Tools for data analyses in Cosmology

- Aula 5 -

Camila Novaes

Observatório Nacional

May 23, 2017

Healpy

Previously on healpy class ...

- Writing fits files -

healpy.fitsfunc.write_map

healpy.fitsfunc.write_map(*filename, m, nest=False, dtype=<type 'numpy.float32'>, fits_IDL=True, coord=None, partial=False, column_names=None, column_units=None, extra_header=()*)

healpy.fitsfunc.mwrfits

healpy.fitsfunc.mwrfits(*filename, data, hdu=1, colnames=None, keys=None*)

ERROR -

Previously on healpy class ...

- Writing fits files - solving partially (and properly) the problem

In `fitsfunc.py` substitute:

`clobber`	$\rightarrow$	`overwrite`
`pf.new_table`	$\rightarrow$	`pf.BinTableHDU.from_columns`
`header.update`	$\rightarrow$	`header.set`

healpy.fitsfunc.write_map

healpy.fitsfunc.write_map(*filename, m, nest=False, dtype=<type 'numpy.float32'>, fits_IDL=True, coord=None, partial=False, column_names=None, column_units=None, extra_header=()*)

healpy.fitsfunc.mwrfits

healpy.fitsfunc.mwrfits(*filename, data, hdu=1, colnames=None, keys=None*)

Previously on healpy class ...

How to use:

```
In [95]: import pyfits
    ...:
    ...: x = data[10]
    ...: hdu = pyfits.PrimaryHDU(x)
    ...: hdulist = pyfits.HDUList([hdu])
    ...: hdulist.writeto('new.fits')
    ...: hdulist.close()
    ...:
    ...: hdulist = pyfits.open('new.fits')
    ...: data2 = hdulist[0].data
    ...:
```

Still from VISUALIZATION tools

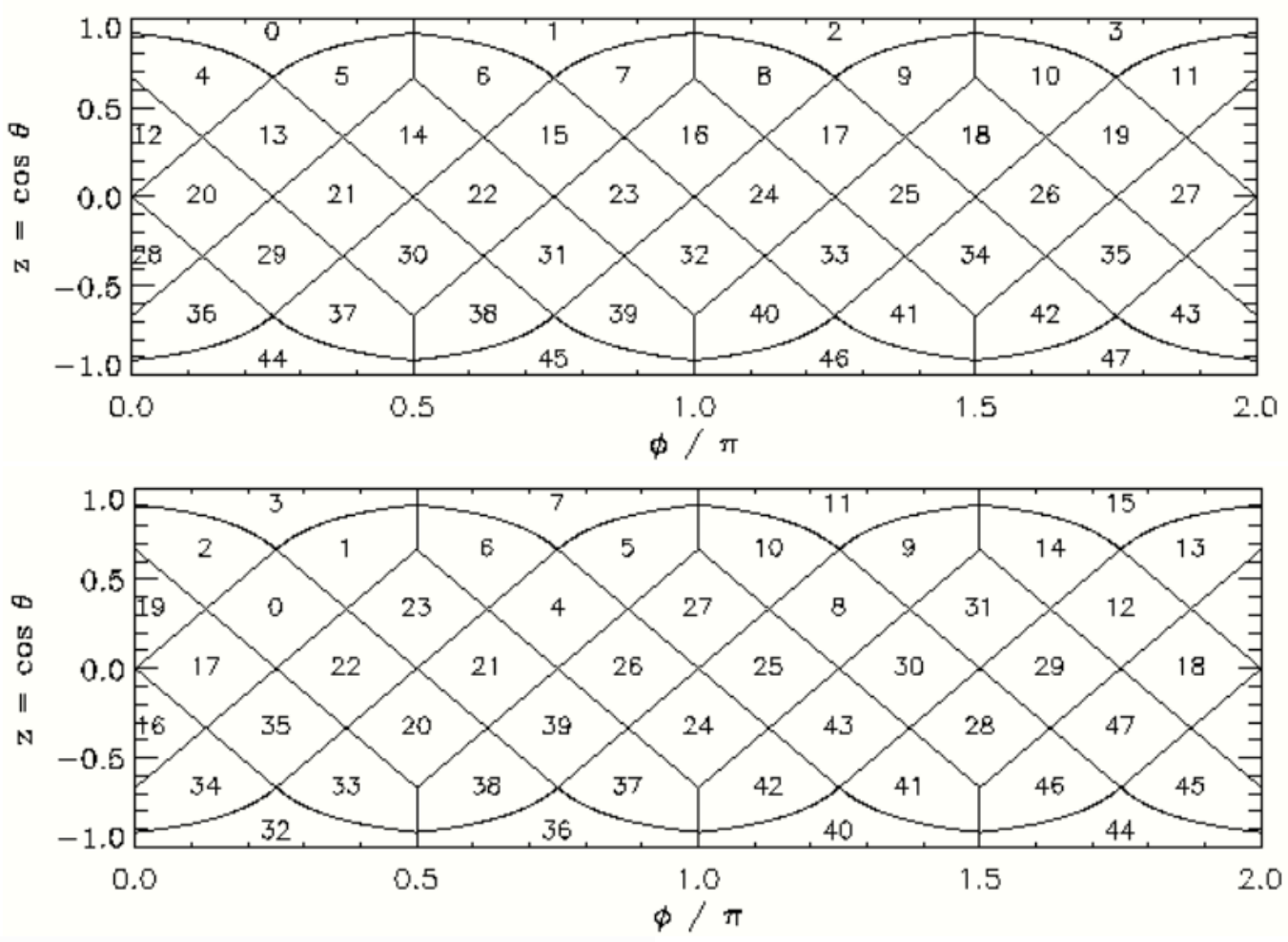
Tracing lines or points

projplot (*args, **kwds)

projscatter (*args, **kwds)

projtext (*args, **kwds)

Conversion between NESTED and RING schemes



healpy.pixelfunc.ring2nest

healpy.pixelfunc.ring2nest(*nside, ipix*)

healpy.pixelfunc.nest2ring

healpy.pixelfunc.nest2ring(*nside, ipix*)

Conversion between NESTED and RING schemes

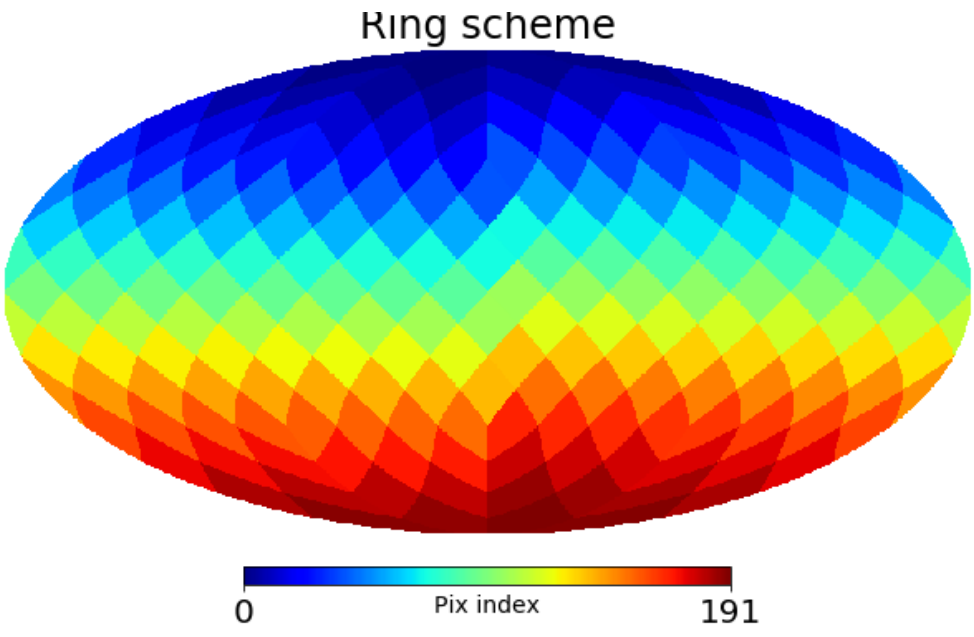
How to use:

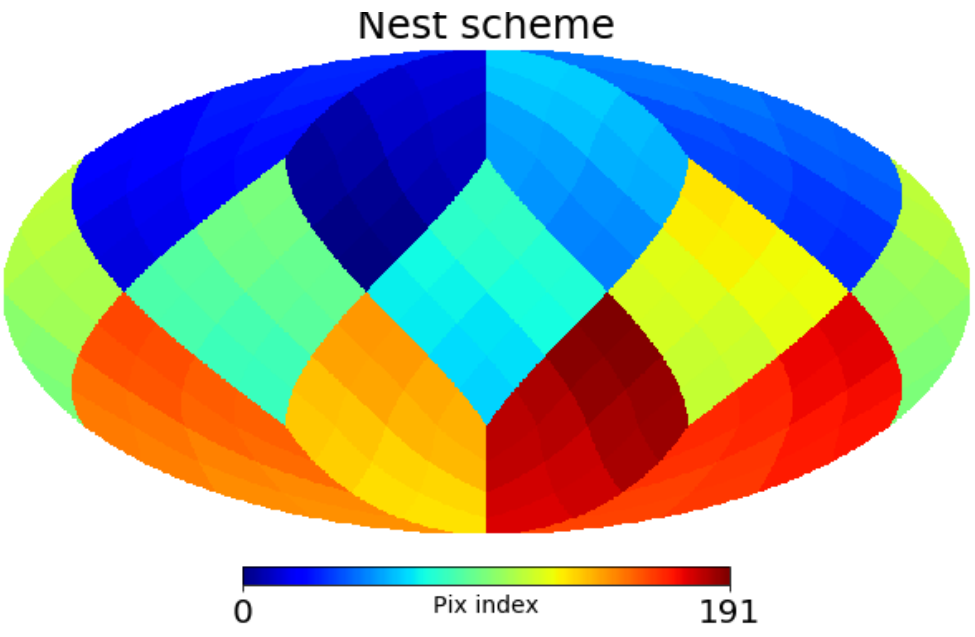
RING ⇒ NEST

```
In [34]: Nside = 4
    ...: pix_ring = 25
    ...: pix_nest = hp.ring2nest(Nside, pix_ring)
    ...: print('pix_nest =', pix_nest)
    ...:
pix_nest = 9

In [35]: all_pix_ring = np.arange(192)
    ...: all_pix_nest = hp.ring2nest(Nside, all_pix_ring)
    ...: hp.mollview(all_pix_nest, title='Nest scheme',
unit='Pix index')
    ...: pyplot.savefig('mollweide_view_nest.png')
    ...:
```

Conversion between NESTED and RING schemes





Conversion between NESTED and RING schemes

How to use:

NEST $\Rightarrow$ RING

```
In [36]: # Nside = 4
    ...: print('pix_nest =', pix_nest)
    ...: pix_ring = hp.nest2ring(Nside, pix_nest)
    ...: print('pix_ring =', pix_ring)
    ...:
pix_nest = 9
pix_ring = 25

In [37]: all_pix_ring = hp.nest2ring(Nside, all_pix_nest)
    ...: hp.mollview(all_pix_ring, title='Ring scheme',
unit='Pix index')
    ...: pyplot.savefig('mollweide_view_ring.png')
    ...:
```

Conversion between NESTED and RING schemes

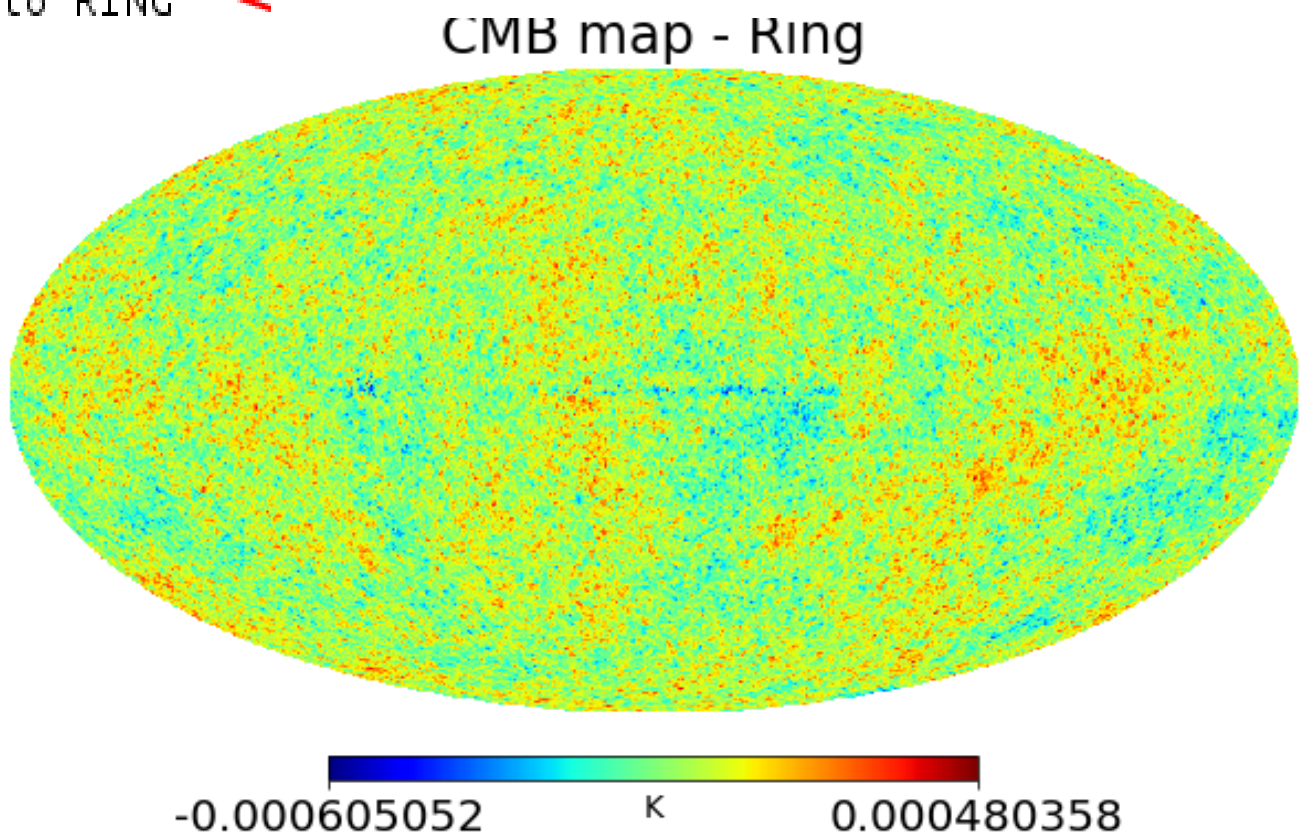
healpy.pixelfunc.reorder

healpy.pixelfunc.reorder(*map_in*, **args*, ***kwds*)

NEST $\rightleftharpoons$ RING

Conversion between NESTED and RING schemes

```
In [50]: mapa_ring = hp.read_map('COM_CMB_IQU-smica_1024_R2.02_full.fits')
    ...: hp.mollview(mapa_ring, title='CMB map - Ring', unit='K')
    ...:
NSIDE = 1024
ORDERING = NESTED in fits file <---
INDXSCHM = IMPLICIT
/home/camila/anaconda3_4p3p1/lib/python3.6/site-packages/healpy/fitsfunc.py:
339: UserWarning: No INDXSCHM keyword in header file : assume IMPLICIT
  "assume {}".format(schm))
Ordering converted to RING <---
```



Conversion between NESTED and RING schemes

How to use:

$$\text{NEST} \rightleftharpoons \text{RING}$$

```
In [52]: mapa_nest = hp.reorder(mapa_ring, inp='RING', out='NEST')

In [53]: # OR:::
    ...:
    ...: mapa_nest = hp.reorder(mapa_ring, r2n=True)
```

Exercise:

- Read the map in ring ordering.
- Reorder it according to the nested scheme.
- Visualize it.
- Reorder back to ring, and visualize it.

N_{side} / N_{pix} / Resolution

`nside2npix` (nside) <---	Give the number of pixels for the given nside.
`npix2nside` (npix) <---	Give the nside parameter for the given number of pixels.
`nside2order` (nside)	Give the resolution order for a given nside.
`order2nside` (order)	Give the nside parameter for the given resolution order.
`nside2resol` (nside[, arcmin]) <---	Give approximate resolution (pixel size in radian or arcmin) for nside.
`nside2pixarea` (nside[, degrees]) <---	Give pixel area given nside in square radians or square degrees.
`max_pixrad` (nside)	Maximum angular distance between any pixel center and its corners
`isnsideok` (nside)	Returns `True` if nside is a valid nside parameter, `False` otherwise.
`isnpixok` (npix)	Return `True` if npix is a valid value for healpix map size, `False` otherwi
`get_map_size` (m)	Returns the npix of a given map (implicit or explicit pixelization).
`get_min_valid_nside` (npix)	Returns the minimum acceptable nside so that npix <= nside2npix(nside)
`get_nside` (m)	Return the nside of the given map.
`maptype` (m)	Describe the type of the map (valid, single, sequence of maps).
`ud_grade` (map_in, *args, **kwds)	Upgrade or degrade resolution of a map (or list of maps).

N_{side} / N_{pix} / Resolution

$$N_{side} \rightleftharpoons N_{pix}$$

```
In [72]: Nside1 = 64 # = 1, 2, 4, 8, 16, 64, ...
    ...: Npix1  = 12*Nside**2 # 12 x Nside²
    ...: print('(1)', Nside1, '-->', Npix1)
    ...:
    ...: # OR:::
    ...:
    ...: Npix2  = hp.nside2npix(Nside1) # 12 x Nside²
    ...: print('(2)', Nside1, '-->', Npix2)
    ...:
    ...: Nside2  = hp.npix2nside(Npix2) # 12 x Nside²
    ...: print('(3)', Nside2, '<--', Npix2)
    ...:
(1) 64 --> 49152
(2) 64 --> 49152
(3) 64 <-- 49152
```

Exercise:

- $N_{side} = 4, 129, 1024.$

N_{side} / N_{pix} / Resolution

$N_{side} \rightarrow$ Resolution

```
In [81]: Nside = 4 # = 1, 2, 4, 8, 16, 64, ...
    ...: Resolution  = hp.nside2resol(Nside)
    ...: print('(1)', Nside, '-->', Resolution, 'radians')
    ...:
(1) 4 --> 0.255831676987 radians

In [82]: Resolution  = hp.nside2resol(Nside, arcmin=True)
    ...: print('(2)', Nside, '-->', Resolution, 'arcmin =', Resolution/60., 'deg')
    ...:
(2) 4 --> 879.484521425 arcmin = 14.6580753571 deg
```

$N_{side} \rightarrow$ Pix Area

```
In [85]: Area  = hp.nside2pixarea(Nside, degrees=True)
    ...: print('(3)', Nside, '-->', Area, 'square deg')
    ...:
(3) 4 --> 214.859173174 square deg
```

Exercise:

- $N_{side} = 2048$ (Planck).

N_{side} / N_{pix} / Resolution

`nside2npix` (nside) <---	Give the number of pixels for the given nside.
`npix2nside` (npix) <---	Give the nside parameter for the given number of pixels.
`nside2order` (nside)	Give the resolution order for a given nside.
`order2nside` (order)	Give the nside parameter for the given resolution order.
`nside2resol` (nside[, arcmin]) <---	Give approximate resolution (pixel size in radian or arcmin) for nside.
`nside2pixarea` (nside[, degrees]) <---	Give pixel area given nside in square radians or square degrees.
`max_pixrad` (nside)	Maximum angular distance between any pixel center and its corners
`isnsideok` (nside)	Returns `True` if nside is a valid nside parameter, `False` otherwise.
`isnpixok` (npix)	Return `True` if npix is a valid value for healpix map size, `False` otherwi
`get_map_size` (m) *	Returns the npix of a given map (implicit or explicit pixelization).
`get_min_valid_nside` (npix)	Returns the minimum acceptable nside so that npix <= nside2npix(nside)
`get_nside` (m) *	Return the nside of the given map.
`maptype` (m)	Describe the type of the map (valid, single, sequence of maps).
`ud_grade` (map_in, *args, **kwds) ***	Upgrade or degrade resolution of a map (or list of maps).

N_{side} / N_{pix} / Resolution

Upgrade or degrade resolution of a map (or list of maps).

`args = nside_out, order_in, order_out, ...`

N_{side} / N_{pix} / Resolution

How to use:

$$N_{side} = 1024 \rightarrow N_{side} = 64$$

```
In [103]: mapa = hp.read_map('COM_CMB_IQU-smica_1024_R2.02_full.fits')
NSIDE = 1024
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
/home/camila/anaconda3_4p3p1/lib/python3.6/site-packages/healpy/fitsfunc.py:339:
UserWarning: No INDXSCHM keyword in header file : assume IMPLICIT
  "assume {}".format(schm))
Ordering converted to RING

In [104]: mapa_deg = hp.ud_grade(mapa, 64, order_in = 'RING', order_out='NEST')
     ...:
     ...: hp.mollview(mapa_deg, title='CMB map - Nside=64', unit='K', nest=True)
     ...:
```

N_{side} / N_{pix} / Resolution

Exercise:

- Repeat the degrading process using Nside_out = 16 and order_out = 'RING'.
- Degrade the mask[1] to Nside=128 and to Nside=2048.
- Visualize the 3 cases.

[1] The fits file with the CMB map has one of its column corresponding to the mask. Use h=True to find it.