Tools for data analyses in Cosmology

- Aula 7 -

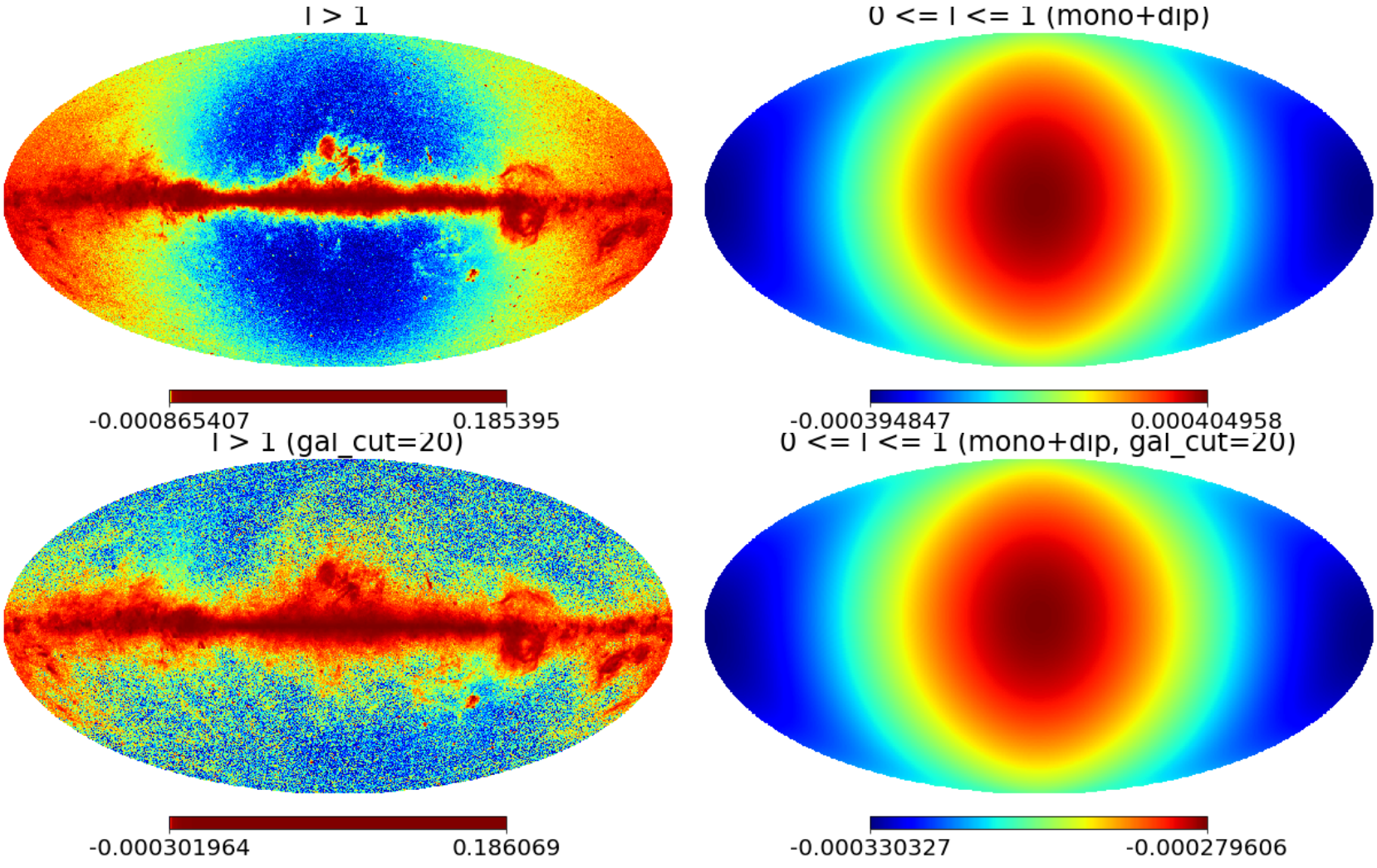
Camila Novaes

Observatório Nacional

May 30, 2017

Healpy

Previously on healpy class ...



Map data manipulation

How to use:

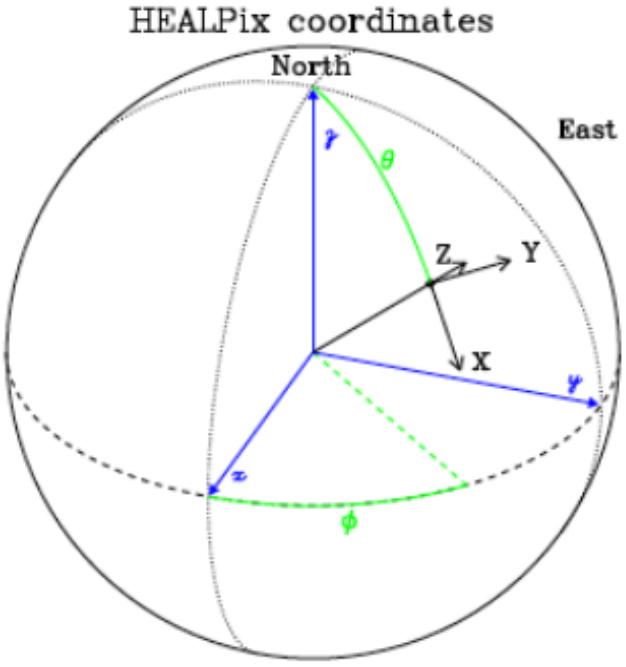
```
In [30]: mapa3 = hp.remove_dipole(mapa, fitval = True)
monopole: 5.05606e-06  dipole: lon: 1.28489, lat: 0.000777875, amp: 0.000399906

In [31]: len(mapa3)
Out[31]: 3

In [32]: mapa3[0]
Out[32]:
masked_array(data = [-0.00038791 -0.00038744 -0.00038133 ..., -0.00032639
-0.00028807
 -0.00043686],
             mask = False,
       fill_value = -1.6375e+30)

In [33]: mapa3[1]
Out[33]: 5.056058058099675 3e-06

In [34]: mapa3[2]
Out[34]: array([  3.99804999e-04,   8.96738494e-06,   5.42931005e-09])

In [35]:
```

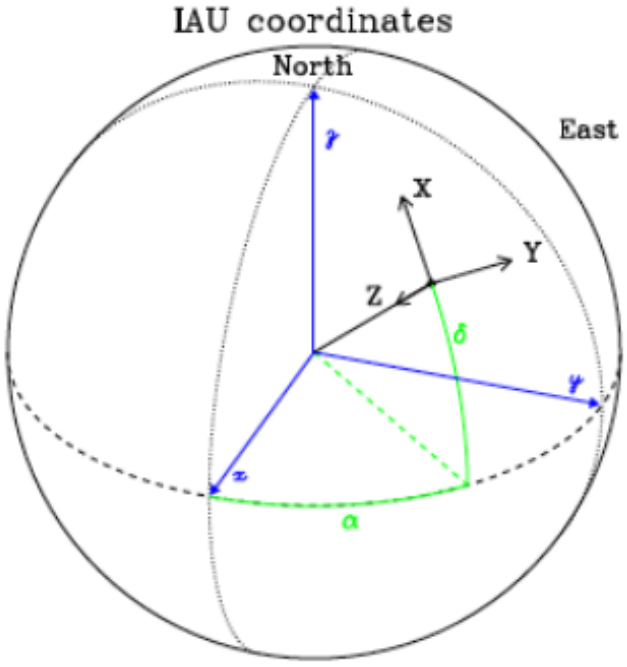
Map data manipulation

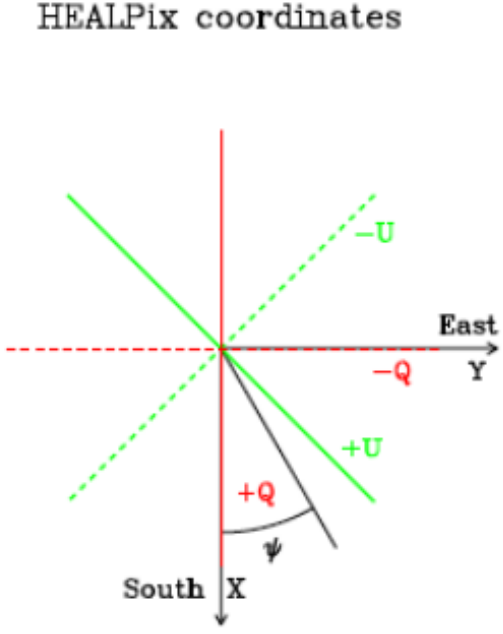
Showing an important example!!!

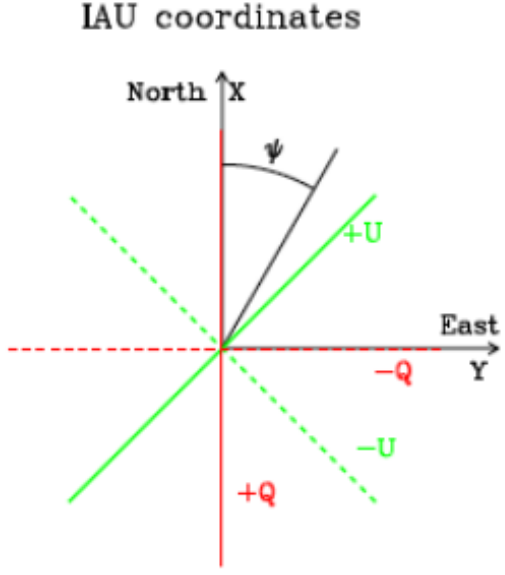
Let's go to spyder ...

HEALPix conventions









HEALPix conventions

Pixel functions: conversions between `pixel number` in the HEALPix map and (θ, ϕ) or `(x,y,z)` coordinates on the sphere.

`Parameters:`

- $\mathtt{N_{side}}$
- `ipix`: pixel identification number in `RING/NEST` scheme over the range $\{\mathtt{0},\ \mathtt{N_{pix}-1}\}$.
- `nest`: True or False
- `theta`: colatitude (`b = 90 - theta`) in radians measured southward from north pole in $\{0, \pi\}$.
- `phi`: longitude in radians, measured eastward in $\{0, 2\pi\}$.
- `vector`: three dimensional cartesian position vector `(x,y,z)`. [The north pole is `(0,0,1)`].

Pixelisation related functions

healpy.pixelfunc.pix2ang

healpy.pixelfunc.pix2ang(*nside, ipix, nest=False*)

pix2ang : nside,ipix,nest=False -> theta[rad],phi[rad] (default RING)

healpy.pixelfunc.ang2pix

healpy.pixelfunc.ang2pix(*nside, theta, phi, nest=False*)

ang2pix : nside,theta[rad],phi[rad],nest=False -> ipix (default:RING)

Pixelisation related functions

How to use:

```
In [7]: ang = hp.pix2ang(16, 1440)
   ...: print('Pix = 1440 --> (Theta,Phi) =',ang)
   ...:
Pix = 1440 --> (Theta,Phi) = (1.5291175943723188, 0.0)
```

```
In [45]: theta0 = np.deg2rad(45) # rad
    ...: phi0 = np.deg2rad(30)    # rad
    ...:
    ...: ipix0 = hp.ang2pix(4, theta0, phi0)
    ...:
    ...: print('(theta,phi)=(',theta0,',',phi0,') --> ipix =',ipix0)
    ...:
(theta,phi)=( 0.785398163397 , 0.523598775598 ) --> ipix = 25
```

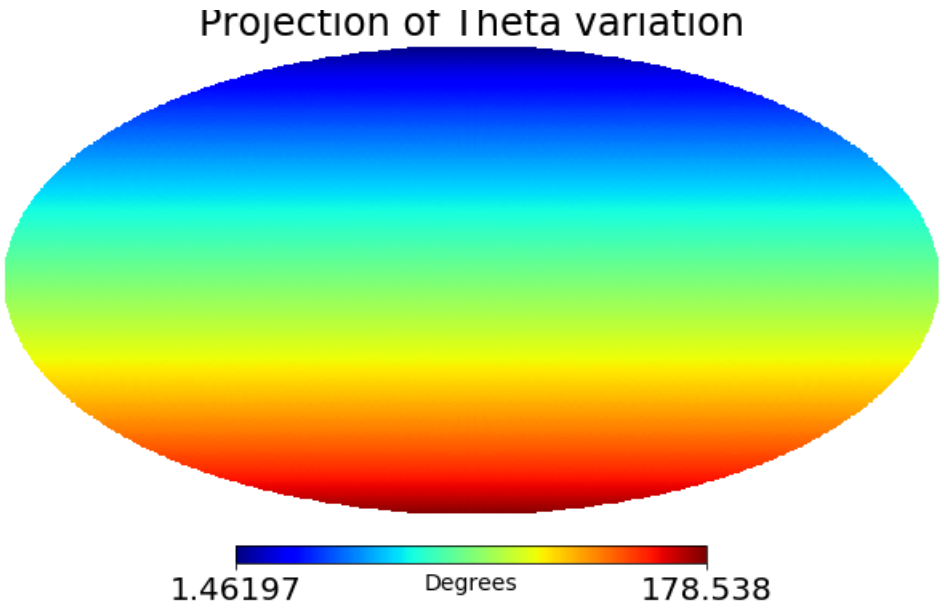
Pixelisation related functions

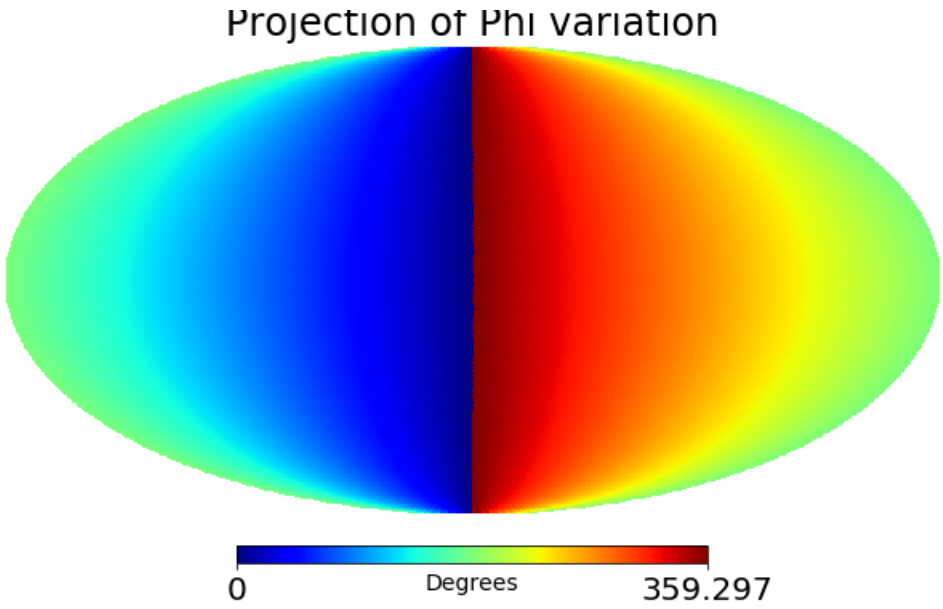
Exercise:

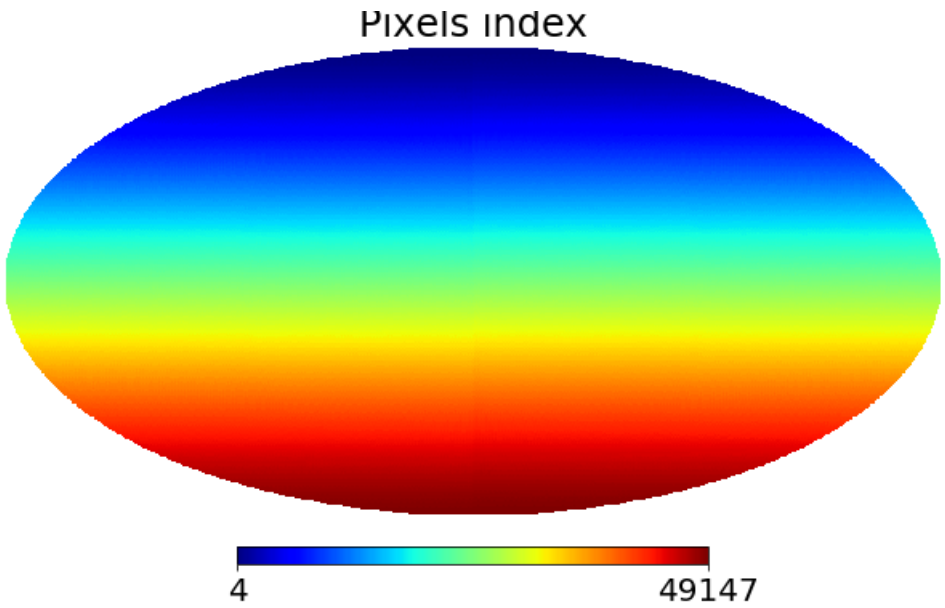
- Calculate the direction (θ, ϕ) for all the pixels in a resolution of $\mathtt{N_{side}} = 64$. [Remind the command `np.arange()` to generate a list of numbers.]
- Visualize in a Mollweide projection the θ and ϕ you just calculate.
- From these θ and ϕ, calculated the corresponding pixel indexes.
- Visualize the array of pixel indexes in a Mollweide.

Pixelisation related functions

Results:







Pixelisation related functions

healpy.pixelfunc.pix2vec

healpy.pixelfunc.pix2vec(*nside, ipix, nest=False*)

pix2vec : nside,ipix,nest=False -> x,y,z (default RING)

healpy.pixelfunc.vec2pix

healpy.pixelfunc.vec2pix(*nside, x, y, z, nest=False*)

vec2pix : nside,x,y,z,nest=False -> ipix (default:RING)

```
In [70]: vec1 = hp.pix2vec(16, 1504)
    ...: print('Pix = 1504 --> (x,y,z) =',vec1)
    ...:
Pix = 1504 --> (x,y,z) = (0.99879545620517241, 0.049067674327418015, 0.0)

In [71]: vec2 = hp.pix2vec(16, [1440,  427])
    ...: vec2
    ...:
Out[71]:
(array([ 0.99913157,  0.5000534 ]),
 array([ 0.        ,  0.5000534]),
 array([ 0.04166667,  0.70703125]))

In [72]: print('Pix = 1440 --> (x,y,z) = (',vec2[0][0], vec2[1][0], vec2[2][0],')')
    ...: print('Pix = 427 --> (x,y,z) = (', vec2[0][1], vec2[1][1], vec2[2][1],')')
    ...:
Pix = 1440 --> (x,y,z) = ( 0.999131567357 0.0 0.0416666666667 )
Pix = 427 --> (x,y,z) = ( 0.50005340291 0.50005340291 0.70703125 )

In [73]: vec3 = hp.pix2vec([1, 2], 11)
    ...: vec3
    ...:
Out[73]:
(array([ 0.52704628,  0.68861915]),
 array([-0.52704628, -0.28523539]),
 array([-0.66666667,  0.66666667]))

In [74]: print('Nside=1 / Pix = 11  --> (x,y,z) = (',vec3[0][0], vec3[1][0], vec3[2]
[0],')')
    ...: print('Nside=2 / Pix = 11  --> (x,y,z) = (',vec3[0][1], vec3[1][1], vec3[2]
[1],')')
    ...:
Nside=1 / Pix = 11  --> (x,y,z) = ( 0.527046276695 -0.527046276695 -0.666666666667 )
Nside=2 / Pix = 11  --> (x,y,z) = ( 0.688619145905 -0.285235389544 0.666666666667 )
```

Pixelisation related functions

How to use:

```
In [83]: vec2 = hp.pix2vec(16, [1440,  427])
    ...: vec2
    ...:
Out[83]:
(array([ 0.99913157,  0.5000534 ]),
 array([ 0.        ,  0.5000534]),
 array([ 0.04166667,  0.70703125]))

In [84]: # Inveting:::
    ...:
    ...: ipix2 = hp.vec2pix(16, vec2[0],vec2[1], vec2[2])
    ...: ipix2
    ...:
Out[84]: array([1440,  427])
```

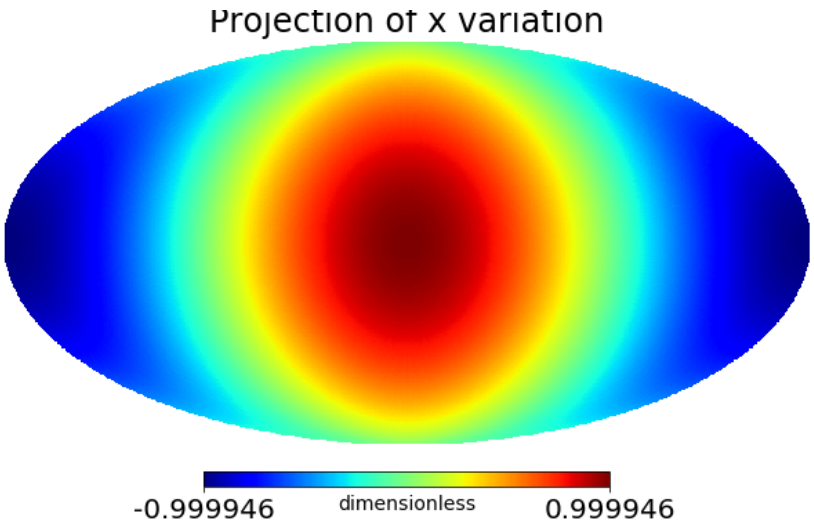
Pixelisation related functions

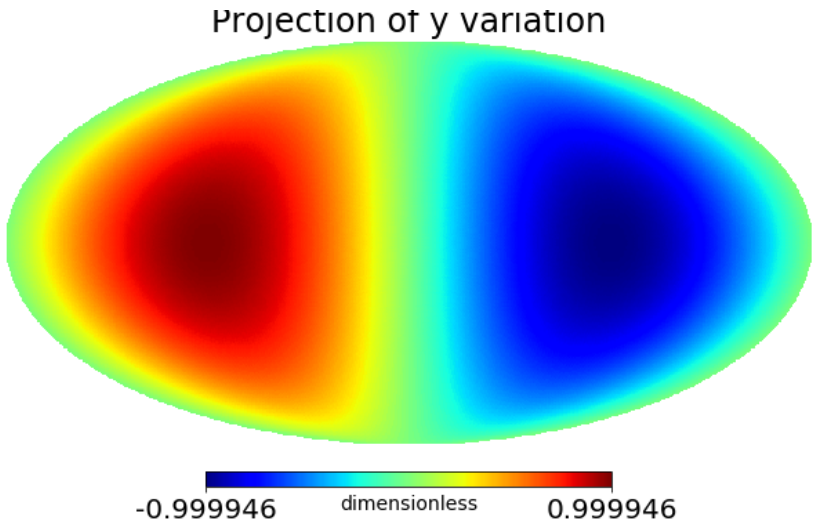
Exercise: Let's see how the `x`, `y` and `z` variate in the sky with N_{side} = 64!!!

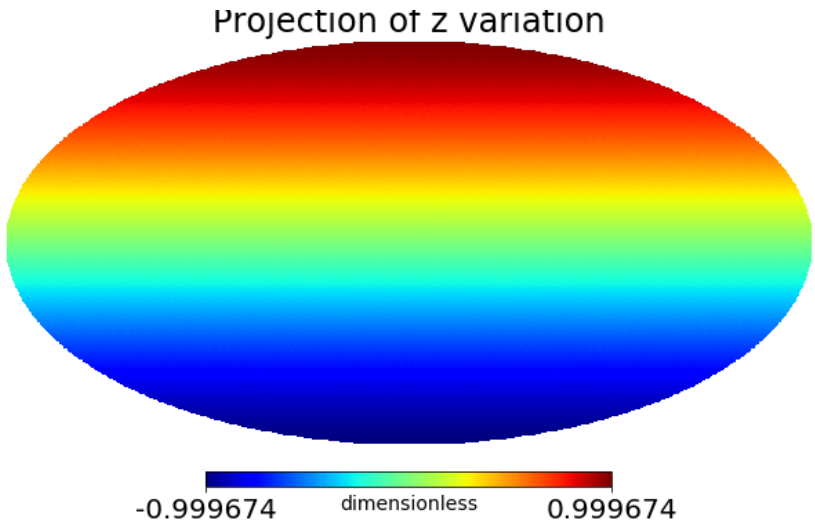
Pixelisation related functions

Exercise: Let's see how the `x`, `y` and `z` variate in the sky with $N_{side} = 64$!!!

Results:







Pixelisation related functions

healpy.pixelfunc.ang2vec

healpy.pixelfunc.ang2vec(*theta, phi*)

ang2vec : convert angles to 3D position vector

healpy.pixelfunc.vec2ang

healpy.pixelfunc.vec2ang(*vectors*)

vec2ang: vectors [x, y, z] -> theta[rad], phi[rad]

Pixelisation related functions

How to use:

```
In [93]: theta = np.deg2rad(45.) # rad
    ...: phi   = np.deg2rad(30.) # rad
    ...:
    ...: vec = hp.ang2vec(theta,phi)
    ...:
    ...: print('(theta,phi) = (',theta,',',phi,') -->
(x,y,z)=', vec)
    ...:
(theta,phi) = ( 0.785398163397 , 0.523598775598 ) -->
(x,y,z)= [ 0.61237244  0.35355339  0.70710678]
```

```
In [94]: theta_phi = hp.vec2ang(vec)
    ...:
    ...: print('(x,y,z)=', vec,' --> (theta,phi) =
(',theta_phi,')')
    ...:
(x,y,z)= [ 0.61237244  0.35355339  0.70710678]  -->
(theta,phi) = ( (array([ 0.78539816]), array([
0.52359878])) )
```

Rotation and geometry functions

healpy.rotator.dir2vec

healpy.rotator.dir2vec(*theta, phi=None, lonlat=False*)

Transform a direction theta,phi to a unit vector.

healpy.rotator.vec2dir

healpy.rotator.vec2dir(*vec, vy=None, vz=None, lonlat=False*)

Transform a vector to angle given by theta,phi.

Rotation and geometry functions

How to use: which is the difference?

```
In [112]: # Create a theta,phi list:
     ...: nside = 64
     ...: npix = hp.nside2npix(nside)
     ...: pixels = np.arange(npix)
     ...: theta_phi = hp.pix2ang(nside, pixels)
     ...:

In [113]: # ang2vec vs. dir2vec
     ...: vec = hp.ang2vec(theta_phi[0],theta_phi[1]) # it works
     ...: vec
     ...:
Out[113]:
array([[ 0.00902091,  0.00902091,  0.99991862],
       [-0.00902091,  0.00902091,  0.99991862],
       [-0.00902091, -0.00902091,  0.99991862],
       ...,
       [-0.00902091,  0.00902091, -0.99991862],
       [-0.00902091, -0.00902091, -0.99991862],
       [ 0.00902091, -0.00902091, -0.99991862]])

In [114]: vec = hp.ang2vec(theta_phi)                    # it DOES NOT work
Traceback (most recent call last):

  File "<ipython-input-114-3771c6580265>", line 1, in <module>
    vec = hp.ang2vec(theta_phi)                  # it DOES NOT work

TypeError: ang2vec() missing 1 required positional argument: 'phi'
```

Rotation and geometry functions

How to use: which is the difference?

```
In [115]: vec = hp.dir2vec(theta_phi[0],phi=theta_phi[1]) # it works!!!
     ...: vec
     ...:
Out[115]:
array([[ 0.00902091, -0.00902091, -0.00902091, ..., -0.00902091,
        -0.00902091,  0.00902091],
       [ 0.00902091,  0.00902091, -0.00902091, ...,  0.00902091,
        -0.00902091, -0.00902091],
       [ 0.99991862,  0.99991862,  0.99991862, ..., -0.99991862,
        -0.99991862, -0.99991862]])

In [116]: vec = hp.dir2vec(theta_phi)                    # it works!!!
     ...: vec
     ...:
Out[116]:
array([[ 0.00902091, -0.00902091, -0.00902091, ..., -0.00902091,
        -0.00902091,  0.00902091],
       [ 0.00902091,  0.00902091, -0.00902091, ...,  0.00902091,
        -0.00902091, -0.00902091],
       [ 0.99991862,  0.99991862,  0.99991862, ..., -0.99991862,
        -0.99991862, -0.99991862]])
```

Rotation and geometry functions

How to use: which is the difference?

```
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 ])
```

Rotation and geometry functions

How to use: which is the difference?

```
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?

$\ell, b = 103.3°, -2.8° \Rightarrow$

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