PyCosmo Documentation

Release 1.0

Ian Harrison & Christopher Lovell

July 24, 2013

CONTENTS

Contents:

CHAPTER ONE

PYCOSMO PACKAGE

1.1 `cluster` Module

`cmb` Module

class `PyCosmo.cluster.`**`clust`**(*lnm*, *redshift*, *cosm*)

> **`display`**(*pixsize=None*)
>
> **`info`**()
>
> **`make_ymap`**(*pixsize=None*)
>
> **`trim`**(*edge*, *npix*)
>
> **`y`**(*theta*)

Python implementation of Geraint Pratten's code for generating flat-sky CMB maps...

class `PyCosmo.cmb.`**`CMBFlatMap`**(*mapsize=10.0*, *pixels=1024*, *cosm=<PyCosmo.cosmology.Cosmology instance at 0x37e44d0>*)

> **`add_cluster_ymap`**(*cl*)
> Add Compton Y from an object of the Cluster class to the map as temp. TO FIX: edges.
>
> **`add_noise`**(*temp*)
> Add per-pixel Gaussian random noise.
>
> **`add_ymap`**(*ymap*, *freq=1.0*)
> Add a Compton Y map (e.g. from LSS tSZ) to the map as temperature. DeltaT(x) = -2*T_{CMB}*Y(x)
>
> **`apply_beam`**(*beamtype*, *fwhm*)
> Apply a beam function to map. Currently 'beamtype' is redundant and a Gaussian beam is always used.
>
> ****fwhm is in degrees****
>
> **`beta_profile_map`**(*theta_crit*, *cutoff_factor=10.0*, *deltaT0=1.0*)
>
> **`build_Pk`**(*cosm*)
> Calculate correct flat-sky P(k) from external Cls (angular power spectrum).
>
> **`display`**()
> Plot the flat-sky CMB.
>
> **`matched_filter`**(*noisesigma*)

matched_filter_Melin (*theta_c*, *xfilt*, *noisesigma*, *deltaT0=1.0*)
Matched filter with a spherical beta profile as in SPT paper

1.2 `constants` Module

constants (dictionary)

1.3 `cosmology` Module

(MILDLY) USEFUL COSMOLOGY CLASS. NOTE ALL UNITS ARE WRT h^-1

Much of the cosmography is from (as ever) Hogg arXiv:astro-ph/9905116

(C) IAN HARRISON 2012- IAN.HARRISON@ASTRO.CF.AC.UK

class `PyCosmo.cosmology.`**Cosmology** (*dc=1.686*, *h0=0.702*, *om=0.274*, *ode=0.725*, *w0=-1.0*, *ob=0.0458*, *o_r=8.6e-05*, *o_k=0.0*, *f_nl=0.0*, *tau_r=0.087*, *z_r=10.4*, *ns=0.96*, *hmf_type='tinker'*)

D_a (*z*)
Angular diameter distance to redshift z.

D_c (*z*)
Comoving radial distance to redshift z.

D_l (*z*)
Luminosity distance to redshift z.

H (*z*)
Hubble function ***upper*** H(z).

O_m (*z*)
Omega matter.

V_between (*z_min*, *z_max*)
Volume between two redshifts.

computeLittleNinZBin (*lnm_min*, *lnm_max*, *z*)
Total number of haloes within a given mass bin at fixed redshift.

computeNinBin (*z_min*, *z_max*, *lnm_min*, *lnm_max*)
Total number of dark matter haloes expected within a given mass, redshift bin.

dNdlnm0dz (*lnm*, *z*)
Total number of dark matter haloes, of equivalent mass at redshift zero, at a given redshift. Product of dndlnm * dVdz

dNdlnmdz (*lnm*, *z*)
Total number of dark matter haloes at a given redshift. Product of dndlnm * dVdz

display ()
Displays what you're working with.

dist_mod (*z*)
Distance modulus. 5 * log(D_l(z)) + 25

dndlnm (*lnm*, *z*)
Comoving number density of dark matter haloes in logarithmic m.

dvdz (*z*)
Comoving volume element at redshift z.

eta (*z*)
Size of particle horizon at redshift z

growth (*z*)
Linear growth function.

h (*z*)
Hubble function ***little*** h(z).

rho_c ()
Critical density

rho_m (*z*)
Average density of Universe at redshift z

set_hmf (*set_mf*)
Set method for the HMF within the Cosmology

set_powspec (*set_pk*)
Set method for power spectrum within the Cosmology

t_lookback (*z*)
Lookback time to redshift z

1.4 `cosmology_prevec` Module

(MILDLY) USEFUL COSMOLOGY CLASS. NOTE ALL UNITS ARE WRT h^-1

Much of the cosmography is from (as ever) Hogg arXiv:astro-ph/9905116

(C) IAN HARRISON 2012- IAN.HARRISON@ASTRO.CF.AC.UK

class PyCosmo.cosmology_prevec.**Cosmology** (*dc=1.686*, *h0=0.702*, *om=0.274*, *ode=0.725*, *w0=-1.0*, *ob=0.0458*, *o_r=8.6e-05*, *o_k=0.0*, *f_nl=0.0*, *tau_r=0.087*, *z_r=10.4*, *ns=0.96*)

D_a (*z*)
Angular diameter distance to redshift z.

D_c (*z*)
Comoving radial distance to redshift z.

D_l (*z*)
Luminosity distance to redshift z.

H (*z*)
Hubble function ***upper*** H(z).

O_m (*z*)
Omega matter.

V_between (*z_min*, *z_max*)
Volume between two redshifts.

computeNinBin (*z_min*, *z_max*, *lnm_min*, *lnm_max*)
Total number of dark matter haloes expected within a given mass, redshift bin.

dNdlnm0dz (*lnm*, *z*)
Total number of dark matter haloes, of equivalent mass at redshift zero, at a given redshift. Product of dndlnm * dVdz

dNdlnmdz (*lnm*, *z*)
Total number of dark matter haloes at a given redshift. Product of dndlnm * dVdz

display ()
Displays what you're working with.

dist_mod (*z*)
Distance modulus. 5 * log(D_l(z)) + 25

dndlnm (*lnm*, *z*)
Comoving number density of dark matter haloes in logarithmic m.

dvdz (*z*)
Comoving volume element at redshift z.

eta (*z*)
Size of particle horizon at redshift z

growth (*z*)
Linear growth function.

h (*z*)
Hubble function ***little*** h(z).

rho_c ()
Critical density

rho_m (*z*)
Average density of Universe at redshift z

set_hmf (*set_mf*)
Set method for the HMF within the Cosmology

set_powspec (*set_pk*)
Set method for power spectrum within the Cosmology

t_lookback (*z*)
Lookback time to redshift z

1.5 `hmf` Module

(HOPEFULLY) USEFUL HALO MASS FUNCTION CLASS. NOTE ALL UNITS ARE WRT h^-1

(C) IAN HARRISON 2012- IAN.HARRISON@ASTRO.CF.AC.UK

class `PyCosmo.hmf.`**Hmf** (*mf_type='tinker'*, *rng_type='pgh'*)

display ()
Display method shows the name and parameters of the current mass function

ps (*sigma*, *z*)
Press-Schechter halo mass function. from Press, W. H., Schechter, P., 1974, ApJ, 187, 425

sigma : the mass variance on a particular scale z : redshift

f_ps : the collapse fraction

r_pgh (*sigma*, *delta_c*, *f_nl*)
Paranjape-Gordon-Hotchkiss non-Gaussian correction factor for halo mass functions.

st (*sigma*, *z*)
Sheth-Tormen halo massfunction, with corrections for ellipsoidal collapse.

Equation 10 from arXiv:astro-ph/9901122

sigma : the mass variance on a particular scale z : redshift

f_st : the collapse fraction

tinker (*sigma*, *z*)
Tinker halo mass function with evolving parameters. Equations 3, 5-8 from arXiv:0803.2706

sigma : the mass variance on a particular scale z : redshift

f_t : the collapse fraction

1.6 `hmf_extremes` Module

Code for reproducing EVS of the z=0 halo mass function (as in Harrison & Coles 2011). New python code to supercede the old (finicky) C++ one.

PyCosmo.hmf_extremes.**evs_bin_pdf** (*z_min=0.0*, *z_max=1.0*, *z_steps=200*, *lnm_min=32.236191301916641*, *lnm_max=39.143946580898778*, *lnm_steps=200*, *cosm=<PyCosmo.cosmology.Cosmology instance at 0x3af5200>*, *fsky=1.0*)
Calculate extreme value statistics of cold dark matter haloes in a given mass and redshift bin.

Uses the method described in Harrison & Coles 2012 arXiv:1111.1184

phi_max : The EVS pdf lnm_arr : The x-points for the pdf

PyCosmo.hmf_extremes.**evs_hypersurface_pdf** (*r_box=20.0*, *lnm_min=27.631021115928547*, *lnm_max=41.446531673892821*, *redshift=0.0*, *cosm=<PyCosmo.cosmology.Cosmology instance at 0x3557c20>*, *lnm_steps=200*)
Calculate Extreme Value Statistics for dark matter haloes in a given cosmology on a specified spatial hypersurface (constant z box).

Uses the method described in Harrison & Coles 2011 arXiv:0000.0000

phi_max : The EVS pdf lnm_arr : The x-points for the pdf

PyCosmo.hmf_extremes.**evs_survey** (*surv=<PyCosmo.survey.Survey instance at 0x3af52d8>*, *cosm=<PyCosmo.cosmology.Cosmology instance at 0x3af5320>*, *n_bins=100*, *CLs=(66.0, 95.0, 99.0)*)
Produce M_max vs z plot for a given survey, cosmology and number of z bins.

Uses the method described in Harrison & Coles 2012 arXiv:1111.1184

n_bins : number of redshift bins CLs : tuple of requested confidence regions

FIXME!

1.7 `powspec` Module

(HOPEFULLY) USEFUL POWER SPECTRUM CLASS. NOTE ALL UNITS ARE WRT h^-1

class `PyCosmo.powspec.PowSpec`(*cosmology*)

`choose`()
Choose between an Eisenstein & HU fitting function or a CAMB power spectrum

`display`()
Display method to show power spectrum currently working with.

`dlnsigma_dlnm`(*mrange*, *z*)
slope of root matter variance wrt log mass: d(log(sigma)) / d(log(M))

Polynomial fit to supplied cosmology. Returns poly1d object

`dlnsigma_dlnr`(*rrange*, *z*)
slope of root matter variance wrt log radius: d(log(sigma)) / d(log(r))

Polynomial fit to supplied cosmology. Returns poly1d object

`dlnsigmadlnm_wmap7fit`(*lnm*)
Slope of root matter variance wrt log mass: d(log(sigma)) / d(log(m))

Polynomial fit to calculation from a CAMB power spectrum with WMAP7 parameters

`growth_func`(*z*)
initialises growth function variable as part of PowSpec instance

`import_powerspectrum`(*ident*, *z=0.0*)
import power spectrum function from a CAMB produced output file

`interpolate`(*array_1*, *array_2*)
returns a function that uses interpolation to find the value of new points

`power_spectrum_P`(*k*, *z*)
returns the power spectrum P(k)

`sigma_fit`(*rrange*, *sigma_r*)

`sigma_integral`(*k*, *r*, *z*)
returns the integral required to calculate sigma squared (Coles & Lucchin pg.266, A.Zentner 06 eq.14)

`sigma_r`(*r*, *z*)
returns root of the matter variance, smoothed with a top hat window function at a radius r

`sigma_r_sq`(*r*, *z*)
integrate the function in sigma_integral between the limits of k : 0 to inf.

`sigma_r_sq_vec` = <numpy.lib.function_base.vectorize object at 0x34b1ad0>

`sigma_wmap7fit`(*lnm*)
Root of matter variance smoothed with top hat window function on a scale specified by log(m)

Polynomial fit to calculation from a CAMB power spectrum with WMAP7 parameters

`tophat_w`(*k*, *r*)
Fourier transform of the real space tophat window function (eq.9 from A.Zentner 06)

`transfer_function_EH`(*k*, *z*)
Calculates transfer function given wavenumber

`vd_initialisation`(*z*, *rrange*, *mrange*)
initialise parameters required for void_distribution.py script

1.8 `survey` Module

(HOPEFULLY) USEFUL OBSERVATIONAL SURVEY CLASS. NOTE ALL UNITS ARE WRT h^-1

(C) IAN HARRISON 2012- IAN.HARRISON@ASTRO.CF.AC.UK

class `PyCosmo.survey.`**`Survey`**(*zmin=0.0*, *zmax=2.0*, *lnmmin=33.845629214350737*, *lnmmax=36.841361487904734*, *fsky=1.0*)

`N_in_survey`(*cosm*)
Calculate total number of haloes expected to exist within the observational survey window.

1.9 `utils` Module

`PyCosmo.utils.`**`as2deg`**(*arcsecs*)
Utility function for converting arcseconds to degrees

1.10 `void_distribution` Module

Python script for reproducing the distribution of number density of voids

`PyCosmo.void_distribution.`**`multiplicity_function_jlh`**(*sigma*, *D*, *void_barrier*, *collapse_barrier*)
Jennings, Li & Hu f(lnsigma) approximation

`PyCosmo.void_distribution.`**`multiplicity_function_jlh_exact`**(*sigma*, *D*, *void_barrier*, *collapse_barrier*)
Jennings, Li & Hu f(lnsigma) approximation

`PyCosmo.void_distribution.`**`multiplicity_function_svdw`**(*nu*, *D*, *void_barrier*, *collapse_barrier*)
calculates equation (4) in Sheth & van de Weygaert approximating the infinite series in equation (1)

`PyCosmo.void_distribution.`**`scaled_void_distribution`**(*nu*, *void_barrier=-2.7*, *collapse_barrier=1.06*)
'A Hierarchy of Voids : Sheth & van de Weygaert' Reproduces a scaled distribution of void masses/sizes shown in figure(7)

`PyCosmo.void_distribution.`**`void_Fr`**(*norm*, *r*, *ps*, *max_record=True*)
F(r) from Harrison & Coles (2012) known distribution of void radii

for max_record=True, calculates the cumulative distribution *upto* a given radii, otherwise calculates EVS for small radii

`PyCosmo.void_distribution.`**`void_Fr_small`**(*norm*, *r*, *ps*)
F(r) from Harrison & Coles (2012) known distribution of void radii

`PyCosmo.void_distribution.`**`void_and_cloud`**(*void_barrier*, *collapse_barrier*)

`PyCosmo.void_distribution.`**`void_fr`**(*norm*, *r*, *ps*)
f(r) from Harrison & Coles (2012) pdf of the original void distribution

`PyCosmo.void_distribution.`**`void_mass_dist`**(*m*, *ps*, *cosm*, *z=0.0*, *void_barrier=-2.7*, *collapse_barrier=1.06*)
produces the differential number density of voids wrt their characteristic mass

`PyCosmo.void_distribution.`**`void_norm`**(*ps*)
n_tot from Harrison & Coles (2012) normalisation factor; gives the total comoving number density of voids

PyCosmo.void_distribution.**void_pdf**(*r*, *norm*, *ps*, *V*, *max_record=True*)

phi(max) from Harrison & Coles (2012) exact extreme value pdf of the original void distribution for a given radius

PyCosmo.void_distribution.**void_radii_dist**(*r*, *ps*, *z=0.0*, *void_barrier=-2.7*, *collapse_barrier=1.06*)

Produces the differential number density of voids wrt to their characteristic radius

1.11 Subpackages

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

p