FITS$_{t}ools Documentation$

Release 0.3.dev297

Adam Ginsburg

Sep 29, 2017

Contents

This is an affiliated package for the AstroPy package. The documentation for this package is here:

Part I

Tools

CHAPTER 1

Image Regridding

FITS_tools.hcongrid.hcongrid is meant to replicate hcongrid and hastrom. It uses scipy's interpolation routines.

FITS_tools.hcongrid.wcsalign does the same thing as hcongrid but uses pyast as its backend.

CHAPTER 2

Cube Regridding

`FITS_tools.cube_regrid.regrid_fits_cube` reprojects a cube to a new grid using scipy's interpolation routines.

`FITS_tools.match_images.match_fits_cubes` takes two cubes, and reprojects the first to the coordinates of the second (it's a wrapper)

For a flux-conserving (but slower) approach, there is a wrapper of montage in python-montage.

Part II

Reference/API

CHAPTER 3

FITS_tools Package

This is an Astropy affiliated package.

Functions

fits_overlap(file1, file2, **kwargs)	Create a header containing the exact overlap region between two .fits files
header_overlap(hdr1, hdr2[, max_separation, ...])	Create a header containing the exact overlap region between two .fits files
match_fits(fitsfile1, fitsfile2[, header, ...])	Project one FITS file into another's coordinates.
project_to_header(fitsfile, header[, ...])	Light wrapper of montage with hcongrid as a backup
regrid_cube(cubedata, cubeheader, targetheader)	Attempt to reproject a cube onto another cube's header.
regrid_cube_hdu(hdu, outheader[, smooth])	Regrid a FITS HDU to a target header.
regrid_fits_cube(cubefilename, outheader[, ...])	Regrid a FITS file to a target header.
test([package, test_path, args, plugins, ...])	Run the tests using py.test.

fits_overlap

FITS_tools.**fits_overlap**(*file1*, *file2*, ***kwargs*)

Create a header containing the exact overlap region between two .fits files

Does NOT check to make sure the FITS files are in the same coordinate system!

Parameters

file1,file2 : str,str

files from which to extract header strings

header_overlap

FITS_tools.**header_overlap**(*hdr1*, *hdr2*, *max_separation=180*, *overlap='union'*)
Create a header containing the exact overlap region between two .fits files

Does NOT check to make sure the FITS files are in the same coordinate system!

Parameters

hdr1,hdr2 : Header

Two pyfits headers to compare

max_separation : int

Maximum number of degrees between two headers to consider before flipping signs on one of them (this to deal with the longitude=0 region)

overlap : 'union' or 'intersection'

Which merger to do

match_fits

FITS_tools.**match_fits**(*fitsfile1*, *fitsfile2*, *header=None*, *sigma_cut=False*, *return_header=False*, ***kwargs*)
Project one FITS file into another's coordinates. If `sigma_cut` is used, will try to find only regions that are significant in both images using the standard deviation, masking out other signal

Parameters

fitsfile1 : str

Offset fits file name

fitsfile2 : str

Reference fits file name

header : Header

Optional - can pass a header that both input images will be projected to match

sigma_cut : bool or int

Perform a sigma-cut on the returned images at this level

Returns

image1,image2,[header] : ndarray, ndarray, Header

Two images projected into the same space, and optionally the header used to project them

See also:

match_fits_cubes
this function, but for cubes

project_to_header

FITS_tools.**project_to_header**(*fitsfile*, *header*, *use_montage=True*, *quiet=True*, ***kwargs*)
Light wrapper of montage with hcongrid as a backup

kwargs will be passed to `hcongrid` if `use_montage==False`

Parameters

fitsfile : string

a FITS file name

header : `Header`

A fits Header instance with valid WCS to project to

use_montage : bool

Use montage or hcongrid (based on `map_coordinates`)

quiet : bool

Silence Montage's output

Returns

image : `ndarray`

image projected to header's coordinates

regrid_cube

FITS_tools.**regrid_cube**(*cubedata*, *cubeheader*, *targetheader*, *preserve_bad_pixels=True*, *order=1*, *mode='constant'*, *out_of_bounds=nan*, ***kwargs*)

Attempt to reproject a cube onto another cube's header. Uses interpolation via `map_coordinates`

Assumptions:

- Both the cube and the target are 3-dimensional, with lon/lat/spectral axes
- Both cube and header use CD/CDELT rather than PC

`kwargs` will be passed to `map_coordinates`

Parameters

cubedata : `ndarray`

A three-dimensional data cube

cubeheader : `Header` or `WCS`

The header or WCS corresponding to the image

targetheader : `Header` or `WCS`

The header or WCS to interpolate onto

preserve_bad_pixels : bool

Try to set NAN pixels to NAN in the zoomed image. Otherwise, bad pixels will be set to zero

out_of_bounds : float or np.nan

Pixels in the output image that were out of bounds in the old image will be set to this value *if* mode='constant'

mode : str

The mode of treatment of out-of-bounds pixels; can be 'constant', 'wrap', 'reflect', or 'nearest'. See `map_coordinates` for details.

Returns

newcubedata : `ndarray`

The regridded cube

regrid_cube_hdu

FITS_tools.**regrid_cube_hdu**(*hdu*, *outheader*, *smooth=False*, ***kwargs*)

Regrid a FITS HDU to a target header. Requires that the FITS object and the target header have spectral and spatial overlap. See `regrid_cube` for additional details or alternative input options.

Parameters

hdu : `PrimaryHDU`

FITS HDU (not HDUlist) containing the cube to be reprojected

outheader : `Header`

A target Header to project to

smooth : bool

Smooth the HDUs to match resolution? Kernel size is determined using `smoothing_kernel_size`

Warning: Smoothing is done in 3D to be maximally general. This can be exceedingly slow!

Returns

rgcube : `PrimaryHDU`

The regridded cube HDU

regrid_fits_cube

FITS_tools.**regrid_fits_cube**(*cubefilename*, *outheader*, *hdu=0*, *outfilename=None*, *clobber=False*, ***kwargs*)

Regrid a FITS file to a target header. Requires that the FITS file and the target header have spectral and spatial overlap.

See `regrid_cube_hdu` and `regrid_cube` for additional details or alternative input options.

Parameters

cubefilename : str

FITS file name containing the cube to be reprojected

outheader : `Header`

A target Header to project to

hdu : int

The hdu to project to the target header

outfilename : str

The output filename to save to

clobber : bool

Overwrite the output file if it exists?

kwargs : dict

Passed to `regrid_cube_hdu`

Returns

rgcube : PrimaryHDU

The regridded cube HDU

test

FITS_tools.**test**(*package=None*, *test_path=None*, *args=None*, *plugins=None*, *verbose=False*, *pastebin=None*, *remote_data=False*, *pep8=False*, *pdb=False*, *coverage=False*, *open_files=False*, ***kwargs*)

Run the tests using py.test. A proper set of arguments is constructed and passed to pytest.main.

Parameters

package : str, optional

The name of a specific package to test, e.g. 'io.fits' or 'utils'. If nothing is specified all default tests are run.

test_path : str, optional

Specify location to test by path. May be a single file or directory. Must be specified absolutely or relative to the calling directory.

args : str, optional

Additional arguments to be passed to pytest.main in the `args` keyword argument.

plugins : list, optional

Plugins to be passed to pytest.main in the `plugins` keyword argument.

verbose : bool, optional

Convenience option to turn on verbose output from py.test. Passing True is the same as specifying `'-v'` in `args`.

pastebin : {'failed','all',None}, optional

Convenience option for turning on py.test pastebin output. Set to `'failed'` to upload info for failed tests, or `'all'` to upload info for all tests.

remote_data : bool, optional

Controls whether to run tests marked with @remote_data. These tests use online data and are not run by default. Set to True to run these tests.

pep8 : bool, optional

Turn on PEP8 checking via the pytest-pep8 plugin and disable normal tests. Same as specifying `'--pep8 -k pep8'` in `args`.

pdb : bool, optional

Turn on PDB post-mortem analysis for failing tests. Same as specifying `'--pdb'` in `args`.

coverage : bool, optional

Generate a test coverage report. The result will be placed in the directory htmlcov.

open_files : bool, optional

Fail when any tests leave files open. Off by default, because this adds extra run time to the test suite. Works only on platforms with a working `lsof` command.

parallel : int, optional

When provided, run the tests in parallel on the specified number of CPUs. If parallel is negative, it will use the all the cores on the machine. Requires the pytest-xdist plugin installed. Only available when using Astropy 0.3 or later.

kwargs

Any additional keywords passed into this function will be passed on to the astropy test runner. This allows use of test-related functionality implemented in later versions of astropy without explicitly updating the package template.

CHAPTER 4

FITS_tools.cube_regrid Module

Functions

downsample_cube(cubehdu, factor[, spectralaxis])	Downsample a cube along the spectral axis
get_cube_mapping(header1, header2)	Determine the pixel mapping from Header 1 to Header 2
gsmooth_cube(cube, kernelsize[, use_fft, ...])	Smooth a cube with a gaussian in 3d
regrid_cube(cubedata, cubeheader, targetheader)	Attempt to reproject a cube onto another cube's header.
regrid_cube_hdu(hdu, outheader[, smooth])	Regrid a FITS HDU to a target header.
regrid_fits_cube(cubefilename, outheader[, ...])	Regrid a FITS file to a target header.
smoothing_kernel_size(hdr_from, hdr_to)	Determine the smoothing kernel size needed to convolve a cube before downsampling it to retain signal.
spatial_smooth_cube(cube, kernelwidth[, ...])	Parallelized spatial smoothing
spectral_smooth_cube(cube, kernelwidth[, ...])	Parallelized spectral smoothing
find_grid_limits(grid)	Determine the min/max of each dimension along a grid returned by get_cube_mapping.

downsample_cube

FITS_tools.cube_regrid.**downsample_cube**(*cubehdu*, *factor*, *spectralaxis=None*)

Downsample a cube along the spectral axis

Parameters

cubehdu : PrimaryHDU

The cube to downsample

factor : int

The factor by which the cube should be downsampled

spectralaxis : int

The 0-indexed ID fo the spectral axis. Will be determined automatically if not specified

get_cube_mapping

FITS_tools.cube_regrid.**get_cube_mapping**(*header1*, *header2*)

Determine the pixel mapping from Header 1 to Header 2

Assumptions are spelled out in `regrid_cube`

gsmooth_cube

FITS_tools.cube_regrid.**gsmooth_cube**(*cube*, *kernelsize*, *use_fft=True*, *psf_pad=False*, *fft_pad=False*, *kernelsize_mult=8*, ***kwargs*)

Smooth a cube with a gaussian in 3d

Because even a tiny cube can become enormous if you have, say, a 1024x32x32 cube, padding is off by default

regrid_cube

FITS_tools.cube_regrid.**regrid_cube**(*cubedata*, *cubeheader*, *targetheader*, *preserve_bad_pixels=True*, *order=1*, *mode='constant'*, *out_of_bounds=nan*, ***kwargs*)

Attempt to reproject a cube onto another cube's header. Uses interpolation via `map_coordinates`

Assumptions:

- Both the cube and the target are 3-dimensional, with lon/lat/spectral axes
- Both cube and header use CD/CDELT rather than PC

`kwargs` will be passed to `map_coordinates`

Parameters

cubedata : `ndarray`

A three-dimensional data cube

cubeheader : `Header` or `WCS`

The header or WCS corresponding to the image

targetheader : `Header` or `WCS`

The header or WCS to interpolate onto

preserve_bad_pixels : bool

Try to set NAN pixels to NAN in the zoomed image. Otherwise, bad pixels will be set to zero

out_of_bounds : float or np.nan

Pixels in the output image that were out of bounds in the old image will be set to this value *if* mode='constant'

mode : str

The mode of treatment of out-of-bounds pixels; can be 'constant', 'wrap', 'reflect', or 'nearest'. See `map_coordinates` for details.

Returns

newcubedata : `ndarray`

The regridded cube

regrid_cube_hdu

FITS_tools.cube_regrid.**regrid_cube_hdu**(*hdu*, *outheader*, *smooth=False*, ***kwargs*)

Regrid a FITS HDU to a target header. Requires that the FITS object and the target header have spectral and spatial overlap. See `regrid_cube` for additional details or alternative input options.

Parameters

hdu : `PrimaryHDU`

FITS HDU (not HDUlist) containing the cube to be reprojected

outheader : `Header`

A target Header to project to

smooth : bool

Smooth the HDUs to match resolution? Kernel size is determined using `smoothing_kernel_size`

Warning: Smoothing is done in 3D to be maximally general. This can be exceedingly slow!

Returns

rgcube : `PrimaryHDU`

The regridded cube HDU

regrid_fits_cube

FITS_tools.cube_regrid.**regrid_fits_cube**(*cubefilename*, *outheader*, *hdu=0*, *outfilename=None*, *clobber=False*, ***kwargs*)

Regrid a FITS file to a target header. Requires that the FITS file and the target header have spectral and spatial overlap.

See `regrid_cube_hdu` and `regrid_cube` for additional details or alternative input options.

Parameters

cubefilename : str

FITS file name containing the cube to be reprojected

outheader : `Header`

A target Header to project to

hdu : int

The hdu to project to the target header

outfilename : str

The output filename to save to

clobber : bool

Overwrite the output file if it exists?

kwargs : dict

Passed to `regrid_cube_hdu`

Returns

rgcube : `PrimaryHDU`

The regridded cube HDU

smoothing_kernel_size

FITS_tools.cube_regrid.**smoothing_kernel_size**(*hdr_from*, *hdr_to*)

Determine the smoothing kernel size needed to convolve a cube before downsampling it to retain signal.

spatial_smooth_cube

FITS_tools.cube_regrid.**spatial_smooth_cube**(*cube*, *kernelwidth*, *kernel=<class 'astropy.convolution.kernels.Gaussian2DKernel'>*, *cubedim=0*, *numcores=None*, *use_fft=True*, ***kwargs*)

Parallelized spatial smoothing

Parameters

cube : ndarray

A data cube, with ndim=3

kernelwidth : float

Width of the kernel. Defaults to Gaussian.

kernel : Kernel2D

A 2D kernel from astropy

cubedim : int

The axis to map across. If you have a normal FITS data cube with AXIS1=RA, AXIS2=Dec, and AXIS3=wavelength, for example, cubedim is 0 (because axis3 -> 0, axis2 -> 1, axis1 -> 2)

numcores : int

Number of cores to use in parallel-processing.

use_fft : bool

Use astropy.convolution.convolve_fft or astropy.convolution.convolve?

kwargs : dict

Passed to convolve

spectral_smooth_cube

FITS_tools.cube_regrid.**spectral_smooth_cube**(*cube*, *kernelwidth*, *kernel=<class 'astropy.convolution.kernels.Gaussian1DKernel'>*, *cubedim=0*, *parallel=True*, *numcores=None*, *use_fft=False*, ***kwargs*)

Parallelized spectral smoothing

Parameters

cube : ndarray

A data cube, with ndim=3

kernelwidth : float

Width of the kernel. Defaults to Gaussian.

kernel : Kernel1D

A 1D kernel from astropy

cubedim : int

The axis *NOT* to map across, i.e. the spectral axis. If you have a normal FITS data cube with AXIS1=RA, AXIS2=Dec, and AXIS3=wavelength, for example, cubedim is 0 (because axis3 -> 0, axis2 -> 1, axis1 -> 2)

numcores : int

Number of cores to use in parallel-processing.

use_fft : bool

Use convolve_fft or convolve?

kwargs : dict

Passed to astropy.convolution.convolve

find_grid_limits

FITS_tools.cube_regrid.**find_grid_limits**(*grid*)

Determine the min/max of each dimension along a grid returned by get_cube_mapping. These parameters can be used to slice a cube prior to passing it to, e.g., scipy.ndimage.map_coordinates to reduce the memory use

CHAPTER 5

FITS_tools.downsample Module

Functions

downsample_axis(myarr, factor, axis[, ...])	Downsample an ND array by averaging over *factor* pixels along an axis.
downsample_header(header, factor, axis)	Downsample a FITS header along an axis using the FITS convention for axis number

downsample_axis

FITS_tools.downsample.**downsample_axis**(*myarr*, *factor*, *axis*, *estimator=<function nanmean>*, *truncate=False*)

Downsample an ND array by averaging over *factor* pixels along an axis. Crops right side if the shape is not a multiple of factor.

This code is pure np and should be fast.

Parameters

myarr : ndarray

The array to downsample

factor : int

The factor to downsample by

axis : int

The axis to downsample along

estimator : function

defaults to mean. You can downsample by summing or something else if you want a different estimator (e.g., downsampling error: you want to sum & divide by sqrt(n))

truncate : bool

Whether to truncate the last chunk or average over a smaller number. e.g., if you downsample [1,2,3,4] by a factor of 3, you could get either [2] or [2,4] if truncate is True or False, respectively.

downsample_header

FITS_tools.downsample.**downsample_header**(*header*, *factor*, *axis*)

Downsample a FITS header along an axis using the FITS convention for axis number

CHAPTER 6

FITS_tools.spectral_regrid Module

Functions

get_spectral_mapping(header1, header2[, ...])	Determine the mapping from header1 pixel units to header2 pixel units along
spec_pix_to_world(pixel, wcs, axisnumber[, unit])	Given a WCS, an axis ID, and a pixel ID, return the WCS spectral value at a
spec_world_to_pix(worldunit, wcs, ...)	Given a WCS, an axis ID, and WCS location, return the pixel index value at a

get_spectral_mapping

FITS_tools.spectral_regrid.**get_spectral_mapping**(*header1*, *header2*, *specaxis1=None*, *specaxis2=None*)

Determine the mapping from header1 pixel units to header2 pixel units along their respective spectral axes

spec_pix_to_world

FITS_tools.spectral_regrid.**spec_pix_to_world**(*pixel*, *wcs*, *axisnumber*, *unit=None*)

Given a WCS, an axis ID, and a pixel ID, return the WCS spectral value at a pixel location

spec_world_to_pix

FITS_tools.spectral_regrid.**spec_world_to_pix**(*worldunit*, *wcs*, *axisnumber*, *unit*)

Given a WCS, an axis ID, and WCS location, return the pixel index value at a pixel location

CHAPTER 7

FITS_tools.hcongrid Module

Functions

get_pixel_mapping(header1, header2)	Determine the mapping from pixel coordinates in header1 to pixel
hastrom(image, header1, header2[, ...])	Interpolate an image from one FITS header onto another
hastrom_hdu(hdu_in, header, **kwargs)	Wrapper of hcongrid to work on HDUs
hcongrid(image, header1, header2[, ...])	Interpolate an image from one FITS header onto another
hcongrid_hdu(hdu_in, header, **kwargs)	Wrapper of hcongrid to work on HDUs
wcsalign(hdu_in, header[, outname, clobber])	Align one FITS image to a specified header
zoom_fits(fitsfile, scalefactor[, ...])	Zoom in on a FITS image by interpolating using zoom

get_pixel_mapping

FITS_tools.hcongrid.**get_pixel_mapping**(*header1*, *header2*)

Determine the mapping from pixel coordinates in header1 to pixel coordinates in header2

Parameters

header1 : Header or WCS

The header or WCS corresponding to the image to be transformed

header2 : Header or WCS

The header or WCS to interpolate onto

Returns

grid : ndarray

ndarray describing a grid of y,x pixel locations in the input header's pixel units but the output header's world units

Raises

TypeError :

If either header is not a Header or WCS instance NotImplementedError if the CTYPE in the header is not recognized

hastrom

FITS_tools.hcongrid.**hastrom**(*image*, *header1*, *header2*, *preserve_bad_pixels=True*, ***kwargs*)

Interpolate an image from one FITS header onto another

kwargs will be passed to `map_coordinates`

Parameters

image : `ndarray`

A two-dimensional image

header1 : `Header` or `WCS`

The header or WCS corresponding to the image

header2 : `Header` or `WCS`

The header or WCS to interpolate onto

preserve_bad_pixels : bool

Try to set NAN pixels to NAN in the zoomed image. Otherwise, bad pixels will be set to zero

Returns

newimage : `ndarray`

ndarray with shape defined by header2's naxis1/naxis2

Raises

TypeError if either is not a Header or WCS instance

Exception if image1's shape doesn't match header1's naxis1/naxis2

Examples

```
>>> fits1 = pyfits.open('test.fits')
>>> target_header = pyfits.getheader('test2.fits')
>>> new_image = hcongrid(fits1[0].data, fits1[0].header, target_header)
```

hastrom_hdu

FITS_tools.hcongrid.**hastrom_hdu**(*hdu_in*, *header*, ***kwargs*)

Wrapper of hcongrid to work on HDUs

See `hcongrid` for details

hcongrid

FITS_tools.hcongrid.**hcongrid**(*image*, *header1*, *header2*, *preserve_bad_pixels=True*, ***kwargs*)

Interpolate an image from one FITS header onto another

kwargs will be passed to `map_coordinates`

Parameters

image : ndarray

A two-dimensional image

header1 : Header or WCS

The header or WCS corresponding to the image

header2 : Header or WCS

The header or WCS to interpolate onto

preserve_bad_pixels : bool

Try to set NAN pixels to NAN in the zoomed image. Otherwise, bad pixels will be set to zero

Returns

newimage : ndarray

ndarray with shape defined by header2's naxis1/naxis2

Raises

TypeError if either is not a Header or WCS instance

Exception if image1's shape doesn't match header1's naxis1/naxis2

Examples

```
>>> fits1 = pyfits.open('test.fits')
>>> target_header = pyfits.getheader('test2.fits')
>>> new_image = hcongrid(fits1[0].data, fits1[0].header, target_header)
```

hcongrid_hdu

FITS_tools.hcongrid.**hcongrid_hdu**(*hdu_in*, *header*, ***kwargs*)

Wrapper of hcongrid to work on HDUs

See hcongrid for details

wcsalign

FITS_tools.hcongrid.**wcsalign**(*hdu_in*, *header*, *outname=None*, *clobber=False*)

Align one FITS image to a specified header

Requires pyast.

Parameters

hdu_in : PrimaryHDU

The HDU to reproject (must have header & data)

header : Header

The target header to project to

outname : str (optional)

The filename to write to.

clobber : bool

Overwrite the file `outname` if it exists

Returns
The reprojected fits.PrimaryHDU

zoom_fits

`FITS_tools.hcongrid.`**`zoom_fits`**(*fitsfile*, *scalefactor*, *preserve_bad_pixels=True*, ***kwargs*)
Zoom in on a FITS image by interpolating using `zoom`

Parameters
fitsfile : str

FITS file name

scalefactor : float

Zoom factor along all axes

preserve_bad_pixels : bool

Try to set NAN pixels to NAN in the zoomed image. Otherwise, bad pixels will be set to zero

CHAPTER 8

FITS_tools.match_images Module

Functions

match_fits(fitsfile1, fitsfile2[, header, ...])	Project one FITS file into another's coordinates.
match_fits_cubes(fitsfile1, fitsfile2[, ...])	Project one FITS file representing a data cube into another's coordinates.
project_to_header(fitsfile, header[, ...])	Light wrapper of montage with hcongrid as a backup

match_fits

FITS_tools.match_images.**match_fits**(*fitsfile1*, *fitsfile2*, *header=None*, *sigma_cut=False*, *return_header=False*, ***kwargs*)

Project one FITS file into another's coordinates. If sigma_cut is used, will try to find only regions that are significant in both images using the standard deviation, masking out other signal

Parameters

fitsfile1 : str

Offset fits file name

fitsfile2 : str

Reference fits file name

header : Header

Optional - can pass a header that both input images will be projected to match

sigma_cut : bool or int

Perform a sigma-cut on the returned images at this level

Returns

image1,image2,[header] : ndarray, ndarray, Header

Two images projected into the same space, and optionally the header used to project them

See also:

`match_fits_cubes`
this function, but for cubes

match_fits_cubes

FITS_tools.match_images.**match_fits_cubes**(*fitsfile1*, *fitsfile2*, *header=None*, *sigma_cut=False*, *return_header=False*, *smooth=False*, ***kwargs*)

Project one FITS file representing a data cube into another's coordinates.

Parameters

fitsfile1 : str

FITS file name to reproject

fitsfile2 : str

Reference FITS file name. If `header` is specified,

smooth : bool

Smooth the HDUs to match resolution? Kernel size is determined using `cube_regrid.smoothing_kernel_size`

Warning: Smoothing is done in 3D to be maximally general. This can be exceedingly slow!

header : `Header`

Optional - can pass a header that both input images will be projected to match

Returns

image1,image2,[header] : `ndarray`, `ndarray`, `Header`

Two images projected into the same space, and optionally the header used to project them

Raises

ValueError :

Will raise an error if the axes are not consistent with a FITS cube, i.e. two spatial and one spectral axis.

See also:

`cube_regrid.regrid_fits_cube`
regrid a single cube This function performs a similar purpose and does the underlying work for `match_fits_cubes`, but it has a different call specification and returns an HDU

project_to_header

FITS_tools.match_images.**project_to_header**(*fitsfile*, *header*, *use_montage=True*, *quiet=True*, ***kwargs*)

Light wrapper of montage with hcongrid as a backup

kwargs will be passed to `hcongrid` if `use_montage==False`

Parameters

fitsfile : string

a FITS file name

header : `Header`

A fits Header instance with valid WCS to project to

use_montage : bool

Use montage or hcongrid (based on `map_coordinates`)

quiet : bool

Silence Montage's output

Returns

image : `ndarray`

image projected to header's coordinates

CHAPTER 9

FITS_tools.strip_headers Module

Functions

flatten_header(header[, delete])	Attempt to turn an N-dimensional fits header into a 2-dimensional header Turns all CRPIX[>2] etc.
speccen_header(header[, lon, lat])	Turn a cube header into a spectrum header, retaining RA/Dec vals where possible

flatten_header

FITS_tools.strip_headers.**flatten_header**(*header*, *delete=False*)

Attempt to turn an N-dimensional fits header into a 2-dimensional header Turns all CRPIX[>2] etc. into new keywords with prefix 'A'

header must be a Header instance

speccen_header

FITS_tools.strip_headers.**speccen_header**(*header*, *lon=None*, *lat=None*)

Turn a cube header into a spectrum header, retaining RA/Dec vals where possible (speccen is like flatten; spec-ify would be better but, specify? nah)

Assumes 3rd axis is velocity

CHAPTER 10

FITS_tools.load_header Module

Functions

get_cd(wcs[, n])	Get the value of the change in world coordinate per pixel across a linear axis.
load_data(data[, extnum])	Attempt to load a header specified as a header, a string pointing to a FITS
load_header(header[, extnum])	Attempt to load a header specified as a header, a string pointing to a FITS

get_cd

FITS_tools.load_header.**get_cd**(*wcs*, *n=1*)

Get the value of the change in world coordinate per pixel across a linear axis. Defaults to wcs.wcs.cd if present. Does not support rotated headers (e.g., with nonzero CDm_n where m!=n)

load_data

FITS_tools.load_header.**load_data**(*data*, *extnum=0*)

Attempt to load a header specified as a header, a string pointing to a FITS file, or a string pointing to a Header text file, or a string that contains the actual header

load_header

FITS_tools.load_header.**load_header**(*header*, *extnum=0*)

Attempt to load a header specified as a header, a string pointing to a FITS file, or a string pointing to a Header text file, or a string that contains the actual header

CHAPTER 11

FITS_tools.header_tools Module

Functions

enclosing_header(header1, header2[, wrapangle])	Find the smallest header that encloses both header1 and header2 in the
header_to_platescale(header, **kwargs)	Attempt to determine the spatial platescale from a
smoothing_kernel_size(hdr_from, hdr_to)	Determine the smoothing kernel size needed to convolve header_from to header_to without losing signal.
wcs_to_platescale(mywcs[, assert_square, ...])	Attempt to determine the spatial plate scale from a WCS

enclosing_header

FITS_tools.header_tools.**enclosing_header**(*header1*, *header2*, *wrapangle*=<*Quantity 180.0 deg*>)

Find the smallest header that encloses both header1 and header2 in the frame of header2

header_to_platescale

FITS_tools.header_tools.**header_to_platescale**(*header*, ***kwargs*)

Attempt to determine the spatial platescale from a Header

Parameters

header : Header

The FITS header to extract the platescale from

kwargs : dict

Passed to wcs_to_platescale. See that function for more details

Returns

platescale : float or Quantity

The platescale in degrees with attached units if use_units is True

smoothing_kernel_size

FITS_tools.header_tools.**smoothing_kernel_size**(*hdr_from*, *hdr_to*)

Determine the smoothing kernel size needed to convolve header_from to header_to without losing signal. Operates only in the spatial dimensions

wcs_to_platescale

FITS_tools.header_tools.**wcs_to_platescale**(*mywcs*, *assert_square=True*, *use_units=False*)

Attempt to determine the spatial plate scale from a WCS

Parameters

mywcs : WCS

The WCS instance to examine

assert_square : bool

Fail if the pixels are non-square

use_units : bool

Return a Quantity if True

Returns

platescale : float or Quantity

The platescale in degrees with attached units if use_units is True

Note: Do not edit this page - instead, place all documentation for the affiliated package inside packagename/

Python Module Index

f

Index

T

W

Z