

---

# **openest Documentation**

**Author**

**Nov 13, 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>37</b>
3.1	Contributor's Guide . . . . .	37
<b>4</b>	<b>Indices and tables</b>	<b>45</b>
	<b>Python Module Index</b>	<b>47</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*)

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

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

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

```
class openest.generate.calculation.ApplicationPassCall (region, subapp, handler,
    *handler_args, **handler_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
```

```
enable_deltamethod ()
    When applied, yield will contain arrays of coefficient multiplicands as a vector the length of the CSVV
    coefficients.
```

```
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)
    Returns an iterator of (yyyy, value, ...).
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 ()
enable_deltamethod ()
    When applied, yield will contain arrays of coefficient multiplicands as a vector the length of the CSVV
    coefficients.
format (lang, *args, **kwargs)
    Returns a dictionary of FormatElements. Only keys in the tree of dependencies will be output.
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)
    Returns an object of type Curve.
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)
    Returns an object of type Curve.
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)  
        Returns an object of type Curve.  
    get_lincom_terms (region, year, predictors)  
    get_lincom_terms_simple (predictors, covariates={})
```

## openest.generate.daily module

```
class openest.generate.daily.ApplyCurve (curvegen, unitses, names, titles, descriptions)  
    Bases: openest.generate.calculation.Calculation  
    apply (region, *args)  
    column_info ()  
        Returns an array of dictionaries, with 'name', 'title', and 'description'.  
    static describe ()  
  
class openest.generate.daily.AverageByMonth (model, units, func=<function <lambda>>, pval=0.5)  
    Bases: openest.generate.calculation.Calculation  
    apply (region)  
    column_info ()  
        Returns an array of dictionaries, with 'name', 'title', and 'description'.  
    static describe ()  
    format (lang)  
        Returns a dictionary of FormatElements. Only keys in the tree of dependencies will be output.  
  
class openest.generate.daily.MonthlyDayBins (model, units, pval=0.5, weather_change=<function <lambda>>)  
    Bases: openest.generate.calculation.Calculation  
    apply (region)  
    column_info ()  
        Returns an array of dictionaries, with 'name', 'title', and 'description'.  
    static describe ()  
    format (lang)  
        Returns a dictionary of FormatElements. Only keys in the tree of dependencies will be output.  
  
class openest.generate.daily.PercentWithin (endpoints)  
    Bases: openest.generate.calculation.Calculation  
    apply (region)  
    column_info ()  
        Returns an array of dictionaries, with 'name', 'title', and 'description'.  
    static describe ()
```

```

class openest.generate.daily.YearlyAverageDay (units, curvegen, curve_description,
                                              weather_change=<function <lambda>>,
                                              norecord=False)
    Bases: openest.generate.calculation.Calculation
    apply (region, *args)
    column_info ()
        Returns an array of dictionaries, with 'name', 'title', and 'description'.
    static describe ()
    format (lang)
        Returns a dictionary of FormatElements. Only keys in the tree of dependencies will be output.

class openest.generate.daily.YearlyDayBins (model, units, pval=0.5)
    Bases: openest.generate.calculation.Calculation
    apply (region, *args)
    column_info ()
        Returns an array of dictionaries, with 'name', 'title', and 'description'.
    static describe ()
    format (lang)
        Returns a dictionary of FormatElements. Only keys in the tree of dependencies will be output.

class openest.generate.daily.YearlyDividedPolynomialAverageDay (units, curvegen,
                                                                curve_description,
                                                                weather_change=<function
                                                                <lambda>>)

    Bases: openest.generate.calculation.Calculation
    apply (region, *args)
    column_info ()
        Returns an array of dictionaries, with 'name', 'title', and 'description'.
    static describe ()

class openest.generate.daily.YearlySumDay (units, curvegen, curve_description,
                                           weather_change=<function <lambda>>,
                                           norecord=False)
    Bases: openest.generate.daily.YearlyAverageDay
    apply (region, *args)
    column_info ()
        Returns an array of dictionaries, with 'name', 'title', and 'description'.
    static describe ()
    format (lang)
        Returns a dictionary of FormatElements. Only keys in the tree of dependencies will be output.

```

## openest.generate.functions module

```

class openest.generate.functions.AuxillaryResult (subcalc_main, subcalc_aux, aux-
                                                  name)
    Bases: openest.generate.calculation.Calculation
    Produce an additional output, but then pass the main result on.

```

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

**column\_info** ()  
Returns an array of dictionaries, with 'name', 'title', and 'description'.

**static describe** ()

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

**class** openest.generate.functions.**AuxillaryResultApplication** (*region*, *sub-*  
*app\_main*, *sub-*  
*app\_aux*)

Bases: [openest.generate.calculation.Application](#)

Perform both main and auxillary calculation, and order as main[0], aux, main[1:]

**done** ()

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

**class** openest.generate.functions.**ConstantScale** (*subcalc*, *coeff*)

Bases: [openest.generate.calculation.Calculation](#)

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

**column\_info** ()  
Returns an array of dictionaries, with 'name', 'title', and 'description'.

**static describe** ()

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

**class** openest.generate.functions.**Exponentiate** (*subcalc*)

Bases: [openest.generate.calculation.Calculation](#)

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

**column\_info** ()  
Returns an array of dictionaries, with 'name', 'title', and 'description'.

**static describe** ()

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

**class** openest.generate.functions.**InstaZScore** (*subcalc*, *lastyear*, *units*='z-score')

Bases: [openest.generate.calculation.CustomFunctionalCalculation](#)

Collects up to *baseyear* of values and then uses them to represent all values as a z-score.

**column\_info** ()  
Returns an array of dictionaries, with 'name', 'title', and 'description'.

**static describe** ()

**init\_apply** ()

**pushhandler** (*ds*, *lastyear*)  
Returns an iterator of (yyyy, value, ...).

```
class openest.generate.functions.Instabase (subcalc,      baseyear,      func=<function
                                             <lambda>>,      units='portion',
                                             skip_on_missing=True)
```

Bases: *openest.generate.calculation.CustomFunctionalCalculation*

Re-base the results of `make_generator(...)` to the values in `baseyear` `baseyear` is the year to use as the 'denominator'; None for the first year Default `func` constructs a porportional change; `x - y` makes simple difference. `skip_on_missing`: If we never encounter the year and this is false,

still print out the existing results.

Tacks on the value to the front of the results

```
column_info ()
```

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

```
static describe ()
```

```
donehandler (baseyear, func, skip_on_missing)
```

```
format_handler (equation, lang, baseyear, func, skip_on_missing)
```

```
init_apply ()
```

```
pushhandler (ds, baseyear, func, skip_on_missing)
```

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

```
class openest.generate.functions.Positive (subcalc)
```

Bases: *openest.generate.calculation.Calculation*

Return 0 if `subcalc` is less than 0

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

```
column_info ()
```

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

```
static describe ()
```

```
class openest.generate.functions.Scale (subcalc,      scale_dict,      from_units,      to_units,
                                             func=<function <lambda>>,      latexpair=('\\bar{I}',
                                             'Region-specific scaling'))
```

Bases: *openest.generate.calculation.Calculation*

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

```
column_info ()
```

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

```
static describe ()
```

```
format (lang, *args, **kwargs)
```

Returns a dictionary of `FormatElements`. Only keys in the tree of dependencies will be output.

```
class openest.generate.functions.SpanInstabase (subcalc, year1, year2, func=<function
                                             <lambda>>,      units='portion',
                                             skip_on_missing=True)
```

Bases: *openest.generate.functions.Instabase*

Re-base the results of a calculation to the average of values between two years. Default `func` constructs a porportional change; `x - y` makes simple difference. `skip_on_missing`: If we never encounter the year and this is false,

still print out the existing results.

```
static describe ()
```

**format\_handler** (*equation, lang, baseyear, func, skip\_on\_missing*)

**init\_apply** ()

**pushhandler** (*ds, baseyear, func, skip\_on\_missing*)  
Returns an iterator of (yyyy, value, ...).

**class** openest.generate.functions.**Sum** (*subcalcs*)  
Bases: *openest.generate.calculation.Calculation*

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

**column\_info** ()  
Returns an array of dictionaries, with 'name', 'title', and 'description'.

**static describe** ()

**enable\_deltamethod** ()  
When applied, yield will contain arrays of coefficient multiplicands as a vector the length of the CSVV coefficients.

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

**class** openest.generate.functions.**Transform** (*subcalc, from\_units, to\_units, func, description, long\_description*)  
Bases: *openest.generate.calculation.Calculation*

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

**column\_info** ()  
Returns an array of dictionaries, with 'name', 'title', and 'description'.

**static describe** ()

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

## openest.generate.latextools module

openest.generate.latextools.**call** (*func, 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()
    Returns an array of dictionaries, with 'name', 'title', and 'description'.

static describe()

push(time, weather)
    Returns an iterator of (yyyy, value, ...).

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

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

    static describe()

    push(time, weather)
        Returns an iterator of (yyyy, value, ...).

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

    Bases: openest.generate.calculation.Calculation

    apply(region, *args)

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

    static describe()

class openest.generate.shortterm.SplitByMonth(subcalc)
    Bases: openest.generate.calculation.Calculation

    apply(region, *args, **kwargs)

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

    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)
        Probability density function at x of the given RV.

    Parameters
    • x(array_like) – quantiles
    • arg2, arg3, .. (arg1,) – The shape parameter(s) for the distribution (see docstring
      of the instance object for more information)
    • loc(array_like, optional) – location parameter (default=0)
    • scale(array_like, optional) – scale parameter (default=1)

    Returns pdf – Probability density function evaluated at x

    Return type ndarray

    prepare_draws(mini, maxi, count=10000)
    prepare_draws_gridded(mini, maxi, count=10000)
    rvs(size=1, random_state=None)
        Random variates of given type.

```

**Parameters**

- **arg2, arg3, .. (arg1,)** – The shape parameter(s) for the distribution (see docstring of the instance object for more information).
- **loc** (*array\_like, optional*) – Location parameter (default=0).
- **scale** (*array\_like, optional*) – Scale parameter (default=1).
- **size** (*int or tuple of ints, optional*) – Defining number of random variates (default is 1).
- **random\_state** (None or int or `np.random.RandomState` instance, optional) – If int or `RandomState`, use it for drawing the random variates. If None, rely on `self.random_state`. Default is None.

**Returns** `rvs` – Random variates of given *size*.

**Return type** ndarray or scalar

**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_maxlntaus (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*)  
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:  $F^{-1}(p | x)$ .

**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*)  
 $E[Y | X]$

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

**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*)  
Conditional Probability Density Evaluation  
Returns unscaled probability density values for given values of  $x$  and  $y$ :  $f(y | x)$ .

**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=None)
    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, clipy=0)
    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=None)
    Bases: openest.models.curve.CurveCurve

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

    eval_pval (x, p, threshold=0.001)
        Inverse CDF Evaluation

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

    eval_pvals (x, p, threshold=0.001)

    get_xx ()
        Listing conditional values

        Provide a list of all sampled conditional values.

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  $\text{pp} \cdot \text{sum}(\text{axis}=1) == 1$ .

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

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

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

**copy** ()

copy data and return DDPModel 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*)

Inverse CDF Evaluation

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

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

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

Slice DDPModel 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 *y* 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*, *locations=None*, *scale=1*)  
Bases: openest.models.univariate\_model.UnivariateModel

```

cdf (xx, yy)
static combine (one, two)
copy ()
draw_sample (x=None)
    Produce a sample value of y from the conditional distribution.
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)
    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  $y$ :  $f(y | x)$ .
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)
        Inverse CDF Evaluation

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

    get_xx ()
        Listing conditional values

        Provide a list of all sampled conditional values.

    interpolate_x (newxx)

    kind ()

    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:  $p(z | x) = p(\frac{y}{a} | x)$ .

    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)
        E[Y | X]

    get_sdev (x=None)
        sqrt Var[Y | X]

    init_from_mean_size_file (file, delimiter, status_callback=None)

    interpolate_x (newxx, kind='quadratic')

    kind ()

    static merge (models, treatment='default')

    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:  $p(z | x) = p(\frac{y}{a} | x)$ .

    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)
        Inverse CDF Evaluation

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

    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()
        Listing conditional values

        Provide a list of all sampled conditional values.

    interpolate_x(newxx)

    kind()

    reset_cache()

    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:  $p(z | x) = p(\frac{y}{a} | x)$ .

    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 = {'bin_model+bin_model': <function combine at 0x7fc306a65410>, 'bin_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 = {'bin_model': <function merge at 0x7fc306a65398>, 'bin_model+ddp_model': <
```

```
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)
        Raise the distribution to the power 'a' and rescales.
        Returns modifies this model and returns it
        Return type self
    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  $\{x: (\text{mean}, \text{variance})\}$ .

**static create\_single** ( $xxs$ ,  $y0s$ ,  $y1s$ , *coeffss*, *order=None*, *xx\_is\_categorical=True*)

**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:  $F^{-1}(p | x)$ .

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

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

**static from\_ddp** (*ddp\_model*, *limits*)

**get\_conditional** ( $x$ )

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

$E[Y | X]$

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

$\sqrt{\text{Var}[Y | X]}$

```

get_xx()
    Listing conditional values

    Provide a list of all sampled conditional values.

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)
    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:  $p(z | x) = p(\frac{y}{a} | x)$ .

to_ddp(ys=None)

to_points_at(x, ys)
    Conditional Probability Density Evaluation

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

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,
                                                    coeffs=None)

    add_segment(y0, y1, coeffs)

    approximate_mean(limits)

    static approximate_sum(conditionals)

    static ascin(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.ttransform(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





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.
```

(continues on next page)

(continued from previous page)

```

`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

```

(continues on next page)

(continued from previous page)

```

    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.

```

(continues on next page)

(continued from previous page)

```

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

```

(continues on next page)

(continued from previous page)

```

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):

```

(continues on next page)

(continued from previous page)

```

"""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

```

(continues on next page)

(continued from previous page)

```
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.daily`, 8  
`openest.generate.functions`, 9  
`openest.generate.latextools`, 12  
`openest.generate.retrieve`, 12  
`openest.generate.shortterm`, 13  
`openest.generate.weathertools`, 15  
`openest.lincombo.continuous_sampled`, 15  
`openest.lincombo.helpers`, 16  
`openest.lincombo.hiernorm`, 16  
`openest.lincombo.hierregress`, 16  
`openest.lincombo.montecarlo`, 17  
`openest.lincombo.multi_delta`, 17  
`openest.lincombo.multi_draws`, 17  
`openest.lincombo.multi_normal`, 17  
`openest.lincombo.multi_sampled`, 18  
`openest.lincombo.multi_uniform`, 18  
`openest.lincombo.pooling`, 18  
`openest.models.bin_model`, 18  
`openest.models.curve`, 20  
`openest.models.ddp_model`, 22  
`openest.models.delta_model`, 24  
`openest.models.distribution_model`, 25  
`openest.models.features_interpreter`, 25  
`openest.models.generate`, 26  
`openest.models.hierarchical_normal`, 27  
`openest.models.integral_model`, 27  
`openest.models.mean_size_model`, 28  
`openest.models.memoizable`, 29  
`openest.models.model`, 29  
`openest.models.multivariate_model`, 31  
`openest.models.outer_multi_model`, 31  
`openest.models.parameter`, 31  
`openest.models.spline_model`, 31  
`openest.models.sum_multi_model`, 34  
`openest.models.univariate_model`, 34  
`openest.swapbin`, 35  
`openest.swapbin.swapmodel`, 35  
`openest.swapbin.transform`, 35



## A

- add\_conditional() (openest.models.spline\_model.SplineModelConditional method), 32
- add\_segment() (openest.models.spline\_model.SplineModelConditional method), 33
- add\_to\_y() (openest.models.ddp\_model.DDPModel method), 23
- alpha\_given\_taus() (in module openest.lincombo.hiernorm), 16
- any\_from\_url() (in module openest.generate.retrieve), 12
- 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), 6
- 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.daily.ApplyCurve method), 8
- apply() (openest.generate.daily.AverageByMonth method), 8
- apply() (openest.generate.daily.MonthlyDayBins method), 8
- apply() (openest.generate.daily.PercentWithin method), 8
- apply() (openest.generate.daily.YearlyAverageDay method), 9
- apply() (openest.generate.daily.YearlyDayBins method), 9
- apply() (openest.generate.daily.YearlyDividedPolynomialAverageDay method), 9
- apply() (openest.generate.daily.YearlySumDay method), 9
- apply() (openest.generate.functions.AuxillaryResult method), 9
- apply() (openest.generate.functions.ConstantScale method), 10
- apply() (openest.generate.functions.Exponentiate method), 10
- apply() (openest.generate.functions.Positive method), 11
- apply() (openest.generate.functions.Scale method), 11
- apply() (openest.generate.functions.Sum method), 12
- apply() (openest.generate.functions.Transform method), 12
- apply() (openest.generate.shortterm.InstaZScoreApply method), 13
- apply() (openest.generate.shortterm.MonthlyClimateApply method), 14
- apply() (openest.generate.shortterm.MonthlyZScoreApply method), 14
- apply() (openest.generate.shortterm.SingleWeatherApply method), 14
- apply() (openest.generate.shortterm.SplitByMonth method), 14
- apply\_as\_distribution() (openest.models.distribution\_model.DistributionModel method), 25
- ApplyCurve (class in openest.generate.daily), 8
- approximate\_mean() (openest.models.spline\_model.SplineModelConditional method), 33
- approximate\_sum() (openest.models.spline\_model.SplineModelConditional static method), 33
- ascinv() (openest.models.spline\_model.SplineModelConditional static method), 33
- Attribute (class in openest.models.model), 29
- attribute\_list() (openest.models.mean\_size\_model.MeanSizeModel method), 28
- attribute\_list() (openest.models.model.Model method), 30

AuxillaryResult (class in openest.generate.functions), 9  
AuxillaryResultApplication (class in openest.generate.functions), 10  
AverageByMonth (class in openest.generate.daily), 8

## B

best\_knot() (openest.models.features\_interpreter.FeaturesInterpreter static method), 26  
best\_spline() (openest.models.features\_interpreter.FeaturesInterpreter static method), 26  
betahat\_given\_tau() (in module openest.lincombo.hierregress), 16  
betahat\_given\_taus() (in module openest.lincombo.hiernorm), 16  
BinModel (class in openest.models.bin\_model), 18

## C

Calculation (class in openest.generate.calculation), 6  
call() (in module openest.generate.latextools), 12  
cdf() (openest.models.bin\_model.BinModel method), 19  
cdf() (openest.models.delta\_model.DeltaModel method), 24  
cdf() (openest.models.spline\_model.SplineModel method), 32  
cdf() (openest.models.spline\_model.SplineModelConditional method), 33  
check\_arguments() (in module openest.lincombo.helpers), 16  
choose\_model() (in module openest.generate.retrieve), 13  
cleanup() (openest.generate.calculation.Calculation method), 6  
cleanup() (openest.generate.calculation.FunctionalCalculation method), 7  
ClippedCurve (class in openest.models.curve), 20  
CoefficientsCurve (class in openest.models.curve), 20  
column\_info() (openest.generate.calculation.Calculation method), 6  
column\_info() (openest.generate.daily.ApplyCurve method), 8  
column\_info() (openest.generate.daily.AverageByMonth method), 8  
column\_info() (openest.generate.daily.MonthlyDayBins method), 8  
column\_info() (openest.generate.daily.PercentWithin method), 8  
column\_info() (openest.generate.daily.YearlyAverageDay method), 9  
column\_info() (openest.generate.daily.YearlyDayBins method), 9  
column\_info() (openest.generate.daily.YearlyDividedPolynomialAverageDay method), 9  
column\_info() (openest.generate.daily.YearlySumDay method), 9

column\_info() (openest.generate.functions.AuxillaryResult method), 10  
column\_info() (openest.generate.functions.ConstantScale method), 10  
column\_info() (openest.generate.functions.Exponentiate method), 10  
column\_info() (openest.generate.functions.Instabase method), 11  
column\_info() (openest.generate.functions.InstaZScore method), 10  
column\_info() (openest.generate.functions.Positive method), 11  
column\_info() (openest.generate.functions.Scale method), 11  
column\_info() (openest.generate.functions.Sum method), 12  
column\_info() (openest.generate.functions.Transform method), 12  
column\_info() (openest.generate.shortterm.InstaZScoreApply method), 13  
column\_info() (openest.generate.shortterm.MonthlyClimateApply method), 14  
column\_info() (openest.generate.shortterm.MonthlyZScoreApply method), 14  
column\_info() (openest.generate.shortterm.SingleWeatherApply method), 14  
column\_info() (openest.generate.shortterm.SplitByMonth method), 14  
combine() (openest.models.bin\_model.BinModel static method), 19  
combine() (openest.models.ddp\_model.DDPModel static method), 23  
combine() (openest.models.delta\_model.DeltaModel static method), 25  
combine() (openest.models.mean\_size\_model.MeanSizeModel static method), 28  
combine() (openest.models.model.Model static method), 30  
combine() (openest.models.spline\_model.SplineModel static method), 32  
combiners (openest.models.model.Model attribute), 30  
combo\_effects() (in module openest.generate.weathertools), 15  
condition() (openest.models.multivariate\_model.MultivariateModel method), 31  
condition() (openest.models.outer\_multi\_model.OuterMultiModel method), 31  
consistent\_bins() (openest.models.bin\_model.BinModel static method), 19  
ConstantCurveGenerator (class in openest.generate.curvegen), 7  
ConstantScale (class in openest.generate.functions), 10  
ContinuousSampled (class in openest.lincombo.continuous\_sampled), 15

convolve() (openest.models.spline\_model.SplineModelConditional static method), 33

copy() (openest.models.bin\_model.BinModel method), 19

copy() (openest.models.ddp\_model.DDPModel method), 23

copy() (openest.models.delta\_model.DeltaModel method), 25

copy() (openest.models.integral\_model.IntegralModel method), 27

copy() (openest.models.mean\_size\_model.MeanSizeModel method), 28

copy() (openest.models.memoizable.MemoizedUnivariate method), 29

copy() (openest.models.model.Model method), 30

copy() (openest.models.spline\_model.SplineModel method), 32

copy() (openest.models.spline\_model.SplineModelConditional method), 33

create\_gaussian() (openest.models.spline\_model.SplineModel static method), 32

create\_lin() (openest.models.ddp\_model.DDPModel static method), 23

create\_single() (openest.models.spline\_model.SplineModel static method), 32

CubicSplineCurve (class in openest.models.curve), 21

CurveCurve (class in openest.models.curve), 21

CurveGenerator (class in openest.generate.curvegen), 7

CustomFunctionalCalculation (class in openest.generate.calculation), 6

## D

date\_to\_datestr() (in module openest.generate.weathertools), 15

ddp\_from\_url() (in module openest.generate.retrieve), 13

DDPModel (class in openest.models.ddp\_model), 22

default\_condition() (openest.models.multivariate\_model.MultivariateModel method), 31

default\_condition() (openest.models.outer\_multi\_model.OuterMultiModel method), 31

DelayedCurveGenerator (class in openest.generate.curvegen), 7

DeltaModel (class in openest.models.delta\_model), 24

derive() (openest.models.parameter.ParameterBase method), 31

describe() (openest.generate.calculation.Calculation static method), 6

describe() (openest.generate.daily.ApplyCurve static method), 8

describe() (openest.generate.daily.AverageByMonth static method), 8

describe() (openest.generate.daily.MonthlyDayBins static method), 8

describe() (openest.generate.daily.PercentWithin static method), 8

describe() (openest.generate.daily.YearlyAverageDay static method), 9

describe() (openest.generate.daily.YearlyDayBins static method), 9

describe() (openest.generate.daily.YearlyDividedPolynomialAverageDay static method), 9

describe() (openest.generate.daily.YearlySumDay static method), 9

describe() (openest.generate.functions.AuxillaryResult static method), 10

describe() (openest.generate.functions.ConstantScale static method), 10

describe() (openest.generate.functions.Exponentiate static method), 10

describe() (openest.generate.functions.Instabase static method), 11

describe() (openest.generate.functions InstaZScore static method), 10

describe() (openest.generate.functions.Positive static method), 11

describe() (openest.generate.functions.Scale static method), 11

describe() (openest.generate.functions.SpanInstabase static method), 11

describe() (openest.generate.functions.Sum static method), 12

describe() (openest.generate.functions.Transform static method), 12

describe() (openest.generate.shortterm InstaZScoreApply static method), 14

describe() (openest.generate.shortterm.MonthlyClimateApply static method), 14

describe() (openest.generate.shortterm.MonthlyZScoreApply static method), 14

describe() (openest.generate.shortterm.SingleWeatherApply static method), 14

describe() (openest.generate.shortterm.SplitByMonth static method), 14

dims() (openest.models.outer\_multi\_model.OuterMultiModel method), 31

DistributionModel (class in openest.models.distribution\_model), 25

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

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

done() (openest.generate.calculation.CustomFunctionalCalculation method), 7

done() (openest.generate.functions.AuxillaryResultApplication method), 10

- donehandler() (openest.generate.calculation.CustomFunctionalCalculation method), 29
- donehandler() (openest.generate.functions.Instabase method), 11
- draw\_from\_counts() (in module openest.models.hierarchical\_normal), 27
- draw\_sample() (openest.models.bin\_model.BinModel method), 19
- draw\_sample() (openest.models.ddp\_model.DDPModel method), 23
- draw\_sample() (openest.models.delta\_model.DeltaModel method), 25
- draw\_sample() (openest.models.model.Model method), 30
- draw\_sample() (openest.models.spline\_model.SplineModel method), 32
- draw\_sample() (openest.models.spline\_model.SplineModelConditional method), 34
- ## E
- enable\_deltamethod() (openest.generate.calculation.Calculation method), 6
- enable\_deltamethod() (openest.generate.calculation.FunctionalCalculation method), 7
- enable\_deltamethod() (openest.generate.functions.Sum method), 12
- english\_function() (in module openest.generate.latextools), 12
- estimated\_maxlintaus() (in module openest.lincombo.pooling), 18
- estimated\_maxtau() (in module openest.lincombo.pooling), 18
- eval\_pval() (openest.models.bin\_model.BinModel method), 19
- eval\_pval() (openest.models.curve.UnivariateCurve method), 21
- eval\_pval() (openest.models.ddp\_model.DDPModel method), 23
- eval\_pval() (openest.models.integral\_model.IntegralModel method), 27
- eval\_pval() (openest.models.memoizable.MemoizedUnivariate method), 29
- eval\_pval() (openest.models.model.Model method), 30
- eval\_pval() (openest.models.spline\_model.SplineModel method), 32
- eval\_pval\_index() (openest.models.bin\_model.BinModel method), 19
- eval\_pval\_index() (openest.models.ddp\_model.DDPModel method), 23
- eval\_pval\_index() (openest.models.memoizable.MemoizableUnivariate method), 29
- eval\_pval\_index() (openest.models.spline\_model.SplineModel method), 32
- eval\_pvals() (openest.models.curve.UnivariateCurve method), 21
- eval\_pvals() (openest.models.memoizable.MemoizedUnivariate method), 29
- evaluate() (openest.models.spline\_model.SplineModelConditional method), 34
- evaluate\_spline() (openest.models.features\_interpreter.FeaturesInterpreter static method), 26
- Exponentiate (class in openest.generate.functions), 10
- ## F
- features\_to\_exponential() (openest.models.features\_interpreter.FeaturesInterpreter static method), 26
- features\_to\_gaussian() (openest.models.features\_interpreter.FeaturesInterpreter static method), 26
- features\_to\_uniform() (openest.models.features\_interpreter.FeaturesInterpreter static method), 26
- FeaturesInterpreter (class in openest.models.features\_interpreter), 26
- filter\_x() (openest.models.bin\_model.BinModel method), 19
- filter\_x() (openest.models.ddp\_model.DDPModel method), 23
- filter\_x() (openest.models.delta\_model.DeltaModel method), 25
- filter\_x() (openest.models.mean\_size\_model.MeanSizeModel method), 28
- filter\_x() (openest.models.spline\_model.SplineModel method), 32
- filter\_x() (openest.models.univariate\_model.UnivariateModel method), 34
- find\_bins() (in module openest.swapbin.swapmodel), 35
- find\_mode() (openest.models.spline\_model.SplineModelConditional method), 34
- find\_nearest() (openest.models.spline\_model.SplineModelConditional static method), 34
- FlatCurve (class in openest.models.curve), 21
- float\_condition() (openest.models.outer\_multi\_model.OuterMultiModel method), 31
- format() (openest.generate.calculation.Calculation method), 6
- format() (openest.generate.calculation.FunctionalCalculation method), 7
- format() (openest.generate.daily.AverageByMonth method), 8



format() (openest.generate.daily.MonthlyDayBins method), 8  
 format() (openest.generate.daily.YearlyAverageDay method), 9  
 format() (openest.generate.daily.YearlyDayBins method), 9  
 format() (openest.generate.daily.YearlySumDay method), 9  
 format() (openest.generate.functions.AuxillaryResult method), 10  
 format() (openest.generate.functions.ConstantScale method), 10  
 format() (openest.generate.functions.Exponentiate method), 10  
 format() (openest.generate.functions.Scale method), 11  
 format() (openest.generate.functions.Sum method), 12  
 format() (openest.generate.functions.Transform method), 12  
 format\_call() (openest.generate.curvegen.ConstantCurveGenerator method), 7  
 format\_call() (openest.generate.curvegen.CurveGenerator method), 7  
 format\_call() (openest.generate.curvegen.DelayedCurveGenerator method), 7  
 format\_call() (openest.generate.curvegen.TransformCurveGenerator method), 8  
 format\_handler() (openest.generate.calculation.FunctionalCalculation method), 7  
 format\_handler() (openest.generate.functions.Instabase method), 11  
 format\_handler() (openest.generate.functions.SpanInstabase method), 11  
 from\_ddp() (openest.models.spline\_model.SplineModel static method), 32  
 from\_file() (openest.models.ddp\_model.DDPModel static method), 23  
 from\_url() (in module openest.generate.retrieve), 13  
 FunctionalCalculation (class in openest.generate.calculation), 7

## G

gaussian\_mean() (openest.models.spline\_model.SplineModelConditional method), 34  
 gaussian\_sdev() (openest.models.spline\_model.SplineModelConditional method), 34  
 generate\_thetas() (in module openest.models.hierarchical\_normal), 27  
 get\_attribute() (openest.models.mean\_size\_model.MeanSizeModel method), 28  
 get\_attribute() (openest.models.model.Model method), 30  
 get\_bin\_at() (openest.models.bin\_model.BinModel method), 19  
 get\_closest() (openest.models.ddp\_model.DDPModel method), 23  
 get\_conditional() (openest.models.spline\_model.SplineModel method), 32  
 get\_crop\_calendar() (in module openest.generate.weathertools), 15  
 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), 8  
 get\_edges() (openest.models.bin\_model.BinModel method), 19  
 get\_edges() (openest.models.memoizable.MemoizableUnivariate method), 29  
 get\_eval\_pval\_spline() (openest.models.memoizable.MemoizedUnivariate method), 29  
 get\_index() (openest.models.memoizable.MemoizedUnivariate method), 29  
 get\_indexes() (openest.models.memoizable.MemoizedUnivariate method), 29  
 get\_lincom\_terms() (openest.generate.curvegen.TransformCurveGenerator method), 8  
 get\_lincom\_terms\_simple() (openest.generate.curvegen.TransformCurveGenerator method), 8  
 get\_mean() (openest.models.bin\_model.BinModel method), 19  
 get\_mean() (openest.models.ddp\_model.DDPModel method), 23  
 get\_mean() (openest.models.mean\_size\_model.MeanSizeModel method), 28  
 get\_mean() (openest.models.model.Model method), 30  
 get\_mean() (openest.models.spline\_model.SplineModel method), 32  
 get\_next\_curve() (openest.generate.curvegen.DelayedCurveGenerator method), 7  
 get\_pval() (openest.models.spline\_model.SplineModelConditional method), 34  
 get\_random() (in module openest.models.hierarchical\_normal), 27  
 get\_sampled\_column() (in module openest.lincombo.hiernorm), 16  
 get\_sampled\_column() (in module openest.lincombo.hierregress), 16

`get_sdev()` (openest.models.bin\_model.BinModel method), 19  
`get_sdev()` (openest.models.ddp\_model.DDPModel method), 23  
`get_sdev()` (openest.models.mean\_size\_model.MeanSizeModel method), 28  
`get_sdev()` (openest.models.model.Model method), 30  
`get_sdev()` (openest.models.spline\_model.SplineModel method), 32  
`get_terms()` (openest.models.curve.CubicSplineCurve method), 21  
`get_xx()` (openest.models.bin\_model.BinModel method), 19  
`get_xx()` (openest.models.curve.UnivariateCurve method), 21  
`get_xx()` (openest.models.ddp\_model.DDPModel method), 23  
`get_xx()` (openest.models.integral\_model.IntegralModel method), 27  
`get_xx()` (openest.models.memoizable.MemoizedUnivariate method), 29  
`get_xx()` (openest.models.spline\_model.SplineModel method), 32  
`get_xx()` (openest.models.univariate\_model.UnivariateModel method), 34  
`get_xx_centers()` (openest.models.bin\_model.BinModel method), 20  
`get_yy()` (openest.models.ddp\_model.DDPModel method), 23  
`growing_seasons_daily_ncdf()` (in module openest.generate.weathertools), 15  
`growing_seasons_mean_ncdf()` (in module openest.generate.weathertools), 15  
`growing_seasons_mean_reader()` (in module openest.generate.weathertools), 15  
`guess_ranges()` (openest.lincombo.continuous\_sampled.ContinuousSampled method), 15  
`guess_ranges()` (openest.lincombo.multi\_sampled.MultivariateSampled method), 18  
`guess_ranges_gridded()` (openest.lincombo.continuous\_sampled.ContinuousSampled method), 15  
`guess_ranges_gridded()` (openest.lincombo.multi\_sampled.MultivariateSampled method), 18  
**H**  
`handler()` (openest.generate.calculation.FunctionalCalculation method), 7  
`helper_params()` (in module openest.models.hierarchical\_normal), 27  
**I**  
`init_apply()` (openest.generate.calculation.CustomFunctionalCalculation method), 7  
`init_apply()` (openest.generate.functions.Instabase method), 11  
`init_apply()` (openest.generate.functions.InstaZScore method), 10  
`init_apply()` (openest.generate.functions.SpanInstabase method), 12  
`init_from()` (openest.models.ddp\_model.DDPModel method), 23  
`init_from_bin_file()` (openest.models.bin\_model.BinModel method), 20  
`init_from_delta_file()` (openest.models.delta\_model.DeltaModel method), 25  
`init_from_feature_file()` (openest.models.features\_interpreter.FeaturesInterpreter static method), 26  
`init_from_mean_size_file()` (openest.models.mean\_size\_model.MeanSizeModel method), 28  
`init_from_other()` (openest.models.ddp\_model.DDPModel method), 24  
`init_from_spline_file()` (openest.models.spline\_model.SplineModel method), 33  
`init_from_union()` (openest.models.outer\_multi\_model.OuterMultiModel method), 31  
`Instabase` (class in openest.generate.functions), 10  
`InstaZScore` (class in openest.generate.functions), 10  
`InstaZScoreApply` (class in openest.generate.shortterm), 13  
`IntegralModel` (class in openest.models.integral\_model), 27  
`interpolate_x()` (openest.models.bin\_model.BinModel method), 20  
`interpolate_x()` (openest.models.ddp\_model.DDPModel method), 24  
`interpolate_x()` (openest.models.delta\_model.DeltaModel method), 25  
`interpolate_x()` (openest.models.integral\_model.IntegralModel method), 27  
`interpolate_x()` (openest.models.mean\_size\_model.MeanSizeModel method), 28  
`interpolate_x()` (openest.models.memoizable.MemoizedUnivariate method), 29  
`interpolate_x()` (openest.models.spline\_model.SplineModel method), 33  
`interpolate_x()` (openest.models.univariate\_model.UnivariateModel method), 35  
`interpolate_y()` (openest.models.ddp\_model.DDPModel method), 24

- `intersect_get_model()` (openest.models.univariate\_model.UnivariateModel static method), 35
- `intersect_get_x()` (openest.models.univariate\_model.UnivariateModel static method), 35
- `intersect_x()` (openest.models.univariate\_model.UnivariateModel static method), 35
- `intersect_x_all()` (openest.models.univariate\_model.UnivariateModel static method), 35
- `is_gaussian()` (openest.models.spline\_model.SplineModel method), 33
- `is_gaussian()` (openest.models.spline\_model.SplineModelConditional method), 34
- `issparse()` (in module openest.lincombo.helpers), 16
- ## K
- `kind()` (openest.models.bin\_model.BinModel method), 20
- `kind()` (openest.models.ddp\_model.DDPModel method), 24
- `kind()` (openest.models.delta\_model.DeltaModel method), 25
- `kind()` (openest.models.integral\_model.IntegralModel method), 27
- `kind()` (openest.models.mean\_size\_model.MeanSizeModel method), 28
- `kind()` (openest.models.memoizable.MemoizedUnivariate method), 29
- `kind()` (openest.models.model.Model method), 30
- `kind()` (openest.models.outer\_multi\_model.OuterMultiModel method), 31
- `kind()` (openest.models.spline\_model.SplineModel method), 33
- ## L
- `latex_function()` (in module openest.generate.latextools), 12
- `lin_p()` (openest.models.ddp\_model.DDPModel method), 24
- `lincombo_hiernorm_taubyalpha()` (in module openest.lincombo.hiernorm), 16
- `lincombo_hiernorm_taubybeta()` (in module openest.lincombo.hiernorm), 16
- `lincombo_hierregress()` (in module openest.lincombo.hierregress), 16
- `lincombo_hierregress_taubybeta()` (in module openest.lincombo.hierregress), 17
- `lincombo_hierregress_taubymu()` (in module openest.lincombo.hierregress), 17
- `lincombo_pooled()` (in module openest.lincombo.pooling), 18
- `LinearCurve` (class in openest.models.curve), 21
- `log_p()` (openest.models.ddp\_model.DDPModel method), 24
- `logpdf()` (openest.lincombo.multi\_normal.MultivariateNormal method), 17
- ## M
- `make_conditional()` (openest.models.features\_interpreter.FeaturesInterpreter static method), 26
- `make_conditional_from_spline()` (openest.models.spline\_model.SplineModelConditional static method), 34
- `make_conditional_respecting()` (openest.models.features\_interpreter.FeaturesInterpreter static method), 26
- `make_gaussian()` (openest.models.spline\_model.SplineModelConditional static method), 34
- `make_linear_spline_curve()` (openest.models.curve.CurveCurve static method), 21
- `make_single()` (openest.models.spline\_model.SplineModelConditional static method), 34
- `maxs()` (openest.lincombo.multi\_uniform.MultivariateUniform method), 18
- `mean()` (openest.lincombo.multi\_draws.MultivariateDraws method), 17
- `MeanSizeModel` (class in openest.models.mean\_size\_model), 28
- `MemoizableUnivariate` (class in openest.models.memoizable), 29
- `MemoizedUnivariate` (class in openest.models.memoizable), 29
- `merge()` (openest.models.bin\_model.BinModel static method), 20
- `merge()` (openest.models.ddp\_model.DDPModel static method), 24
- `merge()` (openest.models.delta\_model.DeltaModel static method), 25
- `merge()` (openest.models.mean\_size\_model.MeanSizeModel static method), 28
- `merge()` (openest.models.model.Model static method), 30
- `merge()` (openest.models.spline\_model.SplineModel static method), 33
- `mergers` (openest.models.model.Model attribute), 30
- `MinimumCurve` (class in openest.models.curve), 21
- `mins()` (openest.lincombo.multi\_uniform.MultivariateUniform method), 18
- `Model` (class in openest.models.model), 29
- `MonthlyClimateApply` (class in openest.generate.shortterm), 14
- `MonthlyDayBins` (class in openest.generate.daily), 8
- `MonthlyZScoreApply` (class in openest.generate.shortterm), 14

mu\_given\_tau() (in module  
est.lincombo.hierregress), 17

MultivariateDelta (class in  
est.lincombo.multi\_delta), 17

MultivariateDraws (class in  
est.lincombo.multi\_draws), 17

MultivariateModel (class in  
est.models.multivariate\_model), 31

MultivariateNormal (class in  
est.lincombo.multi\_normal), 17

MultivariateSampled (class in  
est.lincombo.multi\_sampled), 18

MultivariateUniform (class in  
est.lincombo.multi\_uniform), 18

## N

neginf (openest.models.spline\_model.SplineModel  
attribute), 33

nongaussian\_x2px() (open-  
est.models.spline\_model.SplineModelConditional  
method), 34

nongaussian\_xpx() (open-  
est.models.spline\_model.SplineModelConditional  
method), 34

numvars() (openest.models.multivariate\_model.MultivariateModel  
method), 31

## O

openest.generate.calculation (module), 5

openest.generate.curvegen (module), 7

openest.generate.daily (module), 8

openest.generate.functions (module), 9

openest.generate.latextools (module), 12

openest.generate.retrieve (module), 12

openest.generate.shortterm (module), 13

openest.generate.weathertools (module), 15

openest.lincombo.continuous\_sampled (module), 15

openest.lincombo.helpers (module), 16

openest.lincombo.hiernorm (module), 16

openest.lincombo.hierregress (module), 16

openest.lincombo.montecarlo (module), 17

openest.lincombo.multi\_delta (module), 17

openest.lincombo.multi\_draws (module), 17

openest.lincombo.multi\_normal (module), 17

openest.lincombo.multi\_sampled (module), 18

openest.lincombo.multi\_uniform (module), 18

openest.lincombo.pooling (module), 18

openest.models.bin\_model (module), 18

openest.models.curve (module), 20

openest.models.ddp\_model (module), 22

openest.models.delta\_model (module), 24

openest.models.distribution\_model (module), 25

openest.models.features\_interpreter (module), 25

openest.models.generate (module), 26

openest.models.hierarchical\_normal (module), 27

openest.models.integral\_model (module), 27

openest.models.mean\_size\_model (module), 28

openest.models.memoizable (module), 29

openest.models.model (module), 29

openest.models.multivariate\_model (module), 31

openest.models.outer\_multi\_model (module), 31

openest.models.parameter (module), 31

openest.models.spline\_model (module), 31

openest.models.sum\_multi\_model (module), 34

openest.models.univariate\_model (module), 34

openest.swapbin (module), 35

openest.swapbin.swapmodel (module), 35

openest.swapbin.transform (module), 35

OtherClippedCurve (class in openest.models.curve), 21

OuterMultiModel (class in open-  
est.models.outer\_multi\_model), 31

## P

pdf\_tau\_given\_y() (in module open-  
est.models.hierarchical\_normal), 27

ParameterBase (class in openest.models.parameter), 31

partial\_cdf() (openest.models.spline\_model.SplineModelConditional  
method), 34

Model openest.lincombo.continuous\_sampled.ContinuousSampled  
method), 15

pdf() (openest.lincombo.multi\_delta.MultivariateDelta  
method), 17

pdf() (openest.lincombo.multi\_normal.MultivariateNormal  
method), 17

pdf() (openest.lincombo.multi\_sampled.MultivariateSampled  
method), 18

pdf() (openest.lincombo.multi\_uniform.MultivariateUniform  
method), 18

PercentWithin (class in openest.generate.daily), 8

PiecewiseCurve (class in openest.models.curve), 21

polynomial() (in module openest.models.generate), 26

pos() (in module openest.models.curve), 21

posinf (openest.models.spline\_model.SplineModel  
attribute), 33

Positive (class in openest.generate.functions), 11

prepare\_draws() (open-  
est.lincombo.continuous\_sampled.ContinuousSampled  
method), 15

prepare\_draws() (open-  
est.lincombo.multi\_sampled.MultivariateSampled  
method), 18

prepare\_draws\_gridded() (open-  
est.lincombo.continuous\_sampled.ContinuousSampled  
method), 15

prepare\_draws\_gridded() (open-  
est.lincombo.multi\_sampled.MultivariateSampled  
method), 18

- probability\_tau() (in module openest.lincombo.hiernorm), 16
- probability\_tau() (in module openest.lincombo.hierregress), 17
- ProductCurve (class in openest.models.curve), 21
- propose\_grid() (openest.models.spline\_model.SplineModelConditional static method), 34
- 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), 7
- push() (openest.generate.functions.AuxillaryResultApplication method), 10
- push() (openest.generate.shortterm.InstaZScoreApply method), 14
- push() (openest.generate.shortterm.MonthlyZScoreApply method), 14
- push\_saved() (openest.generate.calculation.ApplicationByChunks method), 5
- push\_saved() (openest.generate.calculation.ApplicationByYear method), 6
- pushhandler() (openest.generate.calculation.CustomFunctionalCalculation method), 7
- pushhandler() (openest.generate.functions.Instabase method), 11
- pushhandler() (openest.generate.functions.InstaZScore method), 10
- pushhandler() (openest.generate.functions.SpanInstabase method), 12
- ## R
- re\_condition() (openest.models.outer\_multi\_model.OuterMultiModel static method), 31
- re\_numeric (openest.models.outer\_multi\_model.OuterMultiModel attribute), 31
- read\_scale\_file() (in module openest.generate.weathertools), 15
- recategorize\_x() (openest.models.ddp\_model.DDPModel method), 24
- recategorize\_x() (openest.models.spline\_model.SplineModel method), 33
- recategorize\_x() (openest.models.univariate\_model.UnivariateModel method), 35
- regress\_distribution() (in module openest.lincombo.montecarlo), 17
- regress\_draws() (in module openest.lincombo.montecarlo), 17
- regress\_summary() (in module openest.lincombo.montecarlo), 17
- rescale() (openest.models.ddp\_model.DDPModel method), 24
- rescale() (openest.models.spline\_model.SplineModelConditional method), 34
- reset\_cache() (openest.models.memoizable.MemoizedUnivariate method), 29
- rough\_limits() (openest.models.spline\_model.SplineModelConditional method), 34
- rough\_span() (openest.models.spline\_model.SplineModelConditional method), 34
- rvs() (openest.lincombo.continuous\_sampled.ContinuousSampled method), 15
- rvs() (openest.lincombo.multi\_delta.MultivariateDelta method), 17
- rvs() (openest.lincombo.multi\_draws.MultivariateDraws method), 17
- rvs() (openest.lincombo.multi\_normal.MultivariateNormal method), 17
- rvs() (openest.lincombo.multi\_sampled.MultivariateSampled method), 18
- rvs() (openest.lincombo.multi\_uniform.MultivariateUniform method), 18
- ## S
- sample\_posterior() (in module openest.lincombo.hiernorm), 16
- sample\_posterior() (in module openest.lincombo.hierregress), 17
- samples (openest.models.spline\_model.SplineModel attribute), 33
- Scale (class in openest.generate.functions), 11
- scale() (openest.models.spline\_model.SplineModelConditional method), 34
- scale() (openest.models.bin\_model.BinModel method), 20
- scale\_p() (openest.models.ddp\_model.DDPModel method), 24
- scale\_p() (openest.models.delta\_model.DeltaModel method), 25
- scale\_p() (openest.models.integral\_model.IntegralModel method), 27
- scale\_p() (openest.models.mean\_size\_model.MeanSizeModel method), 28
- scale\_p() (openest.models.memoizable.MemoizedUnivariate method), 29
- scale\_p() (openest.models.model.Model method), 30
- scale\_p() (openest.models.outer\_multi\_model.OuterMultiModel method), 31



[scale\\_p\(\)](#) (openest.models.spline\_model.SplineModel method), [33](#)  
[scale\\_p\(\)](#) (openest.models.spline\_model.SplineModelConditional method), [34](#)  
[scale\\_y\(\)](#) (openest.models.bin\_model.BinModel method), [20](#)  
[scale\\_y\(\)](#) (openest.models.ddp\_model.DDPModel method), [24](#)  
[scale\\_y\(\)](#) (openest.models.delta\_model.DeltaModel method), [25](#)  
[scale\\_y\(\)](#) (openest.models.integral\_model.IntegralModel method), [27](#)  
[scale\\_y\(\)](#) (openest.models.mean\_size\_model.MeanSizeModel method), [28](#)  
[scale\\_y\(\)](#) (openest.models.memoizable.MemoizedUnivariate method), [29](#)  
[scale\\_y\(\)](#) (openest.models.model.Model method), [30](#)  
[scale\\_y\(\)](#) (openest.models.spline\_model.SplineModel method), [33](#)  
[scale\\_y\(\)](#) (openest.models.spline\_model.SplineModelConditional method), [34](#)  
[segment\\_max\(\)](#) (openest.models.spline\_model.SplineModelConditional method), [34](#)  
[SelectiveInputCurve](#) (class in openest.models.curve), [21](#)  
[set\\_x\\_cache\\_decimals\(\)](#) (openest.models.memoizable.MemoizedUnivariate method), [29](#)  
[ShiftedCurve](#) (class in openest.models.curve), [21](#)  
[simulate\\_normal\\_model\(\)](#) (in module openest.models.hierarchical\_normal), [27](#)  
[SingleWeatherApply](#) (class in openest.generate.shortterm), [14](#)  
[size\(\)](#) (openest.models.spline\_model.SplineModelConditional method), [34](#)  
[skew\\_gaussian\\_construct\(\)](#) (openest.models.features\_interpreter.FeaturesInterpreter static method), [26](#)  
[skew\\_gaussian\\_evaluate\(\)](#) (openest.models.features\_interpreter.FeaturesInterpreter static method), [26](#)  
[SpanInstabase](#) (class in openest.generate.functions), [11](#)  
[spline\\_from\\_url\(\)](#) (in module openest.generate.retrieve), [13](#)  
[SplineModel](#) (class in openest.models.spline\_model), [31](#)  
[SplineModelConditional](#) (class in openest.models.spline\_model), [33](#)  
[SplitByMonth](#) (class in openest.generate.shortterm), [14](#)  
[std\(\)](#) (openest.lincombo.multi\_draws.MultivariateDraws method), [17](#)  
[StepCurve](#) (class in openest.models.curve), [21](#)  
[Sum](#) (class in openest.generate.functions), [12](#)  
[sum\\_multiply\(\)](#) (in module openest.lincombo.pooling), [18](#)  
[sum\\_multiply2\(\)](#) (in module openest.lincombo.pooling), [18](#)  
[SumMultiModel](#) (class in openest.models.sum\_multi\_model), [34](#)  
[swap\\_any\(\)](#) (in module openest.swapbin.swapmodel), [35](#)  
[swap\\_beta\(\)](#) (in module openest.swapbin.transform), [35](#)  
[swap\\_bin\(\)](#) (in module openest.swapbin.swapmodel), [35](#)  
[swap\\_spline\(\)](#) (in module openest.swapbin.swapmodel), [35](#)  
[swap\\_values\(\)](#) (in module openest.swapbin.swapmodel), [35](#)  
[swap\\_vcv\(\)](#) (in module openest.swapbin.transform), [35](#)  

## T

[test\(\)](#) (openest.generate.calculation.Calculation method), [6](#)  
[to\\_ddp\(\)](#) (openest.models.bin\_model.BinModel method), [20](#)  
[to\\_ddp\(\)](#) (openest.models.ddp\_model.DDPModel method), [24](#)  
[to\\_ddp\(\)](#) (openest.models.spline\_model.SplineModel method), [33](#)  
[to\\_points\\_at\(\)](#) (openest.models.spline\_model.SplineModelConditional method), [34](#)  
[to\\_points\\_at\(\)](#) (openest.models.bin\_model.BinModel method), [20](#)  
[to\\_points\\_at\(\)](#) (openest.models.delta\_model.DeltaModel method), [25](#)  
[to\\_points\\_at\(\)](#) (openest.models.model.Model method), [30](#)  
[to\\_points\\_at\(\)](#) (openest.models.spline\_model.SplineModel method), [33](#)  
[Transform](#) (class in openest.generate.functions), [12](#)  
[transform\(\)](#) (in module openest.swapbin.transform), [35](#)  
[TransformCurveGenerator](#) (class in openest.generate.curvegen), [7](#)  
[transpose\(\)](#) (openest.models.ddp\_model.DDPModel method), [24](#)  

## U

[uniform\\_constant\(\)](#) (in module openest.models.generate), [26](#)  
[uniform\\_doseless\(\)](#) (in module openest.models.generate), [26](#)  
[UnivariateCurve](#) (class in openest.models.curve), [21](#)  
[UnivariateModel](#) (class in openest.models.univariate\_model), [34](#)  

## V

[vals\(\)](#) (openest.lincombo.multi\_delta.MultivariateDelta method), [17](#)  

## W

[write\(\)](#) (openest.models.bin\_model.BinModel method), [20](#)

`write()` (openest.models.ddp\_model.DDPModel method), [24](#)      `ZeroInterceptPolynomialCurve` (class in openest.models.curve), [21](#)  
`write()` (openest.models.delta\_model.DeltaModel method), [25](#)  
`write()` (openest.models.integral\_model.IntegralModel method), [27](#)  
`write()` (openest.models.mean\_size\_model.MeanSizeModel method), [28](#)  
`write()` (openest.models.memoizable.MemoizedUnivariate method), [29](#)  
`write()` (openest.models.outer\_multi\_model.OuterMultiModel method), [31](#)  
`write()` (openest.models.spline\_model.SplineModel method), [33](#)  
`write_file()` (openest.models.bin\_model.BinModel method), [20](#)  
`write_file()` (openest.models.ddp\_model.DDPModel method), [24](#)  
`write_file()` (openest.models.delta\_model.DeltaModel method), [25](#)  
`write_file()` (openest.models.integral\_model.IntegralModel method), [27](#)  
`write_file()` (openest.models.mean\_size\_model.MeanSizeModel method), [28](#)  
`write_file()` (openest.models.memoizable.MemoizedUnivariate method), [29](#)  
`write_file()` (openest.models.outer\_multi\_model.OuterMultiModel method), [31](#)  
`write_file()` (openest.models.spline\_model.SplineModel method), [33](#)  
`write_gaussian()` (openest.models.spline\_model.SplineModel method), [33](#)  
`write_gaussian_plus()` (openest.models.spline\_model.SplineModel method), [33](#)

## X

`xmap_apply_model()` (in module openest.generate.weathertools), [15](#)

## Y

`yearly_daily_ncdf()` (in module openest.generate.weathertools), [15](#)  
`YearlyAverageDay` (class in openest.generate.daily), [8](#)  
`YearlyDayBins` (class in openest.generate.daily), [9](#)  
`YearlyDividedPolynomialAverageDay` (class in openest.generate.daily), [9](#)  
`YearlySumDay` (class in openest.generate.daily), [9](#)

## Z

`zero_delta()` (openest.models.delta\_model.DeltaModel static method), [25](#)