smoothy Documentation

Release 0.1

CSRG/LIRAE

Jan 13, 2018

Contents:

CHAPTER 1

Core functions

Note: For Developers and Advanced Users

Functions used by UPI.

1.1 Data Analysis

smoothy.core.analysis.**denoise**(*data*, *threshold*)

Performs denoising of data cube, thresholding over the threshold value.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData or or astropy.nddata.NDData

Astronomical data cube.

threshold : float

Threshold value used for denoising.

Returns **result** : numpy.ndarray

Denoised (thresholded) astronomical data cube.

smoothy.core.analysis.**gaussian_function**(*mu*, *P*, *feat*, *peak*)

Generates an N-dimensional Gaussian using the feature matrix feat, centered at mu, with precision matrix P and with intensity peak.

Parameters **mu** : numpy.ndarray

Centers of gaussians array.

P : numpy.ndarray

Precision matrix.

feat : numpy.ndarray.

Features matrix.

peak : float

Peak value of the resulting evaluation.

Returns result: 2D numpy.ndarray

Returns the gaussian function evaluated at the value on feat.

smoothy.core.analysis.**integrate**(*data*, *mask=None*, *axis=0*)

Sums the slices of a cube of data given an axis.

Parameters **data** : (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

mask : numpy.ndarray (default = None)

axis : int (default=(0))

Returns A numpy array with the integration results.

smoothy.core.analysis.**rms**(*data*, *mask=None*)

Compute the RMS of data. If mask != None, then we use that mask.

Parameters **data** : (M,N,Z) numpy.ndarray or astropy.nddata.NDData or or astropy.nddata.NDDataRef

Astronomical data cube.

mask : numpy.ndarray (default = None)

Returns RMS of the data (float)

1.2 Utils

smoothy.core.utilities.**add**(*data*, *flux*, *lower*, *upper*)

Adds flux to a sub-cube of an astronomical data cube.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData or or astropy.nddata.NDData

Astronomical data cube.

flux : numpy.ndarray

Flux added to the cube.

lower : tuple

Lower bound of the sub-cube to which flux will be added.

upper : tuple

Upper bound of the sub-cube to which flux will be added.

smoothy.core.utilities.**fix_limits**(*data*, *vect*)

Fix vect index to be inside data

Parameters **data** : numpy.ndarray or numpy.ma.MaskedArray

Astronomical data cube.

vect : tuple, list or numpy.ndarray

Array with the indexes to be fixed.

Returns **result** : numpy.ndarray

Fixed array of indexes.

smoothy.core.utilities.**fix_mask**(*data*, *mask*)

Parameters **data** : numpy.ndarray or numpy.ma.MaskedArray

Astronomical data cube.

mask : numpy.ndarray

Boolean that will be applied.

Returns **result** : numpy.ma.MaskedArray

Masked astronomical data cube.

smoothy.core.utilities.**index_features**(*data*, *lower=None*, *upper=None*)

Creates an array with indices in features format

smoothy.core.utilities.**matching_slabs**(*data*, *flux*, *lower*, *upper*)

Obtain the matching subcube inside the lower and upper points.

Parameters **data** : numpy.ndarray

First data cube

flux : numpy.ndarray

Second data cubse

lower : tuple

Lower coordinates for the subcube.

upper : tuple

Upper coordinates for the subcube.

Returns The subcube inside the lower and upper points that matches both data cube dimensions.

smoothy.core.utilities.**slab**(*data*, *lower=None*, *upper=None*)

Obtain the n-dimensional slab from lower to upper (i.e. slab is a vector of slices)

Parameters **data** : numpy.ndarray

Atronomical data cube.

lower : 3-tuple (default=None)

Lower coordinates for the subcube.

upper : 3-tuple (default=None)

Upper coordinates for the subcube.

Returns **result** : list

list of slices using lower and upper coordinates to create a subcube.

smoothy.core.utilities.**standarize**(*data*)

Standarize astronomical data cubes in the 0-1 range.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData or or astropy.nddata.NDData

Astronomical data cube.

Returns **result** : tuple

Tuple containing the standarized numpy.ndarray or astropy.nddata.NDData cube, the factor scale y_fact and the shift y_min.

smoothy.core.utilities.**unstandarize**(*data*, *a*, *b*)

Unstandarize the astronomical data cube: $a \cdot data + b$.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData or or astropy.nddata.NDData

Astronomical data cube.

a : float

Scale value.

b : float

Shift value.

Returns **result** : numpy.ndarray or astropy.nddata.NDData

Unstandarized astronomical cube.

CHAPTER 2

Input-Output (IO)

2.1 Visualization Functions

smoothy.io.graph.**visualize**(*data*, *wcs=None*, *unit=None*, *contour=False*)

Generic function to visualize data, line-plot for 1D and image for 2D.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical image

wcs : astropy.wcs.WCS

World Coordinate System from the image (not needed if contained in NDData)

unit : astropy.unit

Image units (not needed if contained in NDData)

contour : numpy.ndarray

For plotting Contourns

smoothy.io.graph.**visualize_image**(*data*, *wcs=None*, *unit=None*, *contour=False*)

Plot 2D astronomical data.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical image

wcs : astropy.wcs.WCS

World Coordinate System from the image (not needed if contained in NDData)

unit : astropy.unit

Image units (not needed if contained in NDData)

contour : numpy.ndarray

For plotting Contourns

smoothy.io.graph.**visualize_plot**(*data*, *wcs=None*, *unit=None*)
Plot 1D data for astronomical data.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData

Astronomical image

wcs : astropy.wcs.WCS

World Coordinate System from the image (not needed if contained in NDData)

unit : astropy.unit

Image units (not needed if contained in NDData)

smoothy.io.graph.**visualize_spectra**(*data*, *wcs=None*, *unit=None*, *velocities=False*)
Plot spectra from astronomical data.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data

wcs : astropy.wcs.WCS

World Coordinate System from the image (not needed if contained in NDData)

unit : astropy.unit

Image units (not needed if contained in NDData)

velocities: bool

Use spectral velocities

smoothy.io.graph.**visualize_volume**(*data*, *wcs=None*, *unit=None*)
Plot 3D astronomical data.

Parameters **data** : numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical cube

wcs : astropy.wcs.WCS

World Coordinate System from the image (not needed if contained in NDData)

unit : astropy.unit

Image units (not needed if contained in NDData)

2.2 FITS handling functions

smoothy.io.fits.**Data_to_HDU**(*cube*, *primary=False*)
Create a HDU object from an N-dimensional dataset.

Parameters **cube** : numpy.ndarray or astropy.nddata.NDData or or astropy.nddata.NDDataRef

Astronomical data cube.

primary : bool

Whether to pick the primary or an image HDU.

Returns result: HDU object with data from the data cube.

smoothy.io.fits.**HDU_to_Data**(*hdu*)
Create an N-dimensional dataset from an HDU component.

Parameters **hdu** : HDU object

HDU to transform into an N-dimensional dataset.

Returns result: astropy.nddata.NDDataRef with data from the HDU object.

smoothy.io.fits.**HDU_to_Table**(*hdu*)
Create a data table from a HDU component.

Parameters **hdu** : HDU object

HDU to transform into a data table.

Returns result: astropy.table.Table with data from the HDU.

smoothy.io.fits.**Table_to_HDU**(*tab*)
Create a HDU object from a data table.

Parameters **tab** : astropy.table.Table

Table to transform into a HDU object.

Returns result: HDU object with data from the data table.

smoothy.io.fits.**load_fits**(*filePath*, *primary=False*)
Loads a FITS file and converts it into an N-Dimensional Dataset.

Parameters **filepath** : path of the FITS file.

primary : bool

if True it gets only primmary data-cube.

Returns Primary NDData image or astropy table, and/or:

N-Dimensional Datasets and Astropy Tables lists

CHAPTER 3

User Programmatic Interface (UPI)

Functions to simplificate the programming task for standard users.

3.1 Axes Manipulation

smoothy.upi.axes.**axes_names**(*data*, *wcs=None*)

Get the axes's names.

Parameters **data** : (M,N) or (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

Returns result: numpy.ndarray

Numpy ndarray with the axes's names from the WCS.

smoothy.upi.axes.**axes_units**(*data*, *wcs=None*)

Get units of the axes

Parameters **data** : (M,N) or (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

Returns result: (M,N) or (M,N,Z) numpy.ndarray

Vector with the units of the axes

smoothy.upi.axes.**center**(*data*, *wcs=None*)
Get center of the data

Parameters data : (M,N) or (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

Returns result: astropy.units.quantity.Quantity

Center of the data

smoothy.upi.axes.**extent**(*data*, *wcs=None*, *region=None*)
Get the axes extent.

Parameters data : (M,N) or (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

region :(lower : (M,N) or (M,N,Z), upper

Start and End index in data (int tuples)

Returns result: (M, N) tuple of astropy.units.quantity.Quantity

Axes extent

smoothy.upi.axes.**features**(*data*, *wcs=None*, *region=None*)
Creates an array with WCS axes in features format

Parameters data : (M,N) or (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

region :(lower : (M,N) or (M,N,Z), upper

Start and End index in data (int tuples)

Returns result: astropy.table.Table

Table with WCS information of a section from the data.

smoothy.upi.axes.**opening**(*data*, *center*, *window*, *wcs=None*)
Field of view (center +- window) converted to indices

Parameters data : (M,N) or (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

center : astropy.units.quantity.Quantity

Center of the field of view in WCS.

window : astropy.units.quantity.Quantity

Window for the field in WCS.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

Returns

———-

result: ((M1,N1,Z1),(M2,N2,Z2)) tuple of tuple of ints

`smoothy.upi.axes.`**`resolution`**(*data*, *wcs=None*)

Get the resolution of data

Parameters data : (M,N) or (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

Returns result: (M,N) or (M,N,Z) numpy.ndarray

Resolution of the data

`smoothy.upi.axes.`**`spectral_velocities`**(*data*, *wcs=None*, *fqs=None*, *fqis=None*, *restfrq=None*)

Get the spectral velocities from frequencies fqs given a rest frequency (by default search for it in the WCS). If fqs is None, then frequencies indices (fqis) need to be given.

Parameters data : (M,N) or (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

fqs : astropy.units.quantity.Quantity

Array of frequencies with units.

fqis : list of integers

Array of frequencies indices

restfrq : astropy.units.quantity.Quantity

Rest frequency

Returns result: astropy.units.quantity.Quantity

Array of Spectral velocities.

3.2 Data Manipulation

`class` `smoothy.upi.data.`**`Data`**(*data*, *uncertainty=None*, *mask=None*, *wcs=None*, *meta=None*, *unit=None*, *copy=False*)

A generic represenation of astronomical n-dimensional data array. Extends NDData.

`axes_names`()

Get the axes's names.

Returns result: numpy.ndarray

Numpy ndarray with the axes's names from the WCS.

axes_units()
Get units of the axes

Returns result: (M,N) or (M,N,Z) numpy.ndarray

Vector with the units of the axes

center()
Get center of the data

Returns result: astropy.units.quantity.Quantity

Center of the data

extent(*region=None*)
Get the axes extent.

Parameters **region :(lower** : (M,N) or (M,N,Z), upper

Start and End index in data (int tuples)

Returns result: (M, N) tuple of astropy.units.quantity.Quantity

Axes extent

features(*region=None*)
Creates an array with WCS axea in features format

Parameters **region :(lower** : (M,N) or (M,N,Z), upper

Start and End index in data (int tuples)

Returns result: astropy.table.Table

Table with WCS information of a section from the data.

opening(*center*, *window*)
Field of view (center +- window) converted to indices

Parameters **center** : astropy.units.quantity.Quantity

Center of the field of view in WCS.

window : astropy.units.quantity.Quantity

Window for the field in WCS.

Returns result: ((M1,N1,Z1),(M2,N2,Z2)) tuple of tuple of ints

resolution()
Get the resolution of data

Returns result: (M,N) or (M,N,Z) numpy.ndarray

Resolution of the data

spectral_velocities(*fqs=None*, *fqis=None*, *restfrq=None*)
Get the spectral velocities from frequencies fqs given a rest frequency (by default search for it in the WCS). If fqs is None, then frequencies indices (fqis) need to be given.

Parameters **fqs** : astropy.units.quantity.Quantity

Array of frequencies with units.

fqis : list of integers

Array of frequencies indices

restfrq : astropy.units.quantity.Quantity

Rest frequency

Returns result: astropy.units.quantity.Quantity

Array of Spectral velocities.

3.3 Flux Manipulation

smoothy.upi.flux.**add**(*data*, *flux*, *lower=None*, *upper=None*, *wcs=None*, *unit=None*, *meta=None*, *mask=None*)

Create a new data with the new flux added.

Lower and upper are bounds for data. This operation is border-safe and creates a new object at each call.

Parameters **data** : (M,N) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

flux : float

Flux of data

lower : numpy.ndarray

upper : numpy.ndarray

Bounds for data

wcs : World Coordinate System data (http://docs.astropy.org/en/stable/wcs/)

mask : numpy.ndarray

mask for the data

unit : astropy.units.Unit

Astropy Unit (http://docs.astropy.org/en/stable/units/)

meta : FITS metadata

Returns NDDataRef: structure with new flux added

smoothy.upi.flux.**denoise**(*data*, *wcs=None*, *mask=None*, *unit=None*, *threshold=0.0*)

Simple denoising given a threshold (creates a new object)

Parameters **data** : (M,N) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

wcs : World Coordinate System data (http://docs.astropy.org/en/stable/wcs/)

mask : numpy.ndarray

mask for the data

unit : astropy.units.Unit

Astropy Unit (http://docs.astropy.org/en/stable/units/)

threshold : float

Returns NDDataRef: Data denoised

smoothy.upi.flux.**noise_level**(*data*, *mask=None*, *unit=None*)

Compute the RMS of data.

Parameters **data** : (M,N) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

mask : numpy.ndarray

mask for the data

unit : astropy.units.Unit

Astropy Unit (http://docs.astropy.org/en/stable/units/)

Returns **rms** : float

RMS of data

`smoothy.upi.flux.`**`standarize`**(*data*, *wcs=None*, *unit=None*, *mask=None*, *meta=None*)

Standarize data:

Parameters **data** : (M,N) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

wcs : World Coordinate System data (http://docs.astropy.org/en/stable/wcs/)

mask : numpy.ndarray

mask for the data

unit : astropy.units.Unit

Astropy Unit (http://docs.astropy.org/en/stable/units/)

meta : FITS metadata

Returns Standarized data where data = a * res + b

`smoothy.upi.flux.`**`unstandarize`**(*data*, *a*, *b*, *wcs=None*, *unit=None*, *mask=None*, *meta=None*)

Unstandarize data: res = a * data + b

Parameters **data** : (M,N) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

a : float

slope of straight

b : float

Intercept of straight

wcs : World Coordinate System data (http://docs.astropy.org/en/stable/wcs/)

mask : numpy.ndarray

mask for the data

unit : astropy.units.Unit

Astropy Unit (http://docs.astropy.org/en/stable/units/)

meta : FITS metadata

Returns NDDataRef: Unstandarized data: res = a * data + b

`smoothy.upi.flux.`**`world_gaussian`**(*data*, *mu*, *P*, *peak*, *cutoff*, *wcs=None*)

Creates a gaussian flux at mu position (WCS), with P shape, with a maximum value equal to peak, and with compact support up to the cutoff contour

Parameters **data** : (M,N) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

mu : float

P : tuple

Shape of result

peak : float

maximum value

cutoff :

wcs : World Coordinate System data (http://docs.astropy.org/en/stable/wcs/)

Returns Tuple of gaussian flux and borders

3.4 Data Statistics

`smoothy.upi.reduction.`**`moment0`**(*data*, *wcs=None*, *mask=None*, *unit=None*, *restfrq=None*)

Calculate moment 0 from a data cube.

Parameters **data** : (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use.

mask : numpy.ndarray

Mask for data.

unit : astropy.units.Unit

Astropy unit (http://docs.astropy.org/en/stable/units/).

restfrq : astropy.units.quantity.Quantity

Rest frequency

Returns result: astropy.nddata.NDDataRef

Moment 0 of the data cube

`smoothy.upi.reduction.`**`moment1`**(*data*, *wcs=None*, *mask=None*, *unit=None*, *restfrq=None*)

Calculate moment 1 from a data cube.

Parameters **data** : (M,N,Z) numpy.ndarray or astropy.nddata.NDData

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use

mask : numpy.ndarray

Mask for data.

unit : astropy.units.Unit

Astropy unit (http://docs.astropy.org/en/stable/units/)

restfrq : astropy.units.quantity.Quantity

Rest frequency

Returns result: astropy.nddata.NDData

Moment 1 of the data cube

smoothy.upi.reduction.**moment2**(*data*, *wcs=None*, *mask=None*, *unit=None*, *restfrq=None*)

Calculate moment 2 from a data cube.

Parameters **data** : (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use

mask : numpy.ndarray

Mask for data.

unit : astropy.units.Unit

Astropy unit (http://docs.astropy.org/en/stable/units/)

restfrq : astropy.units.quantity.Quantity

Rest frequency

Returns result: astropy.nddata.NDDataRef

Moment 2 of the data cube

smoothy.upi.reduction.**spectra**(*data*, *wcs=None*, *mask=None*, *unit=None*, *restrict=None*)

Parameters **data** : (M,N,Z) numpy.ndarray or astropy.nddata.NDData or astropy.nddata.NDDataRef

Astronomical data cube.

wcs : astropy.wcs.wcs.WCS

World Coordinate System to use

mask : numpy.ndarray

Mask for data.

unit : astropy.units.Unit

Astropy unit (http://docs.astropy.org/en/stable/units/)

restrict : boolean

Returns result: astropy.nddata.NDData

Moment 2 of the data cube

3.5 Formatting

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

T

U

V

W