dessn Documentation

Release 0.0.1

dessn

CONTENTS

| 1 | dessn | | 1 | |
|----|--------|--------|--|---|
| | 1.1 | _ | | 1 |
| | | 1.1.1 | 1 | 1 |
| | | | | 1 |
| | | | | 1 |
| | | | | 3 |
| | | | | 3 |
| | | | y | 5 |
| | | 1.1.2 | | 7 |
| | | | | 7 |
| | | | dessn.entry.sim module | 7 |
| | | 1.1.3 | dessn.model package | 7 |
| | | | Submodules | 7 |
| | | | dessn.model.edge module | 7 |
| | | | dessn.model.model module | 8 |
| | | | dessn.model.node module | 0 |
| | | 1.1.4 | dessn.simple package | 2 |
| | | | Subpackages | 2 |
| | | | Submodules | 6 |
| | | | dessn.simple.example module | 6 |
| | | | dessn.simple.exampleIntegral module | 7 |
| | | | dessn.simple.exampleLatent module | 9 |
| | | 1.1.5 | dessn.simulation package | 1 |
| | | | Submodules | 1 |
| | | | dessn.simulation.observationFactory module | 1 |
| | | | dessn.simulation.simulation module | 1 |
| | | 1.1.6 | dessn.toy package | 1 |
| | | | Submodules | 2 |
| | | | dessn.toy.edges module | 2 |
| | | | dessn.toy.latent module | 2 |
| | | | dessn.toy.observed module | 2 |
| | | | dessn.toy.toyModel module | 3 |
| | | | dessn.toy.transformations module | 3 |
| | | | dessn.toy.underlying module | 4 |
| | | 1.1.7 | dessn.utility package | 4 |
| | | | Submodules | 4 |
| | | | dessn.utility.hdemcee module | 4 |
| | | | dessn.utility.newtonian module | 4 |
| Рy | thon N | Aodule | 2. Index | 5 |

Index 27

CHAPTER

ONE

DESSN PACKAGE

Welcome to the **DESSN** code base.

1.1 Subpackages

1.1.1 dessn.chain package

Submodules

dessn.chain.chain module

```
class dessn.chain.chain.ChainConsumer
    Bases: object
```

A class for consuming chains produced by an MCMC walk

add_chain (chain, parameters=None, name=None)

Add a chain to the consumer.

Parameters chain: strIndarray

The chain to load. Normally a numpy.ndarray, but can also accept a string. If a string is found, it interprets the string as a filename and attempts to load it in.

parameters : list[str], optional

A list of parameter names, one for each column (dimension) in the chain.

name: str, optional

The name of the chain. Used when plotting multiple chains at once.

Returns ChainConsumer

Itself, to allow chaining calls.

get_parameter_text (lower, maximum, upper)

Generates LaTeX appropriate text from marginalised parameter bounds.

Parameters lower : float

The lower bound on the parameter

maximum: float

The value of the parameter with maximum probability

upper: float

```
The upper bound on the parameter
```

```
get summary()
```

Gets a summary of the marginalised parameter distributions.

Returns list of dictionaries

One entry per chain, parameter bounds stored in dictionary with parameter as key

plot (figsize='COLUMN', filename=None, display=False, rainbow=False, contour_kwargs=None)
Plot the chain

Parameters figsize: strltuple(float), optional

The figure size to generate. Accepts a regular two tuple of size in inches, or one of several key words. The default value of COLUMN creates a figure of appropriate size of insertion into an A4 LaTeX document in two-column mode. PAGE creates a full page width figure. String arguments are not case sensitive.

filename: str, optional

If set, saves the figure to this location

display: bool

If True, shows the figure using plt.show().

rainbow: bool

If true, forces the use of rainbow colours when displaying multiple chains. By default, under a certain number of chains to show, this method uses a predefined list of colours.

contour_kwargs : dict

A dictionary of optional arguments to pass to the plot_contour() function.

Returns figure

the matplotlib figure

plot_bars (ax, parameter, chain_row, bins=50, colour='#222222', fit_values=None) Method responsible for plotting the marginalised distributions

Parameters ax: matplotlib axis

Upon which the plot is drawn

parameter : str

The parameter label, if it exists

chain row: np.ndarray

The data corresponding to the parameter

bins: int, optional

The number of bins to use. Default value is overridden by plot ()

colour: str

The colour to use when plotting. Default value is overridden

Returns float

the maximum value of the histogram plot (used to ensure vertical spacing)

Plots contours of the probability surface between two parameters

Parameters ax: figure.axis

The axis to plot to

x : np.ndarray

The x axis array of data

y : np.ndarray

The y axis array of data

bins: int, optional

The number of bins to use. Overridden by the plot () method.

sigmas: np.array, optional

The σ contour levels to plot. Defaults to [0.5, 1, 2, 3]. Number of contours shown decreases with the number of chains to show.

colour: str(hex code), optional

The colour to plot the contours in. Overridden by the plot () method.

fit values: np.array, optional

An array representing the lower bound, maximum, and upper bound of the marignliased parameters

force contourf: bool

Can force the plotting method to plot filled contours even when it would normally be disabled. It is normally disabled when plotting multiple chains.

dessn.chain.demoOneChain module

class dessn.chain.demoOneChain.DemoOneChain

The single chain demo for Chain Consumer. Dummy class used to get documentation caught by sphinx-apidoc, it servers no other purpose.

Running this file in python creates a random data set, representing a single MCMC chain, such as you might get from emcee.

First, we create a consumer and load the chain, and tell it to plot the chain without knowing the parameter labels. It is set to so that the plot should pop up. To continue running the code, close the plot.

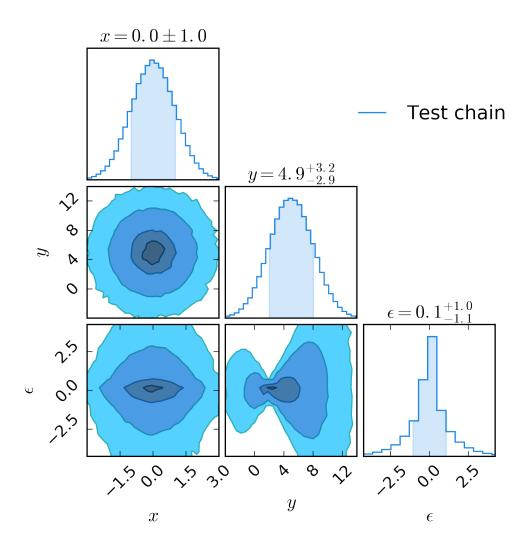
The second thing we do is create a different consumer, and load the chain into it. We also supply the parameter labels. By default, as we only have a single chain, contours are filled, the marginalised histograms are shaded, and the best fit parameter bounds are shown as axis titles.

The plot for this is saved to the png file below:

dessn.chain.demoThreeChains module

class dessn.chain.demoThreeChains.DemoThreeChains

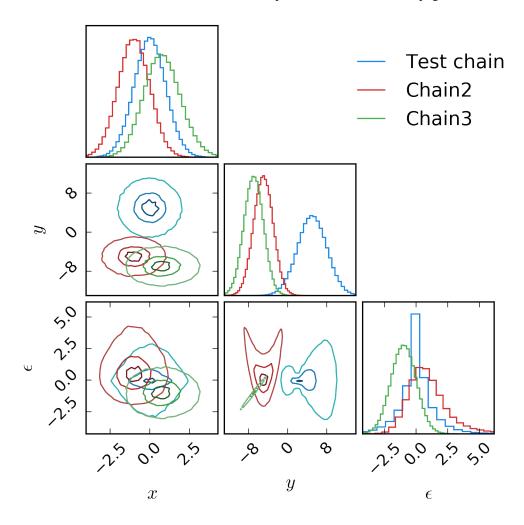
The multiple chain demo for Chain Consumer. Dummy class used to get documentation caught by sphinx-apidoc, it servers no other purpose.



Running this file in python creates three random data sets, representing three separate chains.

First, we create a consumer and load the first two chains, and tell it to plot with filled contours.

The second thing we do is create a different consumer, and load all three chains into it. We also supply the parameter labels the first time we load in a chain. The plot for this is saved to the png file below:



dessn.chain.demoTwoDisjointChains module

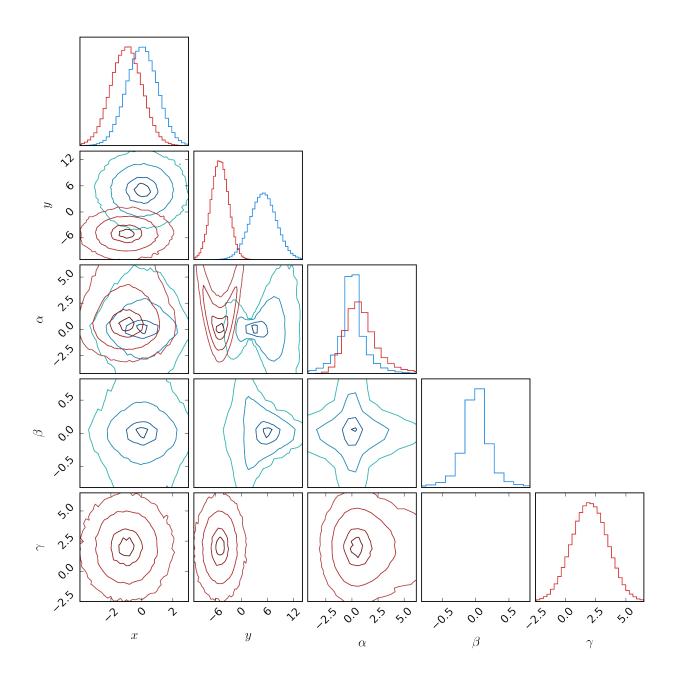
 ${\bf class} \; {\tt dessn.chain.demoTwoDisjointChains.DemoTwoDisjointChains}$

The multiple chain demo for Chain Consumer. Dummy class used to get documentation caught by sphinx-apidoc, it servers no other purpose.

Running this file in python creates two random data sets, representing two separate chains, for two separate models.

It is sometimes the case that we wish to compare models which have partially overlapping parameters. For example, we might fit a model which depends has cosmology dependend on Ω_m and Ω_Λ , where we assume w=1. Alternatively, we might assume flatness, and therefore fix Ω_Λ but instead vary the equation of state w. The good news is, you can visualise them both at once!

The second thing we do is create a consumer, and load both chains into it. As we have different parameters for each chain we supply the right parameters for each chain. The plot for this is saved to the png file below:



1.1.2 dessn.entry package

Submodules

dessn.entry.sim module

1.1.3 dessn.model package

Submodules

dessn.model.edge module

```
class dessn.model.edge.Edge (probability_of, given)
```

Bases: object

An edge connection one or more parameters to one or more different parameters.

An edge is a connection between parameters (not Nodes), and signifies a joint probability distribution. That is, if in our mathematical definition of our model, we find the term P(a,b|c,d,e), this would be represented by a single edge. Similarly, P(a|b)P(b|c,d) would be two edges.

Parameters probability_of: str or list[str]

The dependent parameters. With the example P(a,b|c,d), this input would be ['a','b'].

given: str or list[str]

In the example P(a,b|c,d), this input would be ['c','d'].

get log likelihood(data)

Gets the log likelihood of this edge.

For example, if we had

$$P(a,b|c,d) = \frac{1}{\sqrt{2\pi}d} \exp\left(-\frac{(ab-c)^2}{d^2}\right),\,$$

we could implement this function as return -np.log(np.sqrt(2*np.pi)*data['d']) (data['a']*data['b'] - data['c'])**2/(data['d']**2)

Returns float

the log likelihood given the supplied data and the model parametrisation.

class dessn.model.edge.EdgeTransformation(probability_of, given)

Bases: dessn.model.edge.Edge

This specialised edge is used to connect to transformation nodes.

A transformation edge does not give a likelihood, but - as it is a known transformation - returns a dictionary when *get_transformation* is invoked that is injected into the data dictionary given to regular edges.

See LuminosityToAdjusted for a simple example.

Parameters probability of : str or list[str]

The dependent parameters. With the example P(a,b|c,d), (assuming the functional form is a delta), this input would be ['a','b'].

given: str or list[str]

```
In the example P(a, b|c, d), (assuming the functional form is a delta), this input would be ['c', 'd'].
```

get_log_likelihood(data)

get_transformation(data)

Calculates the new parameters from the given data

Returns dict

a dictionary containing a value for each parameter given in transform_to

dessn.model.model module

```
class dessn.model.model.Model(model_name)
    Bases: object
```

A generalised model for use in arbitrary situations.

A model is, at heart, simply a collection of nodes and edges. Apart from simply being a container in which to place nodes and edges, the model is also responsible for figuring out how to connect edges (which map to parameters) with the right nodes, for sorting edges such that when an edge is evaluated all its required data has been generated by other nodes or edges, for managing the emcee running, and also for generating the visual PGMs.

It is thus a complex class, and I expect, as of writing this summary, it contains numerous bugs.

Parameters model name: str

The model name, used for serialisation

```
add_edge (edge)
```

Adds an edge into the models collection of edges

```
\mathtt{add}\_\mathtt{node} \ (node)
```

Adds a node into the models collection of nodes.

```
chain_plot(**kwargs)
```

Creates a chain plot of the model's chain.

This is my own implementation of a corner plot.

```
Parameters kwargs: dict
```

Arguments to pass to the plot () method. See the method link for more details.

Returns figure

a matplotlib figure of the chain plot

chain_summary()

Gets a summary of fit parameters through ChainConsumer and the get_summary() method. See the method link for more details

```
corner (filename=None, display=True)
```

Creates a corner plot from the model's chain.

Parameters filename: str, optional

If set, saves the figure to this filename

display: bool, optional

If true, shows the plot. If false, simply return the figure

Returns figure

a matplotlib figure of the corner plot

finalise()

Finalises the model.

This method runs consistency checks on the model (making sure there are not orphaned nodes, edges to parameters that do not exist, etc), and in doing so links the right edges to the right nodes and determines the order in which edges should be evaluated.

You can manually call this method after setting all nodes and edges to confirm as early as possible that the model is valid. If you do not call it manually, this method is invoked by the model when requesting concrete information, such as the PGM or model fits.

```
fit_model (num_walkers=None, num_steps=5000, num_burn=3000, temp_dir=None, save_interval=300)
Uses emcee to fit the supplied model.
```

This method sets an emcee run using the EnsembleSampler and manual chain management to allow for very high dimension models. MPI running is detected automatically for less hassle, and chain progress is serialised to disk automatically for convenience.

This method works... but is still a work in progress

```
Parameters num walkers: int, optional
```

The number of walkers to run. If not supplied, it defaults to eight times the model dimensionality

```
num_steps : int, optional
The number of steps to run
num_burn : int, optional
```

The number of steps to discard for burn in

```
temp_dir : str
```

If set, specifies a directory in which to save temporary results, like the emcee chain

```
save_interval: float
```

The amount of seconds between saving the chain to file. Setting to None disables serialisation.

Returns ndarray

```
The final flattened chain of dimensions (num_dimensions, num_walkers * (num_steps - num_burn))
fig
```

The corner plot figure returned from corner.corner(...)

```
get_pgm (filename=None)
```

Renders (and returns) a PGM of the current model.

```
Parameters filename: str, optional
```

if the filename is set, the PGM is saved to file in the top level plots directory.

Returns daft.PGM

The daft PGM class, for further customisation if required.

dessn.model.node module

```
class dessn.model.node (node_name, names, labels, parameter_type)
    Bases: object
```

A node represented on a PGM model. Normally encapsulated by a single parameter, or several related parameters

The Node class can essentially be thought of as a wrapper around a parameter or variable in your model. However, as some parameters are highly related (for example, flux and flux error), Nodes allow you to declare multiple parameters.

This class is an abstract class, and cannot be directly instantiated. Instead, instantiate one of the provided subclasses, as detailed below.

Parameters node name: str

The node name, only used when plotting on a PGM

names : str or list[str]

The model parameter encapsulated by the node, or list of model parameters

labels: str or list[str]

Latex ready labels for the given names. Used in the PGM and corner plots.

parameter_type : NodeType

The type of subclass. Informs the model how to utilise the node.

class dessn.model.node.NodeLatent (node_name, names, labels)

Bases: dessn.model.node.Node

A node representing a latent, or hidden, variable in our model.

Given infinitely powerful computers, these nodes would not be necessary, for they represent marginalisation over unknown / kidden / latent parameters in the model, and we would simple integrate them out when computing the likelihood probability. However, this is not the case, and it is more efficient to simply incorporate latent parameters into our model and essentially marginalise over them using Monte Carlo integration. We thus trade explicit numerical integration in each step of our calculation for increased dimensionality.

For examples on why and how to use latent parameters, see the examples beginning in Example.

Parameters node name: str

The node name, only used when plotting on a PGM

names : str or list[str]

The model parameter encapsulated by the node, or list of model parameters

labels: str or list[str]

Latex ready labels for the given names. Used in the PGM and corner plots.

get num latent()

The number of latent parameters to include in the model.

Running MCMC requires knowing the dimensionality of our model, which means knowing how many latent parameters (realisations of an underlying hidden distribution) we require.

For example, if we observe a hundred supernova drawn from an underlying supernova distribution, we would have to realise a hundred latent variables - one per data point.

Returns int

the number of latent parameters required by this node

class dessn.model.node.NodeObserved (node_name, names, labels, datas)

Bases: dessn.model.node.Node

A node representing one or more observed variables

This node is used for all observables in the model. In addition to a normal node, it also contains data, which can be in arbitrary format. This data is what is given to the incoming and outgoing node edges to calculate likelihoods.

Parameters node_name : str

The node name, only used when plotting on a PGM

names : str or list[str]

The model parameter encapsulated by the node, or list of model parameters

labels: str or list[str]

Latex ready labels for the given names. Used in the PGM and corner plots.

datas : object or list[obj]

One data object for each supplied parameter name. **Must** be the same length as names if names is a list.

get_data()

Returns a dictionary containing keys of the parameter names and values of the parameter data object

class dessn.model.node.NodeTransformation (node name, names, labels)

Bases: dessn.model.node.Node

A node representing a variable transformation.

This node essentially represents latent variables which are fully determined - their probability is given by a delta function. Examples of this might be the luminosity distance, as it is known exactly when given cosmology and redshift. Or it might represent a conversion between observed flux and actual flux, given we have a well defined flux correction.

On a PGM, this node would be represented by a point, not an ellipse.

Note that this node declares all associated parameters to be transformation parameters, although the transformation functions themselves are defined by the edges into and out of this node.

Parameters node_name : str

The node name, only used when plotting on a PGM

names: str or list[str]

The model parameter encapsulated by the node, or list of model parameters

labels : str or list[str]

Latex ready labels for the given names. Used in the PGM and corner plots.

class dessn.model.node.NodeType

Bases: enum. Enum

LATENT = <NodeType.LATENT: 3>

OBSERVED = <NodeType.OBSERVED: 2>

TRANSFORMATION = < NodeType.TRANSFORMATION: 4>

UNDERLYING = <NodeType.UNDERLYING: 1>

class dessn.model.node.NodeUnderlying (node_name, names, labels)

Bases: dessn.model.node.Node

A node representing an underlying parameter in your model.

On the PGM, these nodes would be at the very top, and would represent the variables we are trying to fit for, such as Ω_M .

These nodes are required to implement the abstract method get_log_prior

Parameters node name: str

The node name, only used when plotting on a PGM

names: str or list[str]

The model parameter encapsulated by the node, or list of model parameters

labels: str or list[str]

Latex ready labels for the given names. Used in the PGM and corner plots.

get_log_prior(data)

Returns the log prior for the parameter.

Parameters data: dic

A dictionary containing all data and the model parameters being tested at a given step in the MCMC chain. For this class, if the class was instantiated with a name of "omega_m", the input dictionary would have the key "omega_m", and the value of "omega_m" at that particular step in your chain.

Returns float

the log prior probability given the current value of the parameters

1.1.4 dessn.simple package

This module is designed to give a step by step overview of a very simplified example Bayesian model.

The basic example model is laid out in the parent class <code>Example</code>, and there are three implementations. The first implementation, <code>ExampleIntegral</code>, shows how the problem might be approached in a simple model, where numerical integration is simply done as part of the likelihood calculation.

However, if there are multiple latent parameters, we get polynomial growth of the number of numerical integrations we have to do, and so this does not scale well at all.

This leads us to the implementation in <code>ExampleLatent</code>, where we use the MCMC algorithm to essentially do Monte Carlo integration via marginalisation. This means we do not need to perform the numerical integration in the likelihood calculation, however the cost of doing so is increase dimensionality of our MCMC.

Finally, the <code>ExampleModel</code> implementation shows how the <code>ExampleLatent</code> class might be written to make use of Nodes. This is done in preparation for more complicated models, which will have more than one layer and needs to be configurable.

Subpackages

dessn.simple.modelbased package

I have placed the class based example for implementing the simplified model into its own module, so that the documentation generating for the simple module does not get cluttered with all the small classes this module will have.

The primary class to look at in code is the *ExampleModel* class.

I should finally note that in order to demonstrate parameter transformations, I have modified the model used in the previous two examples (<code>ExampleIntegral</code> and <code>ExampleLatent</code>) to also include a luminosity transformation, where I simply halve the luminosity before converting it to flux. Physically, this could represent a perfect 50% mirror absorption on the primary telescope mirror.

Submodules

```
dessn.simple.modelbased.exampleModel module
```

 ${\bf class} \; {\tt dessn.simple.modelbased.example Model.\bf{Example Model} \\$

Bases: dessn.model.model.Model

An implementation of ExampleLatent using classes instead of procedural code.

The model is set up by declaring nodes, the edges between nodes, and then calling finalise on the model to verify its correctness.

This is the primary class in this package, and you can see that other classes inherit from either *Node* or from *Edge*.

I leave the documentation for *Node* and *Edge* to those classes, and encourage viewing the code directly to understand exactly what is happening.

Running this file in python first generates a PGM of the model, and then runs emcee and creates a corner plot:

 ${\bf class} \; {\tt dessn.simple.modelbased.example Model.Flux To Luminosity}$

Bases: dessn.model.edge.Edge

get_log_likelihood(data)

class dessn.simple.modelbased.exampleModel.LatentLuminosity (n=100)

Bases: dessn.model.node.NodeLatent

get_num_latent()

class dessn.simple.modelbased.exampleModel.LuminosityToAdjusted

Bases: dessn.model.edge.EdgeTransformation

get_transformation(data)

class dessn.simple.modelbased.exampleModel.LuminosityToSupernovaDistribution

Bases: dessn.model.edge.Edge

get_log_likelihood(data)

class dessn.simple.modelbased.exampleModel.ObservedFlux (n=100)

Bases: dessn.model.node.NodeObserved

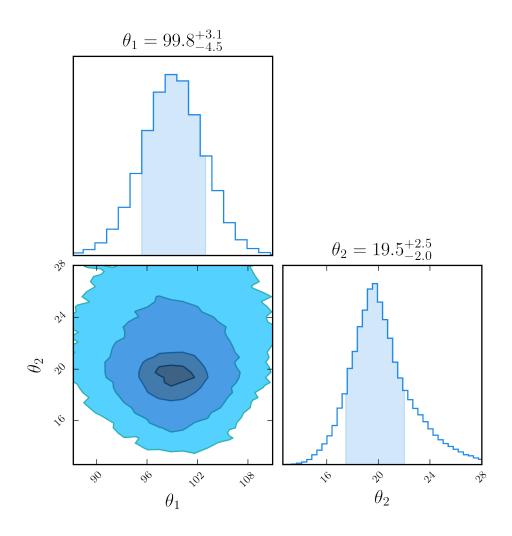
class dessn.simple.modelbased.exampleModel.UnderlyingSupernovaDistribution

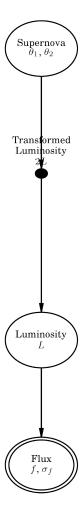
Bases: dessn.model.node.NodeUnderlying

get_log_prior(data)

 ${\bf class} \; {\tt dessn.simple.modelbased.example Model. Useless Transformation}$

 $Bases: \ \textit{dessn.model.node.NodeTransformation}$





Submodules

dessn.simple.example module

class dessn.simple.example.Example $(n=30, theta_1=100.0, theta_2=20.0)$

Bases: object

Setting up the math for some examples.

Let us assume that we are observing supernova that are drawn from an underlying supernova distribution parameterised by θ , where the supernova itself simply a luminosity L. We measure the luminosity of multiple supernovas, giving us an array of measurements D. If we wish to recover the underlying distribution of supernovas from our measurements, we wish to find $P(\theta|D)$, which is given by

$$P(\theta|D) \propto P(D|\theta)P(\theta)$$

Note that in the above equation, we realise that $P(D|L) = \prod_{i=1}^N P(D_i|L_i)$ as our measurements are independent. The likelihood $P(D|\theta)$ is given by

$$P(D|\theta) = \prod_{i=1}^{N} \int_{-\infty}^{\infty} P(D_i|L_i) P(L_i|\theta) dL_i$$

We now have two distributions to characterise. Let us assume both are gaussian, that is our observed luminosity x_i has gaussian error σ_i from the actual supernova luminosity, and the supernova luminosity is drawn from an underlying gaussian distribution parameterised by θ .

$$P(D_i|L_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x_i - L_i)^2}{2\sigma_i^2}\right)$$
$$P(L_i|\theta) = \frac{1}{\sqrt{2\pi}\theta_2} \exp\left(-\frac{(L_i - \theta_1)^2}{2\theta_2^2}\right)$$

This gives us a likelihood of

$$P(D|\theta) = \prod_{i=1}^{N} \frac{1}{2\pi\theta_2\sigma_i} \int_{-\infty}^{\infty} \exp\left(-\frac{(x_i - L_i)^2}{2\sigma_i^2} - \frac{(L_i - \theta_1)^2}{2\theta_2^2}\right) dL_i$$

Working in log space for as much as possible will assist in numerical precision, so we can rewrite this as

$$\log(P(D|\theta)) = \sum_{i=1}^{N} \left[\log\left(\int_{-\infty}^{\infty} \exp\left(-\frac{(x_i - L_i)^2}{2\sigma_i^2} - \frac{(L_i - \theta_1)^2}{2\theta_2^2} \right) dL_i \right) - \log(2\pi\theta_2\sigma_i) \right]$$

Parameters n: int, optional

The number of supernova to 'observe'

theta_1: float, optional

The mean of the underlying supernova luminosity distribution

theta_2: float, optional

The standard deviation of the underlying supernova luminosity distribution

do emcee (nwalkers=None, nburn=None, nsteps=None)

Abstract method to configure the emcee parameters

static get data $(n=50, theta \ 1=100.0, theta \ 2=20.0, scale=1.0, seed=1)$

```
get likelihood(theta, data, error)
```

Abstract method to return the log likelihood

get_posterior (theta, data, error)

Gives the log posterior probability given the supplied input parameters.

Parameters theta: array of model parameters

 \mathbf{data} : array of length n

An array of observed luminosities

error: array of length n

An array of observed luminosity errors

Returns float

the log posterior probability

get_prior(theta)

Get the log prior probability given the input.

The prior distribution is currently implemented as flat prior.

Parameters theta: array of model parameters

Returns float

the log prior probability

plot observations()

Plot the observations and observation distribution.

dessn.simple.exampleIntegral module

```
class dessn.simple.exampleIntegral.ExampleIntegral (n=10, theta_1=100.0, theta_2=30.0)
    Bases: dessn.simple.example.Example
```

An example implementation using integration over a latent parameter.

Building off the math from Example Creating this class will set up observations from an underlying distribution. Invoke emcee by calling the object. In this example, we perform the marginalisation inside the likelihood calculation, which gives us dimensionality only of two (the length of the θ array). However, this is at the expense of performing the marginalisation over dL_i , as this requires computing n integrals for each step in the MCMC.

Note that I believe my numerical integration is not working properly, hence the weird output results. The moral of the story - it takes far, far longer to run than any other way of doing it, should still be the take home message from this.

Parameters n: int, optional

The number of supernova to 'observe'

theta_1: float, optional

The mean of the underlying supernova luminosity distribution

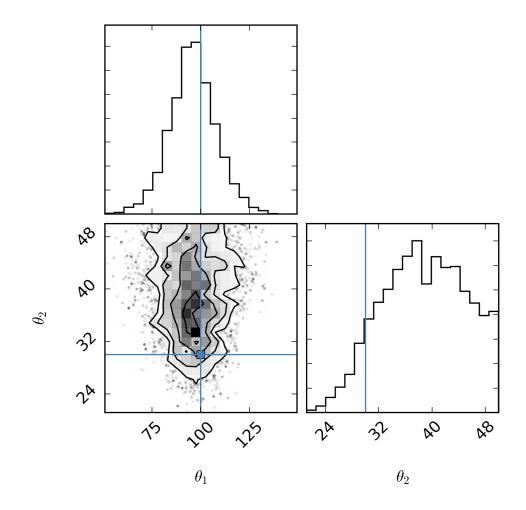
theta_2: float, optional

The standard deviation of the underlying supernova luminosity distribution

do_emcee (nwalkers=16, nburn=500, nsteps=1000)

Run the *emcee* chain and produce a corner plot.

Saves a png image of the corner plot to plots/exampleIntegration.png.



Parameters nwalkers: int, optional

The number of walkers to use. Minimum of four.

nburn: int, optional

The burn in period of the chains.

nsteps: int, optional

The number of steps to run

get_likelihood(theta, data, error)

Gets the log likelihood given the supplied input parameters.

Parameters theta: array of size 2

An array representing $[\theta_1, \theta_2]$

 \mathbf{data} : array of length n

An array of observed luminosities

error: array of length n

An array of observed luminosity errors

Returns float

the log likelihood probability

dessn.simple.exampleLatent module

 $\textbf{class} \; \texttt{dessn.simple.exampleLatent} \; \textbf{.ExampleLatent} \; (\textit{n=30}, \textit{theta_1=100.0}, \textit{theta_2=20.0})$

Bases: dessn.simple.example.Example

An example implementation using marginalisation over latent parameters.

Building off the math from Example, instead of performing the integration numerically in the computation of the likelihood, we can instead use Monte Carlo integration by simply setting the latent parameters \vec{L} as free parameters, giving us

$$\log \left(P(D|\theta, \vec{L}) \right) = -\sum_{i=1}^{N} \left[\frac{(x_i - L_i)^2}{\sigma_i^2} + \frac{(L_i - \theta_1)^2}{\theta_2^2} + \log(2\pi\theta_2\sigma_i) \right]$$

Creating this class will set up observations from an underlying distribution. Invoke emcee by calling the object. In this example, we marginalise over L_i after running our MCMC, and so we no longer have to compute integrals in our chain, but instead have dimensionality of 2 + n, where n are the number of observations.

Parameters n: int, optional

The number of supernova to 'observe'

theta_1: float, optional

The mean of the underlying supernova luminosity distribution

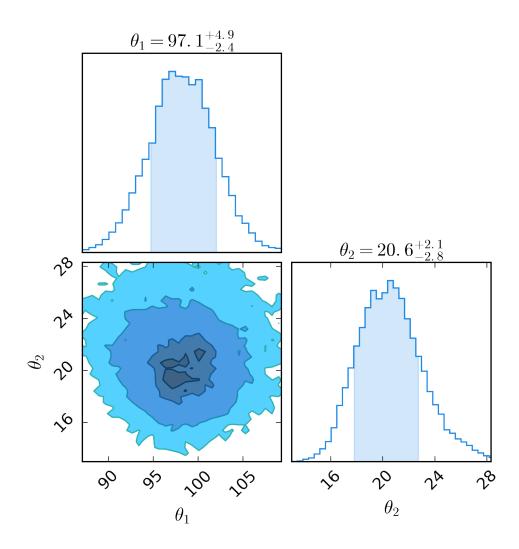
theta_2: float, optional

The standard deviation of the underlying supernova luminosity distribution

do_emcee (*nwalkers*=500, *nburn*=500, *nsteps*=1000)

Run the emcee chain and produce a corner plot.

Saves a png image of the corner plot to plots/exampleLatent.png.



```
Parameters nwalkers: int, optional
```

The number of walkers to use.

nburn: int, optional

The burn in period of the chains.

nsteps: int, optional

The number of steps to run

get_likelihood(theta, data, error)

Gets the log likelihood given the supplied input parameters.

Parameters theta: array of length 2 + n

An array representing $[\theta_1, \theta_2, \vec{L}]$

 \mathbf{data} : array of length n

An array of observed luminosities

 \mathbf{error} : array of length n

An array of observed luminosity errors

Returns float

the log likelihood probability

1.1.5 dessn.simulation package

Submodules

dessn.simulation.observationFactory module

```
class dessn.simulation.observationFactory.ObservationFactory(**kwargs)
    Bases: object
    check_kwargs()
    get_observations(num)
        Still needs massive refactoring
```

dessn.simulation.simulation module

```
class dessn.simulation.simulation
Bases: object

get simulation (num trans=30)
```

1.1.6 dessn.toy package

This module will contain the original toy model when implemented.

Submodules

dessn.toy.edges module

```
class dessn.toy.edges.ToCount
    Bases: dessn.model.edge.EdgeTransformation
    get_transformation (data)
{\bf class} dessn.toy.edges.{f ToFlux}
    Bases: dessn.model.edge.EdgeTransformation
    get transformation(data)
class dessn.toy.edges.ToLuminosity
    Bases: dessn.model.edge.Edge
    get_log_likelihood(data)
class dessn.toy.edges.ToLuminosityDistance
    Bases: dessn.model.edge.EdgeTransformation
    get_transformation (data)
class dessn.toy.edges.ToRate
    Bases: dessn.model.edge.Edge
    get_log_likelihood(data)
class dessn.toy.edges.ToRedshift
    Bases: dessn.model.edge.Edge
    get_log_likelihood(data)
class dessn.toy.edges.ToType
    Bases: dessn.model.edge.Edge
    get_log_likelihood(data)
dessn.toy.latent module
class dessn.toy.latent.Luminosity(n)
    Bases: dessn.model.node.NodeLatent
    get_num_latent()
class dessn.toy.latent.Redshift(n)
    Bases: dessn.model.node.NodeLatent
    get_num_latent()
class dessn.toy.latent.Type (n)
    Bases: dessn.model.node.NodeLatent
    get_num_latent()
dessn.toy.observed module
class dessn.toy.observed.ObservedCounts(counts)
    Bases: dessn.model.node.NodeObserved
```

class dessn.toy.observed.ObservedRedshift (redshifts)

Bases: dessn.model.node.NodeObserved

class dessn.toy.observed.ObservedType (types)

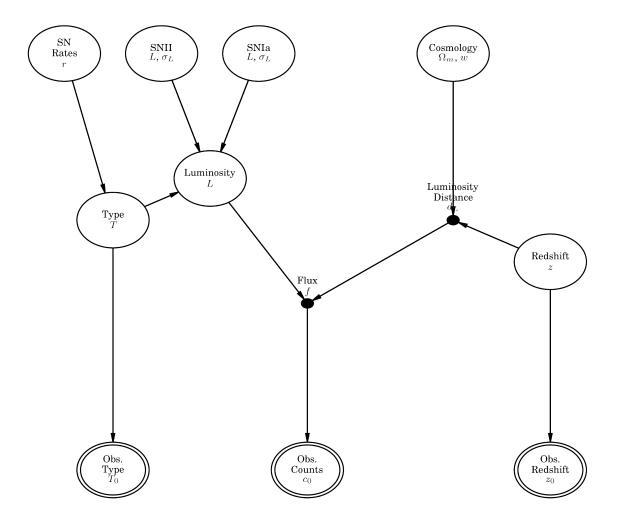
Bases: dessn.model.node.NodeObserved

dessn.toy.toyModel module

class dessn.toy.toyModel.ToyModel

Bases: dessn.model.model.Model

A modified toy model.



dessn.toy.transformations module

 ${\bf class} \; {\tt dessn.toy.transformations.Flux}$

Bases: dessn.model.node.NodeTransformation

```
class dessn.toy.transformations.LuminosityDistance
    Bases: dessn.model.node.NodeTransformation
dessn.toy.underlying module
class dessn.toy.underlying.Cosmology
    Bases: dessn.model.node.NodeUnderlying
    get_log_prior(data)
class dessn.toy.underlying.SupernovalIDist
    Bases: dessn.model.node.NodeUnderlying
    get_log_prior(data)
class dessn.toy.underlying.SupernovalaDist
    Bases: dessn.model.node.NodeUnderlying
    get_log_prior(data)
class dessn.toy.underlying.SupernovaRate
    Bases: dessn.model.node.NodeUnderlying
    get_log_prior(data)
1.1.7 dessn.utility package
Submodules
dessn.utility.hdemcee module
class dessn.utility.hdemcee.EmceeWrapper(sampler)
    Bases: object
    get_results()
    run_chain (num_steps, num_burn, num_walkers, num_dim, start=None, save_interval=300,
               save_dim=None, temp_dir=None)
dessn.utility.newtonian module
class dessn.utility.newtonian.NewtonianPosition (nodes, edges, top=None, bottom=None)
    Bases: object
```

fit (plot=False)iterate (p, v, i)

d

```
dessn.1
dessn.chain,1
dessn.chain.chain, 1
dessn.chain.demoOneChain,3
dessn.chain.demoThreeChains, 3
dessn.chain.demoTwoDisjointChains,5
dessn.entry, 7
dessn.entry.sim, 7
dessn.model, 7
dessn.model.edge, 7
dessn.model.model, 8
dessn.model.node, 10
dessn.simple, 12
dessn.simple.example, 16
dessn.simple.exampleIntegral, 17
dessn.simple.exampleLatent, 19
dessn.simple.modelbased, 12
dessn.simple.modelbased.exampleModel,
       13
dessn.simulation, 21
dessn.simulation.observationFactory, 21
dessn.simulation.simulation, 21
dessn.tov, 21
dessn.toy.edges, 22
dessn.toy.latent,22
dessn.toy.observed, 22
dessn.toy.toyModel, 23
dessn.toy.transformations, 23
dessn.toy.underlying, 24
dessn.utility, 24
dessn.utility.hdemcee, 24
dessn.utility.newtonian, 24
```

26 Python Module Index

| desn.toy.toyModel (module), 23 desn.toy.toyModel (module), 23 desn.toy.toyModel (module), 24 desn.tuility (module), 24 desn.tuility, mewtonian (module), 26 desn.toy.toyModel (module), 24 desn.tuility, mewtonian (module), 24 desn.tuility, mewtonian (module), 24 desn.tuility, mewtonian (module), 26 desn.toy.toyModel (module), 24 desn.tuility, mewtonian (module), 24 desn.tuility, mewtonian (module), 26 desn.toylity (module), 27 desn.tuility, mewtonian (module), 28 desn.toylity (module), 29 desn.tuility, mewtonian (module), 29 desn.toylity (module), 23 desn.toylity (module), 24 desn.tuility, mewtonian (module), 26 desn.tuility, mewtonian (module), 26 desn.tuility, member (module), 27 desn.tuility, member (module), 28 desn.toylity (module), 29 desn.tuility, member (module), 29 desn.tuility, member (module), 29 desn.tuility, member (module), 21 desn.tuility, member (module), 21 desn.tuility, member (module), 23 desn.toylity, member (module), 21 desn.tuility, member (module), 23 desn.tuility, member (module), 21 desn.tuility, member (module), 23 desn.tuility, member (module), 24 desn.tuility, member (module), 26 desn.tuility, member (module), 26 desn.tuility, member (module), 26 desn. | Α | dessn.toy (module), 21 |
|--|--|--|
| dd_edge() (dessn.model.model.Model method), 8 ddd_node() (dessn.model.model.Model method), 8 ddd_node() (dessn.model.model.Model method), 8 ddan_node() (dessn.model.model.Model method), 8 dhain_summary() (dessn.model.model.Model method), 8 dhain_summary() (dessn.model.model.Model method), 8 chain_consumer (class in dessn.chain.chain), 1 dheck_kwargs() (dessn.simulation.observationFactory.Obsermethod), 21 dessn.chain_demo(l.model.Model method), 8 cosmology (class in dessn.chain.demo(l.model.Model method), 8 cosmology (class in dessn.chain.demoOneChain), 3 demoTheeChains (class dessn.chain.demoThreeChains), 3 demoTheeChains (class dessn.chain.demoThreeChains), 3 demoTheeChains (class dessn.chain.demoThreeChains), 3 dessn.chain.demoThreeChains (class dessn.chain.demoThreeChains), 3 dessn.chain.demoThreeChains (class dessn.chain.demoThreeChains), 3 dessn.chain.demoThreeChains (module), 1 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoThreeChains, 3 dessn.chai | add_chain() (dessn.chain.chain.ChainConsumer method), | |
| dessn.toy.toyModel (module), 23 dessn.toy.toyModel (module), 23 dessn.toy.transformations (module), 24 dessn.utility.phemcee (module), 26 dessn.chain.demoThreeChains), 3 (class dessn.chain.demoThreeChains), 3 (class (module), 1 dessn.chain.demoThreeChains), 3 (class (module), 1 dessn.chain.demoThreeChains), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoThreeChains (module), 5 dessn.entry (module), 7 dessn.entry (module), 7 dessn.entry (module), 7 dessn.entry (module), 7 dessn.entry (module), 1 dessn.model.model (module), 10 dessn.simple (module | 1 | · · · · · · · · · · · · · · · · · · · |
| dessn.toy.transformations (module), 23 dessn.toy.underlying (module), 24 dessn.utility, newtonian (module, 32 dessn.chain.demoThreeChains), 3 dessn.chain.demoThreeChains, 4 fetzel Class in dessn.simple.exampleLatent, ExampleLatent, ExampleLatent, Class in dessn.simple.exampleLatent, Class in dessn.simple.exampleLatent, 19 fexampleIntegral (class in dessn.simple.exampleLatent, Class | add_edge() (dessn.model.model.Model method), 8 | |
| dessn.toy.underlying (module), 24 dessn.utility | add_node() (dessn.model.model.Model method), 8 | |
| dessn.utility (module), 24 dessn.utility.hdemcee (method), 19 dessn.utility.hdemcee (dessn.simple.exampleLatent.ExampleIntegral dessn.utility.hdemcee (dessn.simple.exampleLatent.ExampleLatent method), 19 lessn.chain.demoThreeChains, (ass in dessn.simple.exampleLatent.ExampleLatent method), 19 lessn.chain.demoThreeChains, (ass in dessn.simple.exampleLatent.ExampleLatent method), 19 lessn.chain.demoThreeChains, (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoThreeChains (module), 7 lessn.model.model (module), 19 lessn.simple.example (module), 10 lessn.simple.example (module), 10 lessn.simple.example (module), 1 | 2 | · |
| dessn.chain.demotel.model.model method), 8 ChainConsumer (class in dessn.chain.chain), 1 check_kwargs() (dessn.simulation.observationFactory.ObservationFactory.ObservationPactory.Obser | ${\sf G}$ | |
| chain_summary() (dessn.model.model.Model method), 8 ChainConsumer (class in dessn.chain.chain), 1 check_kwargs() (dessn.simulation.observationFactory.ObservationFactory.ObservationFactory.ObservationFactory.ObservationFactory.ObservationFactory.ObservationPactory.ObservationFact | chain plot() (dessn.model.model.Model method), 8 | |
| ChainConsumer (class in dessn.chain.chain), 1 check_ wargs() (dessn.simulation.observationFactory.ObservationFactory.ObservationPactory.Observatio | • | |
| method), 17 do_emce() (dessn.model.model.model.model method), 8 Cosmology (class in dessn.toy.underlying), 24 DemoOneChain (class in dessn.chain.demoOneChain), 3 DemoThreeChains (class dessn.chain.demoThreeChains), 3 DemoTwoDisjointChains (class dessn.chain.demoTwoDisjointChains), 5 Dessn.chain.demoTwoDisjointChains), 5 Dessn.chain.demoTwoDisjointChains), 5 Dessn.chain.demoTwoDisjointChains, 5 Dessn.chain.demoTwoDisjointChains, 5 Dessn.chain.demoTwoDisjointChains, 5 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 7 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 7 Dessn.chain.demoTwoDisjointChains, 8 DemoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 7 Dessn.chain.demoTwoDisjointChains, 8 DemoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 7 DemoTwoDisjointChains, 7 Dessn.chain.demoTwoDisjointChains, 7 DemoTwoDisjointChains, 7 DemoTwoDisjointChains, 7 DemoTwoDisjointChains, 7 DemoTwoDisjointChains, 8 DemoTwoDisjointCha | ChainConsumer (class in desen chain chain) 1 | |
| method), 17 do_emce() (dessn.model.model.model.model method), 8 Cosmology (class in dessn.toy.underlying), 24 DemoOneChain (class in dessn.chain.demoOneChain), 3 DemoThreeChains (class dessn.chain.demoThreeChains), 3 DemoTwoDisjointChains (class dessn.chain.demoTwoDisjointChains), 5 Dessn.chain.demoTwoDisjointChains), 5 Dessn.chain.demoTwoDisjointChains), 5 Dessn.chain.demoTwoDisjointChains, 5 Dessn.chain.demoTwoDisjointChains, 5 Dessn.chain.demoTwoDisjointChains, 5 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 7 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 7 Dessn.chain.demoTwoDisjointChains, 8 DemoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 7 Dessn.chain.demoTwoDisjointChains, 8 DemoTwoDisjointChains, 6 Dessn.chain.demoTwoDisjointChains, 7 DemoTwoDisjointChains, 7 Dessn.chain.demoTwoDisjointChains, 7 DemoTwoDisjointChains, 7 DemoTwoDisjointChains, 7 DemoTwoDisjointChains, 7 DemoTwoDisjointChains, 8 DemoTwoDisjointCha | check_kwargs() (dessn.simulation.observationFactory.Obse | .do .emcee() (dessn.simple.example.Example method), 16 ervationFactory do_emcee() (dessn.simple.exampleIntegral.ExampleIntegra |
| Cosmology (class in dessn.toy.underlying), 24 DemoOneChain (class in dessn.chain.demoOneChain), 3 DemoThreeChains (class | method), 21 | |
| DemoOneChain (class in dessn.chain.demoOneChain), 3 DemoThreeChains (class | | do_emcee() (dessn.simple.exampleLatent.ExampleLatent method), 19 |
| DemoOneChain (class in dessn.chain.demoOneChain), 3 DemoThreeChains (class | D | F |
| dessn.chain.demoThreeChains), 3 DemoTwoDisjointChains (class dessn.chain.demoTwoDisjointChains), 5 dessn (module), 1 dessn.chain (module), 1 dessn.chain.demoTwoDisjointChains (module), 3 dessn.chain.demoOneChain (module), 3 dessn.chain.demoTwoDisjointChains (module), 5 dessn.entry (module), 7 dessn.entry (module), 7 dessn.entry.sim (module), 7 dessn.model.model (module), 7 dessn.model.model (module), 8 dessn.simple (module), 10 dessn.simple.example (module), 12 dessn.simple.exampleIntegral (module), 12 dessn.simple.exampleIntegral (module), 12 dessn.simple.exampleIntegral (module), 12 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.exampleLatent (module), 21 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.exampleLatent (class in dessn.simple.exampleLatent), 19 fix (class in dessn.simple.exampleLatent), 19 fit () (dessn.model.model.Model method), 9 fit () (dessn.model.model.Model method), 9 filux (class in dessn.simple.exampleModel), 13 dessn.simple.example (class in dessn.simple.exampleLatent), 19 fit () (dessn.model.model.Model method), 9 fit () (dessn.simple.exampleModel (module), 13 dessn.simple.example (dass in dessn.simple.exampleLatent), 19 fit () (dessn.model.exampleModel method), 9 fit () (dessn.model.model.Model method), 9 fit () (dessn.simple.example.exampleModel), 13 dessn.simple.example (dass in dessn.simple.example (class in dessn.simple.exampleLatent), 19 fit () (dessn.model.model.Model method), 9 fit () (dessn.simple.example.modelbased.exampleModel), 13 de | DemoOneChain (class in dessn.chain.demoOneChain), 3 | |
| DemoTwoDisjointChains (class dessn.chain.demoTwoDisjointChains), 5 dessn (module), 1 dessn.chain (module), 1 dessn.chain (module), 1 dessn.chain.demoTwoDisjointChains (module), 3 dessn.chain.demoOneChain (module), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoOneChain (module), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoOneChain (module), 3 dessn.chain.demoTwoDisjointChains (module), 5 dessn.entry (module), 7 dessn.entry (module), 7 dessn.entry.sim (module), 7 dessn.model (module), 7 dessn.model.dege (module), 7 dessn.model.node (module), 8 dessn.simple (module), 10 dessn.simple (module), 12 dessn.simple (module), 12 dessn.simple.exampleIntegral (module), 19 dessn.simple.exampleIntegral (module), 19 dessn.simple.exampleIntegral (module), 12 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.exampleAidel (dessn.model.node.NodeObserved method), 11 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.exampleLatent (module), 13 dessn.simple.exampleAidel (dessn.model.node.NodeObserved method), 11 dessn.simple.exampleAidel (dessn.simple.example static | DemoThreeChains (class in | |
| dessn.chain.demoTwoDisjointChains), 5 dessn (module), 1 dessn.chain (module), 1 dessn.chain (module), 1 dessn.chain.demoOneChain (module), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoTwoDisjointChains (module), 5 dessn.chain.demoTwoDisjointChains (module), 5 dessn.chain.demoTwoDisjointChains (module), 5 dessn.chain.demoTwoDisjointChains (module), 5 dessn.entry (module), 7 dessn.entry.sim (module), 7 dessn.model.model (module), 7 dessn.model.model (module), 7 dessn.model.node (module), 8 dessn.simple (module), 10 dessn.simple (module), 12 dessn.simple.exampleLatent (module), 13 dessn.simple.exampleLatent (class in dessn.simple.exampleLatent), 19 finalise() (dessn.model.model.Model method), 9 fit() (dessn.model.nodel.Model m | dessn.chain.demoThreeChains), 3 | |
| dessn.chain.demoTwoDisjointChains), 3 dessn.chain (module), 1 dessn.chain (module), 1 dessn.chain (module), 1 dessn.chain.demoOneChain (module), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoTwoDisjointChains (module), 5 dessn.entry (module), 7 dessn.model (module), 7 dessn.model.model (module), 7 dessn.model.node (module), 8 dessn.simple (module), 10 dessn.simple (module), 12 dessn.simple exampleIntegral (module), 13 dessn.simple (module), 12 dessn.simple.exampleIntegral (class in dessn.simple.exampleLatent), 19 fix (dessn.simple.modelbased.exampleModel), 11 dessn.simple.modelbased.exampleModel), 13 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.example dessn.simple.example dessn.simple.modelbased.exampleModel), 13 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.exampleLatent (class in dessn.simple.exampleLatent), 19 fix (dessn.model.model.model.Model method), 9 fit() (dessn.model.model.Model method), 9 flux (class in dessn.simple.exampleModel method), 9 flux (class in dessn.simple.example model.Model method), 9 flux (class in dessn.simple.example model.model.Model method), 9 flux (class in dessn.simple.example model.model.model.model.model.model.model.model.model.model.model.model.model.model.model.model.model.model.model.mod | · · | |
| lessn.chain (module), 1 lessn.chain (module), 1 lessn.chain.chain (module), 1 lessn.chain.demoOneChain (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoTwoDisjointChains (module), 5 lessn.entry (module), 7 lessn.model (module), 7 lessn.model.edge (module), 7 lessn.model.model (module), 8 lessn.simple (module), 12 lessn.simple.example (module), 12 lessn.simple.exampleIntegral (module), 17 lessn.simple.exampleLatent (module), 19 lessn.simple.modelbased (module), 12 lessn.simple.modelbased (module), 12 lessn.simple.modelbased (module), 12 lessn.simple.modelbased.exampleModel (module), 13 lessn.simple.modelbased.exampleModel (module), 13 lessn.simple.modelbased.exampleModel (module), 13 lessn.simulation (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simple.modelbased.example (module), 16 lessn.simulation.observationFactory (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simple.models.model.mode | dessn.chain.demoTwoDisjointChains), 5 | |
| lessn.chain (module), 1 lessn.chain.chain (module), 1 lessn.chain.chain (module), 1 lessn.chain.demoOneChain (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoTwoDisjointChains (module), 5 lessn.entry (module), 7 lessn.model (module), 7 lessn.model.model (module), 7 lessn.model.model (module), 8 lessn.model.node (module), 10 lessn.simple (module), 12 lessn.simple.exampleIntegral (module), 17 lessn.simple.exampleIntegral (module), 17 lessn.simple.exampleLatent (module), 19 lessn.simple.modelbased (module), 12 lessn.simple.modelbased.exampleModel (module), 13 lessn.simple.modelbased.exampleModel (module), 13 lessn.simulation (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simulation.observationFactory (module), 21 | | |
| lessn.chain.chain (module), 1 lessn.chain.demoOneChain (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoThreeChains (module), 3 lessn.chain.demoTwoDisjointChains (module), 5 lessn.entry (module), 7 lessn.entry.sim (module), 7 lessn.model (module), 7 lessn.model.model (module), 7 lessn.model.model (module), 8 lessn.model.model (module), 10 lessn.simple (module), 12 lessn.simple.example (module), 16 lessn.simple.exampleIntegral (module), 17 lessn.simple.exampleLatent (module), 19 lessn.simple.modelbased (module), 12 lessn.simple.modelbased.exampleModel (module), 13 lessn.simulation (module), 21 lessn.simulation.observationFactory (module), 21 lessn.simulatio | | |
| dessn.chain.demoOneChain (module), 3 dessn.chain.demoThreeChains (module), 3 dessn.chain.demoThreeChains (module), 5 dessn.entry (module), 7 dessn.entry.sim (module), 7 dessn.model (module), 7 dessn.model.edge (module), 7 dessn.model.model (module), 8 dessn.model.node (module), 10 dessn.simple (module), 12 dessn.simple.example (module), 16 dessn.simple.exampleIntegral (module), 17 dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased (module), 12 dessn.simple.modelbased (module), 13 dessn.simple.exampleAndel (mo | | |
| lessn.chain.demoTwoDisjointChains (module), 5 lessn.entry (module), 7 lessn.model (module), 7 lessn.model.model (module), 7 lessn.model.model (module), 8 lessn.model.node (module), 10 lessn.simple (module), 12 lessn.simple.example (module), 16 lessn.simple.exampleIntegral (module), 17 lessn.simple.exampleLatent (module), 19 lessn.simple.modelbased (module), 12 lessn.simple.modelbased (module), 13 lessn.simple.modelbased.exampleModel (module), 13 lessn.simulation (module), 21 lessn.simulation.observationFactory (module), 21 mothed (module), 16 get_data() (dessn.model.mode.NodeObserved method), 11 get_data() (dessn.simple.example.example station method), 11 get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example station method), 11 get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example station method), 11 get_data() (dessn.simple.example.example station method), 11 get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example station method), 11 | | * |
| Finalise() (dessn.model.model.model method), 9 fit() (dessn.model. | | |
| dessn.entry.sim (module), 7 dessn.model (module), 7 dessn.model.edge (module), 7 dessn.model.model (module), 8 dessn.simple (module), 10 dessn.simple (module), 12 dessn.simple.example (module), 16 dessn.simple.exampleIntegral (module), 17 dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 dessn.simulation.observationFactory (module), 21 finalise() (dessn.model.model.Model method), 9 fit() (dessn.model.model.Model method), 9 fit() (dessn.model.model.Model method), 9 fit() (dessn.model.nodel.Model method), 9 fit() (dessn.model.nodel.Model.Model method), 9 fit() (dessn.model.nodel.Mod | · · · · · · · · · · · · · · · · · · · | _ |
| dessn.model (module), 7 dessn.model.edge (module), 8 dessn.model.node (module), 8 dessn.simple (module), 10 dessn.simple.example (module), 12 dessn.simple.exampleIntegral (module), 17 dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 mothed) 16 finalise() (dessn.model.model.Model method), 9 fit() (dessn.model.model.Model.Model method), 9 fit() (dessn.model.model.Mo | · · · · · · · · · · · · · · · · · · · | F |
| fit() (dessn.utility.newtonian.NewtonianPosition method) 24 fit_model() (dessn.utility.newtonian.NewtonianPosition method) 24 fit_model() (dessn.model.model.Model method), 9 flux (class in dessn.toy.transformations), 23 fluxToLuminosity (class ir dessn.simple.exampleIntegral (module), 17 flessn.simple.exampleLatent (module), 19 flessn.simple.modelbased (module), 12 flessn.simple.modelbased.exampleModel (module), 13 flessn.simulation (module), 21 flessn.simulation.observationFactory (module), 21 fit() (dessn.utility.newtonian.NewtonianPosition method) 24 fit_model() (dessn.model.model.Model method), 9 flux (class in dessn.toy.transformations), 23 fluxToLuminosity (class ir dessn.simple.modelbased.exampleModel), 13 G get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example stations) get_data() (dessn.simple.example.example stations) method) | | finalise() (dessn.model.model.Model method), 9 |
| dessn.model.model (module), 8 dessn.model.node (module), 10 dessn.simple (module), 12 dessn.simple.example (module), 16 dessn.simple.exampleIntegral (module), 17 dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 Total dessn.model.mod | | fit() (dessn.utility.newtonian.NewtonianPosition method), |
| dessn.model.node (module), 10 dessn.simple (module), 12 dessn.simple.example (module), 16 dessn.simple.exampleIntegral (module), 17 dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 dessn.simulation.observationFactory (module), 21 fit_model() (dessn.model.model.Model method), 9 Flux (class in dessn.toy.transformations), 23 FluxToLuminosity (class in dessn.simple.modelbased.exampleModel), 13 G get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example station of the properties of | | 24 |
| dessn.simple (module), 12 dessn.simple.example (module), 16 dessn.simple.exampleIntegral (module), 17 dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 dessn.simulation.observationFactory (module), 21 Flux (class in dessn.toy.transformations), 23 FluxToLuminosity (class ir dessn.simple.modelbased.exampleModel), 13 G get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example station of the property of the pr | | fit_model() (dessn.model.model.Model method), 9 |
| dessn.simple.example (module), 16 dessn.simple.exampleIntegral (module), 17 dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 dessn.simulation.observationFactory (module), 21 FluxToLuminosity (class in dessn.simple.modelbased.exampleModel), 13 dessn.simple.modelbased.exampleModel), 13 get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example station method), 14 | | Flux (class in dessn.toy.transformations), 23 |
| dessn.simple.exampleIntegral (module), 17 dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 dessn.simulation.observationFactory (module), 21 dessn.simple.modelbased.exampleModel), 13 get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example station method), 14 | • | FluxToLuminosity (class in |
| dessn.simple.exampleLatent (module), 19 dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 dessn.simulation.observationFactory (module), 21 get_data() (dessn.simple.example.example station (module), 21 get_data() (dessn.simple.example.example) get_data() (dessn.simple.example) get_data() (dessn.simple.example) | | <u> </u> |
| dessn.simple.modelbased (module), 12 dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 G get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.Example station of the destruction of the | | 13 |
| dessn.simple.modelbased.exampleModel (module), 13 dessn.simulation (module), 21 dessn.simulation.observationFactory (module), 21 get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.example station of the destruction of the de | | G |
| dessn.simulation (module), 21 get_data() (dessn.model.node.NodeObserved method), 11 get_data() (dessn.simple.example.Example station (module), 21 get_data() (dessn.simple.example.example station (module), 21 | • | |
| dessn.simulation.observationFactory (module), 21 get_data() (dessn.simple.example.Example static | | |
| mathad) 16 | | |
| acson.simulation.simulation (module), 21 | lessn.simulation.simulation (module), 21 | method), 16 |

| get_likelihood() (dessn.simple.example.Example | |
|--|--|
| method), 16 | method), 2 [ntegral transformation] (dessn.model.edge.EdgeTransformation |
| method), 19 | method), 8 |
| | atemet_transformation() (dessn.simple.modelbased.exampleModel.Luminosity |
| method), 21 | method), 13 |
| | get_transformation() (dessn.toy.edges.ToCount method), |
| get_log_likelihood() (dessn.model.edge.EdgeTransformat | |
| method), 8 | <pre>get_transformation() (dessn.toy.edges.ToFlux method),</pre> |
| get_log_likelihood() (dessn.simple.modelbased.exampleN | |
| method), 13 | get_transformation() (dessn.toy.edges.ToLuminosityDistance |
| get_log_likelihood() (dessn.simple.modelbased.exampleN | Model.Luminmath Td Supernova Distribution |
| method), 13 | , |
| get_log_likelihood() (dessn.toy.edges.ToLuminosity | , [|
| method), 22 | iterate() (dessn.utility.newtonian.NewtonianPosition |
| get_log_likelihood() (dessn.toy.edges.ToRate method), | method), 24 |
| 22 | |
| get_log_likelihood() (dessn.toy.edges.ToRedshift | t L |
| method), 22 | LATENT (dessn.model.node.NodeType attribute), 11 |
| get_log_likelihood() (dessn.toy.edges.ToType method), | LatentLuminosity (class in |
| 22 | dessn.simple.modelbased.exampleModel), |
| get_log_prior() (dessn.model.node.NodeUnderlying | 13 |
| method), 12 | Luminosity (class in dessn.toy.latent), 22 |
| get_log_prior() (dessn.simple.modelbased.exampleModel | l.Underlying Supernova Distributions, 22. Luminosity Distributions (little States and Luminosity Distributions), |
| method), 13 | 23 |
| get_log_prior() (dessn.toy.underlying.Cosmology | LuminosityToAdjusted (class in |
| method), 24 | dessn.simple.modelbased.exampleModel), |
| get_log_prior() (dessn.toy.underlying.SupernovaIaDist | 13 |
| method), 24 | LuminosityToSupernovaDistribution (class in |
| <pre>get_log_prior() (dessn.toy.underlying.SupernovaIIDist</pre> | dessn.simple.modelbased.exampleModel), |
| | 13 |
| get_log_prior() (dessn.toy.underlying.SupernovaRate method), 24 | M |
| get_num_latent() (dessn.model.node.NodeLatent | |
| method), 10 | Model (class in dessn.model.model), 8 |
| get_num_latent() (dessn.simple.modelbased.exampleMod | lel N atentI uminosity |
| method), 13 | • • |
| get_num_latent() (dessn.toy.latent.Luminosity method), | NewtonianPosition (class in dessn.utility.newtonian), 24 |
| 22 | Node (class in dessn.model.node), 10 |
| get_num_latent() (dessn.toy.latent.Redshift method), 22 | NodeLatent (class in dessn.model.node), 10 |
| get_num_latent() (dessn.toy.latent.Type method), 22 | NodeObserved (class in dessn.model.node), 11 |
| get_observations() (dessn.simulation.observationFactory. | NodeTransformation (class in dessn.model.node), 11 ObservationFactory |
| method), 21 | Node Type (class in dessn.model.node), 11 |
| get_parameter_text() (dessn.chain.chain.ChainConsumer | NodeUnderlying (class in dessn.model.node), 11 |
| method), 1 | 0 |
| get_pgm() (dessn.model.model.Model method), 9 | |
| get_posterior() (dessn.simple.example.Example method), | ObservationFactory (class in |
| 17 | dessn.simulation.observationFactory), 21 |
| get_prior() (dessn.simple.example.Example method), 17 | OBSERVED (dessn.model.node.NodeType attribute), 11 |
| get_results() (dessn.utility.hdemcee.EmceeWrapper | ObservedCounts (class in dessn.toy.observed), 22 |
| method), 24 | ObservedFlux (class in |
| get_simulation() (dessn.simulation.simulation.Simulation | |
| method), 21 | 13 ObservedRedshift (class in dessn.tov.observed), 22 |
| | ODSCI VEUIXEUSIIIII (CIASS III UESSII.UV.UUSEI VEU). ZZ |

28 Index

ObservedType (class in dessn.toy.observed), 23 Р plot() (dessn.chain.chain.ChainConsumer method), 2 plot bars() (dessn.chain.chain.ChainConsumer method), plot_contour() (dessn.chain.chain.ChainConsumer method), 2 plot_observations() (dessn.simple.example.Example method), 17 R Redshift (class in dessn.toy.latent), 22 (dessn.utility.hdemcee.EmceeWrapper run chain() method), 24 S Simulation (class in dessn.simulation.simulation), 21 SupernovaIaDist (class in dessn.toy.underlying), 24 SupernovaIIDist (class in dessn.toy.underlying), 24 SupernovaRate (class in dessn.toy.underlying), 24 Т ToCount (class in dessn.toy.edges), 22 ToFlux (class in dessn.toy.edges), 22 ToLuminosity (class in dessn.toy.edges), 22 ToLuminosityDistance (class in dessn.toy.edges), 22 ToRate (class in dessn.toy.edges), 22 ToRedshift (class in dessn.toy.edges), 22 ToType (class in dessn.toy.edges), 22 ToyModel (class in dessn.toy.toyModel), 23 TRANSFORMATION (dessn.model.node.NodeType attribute), 11 Type (class in dessn.toy.latent), 22 U UNDERLYING (dessn.model.node.NodeType attribute), 11 UnderlyingSupernovaDistribution in dessn.simple.modelbased.exampleModel), UselessTransformation (class in dessn.simple.modelbased.exampleModel), 13

Index 29