stistools Documentation

Release 1.3.0 (03-March-2019)

Warren Hack, Nadia Dencheva, Chris Sontag, Megan Sosey, Micha

Apr 03, 2019

Contents

Stistools is a package that provides Python-based data processing tools for working with Space Telescope Imaging Spectrograph (STIS) data. It contains the full STIS calibration pipeline as well as its individual components should the user wish to do their calibrations manually. Additionally, stistools features a selection of analysis tools independent from the pipeline. These analysis tools are in active development, so there is more to come.

Note: The information here reflect the *latest* software info and might not be in-sync with the STIS Data Handbook.

CHAPTER 1

Relationship to the IRAF/PyRAF stis Package

The intent of this package is to provide the user community with many of the useful tools contained within the original IRAF/PyRAF stis package in Python. This is consistent with a larger effort to transition the user community from IRAF/PyRAF to Python, which can be read about here. At this time, there remains several tools from the original IRAF/PyRAF version of the package that are still in development in the Python package. In this transitional period, it's recommended that STIS users familiarize themselves with this package. Understanding that many of the users of this package will be those who may be newer to Python as a result of this transition, this documentation seeks to provide a step-by-step guide for getting started.

CHAPTER 2

Getting Started

2.1 Installation and Setup

2.1.1 Installing stistools (Astroconda)

The simplest installation method for installing stistools is to install the "Standard Software Stack" of AstroConda. AstroConda is an STScI-maintained software distribution channel for Conda, a package and environment management system. The standard software stack of AstroConda contains all of STScI's publicly distributed software, as well as all of the dependencies to run them. Effectively, it takes care of everything for you.

The first step is to download Conda. There are a few different flavors of Conda, but for most cases we'd recommend installing the Python 3 version of miniconda. For a step-by-step guide on installing Conda, consult the AstroConda documentation.

With Conda installed, let's now create an AstroConda environment. For this step, you can continue to follow the AstroConda documentation, which contains information on the different stack choices, or you can just read on and execute the commands we'd recommend for stistools.

First, we need to configure Conda to grant it access to the AstroConda channel, which can be done by running the following command in a BASH shell.

```
$ conda config --add channels http://ssb.stsci.edu/astroconda
# Writes changes to ~/.condarc
```

Now we're ready to install the STScI Software Stack, we'll accomplish this by setting up a fresh Conda environment. Run the following command, you can change "stisenv" to be whatever name for the environment you wish.

```
$ conda create -n stisenv stsci
```

Once the installation is complete, you can access your new environment by activating it:

```
$ source activate stisenv
```

Once activated, you now have access to all of the STScI software, including stistools! If you want to deactivate an environment, you can do so like this:

```
$ source deactivate stisenv
```

Keep in mind that whenever you open a new terminal, by default your environment will not be activated (this can be changed). So be sure to activate it before attempting to use stistools. When in your environment, you can now interact with stistools like any other Python package.

```
$ python
>>> import stistools
The following tasks in the stistools package can be run with TEAL:
   basic2d      calstis     ocrreject     wavecal        x1d          x2d
```

2.1.2 Getting the Latest Version of stistools

Sometimes, it may be the case that new additions to stistools have not yet been packaged into a proper release through Astroconda. In these instances, the installation of stistools through astroconda will not contain the most recent additions to the package. The following instructions outline how to grab and install the latest version of stistools, if you require something that has been released very recently.

To start, we'll assume that you've gone through the process above, installing stistools through astroconda. Even though astroconda does not contain our most up-to-date version of stistools in this case, it does still provide us with all of the necessary dependencies needed to run stistools.

First, let's clone the github stistools repository down to our local machines. This essentially downloads the latest stable version of the package to your computer. We can clone stistools by running the following command:

```
$ git clone https://github.com/spacetelescope/stistools.git
```

Note that this will create a "stistools" folder in your local directory. Navigate into this directory once the clone finishes executing. We want to install this on top of our astroconda environment, so activate your desired environment like so:

```
$ source activate stisenv
```

Because developer versions of stistools share the same version numbers as the last release, we'll need to remove the version of stistools that came with our astroconda environment, we can do this through conda:

```
$ conda uninstall --force stistools
```

The *–force* flag is necessary for instructing conda not to uninstall packages that depend on stistools. We can now install the latest version of stistools. In the stistools directory, run:

```
$ python setup.py install
```

This builds the stistools package up based on the source code we cloned to our local machines. Note that this overwrites the existing version of stistools that was installed through astroconda. With this, you should now have the latest version of stistools installed in your "stisenv" environment.

2.1.3 Setting up CRDS (Recommended)

Some calibration tasks in stistools require additional reference files to successfully run. In the past, users were expected to download these reference files manually by using MAST. While this approach is still valid, it can be inconvenient. The HST Calibration Reference Data System (CRDS) has a python package that can easily download and cache the relevant reference files for your data for you. And in fact, the crds package is a part of the astroconda stack and therefore is already installed if you've installed stistools through AstroConda. To get this setup, all we need to do is run a few commands:

```
$ export CRDS_PATH="$HOME/crds_cache"
$ export CRDS_SERVER_URL="https://hst-crds.stsci.edu"
$ export oref="${CRDS_PATH}/references/hst/oref/"
```

The above syntax define where your personal copies of CRDS reference files will be stored and the CRDS server that is used. Then the following command may be used to assign and obtain the best references files:

```
$ crds bestrefs --update-bestrefs --sync-references=1 --files *.fits
```

Note that in this example bestrefs will run on files currently in your working directly. You can modify where it looks by updating the final input.

CHAPTER 3

Pipeline Software

3.1 calstis

Calibrate STIS data.

The input raw files should be in the default directory. This is not always necessary, but it will always work. For spectroscopic data, if a path is specified for the input file, the wavecal file may not be found unless the wavecal file name (including path) was explicitly specified.

3.1.1 Examples

In Python without TEAL:

```
>>> import stistools
>>> stistools.calstis.calstis("o66p01020_raw.fits", verbose=True,
...                           trailer="o66p01020.trl")
```

In Python with TEAL:

```
>>> from stistools import calstis
>>> from stsci.tools import teal
>>> teal.teal("calstis")
```

From command line:

```
% ./calstis.py -v -s o66p01020_raw.fits out/
% ./calstis.py -r
```

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.

3.2 ocrreject

Add STIS exposures, rejecting cosmic rays.

3.2.1 Examples

In Python without TEAL:

```
>>> import stistools
>>> stistools.ocrreject.ocrreject("o3tt02020_flt.fits",
...     "o3tt02020_crj.fits", verbose=True, trailer="o3tt02020.trl")
```

In Python with TEAL:

```
>>> from stistools import ocrreject
>>> from stsci.tools import teal
>>> teal.teal("ocrreject")
```

From command line:

```
% ./ocrreject.py -v -s o3tt02020_flt.fits o3tt02020_crj.fits
% ./ocrreject.py -r
```

stistools.ocrreject.**getHelpAsString**(*fulldoc=True*)
Return documentation on the ocrreject function.

stistools.ocrreject.**main**(*args*)

stistools.ocrreject.**ocrreject**(*input*, *output*, *all=True*, *crrejtab=''*, *scalense=''*, *initgues=''*, *skysub=''*, *crsigmas=''*, *crradius=None*, *crthresh=None*, *badinpdq=None*, *crmask=''*, *verbose=False*, *timestamps=False*, *trailer=''*, *print_version=False*, *print_revision=False*)
Find and reject cosmic rays in STIS data.

Parameters

input: str Name of the input file or files.

output: str Name of the output file. See all for further information.

all: bool If True (the default), combine all input files into one output file. In this case, output should just be one file name. If False, the number of input and output file names must be the same.

crrejtab: str This argument may be used to override the CRREJTAB value in the primary headers of the input files.

scalense: str If specified, this overrides SCALENSE in the CRREJTAB.

initgues: str If specified, this overrides INITGUES in the CRREJTAB. The allowed values are "min" and "med" and "".

skysub: str If specified, this overrides SKYSUB in the CRREJTAB. The allowed values are "none", "mode" and "".

crsigmas: str If specified, this overrides CRSIGMAS in the CRREJTAB. The value should be a comma-separated string of one or more integer or float values. For each such value, calstis will perform one cosmic-ray-rejection cycle, with the sigma taken from the numerical value that was specified.

crradius: float or None If not None, this overrides CRRADIUS in the CRREJTAB. This is the rejection propagation radius in pixels (e.g. 1.5). After finding an outlier (a cosmic ray hit), adjacent pixels can also be flagged and excluded. Neighboring pixels will be rejected if their values are discrepant by more than crthresh * sigmas * noise, where noise is based on the noise model (i.e. Poisson noise and readout noise).

crthresh: float or None If not None, this overrides CRTHRESH in the CRREJTAB. This is the rejection propagation threshold (e.g. 0.8). If crthresh = 0 then all adjacent pixels (see crradius) will be rejected.

badinpdq: int or None If specified, this overrides BADINPDQ in the CRREJTAB. This is a data quality flag (or bitwise OR of flags) to allow rejection of pixels in the input images when forming the "guess" image (the image with which to compare the input images when looking for outliers).

crmask: str If specified, this overrides CRMASK in the CRREJTAB. crmask = "yes" means that the cosmic rays that are detected should be flagged in the DQ (data quality) extensions of the input 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 cs2.e (the calstis host executable) returned a non-zero status. If verbose is True, the value returned by cs2.e will be printed.

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

stistools.ocrreject.**run**(*configobj=None*)
TEAL interface for the ocrreject function.

3.3 basic2d

Perform basic 2-D calibration of STIS data.

3.3.1 Examples

In Python without TEAL:

```
>>> import stistools
>>> stistools.basic2d.basic2d("o66p01020_raw.fits", verbose=True,
...                           trailer="o66p01020.trl")
```

In Python with TEAL:

```
>>> from stistools import basic2d
>>> from stsci.tools import teal
>>> teal.teal("basic2d")
```

From command line:

```
% ./basic2d.py -v -s o66p01020_raw.fits o66p01020_flt.fits
% ./basic2d.py -r
```

stistools.basic2d.**basic2d**(*input*, *output=''*, *outblev=''*, *dqicorr='perform'*, *atodcorr='omit'*, *blevcorr='perform'*, *doppcorr='perform'*, *lorscorr='perform'*, *glincorr='perform'*, *lflgcorr='perform'*, *biascorr='perform'*, *darkcorr='perform'*, *flatcorr='perform'*, *shadcorr='omit'*, *photcorr='perform'*, *statflag=True*, *darkscale=''*, *verbose=False*, *timestamps=False*, *trailer=''*, *print_version=False*, *print_revision=False*)
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.)

doppcorr: 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.)

lorscorr: str If "perform", bin high-res data to lo-res. (MAMA only.)

glincorr: str If "perform", correct for global non-linearity. (MAMA only.)

lflgcorr: 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 PHOTFLAM, 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.

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.

3.4 x1d

Extract 1-D spectrum.

3.4.1 Examples

In Python without TEAL:

```
>>> import stistools
>>> stistools.x1d.x1d("o66p01020_flt.fits", output="test_x1d.fits",
...                   verbose=True, trailer="o66p01020.trl")
```

In Python with TEAL:

```
>>> from stistools import x1d
>>> from stsci.tools import teal
>>> teal.teal("x1d")
```

From command line:

```
% ./x1d.py -v o66p01020_flt.fits o66p01020_x1d.fits
% ./x1d.py -r
```

stistools.x1d.**getHelpAsString**(*fulldoc=True*)
Return documentation on the x1d function.

stistools.x1d.**main**(*args*)

stistools.x1d.**prtOptions**()
Print a list of command-line options and arguments.

stistools.x1d.**run**(*configobj=None*)
TEAL interface for the x1d function.

stistools.x1d.**x1d**(*input*, *output=''*, *backcorr='perform'*, *ctecorr='perform'*, *dispcorr='perform'*, *helcorr='perform'*, *fluxcorr='perform'*, *sporder=None*, *a2center=None*, *maxsrch=None*, *globalx=False*, *extrsize=None*, *bk1size=None*, *bk2size=None*, *bk1offst=None*, *bk2offst=None*, *bktilt=None*, *backord=None*, *bksmode='median'*, *bksorder=3*, *blazeshift=None*, *algorithm='unweighted'*, *xoffset=None*, *verbose=False*, *timestamps=False*, *trailer=''*, *print_version=False*, *print_revision=False*)
Extract a 1-D spectrum from an flt or crj file.

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.

backcorr: str If "perform", subtract the background.

ctecorr: str If "perform", apply CTE correction (CCD only).

dispcorr: str If "perform", compute wavelengths from the dispersion relation.

helcorr: str If "perform", correct for heliocentric Doppler shift.

fluxcorr: str If "perform", convert to absolute flux.

sporder: int or None The number of the spectral order to extract.

a2center: float or None

maxsrch: float or None

globalx: bool If True, use the global cross correlation offset (i.e. average for all orders) for all spectral orders.

extrsize: float or None Size of extraction box. None means extrsize is not specified.

bk1size: float or None Size of first background region. None means bk1size is not specified.

bk2size: float or None Size of second background region. None means bk2size is not specified.

bk1offst: float or None

Offset of first background region. None means bk1offst is not specified.

bk2offst: float or None Offset of first background region. None means bk2offst is not specified.

bktilt: float or None Background tilt. None means bktilt is not specified.

backord: int or None Background order (0 or 1). None means backord is not specified.

bksmode: str Background smoothing mode ("off", "median" (the default), or "average").

bksorder: int Background smoothing polynomial order (default is 3).

blazeshift: float or None Blaze shift (in pixels). None means blazeshift is not specified.

algorithm: str Extraction algorithm ("unweighted" (the default) or "sc2d")

xoffset: float Offset in X for slitless extraction.

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 cs6.e (the calstis host executable) returned a non-zero status. If verbose is True, the value returned by cs6.e will be printed. 2 is returned if the specified

input file or files were not found, or the numbers of input and output files (if the latter was specified) are not the same.

3.5 x2d

Rectify 2-D STIS spectral data.

3.5.1 Examples

In Python without TEAL:

```
>>> import stistools
>>> stistools.x2d.x2d("o66p01020_flt.fits", output="test_x2d.fits",
...                   verbose=True, trailer="o66p01020.trl")
```

In Python with TEAL:

```
>>> from stistools import x2d
>>> from stsci.tools import teal
>>> teal.teal("x2d")
```

From command line:

```
% ./x2d.py -v o66p01020_flt.fits o66p01020_x2d.fits
% ./x2d.py -r
```

stistools.x2d.**getHelpAsString**(*fulldoc=True*)
Return documentation on the x2d function.

stistools.x2d.**main**(*args*)

stistools.x2d.**prtOptions**()
Print a list of command-line options and arguments.

stistools.x2d.**run**(*configobj=None*)
TEAL interface for the x2d function.

stistools.x2d.**x2d**(*input*, *output=''*, *helcorr='perform'*, *fluxcorr='perform'*, *statflag=True*, *center=False*, *blazeshift=None*, *err_alg='wgt_var'*, *verbose=False*, *timestamps=False*, *trailer=''*, *print_version=False*, *print_revision=False*)
Rectify 2-D STIS spectral data.

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.

helcorr: str If "perform", correct for heliocentric Doppler shift.

fluxcorr: str If "perform", convert to absolute flux.

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

center: bool If True, center the target spectrum in the cross-dispersion direction. For G140L and G140M spectra, the target has at different times been offset to a location either above or below the middle of the detector, to avoid the repeller wire. This argument allows more convenient comparison of data taken at widely different times.

blazeshift: float or None Blaze shift (in pixels). None means blazeshift is not specified.

err_alg: str Algorithm for computing error estimates. The default is "wgt_var", which means that the weight (for bilinear interpolation) is applied to the variances of the input pixels. The alternative is "wgt_err", to specify that the weight should be applied to the errors of the input pixels.

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 cs7.e (the calstis host executable) returned a non-zero status. If verbose is True, the value returned by cs7.e will be printed. 2 is returned if the specified input file or files were not found, or the numbers of input and output files (if the latter was specified) are not the same.

3.6 wavecal

stistools.wavecal.**getHelpAsString**(*fulldoc=True*)
Return documentation on the wavecal function.

stistools.wavecal.**main**(*args*)

stistools.wavecal.**mkRandomNameW**(*prefix='wavecal_'*, *suffix='_tmp.fits'*, *n=100000000*)

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

stistools.wavecal.**run**(*configobj=None*)
TEAL interface for the wavecal function.

stistools.wavecal.**runBasic2d**(*wavecal*, *tempfnames*, *verbose*, *timestamps*, *fd_trailer*)

stistools.wavecal.**runCs11**(*fwv_file*, *infile*, *tempfnames*, *verbose*, *timestamps*, *fd_trailer*)
Subtract a fraction of the science image from the wavecal image.

stistools.wavecal.**runCs12**(*w2d_file*, *infile*, *option*, *verbose*, *timestamps*, *fd_trailer*)

stistools.wavecal.**runWavecal**(*w2d_file*, *dbg*, *angle*, *verbose*, *timestamps*, *fd_trailer*)

stistools.wavecal.**runX2d**(*cwv_file*, *angle*, *tempfnames*, *verbose*, *timestamps*, *fd_trailer*)

stistools.wavecal.**wavecal**(*input*, *wavecal*, *debugfile=''*, *savetmp=False*, *option='linear'*, *angle=None*, *verbose=False*, *timestamps=False*, *trailer=''*, *print_version=False*, *print_revision=False*)
Perform wavecal processing for STIS data.

Parameters

input: str Names of the flt or crj file or files for the science exposure. The SHIFTA1 and SHIFTA2 keywords will be updated in these files, based on the results of processing the wavecal file(s).

wavecal: str Names of the associated wavecal file or files (either raw or calibrated). If this is a raw file, it will first be calibrated using cs1.e (basic2d), then with cs7.e (x2d) for first-order grating data. These calibrated files are regarded as temporary, and (unless savetmp) they will be deleted when processing has been completed.

debugfile: str If specified, debugging information will be written to a file with this name. For echelle data this will be a FITS file, but for first-order data it will be a text file (and possibly a FITS file as well).

savetmp: bool If wavecal is a raw wavecal file, some calibration will be performed, depending on mode. If savetmp is False (the default), the calibrated wavecal files will be deleted after wavecal processing is complete.

option: str If the wavecal file contains more than one image set, the shifts will be interpolated between wavecal exposures that bracket the science exposure. This argument gives the interpolation option, either "linear" (the default) or "nearest". If the science exposure was before the first or after the last exposure in the wavecal file, the shifts will be copied from the first or last exposure respectively.

angle: float or None This argument is only relevant for echelle data for which the wavecal was taken with a long slit (e.g. 6X0.2). The angles have not been measured accurately; they vary from one grating to another, and they even vary depending on location on the detector. This argument specifies the slit angle, in degrees measured clockwise from the Y axis. Here are some approximate values:

E230M 0.9 to 1.2 E230H 4.9 to 6.9 E140H -3.8 to -5.8

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, without checking any other argument.

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 cs4.e (the calstis host executable) returned a non-zero status. If verbose is True, the value returned by cs4.e will be printed. 2 is returned if the specified input file or files were not found, of if the numbers of input and wavecal files (or of debugfiles) are not the same.

CHAPTER 4

Analysis Tools

4.1 sshift

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.

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.

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 replace 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)

4.2 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*)

4.3 mktrace

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 updates in the header of the input file.

4.3.1 Examples

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

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

Authors

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

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= tr.getTraceInd(a2center)

tr_ind is the index of the row in the trace file which brackets from below a2center as computed fro the generated trace

tr.readTrace(tr_ind)

a2center = tr.generateTrace(. . .)

Methods

gFitTrace(specimage, y1, y2)	Fit a gaussian to each column of an image.
generateTrace(data, kwinfo[, tracecen, wind])	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, ...)	The 'writeTrace' method performs the following steps:

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

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

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

4.4 evaldisp

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 = coeff[0] +
    coeff[1] * m * wl +
    coeff[2] * m**2 * wl**2 +
    coeff[3] * m +
    coeff[4] * wl +
    coeff[5] * m**2 * wl +
    coeff[6] * m * wl**2 +
    coeff[7] * m**3 * wl**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

4.5 wx2d

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 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 [fits Header object] primary header of input file

hdr [fits 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 trace 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*, *shifta2=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 shifta2 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

x [float] offset in pixels from the center of the profile

a [float] a measure of the width of the PSF

Returns

val [float] the PSF evaluated at x

stistools.wx2d.**trace_name**(*trace*, *phdr*)

Return the 1dt table name or array.

Parameters

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

phdr [fits Header object] primary header, used only if trace is None

Returns

tracefile [string or array] name of a trace file (with environment variable expanded), or an actual trace array

stistools.wx2d.**wavelet_resampling**(*hdu*, *img*, *errimg*, *original_nrows*, *nrows*, *ncols*, *rows*, *a2center*, *a2displ*, *offset*, *shifta2*, *imset*, *order*, *subdiv*, *psf_width*, *subsampled*, *convolved*)

Resample img and errimg using wavelets.

Parameters

hdu [fits header/data unit object] header/data unit for a SCI extension

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)

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)

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] Fits 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

4.6 radialvel

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:

- Vx = radius_dot,
- Vy = radius * elong_dot,
- Vz = 0

where:

- radius is the radial distance from Earth to Sun
- elong is the ecliptic longitude of the Sun
- eps is the obliquity of the ecliptic
- _dot means the time derivative

Rotate in the XY-plane by elong to get the velocity in ecliptic coordinates:

```
radius_dot * cos (elong) - radius * elong_dot * sin (elong)
radius_dot * sin (elong) + radius * elong_dot * cos (elong)
0
```

Rotate in the YZ-plane by eps to get the velocity in equatorial coordinates:

```
radius_dot * cos (elong) - radius * elong_dot * sin (elong)
(radius_dot * sin (elong) + radius * elong_dot * cos (elong)) * cos (eps)
(radius_dot * sin (elong) + radius * elong_dot * cos (elong)) * sin (eps)
```

stistools.radialvel.**precess**(*mjd*, *target*)

Precess target coordinates from J2000 to the date mjd.

Parameters

mjd [float] time, Modified Julian Date

target [array_like object] unit vector pointing toward the target, J2000 coordinates

Returns

vector [ndarray] 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)

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):

```
target = [[x0, x1, x2, x3, x4],
          [y0, y1, y2, y3, y4],
          [z0, z1, z2, z3, z4]]
```

The algorithm used in this function was based on [1] and [2].

References

[1], [2]

stistools.radialvel.**radialVel**(*ra_targ*, *dec_targ*, *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.

Parameters

ra_targ [float] right ascension of the target (degrees)

dec_targ [float] declination of the target (degrees)

mjd [float] Modified Julian Date at the time of observation

Returns

radial_vel [float] the radial velocity in km/s

4.7 r_util

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 [str] A file name, possibly including an environment variable.

Returns

fullname [str] The file name with environment variable expanded.

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

Interpolate.

Linear interpolation is used. If the specified indpendent 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.

4.8 gettable

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

trace_info [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

expstart [float] exposure start time (MJD)

stistools.gettable.**sortrows**(*rowdata*, *sortcol*, *ascend=True*)
Return a copy of rowdata, sorted on sortcol.

4.9 inttag

The task *inttag* converts an events table of TIME-TAG mode STIS data into a raw, time-integrated ACCUM image. By default, *inttag* only integrates over the good time intervals (GTI), though the user can choose to integrate over the entire exposure time by setting allevents=True. The output image can be calibrated as any other raw image.

The input file for *inttag* is an event stream table of TIME-TAG mode produced by generic conversion. The data will be Doppler corrected (as required for medium and high resolution spectroscopic modes). This file will consist of a primary header with no data, and two binary table extensions. The primary header is identical in structure to the primary header of an ACCUM mode image. The first binary table (EXTNAME=EVENTS) contains a list of the events themselves (i.e. science data as an event stream), and the second binary table (EXTNAME=GTI) contains a list of good time intervals for the TIMETAG exposure. Columns "TIME", "AXIS1", and "AXIS2" in the EVENTS table are read. Columns "START" and "STOP" in the GTI table are read.

The output image is a time integrated (ACCUM mode) image with the same structure as any other STIS MAMA raw image (i.e. primary header followed by a single or series of triplet extensions: SCI, ERR, DQ). The number of triplets is determined by the value of rcount. The time interval in the Nth triplet covers from (starttime + (N-1)*increment) to (starttime + N*increment). The exposure time in each interval need not be identical, because events are included in the image only if they occur during "good time intervals" (as determined by the GTI extension table). The keyword OBSMODE in the primary header of the output image will still be set to "TIME-TAG".

The output science image is ready to be calibrated (see `calstis`, `crreject`, `basic2d`, `x2d`, `x1d`).

4.9.1 Examples

inttag with default values:

```
>>> import stistools
>>> stistools.inttag.inttag("oddv01050_tag.fits", "oddv01050_raw.fits")
```

inttag with highres output:

```
>>> import stistools
>>> stistools.inttag.inttag("oddv01050_tag.fits", "oddv01050_raw.fits", highres=True)
```

inttag with multiple output imsets (5 count regions of 200s each):

```
>>> import stistools
>>> stistools.inttag.inttag("oddv01050_tag.fits", "oddv01050_raw.fits", rcount = 5,␣
↪increment = 200)
```

inttag.**events_to_accum**(*events_data*, *size_x*, *size_y*, *highres*)
Map timetag events to a 2d accum image array.

Parameters

events_data: record array Record array of timetag events.

size_x: int Number of pixels on axis 1 of the detector.

size_y: int Number of pixels on axis 2 of the detector.

highres: bool Boolean value indicating whether the output accum image is in high or low resolution.

Returns

accum: array 2d image of all events in the imset on the detector.

inttag.**exp_range**(*starttime*, *stoptime*, *events_data*, *gti_data*, *tzero_mjd*)
Calculate exposure time, expstart, and expstop and mask imset

Parameters

starttime: float Start time of the imset in seconds

stoptime: float Stop time of the imset in seconds

events_data: record array Record array of timetag events.

gti_data: record array Record array of good time intervals (GTIs).

tzero_mjd: bool Modified Julian Date (MJD) corresponding to the beginning of the exposure

Returns

exp_time: float Total exposure time in seconds for the given imset. This number accounts for any exposure time lost to non-GTI time (if the user is not using allevents).

expstart: float Start time of the imset exposure

expstop: float Stop time of the imset exposure

good_events: float The events list within the imset exposure time and within the GTIs.

inttag.**inttag**(*tagfile*, *output*, *starttime=None*, *increment=None*, *rcount=1*, *highres=False*, *allevents=False*, *verbose=True*)
Convert an events table of TIMETAG into an integrated ACCUM image.

Parameters

tagfile: str input file that contains TIMETAG event stream. This is ordinarily a FITS file containing two tables. The TIMETAG data are in the table with EXTNAME = "EVENTS", and the "good time intervals" are in the table with EXTNAME = "GTI". If the GTI table is missing or empty, all times will be considered "good".

output: str Name of the output FITS file.

starttime: float Start time for integrating events, in units of seconds since the beginning of the exposure. The default value of None means that the start time will be set to the first START time in the GTI table.

increment: float Time interval in seconds. The default value of None means integrate to the last STOP time in the GTI table, divided by rcount.

rcount: int Repeat count, the number of output image sets to create. If rcount is greater than 1 and increment is not specified, will subdivide the total exposure time by rcount.

highres: bool Create a high resolution output image? Default is False.

allevents: bool If allevents is set to True, all events in the input EVENTS table will be accumulated into the output image. The TIME column in the EVENTS table will only be used to determine the exposure time, and the GTI table will be ignored.

verbose: bool Print additional info?

4.10 wavelen

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 [fits Header object] primary header

hdr [fits 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

4.11 doppinfo

This class computes Doppler shift information for each imset of a dataset. Keywords will be read from the science file and from the support file. The Doppler shift information is printed to the standard output by default, but this can be turned off by setting quiet to True. Three task parameters will be updated by doppinfo; these allow the user to compute the Doppler shift in high-res pixels or km/s at any time close to the time of the exposure.

The printed information will be in one of two formats, depending on the value of increment. If increment is zero, the average and extreme values of the Doppler shift during the exposure will be printed. If increment is greater than zero, the actual Doppler shift and radial velocity will be printed at the beginning of the exposure and at every increment seconds thereafter until the end of the exposure. Both the printed value of the Doppler shift and the doppmag parameter will be in high-res pixels.

Most of the Doppler shift information is computed directly from the orbital elements of HST, as given in the support file primary header. Some information, however, is computed based on the approximation of a circular orbit. This approximation is used for the average Doppler shift during the exposure (printed if increment is zero) and for the task parameters doppzero, doppmag and radvel that are updated by doppinfo. These parameters are applied as terms in a sine function, which inherently involves a circular-orbit approximation.

The parameters for the circular orbit are determined as follows. The HST orbital elements are gotten from the primary header of the support file. The target position is taken from the keywords RA_TARG and DEC_TARG in the science file primary header. The velocity of HST is computed from the orbital elements at 64 equally spaced times throughout an orbit, centered on the midpoint of the exposure, (EXPSTART + EXPEND) / 2, and the component of this velocity in the direction away from the target (i.e. the radial velocity) is taken. A sine function is fit to these radial velocities; the amplitude is radvel, and the amplitude and phase are used to compute doppmag and doppzero.

4.11.1 Examples

Doppinfo with dt of 100:

```
>>> import stistools
>>> stistools.doppinfo.Doppinfo("ocb6o2020_raw.fits", dt=100, spt="ocb6o2020_spt.fits
↪")
# orbitper  doppzero      doppmag      doppmag_v    file
  5728.67  56752.114170  11.68643135   7.40391177   ocb6o2020_raw.fits[sci,1]
# time (MJD)    shift   radvel
56752.165175  -11.59   -7.345
56752.166333  -11.37   -7.203
56752.167490  -11.01   -6.975
56752.168647  -10.52   -6.663
56752.169805   -9.90   -6.272
56752.170962   -9.16   -5.805
56752.172120   -8.32   -5.269
56752.173277   -7.37   -4.669
56752.174434   -6.34   -4.014
56752.175592   -5.23   -3.311
56752.176749   -4.05   -2.568
56752.177907   -2.83   -1.794
56752.179064   -1.58   -0.998
56752.180222   -0.30   -0.190
# orbitper  doppzero      doppmag      doppmag_v    file
  5728.67  56752.180505  11.68734454   7.40449032   ocb6o2020_raw.fits[sci,2]
```

(continues on next page)

(continued from previous page)

```
# time (MJD)    shift    radvel
56752.181784     1.42     0.902
56752.182941     2.68     1.700
56752.184099     3.91     2.477
56752.185256     5.09     3.225
56752.186413     6.21     3.935
56752.187571     7.26     4.598
56752.188728     8.22     5.205
56752.189886     9.08     5.750
56752.191043     9.83     6.227
56752.192200    10.46     6.628
56752.193358    10.97     6.950
56752.194515    11.35     7.189
56752.195673    11.59     7.342
56752.196830    11.69     7.406
```

class stistools.doppinfo.**Doppinfo**(*input*, *spt=None*, *dt=0.0*, *update=False*, *quiet=False*)

Compute Doppler parameters and information from HST orbital elements. This class previously supported both COS and STIS data, but now only supports STIS data. The class will print doppler shift information for all imsets contained in the input image.

Results will be printed to standard out. To have the DOPPZERO, DOPPMAG, and DOPPPMAGV keywords inserted/updated in the header you can set update to True.

Methods

printDopplerShift(dt)	Compute and print the Doppler shift at intervals of dt.

printDopplerShift(*dt*)

Compute and print the Doppler shift at intervals of dt.

Parameters

dt: float Time interval (seconds) for printing Doppler shift throughout the orbit, or if dt is zero print the min and max Doppler shift during the orbit.

4.12 ctestis

The purpose of this ctestis task is to correct signal levels of point-like source in photometry tables measured from STIS CCD images for charge loss due to imperfect Charge Transfer Efficiency (CTE). The algorithm used to correct for CTE-induced signal loss is the one published in Goudfrooij, Bohlin, Maiz-Apellaniz, & Kimble, 2006, PASP, October 2006 edition (astro-ph/0608349). The values of CTE loss derived using this algorithm should be accurate to about 3% RMS (tested for data taken between March 1997 and August 2004). No significant differences in CTE loss were found for different aperture sizes, although this has been verified only for a limited range of aperture sizes (2, 3, and 5 pixel radii). The algorithm was derived from measurements of point sources in a relatively sparse field (the outskirts of a Galactic globular cluster), as detailed in the PASP paper mentioned above.

The function also computes the shift in the Y centroid of point sources due to distortions in the steller PDF cause by CTE trails. the algorithm is taken from the Equation 9 of Goodfrooij et al. (2006). Note, however, that the equation has been multiplied by -1, so that the resulting correction may be ADDED to measured Y centriod of each star.

4.12.1 Examples

ctestis with ycol set to 182, net set to 5,000 and sky set to 150.

```
>>> from stistools.ctestis import ctestis
>>> fluxc, dmagc, dyc = ctestis(182., 5000., 150., stisimage='o4qp9g010_crj.fits')
mjd: 50893.30
nread: 2
ybin: 1
gain:  1.0
amp: D
tt0: -2.3865942
lcts: -0.67595399
bck: 75.0
lbck: 2.317577
cti: 1.7314006e-05
fluxc: 2536.7133
dmagc: -0.015828427
cti10000: 0.17314006
dy512: 0.0043051192
dyc: 0.007079903
net: 5000.0
sky: 150.0
ycol: 182.0
fluxc: 2536.7133
dmagc: -0.015828427
dyc: 0.007079903
```

stistools.ctestis.**ctestis**(*ycol*, *net*, *sky*, *stisimage=None*, *mjd=None*, *nread=None*, *ybin=None*, *gain=None*, *amp='D'*, *sx2=False*)

Calculate the STIS empirical correction to magnitude and astrometric shift, given photometry results.

Parameters

ycol [arr] Y-column # of object

net [arr] Net photometric counts

sky [arr] Counts in sky region, scaled to source area (?)

stisimage [str, optional] The name of the SX2 file from which to pull the header keywords.

mjd [float, optional] Modified julian date corresponding to the start time of the 1st exposure, corresponds to the TEXPSTRT keyword. If stisimage file is defined TEXPSTRT keyword will overwrite any provided mjd value.

nread [int, optional] Number of image sets combined during CR rejection, corresponds to the NCOMBINE keyword. If stisimage file is defined NCOMBINE keyword will overwrite any provided nread value.

ybin [int , optional] Axis2 data bin size in unbinned detector pixels, corresponds to the BINAXIS2 keyword. If stisimage file is defined BINAXIS2 keyword will overwrite any provided ybin value.

gain [float, optional] The image gain, corresponds to the CCDGAIN keyword. If stisimage file is defined the CCDGAIN keyword will overwrite any provided gain value. If the gain is 4.0, it will be upated to 4.08.

amp [str, optional] The amplifier used for the observation (default 'D'). Ignored if stisimage is provided.

sx2 [bool, optional] Force the procedure to remove the top/bottom 38 rows. This is automatically done if the file in stisname contains '_sx2'. Default values is False

Returns

fluxc [arr] The empirically-corrected flux (counts)

dmagc [arr] The empirical photometric correction (delta mag)

dyc [arr] The empirical astrometric correction (delta pixels)

4.13 tastis

Analyze STIS target acquisition images. *tastis* will print general information about each input target acquisition image, and will analyze both types of STIS target acquisitions: ACQs and ACQ/PEAKs

ACQ procedure is described in "STIS Onboard CCD Target Acquisitions" in the STIS Instrument Handbook. The ACQ/PEAK procedure is described in "Onboard Target-Acquisition Peakups (ACQ/PEAK)" also in the STIS Instrument Handbook.

Target positions in global and local (subarray) coordinates and the total flux of the target in the maximum checkbox during both acquistions phases (course and fine) are displayed.

If update=True, keywords are added to the header to make problems easier to locate in batch processing. Warnings are given if the spt file is not present when *tastis* is run.

4.13.1 Examples

tastis with the default of update=False:

```
>>> import stistools
>>> stistools.tastis.tastis("ocmv01w6q_raw.fits")
================================================================================
oc7w11viq       HST/STIS    G430L        0.3X0.05ND             ACQ/PEAK-UP
prop: 13465      visit: 11    line: 3   target: HD128621-2
obs date, time: 2014-07-24    22:05:06   exposure time:  0.10
dom GS/FGS: S7QX000330F1    sub-dom GS/FGS: S7QX000694F2
ACQ params:     bias sub: 1510                     method: RETURN-TO-BRIGHTEST
subarray (axis1,axis2):   size=(1022,32)          corner=(25,500)
--------------------------------------------------------------------------------
Scan type: LINEARAXIS2                  Step size (mas): 250
 [210 753   0]
                    axis1 axis2     axis1  axis2         V2   V3
                      (pixels)        (arcsec)           (arcsec)
Estimated slew:      0.0  -0.1     0.000 -0.005     -0.004  0.004
Flux in post-slew confirmation image (751752) - Pedestal (748587) = 3165 DN
--------------------------------------------------------------------------------
The flux in the confirmation image is 320% greater than the maximum flux
in the ACQ/PEAK scan.  An excess greater than 100% indicates
problems in the ACQ/PEAK.
The flux in the confirmation image is 16% of the recommended minimum
of 20000 DN for a dispersed-light ACQ/PEAK.  The signal-to-noise in
the AC
================================================================================
```

tastis with update=True:

```
>>> import stistools
>>> stistools.tastis.tastis("ocmv0lw6q_raw.fits", update=True)
================================================================================
ocmv0lw6q       HST/STIS    MIRVIS      F25ND3             ACQ/POINT
prop: 13760      visit: 0L    line: 1   target: CD-59D3300
obs date, time: 2016-09-29    23:43:50   exposure time:  1.10
dom GS/FGS: S4B0000993F2    sub-dom GS/FGS: S4B0000953F1
ACQ params:     bias sub: 1510   checkbox: 3      method: FLUX CENTROID
subarray (axis1,axis2):   size=(100,100)          corner=(487,466)
--------------------------------------------------------------------------------
Coarse locate phase:           Target flux in max checkbox (DN): 1560
                        global          local
                     axis1 axis2     axis1 axis2
Target location:     534.2  507.0    48.2  42.0
                     axis1 axis2     axis1  axis2         V2      V3
                       (pixels)        (arcsec)            (arcsec)
Estimated slew:      -1.5  -9.0    -0.079 -0.457       -0.379  0.268
--------------------------------------------------------------------------------
Fine locate phase:            Target flux in max checkbox (DN): 1559
                        global            local
                     axis1 axis2     axis1 axis2
Target location:     534.2  516.8    48.2  51.8
Ref ap location:     537.5  517.0    19.5  17.0
                     axis1 axis2     axis1  axis2         V2      V3
                       (pixels)        (arcsec)           (arcsec)
Estimated slew:      -2.1  -0.2     -0.104 -0.010      -0.081 -0.067
--------------------------------------------------------------------------------
Total est. slew:     -3.6  -9.2    -0.183 -0.467        -0.460  0.201
--------------------------------------------------------------------------------
Your ACQ appears to have succeeded, as the fluxes in the coarse
and fine stages agree within 25% and the fine slews were less than
4 pixels as expected
================================================================================
```

stistools.tastis.**tastis**(*raw_filename*, *update=False*)

Analyze STIS target acquisition images.

Parameters

raw_filename: str Name of the input raw file. For some raw files you will need a copy of the spt file in the same directory.

update: bool If True, keywords associated with tastis checks will be updated. Default values is False.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Bibliography

[1] Lieske, et al. 1976, Astron & Astrophys vol 58, p 1.

[2] J.H. Lieske, 1979, Astron & Astrophys vol 73, 282-284.

Python Module Index

i

s

Index