dust$_{e}xtinctionDocumentation$

Release 0.6

Karl Gordon

Nov 15, 2017

Contents

`dust_extinction` is a python package to provide interstellar dust extinction curves.

While there are other python packages that provide some of the extinction curves contained here, the explicit motivation for this package is to provide extinction curves to those using them to model/correct their data and those studying extinction curves directly to better undertand interstellar dust.

This package is developed in the astropy affiliated package template and uses the astropy.modeling framework.

Part I

Installation

To be added

Part II

User Documenation

CHAPTER 1

Model Flavors

There are three differnet types of models: average, R(V)+ dependent prediction, and shape fitting.

1.1 Average models

These models provide averages from the literature with the ability to interpolate between the observed data points. For the Milky Way, one of the R(V) dependent models with R(V) = 3.1 (see next section) are often used for the Milky Way 'average'. Models are provided for the Magellanic Clouds from Gordon et al. (2003). Models for the Milky Way still to be added (both UV/optical/NIR and IR).

1.2 R(V) (+ other variables) dependent prediction models

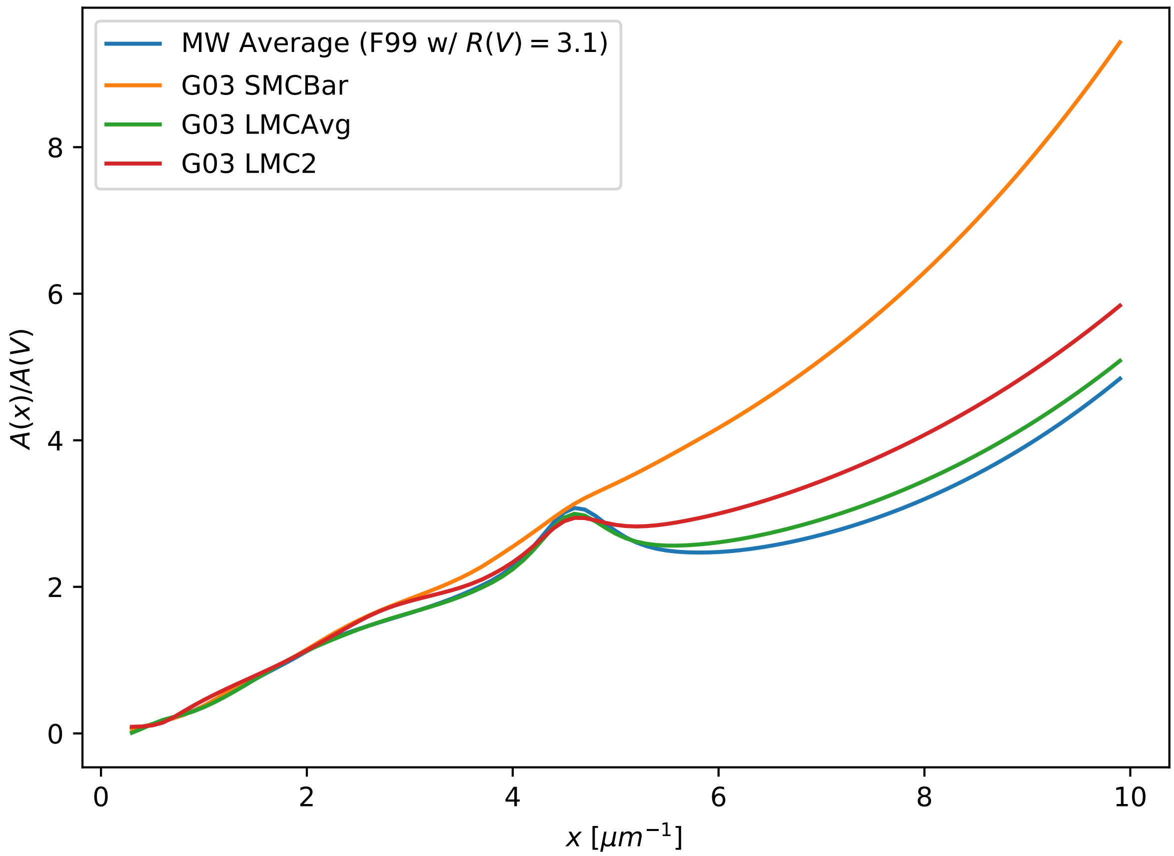
These models provide predictions of the shape of the dust extinction given input variable(s).

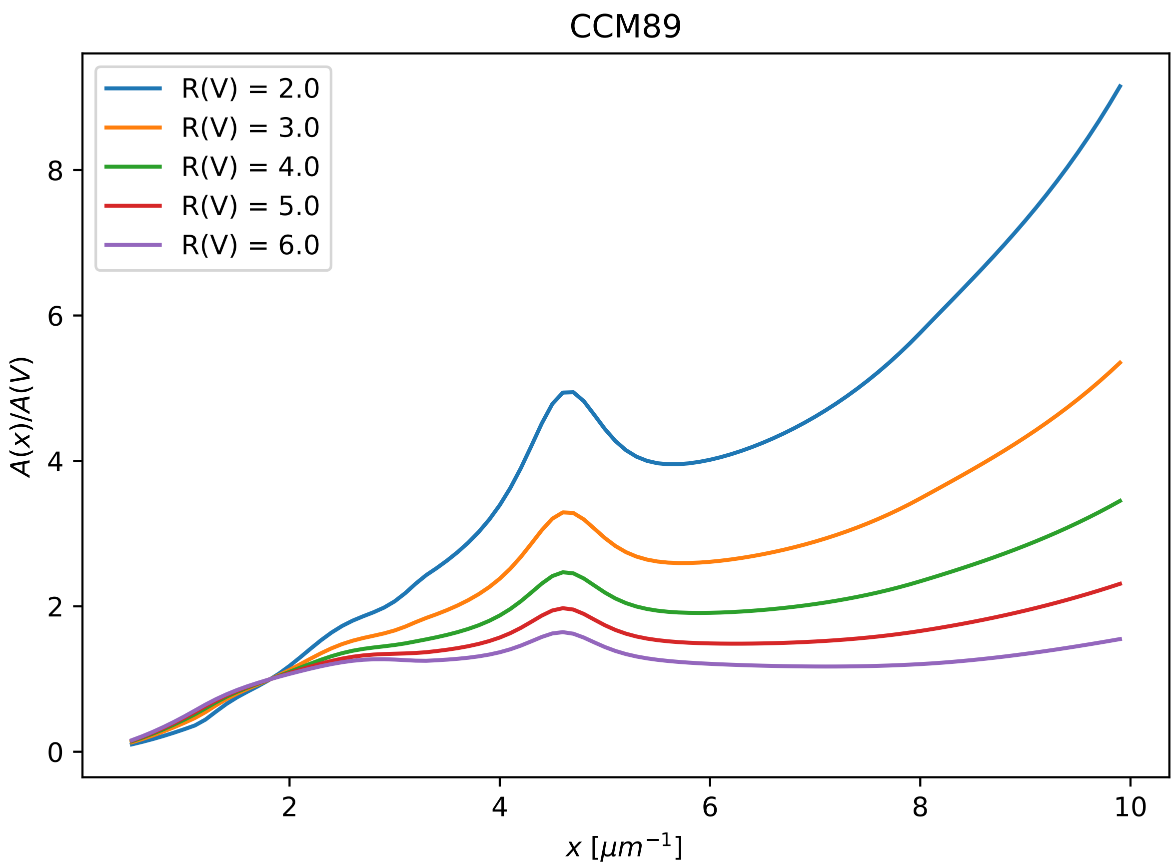
These include CCM89 the original R(V) dependent model from Cardelli, Clayton, and Mathis (1989) and updated versions F99 (Fitzpatrick 1999). These models are based on the average behavior of extinction in the Milky Way.

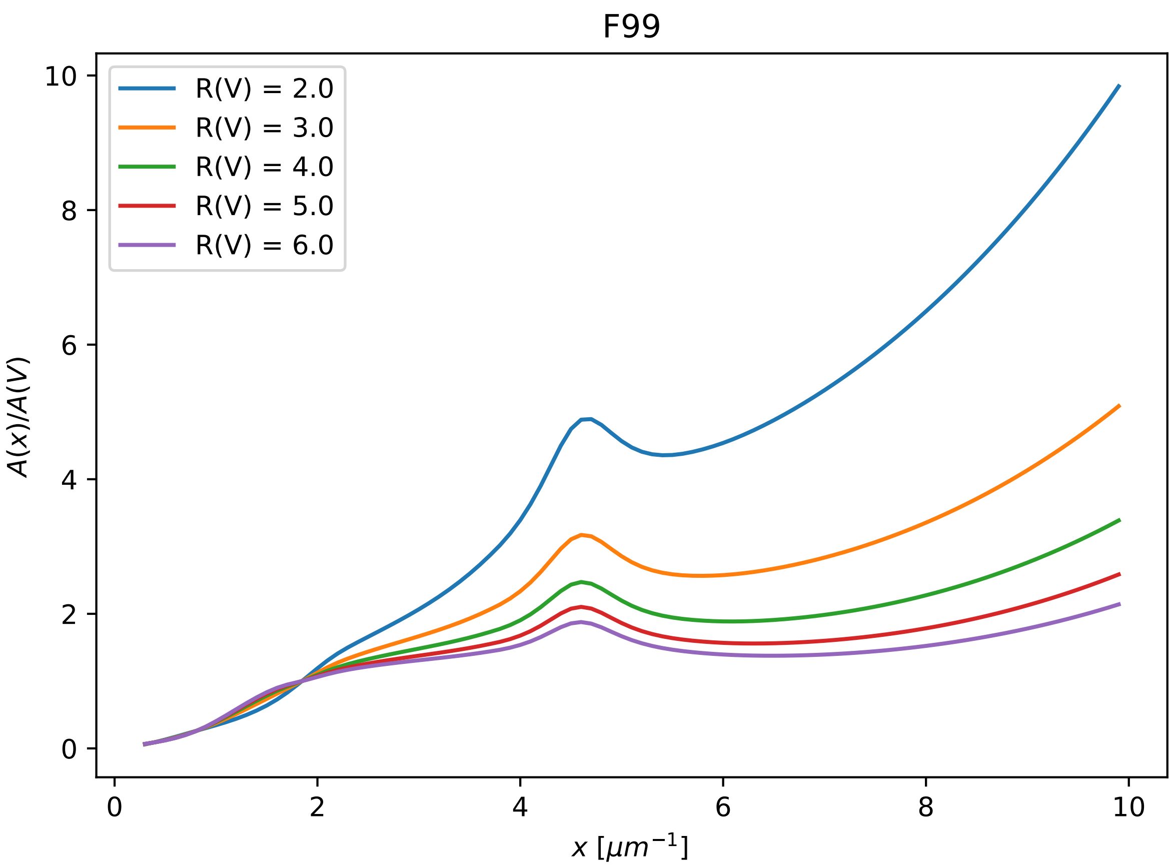
In addition, the (R(V), f_A) two parameter relationship from Gordon et al. (2016) is included. This model is based on the average behavior of extinction in the Milky Way, Large Magellanic Cloud, and Small Magellanic Cloud.

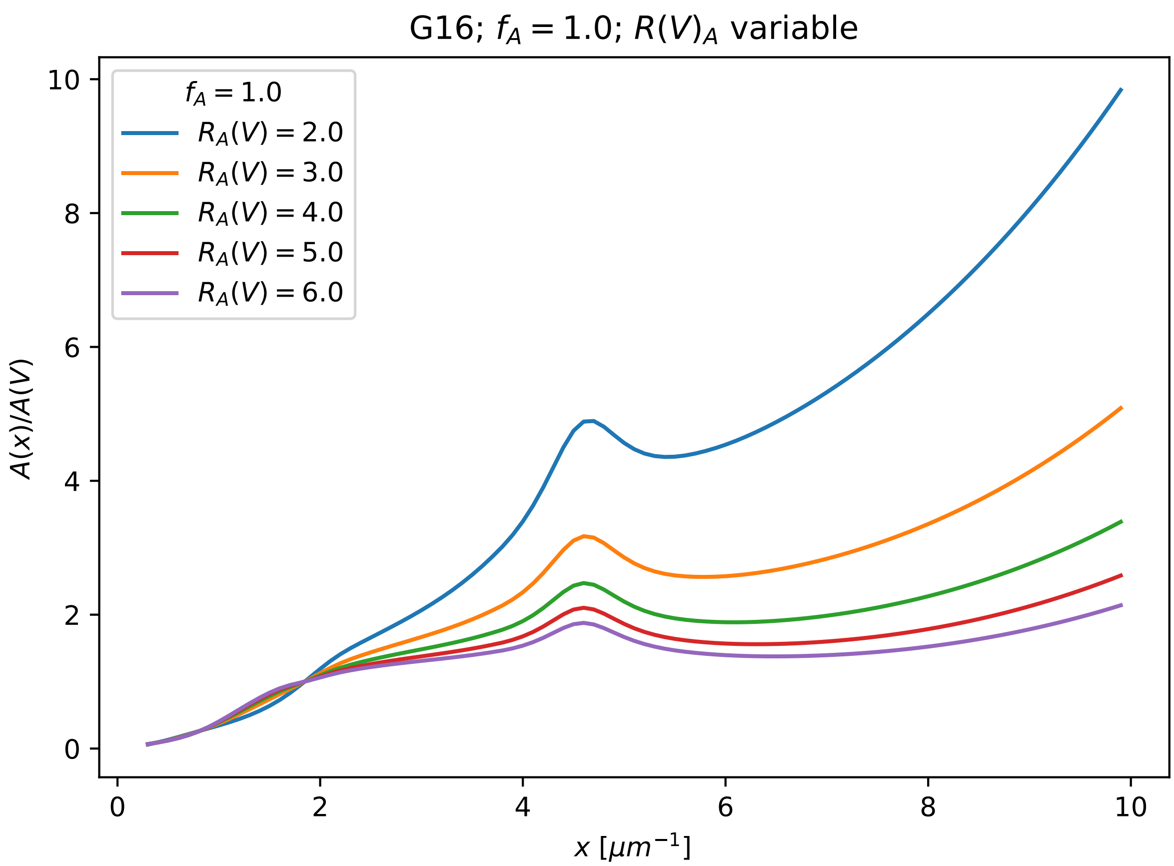
1.3 Shape fitting models

These models are used to fit the detailed shape of dust extinction curves. The FM90 (Fitzpatrick & Mass 1990) model uses 6 parameters to fit the shape of the ultraviolet extinction. The P92 (Pei 1992) uses 19 parameters to fit the shape of the X-ray to far-infrared extinction.

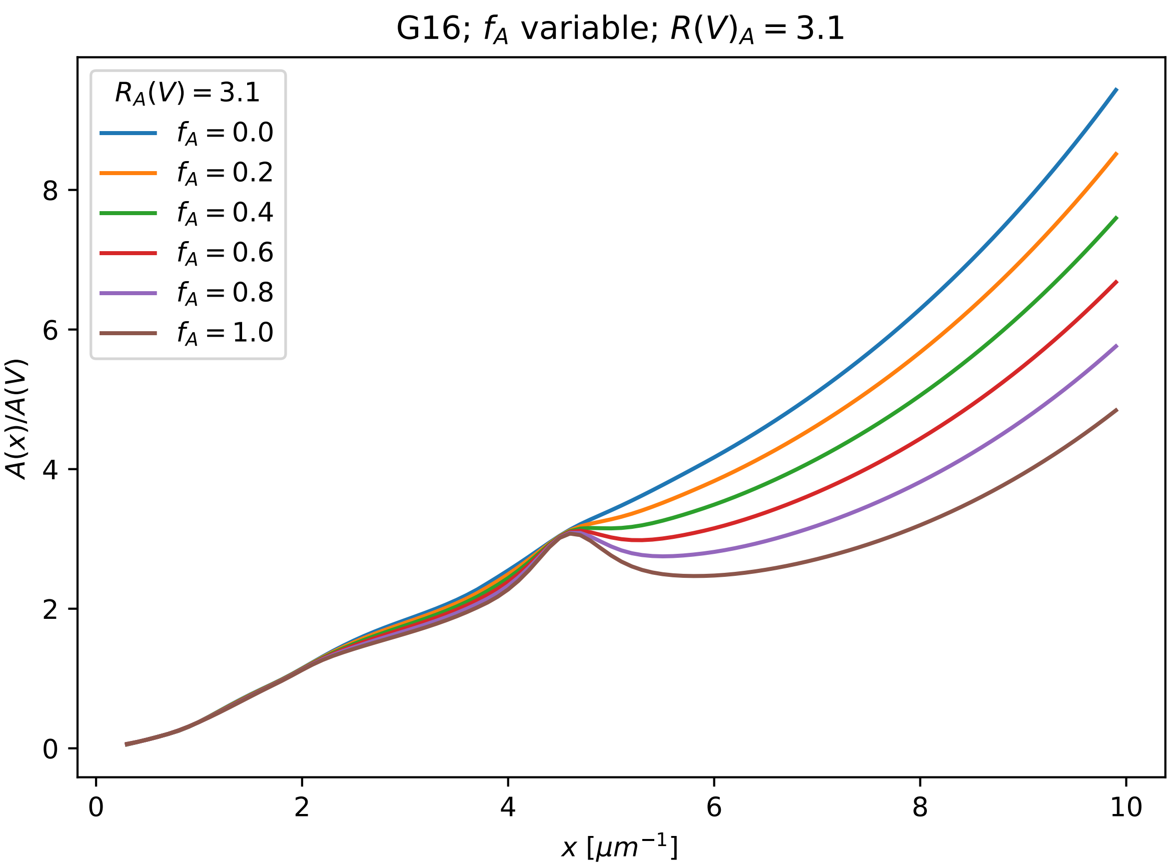

MW Average (F99 w/ $R(V) = 3.1$)
G03 SMCBar
G03 LMCAvg
G03 LMC2
$A(x)/A(V)$
$x\ [\mu m^{-1}]$
0
2
4
6
8
10

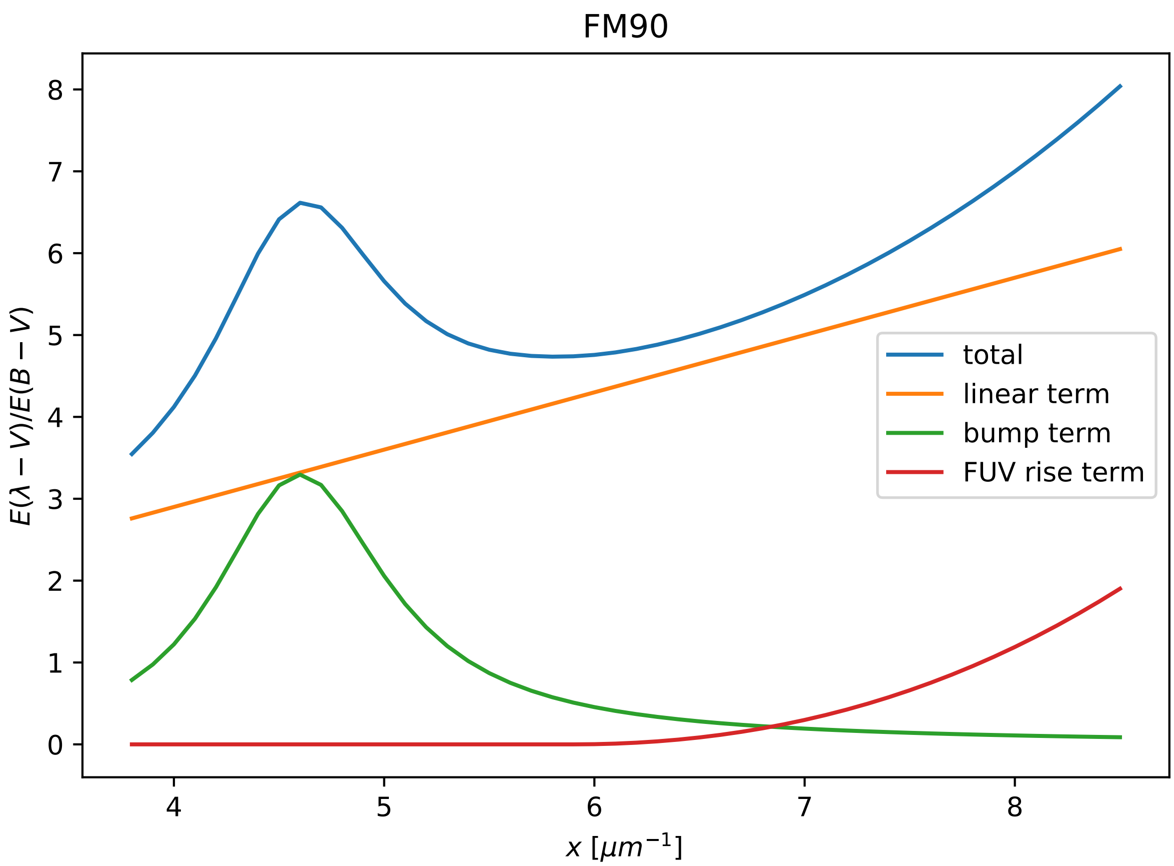

CCM89
R(V) = 2.0
R(V) = 3.0
R(V) = 4.0
R(V) = 5.0
R(V) = 6.0
A(x)/A(V)
x [μm⁻¹]
0
2
4
6
8
2
4
6
8
10

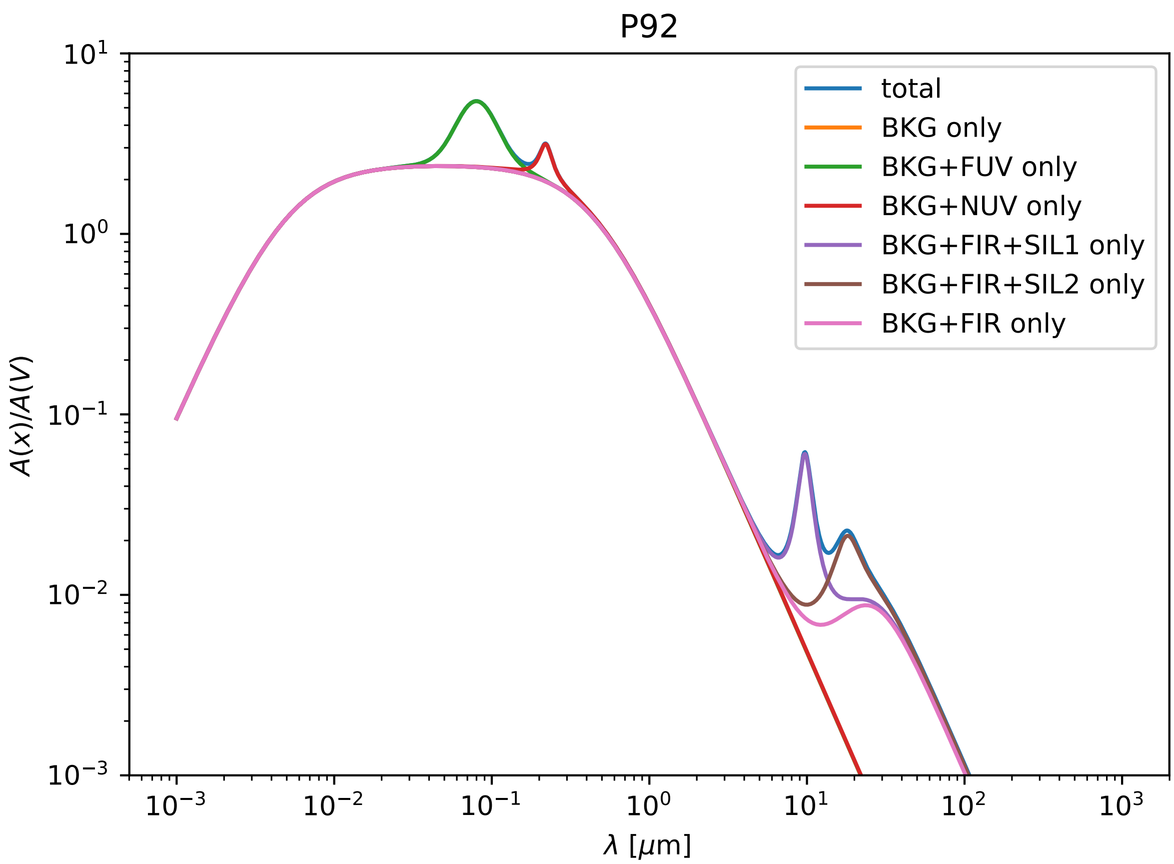

F99
R(V) = 2.0
R(V) = 3.0
R(V) = 4.0
R(V) = 5.0
R(V) = 6.0
A(x)/A(V)
x [μm⁻¹]
0
2
4
6
8
10



G16; $f_A = 1.0$; $R(V)_A$ variable
$f_A = 1.0$
$R_A(V) = 2.0$
$R_A(V) = 3.0$
$R_A(V) = 4.0$
$R_A(V) = 5.0$
$R_A(V) = 6.0$
$A(x)/A(V)$
x [μm^{-1}]
0
2
4
6
8
10


G16; f_A variable; $R(V)_A = 3.1$
$R_A(V) = 3.1$
$f_A = 0.0$
$f_A = 0.2$
$f_A = 0.4$
$f_A = 0.6$
$f_A = 0.8$
$f_A = 1.0$
$A(x)/A(V)$
x [μm^{-1}]
0
2
4
6
8
10


FM90
8
7
6
5
4
3
2
1
0
4
5
6
7
8
$E(\lambda - V)/E(B - V)$
$x\ [\mu m^{-1}]$
total
linear term
bump term
FUV rise term


P92
total
BKG only
BKG+FUV only
BKG+NUV only
BKG+FIR+SIL1 only
BKG+FIR+SIL2 only
BKG+FIR only
A(x)/A(V)
λ [μm]

CHAPTER 2

Extinguish or Unextinguish Data

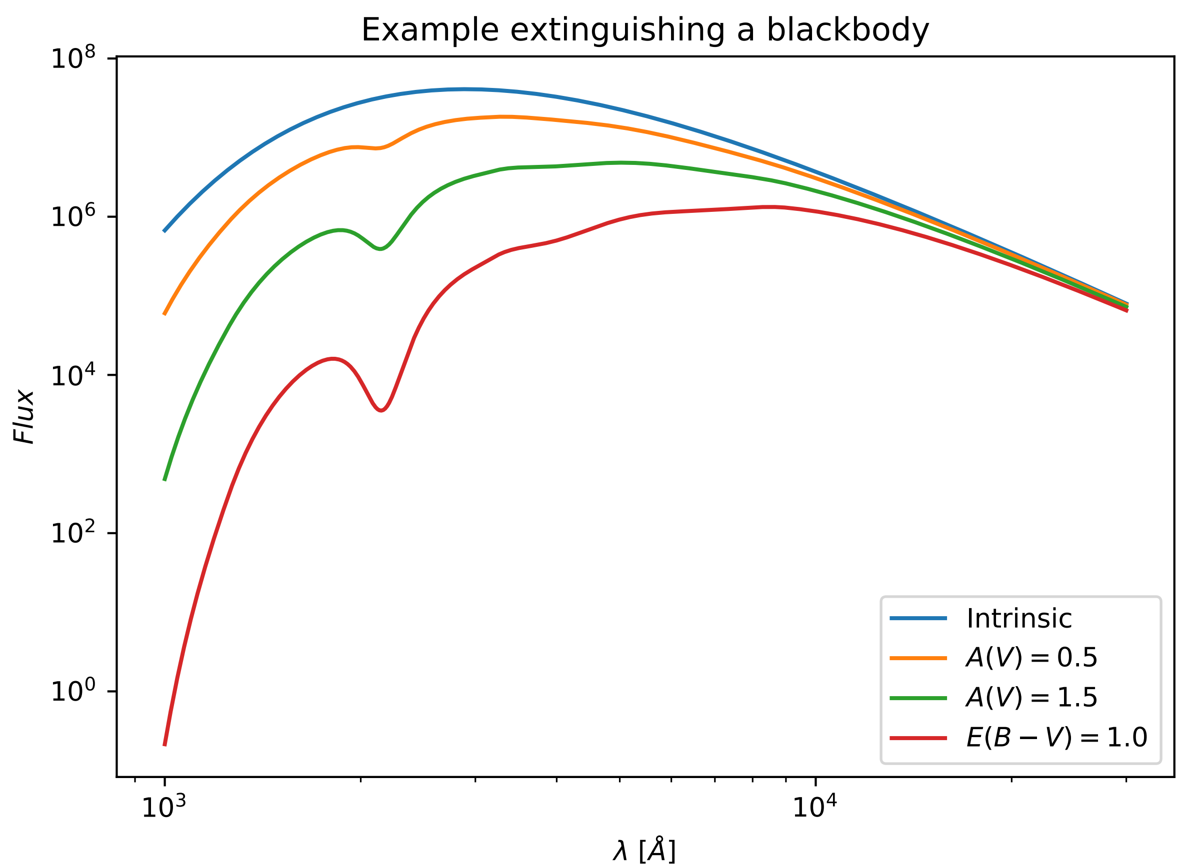
Two of the three flavors of models include a function to calculate the factor to multiple (extinguish) or divide (unextinguish) a spectrum by to add or remove the effects of dust, respectively.

Extinguish is also often called reddening. Extinguishing a spectrum often reddens the flux, but sometimes 'bluens' the flux (e.g, on the short wavelength side of the 2175 A bump). So extinguish is the more generic term.

2.1 Example: Extinguish a Blackbody

```
import matplotlib.pyplot as plt
import numpy as np

import astropy.units as u
from astropy.modeling.blackbody import blackbody_lambda

from dust_extinction.dust_extinction import CCM89

# generate wavelengths between 0.1 and 3 microns
#    within the valid range for the CCM89 R(V) dependent relationship
lam = np.logspace(np.log10(0.1), np.log10(3.0), num=1000)

# setup the inputs for the blackbody function
wavelengths = lam*1e4*u.AA
temperature = 10000*u.K

# get the blackbody flux
flux = blackbody_lambda(wavelengths, temperature)

# initialize the model
ext = CCM89(Rv=3.1)

# get the extinguished blackbody flux for different amounts of dust
flux_ext_av05 = flux*ext.extinguish(wavelengths, Av=0.5)
flux_ext_av15 = flux*ext.extinguish(wavelengths, Av=1.5)
```

```
flux_ext_ebv10 = flux*ext.extinguish(wavelengths, Ebv=1.0)

# plot the intrinsic and extinguished fluxes
fig, ax = plt.subplots()

ax.plot(wavelengths, flux, label='Intrinsic')
ax.plot(wavelengths, flux_ext_av05, label='$A(V) = 0.5$')
ax.plot(wavelengths, flux_ext_av15, label='$A(V) = 1.5$')
ax.plot(wavelengths, flux_ext_ebv10, label='$E(B-V) = 1.0$')

ax.set_xlabel('$\lambda$ [$\AA$]')
ax.set_ylabel('$Flux$')

ax.set_xscale('log')
ax.set_yscale('log')

ax.set_title('Example extinguishing a blackbody')

ax.legend(loc='best')
plt.tight_layout()
plt.show()
```



CHAPTER 3

Fit Extinction Curves

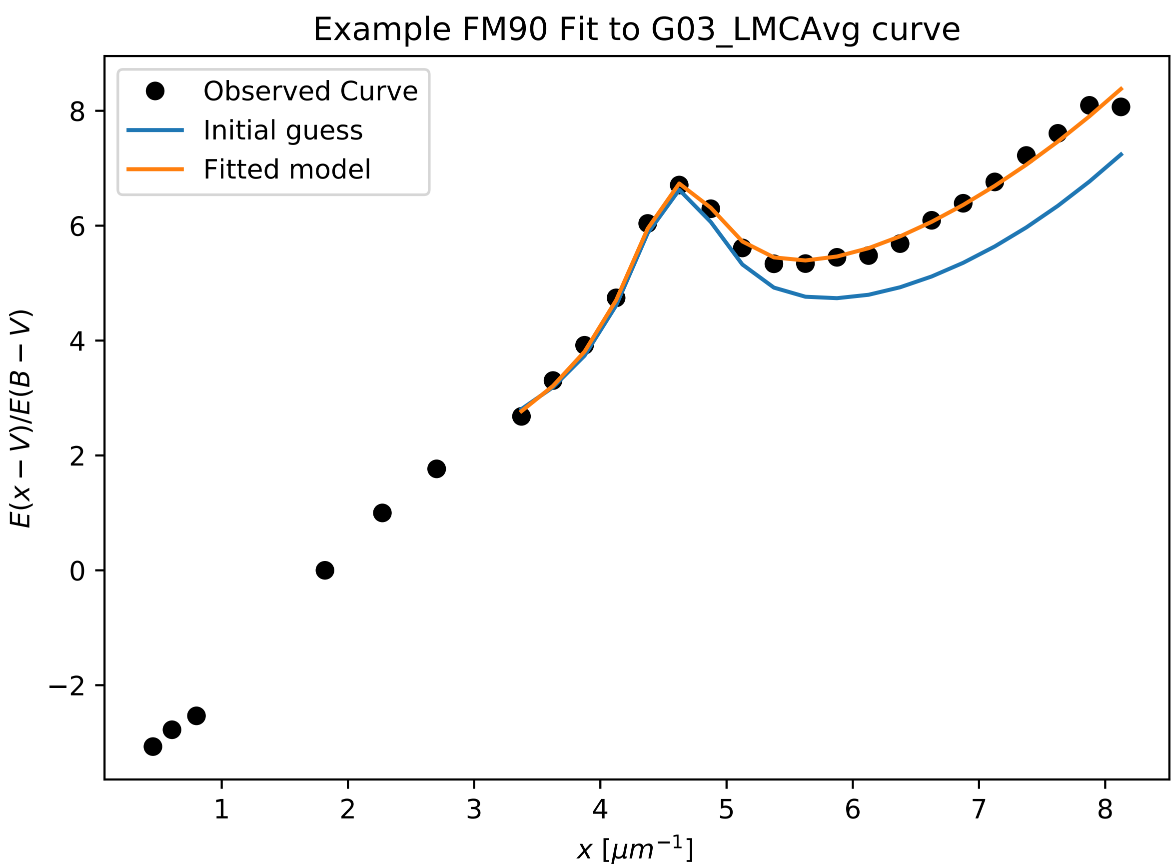
The `dust_extinction` package is built on the astropy.modeling package. Fitting is done in the standard way for this package where the model is initialized with a starting point (either the default or user input), the fitter is choosen, and the fit performed.

3.1 Example: FM90 Fit

In this example, the FM90 model is used to fit the observed average extinction curve for the LMC outside of the LMC2 supershell region (G03_LMCAvg `dust_extinction` model).

```
import matplotlib.pyplot as plt
import numpy as np

from astropy.modeling.fitting import LevMarLSQFitter

from dust_extinction.dust_extinction import G03_LMCAvg, FM90

# get an observed extinction curve to fit
g03_model = G03_LMCAvg()

x = g03_model.obsdata_x
# convert to E(x-V)/E(B0V)
y = (g03_model.obsdata_axav - 1.0)*g03_model.Rv
# only fit the UV portion (FM90 only valid in UV)
gindxs, = np.where(x > 3.125)

# initialize the model
fm90_init = FM90()

# pick the fitter
fit = LevMarLSQFitter()

# fit the data to the FM90 model using the fitter
```

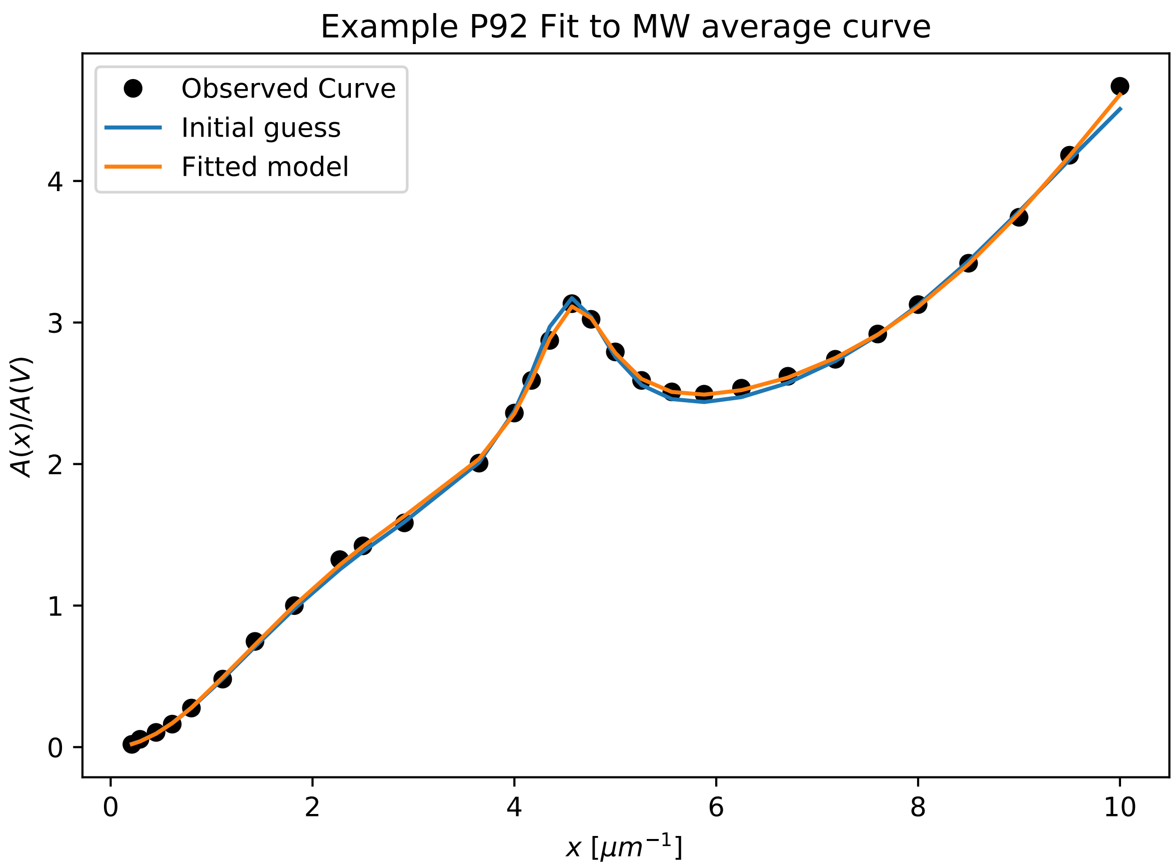
```
#   use the initialized model as the starting point
g03_fit = fit(fm90_init, x[gindxs], y[gindxs])

# plot the observed data, initial guess, and final fit
fig, ax = plt.subplots()

ax.plot(x, y, 'ko', label='Observed Curve')
ax.plot(x[gindxs], fm90_init(x[gindxs]), label='Initial guess')
ax.plot(x[gindxs], g03_fit(x[gindxs]), label='Fitted model')

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$E(x-V)/E(B-V)$')

ax.set_title('Example FM90 Fit to G03_LMCAvg curve')

ax.legend(loc='best')
plt.tight_layout()
plt.show()
```



3.2 Example: P92 Fit

In this example, the P92 model is used to fit the observed average extinction curve for the MW as tabulted by Pei (1992).

```
import matplotlib.pyplot as plt
import numpy as np

from astropy.modeling.fitting import LevMarLSQFitter

from dust_extinction.dust_extinction import P92

# Milky Way observed extinction as tabulated by Pei (1992)
MW_x = [0.21, 0.29, 0.45, 0.61, 0.80, 1.11, 1.43, 1.82,
        2.27, 2.50, 2.91, 3.65, 4.00, 4.17, 4.35, 4.57, 4.76,
        5.00, 5.26, 5.56, 5.88, 6.25, 6.71, 7.18, 7.60,
        8.00, 8.50, 9.00, 9.50, 10.00]
MW_x = np.array(MW_x)
MW_exvebv = [-3.02, -2.91, -2.76, -2.58, -2.23, -1.60, -0.78, 0.00,
             1.00, 1.30, 1.80, 3.10, 4.19, 4.90, 5.77, 6.57, 6.23,
             5.52, 4.90, 4.65, 4.60, 4.73, 4.99, 5.36, 5.91,
             6.55, 7.45, 8.45, 9.80, 11.30]
MW_exvebv = np.array(MW_exvebv)
Rv = 3.08
MW_axav = MW_exvebv/Rv + 1.0

# get an observed extinction curve to fit
x = MW_x
y = MW_axav

# initialize the model
p92_init = P92()

# pick the fitter
fit = LevMarLSQFitter()

# fit the data to the P92 model using the fitter
#   use the initialized model as the starting point
#   accuracy set to avoid warning the fit may have failed
p92_fit = fit(p92_init, x, y, acc=1e-3)

# plot the observed data, initial guess, and final fit
fig, ax = plt.subplots()

ax.plot(x, y, 'ko', label='Observed Curve')
ax.plot(x, p92_init(x), label='Initial guess')
ax.plot(x, p92_fit(x), label='Fitted model')

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$A(x)/A(V)$')

ax.set_title('Example P92 Fit to MW average curve')

ax.legend(loc='best')
plt.tight_layout()
plt.show()
```


Example P92 Fit to MW average curve
Observed Curve
Initial guess
Fitted model
$A(x)/A(V)$
x [μm^{-1}]
0
1
2
3
4
0
2
4
6
8
10

Part III

Reporting Issues

If you have found a bug in `dust_extinction` please report it by creating a new issue on the `dust_extinction` GitHub issue tracker.

Please include an example that demonstrates the issue sufficiently so that the developers can reproduce and fix the problem. You may also be asked to provide information about your operating system and a full Python stack trace. The developers will walk you through obtaining a stack trace if it is necessary.

Part IV

Contributing

Like the Astropy project, `dust_extinction` is made both by and for its users. We accept contributions at all levels, spanning the gamut from fixing a typo in the documentation to developing a major new feature. We welcome contributors who will abide by the Python Software Foundation Code of Conduct.

`dust_extinction` follows the same workflow and coding guidelines as Astropy. The following pages will help you get started with contributing fixes, code, or documentation (no git or GitHub experience necessary):

- How to make a code contribution
- Coding Guidelines
- Try the development version
- Developer Documentation

For the complete list of contributors please see the dust_extinction contributors page on Github.

Part V

Reference API

CHAPTER 4

dust_extinction.dust_extinction Module

4.1 Classes

BaseExtModel(*args, **kwargs)	Base Extinction Model.
BaseExtRvModel(*args, **kwargs)	Base Extinction R(V)-dependent Model.
BaseExtAve(*args, **kwargs)	Base Extinction Average.
CCM89([Rv])	CCM89 extinction model calculation
FM90([C1, C2, C3, C4, xo, gamma])	FM90 extinction model calculation
P92([BKG_amp, BKG_lambda, BKG_b, BKG_n, . . .])	P92 extinction model calculation
F99([Rv])	F99 extinction model calculation
G03_SMCBar(*args, **kwargs)	G03 SMCBar Average Extinction Curve
G03_LMCAvg(*args, **kwargs)	G03 LMCAvg Average Extinction Curve
G03_LMC2(*args, **kwargs)	G03 LMC2 Average Extinction Curve
G16([RvA, fA])	G16 extinction model calculation

4.1.1 BaseExtModel

class dust_extinction.dust_extinction.**BaseExtModel**(**args*, ***kwargs*)

Bases: astropy.modeling.Model

Base Extinction Model. Do not use.

Attributes Summary

inputs
outputs

Methods Summary

`__call__`(x[, model_set_axis, . . .])	Evaluate this model using the given input(s) and the parameter values that were specified when the model was instantiated.
`extinguish`(x[, Av, Ebv])	Calculate the extinction as a fraction

Attributes Documentation

inputs = ('x',)

outputs = ('axav',)

Methods Documentation

__call__(*x*, *model_set_axis=None*, *with_bounding_box=False*, *fill_value=nan*, *equivalencies=None*)
Evaluate this model using the given input(s) and the parameter values that were specified when the model was instantiated.

extinguish(*x*, *Av=None*, *Ebv=None*)
Calculate the extinction as a fraction

Parameters

x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Av: float

A(V) value of dust column Av or Ebv must be set

Ebv: float

E(B-V) value of dust column Av or Ebv must be set

Returns

frac_ext: np array (float)

fractional extinction as a function of x

4.1.2 BaseExtRvModel

class dust_extinction.dust_extinction.**BaseExtRvModel**(**args*, ***kwargs*)
Bases: `dust_extinction.dust_extinction.BaseExtModel`

Base Extinction R(V)-dependent Model. Do not use.

Attributes Summary

Rv	R(V) = A(V)/E(B-V) = total-to-selective extinction
param_names	

Attributes Documentation

Rv
R(V) = A(V)/E(B-V) = total-to-selective extinction

param_names = ('Rv',)

4.1.3 BaseExtAve

class dust_extinction.dust_extinction.**BaseExtAve**(**args*, ***kwargs*)
Bases: astropy.modeling.Model

Base Extinction Average. Do not use.

Attributes Summary

inputs	
outputs	

Methods Summary

__call__(x[, model_set_axis, . . .])	Evaluate this model using the given input(s) and the parameter values that were specified when the model was instantiated.
extinguish(x[, Av, Ebv])	Calculate the extinction as a fraction

Attributes Documentation

inputs = ('x',)

outputs = ('axav',)

Methods Documentation

__call__(*x*, *model_set_axis=None*, *with_bounding_box=False*, *fill_value=nan*, *equivalencies=None*)
Evaluate this model using the given input(s) and the parameter values that were specified when the model was instantiated.

extinguish(*x*, *Av=None*, *Ebv=None*)
Calculate the extinction as a fraction

Parameters
x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Av: float

A(V) value of dust column Av or Ebv must be set

Ebv: float

E(B-V) value of dust column Av or Ebv must be set

Returns

frac_ext: np array (float)

fractional extinction as a function of x

4.1.4 CCM89

class dust_extinction.dust_extinction.**CCM89**(*Rv=3.1*, ***kwargs*)

Bases: dust_extinction.dust_extinction.BaseExtRvModel

CCM89 extinction model calculation

Parameters

Rv: float

R(V) = A(V)/E(B-V) = total-to-selective extinction

Raises

InputParameterError

Input Rv values outside of defined range

Notes

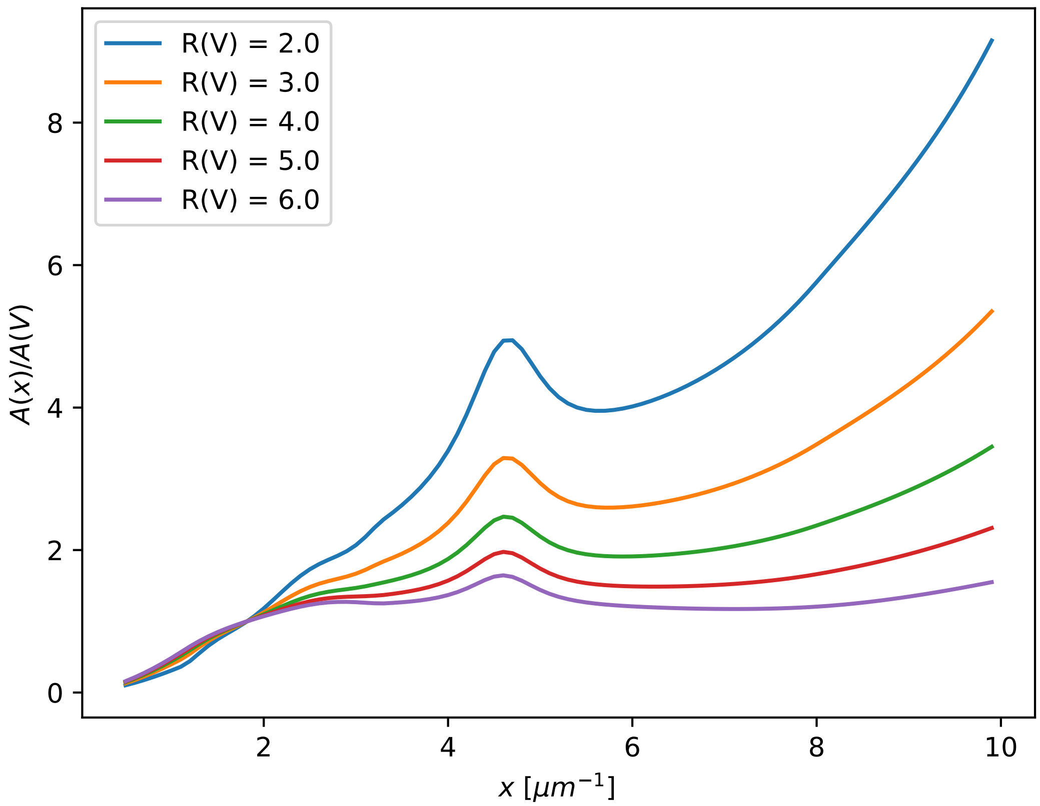
CCM89 Milky Way R(V) dependent extinction model

From Cardelli, Clayton, and Mathis (1989, ApJ, 345, 245)

Example showing CCM89 curves for a range of R(V) values.

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import CCM89

fig, ax = plt.subplots()

# generate the curves and plot them
x = np.arange(0.5,10.0,0.1)/u.micron

Rvs = ['2.0','3.0','4.0','5.0','6.0']
for cur_Rv in Rvs:
   ext_model = CCM89(Rv=cur_Rv)
   ax.plot(x,ext_model(x),label='R(V) = ' + str(cur_Rv))

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$A(x)/A(V)$')
```

```
ax.legend(loc='best')
plt.show()
```



Attributes Summary

Rv_range	
param_names	
x_range	

Methods Summary

evaluate(in_x, Rv)	CCM89 function

Attributes Documentation

Rv_range = [2.0, 6.0]

param_names = ('Rv',)

x_range = [0.3, 10.0]

Methods Documentation

static evaluate(*in_x*, *Rv*)

CCM89 function

Parameters

in_x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Returns

axav: np array (float)

A(x)/A(V) extinction curve [mag]

Raises

ValueError

Input x values outside of defined range

4.1.5 FM90

class dust_extinction.dust_extinction.**FM90**(*C1=0.1*, *C2=0.7*, *C3=3.23*, *C4=0.41*, *xo=4.6*, *gamma=0.99*, ***kwargs*)

Bases: astropy.modeling.Fittable1DModel

FM90 extinction model calculation

Parameters

C1: float

y-intercept of linear term

C2: float

slope of liner term

C3: float

amplitude of "2175 A" bump

C4: float

amplitude of FUV rise

xo: float

centroid of "2175 A" bump

gamma: float

width of "2175 A" bump

Notes

FM90 extinction model

From Fitzpatrick & Massa (1990)

Only applicable at UV wavelengths

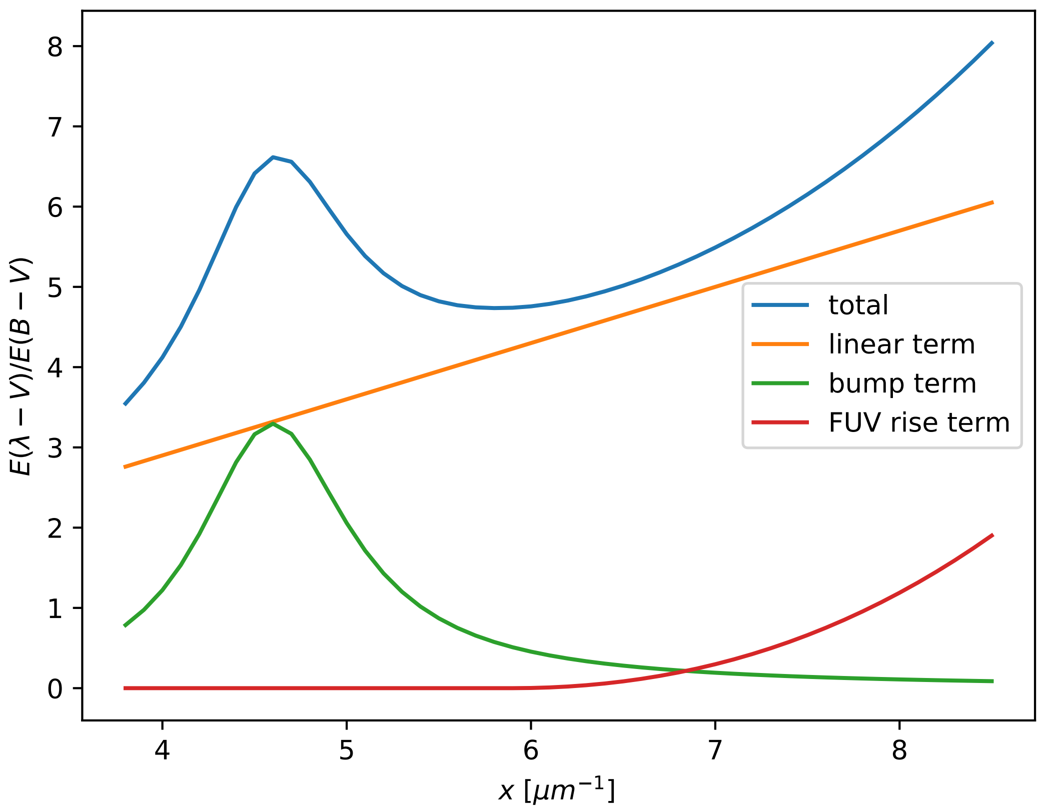
Example showing a FM90 curve with components identified.

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import FM90

fig, ax = plt.subplots()

# generate the curves and plot them
x = np.arange(3.8,8.6,0.1)/u.micron

ext_model = FM90()
ax.plot(x,ext_model(x),label='total')

ext_model = FM90(C3=0.0, C4=0.0)
ax.plot(x,ext_model(x),label='linear term')

ext_model = FM90(C1=0.0, C2=0.0, C4=0.0)
ax.plot(x,ext_model(x),label='bump term')

ext_model = FM90(C1=0.0, C2=0.0, C3=0.0)
ax.plot(x,ext_model(x),label='FUV rise term')

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$E(\lambda - V)/E(B - V)$')

ax.legend(loc='best')
plt.show()
```

Attributes Summary

C1	linear term: y-intercept
C2	linear term: slope
C3	bump: amplitude
C4	FUV rise: amplitude
gamma	bump: width
inputs	
outputs	
param_names	
x_range	
xo	bump: centroid

Methods Summary


8
7
6
5
4
3
2
1
0
4
5
6
7
8
$E(\lambda - V)/E(B - V)$
$x\ [\mu m^{-1}]$
total
linear term
bump term
FUV rise term

`__call__`(x[, model_set_axis, . . .])	Evaluate this model using the given input(s) and the parameter values that were specified when the model was instantiated.
`evaluate`(in_x, C1, C2, C3, C4, xo, gamma)	FM90 function
`fit_deriv`(in_x, C1, C2, C3, C4, xo, gamma)	Derivatives of the FM90 function with respect to the parameters

Attributes Documentation

C1
linear term: y-intercept

C2
linear term: slope

C3
bump: amplitude

C4
FUV rise: amplitude

gamma
bump: width

inputs = ('x',)

outputs = ('exvebv',)

param_names = ('C1', 'C2', 'C3', 'C4', 'xo', 'gamma')

x_range = [3.125, 10.964912280701753]

xo
bump: centroid

Methods Documentation

__call__(*x*, *model_set_axis=None*, *with_bounding_box=False*, *fill_value=nan*, *equivalencies=None*)
Evaluate this model using the given input(s) and the parameter values that were specified when the model was instantiated.

static evaluate(*in_x*, *C1*, *C2*, *C3*, *C4*, *xo*, *gamma*)
FM90 function

Parameters
in_x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Returns
exvebv: np array (float)

E(x-V)/E(B-V) extinction curve [mag]

Raises

ValueError

Input x values outside of defined range

`static fit_deriv`(*in_x*, *C1*, *C2*, *C3*, *C4*, *xo*, *gamma*)

Derivatives of the FM90 function with respect to the parameters

4.1.6 P92

`class dust_extinction.dust_extinction.P92`(*BKG_amp=218.57142857142858*, *BKG_lambda=0.047*, *BKG_b=90.0*, *BKG_n=2.0*, *FUV_amp=18.545454545454547*, *FUV_lambda=0.08*, *FUV_b=4.0*, *FUV_n=6.5*, *NUV_amp=0.05961038961038961*, *NUV_lambda=0.22*, *NUV_b=-1.95*, *NUV_n=2.0*, *SIL1_amp=0.0026493506493506496*, *SIL1_lambda=9.7*, *SIL1_b=-1.95*, *SIL1_n=2.0*, *SIL2_amp=0.0026493506493506496*, *SIL2_lambda=18.0*, *SIL2_b=-1.8*, *SIL2_n=2.0*, *FIR_amp=0.015896103896103898*, *FIR_lambda=25.0*, *FIR_b=0.0*, *FIR_n=2.0*, ***kwargs*)

Bases: `astropy.modeling.Fittable1DModel`

P92 extinction model calculation

Parameters

BKG_amp : float

background term amplitude

BKG_lambda : float

background term central wavelength

BKG_b : float

background term b coefficient

BKG_n : float

background term n coefficient [FIXED at n = 2]

FUV_amp : float

far-ultraviolet term amplitude

FUV_lambda : float

far-ultraviolet term central wavelength

FUV_b : float

far-ultraviolet term b coefficent

FUV_n : float

far-ultraviolet term n coefficient

NUV_amp : float

near-ultraviolet (2175 A) term amplitude

NUV_lambda : float

near-ultraviolet (2175 A) term central wavelength

NUV_b : float

near-ultraviolet (2175 A) term b coefficent

NUV_n : float

near-ultraviolet (2175 A) term n coefficient [FIXED at n = 2]

SIL1_amp : float

1st silicate feature (~10 micron) term amplitude

SIL1_lambda : float

1st silicate feature (~10 micron) term central wavelength

SIL1_b : float

1st silicate feature (~10 micron) term b coefficent

SIL1_n : float

1st silicate feature (~10 micron) term n coefficient [FIXED at n = 2]

SIL2_amp : float

2nd silicate feature (~18 micron) term amplitude

SIL2_lambda : float

2nd silicate feature (~18 micron) term central wavelength

SIL2_b : float

2nd silicate feature (~18 micron) term b coefficient

SIL2_n : float

2nd silicate feature (~18 micron) term n coefficient [FIXED at n = 2]

FIR_amp : float

far-infrared term amplitude

FIR_lambda : float

far-infrared term central wavelength

FIR_b : float

far-infrared term b coefficent

FIR_n : float

far-infrared term n coefficient [FIXED at n = 2]

Notes

P92 extinction model

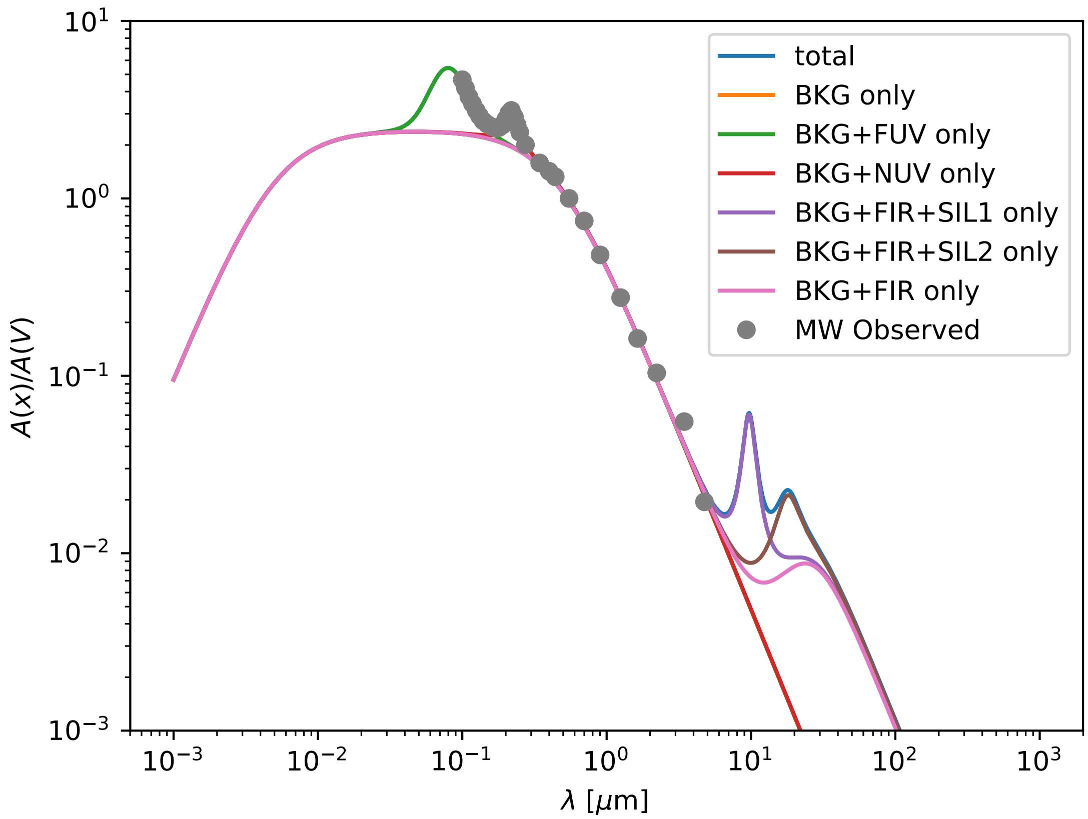
From Pei (1992)

Applicable from the extreme UV to far-IR

Example showing a P92 curve with components identified.

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import P92

fig, ax = plt.subplots()

# generate the curves and plot them
lam = np.logspace(-3.0, 3.0, num=1000)
x = (1.0/lam)/u.micron

ext_model = P92()
ax.plot(1/x,ext_model(x),label='total')

ext_model = P92(FUV_amp=0., NUV_amp=0.0,
                SIL1_amp=0.0, SIL2_amp=0.0, FIR_amp=0.0)
ax.plot(1./x,ext_model(x),label='BKG only')

ext_model = P92(NUV_amp=0.0,
                SIL1_amp=0.0, SIL2_amp=0.0, FIR_amp=0.0)
ax.plot(1./x,ext_model(x),label='BKG+FUV only')

ext_model = P92(FUV_amp=0.,
                SIL1_amp=0.0, SIL2_amp=0.0, FIR_amp=0.0)
ax.plot(1./x,ext_model(x),label='BKG+NUV only')

ext_model = P92(FUV_amp=0., NUV_amp=0.0,
                SIL2_amp=0.0)
ax.plot(1./x,ext_model(x),label='BKG+FIR+SIL1 only')

ext_model = P92(FUV_amp=0., NUV_amp=0.0,
                SIL1_amp=0.0)
ax.plot(1./x,ext_model(x),label='BKG+FIR+SIL2 only')

ext_model = P92(FUV_amp=0., NUV_amp=0.0,
                SIL1_amp=0.0, SIL2_amp=0.0)
ax.plot(1./x,ext_model(x),label='BKG+FIR only')

# Milky Way observed extinction as tabulated by Pei (1992)
MW_x = [0.21, 0.29, 0.45, 0.61, 0.80, 1.11, 1.43, 1.82,
        2.27, 2.50, 2.91, 3.65, 4.00, 4.17, 4.35, 4.57, 4.76,
        5.00, 5.26, 5.56, 5.88, 6.25, 6.71, 7.18, 7.60,
        8.00, 8.50, 9.00, 9.50, 10.00]
MW_x = np.array(MW_x)
MW_exvebv = [-3.02, -2.91, -2.76, -2.58, -2.23, -1.60, -0.78, 0.00,
             1.00, 1.30, 1.80, 3.10, 4.19, 4.90, 5.77, 6.57, 6.23,
             5.52, 4.90, 4.65, 4.60, 4.73, 4.99, 5.36, 5.91,
             6.55, 7.45, 8.45, 9.80, 11.30]
MW_exvebv = np.array(MW_exvebv)
Rv = 3.08
MW_axav = MW_exvebv/Rv + 1.0
ax.plot(1./MW_x, MW_axav, 'o', label='MW Observed')

ax.set_xscale('log')
ax.set_yscale('log')

ax.set_ylim(1e-3,10.)
```

```
ax.set_xlabel('$\lambda$ [$\mu$m]')
ax.set_ylabel('$A(x)/A(V)$')

ax.legend(loc='best')
plt.show()
```



Attributes Summary

AbAv	
BKG_amp	BKG term: amplitude
BKG_b	BKG term: b coefficient
BKG_lambda	BKG term: center wavelength
BKG_n	BKG term: n coefficient
FIR_amp	FIR term: amplitude
FIR_b	FIR term: b coefficient
FIR_lambda	FIR term: center wavelength
FIR_n	FIR term: n coefficient
FUV_amp	FUV term: amplitude
FUV_b	FUV term: b coefficient

Continued on next page

Table 4.11 – continued from previous page

FUV_lambda	FUV term: center wavelength
FUV_n	FUV term: n coefficient
NUV_amp	NUV term: amplitude
NUV_b	NUV term: b coefficient
NUV_lambda	NUV term: center wavelength
NUV_n	NUV term: n coefficient
SIL1_amp	SIL1 term: amplitude
SIL1_b	SIL1 term: b coefficient
SIL1_lambda	SIL1 term: center wavelength
SIL1_n	SIL1 term: n coefficient
SIL2_amp	SIL2 term: amplitude
SIL2_b	SIL2 term: b coefficient
SIL2_lambda	SIL2 term: center wavelength
SIL2_n	SIL2 term: n coefficient
fit_deriv	
inputs	
outputs	
param_names	
x_range	

Methods Summary

`__call__`(x[, model_set_axis, . . .])	Evaluate this model using the given input(s) and the parameter values that were specified when the model was instantiated.
`evaluate`(in_x, BKG_amp, BKG_lambda, BKG_b, . . .)	P92 function

Attributes Documentation

AbAv = 1.3246753246753247

BKG_amp
BKG term: amplitude

BKG_b
BKG term: b coefficient

BKG_lambda
BKG term: center wavelength

BKG_n
BKG term: n coefficient

FIR_amp
FIR term: amplitude

FIR_b
FIR term: b coefficient

FIR_lambda
FIR term: center wavelength

FIR_n
FIR term: n coefficient

FUV_amp
FUV term: amplitude

FUV_b
FUV term: b coefficient

FUV_lambda
FUV term: center wavelength

FUV_n
FUV term: n coefficient

NUV_amp
NUV term: amplitude

NUV_b
NUV term: b coefficient

NUV_lambda
NUV term: center wavelength

NUV_n
NUV term: n coefficient

SIL1_amp
SIL1 term: amplitude

SIL1_b
SIL1 term: b coefficient

SIL1_lambda
SIL1 term: center wavelength

SIL1_n
SIL1 term: n coefficient

SIL2_amp
SIL2 term: amplitude

SIL2_b
SIL2 term: b coefficient

SIL2_lambda
SIL2 term: center wavelength

SIL2_n
SIL2 term: n coefficient

fit_deriv = None

inputs = ('x',)

outputs = ('axav',)

param_names = ('BKG_amp', 'BKG_lambda', 'BKG_b', 'BKG_n', 'FUV_amp', 'FUV_lambda', 'FUV_b', 'FUV_n', 'NU

x_range = [0.001, 1000.0]

Methods Documentation

__call__(*x*, *model_set_axis=None*, *with_bounding_box=False*, *fill_value=nan*, *equivalencies=None*)
Evaluate this model using the given input(s) and the parameter values that were specified when the model was instantiated.

evaluate(*in_x*, *BKG_amp*, *BKG_lambda*, *BKG_b*, *BKG_n*, *FUV_amp*, *FUV_lambda*, *FUV_b*, *FUV_n*, *NUV_amp*, *NUV_lambda*, *NUV_b*, *NUV_n*, *SIL1_amp*, *SIL1_lambda*, *SIL1_b*, *SIL1_n*, *SIL2_amp*, *SIL2_lambda*, *SIL2_b*, *SIL2_n*, *FIR_amp*, *FIR_lambda*, *FIR_b*, *FIR_n*)
P92 function

Parameters
in_x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Returns
axav: np array (float)

A(x)/A(V) extinction curve [mag]

Raises
ValueError

Input x values outside of defined range

4.1.7 F99

class dust_extinction.dust_extinction.**F99**(*Rv=3.1*, ***kwargs*)
Bases: `dust_extinction.dust_extinction.BaseExtRvModel`

F99 extinction model calculation

Parameters
Rv: float

R(V) = A(V)/E(B-V) = total-to-selective extinction

Raises
InputParameterError

Input Rv values outside of defined range

Notes

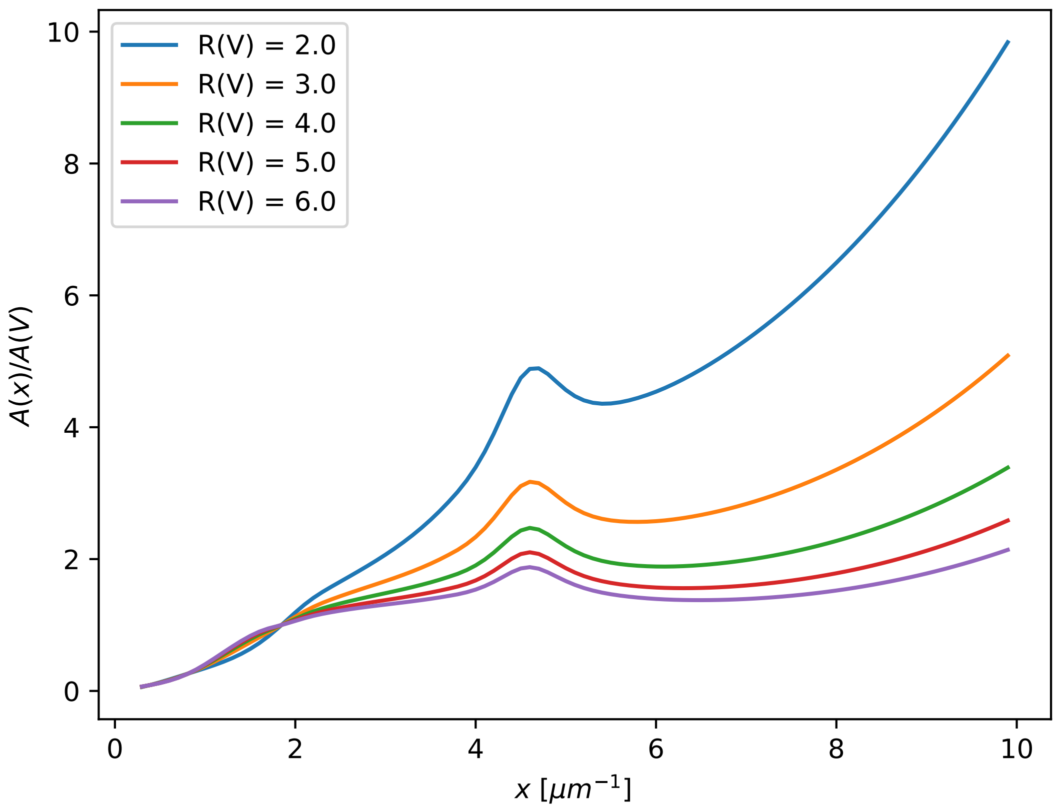
F99 Milky Way R(V) dependent extinction model

From Fitzpatrick (1999, PASP, 111, 63)

Updated for the C1 vs C2 correlation in
Fitzpatrick & Massa (2007, ApJ, 663, 320)

Example showing F99 curves for a range of R(V) values.

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import F99

fig, ax = plt.subplots()

# temp model to get the correct x range
text_model = F99()

# generate the curves and plot them
x = np.arange(text_model.x_range[0], text_model.x_range[1],0.1)/u.micron

Rvs = ['2.0','3.0','4.0','5.0','6.0']
for cur_Rv in Rvs:
   ext_model = F99(Rv=cur_Rv)
   ax.plot(x,ext_model(x),label='R(V) = ' + str(cur_Rv))

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$A(x)/A(V)$')

ax.legend(loc='best')
plt.show()
```



Attributes Summary

Rv_range	
param_names	
x_range	

Methods Summary

evaluate(in_x, Rv)	F99 function

Attributes Documentation

Rv_range = [2.0, 6.0]

param_names = ('Rv',)

x_range = [0.3, 10.0]

Methods Documentation

evaluate(*in_x*, *Rv*)

F99 function

Parameters

in_x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Returns

axav: np array (float)

A(x)/A(V) extinction curve [mag]

Raises

ValueError

Input x values outside of defined range

4.1.8 G03_SMCBar

class dust_extinction.dust_extinction.**G03_SMCBar**(**args*, ***kwargs*)

Bases: dust_extinction.dust_extinction.BaseExtAve

G03 SMCBar Average Extinction Curve

Parameters

None

Raises
None

Notes

SMCBar G03 average extinction curve

From Gordon et al. (2003, ApJ, 594, 279)

Example showing the average curve

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import G03_SMCBar

fig, ax = plt.subplots()

# define the extinction model
ext_model = G03_SMCBar()

# generate the curves and plot them
x = np.arange(ext_model.x_range[0], ext_model.x_range[1],0.1)/u.micron

ax.plot(x,ext_model(x),label='G03 SMCBar')
ax.plot(ext_model.obsdata_x, ext_model.obsdata_axav, 'ko',
        label='obsdata')

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$A(x)/A(V)$')

ax.legend(loc='best')
plt.show()
```

Attributes Summary

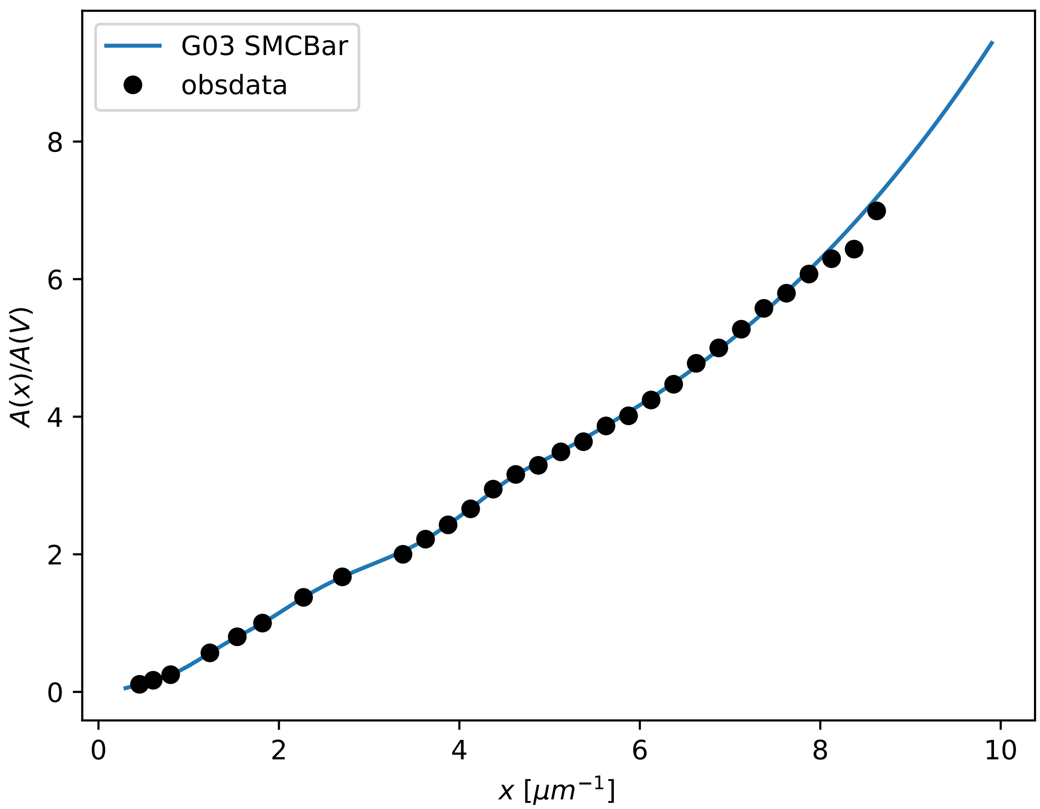
Rv
obsdata_axav
obsdata_tolerance
obsdata_x
x_range

Methods Summary

evaluate(in_x)	G03 SMCBar function

Attributes Documentation

Rv = 2.74


G03 SMCBar
obsdata
8
6
4
2
0
A(x)/A(V)
0
2
4
6
8
10
x [μm⁻¹]

`obsdata_axav = array([ 0.11 , 0.169, 0.25 , 0.567, 0.801, 1. , 1.374, 1.672, 2. , 2.22 , 2.428, 2.661, 2`

`obsdata_tolerance = 0.06`

`obsdata_x = array([ 0.455, 0.606, 0.8 , 1.235, 1.538, 1.818, 2.273, 2.703, 3.375, 3.625, 3.875, 4.125, 4`

`x_range = [0.3, 10.0]`

Methods Documentation

evaluate(*in_x*)
G03 SMCBar function

Parameters
in_x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Returns
axav: np array (float)

A(x)/A(V) extinction curve [mag]

Raises
ValueError

Input x values outside of defined range

4.1.9 G03_LMCAvg

class dust_extinction.dust_extinction.**G03_LMCAvg**(**args*, ***kwargs*)
Bases: `dust_extinction.dust_extinction.BaseExtAve`

G03 LMCAvg Average Extinction Curve

Parameters
None

Raises
None

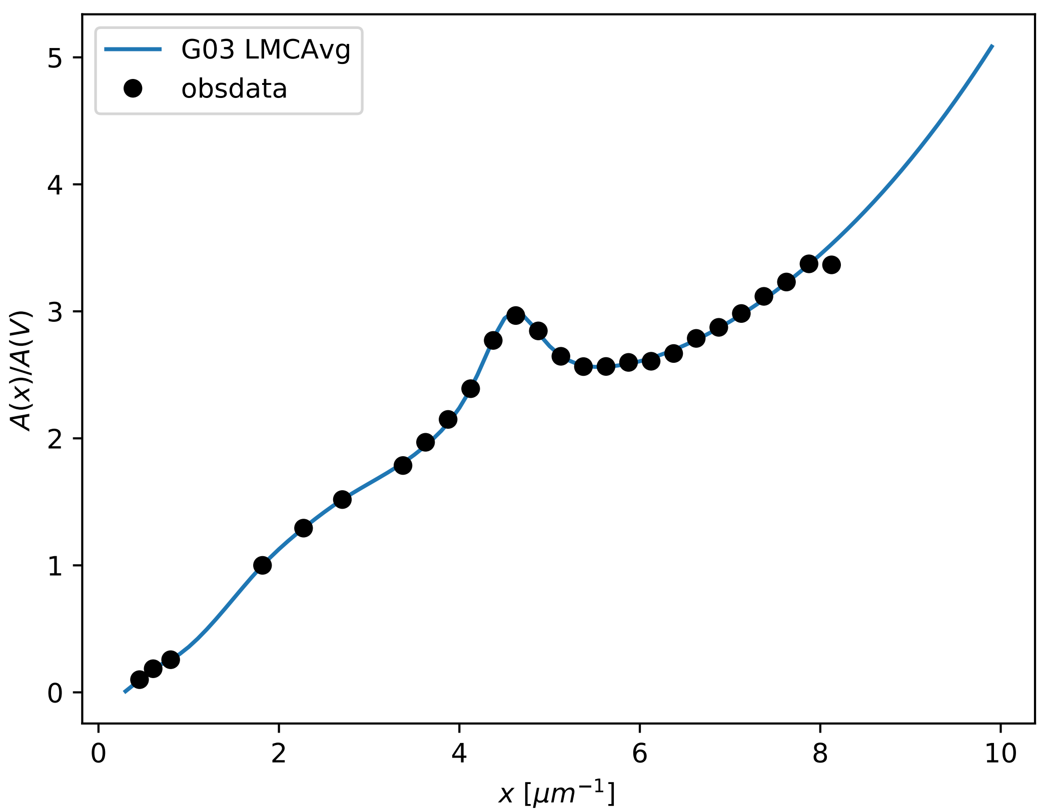
Notes

LMCAvg G03 average extinction curve

From Gordon et al. (2003, ApJ, 594, 279)

Example showing the average curve

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import G03_LMCAvg

fig, ax = plt.subplots()

# define the extinction model
ext_model = G03_LMCAvg()

# generate the curves and plot them
x = np.arange(ext_model.x_range[0], ext_model.x_range[1],0.1)/u.micron

ax.plot(x,ext_model(x),label='G03 LMCAvg')
ax.plot(ext_model.obsdata_x, ext_model.obsdata_axav, 'ko',
        label='obsdata')

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$A(x)/A(V)$')

ax.legend(loc='best')
plt.show()
```



Attributes Summary

Rv	
obsdata_axav	
obsdata_tolerance	
obsdata_x	
x_range	

Methods Summary

evaluate(in_x)	G03 LMCAvg function

Attributes Documentation

Rv = 3.41

obsdata_axav = array([0.1 , 0.186, 0.257, 1. , 1.293, 1.518, 1.786, 1.969, 2.149, 2.391, 2.771, 2.967,

obsdata_tolerance = 0.06

obsdata_x = array([0.455, 0.606, 0.8 , 1.818, 2.273, 2.703, 3.375, 3.625, 3.875, 4.125, 4.375, 4.625,

x_range = [0.3, 10.0]

Methods Documentation

evaluate(*in_x*)

G03 LMCAvg function

Parameters

in_x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Returns

axav: np array (float)

A(x)/A(V) extinction curve [mag]

Raises

ValueError

Input x values outside of defined range

4.1.10 G03_LMC2

class dust_extinction.dust_extinction.**G03_LMC2**(**args*, ***kwargs*)

Bases: dust_extinction.dust_extinction.BaseExtAve

G03 LMC2 Average Extinction Curve

Parameters
None

Raises
None

Notes

LMC2 G03 average extinction curve

From Gordon et al. (2003, ApJ, 594, 279)

Example showing the average curve

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import G03_LMC2

fig, ax = plt.subplots()

# generate the curves and plot them
x = np.arange(0.3,10.0,0.1)/u.micron

# define the extinction model
ext_model = G03_LMC2()

# generate the curves and plot them
x = np.arange(ext_model.x_range[0], ext_model.x_range[1],0.1)/u.micron

ax.plot(x,ext_model(x),label='G03 LMC2')
ax.plot(ext_model.obsdata_x, ext_model.obsdata_axav, 'ko',
        label='obsdata')

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$A(x)/A(V)$')

ax.legend(loc='best')
plt.show()
```

Attributes Summary

Rv
obsdata_axav
obsdata_tolerance
obsdata_x

Continued on next page

Table 4.19 – continued from previous page

x_range

Methods Summary

evaluate(in_x)	G03 LMC2 function

Attributes Documentation

Rv = 2.76

obsdata_axav = array([0.101, 0.15 , 0.299, 1. , 1.349, 1.665, 1.899, 2.067, 2.249, 2.447, 2.777, 2.922,

obsdata_tolerance = 0.06

obsdata_x = array([0.455, 0.606, 0.8 , 1.818, 2.273, 2.703, 3.375, 3.625, 3.875, 4.125, 4.375, 4.625, 4

x_range = [0.3, 10.0]

Methods Documentation

evaluate(*in_x*)

G03 LMC2 function

Parameters

in_x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

Returns

axav: np array (float)

A(x)/A(V) extinction curve [mag]

Raises

ValueError

Input x values outside of defined range

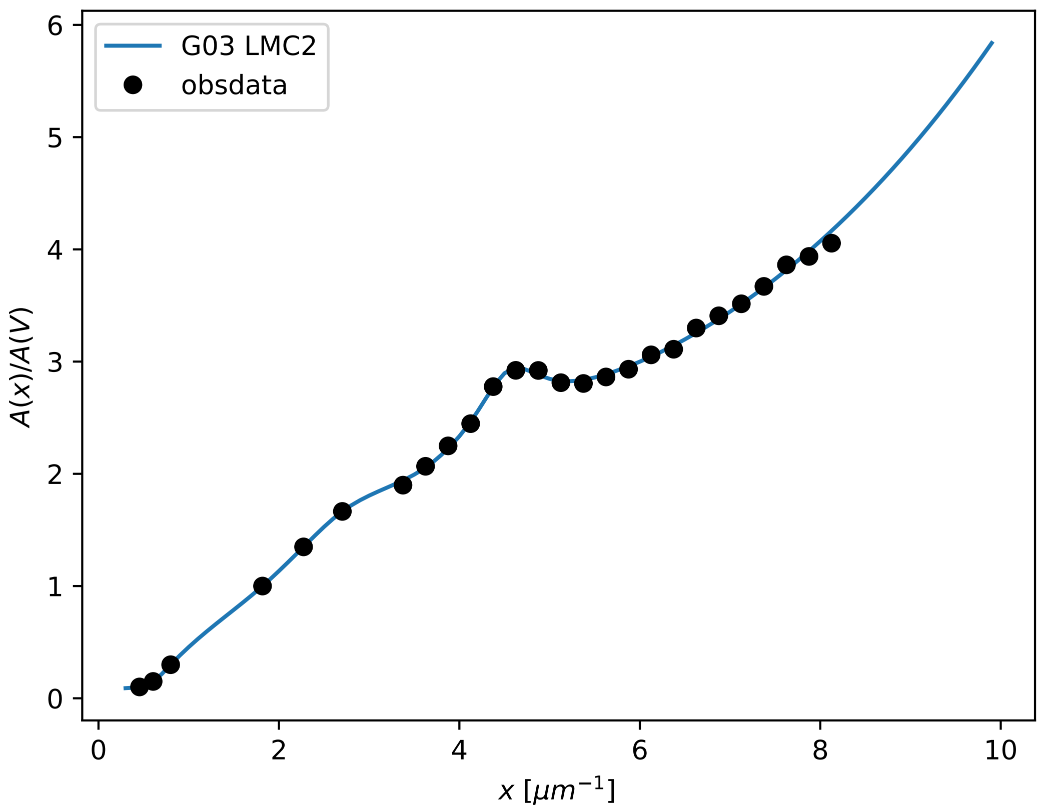
4.1.11 G16

class dust_extinction.dust_extinction.**G16**(*RvA=3.1*, *fA=1.0*, ***kwargs*)

Bases: dust_extinction.dust_extinction.BaseExtModel

G16 extinction model calculation

Mixture model between the F99 R(V) dependent model (component A) and the G03_SMCBar model (component B)


G03 LMC2
obsdata
A(x)/A(V)
x [μm⁻¹]
0
1
2
3
4
5
6
0
2
4
6
8
10

Parameters

RvA: float

R_A(V) = A(V)/E(B-V) = total-to-selective extinction R(V) of the A component

fA: float

f_A is the mixture coefficent between the R(V)

Raises

InputParameterError

Input RvA values outside of defined range Input fA values outside of defined range

Notes

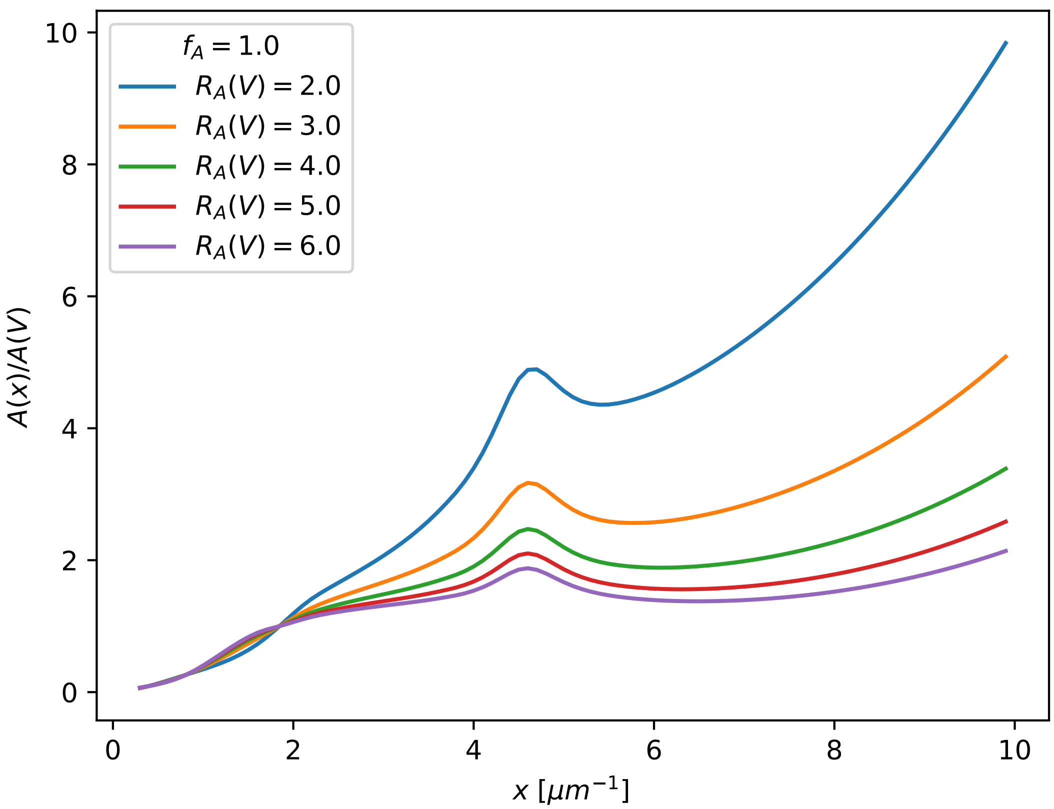
G16 R_A(V) and f_A dependent model

From Gordon et al. (2016, ApJ, 826, 104)

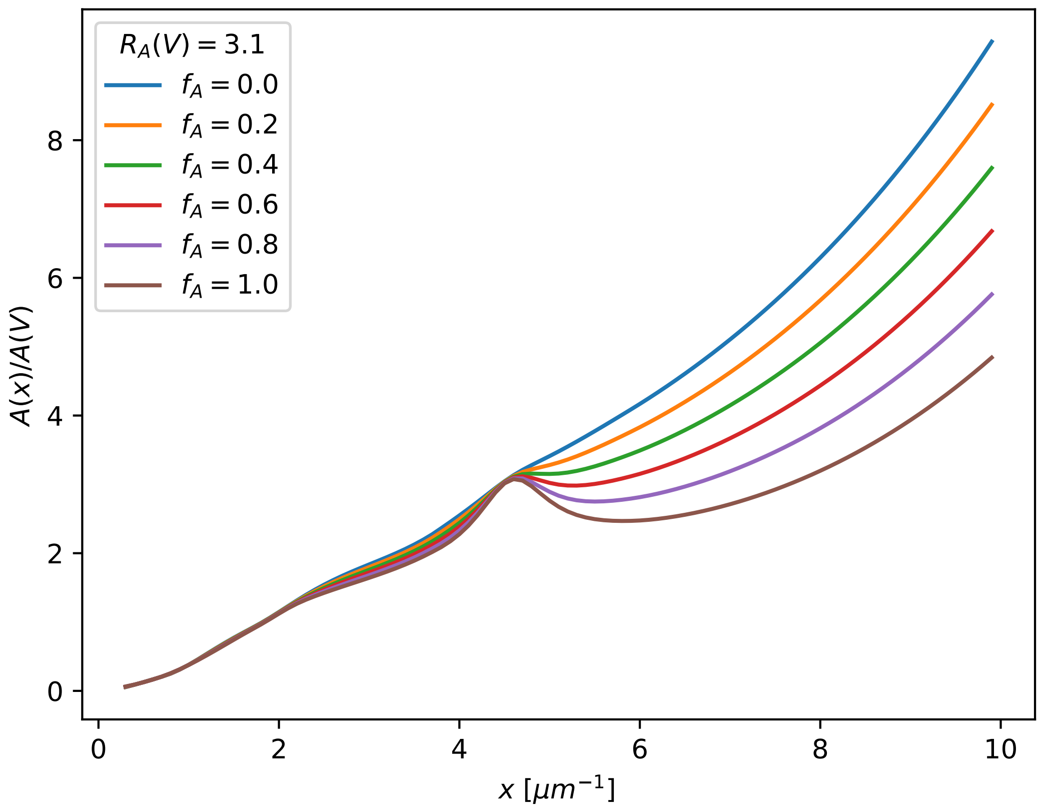
Example showing G16 curves for a range of R_A(V) values and f_A values.

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import G16

fig, ax = plt.subplots()

# temp model to get the correct x range
text_model = G16()

# generate the curves and plot them
x = np.arange(text_model.x_range[0], text_model.x_range[1],0.1)/u.micron

Rvs = ['2.0','3.0','4.0','5.0','6.0']
for cur_Rv in Rvs:
   ext_model = G16(RvA=cur_Rv, fA=1.0)
   ax.plot(x,ext_model(x),label=r'$R_A(V) = ' + str(cur_Rv) + '$')

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$A(x)/A(V)$')

ax.legend(loc='best', title=r'$f_A = 1.0$')
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
import astropy.units as u

from dust_extinction.dust_extinction import G16

fig, ax = plt.subplots()

# temp model to get the correct x range
text_model = G16()

# generate the curves and plot them
```


$f_A = 1.0$
$R_A(V) = 2.0$
$R_A(V) = 3.0$
$R_A(V) = 4.0$
$R_A(V) = 5.0$
$R_A(V) = 6.0$
$A(x)/A(V)$
x [μm^{-1}]
0
2
4
6
8
10

```
x = np.arange(text_model.x_range[0], text_model.x_range[1],0.1)/u.micron

fAs = [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]
for cur_fA in fAs:
    ext_model = G16(RvA=3.1, fA=cur_fA)
    ax.plot(x,ext_model(x),label=r'$f_A = ' + str(cur_fA) + '$')

ax.set_xlabel('$x$ [$\mu m^{-1}$]')
ax.set_ylabel('$A(x)/A(V)$')

ax.legend(loc='best', title=r'$R_A(V) = 3.1$')
plt.show()
```



Attributes Summary

RvA	R_A(V) = A(V)/E(B-V) = total-to-selective extinction of component A
RvA_range	
fA	f_A = mixture coefficent of component A
fA_range	
param_names	

Continued on next page

Table 4.21 – continued from previous page

x_range

Methods Summary

evaluate(in_x, RvA, fA)	G16 function

Attributes Documentation

RvA
R_A(V) = A(V)/E(B-V) = total-to-selective extinction of component A

RvA_range = [2.0, 6.0]

fA
f_A = mixture coefficent of component A

fA_range = [0.0, 1.0]

param_names = ('RvA', 'fA')

x_range = [0.3, 10.0]

Methods Documentation

static evaluate(*in_x*, *RvA*, *fA*)
G16 function

Parameters
in_x: float

expects either x in units of wavelengths or frequency or assumes wavelengths in wavenumbers [1/micron]

internally wavenumbers are used

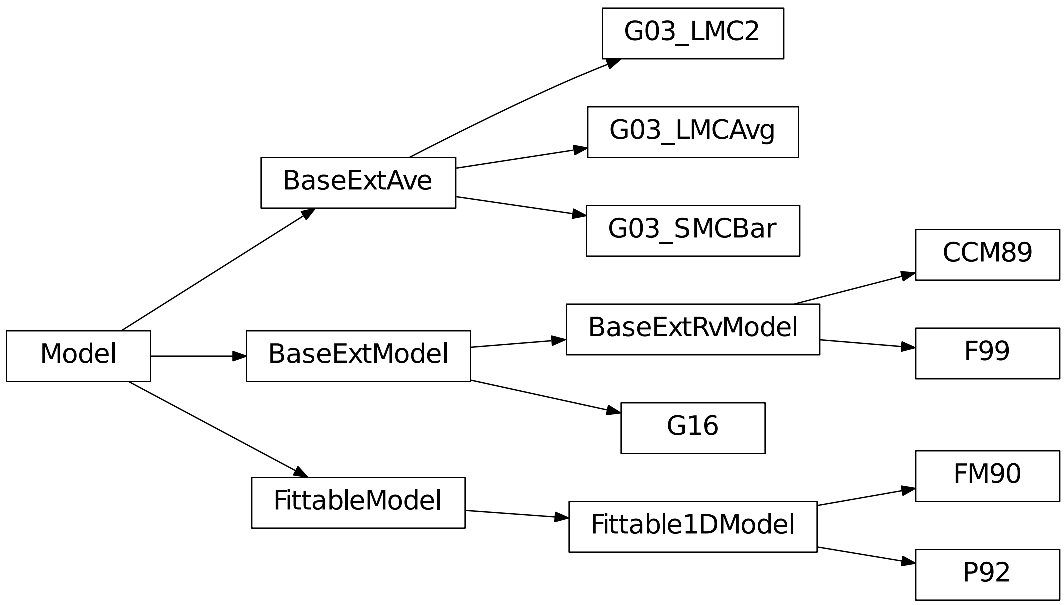
Returns
axav: np array (float)

A(x)/A(V) extinction curve [mag]

Raises
ValueError

Input x values outside of defined range

4.2 Class Inheritance Diagram



Python Module Index

d

Index

Symbols

A

B

C

D

E

F

G

I

N

O

P

R

S

X