This package provides data processing tools for working with STIS data.

Contents:
Perform basic 2-D calibration of STIS raw data.

Some calibration steps are relevant only for CCD or only for MAMA, and since an output file of calstis or basic2d may be used as the input, some steps may have already been done. Most calibration steps will not be done if they are not relevant or if they have already been done, regardless of the value of the calibration switch (e.g. flatcorr).

**Parameters**

*input: str* :
Name of the input raw file.

*output: str* :
Name of the output file, or “” (the default). If no name was specified, the output name will be constructed from the input name.

*outblev: str* :
Name of the output text file for blev info, or “” (the default).

* dqicorr: str* :
If “perform”, update the DQ array.

* atodcorr: str* :
The analog-to-digital correction is ignored because it was never implemented.

* blevcorr: str* :
If “perform”, subtract a bias level based on the overscan values. (CCD only.)

* dopcorr: str* :
If “perform”, convolve reference files (bpixtab, darkfile, flatfile) as needed with the Doppler shift offset throughout the exposure, if Doppler correction was done on-board. (MAMA only, because for the CCD Doppler correction is not done on-board.)

* lorcorr: str* :
If “perform”, bin high-res data to lo-res. (MAMA only.)
glincorr: str :
    If “perform”, correct for global non-linearity. (MAMA only.)

lfigcorr: str :
    If “perform”, flag local non-linearity. (MAMA only.)

biascorr: str :
    If “perform”, subtract the bias image. (CCD only.)

darkcorr: str :
    If “perform”, subtract the dark image, scaled by the exposure time and possibly also a
    temperature-dependent factor.

flatcorr: str :
    If “perform”, divide by the flat field image.

shadcorr: str :
    The shutter shading correction is ignored because it was never implemented.

photcorr: str :
    If “perform”, determine the photometric parameters and populate keywords PHOT-
    FLAM, PHOTZPT, PHOTPLAM and PHOTBW. (Imaging only.)

statflag: bool :
    If True, compute statistics for image arrays and update keywords.

darkscale: str :
    This may be used to override the time and/or temperature dependent scale factor that
    would normally be applied to the dark image before subtracting from the raw data. It’s
    a string rather than a float in order to accept a different scale factor for each image set
    in the input data. calstis reads the value or values (separated by blanks) from the string,
    and if the value is greater than zero, it will be used instead of the value determined from
    the temperature and time. (CCD or NUV-MAMA only.)

verbose: bool :
    If True, calstis will print more info.

timestamps: bool :
    If True, calstis will print the date and time at various points during processing.

trailer: str :
    If specified, the standard output and standard error will be written to this file instead of
    to the terminal. Note, however, that if print_version or print_revision is specified, the
    value will be printed to the terminal, and any name given for the trailer will be ignored.

print_version: bool :
    If True, calstis will print the version number (a string) and then return 0.

print_revision: bool :
    If True, calstis will print the full version string and then return 0.

Returns
    status: int :
0 is OK. 1 is returned if cs1.e (the calstis host executable) returned a non-zero status. If 
verbose is True, the value returned by cs1.e will be printed. 2 is returned if the specified 
input file or files were not found, or if there is a mismatch between the number of input, 
output, and/or outblev files specified.

```python
stistools.basic2d.getHelpAsString(fulldoc=True)
    Return documentation on the basic2d function.

stistools.basic2d.main(args)

stistools.basic2d.prtOptions()
    Print a list of command-line options and arguments.

stistools.basic2d.run(configobj=None)
    TEAL interface for the basic2d function.
```
stistools.calstis.calstis(input, wavecal='', outroot='', savetmp=False, verbose=False, timestamps=False, trailer='', print_version=False, print_revision=False)

Calibrate STIS data.

Parameters

input: str :
Name of the input file.

wavecal: str :
Name of the input wavecal file, or "" (the default). This is only needed if the name is not the “normal” name (rootname.wav.fits).

outroot: str :
Root name for the output files, or "" (the default). This can be a directory name, in which case the string must end in ‘/’.

savetmp: bool :
True if calstis should not delete temporary files.

verbose: bool :
If True, calstis will print more info.

timestamps: bool :
If True, calstis will print the date and time at various points during processing.

trailer: str :
If specified, the standard output and standard error will be written to this file instead of to the terminal. Note, however, that if print_version or print_revision is specified, the value will be printed to the terminal, and any name given for the trailer will be ignored.

print_version: bool :
If True, calstis will print the version number (a string) and then return 0.

print_revision: bool :
If True, calstis will print the full version string and then return 0.

Returns
status: int :

0 is OK. 1 is returned if cs0.e (the calstis host executable) returned a non-zero status. If verbose is True, the value returned by cs0.e will be printed. 2 is returned if the specified input file or files were not found.

```
stistools.calstis.getHelpAsString(fulldoc=True)
```

Return documentation on the calstis function.

```
stistools.calstis.main(args)
```

```
stistools.calstis.prtOptions()
```

Print a list of command-line options and arguments.

```
stistools.calstis.run(configobj=None)
```

TEAL interface for the calstis function.
A Python module for aligning the spectra in different flat-fielded images of an IMSET. These files can then be combined with along-the-slit dithering to reject hot pixels and cosmic rays. The POSTARG2 keyword is used to determine the number of rows to be shifted.

```python
stistools.sshift.shiftimage(infile, outfile, shift=0)
    Shift each image extension of an input file by N rows and write the new image extension to the output file.
```

```python
stistools.sshift.sshift(input, output=None, shifts=None, platescale=None, tolerance=None)
    Align spectra from different images of an imset.
```

**Parameters**

**input**: list

A list of input filenames. These must be STIS flat-fielded (_flt) image FITS files. This argument will accept a single filename or a list of filenames.

**shifts**: list, optional

A list of integers indicating the number of rows to shift each image of each file in the cross-dispersion (Y-) direction.

**platescale**: float, optional

The size of a pixel in arcseconds. Used to convert the value of the POSTARG2 keyword to pixels.

**tolerance**: float, optional

The allowed difference between calculated shifts and integer pixel shifts (fraction of pixel).

**Returns**

**output**: list, optional

A list of output filenames. The number of output filenames must match the number of input filenames. If no output is given, then the _flt substring of the input file is replaced by the _sfl substring to create an output file. This option will accept a single filename or a list of filenames.

**Notes**

**Author**:  

- Paul Barrett (STScI)
STISNOISE

stistools.stisnoise.gauss (x, x0, dx, ymax)

stistools.stisnoise.medianfilter (time_series, width)

stistools.stisnoise.stisnoise (infile, exten=1, outfile=None, dc=1, verbose=1, boxcar=0, wipe=None, window=None)

Computes an FFT on STIS CCD frames to evaluate fixed pattern noise.

Fixed pattern noise is most obvious in a FFT of bias frames. Optional filtering to correct the fixed pattern noise is provided through keywords boxcar, wipe, and window. Filtered data can be saved as an output file.

Parameters

infile : string
  STIS FITS file

exten : int, optional
  fits extension to be read

dc : int, optional
  the power in the first freq bin is set to zero for better plotting of the power spectrum.

verbose : int, optional [Default: 1]
  set to 0 if you do not want brief information about each image.

boxcar : int
  width of boxcar smoothing to be applied.

wipe : ndarray
  a 3-element array, specifying how to modify the data in frequency space. If set, the image is converted to a 1-D time series, fourier transformed to frequency space, modified, inverse transformed back to time space, and converted back to a 2-D image. The first and second elements specify the range in frequencies to be scaled (in hz), and the third element specifies the scaling factor (should be 0-1).

window : ndarray
  a 3 element array, specifying how to modify the data in frequency space. The first element is the center of the window (in hz). The second element is the width of the window (in hz). The third element controls the tapering of the window - it is the scale (in hz) of the tapering width. Specifically, a square bandstop is convolved with a gaussian having the FWHM given by the third parameter.
outfile : string, optional
    name of filtered image file

Returns
noise_terms : tuple of arrays
    A tuple containing the arrays; namely, the arrays:
    
    freq = frequency in power spectrum (hz)
    magn = magnitude in power spectrum

Notes
Authors:

• Original algorithm: Thomas M. Brown (STScI)
• Python version: Paul Barrett (STScI)

stistools.stisnoise.windowfilter(time_series, image_type, sst, freqpeak, width, taper)

stistools.stisnoise.wipefilter(time_series, image_type, sst, freqmin, freqmax, scale)
Refine a STIS trace table.

- A trace is generated from the science file and a trace center is computed.
- The two traces bracketing the trace center are extracted from the trace table and interpolated.
- The correction is computed as the difference between the linear fit to the science and interpolated traces.
- The correction is applied to all traces in the trace file for that particular OPT_ELEM and CENWAVE.
- A new trace table is written to the current directory and the relevant keywords are updated in the header of the input file.

**Usage**

Simple example of running `mktrace` on a STIS file named `file.fits`:

```python
>>> import mktrace
>>> mktrace.mktrace('file.fits', [tracecen=509.4], [weights=[(x1,x2),(x3,x4)])
```

**Authors**

- Author (IDL): Linda Dressel
- Python version: Nadia Dencheva

```python
class stistools.mktrace.Trace(file, kwinfo)
Trace class for a crj or flt file.
```

**Notes**

- `tr=Trace(file)` file is a crj or flt file.
- `opt_elem`, `cenwave`, `sporder` are read from the header of the science file.
- `a2center` is `a2center` of the trace generated from the science file.
- `tr_ind` is the index of the row in the trace file which brackets from below `a2center` as computed from the generated trace.
- `tr.readTrace(tr_ind)`
- `a2center = tr.generateTrace(...)`

```python
gFitTrace(specimage, y1, y2)
Fit a gaussian to each column of an image.
```

```python
generateTrace(data, kwinfo, tracecen=0.0, wind=None)
Generates a trace from a science file.
```
**getTraceInd** *(a2center)*  
Finds the first trace in the trace table whose A2CENTER is larger than the specified a2center

**openTraceFile** *(filename)*  
Returns a spectrum trace table

**readTrace** *(tr_ind)*  
reads the specified row from the 1dttab.fits

**writeTrace** *(fname, sciline, refline, interp_trace, trace1024, tr_ind, a2disp_ind)*  
The ‘writeTrace’ method performs the following steps:
• Adds sciline-refline to all traces with the relevent OPT_ELEM, CENWAVE and SPORDER.
• Writes the new trace table to the current directory.
• Updates the SPTRCTAB keyword in the header to point to the new table.
• Writes out fits files with the
  – science trace - ‘_sci’
  – the fit to the science trace - ‘_scifit’
  – the interpolated trace - ‘_interp’
  – the linear fit to the interpolated trace - ‘_interpfit’

**stistools.mktrace.getKWInfo** *(hdr0, hdr1)*

**stistools.mktrace.interp** *(y, n)*  
Given a 1D array of size m, interpolates it to a size n (m<n).

**stistools.mktrace.iterable** *(v)*

**stistools.mktrace.mktrace** *(fname, tracecen=0.0, weights=None)*  
Refine a stis spectroscopic trace.

**stistools.mktrace.trace_interp** *(tr1, tr2, cen)*
stistools.evaldisp.evalDisp(coeff, wl)
Return the pixel corresponding to wavelength wl.

Parameters
coeff: array_like object
a list of eight elements containing the dispersion coefficients as read from a STIS dsp.fits table
wl: float or ndarray
a single wavelength or an array (numarray) of wavelengths, in Angstroms

Returns
pix_number: float or ndarray
the pixel number (or array of pixel numbers) corresponding to the input wavelength(s);
note that these are zero indexed

Notes
The expression in the calstis code is:

\[ x = \text{coeff[0]} + \]
\[ \text{coeff[1]} \times m \times \text{wl} + \]
\[ \text{coeff[2]} \times \text{m}^{\times 2} \times \text{wl}^{\times 2} + \]
\[ \text{coeff[3]} \times m + \]
\[ \text{coeff[4]} \times \text{wl} + \]
\[ \text{coeff[5]} \times \text{m}^{\times 2} \times \text{wl} + \]
\[ \text{coeff[6]} \times m \times \text{wl}^{\times 2} + \]
\[ \text{coeff[7]} \times \text{m}^{\times 3} \times \text{wl}^{\times 3} \]

This version of the function to evaluate the dispersion relation assumes that the grating is first order, i.e. \( m = 1 \). The dispersion coefficients give one-indexed pixel coordinates (reference pixels), but this function converts to zero-indexed pixels.

stistools.evaldisp.newton(x, coeff, cenwave, niter=4)
Return the wavelength corresponding to pixel x.

The dispersion solution is evaluated iteratively, and the slope (dispersion) for Newton’s method is determined numerically, using a difference in wavelength of one Angstrom. Note that the evalDisp in this file assumes that the grating is first order.

Parameters
x: float or ndarray
a single pixel number or an array of pixel numbers
**coeff**: array_like object

- a list of eight elements containing the dispersion coefficients as read from a STIS _dsp.fits table

**cenwave**: int or float

- central wavelength, in Angstroms

**niter**: int

- number of iterations

**Returns**

**wavelength**: float or ndarray

- a single wavelength or an array (numarray) of wavelengths, in Angstroms
stistools.wavelen.adjust_disp(ncoeff, coeff, delta_offset1, shifta1, inang_info, delta_tan, delta_row, binaxis1)

Adjust the dispersion coefficients.
The changes to the coefficients are for the incidence angle correction, the offset from the SHIFTA1 keyword, and the tilt of the slit. The coefficients will be modified in-place.

Parameters
ncoeff : int
    number of dispersion coefficients
coeff : ndarray of float64
    array of dispersion coefficients, modified in-place
delta_offset1 : float
    incidence angle offset in degrees
shifta1 : float
    MSM offset (ref. pixels) in the dispersion direction
delta_tan : float
    difference in tangents of slit angle and ref angle
delta_row : float
    difference between current row number and CRPIX2
binaxis1 : float
    binning factor in dispersion direction
inang_info : rec_array
    rows from the incidence-angle table

stistools.wavelen.compute_wavelengths(shape, phdr, hdr, helcorr)

Compute a 2-D array of wavelengths, one value for each image pixel.

Parameters
shape : tuple of two ints
    the number of rows and columns in the output image
phdr : pyfits Header object
    primary header
hdr : pyfits Header object
    extension header
helcorr : string
    “PERFORM” if heliocentric correction should be done

Returns
wavelengths : ndarray of float64
    an array of wavelengths, of the same shape (nrows, ncols) as the output image

stistools.wavelen.get_delta_offset1(apdestab, aperture, ref_aper)
Get the incidence angle offset.

Parameters
apdestab : string
    name of the aperture description table
aperture : string
    aperture (slit) name
ref_aper : string
    name of the reference aperture, the one that was used to calculate the dispersion relation

Returns
angle : float
    incidence angle offset in degrees
**stistools.wx2d**.apply_trace(image, a2center, a2displ, subdiv, offset=0.0, shifta2=0.0, extname='SCI')

Add together ‘subdiv’ rows of ‘image’, following the trace.

**Parameters**

- **image** : ndarray
  - input 2-D image array, oversampled by ‘subdiv’ in axis 0
- **a2center** : ndarray
  - 1-D array of Y locations
- **a2displ** : ndarray
  - array of traces, one for each a2center; the length of each trace must be the same as the number of columns in the input image
- **subdiv** : int
  - number of rows to add together
- **offset** : float
  - offset of the first row in ‘image’ from the beginning of the data block in the original file, needed for trace
- **shifta2** : float
  - offset of the row from nominal (from shifta2 keyword)
- **extname** : string
  - which type of extension (SCI, ERR, DQ)?

**Returns**

- **x2d** : ndarray
  - resampled 2-D image array

**Notes**

The function value is a 2-D array containing the resampled image. This is binned by subdiv in Y (axis 0), after shifting by trace (multiplied by subdiv).

For extname = “ERR” the result differs in these ways:

1. fractions of pixels at the endpoints of the extraction region are not included
2. the values are combined as the average of the sum of the squares
For extname = “DQ” the result differs in these ways:

1. the output is type int16
2. the output values are nominally the same as the input, while for SCI the output are subdiv times larger than the input
3. fractions of pixels at the endpoints of the extraction region are not included
4. the values are combined via bitwise OR rather than an average or sum

```
stistools.wx2d.bin_traces(a2displ, binaxis1, ltv)
```

bin the traces by the factor binaxis1

**Parameters**

- `a2displ`: ndarray
  an array of one or more arrays of Y displacements (traces)
- `binaxis1`: int
  binning factor in the dispersion axis
- `ltv`: float
  offset in the dispersion axis (one indexing)

**Returns**

- `a2displ`: ndarray
  an array of traces (a2displ), but with the trace arrays binned and shorter by the factor binaxis1

```
stistools.wx2d.extract(image, locn, subdiv)
```

Add together ‘subdiv’ rows of ‘image’, centered on ‘locn’.

**Parameters**

- `image`: ndarray
  input array, oversampled by ‘subdiv’ in axis 0
- `locn`: ndarray
  a 1-D array giving the location at which to extract; an integer value corresponds to the center of the pixel. The length must be the same as the number of columns in the input image.
- `subdiv`: int
  number of rows to add together

**Returns**

- `spec`: ndarray
  a 1-D array containing the extracted row

```
stistools.wx2d.extract_err(image, locn, subdiv)
```

Average ‘subdiv’ rows of ‘image’, centered on ‘locn’.

**Parameters**

- `image`: ndarray
  input array, oversampled by ‘subdiv’ in axis 0
- `locn`: ndarray
  a 1-D array containing the location at which to extract
a 1-D array giving the location at which to extract; an integer value corresponds to the center of the pixel

subdiv : int
    number of rows to add together

Returns
spec : ndarray
    a 1-D array containing the extracted row

Notes
This takes the square root of the average of the squares, intended to be used for interpolating the ERR array. Fractions of pixels at the upper and lower edges are excluded.

stistools.wx2d.extract_i16(image, locn, subdiv)
Bitwise OR ‘subdiv’ rows of ‘image’, centered on ‘locn’.

Parameters
image : ndarray
    input array, oversampled by ‘subdiv’ in axis 0
locn : ndarray
    a 1-D array giving the location at which to extract; an integer value corresponds to the center of the pixel
subdiv : int
    number of rows to add together

Returns
spec : ndarray
    a 1-D array containing the extracted row

stistools.wx2d.get_trace(tracefile, phdr, hdr)
Read 1-D traces from the 1dt table (sptrctab).

Parameters
tracefile : string or array
    either a trace array or the name of a FITS 1dt table
phdr : pyfits Header object
    primary header of input file
hdr : pyfits Header object
    extension header of input image (for binning info and time of exposure)

Returns
trace_arrays : tuple of 2 arrays
    a pair of arrays, one is the Y location at the middle column, and the other is an array of trace arrays

Notes
If ‘tracefile’ is already a trace array, it will just be returned, together with an arbitrary Y location of 0 (because that will always be within the image).
opt_elem and cenwave are criteria for selecting the relevant rows from the 1dt table. There will normally be several rows that match, and they should have different values of the Y location; the output list will be sorted on Y location.

**stistools.wx2d.interpolate_trace** *(a2center, a2displ, y, length)*
Interpolate within the array of traces, and return a trace.

**Parameters**
- **a2center**: ndarray
  - array of Y locations
- **a2displ**: ndarray
  - array of traces, one for each element of a2center
- **y**: float
  - Y location on the detector
- **length**: int
  - length of a trace; needed only if traces is empty

**stistools.wx2d.inv_avg_interp** *(order, image)*

**stistools.wx2d.inv_haar** *(image)*

**stistools.wx2d.kd_apply_trace** *(image, a2center, a2displ, offset=0.0, shiffa2=0.0)*
Kris Davidson’s resampling algorithm, following the trace.

**Parameters**
- **image**: ndarray
  - input 2-D image array
- **a2center**: ndarray
  - array of Y locations
- **a2displ**: ndarray
  - array of traces, one for each a2center; the length of each trace must be the same as the number of columns in ‘image’
- **offset**: float
  - offset of the first row in ‘image’ from the beginning of the data block in the original file, needed for trace
- **shifta2**: float
  - offset of the row from nominal (from shiffa2 keyword)

**Returns**
- **x2d**: ndarray
  - 2-D array containing the resampled image

**stistools.wx2d.kd_resampling** *(img, errimg, original_nrows, nrows, ncols, rows, a2center, a2displ, offset, shifta2)*
Apply Kris Davidson’s resampling method.

**Parameters**
- **img**: ndarray
SCI image array (could be a subset of full image)

errimg : ndarray
ERR image array (could be a subset of full image)

original_nrows : int
number of image lines (NAXIS2) in input image

nrows : int
number of image lines in subset

ncols : int
number of image columns (NAXIS1)

rows : tuple
tuple giving the slice of rows to process

a2center : ndarray
1-D array of Y locations

a2displ : ndarray
array of traces, one for each a2center; the length of each trace must be the same as the
number of columns in the input image

offset : float
offset of the first row in ‘image’ from the beginning of the data block in the original file,
needed for trace

shifta2 : float
offset of the row from nominal (from shifta2 keyword)

Returns
img_arr : tuple
the image and error arrays (to replace the input img and errimg)

stistools.wx2d.polynomial(x, y, z, n)
used for interpolation

Parameters
x : ndarray
the integer values from 0 through n-1 inclusive (but float64)
y : ndarray
a 2-D array, axis 0 of length n
z : float
n / 2.

n : int
1 + order of polynomial fit

stistools.wx2d.stis_psf(x, a)
Evaluate the cross-dispersion PSF at x.
Parameters

\textbf{x} : float
     offset in pixels from the center of the profile

\textbf{a} : float
     a measure of the width of the PSF

Returns

\textbf{val} : float
     the PSF evaluated at x

\textbf{stistools.wx2d.trace_name\( (\text{trace}, \text{phdr}) \)}}

Return the 1dt table name or array.

Parameters

\textbf{trace} : string or array or None
     if trace is None the header keyword SPTRCTAB will be gotten from phdr; else if this is a string it should be the name of a trace file (possibly using an environment variable); otherwise, it should be a trace, in which case it will be returned unchanged

\textbf{phdr} : pyfits Header object
     primary header, used only if trace is None

Returns

\textbf{tracefile} : string or array
     name of a trace file (with environment variable expanded), or an actual trace array

\textbf{stistools.wx2d.wavelet_resampling\( (\text{hdu}, \text{img}, \text{errimg}, \text{original_nrows}, \text{nrows}, \text{ncols}, \text{rows}, \text{a2center}, \text{a2displ}, \text{offset}, \text{shifta2}, \text{imset}, \text{order}, \text{subdiv}, \text{psf_width}, \text{subsampling}, \text{convolved}) \)}}

Resample img and errimg using wavelets.

Parameters

\textbf{hdu} : pyfits header/data unit object
     header/data unit for a SCI extension

\textbf{img} : ndarray
     SCI image array (could be a subset of full image)

\textbf{errimg} : ndarray
     ERR image array (could be a subset of full image)

\textbf{original_nrows} : int
     number of image lines (NAXIS2) in input image

\textbf{nrows} : int
     number of image lines in subset

\textbf{ncols} : int
     number of image columns (NAXIS1)

\textbf{rows} : tuple
     tuple giving the slice of rows to process

\textbf{a2center} : ndarray
1-D array of Y locations

**a2displ**: ndarray

array of traces, one for each a2center; the length of each trace must be the same as the number of columns in the input image

**offset**: float

offset of the first row in ‘image’ from the beginning of the data block in the original file, needed for trace

**shifta2**: float

offset of the row from nominal (from shifta2 keyword)

**imset**: int

number of the current image set (keyword EXTVER)

**order**: int

polynomial order

**subdiv**: int

number of subpixels per input pixel

**psf_width**: float

width of PSF for convolution (e.g. 1.3);

**subsampled**: string or None

name of the output file with the subsampled image

**convolved**: string or None

name of the output file with the convolved image

**Returns**

**img_arr**: tuple of ndarrays

the image and error arrays (to replace the input img and errimg)

```python
stistools.wx2d.wx2d (input, output, wavelengths=None, helcorr='', algorithm='wavelet', trace=None, order=7, subdiv=8, psf_width=0.0, rows=None, subsampled=None, convolved=None)
```

Resample the input, correcting for geometric distortion.

**Parameters**

**input**: string

name of input file containing an image set

**output**: string

name of the output file

**wavelengths**: string, optional [Default: None]

name of the output file for wavelengths

**helcorr**: string

specify “perform” or “omit” to override header keyword

**algorithm**: {'wavelet', 'kd'}

algorithm to use in resampling the input
trace : string or array, or None
    trace array, or name of FITS table containing trace(s)

order : int [Default: 7]
    polynomial order (an odd number, e.g. 5 or 7)

subdiv : int [Default: 8]
    number of subpixels (a power of 2, e.g. 8 or 16)

psf_width : float [Default: 0.]
    width of PSF for convolution (e.g. 1.3); 0 means no convolution

rows : tuple, optional [Default: None]
    a tuple giving the slice of rows to process; output values in all other rows will be set to zero. The default of None means all rows, same as (0, 1024)

subsampled : string, optional [Default: None]
    name of the output file with the subsampled image

convolved : string, optional [Default: None]
    name of the output file with the convolved image

stistools.wx2d.wx2d_imset (ft, imset, output, wavelengths, helcorr, algorithm, tracefile, order, subdiv, psf_width, rows, subsampled, convolved)

Resample one image set, and append to output file(s).

Parameters

    ft : HDUList
        pyfits HDUList object for the input file

    imset : int
        one-indexed image set number

    output : string
        name of the output file

    wavelengths : string or None
        name of the output file for wavelengths

    helcorr : {'perform', 'omit'}
        specify “perform” or “omit” to override header keyword

    algorithm : {'wavelet','kd'}
        algorithm to use to process input

    tracefile : string or array
        trace array, or name of FITS table containing trace(s)

    order : int
        polynomial order

    subdiv : int
        number of subpixels

    psf_width : float
width of PSF for convolution

**rows** : tuple

  a tuple giving the slice of rows to process

**subsampled** : string, or None

  name of the output file with the subsampled image

**convolved** : string, or None

  name of the output file with the convolved image
stistools.radialvel.earthVel(mjd)
Compute and return the velocity of the Earth at the specified time.

This function computes the Earth’s orbital velocity around the Sun in celestial rectangular coordinates. The expressions are from the Astronomical Almanac, p C24, which gives low precision formulas for the Sun’s coordinates. We’ll apply these formulas directly to get the velocity of the Sun relative to the Earth, then we’ll convert to km per sec and change the sign to get the velocity of the Earth.

Parameters
mjd : float
time, Modified Julian Date

Returns
vel : ndarray
the velocity vector of the Earth around the Sun, in celestial coordinates (shape=(3,), ndtype=float64)

Notes
We get the velocity of the Sun relative to the Earth as follows:

The velocity in the ecliptic plane with the X-axis aligned with the radius vector is:

- \( V_x = \text{radius} \cdot \text{dot} \)
- \( V_y = \text{radius} \cdot \text{elong} \cdot \text{dot} \)
- \( V_z = 0 \)

where:
- \( \text{radius} \) is the radial distance from Earth to Sun
- \( \text{elong} \) is the ecliptic longitude of the Sun
- \( \text{eps} \) is the obliquity of the ecliptic
- \( \_\text{dot} \) means the time derivative

Rotate in the XY-plane by \( \text{elong} \) to get the velocity in ecliptic coordinates:

\[
\begin{align*}
\text{radius} \cdot \text{dot} \cdot \cos(\text{elong}) - \text{radius} \cdot \text{elong} \cdot \text{dot} \cdot \sin(\text{elong}) \\
\text{radius} \cdot \text{dot} \cdot \sin(\text{elong}) + \text{radius} \cdot \text{elong} \cdot \text{dot} \cdot \cos(\text{elong}) \\
0
\end{align*}
\]

Rotate in the YZ-plane by \( \text{eps} \) to get the velocity in equatorial coordinates:
\[
\begin{align*}
\text{radius} \cdot \dot{\text{cos}} (\text{elong}) & - \text{radius} \cdot \text{elong} \cdot \dot{\text{sin}} (\text{elong}) \\
(\text{radius} \cdot \dot{\text{sin}} (\text{elong}) + \text{radius} \cdot \text{elong} \cdot \dot{\text{cos}} (\text{elong})) & \cdot \text{cos} (\text{eps}) \\
(\text{radius} \cdot \dot{\text{sin}} (\text{elong}) + \text{radius} \cdot \text{elong} \cdot \dot{\text{cos}} (\text{elong})) & \cdot \text{sin} (\text{eps})
\end{align*}
\]

\textbf{stistools.radialvel.precess} \((mjd, \text{target})\)

Precess target coordinates from J2000 to the date \(mjd\).

\textbf{Parameters}

\begin{itemize}
  \item \texttt{mjd} : float
    \begin{itemize}
      \item time, Modified Julian Date
    \end{itemize}
  \item \texttt{target} : array_like object
    \begin{itemize}
      \item unit vector pointing toward the target, J2000 coordinates
    \end{itemize}
\end{itemize}

\textbf{Returns}

\begin{itemize}
  \item \texttt{vector} : ndarray
    \begin{itemize}
      \item the target vector (or matrix) precessed to \(mjd\) as an array object of type float64 and the same shape as \(target\), i.e. either (3,) or (n,3)
    \end{itemize}
\end{itemize}

\textbf{Notes}

\(target\) can be a single vector, e.g. \([x0, y0, z0]\), or it can be a 2-D array; in the latter case, the shape should be (n,3):

\[
\text{target} = \begin{bmatrix}
  [x0, x1, x2, x3, x4], \\
  [y0, y1, y2, y3, y4], \\
  [z0, z1, z2, z3, z4]
\end{bmatrix}
\]

The algorithm used in this function was based on [R1] and [R2].

\textbf{References}

[R1], [R2]

\textbf{stistools.radialvel.radialVel} \((\text{ra} \_\text{targ}, \text{dec} \_\text{targ}, \text{mjd})\)

Compute the heliocentric velocity of the Earth.

This function computes the radial velocity of a target based on the Earth’s orbital velocity around the Sun. The space motion of the target is not taken into account. That is, the radial velocity is just the negative of the component of the Earth’s orbital velocity in the direction toward the target.

\textbf{Parameters}

\begin{itemize}
  \item \texttt{ra} \_\texttt{targ} : float
    \begin{itemize}
      \item right ascension of the target (degrees)
    \end{itemize}
  \item \texttt{dec} \_\texttt{targ} : float
    \begin{itemize}
      \item declination of the target (degrees)
    \end{itemize}
  \item \texttt{mjd} : float
    \begin{itemize}
      \item Modified Julian Date at the time of observation
    \end{itemize}
\end{itemize}

\textbf{Returns}

\begin{itemize}
  \item \texttt{radial} \_\texttt{vel} : float
    \begin{itemize}
      \item the radial velocity in km/s
    \end{itemize}
\end{itemize}
stistools.r_util.expandFileName(filename)

Expand environment variable in a file name.

If the input file name begins with either a Unix-style or IRAF-style environment variable (e.g. $lref/name_dqi.fits or lref$name_dqi.fits respectively), this routine expands the variable and returns a complete path name for the file.

**Parameters**

filename : string

a file name, possibly including an environment variable

**Returns**

fullname : string

the file name with environment variable expanded

stistools.r_util.interpolate(x, values, xp)

Interpolate.

Linear interpolation is used. If the specified independent variable value xp is outside the range of the array x, the first (or last) value in values will be returned.

**Parameters**

x : a sequence object, e.g. an array, int or float

array of independent variable values

values : a sequence object, e.g. an array (not character)

array of dependent variable values

xp : int or float

independent variable value at which to interpolate

**Returns**

interp_vals : the same type as one element of values

linearly interpolated value
stistools.gettable.getTable(table, filter, sortcol=None, exactly_one=False, at_least_one=False)

Return row(s) of a table that match the filter.

Rows that match every item in the filter (a dictionary of column_name=value) will be returned. If the value in the table is STRING_WILDCARD or INT_WILDCARD (depending on the data type of the column), that value is considered to match the filter for that column. Also, for a given filter key, if the corresponding value in the filter is STRING_WILDCARD, the test on filter will be skipped for that key (i.e. a wildcard filter element matches any row).

If more than one row matches the filter, there is an option to sort these rows based on the values of one of the table columns.

It is an error if exactly_one or at_least_one is True but no row matches the filter. A warning will be printed if exactly_one is True but more than one row matches the filter.

Parameters
- **table** : string
  name of the reference table
- **filter** : dict
  each key is a column name, and the corresponding value is a possible table value in that column
- **sortcol** : string
  the name of a column on which to sort the table rows (if there is more than one matching row), or None to disable sorting
- **exactly_one** : bool
  set this to True if there must be one and only one matching row
- **at_least_one** : bool
  set this to True if there must be at least one matching row

Returns
- **match_rows** : rec_array
  an array of the rows of the table that match the filter; note that if only one row matches the filter, the function value will still be an array

stistools.gettable.rotateTrace(trace_info, expstart)

Rotate a2displ, if MJD and DEGPERYR are in the trace table.
Parameters

\texttt{trace\_info} : \texttt{rec\_array}

an array of the relevant rows of the table; the A2DISPL column will be modified in-place if the MJD and DEGPERYR columns are present

\texttt{expstart} : float

exposure start time (MJD)

\texttt{stistools.gettable.sortrows} \texttt{(rowdata, sortcol, ascend=True)}

Return a copy of rowdata, sorted on sortcol.
INDICES AND TABLES

- genindex
- modindex
- search
BIBLIOGRAPHY

PYTHON MODULE INDEX

S
stistools.basic2d, 3
stistools.calstis, 7
stistools.evaldisp, 15
stistools.gettable, 33
stistools.mktrace, 13
stistools.r_util, 31
stistools.radialvel, 29
stistools.sshift, 9
stistools.stisnoise, 11
stistools.wavelen, 17
stistools.wx2d, 19
S

stistools.basic2d, 3
stistools.calstis, 7
stistools.evaldisp, 15
stistools.gettable, 33
stistools.mktrace, 13
stistools.r_util, 31
stistools.radialvel, 29
stistools.sshift, 9
stistools.stisnoise, 11
stistools.wavelen, 17
stistools.wx2d, 19
INDEX

A
adjust_disp() (in module stistools.wavelen), 17
apply_trace() (in module stistools.wx2d), 19

B
basic2d() (in module stistools.basic2d), 3
bin_traces() (in module stistools.wx2d), 20

C
calstis() (in module stistools.calstis), 7
compute_wavelengths() (in module stistools.wavelen), 17

E
earthVel() (in module stistools.radialvel), 29
evalDisp() (in module stistools.evaldisp), 15
expandFileName() (in module stistools.r_util), 31
extract() (in module stistools.wx2d), 20
extract_err() (in module stistools.wx2d), 20
extract_i16() (in module stistools.wx2d), 21

G
gauss() (in module stistools.stisnoise), 11
generateTrace() (stistools.mktrace.Trace method), 13
get_delta_offset1() (in module stistools.wavelen), 18
get_trace() (in module stistools.wx2d), 21
getHelpAsString() (in module stistools.basic2d), 5
getHelpAsString() (in module stistools.calstis), 8
getKWInfo() (in module stistools.mktrace), 14
getTable() (in module stistools.gettable), 33
getTraceInd() (stistools.mktrace.Trace method), 13
gFitTrace() (stistools.mktrace.Trace method), 13

I
interp() (in module stistools.mktrace), 14
interpolate() (in module stistools.r_util), 31
interpolate_trace() (in module stistools.wx2d), 22
inv_avg_interp() (in module stistools.wx2d), 22
inv_haar() (in module stistools.wx2d), 22
iterable() (in module stistools.mktrace), 14

K
kd_apply_trace() (in module stistools.wx2d), 22
kd_resampling() (in module stistools.wx2d), 22

M
main() (in module stistools.basic2d), 5
main() (in module stistools.calstis), 8
medianfilter() (in module stistools.stisnoise), 11
mktrace() (in module stistools.mktrace), 14

N
newton() (in module stistools.evaldisp), 15

O
openTraceFile() (stistools.mktrace.Trace method), 14

P
polynomial() (in module stistools.wx2d), 23
precess() (in module stistools.radialvel), 30
prtOptions() (in module stistools.basic2d), 5
prtOptions() (in module stistools.calstis), 8

R
radialVel() (in module stistools.radialvel), 30
readTrace() (stistools.mktrace.Trace method), 14
rotateTrace() (in module stistools.gettable), 33
run() (in module stistools.basic2d), 5
run() (in module stistools.calstis), 8

S
shiftimage() (in module stistools.sshift), 9
sortrows() (in module stistools.gettable), 34
sshift() (in module stistools.sshift), 9
stis_psf() (in module stistools.wx2d), 23
stisnoise() (in module stistools.stisnoise), 11
stistools.basic2d (module), 3
stistools.calstis (module), 7
stistools.evaldisp (module), 15
stistools.gettable (module), 33
stistools.mktrace (module), 15
stistools.r_util (module), 31
stistools Documentation, Release 1.0.6 (14-Aug-2012)

stistools.radialvel (module), 29
stistools.sshift (module), 9
stistools.stisnoise (module), 11
stistools.wavelen (module), 17
stistools.wx2d (module), 19

T
Trace (class in stistools.mktrace), 13
trace_interp() (in module stistools.mktrace), 14
trace_name() (in module stistools.wx2d), 24

W
wavelet_resampling() (in module stistools.wx2d), 24
windowfilter() (in module stistools.stisnoise), 12
wipenfilter() (in module stistools.stisnoise), 12
writeTrace() (stistools.mktrace.Trace method), 14
wx2d() (in module stistools.wx2d), 25
wx2d_imset() (in module stistools.wx2d), 26