IfA_Smeargle

Release 2020

Sparrow

Jun 29, 2020

CODE DOCUMENTATION:

1 ifa_smeargle **3**

1.1 ifa_smeargle package . . . 3

1.1.1 Subpackages . . . 3

1.1.1.1 ifa_smeargle.analysis package . . . 3

1.1.1.2 ifa_smeargle.core package . . . 6

1.1.1.3 ifa_smeargle.masking package . . . 20

1.1.1.4 ifa_smeargle.plotting package . . . 29

1.1.1.5 ifa_smeargle.reformat package . . . 32

1.1.1.6 ifa_smeargle.testing package . . . 38

1.1.1.7 ifa_smeargle.tutorial package . . . 40

1.1.2 Submodules . . . 41

1.1.2.1 ifa_smeargle.runtime module . . . 41

1.1.2.2 ifa_smeargle.special module . . . 42

1.1.3 Module contents . . . 43

2 Indices and tables **45**

Python Module Index **47**

Index **49**

This webpage documents the code's function and class docstrings. For an overview of this module as a whole, see the Github Wiki

CHAPTER

ONE

IFA_SMEARGLE

1.1 ifa_smeargle package

1.1.1 Subpackages

1.1.1.1 ifa_smeargle.analysis package

Submodules

ifa_smeargle.analysis.base_functions module

This is where common functions to all of the analysis code is written to.

`ifa_smeargle.analysis.base_functions.`**`create_directory_analysis_files`**(*data_directory*, *mask_file*, *filter_directory*, *filter_config_file*, *analysis_function*, *analysis_parameters*, *run*, *append=False*)

This function is the common function to compute and save the results of analysis functions.

Parameters

- **`data_directory`** (*`string`*) – The data directory by which all of data for the analysis is contained within.
- **`mask_file`** (*`string`*) – The mask, if desired, that will be applied to this analysis run.
- **`filter_directory`** (*`string`*) – The directory that has all of the synthesized filter files. If it does not exist, then the filters will be calculated if possible.
- **`filter_config_file`** (*`string`*) – The configuration file with the filtering parameters. This is used if only the filter is calculated rather than read.
- **`analysis_function`** (*`function`*) – The analysis function that will be used to calculate the analysis results.

- **`analysis_parameters`** (*`dictionary`*) – The dictionary for the arguments of the analysis function.
- **`run`** (*`boolean`*) – This is a flag to ensure that the analysis script should run. If it is False, it is not run.
- **`append`** (*`boolean (optional)`*) – This parameter specifies what to do with a file conflict. If True, the new analysis is appended to the other header; else, an exception is raised. Defaults to False.

Returns

Return type None

`ifa_smeargle.analysis.base_functions.`**`create_filter_from_configuration`**(*data_array*, *filter_config*)

This function applies all of the filters that are written based on the configurations supplied.

All filters are obtained automatically from the written code. If required configuration parameters are missing for any of the filters, they are automatically skipped.

Parameters

- **`data_array`** (*`ndarray`*) – The data array by which the filters will be calculated from.
- **`filter_config`** (*`ConfigObj or dictionary-like`*) – The configuration that the filters will use.

Returns **filter_array** – The filter, made of each of the individual filters as calculated.

Return type ndarray

`ifa_smeargle.analysis.base_functions.`**`create_filter_from_directory`**(*data_file*, *filter_directory*)

This function extracts the filter from a filter directory that corresponds to the data file.

Parameters

- **`data_file`** (*`string`*) – The name of the data file that is going to used as the reference to find any matching filter files. All directory information is stripped.
- **`filter_directory`** (*`string`*) – The directory by which the filters within will be checked if they match to the data file.

Returns **filter_array** – The filter, made of each of the individual filters as calculated.

Return type ndarray

`ifa_smeargle.analysis.base_functions.`**`get_analysis_fits_filenames`**(*data_directory*, *recursive=False*)

This function is to obtain all of the filter fits files within the directory provided. Mask fits files are those that have the extension *.analysis.fits*.

In general, this is a wrapper function around the normal fits file glob function adapted for filters.

Parameters

- **`data_directory`** (*`string`*) – The data directory that the filter fits files will be searched from.
- **`recursive`** (*`boolean (optional)`*) – If True, also search subdirectories for filter fits files.

Returns **analysis_fits_filenames** – The list of the analysis fits file names.

Return type list

`ifa_smeargle.analysis.base_functions.`**`script_batch_analysis`**(*config*)

This script runs all analysis in a batch fashion.

If the run flag of an analysis in the configuration file is True, it is run according to the parameters set.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

ifa_smeargle.analysis.heatmap module

This function computes all of the heat-map information needed for plotting.

`ifa_smeargle.analysis.heatmap.`**`analysis_heatmap`**(*data_array*, *mask_filter=None*, *run=True*)

This generates the dictionary which is the results from the analysis of the data to create a heat-map of the data.

Parameters

- **`data_array`** (*`ndarray`*) – The array of data that will be used to compute the heat-map results.
- **`mask_filter`** (*`ndarray (optional)`*) – The array of mask and filter values to consider in the calculations. True values denote the mask/filter is applied, False otherwise.
- **`run`** (*`boolean (optional)`*) – If True, the analysis is run, else it is not and will exit with None.

Returns **heatmap_results** – The results of the heat-map analysis.

Return type dictionary

`ifa_smeargle.analysis.heatmap.`**`script_analysis_heatmap`**(*config*)

This is the script form of *analysis_heatmap*. It does all the needed calculations, saves the results as separate fits files and also saves parameters that are not easy to save in fits files.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

ifa_smeargle.analysis.histogram module

This function computes all of the histogram information needed for plotting.

`ifa_smeargle.analysis.histogram.`**`analysis_gaussian_histogram`**(*data_array*, *bin_width*, *mask_filter=None*, *run=True*)

This generates the dictionary which is the results from the analysis of the data to create a heat-map of the data.

Parameters

- **`data_array`** (*`ndarray`*) – The array of data that will be used to compute the heat-map results.

- **bin_width** (*float*) – The bin width of the histogram bars.
- **mask_filter** (*ndarray (optional)*) – The array of mask and filter values to consider in the calculations. True values denote the mask/filter is applied, False otherwise.
- **run** (*boolean (optional)*) – If True, the analysis is run, else it is not and will exit with None.

Returns **histogram_results** – The results of the histogram analysis.

Return type dictionary

ifa_smeargle.analysis.histogram.**script_analysis_gaussian_histogram**(*config*)

This is the script form of *analysis_gaussian_histogram*. It does all the needed calculations, saves the results as separate fits files and also saves parameters that are not easy to save in fits files.

Parameters **config** (*ConfigObj*) – The configuration object that is to be used for this function.

Returns

Return type None

Module contents

1.1.1.2 ifa_smeargle.core package

Submodules

ifa_smeargle.core.configuration module

This contains the common functions dedicated to the reading, writing, and validating of configuration files.

ifa_smeargle.core.configuration.**copy_configuration_file**(*config_type*, *destination*, *file_name=None*)

This function finds a configuration file based on the configuration type, and copies it directly into the destination directory. If the configuration file doesn't not exist within the library, an exception is raised.

Simple substring testing is used to check if the configuration type matches.

Parameters

- **config_type** (*string*) – The type of configuration file that can be copied.
- **destination** (*string*) – The destination directory for the configuration file type copy.
- **file_name** (*string (optional)*) – This is an optional file name. The extension .ini will be added unless it is already present. No directory information should be provided here. Defaults to the copied file name.

Returns **config_path** – The new path that the configuration can be found at.

Return type string

ifa_smeargle.core.configuration.**extract_configuration**(*config_object*, *keys*)

This is a wrapper to obtain the configuration parameter from the configuration tag. This function includes proper error handling.

Parameters

- **config_object** (*ConfigObj*) – The configuration object that is going to be tested.
- **keys** (*list*) – The keys that should be called, in order.

Returns **value** – The value that the keys was containing.

Return type object

`ifa_smeargle.core.configuration.`**`read_configuration_file`**(*config_file_name*, *specification_file_name*)

This reads in a configuration file name specified by the ConfigObj module. Verification is required.

Parameters

- **`config_file_name`** (*`string`*) – The path and file name of the configuration. If None, then the defaults are returned.
- **`specification_file_name`** (*`string`*) – The path and file name of the specification/validation file.

Returns **configuration** – The configuration according to the specification and configuration files.

Return type ConfigObj

`ifa_smeargle.core.configuration.`**`write_configuration_file`**(*config_file_name*, *config_object*, *specification_file_name=None*)

This function takes a configuration object and writes it to file. If provided with a default, it will also write it to file.

Parameters

- **`config_file_name`** (*`string`*) – The file name that the configuration object will be written to.
- **`config_object`** (*`ConfigObj`*) – The configuration object that will be written to file.
- **`specification_file_name`** (*`string (optional)`*) – The specification file that is used in the event defaults are wanted.

Returns

Return type None

ifa_smeargle.core.error module

Defining custom errors because Python does not have all of the needed error categories.

`exception` `ifa_smeargle.core.error.`**`AssumptionError`**(*message=None*)

Bases: *`ifa_smeargle.core.error.Ifas_BaseException`*

This error is reserved for instances where something unexpected has occurred because of a flaw in the understanding of assumptions about Python or module functions.

`exception` `ifa_smeargle.core.error.`**`BrokenLogicError`**(*message=None*)

Bases: *`ifa_smeargle.core.error.Ifas_BaseException`*

This error is encountered when the program enters in a place it should not be able to. Incorporated mostly for safety; usually not the fault of the user.

`exception` `ifa_smeargle.core.error.`**`ConfigurationError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is normally encountered when there are problems with configuration processing (usually because the configuration class is incorrect).

exception `ifa_smeargle.core.error.`**`ConfigurationWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when there are issues with the configuration class and that data is missing. However, the missing data does not warrant an exception.

exception `ifa_smeargle.core.error.`**`DataError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is used when there is an issue with the fundamental data that this program or module cannot fix. The user should be able to figure out what is the problem.

exception `ifa_smeargle.core.error.`**`DataWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when there is an issue with the fundamental data that this program or module cannot fix but can still work around. The user should be able to figure out what is the problem.

exception `ifa_smeargle.core.error.`**`DeprecatedError`**(*message=None*)

Bases: *`ifa_smeargle.core.error.Ifas_BaseException`*

This is used when the code should be using a different equivalent function. This is mostly used for cases where a warning has already been issued, or to clean up the core sections of the code during testing.

exception `ifa_smeargle.core.error.`**`DeprecatedWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when there are some functions that are used but have since been replaced with better functions, or where the previous function is not very stable or integrated with the rest of the functions.

exception `ifa_smeargle.core.error.`**`DevelopmentError`**(*message=None*)

Bases: *`ifa_smeargle.core.error.Ifas_BaseException`*

This is used when the code is improperly written as a result of development solely by one or many parties. An error here usually is because some assumptions on development has been imposed that may have been forgotten about.

exception `ifa_smeargle.core.error.`**`ErrorWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is to inform that there was an error that was thrown as a warning (either as a critical warning log or something else.

exception `ifa_smeargle.core.error.`**`ExportingError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is used when the program attempts to export data in some way, but is unable to. Named ExportingError to keep with ImportingError.

exception `ifa_smeargle.core.error.`**`ExportingWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used to warn the user about exports and some things that may be helpful to know about them. Such exports are generally writing files to disk.

exception `ifa_smeargle.core.error.`**`Ifas_BaseException`**

Bases: `BaseException`

exception `ifa_smeargle.core.error.`**`Ifas_Exception`**

Bases: `Exception`

exception `ifa_smeargle.core.error.`**`Ifas_Warning`**

Bases: `UserWarning`

exception `ifa_smeargle.core.error.`**`IllogicalProsedureError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is thrown when the program would attempt something that does not make scene. This is usually due to issues with configuration errors.

exception `ifa_smeargle.core.error.`**`ImportingError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is used when reading configuration files, or other data files is not going as it should. Named ImportingError to avoid conflicts with ImportError.

exception `ifa_smeargle.core.error.`**`ImportingWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when there are issues loading a file, but it can be handled using some assumptions.

exception `ifa_smeargle.core.error.`**`ImprecisionError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is used when there are critical issues with numerical precision because of the volume of data or the very low/high numbers involved. It is also just used when data may be chaotic.

exception `ifa_smeargle.core.error.`**`ImprecisionWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when there may be issues with numerical precision because of the volume of data or the very low/high numbers involved.

exception `ifa_smeargle.core.error.`**`IncompleteError`**(*message=None*)

Bases: *`ifa_smeargle.core.error.Ifas_BaseException`*

This used when the code is trying to use a function that is incomplete or not usable.

exception `ifa_smeargle.core.error.`**`InputError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is used when the user does not input a proper or logical entry.

exception `ifa_smeargle.core.error.`**`InputWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when the user inputs something that is questionable, but not wrong.

exception `ifa_smeargle.core.error.`**`MagicError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is used when any routine would utilize magic/hard-coded values for the purposes of any process where said numbers are magic. This is mostly as a programming warning to the user that behavior with magic numbers may not always be expected or logical. This error form is used for higher warning levels, and if the user wants an interrupt when upgrading.

exception `ifa_smeargle.core.error.`**`MagicWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when any routine would utilize magic/hard-coded values for the purposes of any process where said numbers are magic. This is mostly as a programming warning to the user that behavior with magic numbers may not always be expected or logical.

exception `ifa_smeargle.core.error.`**`MaskingError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is used when a mask cannot be applied or where there are fatal issues with calculating the mask.

exception `ifa_smeargle.core.error.`**`MaskingWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when any masking or filtering routine (especially in the masking and filtering scripts) fails to mask any pixels or something else is amiss. It is not a bad thing, but it can be helpful to know.

exception `ifa_smeargle.core.error.`**`MemoryWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used to warn the user that the procedures that follow would require a lot of memory RAM. If instead it would produce a large file(s), StorageWarning should be used.

exception `ifa_smeargle.core.error.`**`ModelingError`**

Bases: *`ifa_smeargle.core.error.Ifas_Exception`*

This error is used when there are issues with applying or fitting models to a particular set of data. May be used hand-in-hand with DataError.

exception `ifa_smeargle.core.error.`**`OverwriteWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used to warn the user that a file has been overwritten, most likely because of conflicting file names.

exception `ifa_smeargle.core.error.`**`ReductionWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when normally unusual parameters are used for data reduction. The user is trusted in their procedures.

exception `ifa_smeargle.core.error.`**`StorageWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when the large file(s) would be written to the hard drive. If instead a lot of RAM would be used, it is better to use MemoryWarning.

exception `ifa_smeargle.core.error.`**`TerminalError`**(*message=None*)

Bases: *`ifa_smeargle.core.error.Ifas_BaseException`*

This is used when something has gone terribly wrong. It is best to contact the maintainers or Sparrow.

exception `ifa_smeargle.core.error.`**`TimeWarning`**

Bases: *`ifa_smeargle.core.error.Ifas_Warning`*

This warning is used when any method called may take a long time to compute or execute. This allows the user to stop and change if desired.

`ifa_smeargle.core.error.`**`ifas_absolute_silence`**()

This context manager silences any and all messages, it basically is a wrapper around all other general Ifas context managers (even if there is some overlap).

`ifa_smeargle.core.error.`**`ifas_debug`**(*message*, *console_print=False*)

This is a wrapper function for the printing of debug messages.

Given the nature of debug messages, it should be clear that it is a debug message, and should also have the proper silencing capabilities.

Parameters

- **`message`** (*`string`*) – The message that is to be sent as the debug message.
- **`console_print`** (*`boolean (optional)`*) – This ensures that the info message is always printed regardless of the logging level set by the logging utility. Default is False.

Returns

Return type nothing

`ifa_smeargle.core.error.`**`ifas_debug_block`**`()`

This is a wrapper function for encasing debugging code.

The execution of code within a debug block is used to contain easily printed debug information. Debug messages should use the debug function *`ifas_debug()`*

`ifa_smeargle.core.error.`**`ifas_disable_debug`**`()`

This context manager turns all debug messages off for the duration of the context. Given that debug messages are generally off in the first place, usage may be rare.

`ifa_smeargle.core.error.`**`ifas_enable_debug`**`()`

This context manager turns all debug messages on for the duration of the context.

`ifa_smeargle.core.error.`**`ifas_error`**`(`*type*, *message*`)`

This is a wrapper around the logging's error command. However, it does not raise an error. Instead, an error may be sent to and it will be converted into a proper warning.

Parameters

- **`type`** (*`ExecptionsClass`*) – The error that could be raised, but it is more manageable as a logged error.
- **`message`** (*`string`*) – The message that the error is to give.

Returns

Return type None

`ifa_smeargle.core.error.`**`ifas_info`**`(`*message*, *console_print=None*`)`

This is a wrapper function to print helpful information.

Printing information as the function(s) go on is very helpful. However, using the normal print function doesn't allow for some level of customization and ease of handling. Hence, function for uniformity.

Parameters

- **`message`** (*`string`*) – The informational message that is to be printed.
- **`console_print`** (*`boolean (optional)`*) – This ensures that the info message is always printed regardless of the logging level set by the logging utility.

Returns

Return type nothing

`ifa_smeargle.core.error.`**`ifas_log_warning`**`(`*type*, *message*`)`

Just a wrapper function around logging's built-in warn. This is to only log a warning but not to display it as a standard warning.

Parameters

- **`type`** (*`Warnings Class`*) – The warning class type.
- **`message`** (*`string`*) – The message that the warning is to give to the user.

`ifa_smeargle.core.error.`**`ifas_silence_all_warnings`**`()`

This context manager silences all warnings. Warnings should not be printed.

`ifa_smeargle.core.error.`**`ifas_silence_ifas_warnings`**`()`

This context manager silences all Ifas based warnings, all other warnings are still valid.

`ifa_smeargle.core.error.`**`ifas_silence_info_message`**`()`

This context manager silences all informational messages that may be printed.

`ifa_smeargle.core.error.`**`ifas_silence_nonifas_warnings`**`()`

This context manager silences all non-Ifas based warnings, all other warnings are still valid.

`ifa_smeargle.core.error.`**`ifas_silence_specific_warnings`**`(`*silenced_warning_type*`)`

This context manager silences all warnings of a given type. Depending on what was inputed.

Parameters **`silenced_warning_type`** (*`WarningType`*) – The warning that should be silenced.

`ifa_smeargle.core.error.`**`ifas_warning`**`(`*type*, *message*`)`

Just a wrapper function around the warning's warn command.

This wrapper was really only for the logical flow of Sparrow.

Parameters

- **`type`** (*`Warnings Class`*) – The warning class type.
- **`message`** (*`string`*) – The message that the warning is to give to the user.

Returns

Return type None

ifa_smeargle.core.io module

These are functions related to the reading, writing, and manipulation of files (data, configuration, or otherwise).

`ifa_smeargle.core.io.`**`append_astropy_header_card`**`(`*file_name*, *header_cards*, *comment_cards=None*`)`

This is a function to add header card entries into the header of a fits file. This uses dictionaries to achieve said result.

Parameters

- **`file_name`** (*`string`*) – This is the path of the file to be written, either relative or absolute.
- **`header_cards`** (*`dictionary`*) – The header entries to be added to the header file. Please note that the keys of the dictionary must be no more than 8 characters; otherwise a HIERARCH card will be used.
- **`comment_cards`** (*`dictionary (optional)`*) – The comment entries to be added to the header file. The keys of the comment dictionary and the *header_cards* must line up.

Returns **hdul_file** – The file object that was written to disk. If `hdu_object` was provided, it is returned with its header changed.

Return type Astropy PrimaryHDU

`ifa_smeargle.core.io.`**`archive_data_directory`**`(`*data_directory*, *archive_name=None*, *archive_type='bztar'*, *silent=False*`)`

Creates a file archive of a copy of the data files contained within a directory.

This function creates an archive of a data directory, preserving a copy of data. However, note that this function generally takes a bit of time if there are a lot of files or if the files are particularly large.

Please note that this function archives recursively. Non-data files and non-required files should not be in the a given data/archiving directory.

Parameters

- **`data_directory`** (*`string`*) – The directory that the data is contained within.

- **`archive_name`** (*`string (optional)`*) – The name of the archive that is to be created. If not provided, a default is provided that contains the time-stamp of its creation.
- **`archive_extension`** (*`string (optional)`*) – The extension of the archive. Note that only some archives are supported. Default is `bztar`. See `shutil.get_archive_formats()` for more information on available archive formats.
- **`silent`** (*`boolean (optional)`*) – Turn off all warnings and information sent by this function and functions below it.

Returns **archive_path** – The path of the archive file.

Return type string

`ifa_smeargle.core.io.`**`dearchive_data_directory`**(*archive_name*, *alternate_directory=None*, *silent=False*)

Unpacks a file archive of a copy of the data files contained within an archive.

This function unpacks an archive of a data directory. However, note that this function generally takes a bit of time if there are a lot of files or if the files are particularly large.

Parameters

- **`archive_name`** (*`string`*) – The name of the archive that is to be unpacked.
- **`alternate_directory`** (*`string`*) – The path that an alternate directory that the archive should be unpacked it.
- **`silent`** (*`boolean (optional)`*) – Turn off all warnings and information sent by this function and functions below it.

Returns **unarchive_path** – The path of the archive file.

Return type string

`ifa_smeargle.core.io.`**`get_fits_filenames`**(*data_directory*, *sub_extension=None*, *recursive=False*)

This function is a wrapper function around the glob command to get all fits files.

Parameters

- **`data_directory`** (*`string`*) – The data directory that the fits files will be search for from.
- **`sub_extension`** (*`string (optional)`*) – This allows for the focus of obtaining fits files that is only like a .subextension.fits file, depending on what the string *sub-extension* is.
- **`recursive`** (*`boolean (optional)`*) – If True, also search subdirectories for fits files.

Returns

- **fits_filenames** (*list*) – The list of the fits file names.
- .. *note::* – If *data_directory* is instead a valid fit file, then only that fit file is returned in a list. This allows scripts to only be run on single files. (Recursive must be False.)

`ifa_smeargle.core.io.`**`get_subdirectories`**(*directory*)

This returns a list of the immediate subdirectories of the provided directory.

Credit: https://stackoverflow.com/a/59938961

Parameters **`directory`** (*`string`*) – The directory that subdirectories will be found from.

Returns **subdirectories** – The list of subdirectories.

Return type list

`ifa_smeargle.core.io.`**`read_fits_file`**(*file_name*, *extension=0*, *silent=False*)

A function to ensure proper loading/reading of fits files.

This function, as its name, opens a fits file. It returns the Astropy HDU file. This function is mostly done to ensure that files are properly closed. It also extracts the needed data and header information from the file.

Parameters

- **`file_name`** (*`string`*) – This is the path of the file to be read, either relative or absolute.
- **`extension`** (*`int or string (optional)`*) – The desired extension of the fits file. Defaults to primary structure.
- **`silent`** (*`boolean (optional)`*) – Turn off all warnings and information sent by this function and functions below it.

Returns

- **hdu_file** (*HDULists*) – The Astropy object representing the fits file.
- **hdu_header** (*Header*) – The Astropy header object representing the headers of the given file.
- **hdu_data** (*ndarray*) – The Numpy representation of a fits file data.

`ifa_smeargle.core.io.`**`rename_by_parallel_append`**(*file_names*, *appending_names*, *directory=None*)

This function renames all files by appending a string to their name. The file name extension is not modified.

Parameters

- **`file_names`** (*`array_like`*) – The list of the file names that is to be renamed.
- **`appending_names`** (*`array_like`*) – The list of the appending strings that are going to be used for the renaming process.
- **`directory`** (*`string (optional)`*) – A directory that contains all of the files that are going to be renamed. Does not handle directories recursively.

Returns

Return type None

`ifa_smeargle.core.io.`**`rename_by_parallel_replace`**(*file_names*, *file_renames*, *directory=None*)

Renames files provided parallel name arrays. This only works for fits files.

Given two same length lists of file names, one pre-rename and one post-rename, this function renames them accordingly. A directory is also an option, and the file name list will be derived from that.

This only works for one type of file extension, or leave the string blank for all files.

Parameters

- **`file_names`** (*`array_like`*) – The list of the file names that is to be renamed.
- **`file_renames`** (*`array_like`*) – The list of the file names that are going to be used for the renaming process.
- **`directory`** (*`string (optional)`*) – A directory that contains all of the files that are going to be renamed. Does not handle directories recursively.

Returns

Return type None

`ifa_smeargle.core.io.`**`write_fits_file`**(*file_name*, *hdu_header*, *hdu_data*, *hdu_object=None*, *save_file=True*, *overwrite=False*, *silent=False*)

A function to ensure proper writing of fits files.

This function writes fits files given the data and header file. The file name should be a complete path and must also include the file name.

Parameters

- **`file_name`** (*`string`*) – This is the path of the file to be written, either relative or absolute.
- **`hdu_header`** (*`Header`*) – The Astropy header object representing the headers of the given file.
- **`hdu_data`** (*`ndarray`*) – The Numpy representation of a fits file data.
- **`hdu_object`** (*`Astropy HDUList (optional)`*) – An astropy HDUList object, if provided, this object takes priority to be written, the rest are ignored.
- **`save_file`** (*`boolean (optional)`*) – If `True`, then the fits file will be written to file, else, just the instance will be returned.
- **`overwrite`** (*`boolean (optional)`*) – If `True`, if there exists a file of the same name, overwrite.
- **`silent`** (*`boolean (optional)`*) – Turn off all warnings and information sent by this function and functions below it.

Returns **hdul_file** – The file object that was written to disk. If `hdu_object` was provided, it is returned untouched.

Return type Astropy HDUList

ifa_smeargle.core.magic module

This is to hold functions that are pure 'magic'. That is, functions that perform what they are supposed to do, but, are fragile because their operations are not defined by conventional understanding, but instead as "it works, I don't know why, but it does".

`ifa_smeargle.core.magic.`**`magic_inital_gaussian_parameters`**(*x_data*, *y_data*)

This is a magic program to derive the guesses of Gaussian fit parameters from an array of values.

ifa_smeargle.core.mathematics module

`ifa_smeargle.core.mathematics.`**`generate_numpy_bin_width_array`**(*data_array*, *bin_width*, *local_minimum=None*, *local_maximum=None*)

Matplotlib does not support having input bin withs; this returns a valid form.

This function just generates a valid bin value list provided a given bin widths. If the `local_maximum` or `local_minimum value` is not provided, then the absolute maximum and minimum of the provided array is used.

If mod[(max - min), bin_width] != 0, the last/highest bin is disenfranchised. This function can adapt to masked arrays.

Parameters

- **`data_array`** (*`ndarray`*) – The data to which the bins will be calculated from; ignored if the two local maximum and minimum parameters are provided.
- **`bin_width`** (*`float`*) – The width of the bins.
- **`local_minimum`** (*`float (optional)`*) – A predefined minimum that the calculating function should use.
- **`local_maximum`** (*`float (optional)`*) – A predefined maximum that the calculating function should use.

Returns **bin_list_values** – A list of values that can be fed into matplotlib to emulate binning by a value width.

Return type ndarray

`ifa_smeargle.core.mathematics.`**`generate_prime_numbers`**(*index*, *count=1*)

This function returns prime numbers from a downloaded list provided by OEIS.

This function returns the prime number at the 0-indexed location (e.g. 0 index is 2, 1 index is 3, etc.). It will also return the *count* number of prime numbers including and including the prime number at the specified index.

Parameters

- **`index`** (*`int`*) – The 0-indexed index for the prime number to start at. If it is negative, the prime numbers will be randomized from 2 to 104729 (the first and last prime numbers in the list).
- **`count`** (*`int`*) – The number of prime numbers that will be returned. It must be greater than 1.

Returns **prime_array** – The prime numbers. The numbers are sorted.

Return type array

`ifa_smeargle.core.mathematics.`**`ifas_gaussian_function`**(*input*, *mean*, *stddev*, *amplitude*)

This is a wrapper function around Astropy's Gaussian function. This takes the input of a function, and gives it an output according to the Gaussian parameters provided. The function itself is also returned.

Parameters

- **`input`** (*`array`*) – The input into the Gaussian function.
- **`mean`** (*`float`*) – The mean of the Gaussian function.
- **`stddev`** (*`float`*) – The standard deviation of the Gaussian function.
- **`amplitude`** (*`float`*) – The amplitude to of the Gaussian function.

Returns

- **output** (*array*) – The output when the Gaussian function is calculated from the input.
- **gaussian_function** (*function*) – The Gaussian function with the parameters provided already given.

`ifa_smeargle.core.mathematics.`**`ifas_large_integer_array_product`**(*integer_array*)

Arrays of large integer numbers, or large integer results, do not operate as well with multiplication.

The purpose of this function is to compute the product of the integer as accurately as possible without error. The main source of error is over/underflow and precision error from the lack of byte allocation.

If the numbers are not integers, they will be forced into integers.

Parameters **`integer_array`** (*`array-like`*) – The array of integers that will be multiplied.

Returns

- **product** (*integer*) – The product of the entire array.
- **ln_product** (*float*) – The natural log of the product to a high precision.
- **log10_product** (*float*) – The base 10 log of the product to a high precision.

`ifa_smeargle.core.mathematics.`**`ifas_masked_mean`**(*array*, *axis=None*)

This returns the true mean of the data. It only counts valid data.

There are outstanding problems with how the masked arrays handle means. For some reason, there is no np.ma.nanmean function. This adds that functionality.

Parameters

- **`array`** (*`ndarray`*) – The value or array of values by which the mean will be taken from.
- **`axis`** (*`int`*) – The axis that the mean will be taken over.

Returns **true_mean** – The mean of the array along which ever axis was given.

Return type float or ndarray

`ifa_smeargle.core.mathematics.`**`ifas_masked_median`**(*array*, *axis=None*)

This returns the true median of the data. It only counts valid data.

There are outstanding problems with how the masked arrays handle medians. For some reason, there is no np.ma.nanmedian function. This adds that functionality.

Parameters

- **`array`** (*`ndarray`*) – The value or array of values by which the median will be taken from.
- **`axis`** (*`int`*) – The axis that the median will be taken over.

Returns **true_median** – The median of the array along which ever axis was given.

Return type float or ndarray

`ifa_smeargle.core.mathematics.`**`ifas_masked_std`**(*array*, *axis=None*)

This returns the true standard deviation of the data. It only counts valid data.

There are outstanding problems with how the masked arrays handle stds. For some reason, there is no np.ma.nanstd function. This adds that functionality.

Parameters

- **`array`** (*`ndarray`*) – The value or array of values by which the standard deviation will be taken from.
- **`axis`** (*`int`*) – The axis that the median will be taken over.

Returns **true_std** – The standard deviation of the array along which ever axis was given.

Return type float or ndarray

`ifa_smeargle.core.mathematics.`**`ifas_robust_mean`**(*array*)

This provides a more robust measurement of the mean of the array of values.

Parameters **`array`** (*`ndarray`*) – The array of values by which the mean will be calculated from.

Returns **robust_mean** – The robust mean value.

Return type float

`ifa_smeargle.core.mathematics.`**`ifas_robust_std`**(*array*)

This provides a more robust measurement of the standard deviation of the array of values.

Parameters **`array`** (*`ndarray`*) – The array of values by which the standard deviation will be calculated from.

Returns **robust_std** – The robust standard deviation value.

Return type float

ifa_smeargle.core.modeling module

This module contains functions that are strictly the backbone of many of the modeling and fitting functionality.

`ifa_smeargle.core.modeling.`**`fit_gaussian_function`**(*x_data*, *y_data*, *inital_guesses*)

This function fits a Gaussian function to a specific set of data.

Gaussian fitting is hard, this function exists as a port so that all fitting functions use the same algorithm and said algorithm is easy to change.

Parameters

- **`x_data`** (*`ndarray`*) – The X data that the Gaussian will be fit over.
- **`y_data`** (*`ndarray`*) – The Y data that the Gaussian will be fit over.
- **`inital_guesses`** (*`dictionary`*) – Initial guesses for the Gaussian fitting function. The allowed inputs and their keys are:
  - **'mean'** [The initial guess of the Gaussian function's] mean. Defaults to 0.
  - **'studded'** [The initial guess of the Gaussian] function's standard deviation. Defaults to 1.
  - **'amplitude'** [The initial guess of the Gaussian] function's amplitude. Defaults to 1.

Returns

- **gaussian_function** (*function*) – A callable function that when provided an X value, it will return the value of the function.
- **gaussian_parameters** (*dictionary*) – A compiled dictionary of all of the parameters of the Gaussian fit.

`ifa_smeargle.core.modeling.`**`fit_histogram_gaussian_function`**(*data_array*, *bin_width*)

This function fits a Gaussian function to a specific set of data.

Gaussian fitting is hard, this function exists as a port so that all fitting functions use the same algorithm and said algorithm is easy to change. This applies it to the histogram of the data.

Parameters

- **`data_array`** (*`ndarray`*) – The data that the histogram Gaussian function is fitting.
- **`bin_width`** (*`float`*) – The width of the bins to use for the histogram fitting function.

Returns

- **gaussian_function** (*function*) – A callable function that when provided an X value, it will return the value of the function.
- **gaussian_parameters** (*dictionary*) – A compiled dictionary of all of the parameters of the Gaussian fit.

ifa_smeargle.core.string_formatting module

This module deals with the formatting and processing of strings where either otherwise not provided or as a wrapper function for convience.

`ifa_smeargle.core.string_formatting.`**`combine_pathname`**(*directory=None*, *file_name=None*, *extension=None*)

This is the opposite of splitting path names.

Parameters

- **`directory`** (*`string or list (optional)`*) – This is the directory component that the path should be attached to. If it is a list, the directory components are strung together in order.
- **`file_name`** (*`string or list (optional)`*) – This is the file name component that the path should be attached to. If it is a list, the file name components are strung together in order.
- **`extension`** (*`string or list (optional)`*) – This is the extension component that the path should be attached to. If it is a list, the extension components are strung together in order.

Returns **pathname** – The pathname that is created by combining the parts above.

Return type string

`ifa_smeargle.core.string_formatting.`**`format_shutil_archive_extensions`**(*archive_string*)

This function formats an extension and type string to conform with shutil allowed extensions. This does not handle entire file names.

Parameters **`archive_string`** (*`string or boolean`*) – This is any version (either archive type or file extension) that is allowed under shutil archive maker.

Returns

- **archive_type** (*string*) – The archive type that shutil allows for archive making.
- **archive_extension** (*string*) – The associated file extension that shutil allows.

`ifa_smeargle.core.string_formatting.`**`purge_substrings`**(*string*, *substrings*)

Deletes all occurrences of any substring provided from the original string.

Parameters

- **`string`** (*`string`*) – The string to be purged of all substrings.
- **`substrings`** (*`string or list`*) – The substrings in a list, or a string by itself.

Returns **purged_string** – The string after it had all substring occurrences taken out.

Return type string

`ifa_smeargle.core.string_formatting.`**`random_string`**(*characters*, *length*)

This function returns a random string of characters of some length from a set of characters to use.

Credit to: https://stackoverflow.com/a/23728630

Parameters

- **`characters`** (*`string`*) – The total available characters to use. The order does not matter.
- **`length`** (*`int`*) – The length of the random string.

Returns **random_string** – The random string of proper length.

Return type string

`ifa_smeargle.core.string_formatting.`**`remove_prefix`**(*string*, *prefix*)

This function removes the prefix string of a string. If the prefix does not exist, the string is returned unchanged.

See https://stackoverflow.com/a/16891418

Parameters

- **`string`** (*`string`*) – The string that the prefix of which will be taken out.
- **`prefix`** (*`string`*) – The prefix.

Returns **base** – The string, without the prefix.

Return type string

`ifa_smeargle.core.string_formatting.`**`split_pathname`**(*pathname*)

This function splits a pathname into the directory, file, and extension names.

Parameters **`pathname`** (*`string`*) – The path that is to be split.

Returns

- **directory** (*string*) – The directory component of the path.
- **file_name** (*string*) – The file name component of the path.
- **extension** (*string*) – The extension component of the path.

Module contents

This module is dedicated to holding core or common functions to all other modules related to the IfA-Smeargle library.

In general, these should not be used to manipulate data from a pipeline script directly.

1.1.1.3 ifa_smeargle.masking package

Submodules

ifa_smeargle.masking.base_functions module

This is the common functions that are used through the masking modules.

`ifa_smeargle.masking.base_functions.`**`create_directory_filter_files`**(*data_directory*, *filter_function*, *filter_arguments*, *filter_file_tag*, *recursive=False*, *subfolder=True*, *run=False*)

This function is the common function to create filters for the data within the data directory.

Parameters

- **`data_directory`** (*`string`*) – The data directory that contain the data which the mask are for.
- **`filter_function`** (*`function`*) – The filtering function which contains the main code operations for creating the filter.
- **`filter_arguments`** (*`dictionary`*) – The dictionary for the arguments of the filtering function.
- **`recursive`** (*`boolean (optional)`*) – If True, then the subdirectories of the data_directory will also be searched for data.
- **`subfolder`** (*`boolean (optional)`*) – If True, a sub-folder containing all of the masks will be created. If it exists, then the filters are added to it.
- **`run`** (*`boolean (optional)`*) – If True, the filter is ran and completed. Else, a warning is raised, it is not computed, and nothing is returned.

Returns

Return type None

`ifa_smeargle.masking.base_functions.`**`create_directory_mask_file`**`(`*data_directory*, *mask_function*, *mask_arguments*, *mask_file_name*, *recursive=False*, *subfolder=True*, *run=False*`)`

This function is the common function to create a mask for the data within the data directory.

Parameters

- **`data_directory`** (*`string`*) – The data directory that contain the data which the mask are for.
- **`mask_function`** (*`function`*) – The masking function which contains the main code operations for creating the mask.
- **`mask_arguments`** (*`dictionary`*) – The dictionary for the arguments of the masking function.
- **`recursive`** (*`boolean (optional)`*) – If True, then the subdirectories of the data_directory will also be searched for data.
- **`subfolder`** (*`boolean (optional)`*) – If True, a sub-folder containing all of the masks will be created. If it exists, then the masks are added to it.
- **`run`** (*`boolean (optional)`*) – If True, the mask is ran and completed. Else, a warning is raised, it is not computed, and nothing is returned.

Returns

Return type None

`ifa_smeargle.masking.base_functions.`**`get_filter_fits_filenames`**`(`*data_directory*, *recursive=False*`)`

This function is to obtain all of the filter fits files within the directory provided. Mask fits files are those that have the extension *.filter.fits*.

In general, this is a wrapper function around the normal fits file glob function adapted for filters.

Parameters

- **data_directory** (*string*) – The data directory that the filter fits files will be searched from.
- **recursive** (*boolean (optional)*) – If True, also search subdirectories for filter fits files.

Returns **filter_fits_filenames** – The list of the filter fits file names.

Return type list

`ifa_smeargle.masking.base_functions.`**`get_mask_fits_filenames`**(*data_directory*, *recursive=False*)

This function is to obtain all of the mask fits files within the directory provided. Mask fits files are those that have the extension *.mask.fits*.

In general, this is a wrapper function around the normal fits file glob function adapted for masks.

Parameters

- **data_directory** (*string*) – The data directory that the mask fits files will be search from.
- **recursive** (*boolean (optional)*) – If True, also search subdirectories for mask fits files.

Returns **mask_fits_filenames** – The list of the mask fits file names.

Return type list

`ifa_smeargle.masking.base_functions.`**`synthesize_masks`**(**args*, ***kwargs*)

This is a function to combine many masks into one single mask.This argument does not take any keyword arguments. All of the masks must be the same size. (In general, the first mask is considered the correct mask.)

Parameters

- ***args** (*list*) – This should be a collection of array based masks.
- ****kwargs** (*dictionary*) – This catches any keyword arguments sent through.

Returns **synthesized_mask** – The combined mask made of all of the inputted masks.

Return type array

ifa_smeargle.masking.filters module

This contains all of the value based filters when doing analysis.

`ifa_smeargle.masking.filters.`**`filter_exact_value`**(*data_array*, *exact_value*)

This function computes a filter for all pixel values equal to some exact value.

Float equality comparisons are dependent on tolerances. The main back function used is *numpy.isclose*.

Parameters

- **data_array** (*ndarray*) – The data array that the filter will be calculated from.
- **exact_value** (*float*) – The value that data values close will be tagged as filtered.

Returns **final_filter** – The filter as computed by this function.

Return type ndarray

`ifa_smeargle.masking.filters.`**`filter_invalid_value`**(*data_array*)

This filter applies a mask to all numerically invalid inputs on a programing side.

Numbers that are usually infinite or some other nonsensical quantity serve no real usage in calculations further downstream. Therefore, they are masked here.

See numpy.ma.fix_invalid for what is considered invalid.

Parameters **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.

Returns **final_mask** – A boolean array for pixels that are masked (True) or are valid (False).

Return type ndarray -> dictionary

`ifa_smeargle.masking.filters.`**`filter_maximum_value`**(*data_array*, *maximum_value*)

This function computes a filter for all pixel values strictly more than some maximum value.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the filter will be calculated from.
- **`maximum_value`** (*`float`*) – The value that data values strictly more than will be tagged as filtered.

Returns **final_filter** – The filter as computed by this function.

Return type ndarray

`ifa_smeargle.masking.filters.`**`filter_minimum_value`**(*data_array*, *minimum_value*)

This function computes a filter for all pixel values strictly less than some minimum value.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the filter will be calculated from.
- **`minimum_value`** (*`float`*) – The value that data values strictly less than will be tagged as filtered.

Returns **final_filter** – The filter as computed by this function.

Return type ndarray

`ifa_smeargle.masking.filters.`**`filter_percent_truncation`**(*data_array*, *top_percent*, *bottom_percent*)

This filter truncates the top and bottom percent of pixels provided.

The values `top_percent` and `bottom_percent` notate the percentage of pixels from top and bottom of the data array (in value) that should be filtered. The pixels masked are independent on the previous masks applied.

If the percentage of pixels leads to a non-integer number of pixels to be masked, the number is floored. All pixels that have a same value as the limiting pixel value for the top and bottom percent of pixels are also masked.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the filter will be calculated from.
- **`top_count`** (*`float`*) – The percent of pixels from the top (highest value) of the array that is to be masked.
- **`bottom_count`** (*`int`*) – The percent of pixels from the bottom (lowest value) of the array that is to be masked.

Returns **final_filter** – The filter as computed by this function.

Return type ndarray

`ifa_smeargle.masking.filters.`**`filter_pixel_truncation`**(*data_array*, *top_count*, *bottom_count*)

This filter truncates the top and bottom number of pixels provided.

The values `top_count` and `bottom_count` notate the number of pixels from top and bottom of the data array (in value) that should be cut. The pixels masked are independent on the previous masks applied.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the filter will be calculated from.
- **`top_count`** (*`int`*) – The number of pixels from the top (highest value) of the array that is to be masked.
- **`bottom_count`** (*`int`*) – The number of pixels from the bottom (lowest value) of the array that is to be masked.

Returns **final_filter** – The filter as computed by this function.

Return type ndarray

`ifa_smeargle.masking.filters.`**`filter_sigma_value`**(*data_array*, *sigma_multiple*, *sigma_iterations=1*)

This applies a mask on pixels outside a given multiple of a sigma value.

This function masks values if they are outsize of a sigma range from the mean. The mean and sigma values are automatically calculated from the array provided.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.
- **`sigma_multiple`** (*`float or array-like`*) – The multiple of sigma which will be applied. Unequal bottom-top bounds may be set as a list-like input. The first element is the bottom bound; the last element is the top bound.
- **`iterations`** (*`int`*) – The number of iterations this filler will run through to develop the proper filter.

Returns **final_filter** – The filter as computed by this function.

Return type ndarray

ifa_smeargle.masking.geometric module

These are masks that are applied in a purely geometric method.

`ifa_smeargle.masking.geometric.`**`mask_columns`**(*data_array*, *column_list*)

This applies a column mask on the data array provided its locations.

The column mask takes a list of column numbers (0-indexed x-axis values). All pixels within these columns are then masked.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.
- **`column_list`** (*`list or ndarray`*) – The list of column x-axis values that will be masked. Should be 0-indexed.

Returns **final_mask** – A boolean array for pixels that are masked (True) or are valid (False).

Return type ndarray -> dictionary

`ifa_smeargle.masking.geometric.`**`mask_everything`**(*data_array*)

This applies a blanket blank (all pixels are valid) mask on the data array.

As the name says, this applies a mask. . . to. . . well. . . everything. As such, all that is returned is a full mask.

Parameters **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.

Returns **final_mask** – A boolean array for pixels that are masked (True) or are valid (False).

Return type ndarray

`ifa_smeargle.masking.geometric.`**`mask_nothing`**(*data_array*)

This applies a blanket blank (all pixels are valid) mask on the data array.

As the name says, this applies a mask…to…well…nothing. As such, all that is returned is a blank mask.

Parameters **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.

Returns **final_mask** – A boolean array for pixels that are masked (True) or are valid (False).

Return type ndarray

`ifa_smeargle.masking.geometric.`**`mask_rectangle`**(*data_array*, *column_range*, *row_range*)

This mask function applies rectangular masks to the data array.

The rectangles defined by subsequent xy-ranges (0-indexed) are masked. The rectangle bounds provided are also masked as the rectangle is inclusive of said bounds.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.
- **`column_range`** (*`list or ndarray`*) – The range of 0-indexed columns to be masked.
- **`row_range`** (*`list or ndarray`*) – The range of 0-indexed row to be masked.

Returns **final_mask** – A boolean array for pixels that are masked (True) or are valid (False).

Return type ndarray -> dictionary

`ifa_smeargle.masking.geometric.`**`mask_rows`**(*data_array*, *row_list*)

This applies a row mask on the data array provided its locations.

The row mask takes a list of column numbers (0-indexed x-axis values). All pixels within these rows are then masked.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.
- **`row_list`** (*`list or ndarray`*) – The list of row y-axis values that will be masked. Should be 0-indexed.

Returns **final_mask** – A boolean array for pixels that are masked (True) or are valid (False).

Return type ndarray

`ifa_smeargle.masking.geometric.`**`mask_single_pixels`**(*data_array*, *column_indexes*, *row_indexes*)

This applies a single mask on a single pixel(s)

As the name implies, this function masks a single pixel value or a list of single pixel pairs.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.
- **`column_indexes`** (*`list or ndarray`*) – The successive 0-indexed list of column indexes that specify the pixel to be masked.
- **`row_indexes`** (*`list or ndarray`*) – The successive 0-indexed list of row indexes that specify the pixel to be masked.

Returns **final_mask** – A boolean array for pixels that are masked (True) or are valid (False).

Return type ndarray -> dictionary

`ifa_smeargle.masking.geometric.`**`mask_subarray`**(*data_array*, *column_range*, *row_range*)

This applies a mask on the entire array except for a single sub-array rectangle.

This function subsets a sub-array of the data array from a mask. Only one sub-array can be defined using this function. The bounds of the sub-array is inclusively defined by the x-ranges and y-ranges.

If you want to mask a rectangular section of your array, use *mask_rectangle*.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the mask will be calculated from.
- **`column_range`** (*`list or ndarray`*) – The inclusive column bounds of the sub-array.
- **`row_range`** (*`list or ndarray`*) – The inclusive row bounds of the sub-array.

Returns **final_mask** – A boolean array for pixels that are masked (True) or are valid (False).

Return type ndarray -> dictionary

ifa_smeargle.masking.scripting module

This contains the scripts of the masking functions.

`ifa_smeargle.masking.scripting.`**`script_batch_masking`**(*config*)

This script runs all masks in a batch fashion.

If the run flag of a mask in the configuration file is True, it is run according to the parameters set.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_filter_exact_value`**(*config*)

The scripting version of *filter_exact_value*. This function applies the filter to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels filtered for this filter.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_filter_invalid_value`**(*config*)

The scripting version of *filter_invalid_value*. This function applies the filter to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels filtered for this filter.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_filter_maximum_value`**(*config*)

The scripting version of *filter_maximum_value*. This function applies the filter to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels filtered for this filter.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_filter_minimum_value`**`(`*config*`)`

The scripting version of *filter_minimum_value*. This function applies the filter to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels filtered for this filter.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_filter_percent_truncation`**`(`*config*`)`

The scripting version of *filter_percent_truncation*. This function applies the filter to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels filtered for this filter.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_filter_pixel_truncation`**`(`*config*`)`

The scripting version of *filter_pixel_truncation*. This function applies the filter to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels filtered for this filter.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_filter_sigma_value`**`(`*config*`)`

The scripting version of *filter_sigma_value*. This function applies the filter to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels filtered for this filter.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_mask_columns`**`(`*config*`)`

The scripting version of *mask_columns*. This function applies the mask to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels masked for this mask.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_mask_everything`**`(`*config*`)`

The scripting version of *mask_everything*. This function applies the mask to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels masked for this mask.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_mask_nothing`**`(`*config*`)`

The scripting version of *mask_nothing*. This function applies the mask to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels masked for this mask.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_mask_rectangle`**(*config*)

The scripting version of *mask_rectangle*. This function applies the mask to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels masked for this mask.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_mask_rows`**(*config*)

The scripting version of *mask_rows*. This function applies the mask to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels masked for this mask.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_mask_single_pixels`**(*config*)

The scripting version of *mask_single_pixels*. This function applies the mask to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels masked for this mask.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_mask_subarray`**(*config*)

The scripting version of *mask_subarray*. This function applies the mask to the entire directory (or single file). It also adds the tags to the header file of each fits file indicating the number of pixels masked for this mask.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_synthesize_filters`**(*config*)

The scripting version of *synthesize_masks*. This function applies the filter to the entire directory (or single file). It combines the filter files from the data directory.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.masking.scripting.`**`script_synthesize_masks`**(*config*)

The scripting version of *synthesize_masks*. This function applies the mask to the entire directory (or single file). It combines the mask files from the data directory.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

Module contents

This covers all of the masking programs, establishing the filters and how they operate. In general, they are divided by how they are applied.

1.1.1.4 ifa_smeargle.plotting package

Submodules

ifa_smeargle.plotting.base_functions module

These are the base functions that are common to the entire plotting functionality.

`ifa_smeargle.plotting.base_functions.`**`create_directory_plot_files`**(*data_directory*, *plotting_function*, *figure_arguments*, *plot_arguments*, *matplotlib_arguments*)

This function is the common function to create plots for all analysis fits files.

Parameters

- **`data_directory`** (*`string`*) – The data directory by which all of data for the analysis is contained within.
- **`plotting_function`** (*`function`*) – This is the plotting function that will be applied to all analysis fits files.
- **`figure_arguments`** (*`dictionary`*) – The plotting arguments that will be sent to the function that creates the function.s
- **`plot_arguments`** (*`dictionary`*) – Custom arguments that shall be given to the plotting function. For arguments that should be sent to the matplotlib function, use *matplotlib_arguments*.
- **`matplotlib_arguments`** (*`dictionary`*) – Custom arguments that shall be given to the matplotlib functions. For arguments that should be sent to the plotting function, use *plot_arguments*.

Returns

Return type None

`ifa_smeargle.plotting.base_functions.`**`write_plot_file`**(*file_name*, *figure*, *title=None*, *close_figure=True*)

This function just saves a figure to a file to a name provided.

Because of some oddities with Matplotlib, saving a file within a function and exiting said function (or overwriting in a loop its variable) with a new figure causes two figures to be saved in parallel. This is very memory intensive, so closing a saved figure is ideal. This function does this by default.

See https://stackoverflow.com/a/8862575 and https://cutt.ly/gyK4HlE for more information.

Parameters

- **`file_name`** (*`string`*) – This is the file string name for the figure to be saved. It should already have the appropriate extension. If not, it defaults to the configuration default.

- **figure** (*Matplotlib Figure*) – This is the figure to be saved to a file.
- **title** (*string (optional)*) – This is the title for the figure plot. Although it can be added from here, it is not advised; the original function creating the plot should do it instead.
- **close_figure** (*boolean (optional)*) – This specifies if the figure should be closed. Given that this function is built for that, this should not be changed.

Returns

Return type Nothing

Note: The file type checking logic is not the smartest implementation.

ifa_smeargle.plotting.heatmap module

This creates a plot of a heat-map from analysis fits files.

`ifa_smeargle.plotting.heatmap.`**`plot_heatmap`**(*data_array*, *data_header=None*, *data_mask=None*, *figure_axes=None*, *matplotlib_arguments=None*, ***kwargs*)

A function to create a heat-map image of the data array provided.

This function replicates the image plotting functionality of Tino Well's program. (Found here: https://github.com/tinowells/ifa). More specifically, the plot assigns a color according to the value a pixel has, and plots it corresponding to its location in the provided data array.

Parameters

- **data_array** (*ndarray*) – This is the provided array that is to be plotted. Dimensions matter!
- **data_header** (*Astropy Header (optional)*) – This is the data header of the fits file. If it is not provided, plotting parameters which use it will not be plot. An error may also be raised.
- **data_mask** (*ndarray (optional)*) – The mask that should be applied to the *data_array*.
- **figure_axes** (*Matplotlib Axes (optional)*) – This is a predefined axes variable that the user may desire to have the heat-map plot to. This defaults to either making new ones, or using the currently defined axes. Note: This is not deep-copied!
- **matplotlib_arguments** (*dictionary (optional)*) – These are options the user may use to pass customization parameters into the heat-map plot or the color-bar functionalities. See `imshow()` and `colorbar()`.

Returns **heatmap_plot_axes** – This is the heatmap plot made on the (provided, borrowed, or generated) plotting axes.

Return type Matplotlib Axes

`ifa_smeargle.plotting.heatmap.`**`script_plot_heatmap`**(*config*)

The scripting version of *plot_heatmap*. This function automatically creates heat-map plots for each and every analysis data file.

Parameters **config** (*ConfigObj*) – The configuration object that is to be used for this function.

Returns

Return type None

ifa_smeargle.plotting.histogram module

This is the function for plotting the histogram and Gaussian fits.

`ifa_smeargle.plotting.histogram.`**`plot_gaussian_histogram`**(*data_array*, *data_header=None*, *data_mask=None*, *figure_axes=None*, *matplotlib_arguments=None*, *fit_gaussian=True*, ***kwargs*)

A function to create and plot histogram plots for better analysis of a given array.

This function replicates the histogram plotting functionality of Tino Well's program. (Found here: https://github.com/tinowells/ifa). More specifically, it attempts to plot histograms of pixel data; then, the program attempts to fit a Gaussian function.

Parameters

- **`data_array`** (*`ndarray`*) – This is the provided array that is to be plotted. Dimensions matter!
- **`data_header`** (*`Astropy Header (optional)`*) – This is the data header of the fits file. If it is not provided, plotting parameters which use it will not be plot. An error may also be raised.
- **`data_mask`** (*`ndarray (optional)`*) – The mask that should be applied to the *data_array*.
- **`figure_axes`** (*`Matplotlib Axes (optional)`*) – This is a predefined axes variable that the user may desire to have the heat-map plot to. This defaults to either making new ones, or using the currently defined axes. Note: This is not deep-copied!
- **`matplotlib_arguments`** (*`dictionary (optional)`*) – These are options the user may use to pass customization parameters into the histogram plot or the Gaussian function plot. See `bar()` and `plot()`.

Returns **heatmap_plot_axes** – This is the heatmap plot made on the (provided, borrowed, or generated) plotting axes.

Return type Matplotlib Axes

`ifa_smeargle.plotting.histogram.`**`script_plot_gaussian_histogram`**(*config*)

The scripting version of *plot_gaussian_histogram*. This function automatically creates heat-map plots for each and every analysis data file.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

Module contents

1.1.1.5 ifa_smeargle.reformat package

Submodules

ifa_smeargle.reformat.base_functions module

These are functions common to the reformatting module.

ifa_smeargle.reformat.base_functions.**format_slice_appending_name**(*reference_frame*, *averaging_frame*)

The formatting for the string alignment for slices.

ifa_smeargle.reformat.collapse module

This contains the functions required for making effective collapsed frames; usually by extracting and subtracting subsets of frames and normalizing over some time.

ifa_smeargle.reformat.collapse.**collapse_by_average_endpoints**(*data_array*, *start_chunk*, *end_chunk*, *average_method='median'*, *frame_exposure_time=None*)

This function reads a fits file and computes its end section values.

This function reads in a fits file of 3 dimensions, averaging some top chunk and bottom chunk of their "temporal" axis.

If there is no temporal axis, this program raises an error.

Parameters

- **data_array** (*ndarray*) – The data array that the average will be taken from.
- **start_chunk** (*array-like*) – The exact range of frames from the beginning that will be averaged as per the averaging method.
- **end_chunk** (*array-like*) – The exact range of frames from the end that will be averaged as per the averaging method.
- **average_method** (*string (optional)*) – The current available methods for determining the file size that is proper. Defaults to median:
  - **'mean'** This takes the mean along the frames of each chunk
  - **'median'** This takes the median along the frames of each chunk. Even sets use the mean of the middle most two values.
- **frame_exposure_time** (*float*) – The duration, per frame (in seconds), of each exposure. This is really not used in this function, but, it is added for uniformity with the other functions.

Returns **final_data** – The final data array of the median-ed frames as desired.

Return type ndarray

`ifa_smeargle.reformat.collapse.`**`collapse_by_average_endpoints_per_kilosecond`**`(`*data_array, start_chunk, end_chunk, frame_exposure_time av- er- age_method='median*

This function reads a fits file and computes its end section values, normalizing per kilosecond.

This function reads in a fits file of 3 dimensions, averaging some top chunk and bottom chunk of their "temporal" axis, normalizing and dividing over a timespan. The time is measured in kiloseconds. This is basically a wrapper function around the per second version.

If there is no temporal axis, this program raises an error.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the average will be taken from.
- **`start_chunk`** (*`array-like`*) – The exact range of frames from the beginning that will be averaged as per the averaging method.
- **`end_chunk`** (*`array-like`*) – The exact range of frames from the end that will be averaged as per the averaging method.
- **`frame_exposure_time`** (*`float`*) – The duration, per frame (in seconds), of each exposure. This is really not used in this function, but, it is added for uniformity with the other functions.
- **`average_method`** (*`string (optional)`*) – The current available methods for determining the file size that is proper. Defaults to median.
  - **'mean'** This takes the mean along the frames of each chunk
  - **'median'** This takes the median along the frames of each chunk. Even sets use the mean of the middle most two values.

`ifa_smeargle.reformat.collapse.`**`collapse_by_average_endpoints_per_second`**`(`*data_array, start_chunk, end_chunk, frame_exposure_time, av- er- age_method='median'*`)`

This function reads a fits file and computes its end section values, normalizing per second.

This function reads in a fits file of 3 dimensions, averaging some top chunk and bottom chunk of their "temporal" axis, normalizing and dividing over a timespan. The time is measured in seconds.

If there is no temporal axis, this program raises an error.

Parameters

- **`data_array`** (*`ndarray`*) – The data array that the average will be taken from.
- **`start_chunk`** (*`array-like`*) – The exact range of frames from the beginning that will be averaged as per the averaging method.
- **`end_chunk`** (*`array-like`*) – The exact range of frames from the end that will be averaged as per the averaging method.

- **`frame_exposure_time`** (*`float`*) – The duration, per frame (in seconds), of each exposure. This is really not used in this function, but, it is added for uniformity with the other functions.
- **`average_method`** (*`string (optional)`*) – The current available methods for determining the file size that is proper. Defaults to median.
  - **'mean'** This takes the mean along the frames of each chunk
  - **'median'** This takes the median along the frames of each chunk. Even sets use the mean of the middle most two values.

`ifa_smeargle.reformat.collapse.`**`script_collapse_by_average_endpoints`**(*config*)

The scripting version of *collapse_by_average_endpoints*. This function applies the rename to the entire directory. It also adds the tags to the header file of each fits.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.reformat.collapse.`**`script_collapse_by_average_endpoints_per_kilosecond`**(*config*)

The scripting version of *collapse_by_average_endpoints_per_kilosecond*. This function applies the rename to the entire directory. It also adds the tags to the header file of each fits.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.reformat.collapse.`**`script_collapse_by_average_endpoints_per_second`**(*config*)

The scripting version of *collapse_by_average_endpoints_per_second*. This function applies the rename to the entire directory. It also adds the tags to the header file of each fits.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

ifa_smeargle.reformat.renaming module

This file contains methods used to determine proper naming conventions and to reformat the fits file names from raw output (often just timestamps) to something more useful and accurate to the data.

`ifa_smeargle.reformat.renaming.`**`rename_detector`**(*data_directory*, *detector_name*)

Creates file tags according to the detector qualifier name.

Parameters

- **`data_directory`** (*`string`*) – This is the directory that contain all of the data files to be renamed.
- **`detector_name`** (*`string`*) – The detector name that should be applied.

Returns

- **detector_string_list** (*list*) – This is the list of the state on if the object is garbage or not.
- **detector_raw_list** (*list*) – This is the raw, unformatted values that was formatted into the string version.

`ifa_smeargle.reformat.renaming.`**`rename_garbage`**(*data_directory*, *begin_garbage=0*)

Creates file tags according to their status as garbage. If a file is garbage, it should generally not be used.

Parameters

- **`data_directory`** (*`string`*) – This is the directory that contain all of the data files to be renamed.
- **`begin_garbage`** (*`int (optional)`*) – The number of files, in the beginning, that should not count as data, (i.e. the number of garbage files).

Returns

- **garbage_string_list** (*list*) – This is the list of the state on if the object is garbage or not.
- **garbage_raw_list** (*list*) – This is the raw, unformatted values that was formatted into the string version.

`ifa_smeargle.reformat.renaming.`**`rename_number`**(*data_directory*, *begin_garbage=0*)

Creates file tags according to their number in order.

Some data filename outputs only give timestamps. This function renames said filenames for better processing.

Parameters

- **`data_directory`** (*`string`*) – This is the directory that contain all of the data files to be renamed.
- **`begin_garbage`** (*`int (optional)`*) – The number of files, in the beginning, that should not count as data.

Returns

- **number_string_list** (*list*) – This is the list of the numbered strings applied, given in a parallel ordered form. Does not include prefixes/suffixes.
- **number_raw_list** (*list*) – This is the raw, unformatted values that was formatted into the string version.

`ifa_smeargle.reformat.renaming.`**`rename_set`**(*data_directory*, *set_length*, *begin_garbage=0*)

Creates file tags according to their set number, as determined by the number of files in a set. Sets are assumed to be consecutive.

Some data filename outputs only give timestamps. This function renames said filenames for better processing.

Parameters

- **`data_directory`** (*`string`*) – This is the directory that contain all of the data files to be renamed.
- **`set_length`** (*`int`*) – This is the length of a set.
- **`begin_garbage`** (*`int (optional)`*) – The number of files, in the beginning, that should not count as data.

Returns

- **set_string_list** (*list*) – This is the list of the numbered strings applied, given in a parallel ordered form. Does not include prefixes/suffixes.
- **set_raw_list** (*list*) – This is the raw, unformatted values that was formatted into the string version.

`ifa_smeargle.reformat.renaming.`**`rename_voltage_pattern`**(*data_directory*, *voltage_pattern*, *begin_garbage=0*)

Creates file tags according to their voltage pattern specified.

Some data filename outputs only give timestamps. This function renames said filenames for better processing.

The output files created are according to the voltage pattern that the user specified. This function assumes that the pattern provided is one 'set', where a set contains some amount of fits files. Moreover, this function weakly determines which are up-mid-down ramps.

Parameters

- **`data_directory`** (*`string`*) – This is the directory that contain all of the data files to be renamed.
- **`voltage_pattern`** (*`array_like`*) – These are voltage values, assuming that the first element is the first voltage element to be used, proceeding from there in order.
- **`begin_garbage`** (*`int (optional)`*) – The number of files, in the beginning, that should not count as data.

Returns

- **voltage_string_list** (*list*) – This is the list of the voltage strings applied, given in a parallel ordered form. Does not include prefixes/suffixes.
- **voltage_raw_list** (*dictionary*) – This is the raw, unformatted values that was formatted into the string version. The voltage values and trends are separated (as *value* and *trend*).

`ifa_smeargle.reformat.renaming.`**`script_rename_detector`**(*config*)

The scripting version of *rename_detector*. This function applies the rename to the entire directory. It also adds the tags to the header file of each fits.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.reformat.renaming.`**`script_rename_garbage`**(*config*)

The scripting version of *rename_garbage*. This function applies the rename to the entire directory. It also adds the tags to the header file of each fits.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.reformat.renaming.`**`script_rename_number`**(*config*)

The scripting version of *rename_number*. This function applies the rename to the entire directory. It also adds the tags to the header file of each fits.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.reformat.renaming.`**`script_rename_set`**(*config*)

The scripting version of *rename_set*. This function applies the rename to the entire directory. It also adds the tags to the header file of each fits.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.reformat.renaming.`**`script_rename_voltage_pattern`**(*config*)

The scripting version of *rename_voltage_pattern*. This function applies the rename to the entire directory. It also adds the tags to the header file of each fits.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

ifa_smeargle.reformat.sanitization module

`ifa_smeargle.reformat.sanitization.`**`sanitize_file_size`**(*data_directory, method='largest', delete=False, exact_size=None*)

A function to clean the data directory of any file abnormalities stemming from incomplete or over-complete files.

It is often to have too large files or too small files in a data set. These files hold no consequence for removal and would only contaminate or crash the pipeline if they went through. The good files are kept, and the bad files are deleted. This method only affects `.fits` files.

Parameters

- **`data_directory`** (*`string`*) – The directory of the data that is to be sanitized with the non-conforming files deleted.
- **`method`** (*`string (optional)`*) – The current available methods for determining the file size that is proper.
  - **'largest'** The largest `.fits` file is considered to be the right file size (default).
  - **'smallest'** The smallest `.fits` file is considered to be the right file size.
  - **'exact'** The `.fits` file that is exactly the size specified.
- **`delete`** (*`boolean (optional)`*) – If True, then all bad files found by this method are deleted.
- **`exact_size`** (*`int (optional)`*) – The exact size a proper file size should be (unit is in bytes). Only applied if the method used is *exact*

Returns **bad_file_list** – The list of all of the file names/paths that are flagged for sanitization.

Return type list

`ifa_smeargle.reformat.sanitization.`**`script_sanitize_file_size`**(*config*)

The scripting version of *sanitize_file_size*. This function sanitizes the documents as prescribed by the configuration and the inner function.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

Module contents

This module handles the reformatting of data into a more standard and manageable form for functions within this library.

1.1.1.6 ifa_smeargle.testing package

Submodules

ifa_smeargle.testing.base_functions module

This contains a lot of functions that are common across all of the testing functions and modules.

`ifa_smeargle.testing.base_functions.`**`create_prime_test_array`**`(`*shape*, *index=0*`)`
This creates a test array for the purposes of creating masks and to apply them and test them. This ensures a uniform test array for all mask tests.

Parameters

- **shape** (*tuple*) – The shape of the data array.
- **index** (*integer*) – The index that the prime numbers should start from.

Returns **prime_test_array** – The array filled with prime numbers.

Return type ndarray

ifa_smeargle.testing.test_global module

These are for tests that are global, they apply to both development and numerical accuracy; or they apply to neither.

`ifa_smeargle.testing.test_global.`**`test_pass`**`()`
This is a test that should always pass.

ifa_smeargle.testing.test_numerical_core_mathematics module

This module is made to test the mathematical functions of the core.mathematics section.

`ifa_smeargle.testing.test_numerical_core_mathematics.`**`test_ifas_large_integer_array_product`**
This tests the multiplication of large integers.

`ifa_smeargle.testing.test_numerical_core_mathematics.`**`test_ifas_masked_mean`**`()`
This tests the mean computation with masked arrays.

`ifa_smeargle.testing.test_numerical_core_mathematics.`**`test_ifas_masked_median`**`()`
This tests the median computation with masked arrays.

`ifa_smeargle.testing.test_numerical_core_mathematics.`**`test_ifas_masked_std`**`()`
This tests the standard deviation computation with masked arrays.

`ifa_smeargle.testing.test_numerical_core_mathematics.`**`test_ifas_robust_mean`**`()`
This tests the computation of means using a more robust approach.

`ifa_smeargle.testing.test_numerical_core_mathematics.`**`test_ifas_robust_std`**`()`
This tests the computation of means using a more robust approach.

ifa_smeargle.testing.test_numerical_filters module

This tests the filter functions to ensure that they are appropriately calculating the filters as expected.

These filter tests operate on the principle that the product of single power prime integers is always unique, and by extension, so are their logarithms. Prime number arrays are filtered, multiplied together, and compared against an expected hard-coded result.

`ifa_smeargle.testing.test_numerical_filters.`**`test_filter_exact_value`**`()`
    This tests the filtering of exact values.

`ifa_smeargle.testing.test_numerical_filters.`**`test_filter_invalid_value`**`()`
    This tests the filtering of invalid values.

`ifa_smeargle.testing.test_numerical_filters.`**`test_filter_maximum_value`**`()`
    This tests the filtering of values above a maximum.

`ifa_smeargle.testing.test_numerical_filters.`**`test_filter_minimum_value`**`()`
    This tests the filtering of values below a minimum.

`ifa_smeargle.testing.test_numerical_filters.`**`test_filter_percent_truncation`**`()`
    This tests the filtering of percent truncations.

`ifa_smeargle.testing.test_numerical_filters.`**`test_filter_pixel_truncation`**`()`
    This tests the filtering of pixel boundaries.

`ifa_smeargle.testing.test_numerical_filters.`**`test_filter_sigma_value`**`()`
    This tests the filtering of sigma boundaries.

ifa_smeargle.testing.test_numerical_masking module

This tests the masking functions to ensure that they are appropriately calculating the masks as expected.

These mask tests operate on the principle that the product of single power prime integers is always unique, and by extension, so are their logarithms. Prime number arrays are masked, multiplied together, and compared against an expected hard-coded result.

`ifa_smeargle.testing.test_numerical_masking.`**`test_mask_columns`**`()`
    This tests the masking of every other column to ensure none are masked.

`ifa_smeargle.testing.test_numerical_masking.`**`test_mask_everything`**`()`
    This tests the masking of all pixels to ensure none are missed.

`ifa_smeargle.testing.test_numerical_masking.`**`test_mask_nothing`**`()`
    This tests the masking of all pixels to ensure none are masked.

`ifa_smeargle.testing.test_numerical_masking.`**`test_mask_rectangle`**`()`
    This tests the masking of a rectangle, checking for inclusive bounds as documented.

`ifa_smeargle.testing.test_numerical_masking.`**`test_mask_rows`**`()`
    This tests the masking of every other row to ensure none are masked.

`ifa_smeargle.testing.test_numerical_masking.`**`test_mask_single_pixels`**`()`
    This tests the masking of single pixels.

`ifa_smeargle.testing.test_numerical_masking.`**`test_mask_subarray`**`()`
    This tests the masking of the sub-arrays, checking for inclusive bounds as documented.

Module contents

All testing functions should be available through this module. There does not seem a reason to break them down further into submodules within the testing submodule.

1.1.1.7 ifa_smeargle.tutorial package

Submodules

ifa_smeargle.tutorial.copying module

This module deals with the copying of files from one location to another. This is usually used for building the tutorial.

ifa_smeargle.tutorial.generation module

These functions generates all of the required tutorial data, configurations, and other needed objects to begin or run through the tutorial.

`ifa_smeargle.tutorial.generation.`**`tutorial_generate_fits_file`**(*generation_mode*, *data_shape*, *fill_value=None*, *seed=None*, *range=None*)

This function generates a fits file that emulate the real data in some regards. The different modes can create different data representations.

Parameters

- **`generation_mode`** (*`string`*) – The generation mode that the data will adhere to.
  - **'fill'** The fits data values are some constant number.
  - **'increment'** The files data values are created by counting up individually. Order is based on C indexing.
  - **'pseudorandom'** The fits data values are generated using a set seed and a pseudorandom number generator.
  - **'random'** The fits data values are generated randomly as determined by undetermined-seed random number generator.
- **`data_shape`** (*`tuple`*) – The shape of the data that will be created.
- **`fill_value`** (*`float (optional)`*) – The value that fills the data array. Only used if the generation mode is *fill*.
- **`seed`** (*`int (optional)`*) – The seed value for the pseudorandom number generator. Only used if the generation mode is *pseudorandom*.
- **`range`** (*`array-like (optional)`*) – The range for the pseudorandom and the random number generator. It takes only the highest and lowest numbers in the tuple as the appropriate range. The range is [min, max). Defaults to [0.0, 1.0).

Returns **hdu_object** – An object representation of the fits file. The header is also generated.

Return type Astropy HDU object

ifa_smeargle.tutorial.scripting module

This is where all of the scripts that create different tutorials for each different type of detector lies.

`ifa_smeargle.tutorial.scripting.`**`script_generate_saphira_tutorial`**`(`*config*`)`

This function generates a tutorial directory for new users to experiment with.

This script generates a tutorial for the SAPHIRA detectors and what kind of data they would spit out.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

Module contents

This is the section of the code-base that is more or less geared towards the tutorial for the user.

1.1.2 Submodules

1.1.2.1 ifa_smeargle.runtime module

This module contains runtime-variables that are important for the function of this module.

`ifa_smeargle.runtime.`**`extract_runtime_configuration`**

This function extracts the configuration based solely on the key of the configuration value.

The conversion of 'True'->bool(True) and 'False'->bool(False) is implicitly assumed. Letter case doesn't matter.

Parameters

- **`config_key`** (*`string`*) – The tag of the configuration parameter that is being sought.
- **`type_convert`** (*`boolean`*) – A flag to determine if type conversion is desired. If False, the raw value is returned instead.

Returns **config_value** – The configuration object that is present within the file.

Return type object

`ifa_smeargle.runtime.`**`get_configuration_files`**`()`

This function obtains all of the configuration files, returning a dictionary of them for each of their paths.

The qualifier to be a configuration file is just to have *.ini* extension to the file name.

Parameters **`None`** –

Returns **config_files** – The configuration files that have been found in the module and its submodules.

Return type dictionary

`ifa_smeargle.runtime.`**`get_filter_functions`**`()`

This function obtains all possible *filter* based functions, returning a dictionary of them.

The qualifier to be a script function is just to have *filter_* as the prefix to the function.

Parameters **`None`** –

Returns **filters** – The filters that have been found in the module and its sub-modules which have the filtering prefix.

Return type dictionary

`ifa_smeargle.runtime.`**`get_mask_functions`**`()`

This function obtains all possible *mask* based functions, returning a dictionary of them.

The qualifier to be a masking function is just to have *mask_* as the prefix to the function.

Parameters **`None`** –

Returns **masks** – The masking functions that have been found in the module and its sub-modules which have the scripting prefix.

Return type dictionary

`ifa_smeargle.runtime.`**`get_module_directory`**`()`

This function returns the path of this module.

Parameters **`None`** –

Returns **module_directory** – The directory of this module.

Return type string

`ifa_smeargle.runtime.`**`get_script_functions`**`()`

This function obtains all possible *script* based functions, returning a dictionary of them.

The qualifier to be a script function is just to have *script_* as the prefix to the function.

Parameters **`None`** –

Returns **scripts** – The scripts that have been found in the module and its sub-modules which have the scripting prefix.

Return type dictionary

`ifa_smeargle.runtime.`**`get_specification_files`**`()`

This function obtains all of the configuration specification files, returning a dictionary of them for each of their paths.

The qualifier to be a specification file is just to have *.spec* extension to the file name.

Parameters **`None`** –

Returns **spec_files** – The specification files that have been found in the module and its sub-modules.

Return type dictionary

1.1.2.2 ifa_smeargle.special module

This is a special file. It contains many script functions which do not fit the normal work-flow of this package.

`ifa_smeargle.special.`**`script_special_create_configuration_file`**`(`*config*`)`

This function copies a given configuration file into the current working directory. The type of configuration is given by the configuration object.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

`ifa_smeargle.special.`**`script_special_list_scripts`**`(`*config*`)`

This lists all of the scripts in alphabetical order in two columns.

Parameters **`config`** (*`ConfigObj`*) – The configuration object that is to be used for this function.

Returns

Return type None

1.1.3 Module contents

This is the general import space of the library.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i

ifa_smeargle, 43
ifa_smeargle.analysis, 6
ifa_smeargle.analysis.base_functions, 3
ifa_smeargle.analysis.heatmap, 5
ifa_smeargle.analysis.histogram, 5
ifa_smeargle.core, 20
ifa_smeargle.core.configuration, 6
ifa_smeargle.core.error, 7
ifa_smeargle.core.io, 12
ifa_smeargle.core.magic, 15
ifa_smeargle.core.mathematics, 15
ifa_smeargle.core.modeling, 18
ifa_smeargle.core.string_formatting, 19
ifa_smeargle.masking, 29
ifa_smeargle.masking.base_functions, 20
ifa_smeargle.masking.filters, 22
ifa_smeargle.masking.geometric, 24
ifa_smeargle.masking.scripting, 26
ifa_smeargle.plotting, 32
ifa_smeargle.plotting.base_functions, 29
ifa_smeargle.plotting.heatmap, 30
ifa_smeargle.plotting.histogram, 31
ifa_smeargle.reformat, 38
ifa_smeargle.reformat.base_functions, 32
ifa_smeargle.reformat.collapse, 32
ifa_smeargle.reformat.renaming, 34
ifa_smeargle.reformat.sanitization, 37
ifa_smeargle.runtime, 41
ifa_smeargle.special, 42
ifa_smeargle.testing, 40
ifa_smeargle.testing.base_functions, 38
ifa_smeargle.testing.test_global, 38
ifa_smeargle.testing.test_numerical_core_mathematics, 38
ifa_smeargle.testing.test_numerical_filters, 39
ifa_smeargle.testing.test_numerical_masking, 39
ifa_smeargle.tutorial, 41
ifa_smeargle.tutorial.copying, 40
ifa_smeargle.tutorial.generation, 40
ifa_smeargle.tutorial.scripting, 41

INDEX

A

analysis_gaussian_histogram() (*in module ifa_smeargle.analysis.histogram*), 5
analysis_heatmap() (*in module ifa_smeargle.analysis.heatmap*), 5
append_astropy_header_card() (*in module ifa_smeargle.core.io*), 12
archive_data_directory() (*in module ifa_smeargle.core.io*), 12
AssumptionError, 7

B

BrokenLogicError, 7

C

collapse_by_average_endpoints() (*in module ifa_smeargle.reformat.collapse*), 32
collapse_by_average_endpoints_per_kilosecond() (*in module ifa_smeargle.reformat.collapse*), 32
collapse_by_average_endpoints_per_second() (*in module ifa_smeargle.reformat.collapse*), 33
combine_pathname() (*in module ifa_smeargle.core.string_formatting*), 19
ConfigurationError, 7
ConfigurationWarning, 7
copy_configuration_file() (*in module ifa_smeargle.core.configuration*), 6
create_directory_analysis_files() (*in module ifa_smeargle.analysis.base_functions*), 3
create_directory_filter_files() (*in module ifa_smeargle.masking.base_functions*), 20
create_directory_mask_file() (*in module ifa_smeargle.masking.base_functions*), 21
create_directory_plot_files() (*in module ifa_smeargle.plotting.base_functions*), 29
create_filter_from_configuration() (*in module ifa_smeargle.analysis.base_functions*), 4
create_filter_from_directory() (*in module ifa_smeargle.analysis.base_functions*), 4
create_prime_test_array() (*in module ifa_smeargle.testing.base_functions*), 38

D

DataError, 8
DataWarning, 8
dearchive_data_directory() (*in module ifa_smeargle.core.io*), 13
DeprecatedError, 8
DeprecatedWarning, 8
DevelopmentError, 8

E

ErrorWarning, 8
ExportingError, 8
ExportingWarning, 8
extract_configuration() (*in module ifa_smeargle.core.configuration*), 6
extract_runtime_configuration (*in module ifa_smeargle.runtime*), 41

F

filter_exact_value() (*in module ifa_smeargle.masking.filters*), 22
filter_invalid_value() (*in module ifa_smeargle.masking.filters*), 22
filter_maximum_value() (*in module ifa_smeargle.masking.filters*), 23
filter_minimum_value() (*in module ifa_smeargle.masking.filters*), 23
filter_percent_truncation() (*in module ifa_smeargle.masking.filters*), 23
filter_pixel_truncation() (*in module ifa_smeargle.masking.filters*), 23
filter_sigma_value() (*in module ifa_smeargle.masking.filters*), 24
fit_gaussian_function() (*in module ifa_smeargle.core.modeling*), 18
fit_histogram_gaussian_function() (*in module ifa_smeargle.core.modeling*), 18
format_shutil_archive_extensions() (*in module ifa_smeargle.core.string_formatting*), 19

format_slice_appending_name() (*in module ifa_smeargle.reformat.base_functions*), 32

G

generate_numpy_bin_width_array() (*in module ifa_smeargle.core.mathematics*), 15
generate_prime_numbers() (*in module ifa_smeargle.core.mathematics*), 16
get_analysis_fits_filenames() (*in module ifa_smeargle.analysis.base_functions*), 4
get_configuration_files() (*in module ifa_smeargle.runtime*), 41
get_filter_fits_filenames() (*in module ifa_smeargle.masking.base_functions*), 21
get_filter_functions() (*in module ifa_smeargle.runtime*), 41
get_fits_filenames() (*in module ifa_smeargle.core.io*), 13
get_mask_fits_filenames() (*in module ifa_smeargle.masking.base_functions*), 22
get_mask_functions() (*in module ifa_smeargle.runtime*), 42
get_module_directory() (*in module ifa_smeargle.runtime*), 42
get_script_functions() (*in module ifa_smeargle.runtime*), 42
get_specification_files() (*in module ifa_smeargle.runtime*), 42
get_subdirectories() (*in module ifa_smeargle.core.io*), 13

I

ifa_smeargle (*module*), 43
ifa_smeargle.analysis (*module*), 6
ifa_smeargle.analysis.base_functions (*module*), 3
ifa_smeargle.analysis.heatmap (*module*), 5
ifa_smeargle.analysis.histogram (*module*), 5
ifa_smeargle.core (*module*), 20
ifa_smeargle.core.configuration (*module*), 6
ifa_smeargle.core.error (*module*), 7
ifa_smeargle.core.io (*module*), 12
ifa_smeargle.core.magic (*module*), 15
ifa_smeargle.core.mathematics (*module*), 15
ifa_smeargle.core.modeling (*module*), 18
ifa_smeargle.core.string_formatting (*module*), 19
ifa_smeargle.masking (*module*), 29
ifa_smeargle.masking.base_functions (*module*), 20
ifa_smeargle.masking.filters (*module*), 22
ifa_smeargle.masking.geometric (*module*), 24
ifa_smeargle.masking.scripting (*module*), 26
ifa_smeargle.plotting (*module*), 32
ifa_smeargle.plotting.base_functions (*module*), 29
ifa_smeargle.plotting.heatmap (*module*), 30
ifa_smeargle.plotting.histogram (*module*), 31
ifa_smeargle.reformat (*module*), 38
ifa_smeargle.reformat.base_functions (*module*), 32
ifa_smeargle.reformat.collapse (*module*), 32
ifa_smeargle.reformat.renaming (*module*), 34
ifa_smeargle.reformat.sanitization (*module*), 37
ifa_smeargle.runtime (*module*), 41
ifa_smeargle.special (*module*), 42
ifa_smeargle.testing (*module*), 40
ifa_smeargle.testing.base_functions (*module*), 38
ifa_smeargle.testing.test_global (*module*), 38
ifa_smeargle.testing.test_numerical_core_mathematic (*module*), 38
ifa_smeargle.testing.test_numerical_filters (*module*), 39
ifa_smeargle.testing.test_numerical_masking (*module*), 39
ifa_smeargle.tutorial (*module*), 41
ifa_smeargle.tutorial.copying (*module*), 40
ifa_smeargle.tutorial.generation (*module*), 40
ifa_smeargle.tutorial.scripting (*module*), 41
ifas_absolute_silence() (*in module ifa_smeargle.core.error*), 10
Ifas_BaseException, 8
ifas_debug() (*in module ifa_smeargle.core.error*), 10
ifas_debug_block() (*in module ifa_smeargle.core.error*), 11
ifas_disable_debug() (*in module ifa_smeargle.core.error*), 11
ifas_enable_debug() (*in module ifa_smeargle.core.error*), 11
ifas_error() (*in module ifa_smeargle.core.error*), 11
Ifas_Exception, 8
ifas_gaussian_function() (*in module ifa_smeargle.core.mathematics*), 16

`ifas_info()` (*in module ifa_smeargle.core.error*), 11
`ifas_large_integer_array_product()` (*in module ifa_smeargle.core.mathematics*), 16
`ifas_log_warning()` (*in module ifa_smeargle.core.error*), 11
`ifas_masked_mean()` (*in module ifa_smeargle.core.mathematics*), 17
`ifas_masked_median()` (*in module ifa_smeargle.core.mathematics*), 17
`ifas_masked_std()` (*in module ifa_smeargle.core.mathematics*), 17
`ifas_robust_mean()` (*in module ifa_smeargle.core.mathematics*), 17
`ifas_robust_std()` (*in module ifa_smeargle.core.mathematics*), 17
`ifas_silence_all_warnings()` (*in module ifa_smeargle.core.error*), 11
`ifas_silence_ifas_warnings()` (*in module ifa_smeargle.core.error*), 11
`ifas_silence_info_message()` (*in module ifa_smeargle.core.error*), 11
`ifas_silence_nonifas_warnings()` (*in module ifa_smeargle.core.error*), 11
`ifas_silence_specific_warnings()` (*in module ifa_smeargle.core.error*), 12
`Ifas_Warning`, 8
`ifas_warning()` (*in module ifa_smeargle.core.error*), 12
`IllogicalProsedureError`, 8
`ImportingError`, 9
`ImportingWarning`, 9
`ImprecisionError`, 9
`ImprecisionWarning`, 9
`IncompleteError`, 9
`InputError`, 9
`InputWarning`, 9

M

`magic_inital_gaussian_parameters()` (*in module ifa_smeargle.core.magic*), 15
`MagicError`, 9
`MagicWarning`, 9
`mask_columns()` (*in module ifa_smeargle.masking.geometric*), 24
`mask_everything()` (*in module ifa_smeargle.masking.geometric*), 24
`mask_nothing()` (*in module ifa_smeargle.masking.geometric*), 25
`mask_rectangle()` (*in module ifa_smeargle.masking.geometric*), 25
`mask_rows()` (*in module ifa_smeargle.masking.geometric*), 25
`mask_single_pixels()` (*in module ifa_smeargle.masking.geometric*), 25
`mask_subarray()` (*in module ifa_smeargle.masking.geometric*), 26
`MaskingError`, 9
`MaskingWarning`, 9
`MemoryWarning`, 10
`ModelingError`, 10

O

`OverwriteWarning`, 10

P

`plot_gaussian_histogram()` (*in module ifa_smeargle.plotting.histogram*), 31
`plot_heatmap()` (*in module ifa_smeargle.plotting.heatmap*), 30
`purge_substrings()` (*in module ifa_smeargle.core.string_formatting*), 19

R

`random_string()` (*in module ifa_smeargle.core.string_formatting*), 19
`read_configuration_file()` (*in module ifa_smeargle.core.configuration*), 7
`read_fits_file()` (*in module ifa_smeargle.core.io*), 13
`ReductionWarning`, 10
`remove_prefix()` (*in module ifa_smeargle.core.string_formatting*), 20
`rename_by_parallel_append()` (*in module ifa_smeargle.core.io*), 14
`rename_by_parallel_replace()` (*in module ifa_smeargle.core.io*), 14
`rename_detector()` (*in module ifa_smeargle.reformat.renaming*), 34
`rename_garbage()` (*in module ifa_smeargle.reformat.renaming*), 34
`rename_number()` (*in module ifa_smeargle.reformat.renaming*), 35
`rename_set()` (*in module ifa_smeargle.reformat.renaming*), 35
`rename_voltage_pattern()` (*in module ifa_smeargle.reformat.renaming*), 35

S

`sanitize_file_size()` (*in module ifa_smeargle.reformat.sanitization*), 37
`script_analysis_gaussian_histogram()` (*in module ifa_smeargle.analysis.histogram*), 6
`script_analysis_heatmap()` (*in module ifa_smeargle.analysis.heatmap*), 5
`script_batch_analysis()` (*in module ifa_smeargle.analysis.base_functions*), 5
`script_batch_masking()` (*in module ifa_smeargle.masking.scripting*), 26

script_collapse_by_average_endpoints() (*in module ifa_smeargle.reformat.collapse*), 34
script_collapse_by_average_endpoints_per_kilosecond() (*in module ifa_smeargle.reformat.collapse*), 34
script_collapse_by_average_endpoints_per_second() (*in module ifa_smeargle.reformat.collapse*), 34
script_filter_exact_value() (*in module ifa_smeargle.masking.scripting*), 26
script_filter_invalid_value() (*in module ifa_smeargle.masking.scripting*), 26
script_filter_maximum_value() (*in module ifa_smeargle.masking.scripting*), 26
script_filter_minimum_value() (*in module ifa_smeargle.masking.scripting*), 26
script_filter_percent_truncation() (*in module ifa_smeargle.masking.scripting*), 27
script_filter_pixel_truncation() (*in module ifa_smeargle.masking.scripting*), 27
script_filter_sigma_value() (*in module ifa_smeargle.masking.scripting*), 27
script_generate_saphira_tutorial() (*in module ifa_smeargle.tutorial.scripting*), 41
script_mask_columns() (*in module ifa_smeargle.masking.scripting*), 27
script_mask_everything() (*in module ifa_smeargle.masking.scripting*), 27
script_mask_nothing() (*in module ifa_smeargle.masking.scripting*), 27
script_mask_rectangle() (*in module ifa_smeargle.masking.scripting*), 28
script_mask_rows() (*in module ifa_smeargle.masking.scripting*), 28
script_mask_single_pixels() (*in module ifa_smeargle.masking.scripting*), 28
script_mask_subarray() (*in module ifa_smeargle.masking.scripting*), 28
script_plot_gaussian_histogram() (*in module ifa_smeargle.plotting.histogram*), 31
script_plot_heatmap() (*in module ifa_smeargle.plotting.heatmap*), 30
script_rename_detector() (*in module ifa_smeargle.reformat.renaming*), 36
script_rename_garbage() (*in module ifa_smeargle.reformat.renaming*), 36
script_rename_number() (*in module ifa_smeargle.reformat.renaming*), 36
script_rename_set() (*in module ifa_smeargle.reformat.renaming*), 36
script_rename_voltage_pattern() (*in module ifa_smeargle.reformat.renaming*), 37
script_sanitize_file_size() (*in module ifa_smeargle.reformat.sanitization*), 37
script_special_create_configuration_file() (*in module ifa_smeargle.special*), 42
script_special_list_scripts() (*in module ifa_smeargle.special*), 42
script_synthesize_filters() (*in module ifa_smeargle.masking.scripting*), 28
script_synthesize_masks() (*in module ifa_smeargle.masking.scripting*), 28
split_pathname() (*in module ifa_smeargle.core.string_formatting*), 20
StorageWarning, 10
synthesize_masks() (*in module ifa_smeargle.masking.base_functions*), 22

T

TerminalError, 10
test_filter_exact_value() (*in module ifa_smeargle.testing.test_numerical_filters*), 39
test_filter_invalid_value() (*in module ifa_smeargle.testing.test_numerical_filters*), 39
test_filter_maximum_value() (*in module ifa_smeargle.testing.test_numerical_filters*), 39
test_filter_minimum_value() (*in module ifa_smeargle.testing.test_numerical_filters*), 39
test_filter_percent_truncation() (*in module ifa_smeargle.testing.test_numerical_filters*), 39
test_filter_pixel_truncation() (*in module ifa_smeargle.testing.test_numerical_filters*), 39
test_filter_sigma_value() (*in module ifa_smeargle.testing.test_numerical_filters*), 39
test_ifas_large_integer_array_product() (*in module ifa_smeargle.testing.test_numerical_core_mathematic*
38
test_ifas_masked_mean() (*in module ifa_smeargle.testing.test_numerical_core_mathematics*), 38
test_ifas_masked_median() (*in module ifa_smeargle.testing.test_numerical_core_mathematics*), 38
test_ifas_masked_std() (*in module ifa_smeargle.testing.test_numerical_core_mathematics*), 38
test_ifas_robust_mean() (*in module ifa_smeargle.testing.test_numerical_core_mathematics*), 38
test_ifas_robust_std() (*in module ifa_smeargle.testing.test_numerical_core_mathematics*), 38
test_mask_columns() (*in module ifa_smeargle.testing.test_numerical_masking*), 39
test_mask_everything() (*in module ifa_smeargle.testing.test_numerical_masking*), 39
test_mask_nothing() (*in module*

ifa_smeargle.testing.test_numerical_masking), 39
`test_mask_rectangle()` (*in module ifa_smeargle.testing.test_numerical_masking*), 39
`test_mask_rows()` (*in module ifa_smeargle.testing.test_numerical_masking*), 39
`test_mask_single_pixels()` (*in module ifa_smeargle.testing.test_numerical_masking*), 39
`test_mask_subarray()` (*in module ifa_smeargle.testing.test_numerical_masking*), 39
`test_pass()` (*in module ifa_smeargle.testing.test_global*), 38
`TimeWarning`, 10
`tutorial_generate_fits_file()` (*in module ifa_smeargle.tutorial.generation*), 40

W

`write_configuration_file()` (*in module ifa_smeargle.core.configuration*), 7
`write_fits_file()` (*in module ifa_smeargle.core.io*), 14
`write_plot_file()` (*in module ifa_smeargle.plotting.base_functions*), 29