Tools for data analyses in Cosmology

- Aula 8 -

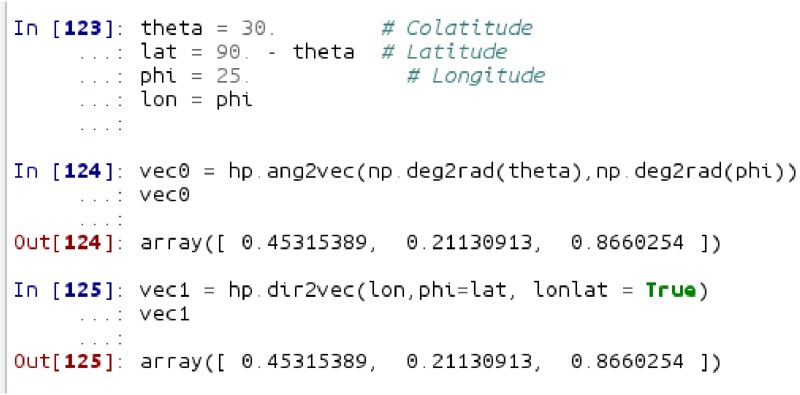
Camila Novaes

Observatório Nacional

June 1, 2017

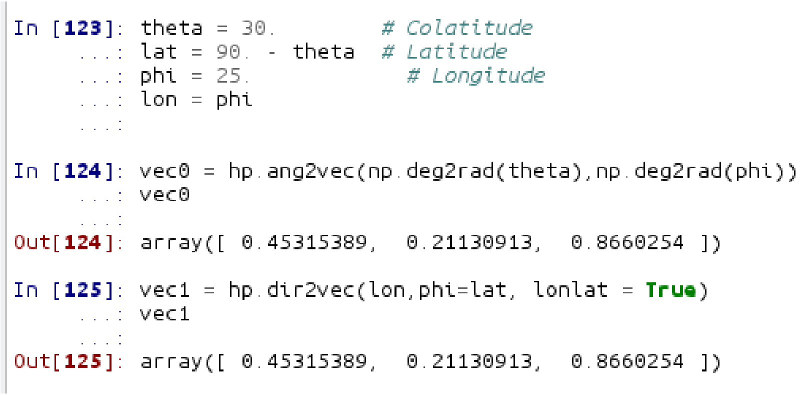
Healpy

Previously on healpy class ...



```
In [123]: theta = 30.         # Colatitude
     ...: lat = 90. - theta  # Latitude
     ...: phi = 25.            # Longitude
     ...: lon = phi
     ...:

In [124]: vec0 = hp.ang2vec(np.deg2rad(theta),np.deg2rad(phi))
     ...: vec0
     ...:
Out[124]: array([ 0.45315389,  0.21130913,  0.8660254 ])

In [125]: vec1 = hp.dir2vec(lon,phi=lat, lonlat = True)
     ...: vec1
     ...:
Out[125]: array([ 0.45315389,  0.21130913,  0.8660254 ])
```

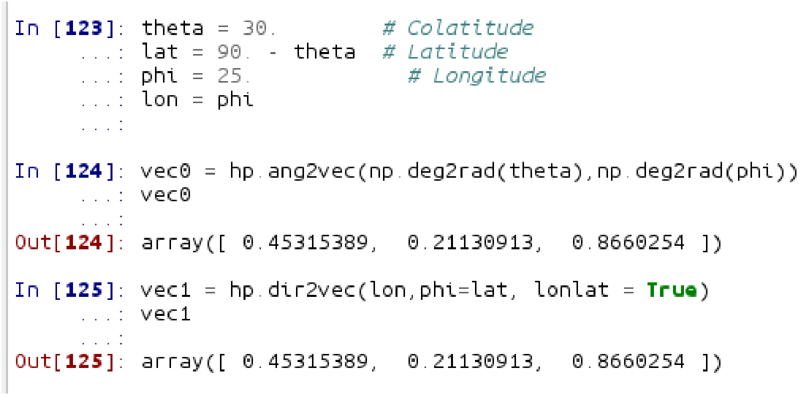
Previously on healpy class ...



```
In [123]: theta = 30.        # Colatitude
     ...: lat = 90. - theta  # Latitude
     ...: phi = 25.            # Longitude
     ...: lon = phi
     ...:

In [124]: vec0 = hp.ang2vec(np.deg2rad(theta),np.deg2rad(phi))
     ...: vec0
     ...:
Out[124]: array([ 0.45315389,  0.21130913,  0.8660254 ])

In [125]: vec1 = hp.dir2vec(lon,phi=lat, lonlat = True)
     ...: vec1
     ...:
Out[125]: array([ 0.45315389,  0.21130913,  0.8660254 ])
```

Exercise: Which is the vector corresponding to the position of the Virgo cluster?
$\ell, b = 283.8^\circ, 74.4^\circ \Rightarrow$

Previously on healpy class ...



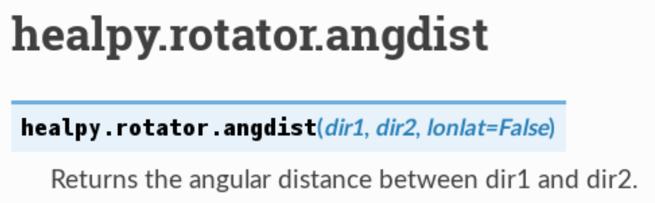
```
In [123]: theta = 30.        # Colatitude
     ...: lat = 90. - theta  # Latitude
     ...: phi = 25.            # Longitude
     ...: lon = phi
     ...:

In [124]: vec0 = hp.ang2vec(np.deg2rad(theta),np.deg2rad(phi))
     ...: vec0
     ...:
Out[124]: array([ 0.45315389,  0.21130913,  0.8660254 ])

In [125]: vec1 = hp.dir2vec(lon,phi=lat, lonlat = True)
     ...: vec1
     ...:
Out[125]: array([ 0.45315389,  0.21130913,  0.8660254 ])
```

Exercise: Which is the vector corresponding to the position of the Virgo cluster?

$\ell, b = 283.8^\circ, 74.4^\circ \Rightarrow$

x,y,z = [0.06414637, -0.26115726, 0.96316257]

Rotation and geometry functions

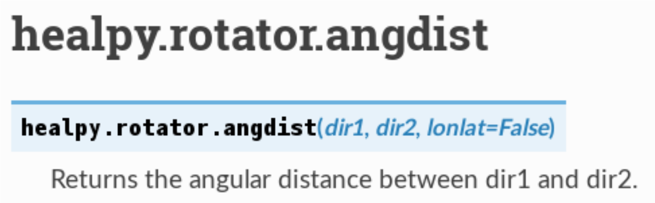


healpy.rotator.angdist

healpy.rotator.angdist(*dir1, dir2, lonlat=False*)

Returns the angular distance between dir1 and dir2.

Rotation and geometry functions



healpy.rotator.angdist

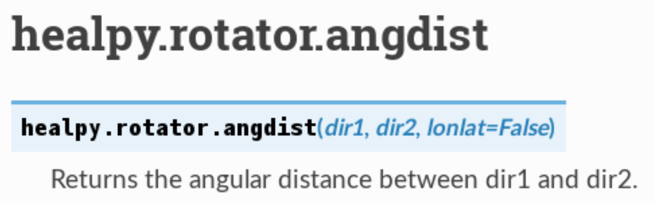
```
healpy.rotator.angdist(dir1, dir2, lonlat=False)
```

Returns the angular distance between dir1 and dir2.

Exercise: Which is the angular distance between `Virgo` and `Coma` clusters?

`Virgo:`	$\ell, b = 283.8^\circ, 74.4^\circ$
`Coma:`	$\ell, b = 235.1^\circ, 73.0^\circ$

Rotation and geometry functions



healpy.rotator.angdist

`healpy.rotator.angdist(dir1, dir2, lonlat=False)`

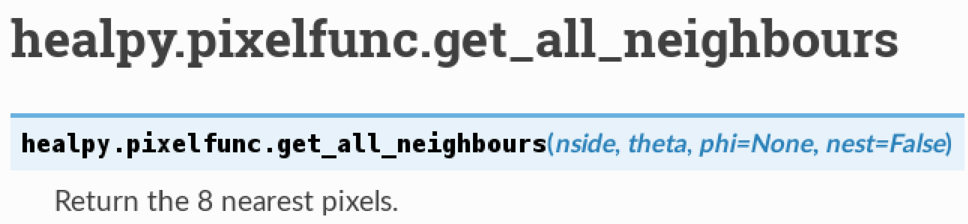
Returns the angular distance between dir1 and dir2.

Exercise: Which is the angular distance between `Virgo` and `Coma` clusters?

`Virgo:` $\ell, b = 283.8^\circ, 74.4^\circ$
`Coma:` $\ell, b = 235.1^\circ, 73.0^\circ$

`Result:  Distance = 0.23 rad = 13.35 deg`

Rotation and geometry functions



healpy.pixelfunc.get_all_neighbours

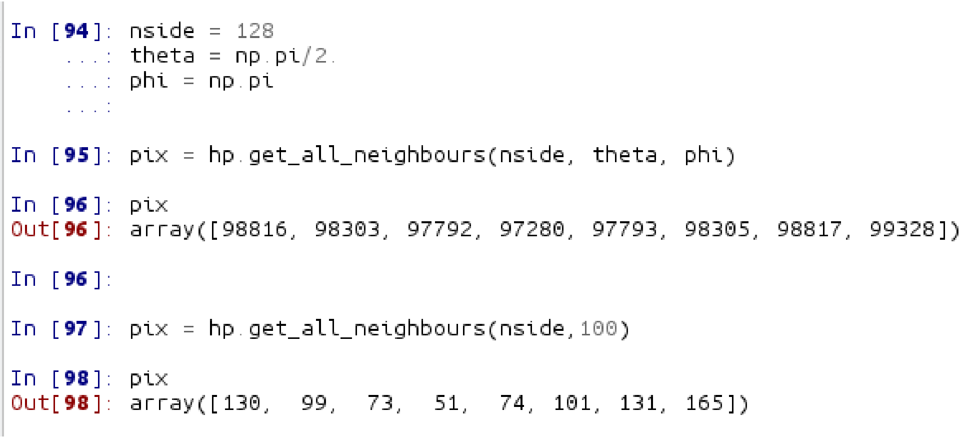
```
healpy.pixelfunc.get_all_neighbours(nside, theta, phi=None, nest=False)
```

Return the 8 nearest pixels.

Sequence: SW, W, NW, N, NE, E, SE and S neighbours

Rotation and geometry functions

How to use:

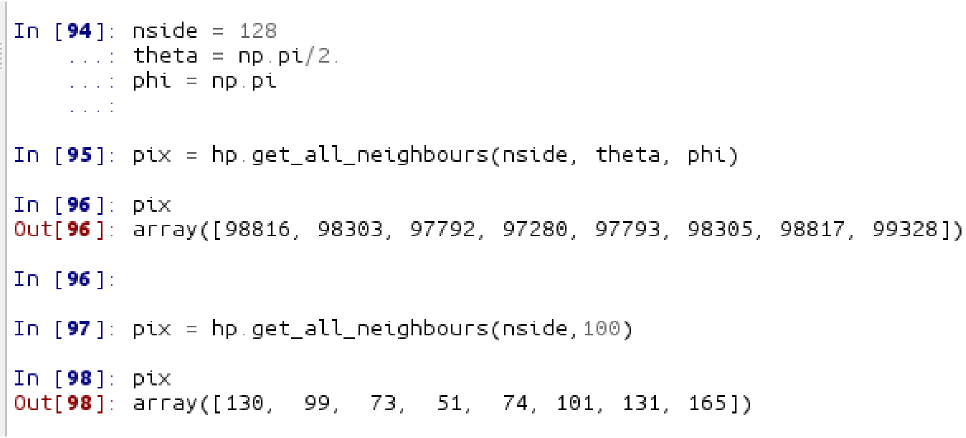


```
In [94]: nside = 128
    ...: theta = np.pi/2.
    ...: phi = np.pi
    ...:

In [95]: pix = hp.get_all_neighbours(nside, theta, phi)

In [96]: pix
Out[96]: array([98816, 98303, 97792, 97280, 97793, 98305, 98817, 99328])

In [96]:

In [97]: pix = hp.get_all_neighbours(nside,100)

In [98]: pix
Out[98]: array([130,  99,  73,  51,  74, 101, 131, 165])
```

Neighbours of pixel 5432 at Nside=1024?

Rotation and geometry functions

How to use:



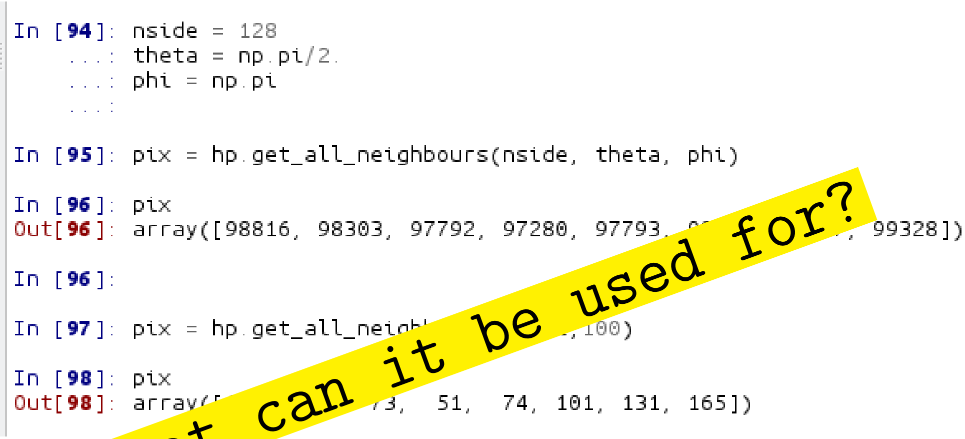
```
In [94]: nside = 128
    ...: theta = np.pi/2.
    ...: phi = np.pi
    ...:

In [95]: pix = hp.get_all_neighbours(nside, theta, phi)

In [96]: pix
Out[96]: array([98816, 98303, 97792, 97280, 97793, 98305, 98817, 99328])

In [96]:

In [97]: pix = hp.get_all_neighbours(nside,100)

In [98]: pix
Out[98]: array([130,  99,  73,  51,  74, 101, 131, 165])
```

Neighbours of pixel 5432 at Nside=1024?

```
Out[1]:  array([5642, 5431, 5225, 5023, 5226, 5433,
5643, 5857])
```

Rotation and geometry functions

How to use:



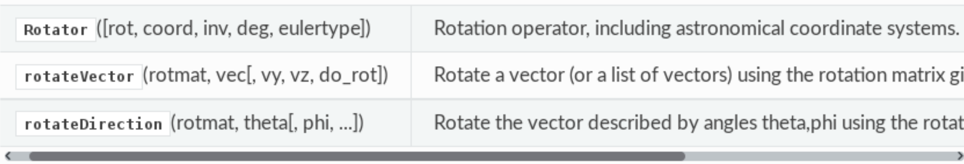
```
In [94]: nside = 128
    ...: theta = np.pi/2.
    ...: phi = np.pi
    ...:

In [95]: pix = hp.get_all_neighbours(nside, theta, phi)

In [96]: pix
Out[96]: array([98816, 98303, 97792, 97280, 97793, 9                , 99328])

In [96]:

In [97]: pix = hp.get_all_neigh                ,100)

In [98]: pix
Out[98]: array(            3,   51,   74,  101,  131,  165])
```

What can it be used for?

of pixel 5432 at Nside=1024?

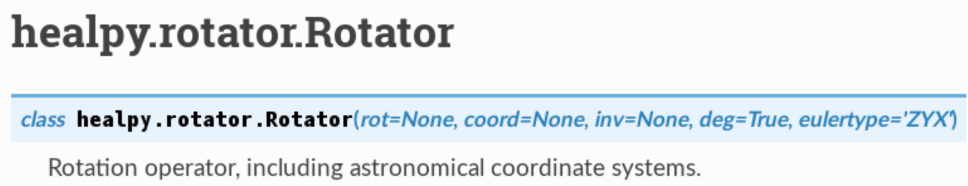
O [1]: array([5642, 5431, 5225, 5023, 5226, 5433, 5643, 5857])

Rotation and geometry functions



`Rotator` ([rot, coord, inv, deg, eulertype])	Rotation operator, including astronomical coordinate systems.
`rotateVector` (rotmat, vec[, vy, vz, do_rot])	Rotate a vector (or a list of vectors) using the rotation matrix gi
`rotateDirection` (rotmat, theta[, phi, ...])	Rotate the vector described by angles theta,phi using the rotat

Rotation and geometry functions



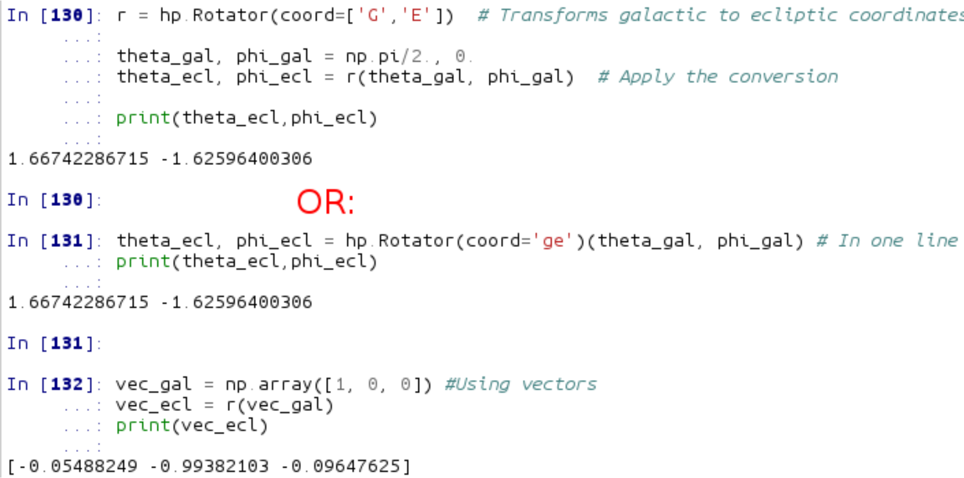
healpy.rotator.Rotator

class **healpy.rotator.Rotator**(*rot=None, coord=None, inv=None, deg=True, eulertype='ZYX'*)

Rotation operator, including astronomical coordinate systems.

Rotation and geometry functions

How to use:



```
In [130]: r = hp.Rotator(coord=['G','E'])  # Transforms galactic to ecliptic coordinates
     ...:
     ...: theta_gal, phi_gal = np.pi/2., 0.
     ...: theta_ecl, phi_ecl = r(theta_gal, phi_gal)  # Apply the conversion
     ...:
     ...: print(theta_ecl,phi_ecl)
     ...:
1.66742286715 -1.62596400306

In [130]:
```

OR:

```
In [131]: theta_ecl, phi_ecl = hp.Rotator(coord='ge')(theta_gal, phi_gal) # In one line
     ...: print(theta_ecl,phi_ecl)
     ...:
1.66742286715 -1.62596400306

In [131]:

In [132]: vec_gal = np.array([1, 0, 0]) #Using vectors
     ...: vec_ecl = r(vec_gal)
     ...: print(vec_ecl)
     ...:
[-0.05488249 -0.99382103 -0.09647625]
```

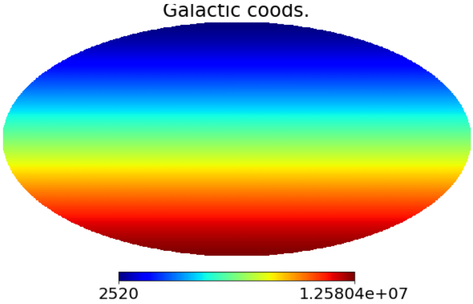
Rotation and geometry functions

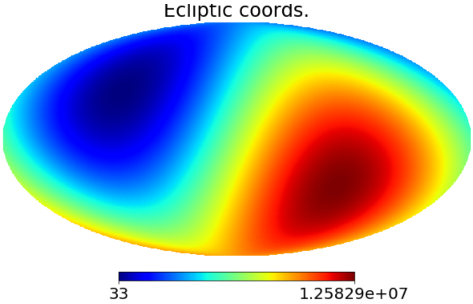
Exercise:

- Change the coordinates for the whole sky. Consider the pixels in $\mathtt{N}_{\mathtt{side}}$ = 1024. Steps:
  - [1] Generate an array (`pixels`) with the pixel indexes corresponding to this resolution (`np.arange`).
  - [2] Calculate the θ, ϕ coordinates of all the pixels (`hp.pix2ang`).
  - [3] Rotate them to find the new $\theta_{\mathtt{ecl}}, \phi_{\mathtt{ecl}}$.
  - [4] Convert these ecliptic coordinates back to pixel indexes (`pixels2`) (`hp.ang2pix`).
  - [5] Visualize the distribution.

Rotation and geometry functions

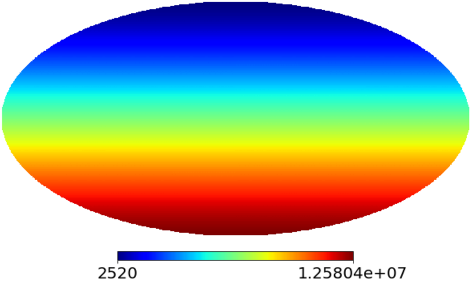
Results:



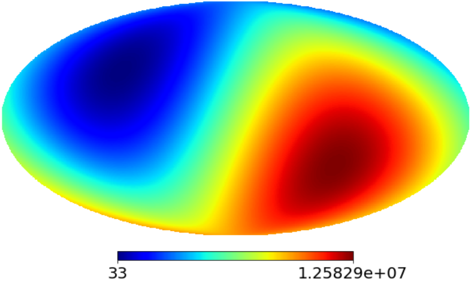


Rotation and geometry functions

Results:



Galactic coods.

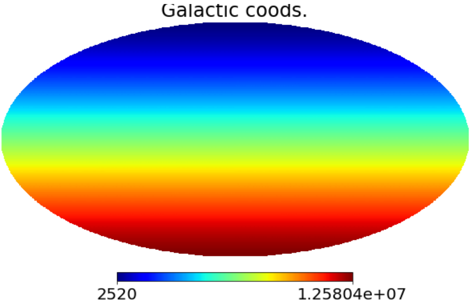


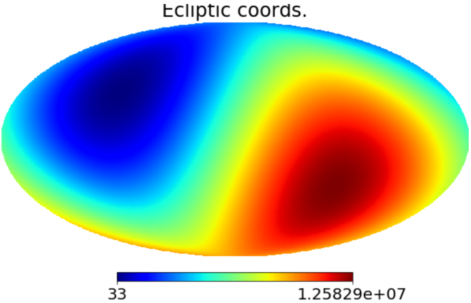
Ecliptic coords.

How to apply this coordinate transform in a map?

Rotation and geometry functions

Results:



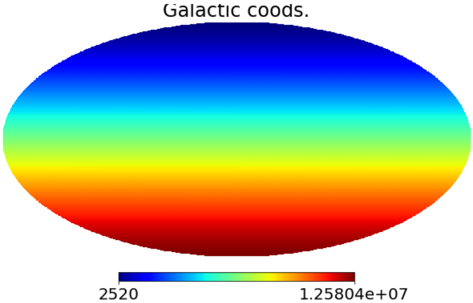


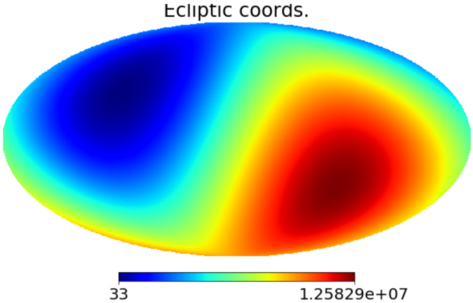
How to apply this coordinate transform in a map?

How to perform a rotation instead of a coordinate transform?

Rotation and geometry functions

Results:





How to apply this coordinate transform in a map?

How to perform a rotation instead of a coordinate transform?

Lets see ...