hpproj Documentation

Release 0.6.1

Alexandre Beelen, Marian Douspis

Nov 22, 2017

Contents

HealPixProjection is a project to allow easy and efficient projection of healpix maps onto planar grids. It can be used as a standalone program *hpproj.cutsky()*

```
$ cutsky 0.0 0.0 --mapfilenames HFI_SkyMap_857_2048_R2.00_full.fits
```

or as a python module

```
from hpproj import cutsky
result = cutsky([0.0, 0.0], maps={'HFI 857':
                                  {'filename': 'HFI_SkyMap_857_2048_R2.00_full.fits'}
                                  } )
```

CHAPTER 1

Features

- Galactic and equatorial system supported
- All projection system from `wcs`
- Project several healpix maps at once, efficiently !
- Output in `fits`, `png` or `votable` for the central point source photometry

See usage for more information on how to use `cutsky`

CHAPTER 2

Installation

Install `hpproj` using pip :

```
$ pip install hpproj
```

or by running setuptools on source. For more information see the installation page.

CHAPTER 3

Contribute

- Issues Tracker
- Source Code

CHAPTER 4

Support

If you are having issues, please let us know.

CHAPTER 5

License

This project is licensed under the LGPL+3.0 license.

CHAPTER 6

Contents:

6.1 Installation

`hpproj` is tested against python 2.7 and 3.5 and can be installed using *pip* or from *source*

6.1.1 pip

This will install the latest release of `hpproj`

6.1.2 source

```
$ git clone https://git.ias.u-psud.fr/abeelen/hpproj.git
$ cd hpproj
$ python setup.py install
```

This will install the master tree of `hpproj`. It is probably wiser to checkout a specific version before installation

```
$ git clone https://git.ias.u-psud.fr/abeelen/hpproj.git
$ cd hpproj
$ git checkout 0.4
$ python setup.py install
```

6.1.3 Dependencies

`hpproj` require the following librairies

- numpy>=1.11
- matplotlib>=1.5
- astropy>=1.2

- healpy>=1.9
- photutils>=0.2
- wcsaxes>=0.9

The specific versionning are those you are being used in the test suit. Both *pip* and *source* install should install those library if they are missing.

6.2 Usage

There is two main way to use `hpproj`, the first way is to use the standalone program on the command line, this will efficiently produce cuts for similar maps, or use it programmatically from within a python script or program which will offer an additional speed-up on high memory system.

6.2.1 From the command line - `cutsky`

The command line program is called *cutsky* and takes 3 arguments at minimum, the longitude and latitude of the desired projection (by default in `galactic` coordinate, but see below) and a list of healpix map to cut from :

```
$ cutsky 0.0 0.0 --mapfilenames data/HFI_SkyMap_100_2048_R2.00_full.fits  data/HFI_
↪SkyMap_857_2048_R2.00_full.fits
```

by default this will produce two `png` files centered on galactic longitude and latitude (0,0). Fits images of central photometries can be obtain using the `--fits` or `--phot` options. Help on `cutsky` can be obtain by

```
$ cutsky -h

usage: cutsky [-h] [--npix NPIX | --radius RADIUS] [--pixsize PIXSIZE]
              [--coordframe {galactic,fk5}]
              [--ctype {AZP,SZP,TAN,STG,SIN,ARC,ZPN,ZEA,AIR,CYP,CEA,CAR,MER,COP,COE,
↪COD,COO,SFL,PAR,MOL,AIT,BON,PCO,TSC,CSC,QSC,HPX,XPH}]
              [--mapfilenames MAPFILENAMES [MAPFILENAMES ...]] [--fits]
              [--png] [--votable] [--outdir OUTDIR] [-v | -q] [--conf CONF]
              lon lat

Reproject the spherical sky onto a plane.

positional arguments:
  lon                   longitude of the projection [deg]
  lat                   latitude of the projection [deg]

optional arguments:
  -h, --help            show this help message and exit
  --npix NPIX           number of pixels (default 256)
  --radius RADIUS       radius of the requested region [deg]
  --pixsize PIXSIZE     pixel size [arcmin] (default 1)
  --coordframe {galactic,fk5}
                        coordinate frame of the lon. and lat. of the
                        projection and the projected map (default: galactic)
  --ctype {AZP,SZP,TAN,STG,SIN,ARC,ZPN,ZEA,AIR,CYP,CEA,CAR,MER,COP,COE,COD,COO,SFL,
↪PAR,MOL,AIT,BON,PCO,TSC,CSC,QSC,HPX,XPH}
                        any projection code supported by wcslib (default:TAN)

input maps:
  one of the two options must be present
```

```
  --mapfilenames MAPFILENAMES [MAPFILENAMES ...]
                        absolute path to the healpix maps
  --conf CONF           absolute path to a config file

output:
  --fits                output fits file
  --png                 output png file (Default: True if nothing else)
  --votable             output votable file
  --outdir OUTDIR       output directory (default:.)

general:
  -v, --verbose         verbose mode
  -q, --quiet           quiet mode
```

It takes two float arguments, the latitude and longitude center of the requested projection, either in galactic or equatorial coordinate frame (controled by the `--coordframe` option) and a list of healpix maps, either on the command line with the `--mapfilenames` argument or describe in a config file (with the `--conf` option). Several other optional arguments can also be set like `--npix` the number of pixels, their size (`--pixsize`) or the projection type `--ctype`.

The cutted maps can be saved as fits (`--fits`) or png (`--png`) and central circular aperture photometry can be performed and saved as a votable (`--votable`). The output products directory can be tune using the `--outdir` option. All theses options can also be provided by the config file.

The config file follows a simple ini syntax with a global section `[cutsky]` to gather all previous options. The rest of the sections is used to describe the healpix maps used by `cutsky`. The section name `[test]` will be used as a legend and index by `cutsky`.

```
[cutsky]
npix = 256
pixsize = 2
coordframe = galactic
png = True

[SMICA]
filename = hpproj/data/CMB_I_SMICA_128_R2.00.fits
doCut = False

[HFI 100]
filename = hpproj/data/HFI_SkyMap_100_128_R2.00_RING.fits

[HFI 857]
filename = hpproj/data/HFI_SkyMap_857_128_R2.00_NESTED.fits
doCut = True
doContour = True
```

6.2.2 As a function call - `cutsky()`

It is also possible to call `cutsky` from a python program or script, as a function :

```
from hpproj import cutsky
result = cutsky([0.0, 0.0], maps=[ ( 'data/HFI_SkyMap_100_2048_R2.00_full.fits', {
↪'legend': 'HFI 100'} ),
                                   ( 'data/HFI_SkyMap_857_2048_R2.00_full.fits', {
↪'legend': 'HFI 857', 'doContour': True} ) ] )
```

The first argument is the latitude and longitude of the requested maps, by default in `galactic` frame (see the `coordframe` keyword), and the `maps` list define the healpix maps.

This will produce a list of dictionnaries containing 4 keys:

- `legend`,
- `fits` an ~astropy.io.fits.ImageHDU,
- `png`, a b61encoded png image of the fits
- `phot`, the corresponding photometry

Additionnal parameters can by passed to the function :

- `patch=[256,1]` : the size of the patch in pixel, and the size of the pixels in arcmin
- `ctype='TAN'` : the desired type of projection

6.2.3 As an object - `CutSky`

It is however more efficient to use cutsky as an object :

```
from hpproj import CutSky
cutsky = CutSky([ ( 'hpproj/data/HFI_SkyMap_100_2048_R2.00_full.fits', {'legend':
↪'HFI 100'} ),
                  ( 'hpproj/data/HFI_SkyMap_857_2048_R2.00_full.fits', {'legend':
↪'HFI 857', doContour: True} ) ], low_mem=False)

lonlat = [0.0,0.0]
result = cutsky.cut_fits(lonlat) # Will only produce the 'fits' key
result = cutsky.cut_png(lonlat)  # Will only produce the 'png' key (and 'fits' if␣
↪absent)
result = cutsky.cut_phot(lonlat) # Will only produce the 'phot' key (and fits' if␣
↪absent)
```

The result product should be similar to the `cutsky()` function. However with the `low_mem` keyword the healpix maps will be read only once in memory, for all `cut_*` calls. Similar to `cutsky()` several keyword parameters can be passed to `CutSky()` :

- `npix=256` : the size of the patch in pixels
- `pixsize=1` : the size of the pixels in arcmin
- `ctype='TAN'`: the desired type of projection

6.2.4 Limitations

All the healpix maps *must* have a proper header defining their :

- frame using the `COORDSYS` keyword,
- order using the `ORDERING` keyword.

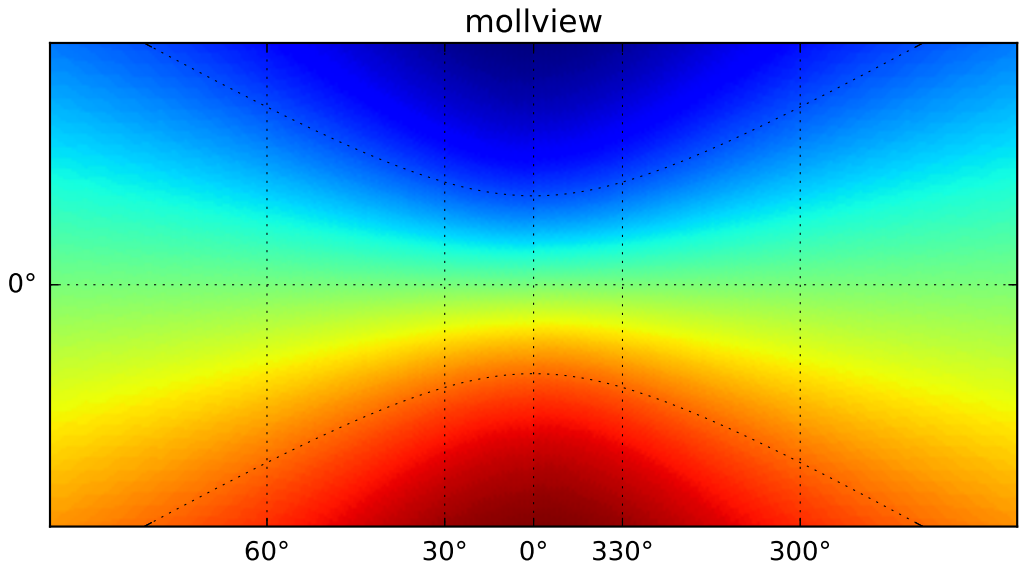
6.3 Visualization

The HealPixProjection routines can easily be used to display a full sky map with different projections. In the *`hpproj.visu`* module, several projection have been implemented

```
import matplotlib.pyplot as plt
import numpy as np
import healpy as hp
from astropy.wcs import WCS
from hpproj import mollview

# Ring like healpix map
nside = 2**6
hp_map = np.arange(hp.nside2npix(nside))
hp_header = {'NSIDE': nside,
             'ORDERING': 'RING',
             'COORDSYS': 'G'}

# Projection of the map and plotting
_ = mollview(hp_map, hp_header)
fig = plt.figure()
ax = fig.add_subplot(1,1,1, projection=WCS(_.header))
ax.imshow(_.data, origin='lower', interpolation='none')
ax.grid()
ax.set_title('mollview')
```



Note that these maps have a proper `WCS` header and thus can be easily used to overplot markers and artists. Different type of projections have been implementde

6.4 API

6.4.1 cutsky

cutsky module, mainly use *hpproj.hp_helper* functions

class hpproj.cutsky.**CutSky**(*maps=None*, *npix=256*, *pixsize=1*, *ctype='TAN'*, *low_mem=True*)
Container for Healpix maps and cut_* methods

...

Attributes

npix	(int) the number of pixels for the square maps
pixsize	(float) the size of the pixels [arcmin]
ctype	(str) a valid projection type (default : TAN)
maps	(dictonnary) a grouped dictionnary of gen_hpmap tuples (filename, map, header) (see :func:~init)

Methods

cut(*cut_type*, ***kwargs*)
helper function to cut into the maps

Parameters **cut_type** : str (fits|png|phot|votable)

define what to cut_type

lonlat : array of 2 floats

the longitude and latitude of the center of projection [deg]

coordframe : str

the coordinate frame used for the position AND the projection

maps_selection : list

optionnal list of the 'legend' or filename of the map to select a sub-sample of them.

Returns list of dictionnaries

the dictionnary output depends on cut_type

cut_fits(*lonlat=None*, *coordframe='galactic'*, *maps_selection=None*)
Efficiently cut the healpix maps and return cutted fits file with proper header

Parameters **lonlat** : array of 2 floats

the longitude and latitude of the center of projection [deg]

coordframe : str

the coordinate frame used for the position AND the projection

maps_selection : list

optionnal list of the 'legend' or filename of the map to select a sub-sample of them.

Returns list of dictionnaries

the dictionnary has 2 keys : * 'legend' (the opts{'legend'} see __init()) * 'fits' an `ImageHDU`

cut_phot (*lonlat=None*, *coordframe='galactic'*, *maps_selection=None*)
Efficiently cut the healpix maps and return cutted fits file with proper header and corresponding photometry

Parameters **lonlat** : array of 2 floats

the longitude and latitude of the center of projection [deg]

coordframe : str

the coordinate frame used for the position AND the projection

maps_selection : list

optionnal list of the 'legend' or filename of the map to select a sub-sample of them.

Returns list of dictionnaries

the dictionnary has 3 keys : * 'legend' (the opts{'legend'} see __init()), * 'fits' an `ImageHDU`, * 'phot', the corresponding photometry

cut_png (*lonlat=None*, *coordframe='galactic'*, *maps_selection=None*)
Efficiently cut the healpix maps and return cutted fits file with proper header and corresponding png

Parameters **lonlat** : array of 2 floats

the longitude and latitude of the center of projection [deg]

coordframe : str

the coordinate frame used for the position AND the projection

maps_selection : list

optionnal list of the 'legend' or filename of the map to select a sub-sample of them.

Returns list of dictionnaries

the dictionnary has 3 keys : * 'legend' (the opts{'legend'} see __init()), * 'fits' an `ImageHDU`, * 'png', a b61encoded png image of the fits

`hpproj.cutsky.`**cutsky** (*lonlat=None*, *maps=None*, *patch=None*, *coordframe='galactic'*, *ctype='TAN'*)
Old interface to cutsky – Here mostly for compability

Parameters **lonlat** : array of 2 floats

the longitude and latitude of the center of projection [deg]

maps: a dict or a list

either a dictionnary (old interface) or a list of tuple (new interface) : ``` {legend: {'filename': full_filename_to_healpix_map.fits,

'doContour': True }, # optionnal

... } ` or ` [(full_filename_to_healpix_map.fits, {'legend': legend,

'doContour': True}), # optionnal

...] ```

patch : array of [int, float] proj_type.upper()

if proj_type not in VALID_PROJ:

raise ValueError('Unsupported projection') [int] the number of pixels and [float] the size of the pixel [arcmin]

coordframe : str

the coordinate frame used for the position AND the projection

ctype: str

a valid projection type (default: TAN)

Returns list of dictionnaries

the dictionnary has 4 keys : * 'legend' (see maps above), * 'fits' an `ImageHDU`, * 'png', a b61encoded png image of the fits * 'phot', the corresponding photometry

`hpproj.cutsky.`**`main`**(*argv=None*)
The main routine.

`hpproj.cutsky.`**`save_result`**(*output*, *result*)
Save the results of the main function

`hpproj.cutsky.`**`to_new_maps`**(*maps*)
Transform old dictionnary type healpix map list used by cutsky to list of tuple used by Cutsky

Parameters **maps** : dict

a dictionnary with key being the legend of the image : ``` {legend: {'filename': full_filename_to_healpix_map.fits,

'doContour': True },

... }

```

**Returns** a list of tuple following the new convention:

```
[(full_filename_to_healpix_map.fits, {'legend': legend,

'doContour': True}),

...]
```

### 6.4.2 hp_helper

Series of helper function to deal with healpix maps

`hpproj.hp_helper.`**`build_wcs`**(*\*args*, *\*\*kwargs*)
Construct a `WCS` object for a 2D image Parameters ———- coord : `astropy.coordinate.SkyCoord`

the sky coordinate of the center of the projection

or

**lon,lat** [floats] the sky coordinates of the center of projection and

**src_frame** [str, ('GALACTIC', 'EQUATORIAL')] the coordinate system of the longitude and latitude

pixsize : float size of the pixel (in degree)

**shape_out** [tuple] shape of the output map (n_y,n_x)

**npix** [int] number of pixels in the final square map, the reference pixel will be at the center, superseed shape_out
```

proj_sys [str ('GALACTIC', 'EQUATORIAL')] the coordinate system of the plate (from HEALPIX maps....)

proj_type [str ('TAN', 'SIN', 'GSL', ...)] the projection system to use

Returns WCS: `WCS`

An corresponding wcs object

Notes

You can access a function using only catalogs with the ._coord() method

`hpproj.hp_helper.`**`build_wcs_cube`**(**args*, ***kwargs*)

Construct a `WCS` object for a 3D cube, where the 3rd dimension is an index Parameters ———- coord : `astropy.coordinate.SkyCoord`

the sky coordinate of the center of the projection

or

lon,lat [floats] the sky coordinates of the center of projection and

src_frame [str, ('GALACTIC', 'EQUATORIAL')] the coordinate system of the longitude and latitude

index : int reference index

pixsize [float] size of the pixel (in degree)

shape_out [tuple] shape of the output map (n_y, n_x)

npix [int] number of pixels in the final map, the reference pixel will be at the center, override shape_out

proj_sys [str ('GALACTIC', 'EQUATORIAL')] the coordinate system of the plate (from HEALPIX maps....)

proj_type [str ('TAN', 'SIN', 'GSL', ...)] the projection system to use

Returns WCS: `WCS`

An corresponding wcs object

Notes

You can access a function using only catalogs with the ._coord() method

`hpproj.hp_helper.`**`build_wcs_2pts`**(*coords*, *pixsize=None*, *shape_out=(512, 512)*, *npix=None*, *proj_sys='EQUATORIAL'*, *proj_type='TAN'*, *relative_pos=(0.4, 0.6)*)

Construct a `WCS` object for a 2D image

Parameters **coords** : class:*astropy.coordinate.SkyCoord*

the 2 sky coordinates of the projection, they will be horizontal in the resulting wcs

pixsize : float

size of the pixel (in degree) (default: None, use relative_pos and shape_out)

shape_out : tuple

shape of the output map (n_y,n_x)

npix : int

number of pixels in the final square map, superseed shape_out

coordsys : str ('GALACTIC', 'EQUATORIAL')

the coordinate system of the plate (from HEALPIX maps....) will be rotated anyway

proj_type : str ('TAN', 'SIN', 'GSL', ...)

the projection system to use, the first coordinate will be the projection center

relative_pos : tuple

the relative position of the 2 sources along the x direction [0-1] (will be computed if pixsize is given)

Returns WCS: `WCS`

An corresponding wcs object

Notes

By default relative_pos is used to place the sources, and the pixsize is derived, but if you define pixsize, then the relative_pos will be computed and the sources placed at the center of the image

`hpproj.hp_helper.`**`build_ctype`**(*coordsys*, *proj_type*)

Build a valid spatial ctype for a wcs header

Parameters **coordsys** : str ('GALATIC', 'EQUATORIAL')

the coordinate system of the plate

proj_type: str ('TAN', 'SIN', 'GSL', ...)

any projection system supported by WCS

Returns list:

a list with the 2 corresponding spatial ctype

`hpproj.hp_helper.`**`hp_is_nest`**(*hp_header*)

Return True if the healpix header is in nested

Parameters **hp_header** : `Header`

the header 100

———-

bool :

True if the header is nested

`hpproj.hp_helper.`**`hp_celestial`**(*hp_header*)

Retrieve the celestial system used in healpix maps. From Healpix documentation this can have 3 forms :

- 'EQ', 'C' or 'Q' : Celestial2000 = eQuatorial,
- 'G' : Galactic
- 'E' : Ecliptic,

only Celestial and Galactic are supported right now as the Ecliptic coordinate system was just recently pulled to astropy

Similar to `wcs_to_celestial_frame` but for header from healpix maps

Parameters **hp_header** : `Header`

the header of the healpix map

Returns **frame** : `BaseCoordinateFrame` subclass instance

An instance of a `BaseCoordinateFrame` subclass instance that best matches the specified WCS.

`hpproj.hp_helper.`**`hp_to_wcs`**(**args*, ***kargs*)

Project an Healpix map on a wcs header, using nearest neighbors. Parameters ———- hp_hdu : *:class:astropy.io.fits.ImageHDU*

a pseudo ImageHDU with the healpix map and the associated header

or

hp_map [array_like] healpix map with corresponding

hp_header : `astropy.fits.header.Header` wcs : `astropy.wcs.WCS` wcs object to project with

shape_out [tuple] shape of the output map (n_y, n_x)

npix [int] number of pixels in the final square map, superseed shape_out

order [int (0|1)] order of the interpolation 0: nearest-neighbor, 1: bi-linear interpolation

Returns array_like

the projected map in a 2D array of shape shape_out

Notes

You can access a function using only catalogs with the ._coord() method

`hpproj.hp_helper.`**`hp_to_wcs_ipx`**(*hp_header*, *wcs*, *shape_out=(512*, *512)*, *npix=None*)

Return the indexes of pixels of a given wcs and shape_out, within a nside healpix map.

Parameters **hp_header** : `astropy.fits.header.Header`

header of the healpix map, should contain nside and coordsys and ordering

wcs : `astropy.wcs.WCS`

wcs object to project with

shape_out : tuple

shape of the output map (n_y, n_x)

npix : int

number of pixels in the final square map, superseed shape_out

Returns 2D array_like

mask for the given map

array_like

corresponding pixel indexes

Notes

The map could then easily be constructed using

proj_map = np.ma.array(np.zeros(shape_out), mask=~mask, fill_value=np.nan) proj_map[mask] = healpix_map[ipix]

hpproj.hp_helper.**hp_project**(**args*, ***kargs*)

Project an healpix map at a single given position Parameters ———- hp_hdu : *:class:astropy.io.fits.ImageHDU*

a pseudo ImageHDU with the healpix map and the associated header

or

hp_map [array_like] healpix map with corresponding

hp_header : `astropy.fits.header.Header` coord : `astropy.coordinate.SkyCoord` the sky coordinate of the center of the projection

pixsize [float] size of the pixel (in degree)

npix [int] number of pixels in the final map, the reference pixel will be at the center

order [int (0|1)] order of the interpolation 0: nearest-neighbor, 1: bi-linear interpolation

projection [tuple of str] the coordinate ('GALACTIC', 'EQUATORIAL') and projection ('TAN', 'SIN', 'GSL', ...) system

hdu [bool] return a `astropy.io.fits.PrimaryHDU` instead of just a ndarray

Returns `astropy.io.fits.PrimaryHDU`

containing the array and the corresponding header

Notes

You can access a function using only catalogs with the ._coord() method

hpproj.hp_helper.**gen_hpmap**(*maps*)

Generator function for large maps and low memory system

Parameters **maps** : list

A list of Nmap tuples with either:

- (filename, path_to_localfilename, healpix header)
- (filename, healpix vector, healpix header)

Returns tuple

Return a tuple (filename, healpix map, healpix header) corresponding to the inputed list

hpproj.hp_helper.**build_hpmap**(*filenames*, *low_mem=True*)

From a filename list, build a tuple usable with gen_hmap()

Parameters **filenames: list**

A list of Nmap filenames of healpix maps

low_mem : bool

On low memory system, do not read the maps themselves (default: only header)

Returns tuple list

A list of tuple which can be used by gen_hpmap

hpproj.hp_helper.**hpmap_key**(*hp_map*)
Generate an key from the hp_map tuple to sort the hp_maps by map properties

Parameters **hp_map: tuple**

A tuple from (buildlgen)_hpmap : (filename, healpix map, healpix header)

Returns str

A string with the map properties

hpproj.hp_helper.**equiv_celestial**(*frame*)
Return an equivalent ~astropy.coordfinates.builtin_frames

Notes

We do not care of the differences between ICRS/FK4/FK5

hpproj.hp_helper.**get_lonlat**(*coord*, *proj_sys*)
Retrieve the proper longitude and latitude

Parameters **coord** : `astropy.coordinate.SkyCoord`

the sky coordinate of the center of the projection

proj_sys : str ('GALACTIC', 'EQUATORIAL')

the coordinate system of the plate (from HEALPIX maps....)

Returns tuples of float

the longitude and latitude in the requested system

6.4.3 visu

Series of full sky visualization function, with proper wcs header

hpproj.visu.**view**(**args*, ***kargs*)
projection of the full sky Parameters ———- hp_hdu : *:class:astropy.io.fits.ImageHDU*

a pseudo ImageHDU with the healpix map and the associated header

or

hp_map [array_like] healpix map with corresponding

hp_header : `astropy.fits.header.Header` coord : `astropy.coordinate.SkyCoord`
the sky coordinate of the center of the projection

npix [int] number of pixels in the latitude direction

proj_sys [str, ('GALACTIC', 'EQUATORIAL')] the coordinate system of the projection

proj_type: str ('TAN', 'SIN', 'GSL', ...) any projection system supported by WCS

aspect [float] the resulting figure aspect ratio 1:aspect_ratio

pv [array_like] 2x2 PV matrix needed by some projection

Returns `astropy.io.fits.ImageHDU`

2D images with header

Notes

You can access a function using only catalogs with the ._coord() method

hpproj.visu.**orthview**(*hp_hdu*, *coord=None*, *npix=360*, *proj_sys='GALACTIC'*)

Slant orthographic projection of the full sky

Parameters **hp_hdu** : *:class:astropy.io.fits.ImageHDU*

a pseudo ImageHDU with the healpix map and the associated header to be projected

coord : astropy.coordinate.SkyCoord

the sky coordinate of the center of the projection

npix : int

number of pixels in the latitude direction

proj_sys : str, ('GALACTIC', 'EQUATORIAL')

the coordinate system of the projection

Returns astropy.io.fits.ImageHDU

2D images with header

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index