Working with Cosmologies

astropy.cosmology

Create and realize cosmologies

Key concepts:

Cosmology objects

Built-in Realizations

Redshifts and Distances

Equivalencies

astropy.cosmology

Cosmology objects

Built-in Realizations

Redshifts and Distances

Equivalencies

- Objects that represent cosmologies (e.g. FLRW)
- Describe with standard parameters (e.g. Neff)
- Compute properties (e.g. the age, …)

```
>>> from astropy.cosmology import FlatLambdaCDM
>>> cosmo.FlatLambdaCDM(H0=70, Om0=0.3)
>>> cosmo.H0
<Quantity 70. km / (Mpc s)>

>>> cosmo.age(z=0.1)
<Quantity 12.16041497 Gyr>
```

astropy.cosmology

Cosmology objects

Built-in Realizations

Redshifts and Distances

Equivalencies

- Standard cosmology realizations are built in :

  WMAP[5,7,9], Planck[13,15,18]

```
>>> from astropy.cosmology import Planck18
>>> Planck18
FlatLambdaCDM(name="Planck18",H0=67.7 km/(Mpc s),
              ...)

>>> Planck18.lookback_time(z=2)
<Quantity 10.51005492 Gyr>
```

astropy.cosmology

Cosmology objects

Built-in Realizations

Redshifts and Distances

Equivalencies

- Finding the redshift at a given value of a cosmological quantity.

```
>>> import astropy.units as u
>>> from astropy.cosmology import z_at_value
>>> z_at_value(Planck18.critical_density, 1e-10*
u.g/u.cm**3)
2.6180339515344797
```

- Cosmologies integrate with coordinate distances

```
>>> from astropy.coordinates import Distance
>>> d = Distance(z=4e3, cosmology=Planck18)
>>> d.distmod
<Quantity 63.75184309 mag>
```

astropy.cosmology

Cosmology objects

Built-in Realizations

Redshifts and Distances

Equivalencies

- Cosmologies also support unit *equivalency* conversions, e.g., in terms of the Hubble constant.

```
>>> distance = 67.66 * (u.Mpc/u.littleh)
>>> distance.to(u.Mpc)
```

fails: incompatible units

```
>>> eqvlt = u.with_H0(Planck18.H0))
>>> with u.add_enabled_equivalencies(eqvlt):
>>>     print(distance.to(u.Mpc))
<Quantity 100. Mpc>
```