How to: HEALPix (healpy)

Edgar Martin Salazar

September, 2017

Our goal is to make a plot of a catalog using a tool called **HEALPix**. HEALPix is an acronym for Hierarchical Equal Area isoLatitude Pixelization of a sphere. This tool produces a subdivision of a spherical surface into $12 \cdot N_{\text{side}}^2$ pixels of the same surface area. These act like bins in which we sort data from a measurement that belongs to that location in the sky. HEALPix takes twelve pixels of equal area and as one increases the number of divisions in a side, namely N_{side}, it increases the number of pixels by a factor N_{side}^2 in each pixel. Increasing the number of pixels enhances the resolution of the map.

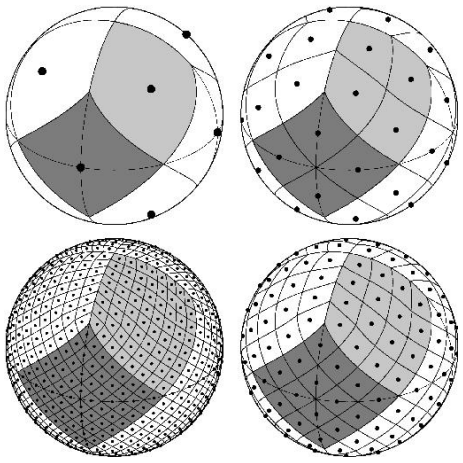


Figure 1: Diagram showing how HEALPix divides a sphere according to N_{side}^2.

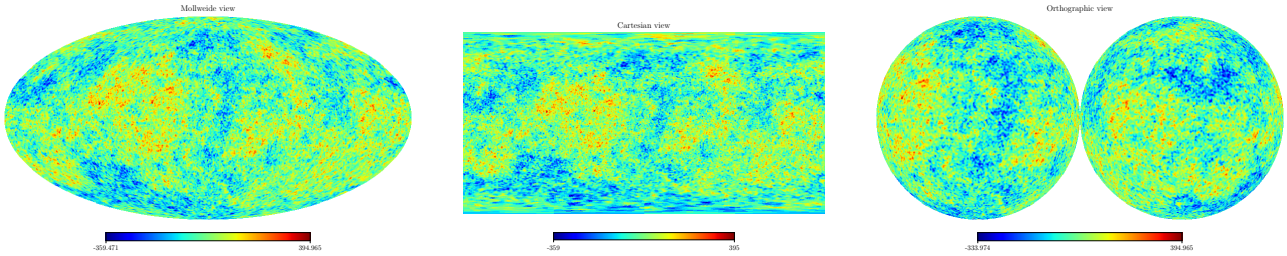
N_{side} is the factor by which the side of each of the 12 pixels is divided. Sets the resolution of the map and it's value must be some power of 2^n. This is probably the most important parameter as it defines the resolution of the map.

N_{pix} is the number of pixels that make up the map. It is related to N_{side} by the next expression:

$$N_{\text{pix}} = 12 \cdot (N_{\text{side}})^2 \tag{1}$$

RA is the right ascension coordinate of an object. It's equivalent to the longitude coordinate along the equatorial coordinate system and it has values from $[0, 2\pi]$.

DEC is the declination coordinate of an object in the sky. It's equivalent to the latitude coordinate in the equatorial coordinate and it has values from $[0, \pm\pi/2]$.



(a) Mollweide (b) Cartesian (c) Orthographic

Figure 2: Map projections.

HEALPix can plot different projections of a map as shown in Figure 2 but the most used is Mollwiede. To understand better how to *feed* HEALPix with a catalog let's follow the next example. Say we have a list of 20 elements and that somehow we can take those and sort them into bins obtaining a list of same length but with the bin number in which it is going to be placed. This list can be found in catalogs as `HPIX` and it is pretty useful when it comes to plotting catalogs.

$$\texttt{HPIX} = \{\ 3\ 2\ 8\ 4\ 4\ 3\ 5\ 1\ 5\ 4\ 7\ 9\ 2\ 4\ 5\ 1\ 1\ 3\ 6\ 8\ \}$$

Consider the simplest map we can make, that would be with a $N_{\text{side}} = 1$, giving the 12 base pixels covering the sphere. Obviously, `healpy`[1] won't be able to place the items in the correct bins because it needs a list of N_{pix} elements, in which each value is representing the number of elements in a given bin. We can use the next function to obtain such list.

```
def Bin_Data(Pixels,NPIX):
    BinnedPixels = np.zeros(NPIX, dtype=int)
    for i in Pixels:
        BinnedPixels[i] = BinnedPixels[i] + 1
    return BinnedPixels
```

Listing 1: Function for generating count maps (np stands for numpy).

This function takes a value `i` from the list `Pixels`, seeks that location in `BinnedPixels`, and adds +1 to the previous value in that position. After running the function `Binned_Data` one can notice that it stores the number of times a given pixel[2] is repeated in the data coordinates. The final list is

$$\texttt{BinnedPixels} = \{\ 0\ 3\ 2\ 3\ 4\ 3\ 1\ 1\ 2\ 1\ 0\ 0\ \}$$

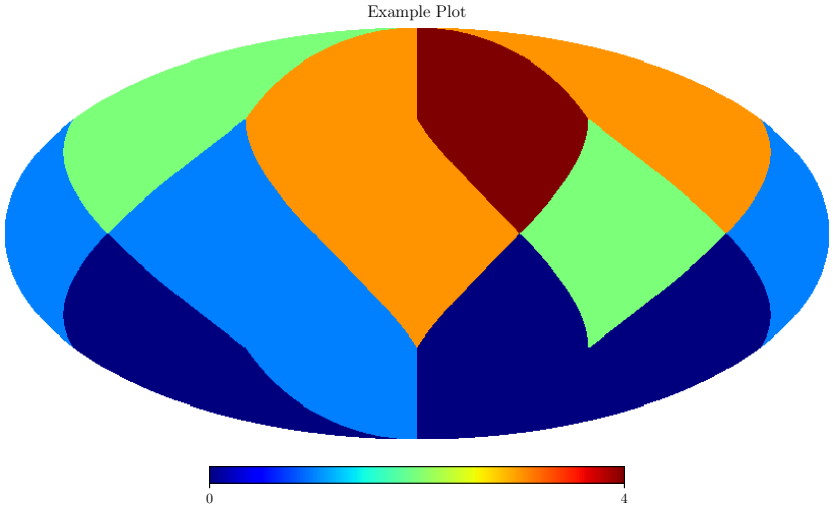
Of course, if we sum up the values of the list we would get the total number of elements in our data. We can now map this list using the function

```
    hp.mollview(map, title=, unit= , max=, min=)
```

Listing 2: Basic plot (hp stands for healpy)

[1] **healpy** is the HEALPix package for Python and is the one used in this project.
[2] Remember Python starts numbering from 0 rather than 1.

And get the following plot:



This is an awful plot given the resolution, but it gives the idea of how to work things around before giving the data to `healpy`.

There are some `healpy` functions that are usefull:

- Takes the RA and DEC coordinates of each object in the catalog, and converts them to a pixel location with a given N_{side}.

  ```
  hp.pixelfunc.ang2pix(NSIDE, RAS, DECS, lonlat=True)
  ```

  If the coordinates are in logitude and latitude it must have `lonlat=True`, else (coordinates in rads) it can be ommited.

- HEALPix has a function that returns the area per pixel given the number of pixels.

  ```
  hp.pixelfunc.nside2pixarea(NSIDE, degrees=True)
  ```

- HEALPix has two ways of ordering pixels: ring or nested. Ring ordering sorts pixels in sets of isolatitude pixels. This means that it stores pixels in *rings*. Nested ordering stores pixels in the order they are inputed. There is a `healpy` function that can easily change between ordering.

  ```
  hp.pixelfunc.reorder(old_map, inp='NESTED', out='RING')
  ```

  Or

  ```
  hp.pixelfunc.nest2ring(npix, ipix)
  ```

  As well as

  ```
  hp.pixelfunc.ring2nest(npix, ipix)
  ```

- For plotting we only use the command shown in Listings 2. We can add graticles and of course change the title, units, etc.

To know more about functions visit the website for healpy.