FITSImageQA

Release 0.0.1

Will Bowman, Omari Paul

Jul 19, 2023

CONTENTS:

CHAPTER

ONE

IMAGEQA

class imageqa.**ImageQA**(*filename=None*)

Bases: object

check_is_corrupt()

check whether the FITS file is corrupt

TODO: fill in

class imageqa.**QAHeader**(*filename_or_hdr: str | HDUList | Header*, *expected_fields: list[str] | None = None*, *expected_fields_dtype: dict | None = None*, *qadata:* QAData *| None = None*)

Bases: *ImageQA*

check_header_fields_dtype(*expected_fields_dtype: dict | None = None*, *return_incorrect_fields: bool = False*, *exit_on_fail: bool = True*, *suppress_unknown: bool = True*, *overwrite_attribute: bool = False*, *verbose: bool = False*)

Check that data type of header fields are as expected

Parameters

- **expected_fields_dtype** (*iter of key-value pairs*) – keys = header names (str) values = data types (result of *type()* or iter of result of *type()*)
- **return_incorrect_fields** (*bool*) – which fields are of the wrong data type (empty, if valid=True)
- **exit_on_fail** (*bool*) – raise TypeError at first instance of failure
- **suppress_unknown** (*bool*) – if True, do not break if header datatype cannot be checked (i.e., column exists in *self.header_fields* but is not a key in *expected_fields_dtype* list)
- **overwrite_attribute** (*bool (default=False)*) – reset the *expected_fields_dtype* attribute with the locally-passed list
- **verbose** (*bool*) – print warnings if fields cannot be checked, or if fields are wrong data type

Returns

- **passed** (*bool*) – are the header fields of the expected data type?
- **incorrect_fields** (*dict (optional)*) – keys : str, column names that did not pass the dtype check values : list of str [header_value, type(header_value)]

Notes

self.header_fields and keys(expected_fields_dtype) can be overlapping or subsets of one another

check_header_fields_present(*expected_fields: list[str] | None = None*, *return_missing_fields: bool = False*, *overwrite_attribute: bool = False*)

Parameters

- **expected_fields** (*iterable of str (optional)*) – if passed, use instead of self.expected_fields
- **return_missing_fields** (*bool (default=False)*) – if True, return fields from *expected_fields* that are not present in image header
- **overwrite_attribute** (*bool (default=False)*) – reset the *expected_fields* attribute with the locally-passed list

Returns

- **valid** (*bool*) – are the desired fields present in the header?
- **missing_fields** (*iter, optional*) – which fields are missing from the header (empty, if valid=True)

fetch_header_info(*column_name: str*, *suppress_error: bool = False*)

Get info from the header

Parameters

- **column_name** (*str*) – should be an element of list(self.hdr.keys())
- **suppress_error** (*bool*) – fail silently, if the value is not found

Returns
return the value from the header, corresponding to the key *column_name*

Return type
info

class imageqa.**QAData**(*filename_or_data: str | HDUList | ndarray*, *detection_config: dict | None = None*, *qahdr: QAHeader | None = None*)

Bases: *ImageQA*

detect_sources(*detection_config: dict | None = None*, *overwrite: bool = True*, ***kwargs*)

Detect sources in the image and store the result as an attribute

Parameters

- **detection_config** (*dict*) – dictionary of source detection parameters that will be passed to detection.extract_sources
- **overwrite** (*bool*) – if True, will automatically overwrite existing *sources*
- **kwargs** (*additional arguments to pass to detection.extract_sources*) –

Returns
set (or overwrite) self.sources with an instance of the detection.Sources class

Return type
None

Notes

If *sources* attribute already exists, it will be overwritten

display_image(*add_detections: bool = False*, *hdr:* QAHeader | *None = None*, ***kwargs*)

Display the image

Parameters

- **add_detections** (*(bool)*) – overplot sources detected via the *detect_sources* method
- **(optional)** (*kwargs*) – passed to <INSERT_PLOTTING_FUNCTION>

Returns
fig

Return type
<INSERT_PLOTTING_FUNCTION>

is_focus_good(*max_focus_fwhm: float | int = 2.5*)

Determine whether a focus run is needed while observing, by comparing the FWHM of sources that are detected in the image to a user-specified maximum threshold

Parameters
max_focus_fwhm (*float*) – maximum value of the FWHM, to consider the image to be in focus

Returns

- **in_focus** (*bool*) – True, if med_fwhm <= max_focus_fwhm
- **med_fwhm** (*float*) – median FWHM of detected sources

Notes

Source detection will be run via self.detect_sources
if no result already exists (stored in self.sources)

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i

INDEX