy\(\)](#) (openest.models.mean\_size\_model.MeanSizeModel method), 22  
[copy\(\)](#) (openest.models.memoizable.MemoizedUnivariate method), 22  
[copy\(\)](#) (openest.models.model.Model method), 23  
[copy\(\)](#) (openest.models.spline\_model.SplineModel method), 25  
[copy\(\)](#) (openest.models.spline\_model.SplineModelConditional method), 27  
[create\\_gaussian\(\)](#) (openest.models.spline\_model.SplineModel static method), 26  
[create\\_lin\(\)](#) (openest.models.ddp\_model.DDPModel static method), 17  
[create\\_single\(\)](#) (openest.models.spline\_model.SplineModel static method), 26  
[CubicSplineCurve](#) (class in openest.models.curve), 15  
[CurveCurve](#) (class in openest.models.curve), 15  
[CurveGenerator](#) (class in openest.generate.curvegen), 7  
[CustomFunctionalCalculation](#) (class in openest.generate.calculation), 6

## D

[date\\_to\\_datestr\(\)](#) (in module openest.generate.weathertools), 10  
[ddp\\_from\\_url\(\)](#) (in module openest.generate.retrieve), 9  
[DDPModel](#) (class in openest.models.ddp\_model), 16  
[default\\_condition\(\)](#) (openest.models.multivariate\_model.MultivariateModel method), 24  
[default\\_condition\(\)](#) (openest.models.outer\_multi\_model.OuterMultiModel method), 24  
[DelayedCurveGenerator](#) (class in openest.generate.curvegen), 7  
[DeltaModel](#) (class in openest.models.delta\_model), 19  
[delta\\_model\(\)](#) (openest.models.parameter.ParameterBase method), 25  
[describe\(\)](#) (openest.generate.calculation.Calculation static method), 6  
[describe\(\)](#) (openest.generate.shortterm.InstaZScoreApply static method), 9  
[describe\(\)](#) (openest.generate.shortterm.MonthlyClimateApply static method), 9  
[describe\(\)](#) (openest.generate.shortterm.MonthlyZScoreApply static method), 9  
[describe\(\)](#) (openest.generate.shortterm.SingleWeatherApply static method), 10  
[describe\(\)](#) (openest.generate.shortterm.SplitByMonth static method), 10  
[dims\(\)](#) (openest.models.outer\_multi\_model.OuterMultiModel method), 24  
[DistributionModel](#) (class in openest.models.distribution\_model), 20



done() (openest.generate.calculation.Application method), 5

done() (openest.generate.calculation.ApplicationEach method), 6

done() (openest.generate.calculation.CustomFunctionalCalculation method), 6

donehandler() (openest.generate.calculation.CustomFunctionalCalculation method), 6

draw\_from\_counts() (in module openest.models.hierarchical\_normal), 21

draw\_sample() (openest.models.bin\_model.BinModel method), 14

draw\_sample() (openest.models.ddp\_model.DDPModel method), 17

draw\_sample() (openest.models.delta\_model.DeltaModel method), 19

draw\_sample() (openest.models.model.Model method), 23

draw\_sample() (openest.models.spline\_model.SplineModel method), 26

draw\_sample() (openest.models.spline\_model.SplineModelConditionalFeaturesInterpreter method), 27

eval\_pvals() (openest.models.curve.UnivariateCurve method), 16

eval\_pvals() (openest.models.memoizable.MemoizedUnivariate method), 23

eval\_pvals() (openest.models.spline\_model.SplineModelConditionalFeaturesInterpreter method), 27

eval\_pvals\_spline() (openest.models.features\_interpreter.FeaturesInterpreter static method), 20

## F

features\_to\_exponential() (openest.models.features\_interpreter.FeaturesInterpreter static method), 20

features\_to\_gaussian() (openest.models.features\_interpreter.FeaturesInterpreter static method), 20

features\_to\_uniform() (openest.models.features\_interpreter.FeaturesInterpreter static method), 20

features\_to\_uniform() (openest.models.features\_interpreter.FeaturesInterpreter static method), 20

features\_to\_uniform() (openest.models.features\_interpreter.FeaturesInterpreter static method), 20

## E

english_function()	(in module open-	filter_x()	(openest.models.ddp_model.DDPModel
est.generate.latextools), 8			method), 17
estimated_maxlintaus()	(in module open-	filter_x()	(openest.models.delta_model.DeltaModel
est.lincombo.pooling), 13			method), 19
estimated_maxtau()	(in module open-	filter_x()	(openest.models.mean_size_model.MeanSizeModel
est.lincombo.pooling), 13			method), 22
eval_pval()	(openest.models.bin_model.BinModel	filter_x()	(openest.models.spline_model.SplineModel
method), 14			method), 26
eval_pval()	(openest.models.curve.UnivariateCurve	filter_x()	(openest.models.univariate_model.UnivariateModel
method), 16			method), 28
eval_pval()	(openest.models.ddp_model.DDPModel	find_bins()	(in module openest.swapbin.swapmodel), 28
method), 17		find_mode()	(openest.models.spline_model.SplineModelConditional
eval_pval()	(openest.models.integral_model.IntegralModel	method), 27	
method), 21		find_nearest()	(openest.models.spline_model.SplineModelConditional
eval_pval()	(openest.models.memoizable.MemoizedUnivariate	static method), 27	
method), 23		FlatCurve	(class in openest.models.curve), 15
eval_pval()	(openest.models.model.Model method), 23	float_condition()	(open-
eval_pval()	(openest.models.spline_model.SplineModel	est.models.outer_multi_model.OuterMultiModel	
method), 26		method), 24	
eval_pval_index()	(openest.models.bin_model.BinModel	format()	(openest.generate.calculation.Calculation
method), 14		method), 6	
eval_pval_index()	(open-	format()	(openest.generate.calculation.FunctionalCalculation
est.models.ddp_model.DDPModel method),	17	format_call()	(openest.generate.curvegen.ConstantCurveGenerator
eval_pval_index()	(open-	method), 7	
est.models.memoizable.MemoizableUnivariate		format_call()	(openest.generate.curvegen.CurveGenerator
method), 22		method), 7	
eval_pval_index()	(open-	format_call()	(openest.generate.curvegen.DelayedCurveGenerator
est.models.spline_model.SplineModel		method), 7	
method), 26			

format\_call() (openest.generate.curvegen.TransformCurveGenerator method), 7  
 format\_handler() (openest.generate.calculation.FunctionalCalculation method), 7  
 from\_ddp() (openest.models.spline\_model.SplineModel static method), 26  
 from\_file() (openest.models.ddp\_model.DDPModel static method), 17  
 from\_url() (in module openest.generate.retrieve), 9  
 FunctionalCalculation (class in openest.generate.calculation), 7

## G

gaussian\_mean() (openest.models.spline\_model.SplineModelConditional method), 27  
 gaussian\_sdev() (openest.models.spline\_model.SplineModelConditional method), 27  
 generate\_thetas() (in module openest.models.hierarchical\_normal), 21  
 get\_attribute() (openest.models.mean\_size\_model.MeanSizeModel method), 22  
 get\_attribute() (openest.models.model.Model method), 23  
 get\_bin\_at() (openest.models.bin\_model.BinModel method), 14  
 get\_closest() (openest.models.ddp\_model.DDPModel method), 18  
 get\_conditional() (openest.models.spline\_model.SplineModel method), 26  
 get\_crop\_calendar() (in module openest.generate.weathertools), 10  
 get\_curve() (openest.generate.curvegen.ConstantCurveGenerator method), 7  
 get\_curve() (openest.generate.curvegen.CurveGenerator method), 7  
 get\_curve() (openest.generate.curvegen.DelayedCurveGenerator method), 7  
 get\_curve() (openest.generate.curvegen.TransformCurveGenerator method), 7  
 get\_edges() (openest.models.bin\_model.BinModel method), 14  
 get\_edges() (openest.models.memoizable.MemoizableUnivariate method), 22  
 get\_eval\_pval\_spline() (openest.models.memoizable.MemoizedUnivariate method), 23  
 get\_index() (openest.models.memoizable.MemoizedUnivariate method), 23  
 get\_indexes() (openest.models.memoizable.MemoizedUnivariate method), 23  
 get\_mean() (openest.models.bin\_model.BinModel method), 14  
 get\_mean() (openest.models.ddp\_model.DDPModel method), 18  
 get\_mean() (openest.models.mean\_size\_model.MeanSizeModel method), 22  
 get\_mean() (openest.models.model.Model method), 23  
 get\_mean() (openest.models.spline\_model.SplineModel method), 26  
 get\_next\_curve() (openest.generate.curvegen.DelayedCurveGenerator method), 7  
 get\_pval() (openest.models.spline\_model.SplineModelConditional method), 27  
 get\_random() (in module openest.models.hierarchical\_normal), 21  
 get\_sampled\_column() (in module openest.lincombo.hiernorm), 11  
 get\_sampled\_column() (in module openest.lincombo.hierregress), 11  
 get\_sdev() (openest.models.bin\_model.BinModel method), 14  
 get\_sdev() (openest.models.ddp\_model.DDPModel method), 18  
 get\_sdev() (openest.models.mean\_size\_model.MeanSizeModel method), 22  
 get\_sdev() (openest.models.model.Model method), 24  
 get\_sdev() (openest.models.spline\_model.SplineModel method), 26  
 get\_terms() (openest.models.curve.CubicSplineCurve method), 15  
 get\_xx() (openest.models.bin\_model.BinModel method), 14  
 get\_xx() (openest.models.curve.UnivariateCurve method), 16  
 get\_xx() (openest.models.ddp\_model.DDPModel method), 18  
 get\_xx() (openest.models.integral\_model.IntegralModel method), 21  
 get\_xx() (openest.models.memoizable.MemoizedUnivariate method), 23  
 get\_xx() (openest.models.spline\_model.SplineModel method), 26  
 get\_xx() (openest.models.univariate\_model.UnivariateModel method), 28  
 get\_xx\_centers() (openest.models.bin\_model.BinModel method), 14  
 get\_yy() (openest.models.ddp\_model.DDPModel method), 18  
 growing\_seasons\_daily\_ncdf() (in module openest.generate.weathertools), 10  
 growing\_seasons\_mean\_ncdf() (in module openest.generate.weathertools), 10  
 growing\_seasons\_mean\_reader() (in module openest.generate.weathertools), 10

est.generate.weathertools), 10

guess\_ranges() (openest.lincombo.continuous\_sampled.ContinuousSampled method), 10

guess\_ranges() (openest.lincombo.multi\_sampled.MultivariateSampled method), 12

guess\_ranges\_gridded() (openest.lincombo.continuous\_sampled.ContinuousSampled method), 10

guess\_ranges\_gridded() (openest.lincombo.multi\_sampled.MultivariateSampled method), 12

## H

handler() (openest.generate.calculation.FunctionalCalculation method), 7

helper\_params() (in module openest.models.hierarchical\_normal), 21

## I

init\_apply() (openest.generate.calculation.CustomFunctionalCalculation method), 6

init\_from() (openest.models.ddp\_model.DDPModel method), 18

init\_from\_bin\_file() (openest.models.bin\_model.BinModel method), 14

init\_from\_delta\_file() (openest.models.delta\_model.DeltaModel method), 19

init\_from\_feature\_file() (openest.models.features\_interpreter.FeaturesInterpreter static method), 20

init\_from\_mean\_size\_file() (openest.models.mean\_size\_model.MeanSizeModel method), 22

init\_from\_other() (openest.models.ddp\_model.DDPModel method), 18

init\_from\_spline\_file() (openest.models.spline\_model.SplineModel method), 26

init\_from\_union() (openest.models.outer\_multi\_model.OuterMultiModel method), 24

InstaZScoreApply (class in openest.generate.shortterm), 9

IntegralModel (class in openest.models.integral\_model), 21

interpolate\_x() (openest.models.bin\_model.BinModel method), 14

interpolate\_x() (openest.models.ddp\_model.DDPModel method), 18

interpolate\_x() (openest.models.delta\_model.DeltaModel method), 19

interpolate\_x() (openest.models.integral\_model.IntegralModel method), 21

interpolate\_x() (openest.models.mean\_size\_model.MeanSizeModel method), 22

interpolate\_x() (openest.models.memoizable.MemoizedUnivariate method), 23

interpolate\_x() (openest.models.spline\_model.SplineModel method), 26

interpolate\_x() (openest.models.univariate\_model.UnivariateModel method), 28

interpolate\_y() (openest.models.ddp\_model.DDPModel method), 18

intersect\_get\_model() (openest.models.univariate\_model.UnivariateModel static method), 28

intersect\_get\_x() (openest.models.univariate\_model.UnivariateModel static method), 28

intersect\_x() (openest.models.univariate\_model.UnivariateModel static method), 28

intersect\_x\_all() (openest.models.univariate\_model.UnivariateModel static method), 28

is\_gaussian() (openest.models.spline\_model.SplineModel method), 26

is\_gaussian() (openest.models.spline\_model.SplineModelConditional method), 27

issparse() (in module openest.lincombo.helpers), 11

## K

kind() (openest.models.bin\_model.BinModel method), 14

kind() (openest.models.ddp\_model.DDPModel method), 18

kind() (openest.models.delta\_model.DeltaModel method), 19

kind() (openest.models.integral\_model.IntegralModel method), 21

kind() (openest.models.mean\_size\_model.MeanSizeModel method), 22

kind() (openest.models.memoizable.MemoizedUnivariate method), 23

kind() (openest.models.model.Model method), 24

kind() (openest.models.outer\_multi\_model.OuterMultiModel method), 24

kind() (openest.models.spline\_model.SplineModel method), 26

## L

latex\_function() (in module openest.generate.latextools), 8

lin\_p() (openest.models.ddp\_model.DDPModel method), 18

lincombo\_hiernorm\_taubyalpha() (in module openest.lincombo.hiernorm), 11

[lincombo\\_hiernorm\\_taubybeta\(\)](#) (in module `openest.lincombo.hiernorm`), 11  
[lincombo\\_hierregress\(\)](#) (in module `openest.lincombo.hierregress`), 11  
[lincombo\\_hierregress\\_taubybeta\(\)](#) (in module `openest.lincombo.hierregress`), 11  
[lincombo\\_hierregress\\_taubymu\(\)](#) (in module `openest.lincombo.hierregress`), 11  
[lincombo\\_pooled\(\)](#) (in module `openest.lincombo.pooling`), 13  
[LinearCurve](#) (class in `openest.models.curve`), 15  
[log\\_p\(\)](#) (`openest.models.ddp_model.DDPModel` method), 18  
[logpdf\(\)](#) (`openest.lincombo.multi_normal.MultivariateNormal` method), 12

## M

[make\\_conditional\(\)](#) (`openest.models.features_interpreter.FeaturesInterpreter` static method), 21  
[make\\_conditional\\_from\\_spline\(\)](#) (`openest.models.spline_model.SplineModelConditional` static method), 27  
[make\\_conditional\\_respecting\(\)](#) (`openest.models.features_interpreter.FeaturesInterpreter` static method), 21  
[make\\_gaussian\(\)](#) (`openest.models.spline_model.SplineModelConditional` static method), 27  
[make\\_linear\\_spline\\_curve\(\)](#) (`openest.models.curve.CurveCurve` static method), 15  
[make\\_single\(\)](#) (`openest.models.spline_model.SplineModelConditional` static method), 27  
[maxs\(\)](#) (`openest.lincombo.multi_uniform.MultivariateUniform` method), 13  
[mean\(\)](#) (`openest.lincombo.multi_draws.MultivariateDraws` method), 12  
[MeanSizeModel](#) (class in `openest.models.mean_size_model`), 22  
[MemoizableUnivariate](#) (class in `openest.models.memoizable`), 22  
[MemoizedUnivariate](#) (class in `openest.models.memoizable`), 22  
[merge\(\)](#) (`openest.models.bin_model.BinModel` static method), 14  
[merge\(\)](#) (`openest.models.ddp_model.DDPModel` static method), 18  
[merge\(\)](#) (`openest.models.delta_model.DeltaModel` static method), 19  
[merge\(\)](#) (`openest.models.mean_size_model.MeanSizeModel` static method), 22  
[merge\(\)](#) (`openest.models.model.Model` static method), 24  
[merge\(\)](#) (`openest.models.spline_model.SplineModel` static method), 26  
[mergers](#) (`openest.models.model.Model` attribute), 24  
[MinimumCurve](#) (class in `openest.models.curve`), 15  
[mins\(\)](#) (`openest.lincombo.multi_uniform.MultivariateUniform` method), 13  
[Model](#) (class in `openest.models.model`), 23  
[MonthlyClimateApply](#) (class in `openest.generate.shortterm`), 9  
[MonthlyZScoreApply](#) (class in `openest.generate.shortterm`), 9  
[mu\\_given\\_tau\(\)](#) (in module `openest.lincombo.hierregress`), 11  
[MultivariateDelta](#) (class in `openest.lincombo.multi_delta`), 12  
[MultivariateDraws](#) (class in `openest.lincombo.multi_draws`), 12  
[MultivariateModel](#) (class in `openest.models.multivariate_model`), 24  
[MultivariateNormal](#) (class in `openest.lincombo.multi_normal`), 12  
[MultivariateSampled](#) (class in `openest.lincombo.multi_sampled`), 12  
[MultivariateUniform](#) (class in `openest.lincombo.multi_uniform`), 13

## N

[neglnf](#) (`openest.models.spline_model.SplineModel` attribute), 26  
[nongaussian\\_x2px\(\)](#) (`openest.models.spline_model.SplineModelConditional` method), 27  
[nongaussian\\_xpx\(\)](#) (`openest.models.spline_model.SplineModelConditional` method), 27  
[numvars\(\)](#) (`openest.models.multivariate_model.MultivariateModel` method), 24

## O

[openest.generate.calculation](#) (module), 5  
[openest.generate.curvegen](#) (module), 7  
[openest.generate.latextools](#) (module), 8  
[openest.generate.retrieve](#) (module), 8  
[openest.generate.shortterm](#) (module), 9  
[openest.generate.weathertools](#) (module), 10  
[openest.lincombo.continuous\\_sampled](#) (module), 10  
[openest.lincombo.helpers](#) (module), 11  
[openest.lincombo.hiernorm](#) (module), 11  
[openest.lincombo.hierregress](#) (module), 11  
[openest.lincombo.montecarlo](#) (module), 12  
[openest.lincombo.multi\\_delta](#) (module), 12  
[openest.lincombo.multi\\_draws](#) (module), 12  
[openest.lincombo.multi\\_normal](#) (module), 12  
[openest.lincombo.multi\\_sampled](#) (module), 12

[openest.lincombo.multi\\_uniform \(module\)](#), 13  
[openest.lincombo.pooling \(module\)](#), 13  
[openest.models.bin\\_model \(module\)](#), 13  
[openest.models.curve \(module\)](#), 15  
[openest.models.ddp\\_model \(module\)](#), 16  
[openest.models.delta\\_model \(module\)](#), 19  
[openest.models.distribution\\_model \(module\)](#), 20  
[openest.models.features\\_interpreter \(module\)](#), 20  
[openest.models.generate \(module\)](#), 21  
[openest.models.hierarchical\\_normal \(module\)](#), 21  
[openest.models.integral\\_model \(module\)](#), 21  
[openest.models.mean\\_size\\_model \(module\)](#), 22  
[openest.models.memoizable \(module\)](#), 22  
[openest.models.model \(module\)](#), 23  
[openest.models.multivariate\\_model \(module\)](#), 24  
[openest.models.outer\\_multi\\_model \(module\)](#), 24  
[openest.models.parameter \(module\)](#), 25  
[openest.models.spline\\_model \(module\)](#), 25  
[openest.models.sum\\_multi\\_model \(module\)](#), 27  
[openest.models.univariate\\_model \(module\)](#), 28  
[openest.swapbin \(module\)](#), 29  
[openest.swapbin.swapmodel \(module\)](#), 28  
[openest.swapbin.transform \(module\)](#), 28  
[OtherClippedCurve \(class in openest.models.curve\)](#), 15  
[OuterMultiModel \(class in openest.models.outer\\_multi\\_model\)](#), 24

## P

[p\\_tau\\_given\\_y\(\) \(in module openest.models.hierarchical\\_normal\)](#), 21  
[ParameterBase \(class in openest.models.parameter\)](#), 25  
[partial\\_cdf\(\) \(openest.models.spline\\_model.SplineModelConditional static method\)](#), 27  
[pdf\(\) \(openest.lincombo.continuous\\_sampled.ContinuousSampled method\)](#), 10  
[pdf\(\) \(openest.lincombo.multi\\_delta.MultivariateDelta method\)](#), 12  
[pdf\(\) \(openest.lincombo.multi\\_normal.MultivariateNormal method\)](#), 12  
[pdf\(\) \(openest.lincombo.multi\\_sampled.MultivariateSampled method\)](#), 12  
[pdf\(\) \(openest.lincombo.multi\\_uniform.MultivariateUniform method\)](#), 13  
[PiecewiseCurve \(class in openest.models.curve\)](#), 16  
[polynomial\(\) \(in module openest.models.generate\)](#), 21  
[pos\(\) \(in module openest.models.curve\)](#), 16  
[posinf \(openest.models.spline\\_model.SplineModel attribute\)](#), 26  
[prepare\\_draws\(\) \(openest.lincombo.continuous\\_sampled.ContinuousSampled method\)](#), 10  
[prepare\\_draws\(\) \(openest.lincombo.multi\\_sampled.MultivariateSampled method\)](#), 12

[prepare\\_draws\\_gridded\(\) \(openest.lincombo.continuous\\_sampled.ContinuousSampled method\)](#), 10  
[prepare\\_draws\\_gridded\(\) \(openest.lincombo.multi\\_sampled.MultivariateSampled method\)](#), 13  
[probability\\_tau\(\) \(in module openest.lincombo.hiernorm\)](#), 11  
[probability\\_tau\(\) \(in module openest.lincombo.hierregress\)](#), 11  
[ProductCurve \(class in openest.models.curve\)](#), 16  
[propose\\_grid\(\) \(openest.models.spline\\_model.SplineModelConditional static method\)](#), 27  
[push\(\) \(openest.generate.calculation.Application method\)](#), 5  
[push\(\) \(openest.generate.calculation.ApplicationByChunks method\)](#), 5  
[push\(\) \(openest.generate.calculation.ApplicationByIrregular method\)](#), 5  
[push\(\) \(openest.generate.calculation.ApplicationEach method\)](#), 6  
[push\(\) \(openest.generate.calculation.ApplicationPassCall method\)](#), 6  
[push\(\) \(openest.generate.calculation.CustomFunctionalCalculation method\)](#), 6  
[push\(\) \(openest.generate.shortterm.InstaZScoreApply method\)](#), 9  
[push\(\) \(openest.generate.shortterm.MonthlyZScoreApply method\)](#), 9  
[push\\_saved\(\) \(openest.generate.calculation.ApplicationByChunks method\)](#), 5  
[push\\_saved\(\) \(openest.generate.calculation.ApplicationByYear method\)](#), 6  
[push\\_handler\(\) \(openest.generate.calculation.CustomFunctionalCalculation method\)](#), 7

## R

[re\\_condition\(\) \(openest.models.outer\\_multi\\_model.OuterMultiModel static method\)](#), 25  
[re\\_numeric \(openest.models.outer\\_multi\\_model.OuterMultiModel attribute\)](#), 25  
[read\\_scale\\_file\(\) \(in module openest.generate.weathertools\)](#), 10  
[recategorize\\_x\(\) \(openest.models.ddp\\_model.DDPModel method\)](#), 18  
[recategorize\\_x\(\) \(openest.models.spline\\_model.SplineModel method\)](#), 26  
[recategorize\\_x\(\) \(openest.models.univariate\\_model.UnivariateModel method\)](#), 28  
[regress\\_distribution\(\) \(in module openest.lincombo.montecarlo\)](#), 12

`regress_draws()` (in module `openest.lincombo.montecarlo`), 12  
`regress_summary()` (in module `openest.lincombo.montecarlo`), 12  
`rescale()` (`openest.models.ddp_model.DDPModel` method), 18  
`rescale()` (`openest.models.spline_model.SplineModelConditional` method), 27  
`reset_cache()` (`openest.models.memoizable.MemoizedUnivariate` method), 23  
`rough_limits()` (`openest.models.spline_model.SplineModelConditional` method), 27  
`rough_span()` (`openest.models.spline_model.SplineModelConditional` method), 27  
`rvs()` (`openest.lincombo.continuous_sampled.ContinuousSampled` method), 11  
`rvs()` (`openest.lincombo.multi_delta.MultivariateDelta` method), 12  
`rvs()` (`openest.lincombo.multi_draws.MultivariateDraws` method), 12  
`rvs()` (`openest.lincombo.multi_normal.MultivariateNormal` method), 12  
`rvs()` (`openest.lincombo.multi_sampled.MultivariateSampled` method), 13  
`rvs()` (`openest.lincombo.multi_uniform.MultivariateUniform` method), 13  
  
**S**  
`sample_posterior()` (in module `openest.lincombo.hiernorm`), 11  
`sample_posterior()` (in module `openest.lincombo.hierregress`), 12  
`samples` (`openest.models.spline_model.SplineModel` attribute), 26  
`scale()` (`openest.models.spline_model.SplineModelConditional` method), 27  
`scale_p()` (`openest.models.bin_model.BinModel` method), 14  
`scale_p()` (`openest.models.ddp_model.DDPModel` method), 19  
`scale_p()` (`openest.models.delta_model.DeltaModel` method), 19  
`scale_p()` (`openest.models.integral_model.IntegralModel` method), 21  
`scale_p()` (`openest.models.mean_size_model.MeanSizeModel` method), 22  
`scale_p()` (`openest.models.memoizable.MemoizedUnivariate` method), 23  
`scale_p()` (`openest.models.model.Model` method), 24  
`scale_p()` (`openest.models.outer_multi_model.OuterMultiModel` method), 25  
`scale_p()` (`openest.models.spline_model.SplineModel` method), 26  
`scale_p()` (`openest.models.spline_model.SplineModelConditional` method), 27  
`scale_y()` (`openest.models.bin_model.BinModel` method), 15  
`scale_y()` (`openest.models.ddp_model.DDPModel` method), 19  
`scale_y()` (`openest.models.delta_model.DeltaModel` method), 19  
`scale_y()` (`openest.models.integral_model.IntegralModel` method), 21  
`scale_y()` (`openest.models.mean_size_model.MeanSizeModel` method), 22  
`scale_y()` (`openest.models.memoizable.MemoizedUnivariate` method), 23  
`scale_y()` (`openest.models.model.Model` method), 24  
`scale_y()` (`openest.models.spline_model.SplineModel` method), 26  
`scale_y()` (`openest.models.spline_model.SplineModelConditional` method), 27  
`segment_max()` (`openest.models.spline_model.SplineModelConditional` method), 27  
`SelectiveInputCurve` (class in `openest.models.curve`), 16  
`set_x_cache_decimals()` (`openest.models.memoizable.MemoizedUnivariate` method), 23  
`ShiftedCurve` (class in `openest.models.curve`), 16  
`simulate_normal_model()` (in module `openest.models.hierarchical_normal`), 21  
`SingleWeatherApply` (class in `openest.generate.shortterm`), 9  
`size()` (`openest.models.spline_model.SplineModelConditional` method), 27  
`skew_gaussian_construct()` (`openest.models.features_interpreter.FeaturesInterpreter` static method), 21  
`skew_gaussian_evaluate()` (`openest.models.features_interpreter.FeaturesInterpreter` static method), 21  
`spline_from_url()` (in module `openest.generate.retrieve`), 9  
`SplineModel` (class in `openest.models.spline_model`), 25  
`SplineModelConditional` (class in `openest.models.spline_model`), 26  
`SplitByMonth` (class in `openest.generate.shortterm`), 10  
`std()` (`openest.lincombo.multi_draws.MultivariateDraws` method), 12  
`StepCurve` (class in `openest.models.curve`), 16  
`sum_multiply()` (in module `openest.lincombo.pooling`), 13  
`sum_multiply2()` (in module `openest.lincombo.pooling`), 13  
`SumMultiModel` (class in `openest.models.sum_multi_model`), 27  
`swap_any()` (in module `openest.swapbin.swapmodel`), 28



swap\_beta() (in module openest.swapbin.transform), 28  
 swap\_bin() (in module openest.swapbin.swapmodel), 28  
 swap\_spline() (in module openest.swapbin.swapmodel), 28  
 swap\_values() (in module openest.swapbin.swapmodel), 28  
 swap\_vcv() (in module openest.swapbin.transform), 28

## T

test() (openest.generate.calculation.Calculation method), 6  
 to\_ddp() (openest.models.bin\_model.BinModel method), 15  
 to\_ddp() (openest.models.ddp\_model.DDPModel method), 19  
 to\_ddp() (openest.models.spline\_model.SplineModel method), 26  
 to\_points() (openest.models.spline\_model.SplineModelConditional method), 27  
 to\_points\_at() (openest.models.bin\_model.BinModel method), 15  
 to\_points\_at() (openest.models.delta\_model.DeltaModel method), 19  
 to\_points\_at() (openest.models.model.Model method), 24  
 to\_points\_at() (openest.models.spline\_model.SplineModel method), 26  
 transform() (in module openest.swapbin.transform), 28  
 TransformCurveGenerator (class in openest.generate.curvegen), 7  
 transpose() (openest.models.ddp\_model.DDPModel method), 19

## U

uniform\_constant() (in module openest.models.generate), 21  
 uniform\_doseless() (in module openest.models.generate), 21  
 UnivariateCurve (class in openest.models.curve), 16  
 UnivariateModel (class in openest.models.univariate\_model), 28

## V

vals() (openest.lincombo.multi\_delta.MultivariateDelta method), 12

## W

write() (openest.models.bin\_model.BinModel method), 15  
 write() (openest.models.ddp\_model.DDPModel method), 19  
 write() (openest.models.delta\_model.DeltaModel method), 19  
 write() (openest.models.integral\_model.IntegralModel method), 21

write() (openest.models.mean\_size\_model.MeanSizeModel method), 22  
 write() (openest.models.memoizable.MemoizedUnivariate method), 23  
 write() (openest.models.outer\_multi\_model.OuterMultiModel method), 25  
 write() (openest.models.spline\_model.SplineModel method), 26  
 write\_file() (openest.models.bin\_model.BinModel method), 15  
 write\_file() (openest.models.ddp\_model.DDPModel method), 19  
 write\_file() (openest.models.delta\_model.DeltaModel method), 19  
 write\_file() (openest.models.integral\_model.IntegralModel method), 21  
 write\_file() (openest.models.mean\_size\_model.MeanSizeModel method), 22  
 write\_file() (openest.models.memoizable.MemoizedUnivariate method), 23  
 write\_file() (openest.models.outer\_multi\_model.OuterMultiModel method), 25  
 write\_file() (openest.models.spline\_model.SplineModel method), 26  
 write\_gaussian() (openest.models.spline\_model.SplineModel method), 26  
 write\_gaussian\_plus() (openest.models.spline\_model.SplineModel method), 26

## X

xmap\_apply\_model() (in module openest.generate.weathertools), 10

## Y

yearly\_daily\_ncdf() (in module openest.generate.weathertools), 10

## Z

zero\_delta() (openest.models.delta\_model.DeltaModel static method), 19  
 ZeroInterceptPolynomialCurve (class in openest.models.curve), 16