dessn Documentation

Release 0.0.1

dessn

March 01, 2016

CHAPTER

ONE

DESSN PACKAGE

1.1 Subpackages

1.1.1 dessn.entry package

Submodules

dessn.entry.sim module

Module contents

1.1.2 dessn.model package

Subpackages

dessn.model.nodes package

Submodules

dessn.model.nodes.cosmology module

class `dessn.model.nodes.cosmology.`**`Cosmology`**

Bases: *`dessn.model.node.Node`*

`get_name``()`

class `dessn.model.nodes.cosmology.`**`FlatWCDM`**

Bases: *`dessn.model.nodes.cosmology.Cosmology`*

dessn.model.nodes.supernova module

class `dessn.model.nodes.supernova.`**`Supernova`**

Bases: *`dessn.model.node.Node`*

Abstract supernova which others must implement

`type`

class `dessn.model.nodes.supernova.`**`SupernovaIa`**(*log_luminosity*, *sigma_luminosity*)

Bases: `object`

Models a type Ia supernova

Models the luminosity distribution of type Ia supernovas statically (without internal parameters).

$$P(L) \sim N($$

Parameters **log_luminosity** : float

Represented by the variable μ

sigma_luminosity : float

Represented by the variable σ

get_luminosity_prob (*log_luminosity*)

type ()

dessn.model.nodes.typeProb module

class dessn.model.nodes.typeProb.**TypeProbability**

Bases: *dessn.model.node.Node*

Abstract type probability node, from which all implementations should inherit

get_name ()

class dessn.model.nodes.typeProb.**TypeProbabilitySimple** (*relative_rate=0.333*)

Bases: *dessn.model.nodes.typeProb.TypeProbability*

The Type probability node.

Takes *some information* to determine the probability of of the object in question being a type of supernova.

Parameters **relative_rate** : Optional[str]

Relative rate of SnIa / SnII

class dessn.model.nodes.typeProb.**Types**

Bases: enum.Enum

Possible target types

snII = <Types.snII: 2>

snIa = <Types.snIa: 1>

Module contents

Submodules

dessn.model.abstracts module

dessn.model.model module

class dessn.model.model.**Model**

Bases: object

dessn.model.node module

class dessn.model.node.**Node**

Bases: object

add_dependency (*dependency_class*)

get_dependencies()

get_name()

Module contents

1.1.3 dessn.simple package

Submodules

dessn.simple.exampleIntegral module

class `dessn.simple.exampleIntegral.`**`ExampleIntegral`**(*n=900*, *theta_1=100.0*, *theta_2=30.0*)

Bases: `object`

An example implementation using integration over a latent parameter.

Let us assume that we are observing supernova that a 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.

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

This gives us a likelihood of

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

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

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

Creating this class will set up observations from an underlying distribution. Invoke `emcee` by calling the object. Notice that performing the marginalisation over dL_i requires computing n integrals for each step in the MCMC.

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=20*, *nburn=500*, *nsteps=3000*)

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 $[\theta_1, \theta_2]$

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

`get_posterior`(*theta*, *data*, *error*)

Gives the log posterior probability given the supplied input parameters.

Parameters **theta** : array of size 2

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

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 size 2

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

Returns float

the log prior probability

plot_observations()

Plot the observations and observation distribution.

Module contents

1.1.4 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.`**Simulation**

Bases: `object`

get_simulation(*num_trans=30*)

Module contents

1.2 Module contents

PYTHON MODULE INDEX

d