lightspot Documentation

Release 1.0b3

Eduardo Nunes

Jul 21, 2021

USER GUIDE

CHAPTER

ONE

INSTALLATION

CHAPTER

TWO

CHANGELOG

2.1 1.0 (2020-00-00)

Initial beta release

2.1.1 API

Prior functions

class lightspot.priors.**Dirac**(*x0*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**LogNormal**(*logmu=0.0*, *logsd=1.0*, *ndim=1*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**LogUniform**(*logxmin=0.0*, *logxmax=1.0*, *ndim=1*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**Normal**(*mu=0.0*, *sd=1.0*, *ndim=1*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**Polygon**(*poly*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**Prior**
Bases: object

sample(**args*)

class lightspot.priors.**Quadratic**(*amin=0.0*, *amax=2.0*, *bmin=- 1.0*, *bmax=1.0*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**SameAs**(*prior*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**SineUniform**(*sinxmin=0.0*, *sinxmax=1.0*, *ndim=1*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**ThreeParam**
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**Triangular**(*triangle*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**TruncNormal**(*mu=0.0*, *sd=1.0*, *xmin=0.0*, *xmax=1.0*, *ndim=1*)
Bases: *lightspot.priors.Prior*

sample(**args*)

class lightspot.priors.**Uniform**(*xmin=0.0*, *xmax=1.0*, *ndim=1*)
Bases: *lightspot.priors.Prior*

sample(**args*)

lightspot.priors.**constrain_Pvec**(*Pmin=0*, *Pmax=50*, *sinimin=0*, *sinimax=1*, *vmin=1e-09*, *vmax=1000000000.0*, *rmin=1e-09*, *rmax=1000000000.0*)

Modeler classes

class lightspot.model.**SpotModel**(*t*, *y*, *nspots*, *dy=None*, *use_gpu=False*, *Pvec=None*, *k2=None*, *k4=None*, *c=None*, *d=None*, *same_limb=False*, *lon=None*, *lat=None*, *alpha=None*, *fspot=None*, *tmax=None*, *life=None*, *ingress=None*, *egress=None*, *U=None*, *B=None*, *tstart=None*, *tend=None*)
Bases: object

Modeler class

t

Type time array

y

Type flux array

nspots

Type number of spots

dy

Type flux uncertainties (optional)

Pvec

Type 2-D array containing rotation period at equator and stellar inclination

k2

Type 2nd-order differential rotation coefficient

k4

Type 4th-order differential rotation coefficient

c

Type stellar limb-darkening coefficients

d

Type spot limb-darkening coefficients

same_limb

Type whether to always assume c==d in the model

lon

Type spot longitudes (rad)

lat

Type spot latitudes (rad)

alpha

Type spot radius (rad)

fspot

Type spot-to-photosphere intensity ratio

tmax

Type time of greatest spot area

life

Type spot lifetimes

ingress

Type spot ingress times

egress

Type spot egress times

U

Type unspotted surface flux value

B

Type instrumental blending factor

tstart

Type start time for each of the stitched curves

tend

Type end time for each of the stitched curves

chi(*theta*)

Chi squared of parameters given a set of observations

Parameters **theta** (*array-like with shape (N,)*) – full parameter vector (physical units)

Returns **sse** – sum of squared errors weighted by observation uncertainties

Return type float

property inst_pars

loglike(*x*)

multinest(*sampling_efficiency=0.01*, *const_efficiency_mode=True*, *n_live_points=4000*, ***kwargs*)

property parameters

predict(*t*, *theta*)

Calculates the model flux for given parameter values

Parameters

- **t** (*array-like with shape (n,)*) – time samples where the flux function should be evaluated
- **theta** (*array-like with shape (N,)*) – full parameter vector (physical units)

Returns **yf** – model flux

Return type array-like with shape (n,)

reduced_chi(*theta*)

run(*nlive=1000*, *cores=None*, *filename=None*, ***kwargs*)

sample(*x*)

property spot_pars

property star_pars

2.1.2 Prior functions illustration

```
%matplotlib inline
```

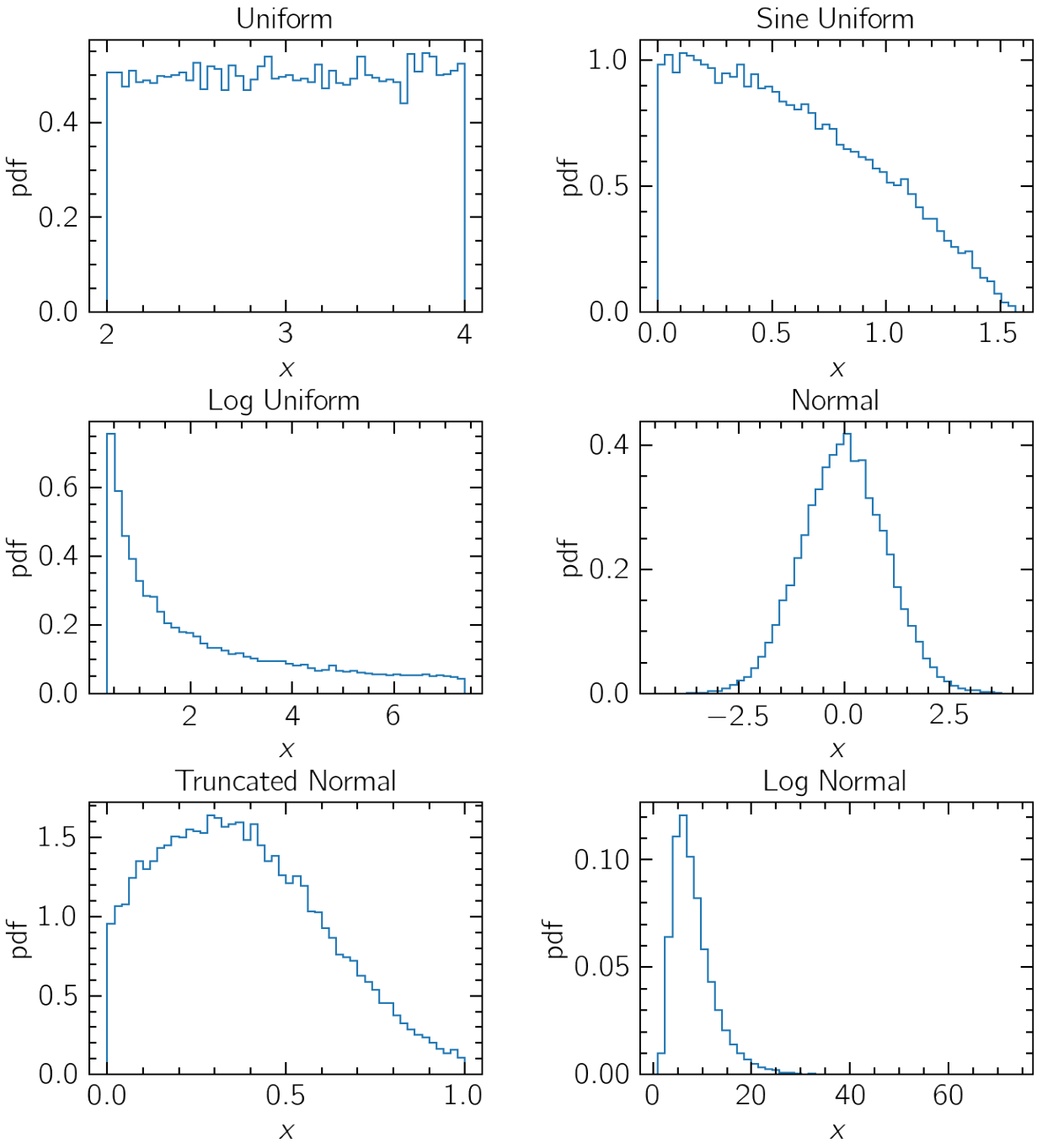
```
%config InlineBackend.figure_format = "retina"
```

```
import numpy as np
import matplotlib.pyplot as plt
from lightspot.priors import *
```

```
plt.rc('lines', linewidth=1.0, linestyle='-', color='black')
plt.rc('font', family='sans-serif', weight='normal', size=12.0)
plt.rc('text', color='black', usetex=True)
plt.rc('text.latex', preamble=r'\usepackage{cmbright}')
plt.rc('axes', edgecolor='black', facecolor='white', linewidth=1.0, grid=False,
       titlesize='x-large', labelsize='x-large', labelweight='normal', labelcolor=
↪'black')
plt.rc('axes.formatter', limits=(-4, 4))
plt.rc(('xtick', 'ytick'), labelsize='x-large', direction='in')
plt.rc('xtick', top=True)
plt.rc('ytick', right=True)
plt.rc(('xtick.major', 'ytick.major'), size=7, pad=6, width=1.0)
plt.rc(('xtick.minor', 'ytick.minor'), size=4, pad=6, width=1.0, visible=True)
plt.rc('legend', numpoints=1, fontsize='x-large', shadow=False, frameon=False)
```

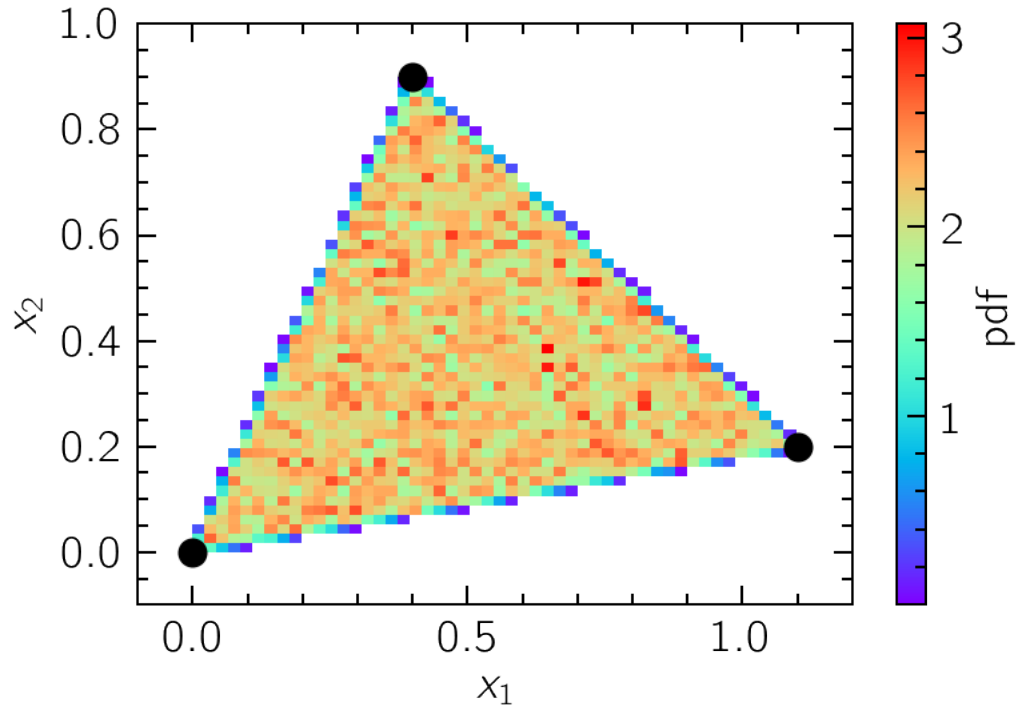
Univariate distributions

```
uni_priors = {
    "Uniform" : Uniform(xmin=2, xmax=4),
    "Sine Uniform" : SineUniform(sinxmin=0, sinxmax=1),
    "Log Uniform" : LogUniform(logxmin=-1, logxmax=2),
    "Normal" : Normal(mu=0, sd=1),
    "Truncated Normal" : TruncNormal(mu=0.3, sd=0.3, xmin=0, xmax=1),
    "Log Normal" : LogNormal(logmu=2, logsd=0.5)
}

fig, axs = plt.subplots(3, 2, figsize=(9.6, 10.8))
fig.subplots_adjust(hspace=0.4, wspace=0.4)

for i, (prior_name, prior) in enumerate(uni_priors.items()):
    samples = prior.sample(*np.random.random(30_000))
    ax = axs[i // 2][i % 2]
    _ = ax.hist(samples, density=True, histtype='step', bins=50)
    ax.set(title=prior_name, xlabel='$x$', ylabel='pdf')
```

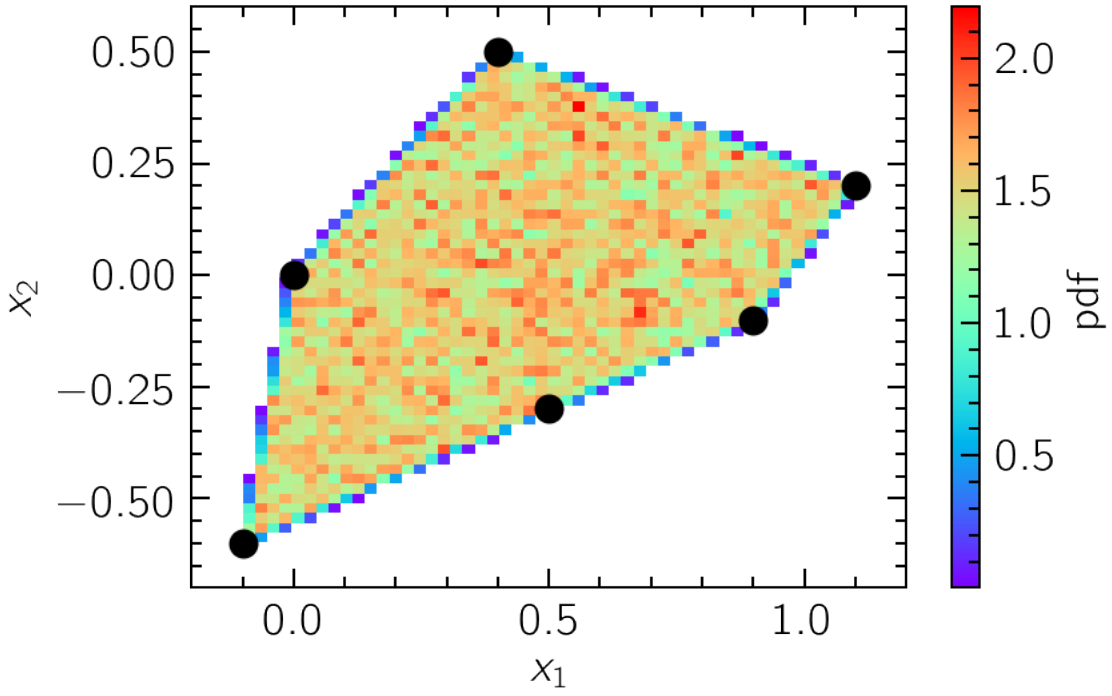

Uniform
pdf
x
Sine Uniform
pdf
x
Log Uniform
pdf
x
Normal
pdf
x
Truncated Normal
pdf
x
Log Normal
pdf
x

Bivariate distributions

```
tri = np.array([[0, 0], [0.4, 0.9], [1.1, 0.2]])
prior = Triangular(tri)
samples = np.array([prior.sample(*val) for val in np.random.random([100_000, 2])])

cmap = plt.cm.get_cmap('rainbow')
cmap.set_under('w')
_ = plt.plot(tri[:, 0], tri[:, 1], 'o', ms=10, color='k')
_ = plt.hist2d(x=samples[:,0], y=samples[:,1], bins=50, density=True, cmap=cmap,␣
↪vmin=1e-3)
_ = plt.colorbar(label="pdf")
_ = plt.gca().set(xlim=(-0.1, 1.2), ylim=(-0.1, 1.0), xlabel="$x_1$", ylabel="$x_2$")
```




```
poly = np.array([[0, 0], [0.4, 0.5], [1.1, 0.2], [0.9, -0.1], [0.5, -0.3], [-0.1, -0.
↪6]])
prior = Polygon(poly)
samples = np.array([prior.sample(*val) for val in np.random.random([100_000, 2])])

cmap = plt.cm.get_cmap('rainbow')
cmap.set_under('w')
_ = plt.plot(poly[:, 0], poly[:, 1], 'o', ms=10, color='k')
_ = plt.hist2d(x=samples[:,0], y=samples[:,1], bins=50, density=True, cmap=cmap,␣
↪vmin=1e-3)
_ = plt.colorbar(label="pdf")
_ = plt.gca().set(xlim=(-0.2, 1.2), ylim=(-0.7, 0.6), xlabel="$x_1$", ylabel="$x_2$")
```


0.50
0.25
0.00
−0.25
−0.50
x_2
0.0
0.5
1.0
x_1
2.0
1.5
1.0
0.5
pdf

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

l

INDEX

T

U

Y