
hpproj Documentation

Release 0.9.1

Alexandre Beelen, Marian Douspis

Jan 30, 2019

Contents

1	Features	3
2	Installation	5
3	Contribute	7
4	Support	9
5	License	11
6	Contents:	13
6.1	Installation	13
6.2	Basic Usage	14
6.3	<i>n-d</i> Projections !	17
6.4	Visualization	23
6.5	API	29
7	Indices and tables	45
	Python Module Index	47

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 function

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

or as a python class, for optimization when producing several cuts

```
from hpproj import CutSky, to_coord
cutsky = CutSky({'Planck 857': {'filename': 'HFI_SkyMap_857_2048_R2.00_full.fits'}})
result = cutsky.cut_fits(to_coord([0., 0.]
```

Note: For science applications requiring high photometric accuracy, we recommend the drizzlib software developed by CADE, which uses a flux-conserving drizzling method to reproject data between HEALPix and local WCS. Drizzlib is available as standalone software (IDL python) here: <http://cade.irap.omp.eu/dokuwiki/doku.php?id=software> . An online interface, drizzweb, is available here: <http://drizzweb.irap.omp.eu/> .

You can also have a look into the `reproject.reproject_from_healpix()` function from the `reproject` package.

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`
- Perform *n-dim projections* !

See *Basic 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.

6.1 Installation

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

6.1.1 pip

```
$ pip install hproj
```

This will install the latest release of hproj

6.1.2 source

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

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

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

6.1.3 Dependencies

hproj require the following librairies

- numpy>=1.13

- matplotlib>=2.0
- astropy>=2.0
- healpy>=1.9
- photutils>=0.4

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 Basic Usage

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

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

However you can correct the headers in the construction of the list of maps

```
maps = [ {'HFI 100': {'filename': 'data/HFI_SkyMap_100_2048_R2.00_full.fits',
↳ 'COORDSYS': 'C'}}]
```

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 aperture [aperture ...]] [--outdir OUTDIR] [-v | -q]
↳ [--conf CONF]
              lon lat

Reproject the spherical sky onto a plane.

positional arguments:
  lon                    longitude of the projection [deg]
```

(continues on next page)

(continued from previous page)

```

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 aperture [aperture ...]
                    list of aperture [arcmin] to make circular aperture photometry
--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 (controlled 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 aperture`). 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 = hproproj/data/CMB_I_SMICA_128_R2.00.fits
doCut = False

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

```

(continues on next page)

(continued from previous page)

```
[HFI 857]
filename = hproj/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. You first need to define a list of maps on which to perform the cuts, as list of tuple with at minimum (`filename.fits`, `{'legend': "legend"}`) given the full path to the healpix map, and a dictionary with a least the key `legend`

```
from hproj import cutsky

maps = [('data/HFI_SkyMap_100_2048_R2.00_full.fits', {'legend': 'HFI 100', 'aperture
↳': [1, 2, 3]}),
        ('data/HFI_SkyMap_857_2048_R2.00_full.fits', {'legend': 'HFI 857', 'doContour
↳': True})]

result = cutsky([0.0, 0.0], maps=maps)
```

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 dictionaries containing 4 keys:

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

Additional parameters can be 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 hproj import CutSky, to_coord

maps = [('data/HFI_SkyMap_100_2048_R2.00_full.fits', {'legend': 'HFI 100', 'aperture
↳': [1, 2, 3]}),
        ('data/HFI_SkyMap_857_2048_R2.00_full.fits', {'legend': 'HFI 857', 'doContour
↳': True})]

cutsky = CutSky(maps, low_mem=False)

coord = to_coord([0.0, 0.0])
result = cutsky.cut_fits(coord) # Will only produce the 'fits' key
```

(continues on next page)

(continued from previous page)

```
result = cutsky.cut_png(coord) # Will only produce the 'png' key (and 'fits' if_
↪absent)
result = cutsky.cut_phot(coord) # 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 As internal calls - `hp_helper`

Alternatively if you simply want to get a projected array, you can use the `hp_project()` function

```
from astropy.io import fits
from astropy.coordinates import SkyCoord
import healpy as hp
from hproproj import hp_project

hp_data, hp_header = hp.read_map('data/HFI_SkyMap_100_2048_R2.00_full.fits', h=True)
hp_header = fits.Header(hp_header)

hdu = hp_project(hp_data, hp_header, SkyCoord(0, 0, unit='deg'))
```

Or, if you prefer to get full control, you can also use the internal functions like `build_wcs()` and `hp_to_wcs()`

```
from astropy.io import fits
import healpy as hp
import hproproj as hpp

hp_data, hp_header = hp.read_map('data/HFI_SkyMap_100_2048_R2.00_full.fits', h=True)
hp_header = fits.Header(hp_header)
hp_hdu = fits.ImageHDU(hp_data, hp_header)

w = hpp.build_wcs(0, 0)

proj_map = hpp.hp_to_wcs(hp_data, hp_header, w)
```

Note that both `hp_project` and `hp_to_wcs` accept either an `~astropy.io.fits.ImageHDU`, or both `hp_data`, `hp_header`

6.3 *n-d* Projections !

hproproj allow for *n-d* projections, for $d=3$ to $d=0$

Caution: `healpy` by default change any healpix map into the RING pixelization scheme without changing its header. Be sure to read the maps with the `nest=None`

First let's setup the dataset :

```
import numpy as np
import matplotlib.pyplot as plt
import healpy as hp

from astropy.io import fits
from astropy.coordinates import SkyCoord
from astropy.utils.data import download_file
from astropy.wcs import WCS
from astropy.table import Table

from hproj import hp_stack

# Retrieve the Planck 857 GHz all sky map
irsa_url = 'http://irsa.ipac.caltech.edu/data/Planck/release_2/all-sky-maps/maps/'
url = irsa_url + 'HFI_SkyMap_857_2048_R2.02_full.fits'
filename = download_file(url, cache=True)

hp_data, hp_header = hp.read_map(filename, h=True, nest=None)
hp_hdu = fits.ImageHDU(hp_data, fits.Header(hp_header))
hp_hdu.header['UNIT'] = 'MJy/sr'

# Fetch the PCCS catalog
cds_url = 'http://cdsarc.u-strasbg.fr/ftp/cats/J/A+A/594/A26/fits/'
url = cds_url + 'PCCS_857_R2.01.fits'

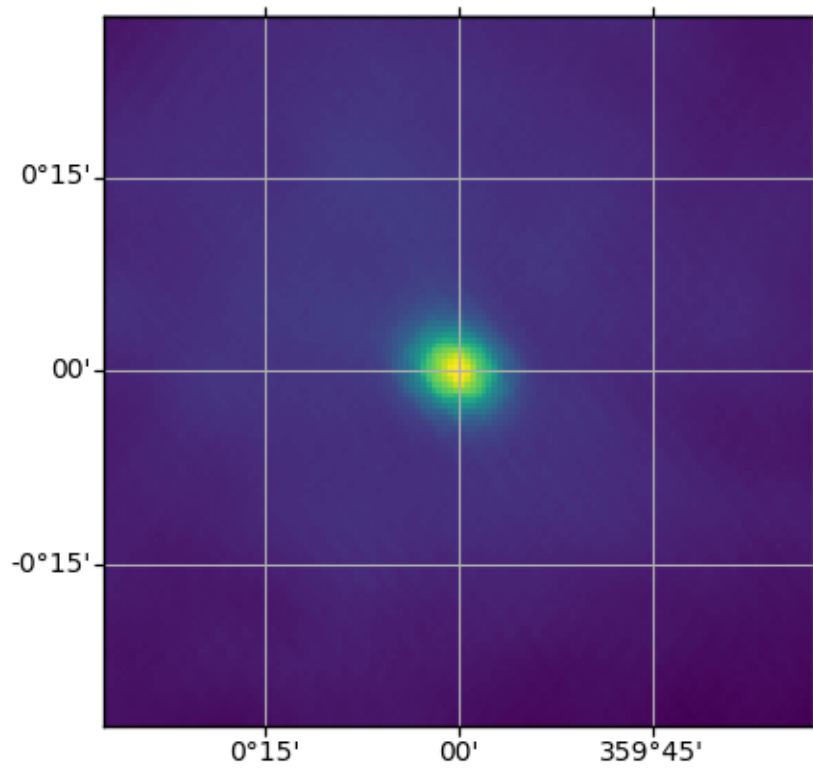
PCCS = Table.read(download_file(url, cache=True))
# Select a few sources
PCCS = PCCS[np.abs(PCCS['GLAT']) > 30]
PCCS = PCCS[:500]
```

6.3.1 3-d projections

hproj allow for stacking in the healpix map

```
coords = SkyCoord(PCCS['RA'].data, PCCS['DEC'].data, unit="deg")
pixsize = hp.nside2resol(hp_hdu.header['NSIDE'], arcmin=True) / 60 / 4
hdu = hp_stack(hp_hdu, coords, pixsize=pixsize, shape_out=(128, 128))
```

hdu is an `astropy.io.fits.ImageHDU` containing the stack of all the requested positions. One can also use the *keep* option to retrieve all the individual maps



6.3.2 2.5-d projections

Using some more in-depth routine of *hproj*, it is possible to place 2 sources at a relative given position on a map

```
from astropy.visualization import ImageNormalize, HistEqStretch

from hproj import build_wcs_2pts
from hproj import hp_to_wcs

coord_LMC = SkyCoord("05:23:34.60", "-69:45:22.0", unit=(u.hourangle, u.deg))
coord_SMC = SkyCoord("00h52m38s", "-72:48:01", unit=(u.hourangle, u.deg))

pair_coord = (coord_LMC, coord_SMC)

relative_pos = [0.3, 0.7]
shape_out = (512, 1024)

wcs = build_wcs_2pts(pair_coord, shape_out=shape_out, relative_pos=relative_pos)
img = hp_to_wcs(hp_hdu, wcs, shape_out)

norm = ImageNormalize(stretch=HistEqStretch(img))

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, projection=wcs)
```

(continues on next page)

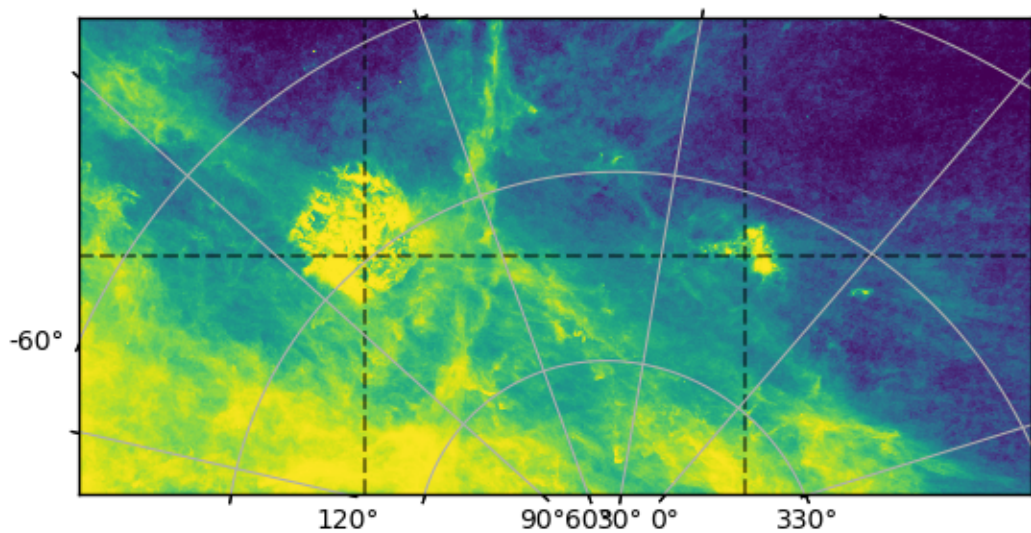
(continued from previous page)

```

ax.imshow(img, origin='lower', interpolation='none', norm=norm)
ax.grid()

ax.axhline(0.5 * img.shape[0], linestyle='--', c='k', alpha=0.5)
for pos in relative_pos:
    ax.axvline(pos * img.shape[1], linestyle='--', c='k', alpha=0.5)

```



6.3.3 2-d Projections

This is the most common projection, from an healpix map to a 2D map

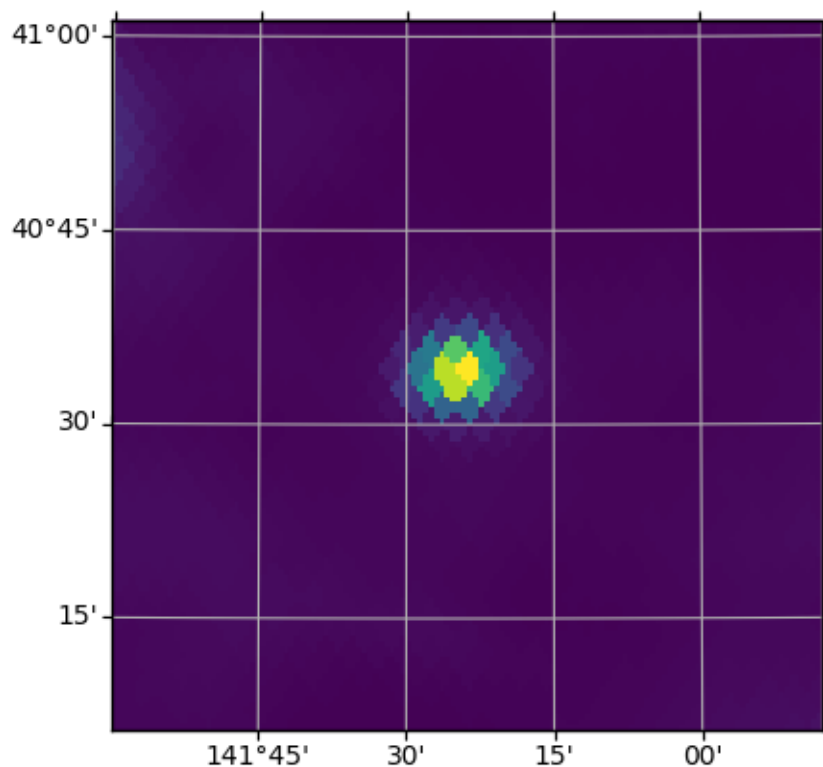
```

from hproy import hp_project

coord = SkyCoord(141.397513059, 40.5638050454, unit="deg", frame="galactic")
pixsize = hp.nside2resol(hp_hdu.header['NSIDE'], arcmin=True) / 60 / 4
hdu = hp_project(hp_hdu, coord, pixsize=pixsize, shape_out=(128, 128))

```

hdu is then an `astropy.io.fits.ImageHDU` containing the requested region on the sky and its corresponding header, which can be easily plotted with, for e.g., *matplotlib*



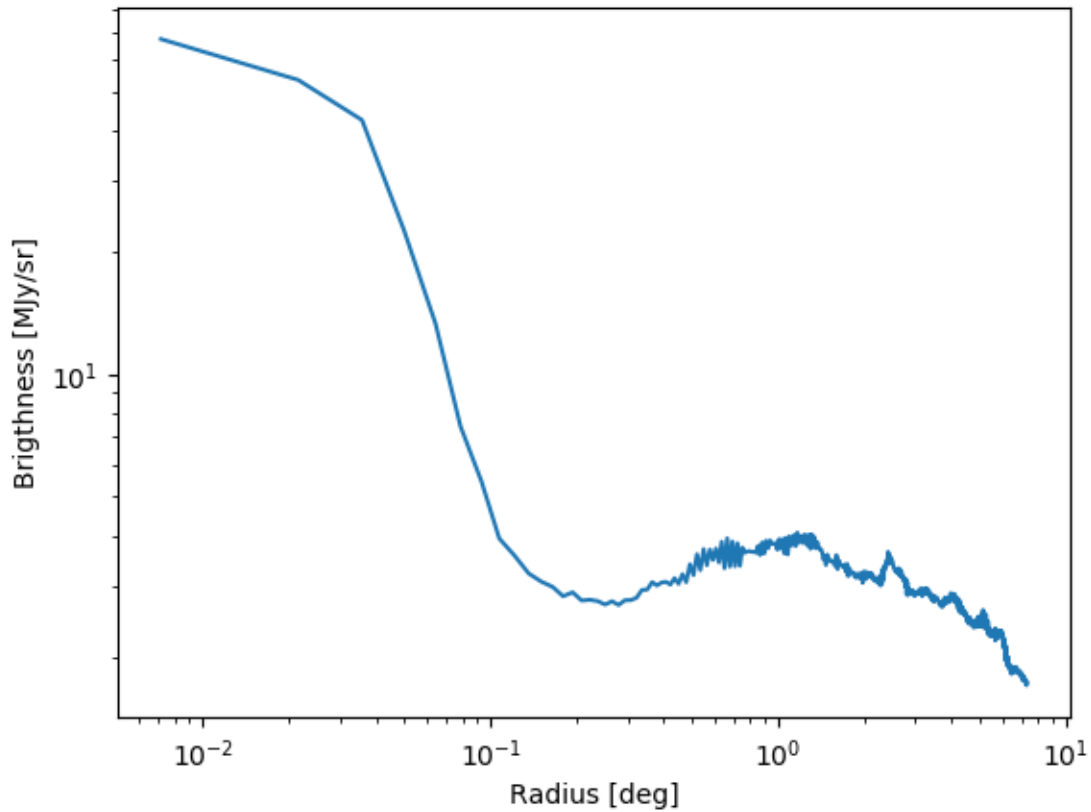
6.3.4 1-d Projections

The 1-d projection goes from an healpix map to a intensity profile

```
from hproproj import hp_profile

coord = SkyCoord(202.4865871680179, 47.181795866475426, unit="deg")
hdu = hp_profile(hp_hdu, coord)
```

hdu is then an `astropy.io.fits.ImageHDU` with the profile centered on the requested coordinates



6.3.5 0-d Projections

The 0-d projection goes from an healpix map to an aperture photometry, of a given position

```
from hproproj import hp_photometry
from astropy.coordinates import SkyCoord, Angle

coord = SkyCoord(202.4865871680179, 47.181795866475426, unit="deg")
apertures = Angle(hp.nside2resol(hp_hdu.header['NSIDE'], arcmin=True) / 60, "deg") *
↳ [7, 10, 15]
result = hp_photometry(hp_hdu, coord, apertures=apertures)
```

result is then an `astropy.table.Table` with the aperture photometry

```
Out[1]:
<Table length=1>
brightness    background    n_pix
MJy / sr      MJy / sr
float64        float64      int64
-----
2.51999285723 1.2330693081    151
```

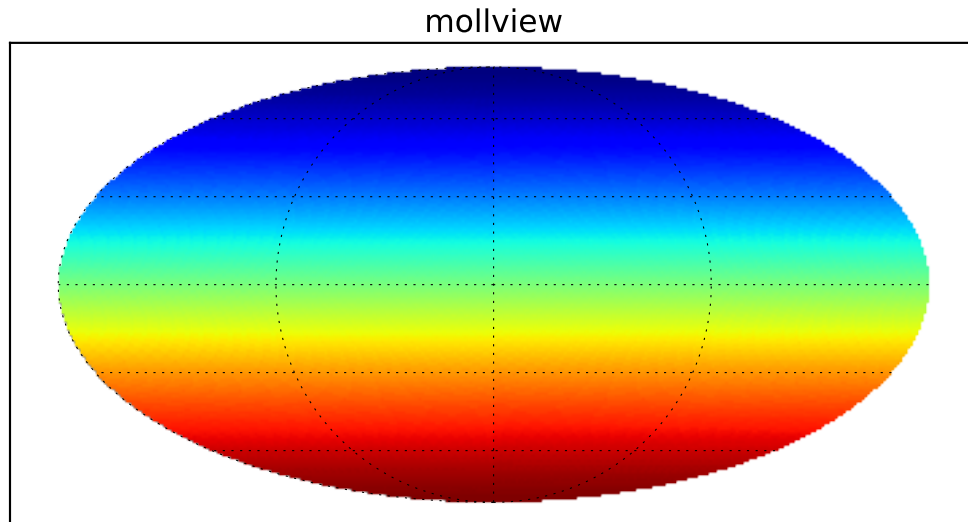
6.4 Visualization

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

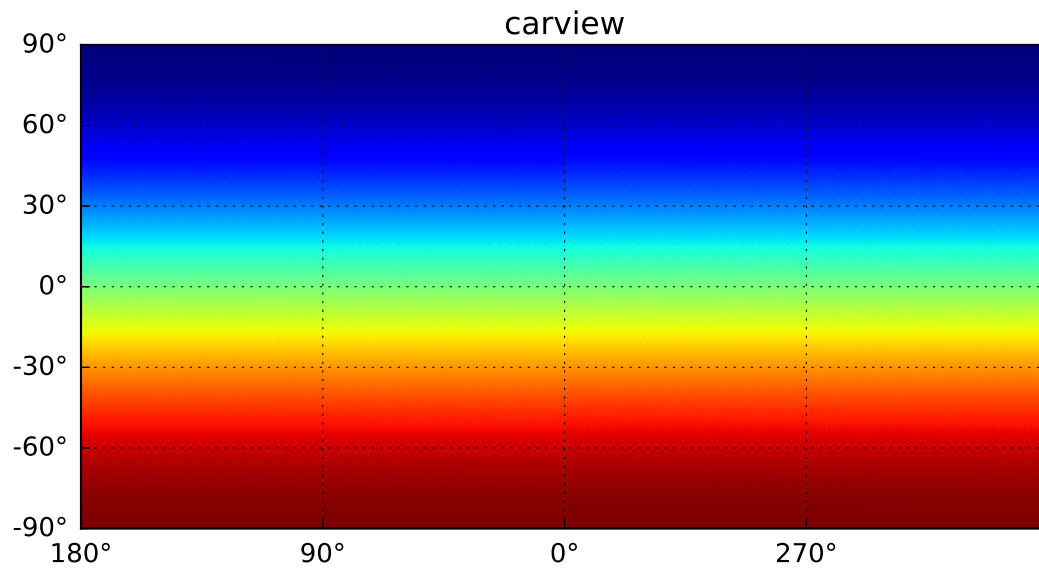
```
import matplotlib.pyplot as plt
import numpy as np
import healpy as hp
from astropy.wcs import WCS
from hproj import mollview

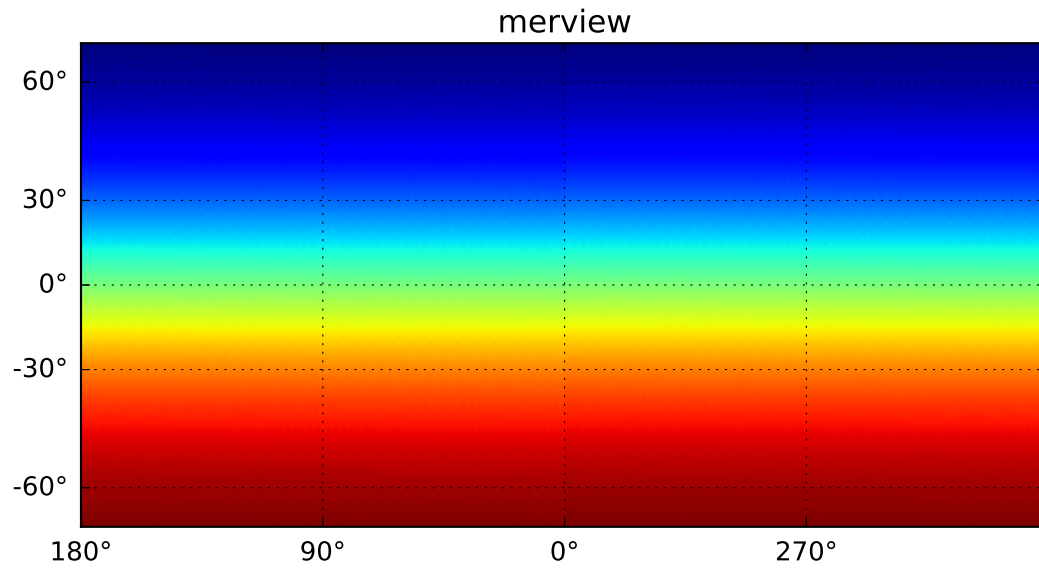
# Ring like healpix map
nside = 2**6
hp_map = np.arange(hp.nside2npix(nside))
hp_header = {'NSIDE': nside,
             'ORDERING': 'RING',
             'COORDSYS': 'G'}

# 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')
```

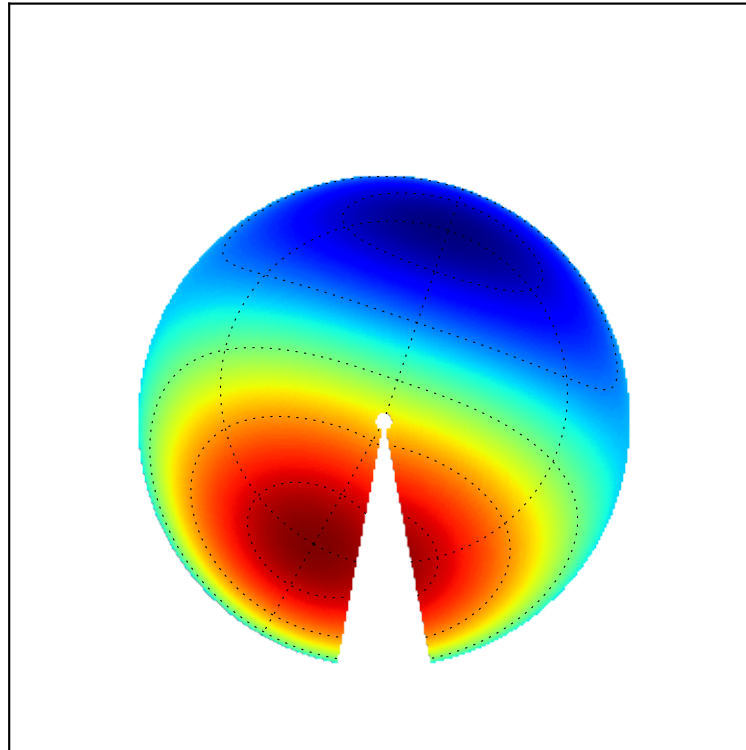


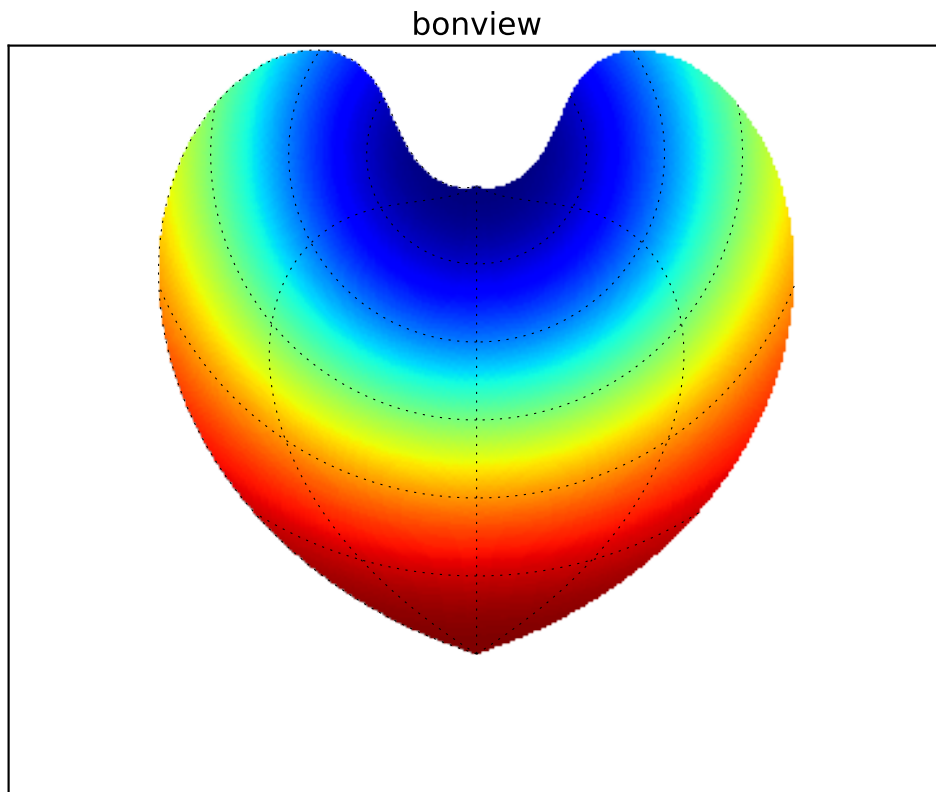
Note that these maps have a proper `WCS` header and thus can be easily used to overplot markers and artists. Other classical projections have been implemented

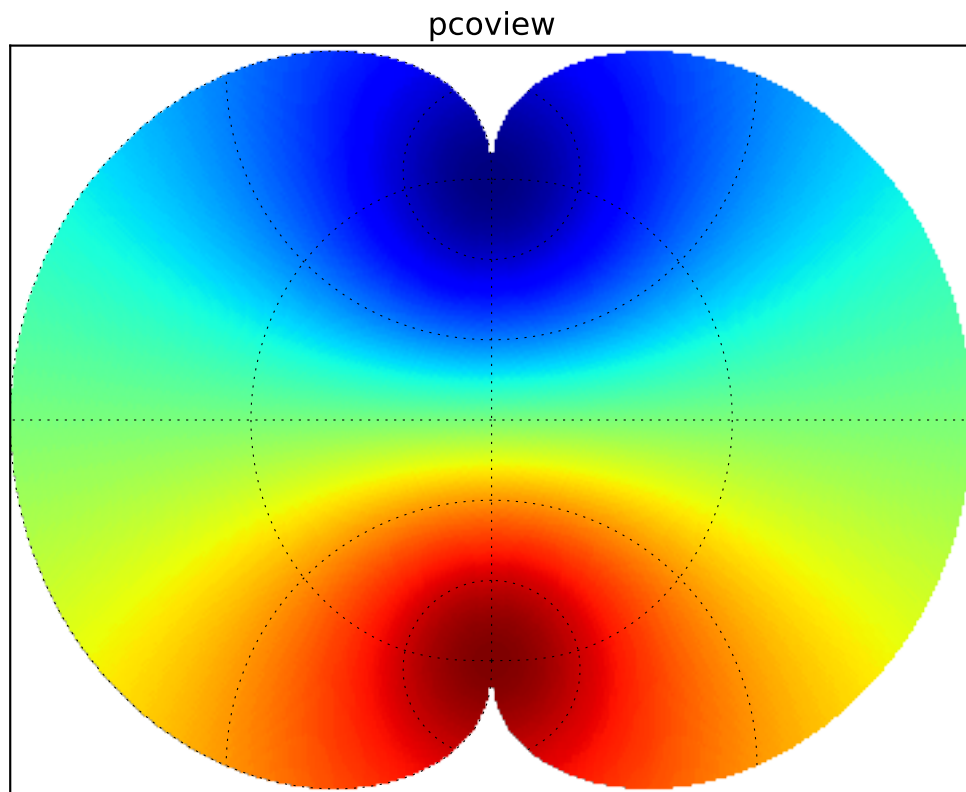


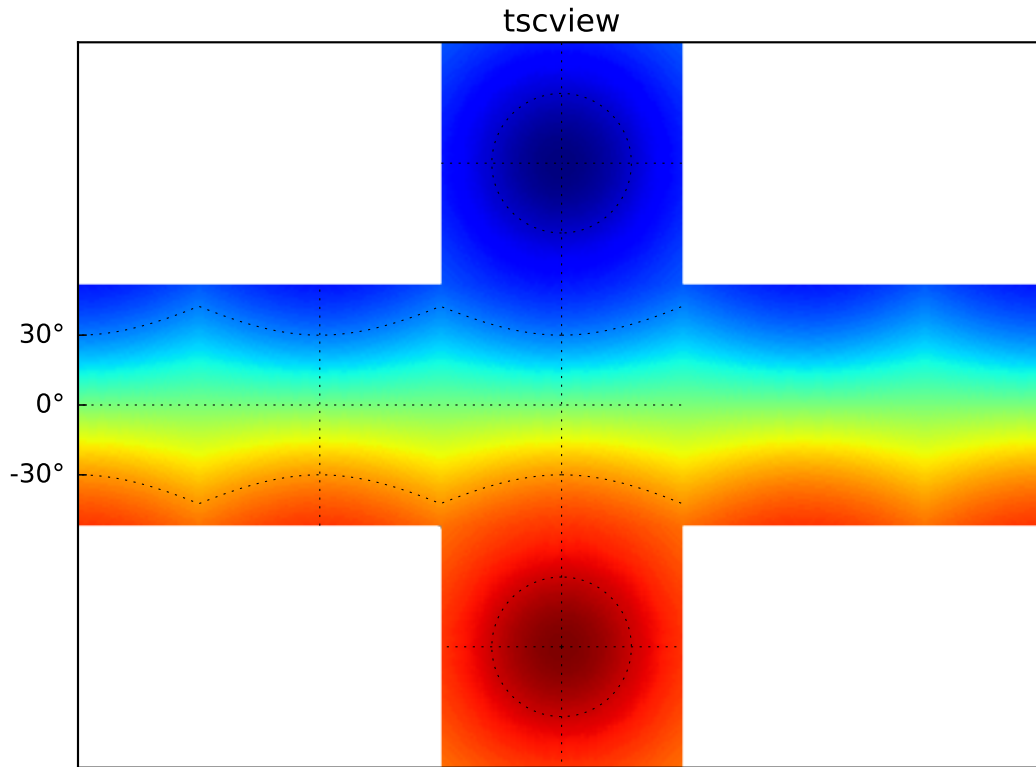


coeview









6.5 API

6.5.1 cutsky

cutsky module, mainly use *hproj.hp_helper* functions

class `hproj.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 [dictionnary] a grouped dictionnary of gen_hmap tuples (filename, map, header) (see :func:~init)

Methods

<code>cut(cut_type, **kwargs)</code>	helper function to cut into the maps
<code>cut_fits(coord[, maps_selection])</code>	Efficiently cut the healpix maps and return cutted fits file with proper header
<code>cut_phot([coord, maps_selection, apertures])</code>	Efficiently cut the healpix maps and return cutted fits file with proper header and corresponding photometry
<code>cut_png([coord, maps_selection])</code>	Efficiently cut the healpix maps and return cutted fits file with proper header and corresponding png

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

Parameters

- cut_type** [str (fits|png|phot|votable)] define what to cut_type
- coord** [[SkyCoord](#)] the sky coordinate for the projection. Its frame will be used for 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 dictionnaires the dictionnaire output depends on cut_type

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

Parameters

- coord** [[SkyCoord](#)] the sky coordinate for the projection. Its frame will be used for 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 dictionnaires the dictionnaire has 2 keys : * ‘legend’ (the opts{‘legend’} see `__init__()`) * ‘fits’ an [ImageHDU](#)

cut_phot (*coord=None*, *maps_selection=None*, *apertures=None*)
 Efficiently cut the healpix maps and return cutted fits file with proper header and corresponding photometry

Parameters

- coord** [[SkyCoord](#)] the sky coordinate for the projection. Its frame will be used for the projection
- maps_selection** [list] optionnal list of the ‘legend’ or filename of the map to select a sub-sample of them.
- apertures: float or list of float** aperture size in arcmin, if None, the aperture are guessed from the input map

Returns

list of dictionnaires the dictionnaire has 3 keys : * ‘legend’ (the opts{‘legend’} see `__init__()`), * ‘fits’ an [ImageHDU](#), * ‘phot’, the corresponding photometry

cut_png (*coord=None, maps_selection=None*)

Efficiently cut the healpix maps and return cutted fits file with proper header and corresponding png

Parameters

coord [*SkyCoord*] the sky coordinate for the projection. Its frame will be used for 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

`hproj.cutsky.cutsky` (*lonlat=None, maps=None, patch=None, coordframe='galactic', ctype='TAN', apertures=None*)

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 dictionary (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]] [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)

apertures: float of list of floats aperture in arcmin for the circular aperture photometry

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

`hproj.cutsky.main` (*argv=None*)

The main routine.

`hproj.cutsky.save_result` (*output, result*)

Save the results of the main function

`hproj.cutsky.to_coord` (*lonlat=None, coordframe='galactic'*)

helper function to get a *SkyCoord* object using the old interface

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

Returns

:class:~astropy.coordinates.SkyCoord the corresponding *SkyCoord*

`hpproj.cutsky.to_new_maps` (*maps*)

Transform old dictionary type healpix map list used by cutsky to list of tuple used by Cutsky

Parameters

maps [dict] a dictionary 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}), ...] ““

6.5.2 helpers

Series of helper function to deal with healpix maps

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

Return True if the healpix header is in nested

Parameters

hp_header [Header] the header

Returns

boolean : 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.

`hproj.hp_helper.hp_to_wcs(*args, **kwargs)`

Project an Healpix map on a wcs header, using nearest neighbors.

Parameters

hp_hdu [`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`]...header

wcs [`astropy.wcs.WCS`] wcs object to project with

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

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

`hproj.hp_helper.hp_to_wcs_ipx(hp_header, wcs, shape_out=(512, 512))`

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)

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]
```

`hproj.hp_helper.hp_project(*args, **kwargs)`

Project an healpix map at a single given position

Parameters

hp_hdu [`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`]...header

coord [`astropy.coordinate.SkyCoord`] the sky coordinate of the center of the projection

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

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

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

Returns

:class:'`astropy.io.fits.PrimaryHDU`' containing the array and the corresponding header

Notes

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

`hproj.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 inputted list

`hproj.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`

`hproj.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 `(build|gen)_hpmap` : (filename, healpix map, healpix header)

Returns

str A string with the map properties

`hproj.hp_helper.wcs_to_profile(hdu, wcs, shape_out=512)`

Centered profile from 2D map

Parameters

hdu [`astropy.fits.ImageHDU`] hdu containing the 2D array and corresponding header, the profile will be made from the CRVAL position

wcs [`astropy.wcs.WCS`] wcs object to describe the radius of the profile

shape_out [int] shape of the output profile

Returns

:class:‘`astropy.fits.ImageHDU`‘ 1D hdu image containing the profile and the corresponding header

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

Extract radial profile from healpix map

Parameters

hp_hdu [`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`]...header

wcs [`astropy.wcs.WCS`] wcs object to describe the radius of the profile

coord [`astropy.coordinate.SkyCoord`] the sky coordinate of the center of the profile

shape_out [int] shape of the output profile

std [bool] return the standard deviation

Returns

:class:‘`astropy.fits.ImageHDU`‘ 1D hdu image containing the profile and the corresponding header, optionally a second ImageHDU containing the standard deviation

Notes

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

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

Project an healpix map at a single given position

Parameters

hp_hdu [`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`]...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

Returns

:class:‘`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.hp_stack(*args, **kwargs)`
Perform stacking on an healpix map

Parameters

hp_hdu [`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`]...header

coords [list of `astropy.coordinate.SkyCoord`] list of sky coordinates for the center of the cropped maps

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

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

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

keep [boolean (default False)] return all the cropped maps as a 3D cube instead of one stack map

Returns

:class:~`fits.ImageHDU` hdu containing the stack image or cube and corresponding header

Notes

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

`hpproj.hp_helper.hp_to_aperture(*args, **kwargs)`
Raw aperture summation on an healpix map

Parameters

hp_hdu [`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`]...header

coords [list of `astropy.coordinate.SkyCoord`] the sky coordinates for the center of the apertures

apertures [list of :class:astropy.coordinates.Angles] aperture angle in which we perform summation

Returns

npix, apertures [array_like] 2 arrays containing the number of pixels, and sum of the pixels within the aperture respectively

Notes

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

`hproj.hp_helper.hp_photometry(*args, **kwargs)`

Aperture photometry on an healpix map at a single given position

Parameters

hp_hdu [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]...header

coords [astropy.coordinate.SkyCoord] the sky coordinates for the center of the apertures

apertures [3 :class:astropy.coordinates.Angles] 3 floats defining the aperture radius and inner/outer annulus radii

Returns

:class:'astropy.io.fits.BinaryHDU' table containing the photometry

Notes

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

Series of helper function to deal with building wcs objects

`hproj.wcs_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 [keyword, str, ('GALACTIC', 'EQUATORIAL')] the coordinate system of the longitude and latitude (default EQUATORIAL)

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

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

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: :class:`~astropy.wcs.WCS` An corresponding wcs object

Notes

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

```
hpproj.wcs_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 [keyword, str, ('GALACTIC', 'EQUATORIAL')] the coordinate system of the longitude and latitude
(default EQUATORIAL)

index [int] reference index

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

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

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: :class:`~astropy.wcs.WCS` An corresponding wcs object

Notes

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

```
hpproj.wcs_helper.build_wcs_2pts (coords,      pixsize=None,      shape_out=(512,      512),  
                                proj_sys='EQUATORIAL',      proj_type='TAN',      rela-  
                                tive_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)

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: :class:`~astropy.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.wcs_helper.build_ctype(coordsys, proj_type)`

Build a valid spatial ctype for a wcs header

Parameters

coordsys [str ('GALACTIC', '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.wcs_helper.equiv_celestial(frame)`

Return an equivalent ~astropy.coordinates.builtin_frames

Notes

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

`hpproj.wcs_helper.rot_frame(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

:class:`~astropy.coordinate.SkyCoord` rotated frame

6.5.3 visu

Series of full sky visualization function, with proper wcs header

`hpproj.visu.view(*args, **kwargs)`

projection of the full sky

Parameters

hp_hdu [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]...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

Returns

:class:'`astropy.io.fits.ImageHDU`' 2D images with header

Notes

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

`hpproj.visu.mollview()`
Mollweide projection of the full sky

Parameters

hp_hdu [`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`]...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

Returns

:class:'`astropy.io.fits.ImageHDU`' 2D images with header

Notes

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

`hpproj.visu.orthview(*args, **kwargs)`
Slant orthographic projection of the full sky

Parameters

hp_hdu [`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`]...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

Returns

:class:'`astropy.io.fits.ImageHDU`' 2D images with header

Notes

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

`hproproj.visu.carview()`

Plate carrée projection of the full sky

Parameters

hp_hdu [`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`]...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

Returns

:class:'`astropy.io.fits.ImageHDU`' 2D images with header

Notes

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

`hproproj.visu.merview()`

Mercator projection of the full sky

Parameters

hp_hdu [`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`]...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

Returns

:class:'`astropy.io.fits.ImageHDU`' 2D images with header

Notes

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

`hpproj.visu.coeview()`

Conic Equal Area projection of the full sky

Parameters

hp_hdu [`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`]...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

Returns

:class:'`astropy.io.fits.ImageHDU`' 2D images with header

Notes

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

`hpproj.visu.bonview()`

Bonne's Equal Area projection of the full sky

Parameters

hp_hdu [`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`]...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

Returns

:class:'`astropy.io.fits.ImageHDU`' 2D images with header

Notes

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

`hpproj.visu.pcoview()`

Hassler's polyconic projection of the full sky

Parameters

hp_hdu [`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`]...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

Returns

:class:'`astropy.io.fits.ImageHDU`' 2D images with header

Notes

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

`hpproj.visu.tscview()`

Tangential spherical cube projection of the full sky

Parameters

hp_hdu [`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]...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

Returns

:class:'astropy.io.fits.ImageHDU' 2D images with header

Notes

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

CHAPTER 7

Indices and tables

- `genindex`
- `modindex`
- `search`

h

`hproj.cutsky`, [29](#)
`hproj.hp_helper`, [32](#)
`hproj.visu`, [39](#)
`hproj.wcs_helper`, [37](#)

B

[bonview\(\)](#) (in module [hproj.visu](#)), 42
[build_ctype\(\)](#) (in module [hproj.wcs_helper](#)), 39
[build_hmap\(\)](#) (in module [hproj.hp_helper](#)), 34
[build_wcs\(\)](#) (in module [hproj.wcs_helper](#)), 37
[build_wcs_2pts\(\)](#) (in module [hproj.wcs_helper](#)), 38
[build_wcs_cube\(\)](#) (in module [hproj.wcs_helper](#)), 38

C

[carview\(\)](#) (in module [hproj.visu](#)), 41
[coevview\(\)](#) (in module [hproj.visu](#)), 42
[cut\(\)](#) ([hproj.cutsky.CutSky](#) method), 30
[cut_fits\(\)](#) ([hproj.cutsky.CutSky](#) method), 30
[cut_phot\(\)](#) ([hproj.cutsky.CutSky](#) method), 30
[cut_png\(\)](#) ([hproj.cutsky.CutSky](#) method), 30
[CutSky](#) (class in [hproj.cutsky](#)), 29
[cutsky\(\)](#) (in module [hproj.cutsky](#)), 31

E

[equiv_celestial\(\)](#) (in module [hproj.wcs_helper](#)), 39

G

[gen_hmap\(\)](#) (in module [hproj.hp_helper](#)), 34

H

[hp_celestial\(\)](#) (in module [hproj.hp_helper](#)), 32
[hp_is_nest\(\)](#) (in module [hproj.hp_helper](#)), 32
[hp_photometry\(\)](#) (in module [hproj.hp_helper](#)), 37
[hp_profile\(\)](#) (in module [hproj.hp_helper](#)), 35
[hp_project\(\)](#) (in module [hproj.hp_helper](#)), 33
[hp_stack\(\)](#) (in module [hproj.hp_helper](#)), 36
[hp_to_aperture\(\)](#) (in module [hproj.hp_helper](#)), 36
[hp_to_profile\(\)](#) (in module [hproj.hp_helper](#)), 35
[hp_to_wcs\(\)](#) (in module [hproj.hp_helper](#)), 32
[hp_to_wcs_ipx\(\)](#) (in module [hproj.hp_helper](#)), 33
[hmap_key\(\)](#) (in module [hproj.hp_helper](#)), 34
[hproj.cutsky](#) (module), 29
[hproj.hp_helper](#) (module), 32
[hproj.visu](#) (module), 39

[hproj.wcs_helper](#) (module), 37

M

[main\(\)](#) (in module [hproj.cutsky](#)), 31
[mervview\(\)](#) (in module [hproj.visu](#)), 41
[mollview\(\)](#) (in module [hproj.visu](#)), 40

O

[orthview\(\)](#) (in module [hproj.visu](#)), 40

P

[pcovview\(\)](#) (in module [hproj.visu](#)), 43

R

[rot_frame\(\)](#) (in module [hproj.wcs_helper](#)), 39

S

[save_result\(\)](#) (in module [hproj.cutsky](#)), 31

T

[to_coord\(\)](#) (in module [hproj.cutsky](#)), 31
[to_new_maps\(\)](#) (in module [hproj.cutsky](#)), 31
[tscview\(\)](#) (in module [hproj.visu](#)), 43

V

[view\(\)](#) (in module [hproj.visu](#)), 39

W

[wcs_to_profile\(\)](#) (in module [hproj.hp_helper](#)), 34