refstis Documentation

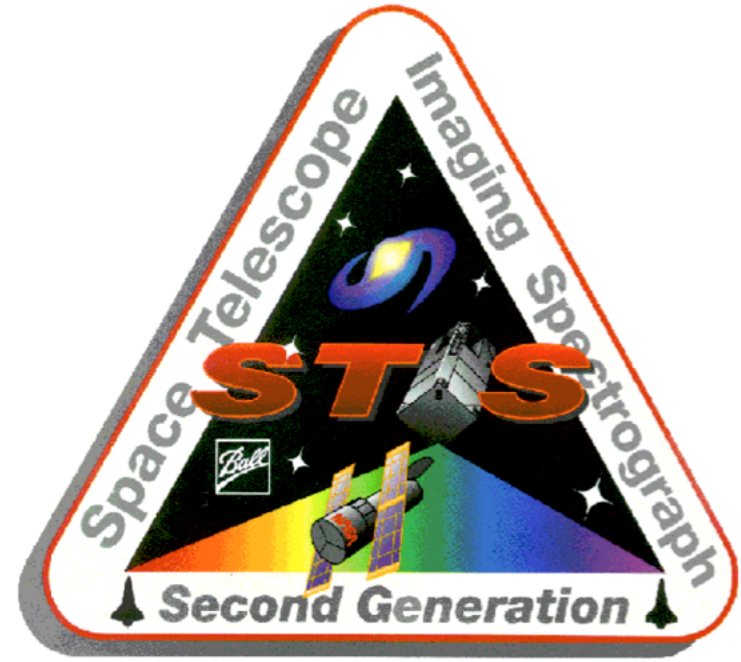
Release 0.8.0a1

Justin Ely

Apr 02, 2020

CONTENTS

Refstis is a collection of tasks and pipeline processes to create the superdark and superbias reference files for the Space Telescope Imaging Spectrograph (STIS) on board the Hubble Space Telescope (HST).



Note: These routines, and particularly the pipeline, are designed to regularly create and deliver the reference files that are applied in the MAST archive. As such, everything is geared particularly to the needs of the STIS instrument team at STScI.

CHAPTER

ONE

INSTALLATION

1.1 Installation Instructions

1.1.1 Install via Anaconda

TBD.

1.1.2 Install from source

`Refstis` can be installed manually using the source code:

```
$ git clone https://github.com/spacetelescope/refstis.git
$ cd refstis
$ python setup.py install
```

CHAPTER

TWO

USAGE

2.1 Usage

2.1.1 Command-line

Many individual tasks have been pulled out into individual modules which are copied into your path when the package install is run. This allows a particular procedure to be run on a collection of data without going through the entire pipeline, and without the limitations that it imposes (darks/biases from specific proposals, specifically split months/weeks, etc).

Basejoint

Basejoint is a confusingly named script that actually creates a baseline bias file for use in creating baseline dark files.

```
$ basejoint *.fits outname.fits
```

Refbias

Refbias is the preferred way to create the weekly bias file. In some cases, weekbias may be run instead.

```
$ refbias *.fits outname.fits
```

Weekbias

If the number of datasets is fewer than a set threshold the weekbias procedure is run instead of the refbias. A main difference between the two is that weekbias uses the baseline bias in some of the calculations.

```
$ weekbias *.fits outname.fits basebias.fits
```

Basedark

The basedark task is designed to create a monthly baseline dark. This is a combination of all darks for a month that will be used as input when creating a weekly dark by the weekdark task.

```
$ basedark *raw.fits outname.fits basebias.fits
```

Weekdark

The weekdark uses ~a week's worth of data, along with the monthly dark produced by basedark, to create a dark appriate to a particular week's worth of data.:

```
$ weekdark *raw.fits outname.fits basedark.fits basebias.fits
```

2.1.2 The Pipeline

Warning: This section is deprecated. A new set of scripts to automatically derive pipeline reference files is being developed and will be integrated here soon.

The pipeline can be run from any directory, with input/output directories determined from the config file outlined in the section below. Simply executing the pipeline command from the terminal will kick everything off:

```
$ refstis_pipeline
```

This will perform all the necessary pipeline steps to create the full suite of superdarks and superbiases, includng;

1. Checking for all the STIS anneal datasets and determining anneal months.
2. Retrieving new raw dark and bias observations for each month.
3. Reducing the raw datasets and combining into weekly and bi-weekly reference files.
4. Prepping the reference files for delivery into CRDS.
5. Running the data against a test-suite and verifing no errors are produced.

After all steps have been completed for a given anneal month, the pipeline will send you an email with the delivery form. This should be forwarded to redcat@stsci.edu once any final checks on the data are perfomed.

The configure file

The pipeline functionality of the refstis package needs to know some things about you and where to put stuff. This is accomplished by parsing a config file that is assumed to live at ~/refstis_config.yaml.

The necessary contents of the file are shown below, though the content is dummy and will need to be configured for you specifically.

```
#  Directories to read/write
products_directory : '/Users/myself/refstis/data/'
retrieve_directory : '/Users/myself/refstis/requested/'
delivery_directory : '/Users/myself/refstis/to_deliver/'
```

(continues on next page)

(continued from previous page)

```
# config for querying MAST for data
mast_server : 'server@name.stsci.edu'
mast_database : 'db_name'
mast_account : 'username'
mast_password : 'Pa$$w3rD'
dads_host : 'dads_host.stsci.edu'

# config for retrieving from MAST
archive : 'archive.stsci.edu'
archive_user : 'myself'
email : 'myself@stsci.edu'
ftp_user : 'myself'
host : 'host.domain.com'

# Proposals to use for darks/biases
dark_proposals:
 - 12000
 - 13001
 - 14243

bias_proposals:
 - 12001
 - 13005
 - 14244
```

CHAPTER

THREE

API

3.1 Api

3.1.1 Weekdark

Functions to create weekly superdarks for the STIS instrument.

refstis.weekdark.**create_superdark**(*crj_filename*, *basedark*)

Create a superdark from the crj and basedark

Create the science portion of the forthcoming reference dark by adding the 'only baseline dark current' image to the 'only hot pixels' image.

Note: input file will be updated in-place.

Parameters

- **crj_filename** (*str*) – filename of the cosmic-ray rejected file
- **basedark** (*str*) – basedark name

refstis.weekdark.**make_weekdark**(*input_list*, *refdark_name*, *thebasedark*, *thebiasfile=None*)

Create a weekly dark reference file

1. If not already done, run basic2d with blevcorr, biascorr, and dqicorr set to perform
2. Apply temperature correction to the data
3. split all raw images into their imsets
4. join imsets together into a single file
5. combine and cr-reject
6. normalize to e/s by dividing by (exptime/gain)
7. do hot pixel things

Note: Update ERR extension of new superdark by assigning the ERR values of the basedark except for the new hot pixels that are updated from the weekly superdark, for which the error extension of the weekly superdark is taken.

Parameters

- **input_list** (*list*) – list of input STIS dark files
- **refdark_name** (*str*) – output name of the reference dark file
- **thebasedark** (*str*) – Monthly basedark
- **thebiasfile** (*str, bool, optional*) – biasfile to use for calibration

3.1.2 Basedark

Functions to create a BaseDark for the STIS instrument

1. If not already done, perform bias subtraction
2. If after switch to side-2 electronics, perform temperature scaling
3. Join all imsets from input list into single file
4. combine and cr-reject
5. normalize to e/s by dividing by (exptime/gain)
6. update DQ array with hot pixel information

Note:

- Side 1 operations ended on May 16, 2001.
- Side 2 operations started on July 10, 2001.
- The dark correction will only be applied to datasets after July 1, 2001 (MJD 52091.0).

---

refstis.basedark.**find_hotpix**(*filename*)
Find hotpixels and update DQ array

Pixels hotter that median + 5*sigma will be updated to have a DQ value of 16.

Note: The input file will be updated in-place.

Parameters **filename** (*str*) – filename of the input biasfile

refstis.basedark.**make_basedark**(*input_list*, *refdark_name='basedark.fits'*, *bias_file=None*)
Make a monthly baseline dark from the input list.

Parameters

- **input_list** (*list*) – list of input dark files
- **refdark_name** (*str*) – name of the output reference file
- **bias_file** (*str or None*) – bias file to be used in calibration (optional)

refstis.basedark.**update_sci**(*filename*)
Create the science extension of the baseline dark

Note: The input file will be updated in-place.

Parameters **filename** (*str*) – name of the file to be updated

3.1.3 Weekbias

Functions to create a weekly bias for the STIS instrument.

refstis.weekbias.**make_weekbias**(*input_list*, *refbias_name*, *basebias*)

Make 'weekly' bias from list of input bias files

1. join imsets from each datset together into one large file
2. combine and cosmic ray screen joined imset
3. find hot colums
4. add hot colums in to basebias sci data as output science data
5. update error, dq, and headers

Note: For the SCI extensions, the baseline bias rate is taken from the aptly named basebias because anything that is "hot" is assumed to be potentially transient and is thus taken from the weekly biases.

Update ERR extension of new superbias by assigning the ERR values of the baseline superbias except for the new hot pixels that are updated from the weekly superbias, for which the error extension of the weekly superbias is taken. Put the result in temporary ERR image.

Parameters

- **input_list** (*list*) – list of STIS bias files
- **refbias_name** (*str*) – filename of the output reference file
- **basebias** (*str*) – filename of the monthly basebias

3.1.4 Refbias

Procedure to create a suberbias using the 'refbias' method.

Basic premise for making a refbias.

1. join imsets from each datset together into one large file
2. combine, cosmic ray screen, and normalize to number of imsets
3. set header keywords
4. clean up intermediate files

refstis.refbias.**flag_hot_pixels**(*refbias_name*)

Flag hotpixels in the DQ array

Pixels more than (mean + 3*sigma) away from a median smoothed image are flagged as DQ=16 in the DQ array.

Note: The input file is updated in-place.

Note: The IRAF version of this pipeline specified a 2x15 pixel median filter to calculate the smoothed imaged, but the IRAF task documentation says that even sizes are increased by 1 for the computation. A 3x5 pixel filter is used directly here.

Parameters **refbias_name** (*str*) – name of the reference file to flag

refstis.refbias.**make_refbias**(*input_list*, *refbias_name='refbias.fits'*)
Create a refbias FITS file

Parameters

- **input_list** (*list*) – list of input bias files
- **refbias_name** (*str*) – name of the output bias reference file

3.1.5 Basejoint

Procedure to make a monthyl bias for the STIS CCD.

Python translation of a python translation of an original IRAF cl script to create superbias reference file from a list of bias frames.

The input image (with multiple extensions) is overscan-subtracted and cosmic-ray-rejected using the CALSTIS algorithms within STSDAS. The cosmic-ray-rejected image is divided by the number of imsets present in the input image (since ocrreject adds up the individual imsets). After that, the superbias is median filtered using a window of 15 x 1 pixels. The median-filtered is subtracted from the superbias to produce a "residual" image containing hot columns and such. This "residual" image is averaged along rows and replicated back to the original image size, so that hot columns clearly show up. After that, the image values in hot columns and pixels of the original superbias image (defined as those pixels having values greater than (mean + 5 sigma of Poisson noise) are replaced by those in the median-filtered bias image. Plots are made of the row- and column-averaged superbias, with plotting scales appropriate to the gain and binning settings of the superbias.

refstis.basejoint.**average_biases**(*bias_list*)
Create a weighted sum of the individual input files.

First make sure all individual input files have been ocrrejected.

Parameters **bias_list** (*list*) – list of the input biases

Returns

- **mean_file** (*str*) – name of the averaged filename
- **totalweight** (*float*) – sum of the NCOMBINE header keywords from the data

refstis.basejoint.**calibrate**(*input_file*)
calibrate input file

refstis.basejoint.**make_basebias**(*input_list*, *refbias_name='basebias.fits'*)
Make the basebias for an anneal month

1- Calbrate each bias in the list 2- Average together the biases 3- Replace pixels and colums with median values 4- Set header keywords

Parameters

- **input_list** (*list*) – list of input bias files.
- **refbias_name** (*str*) – name of the output reference file.

refstis.basejoint.**replace_hot_cols**(*mean_bias*, *median_image*, *residual_image*, *yfrac=1*)
Replace hot columns in the mean_bias as identified from the residual image with values from the bias_median

'hot' is 3* sigma

mean_bias will be updated in place

refstis.basejoint.**replace_hot_pix**(*mean_bias*, *median_image*)
Replace image values in residual single hot pixels

defined as those having values greater than (mean + 5 sigma of Poisson noise) by those in median-filtered bias image. This represents the science extension of the final output reference superbias.

mean_bias will be updated in place.

Parameters

- **mean_bias** (*str*) – name of the mean bias bias
- **median_image** (*np.ndarray*) – 2d median image of the bias

3.1.6 Pipeline

Warning: Deprecated code. A new set of scripts to automatically derive pipeline reference files is being developed and will be integrated soon.

Create STIS Superdarks and Superbiases for the CCD detector.

refstis.pipeline.**clean_directory**(*root_path*)
Cleans directory of any fits files that do not end in _raw.fits

Warning: This WILL remove ANY files that do not match *_raw.fits. This includes any plots, txt files, other fits files, anything.

refstis.pipeline.**collect_new**(*observations_to_get*, *settings*)
Function to find and retrieve new datasets for given proposal.

refstis.pipeline.**grab_between**(*file_list*, *mjd_start*, *mjd_end*)
Select files between given start/end times

Parameters

- **file_list** (*list*) – files on which to filter
- **mjd_start** (*float*) – earliest time
- **mjd_end** (*float*) – latest time

Yields *filenames with mjd_start < TEXPSTRT < mjd_end*

refstis.pipeline.**make_pipeline_reffiles**(*root_folder*, *last_basedark=None*, *last_basebias=None*)
Make reference files like the refstis pipeline

1. Separate dark and bias datasets into week folders 2.

refstis.pipeline.**pull_info**(*foldername*)
Pull proposal and week number from folder name

A valid proposal is a string of 5 numbers from 0-9 A valid week is a string of 'wk' + 2 numbers ranging from 0-9. A valid biweek is the string 'biwk' + 2 numbers from 0-9.

Parameters **foldername** – string, name of folder to search in

Returns

- *proposal* – string, proposal number as string

- *week* – string, week of the anneal (wk01, etc)

refstis.pipeline.**pull_out_subfolders**(*root_folder*)
Walk through input folder and use regular expressions to return lists of the folders for each gain and for each week

Parameters **root_folder** – string, the folder to walk through

Returns

- *gain_folders* – list, containing folders for each gain
- *week_folders* – list, containing folders for each week

refstis.pipeline.**run**(*config_file='config.yaml'*)
Run the reference file pipeline

refstis.pipeline.**separate_period**(*base_dir*)
Separate observations in the base dir into needed folders.

Parameters **str** (*base_dir,*) – directory containing darks and biases to be split.

refstis.pipeline.**split_files**(*all_files*)
Split file list into two smaller lists for iraf tasks

Each list will have a selection from both early and late in time.

Parameters **all_files** (*list*) – full list of files

Returns **super_list** – list of lists containing the split list of all files

Return type list

3.1.7 Functions

refstis.functions.**RemoveIfThere**(*item*)
Remove a file only if it already exists

Parameters **item** (*str*) – file to be removed

Examples

```
>>> RemoveIfThere('/path/to/file.txt')
```

refstis.functions.**apply_dark_correction**(*filename*, *expstart*)
Perform temperature scaling to input dark file

All science extensions in the input filename will be scaled to the reference temperatue of 18.0 c.

Parameters

- **filename** (*str*) – full path to input FITS file
- **expstart** (*str*) – start time in MJD of the dataset

refstis.functions.**bd_calstis**(*joinedfile*, *thebiasfile=None*)
Run CalSTIS on the joined file

Header keywords will be set for ocrreject to work correctly and not flag regions outside the original aperture: APERTURE –> 50CCD APER_FOV –> '50x50' DARKCORR –> 'OMIT' FLATCORR –> 'OMIT'

Parameters

- **joinedfile** (*str*) – join of multiple input darks
- **thebiasfile** (*str, bool*) – the biasfile to be subtracted by basic2d

refstis.functions.**bd_crreject**(*joinedfile*)
Check if cosmic-ray rejection has been performed on input file

if cosmic-ray rejection has already been done on the input bias image, skip all calstis-related calibration steps

refstis.functions.**bias_subtract_data**(*filename*, *biasfile*)
Perform bias subtraction on input dataset

basic2d from calstis will be run on the input dataset with the steps DQICORR, BLEVCORR, and BIASCORR set to perform.

Parameters

- **filename** (*str*) – full path to input FITS file
- **biasfile** (*str*) – full path to the bias FITS file to be subtracted

Returns **filename** – full_path to the bias subtracted file

Return type str

refstis.functions.**count_imsets**(*file_list*)
Count the total number of imsets in a file list.

The total number of imsets is counted by dividing the total number of extensions (NEXTEND) by 3 (SCI, ERR, DQ).

Parameters **file_list** (*list*) – list of all files to be counted

Returns **total** – number of imsets

Return type int

refstis.functions.**divide_anneal_month**(*data_begin*, *data_end*, *database_path*, *N_period*)
This function divides an anneal month into anneal weeks and returns tuples of the start and end dates of the anneal weeks

refstis.functions.**figure_days_in_period**(*N_periods*, *N_days*, *add_remainder=False*)
Spreads out the extra days among the periods.

Notes

Extra days will be added to the periods beginning with the first, until there are no more, so the lengths may not be perfectly even but will not differ by more than a day.

Parameters

- **N_periods** (*int*) – total number of periods to be split into
- **N_days** (*float, int*) – total number of days to be divided

Returns **periods** – list of days/period: e.g. [8,8,8,7]

Return type list

Examples

```
>>> figure_days_in_period(4, 28)
[7, 7, 7, 7]
```

```
>>> figure_days_in_period(4, 29)
[8, 7, 7, 7]
```

```
>>> figure_days_in_period(3, 21)
[7, 7, 7]
```

refstis.functions.**figure_number_of_periods**(*number_of_days*, *mode*)

Determines the number of periods ('weeks') that the anneal 'month' should be split into.

Takes the number of days in the period and the mode ('WK' or 'BIWK') and returns the total number of periods.

Parameters

- **number_of_days** (*int*) – total number of days in the 'month'
- **mode** (*str*) – wk or biwk

Returns **number_of_periods** – the total number of periods to split the month into

Return type int

refstis.functions.**get_anneal_month_dates**(*data_begin*, *data_end*, *database_path*)

This function uses the anneal database to get the dates of the anneal

This is written under the assumption that data_begin and data_end fall in the same anneal period

data_begin and data_end are the start and end dates of the data and should be in mjd

refstis.functions.**get_keyword**(*file_list*, *keyword*, *ext=0*)

return the value from a header keyword over a list of files

if the value is not consistent accross the input files, an assertion error will be raised

refstis.functions.**make_path_safe**(*filename*)

Make a full path to file safe for use in FITS headers.

For full paths that are less than 67 characters, the filename is simply returned. When the full path is equal to or greater than 67, then the path is inserted into the environment as 'refdir' and the filename is returned as 'refdir$filename'. This will prevent the filename from being split accross multiple FITS keywords, and is a convention understood by many tasks/pipelines.

Parameters **filename** (*str*) – Full path + name to the file.

Returns **filename** – Safe filename that can fit in a single FITS keyword.

Return type str

Examples

```
>>> make_path_safe('/short/path/reference_file.fits')
/short/path/reference_file.fits
```

```
>>> make_path_safe('/really/really/really/really/really/really/really/really/
↪reference_file.fits')
refdir$reference_file.fits
```

refstis.functions.**make_residual**(*mean_bias*, *kern=(3, 15)*)
Create residual image

Median filter the median with a 15 x 3 box and subtract from the mean to produce the residual image.

refstis.functions.**msjoin**(*imset_list*, *out_name='joined_out.fits'*)
Replicate msjoin functionality in pure python

refstis.functions.**normalize_crj**(*filename*)
Normalize the input filename by exptim/gain and flush hdu

refstis.functions.**refaver**(*reffiles*, *combined_name*)
Average two reference files together using itools msarith.

Parameters

- **reffiles** (*list*) – List of reference files to be averaged together
- **combined_name** (*str*) – Output name of the combined file

refstis.functions.**send_email**(*subject=None*, *message=None*, *from_addr=None*, *to_addr=None*)
Send am email via SMTP server. This will not prompt for login if you are already on the internal network.

refstis.functions.**update_header_from_input**(*filename*, *input_list*)
Updates header of output file using keywords from the input data

If a header keyword is not consistent in this step, an error will be raised

3.1.8 Delivery

Warning: Deprecated code.

Functions necessary to check the reference files before delivery to CDBS

refstis.delivery.**plot_obset**(*folder*)
Check collapsed columns and rows against last month for irregularities

refstis.delivery.**regress**(*folder*)
Run *drk and *bia files in folder through CalSTIS to check for errors in processing

refstis.delivery.**set_descrip**(*folder*)
Make sure the descriptions are useful. Should be removed when using the new version of the pipeline.

3.1.9 pop_db

Warning: Deprecated code.

Script to populate database with STIS anneal month start and end times.

This is used by the STIS darks and bias reference file pipeline to determine the start and end of each anneal period, and to retrieve the appropriate dark and bias files.

refstis.pop_db.**get_directories**()
Loop over the anneal monitor directory and return list of directories containing anneal observations

Returns list, containing directory paths

Return type directories

refstis.pop_db.**grab_anneal_mjds**()
Loop over anneal directories and return a list of tuples containing useful mjd info to be populated into anneal database

Returns list, full of tupes containing information for database

Return type anneal_info

refstis.pop_db.**main**()
Main function to retrieve anneal info and populate database

refstis.pop_db.**pop_database**(*anneal_info*)
Populate anneal database with information gained pulled from anneal monitor observations

Parameters **anneal_info** – list, list of tuples to be populated into db

Returns

Return type None

3.1.10 retrieval

Warning: Deprecated in favor of Astroquery data retrieval.

refstis.retrieval.**build_xml_request**(*datasets*, *settings*)
Build the XML request for the given datasets

Parameters **datasets** (*list*) – A list of rootnames to download from MAST.

Returns **xml_request** – The XML request string.

Return type string

refstis.retrieval.**everything_retrieved**(*tracking_id*)
Check request status page to see if retreival is done.

Parameters **tracking_id** (*string*) – A submission ID string.

Returns

- **done** (*bool*) – Boolean specifying is data is retrieved or not.
- **killed** (*bool*) – Boolean specifying is request was killed.

refstis.retrieval.**submit_xml_request**(*xml_request*, *settings*)
Submit the XML request to the MAST archive.

Parameters **xml_request** (*string*) – The request XML string.

Returns **submission_results** – The XML request submission results.

Return type httplib object

3.2 Indices and tables

- genindex
- modindex

PYTHON MODULE INDEX

r

INDEX

S

U