Pandeia Reports

This document describes the structure and content of JWST Pandeia output reports.

Introduction

Pandeia can produce two possible outputs from run_calculation. If the command is run as "`perform_calculation(<configuration>, dict_report=True)`", (or simply as "`perform_calculation(<configuration>)`"), since "`dict_report`" defaults to True), it will return a dictionary with various information about the calculation inputs and results. If, instead, the command is run as "`perform_calculation(<configuration>, dict_report=False)`", it will return a "pandeia.engine.Report" object, which will have the same information available, and which can output a dictionary (with exactly the same contents as if "`dict_report`" had been set to true) via the "`as_dict()`" method. In general, the dictionary report is recommended unless there some specific output information is desired but not obtainable from the dictionary (see below for examples of this).

Pandeia report dictionary structure

The pandeia report dictionary contains the following keys and content:

- **sub_reports**: list, containing information on each exposure for multiple exposures (e.g. IFU dither patterns or coronagraphic imaging). Each report in the sub_reports list will be in the same format as a standard dictionary report.
- **information**: dictionary, containing information on the exposure specifications that Pandeia used to generate its result. It contains
  - **calc_type**: string, the type of projection performed. One of 'slitless', 'multiorder', 'image', 'spec'
  - **exposure_specification**: dictionary. Contains information on how the exposure was conducted and the assumed telescope and instrument status. Contains the following:
    - **exposure_time**: floating point, seconds of exposure time for each exposure.
    - **total_exposure_time**: floating point, total number of seconds spent observing the target. Equal to exposure_time multiplied by the number of exposures.
    - **measurement_time**: floating point, number of seconds between the first and last measurements of a pixel in an integration multiplied by the number of integrations per exposure.
    - **saturation_time**: floating point, number of seconds from the reset of a pixel to the final read of that pixel in an integration multiplied by the number of integrations per exposure

        - **duty_cycle**: floating point, the fraction of the exposure time that was devoted to observing the target. duty_cycle = measurement_time/exposure_time
        - **nexp**: integer, number of exposures
        - **nframe**: integer, number of frames per group
        - **tframe**: floating point, number of seconds per frame
        - **nskip**: integer, number of skipped frames per group. Only supported by some readout patterns.
        - **frame0**: boolean, whether the first frame was downlinked and used in the ramp fit. Only applies to some detectors.
        - **ngroup**: integer, number of groups per integration
        - **tgroup**: floating point, seconds of telescope time per group.
        - **nprerej**: integer, either 0 or 1 (default 0). Applies only to MIRI. Number of groups at the beginning of the ramp that were rejected by the pipeline.
        - **nint**: integer, number of integrations per exposure
        - **npostrej**: integer, either 0 or 1 (default 0). Applies only to MIRI. Number of groups at the end of the ramp that were rejected by the pipeline.
        - **total_integrations**: integer, number of integrations in all exposures combined.
        - **nramps**: integer, total number of ramps in the observation. nramps is equal to nint multiplied by the number of exposures
        - **pattern**: string, name of the readout pattern
        - **subarray**: string, name of subarray used for the exposure
        - **det_type**: string, type of detector involved (e.g. 'h2rg' for NIRCam short-wavelength detectors)
        - **nsample**: integer, number of samples averaged when reading a single pixel. Applies only to MIRI.
        - **tsample**: floating point, time averaged when reading a single pixel. Applies only to MIRI.
        - **nsample_skip**: integer, number of samples skipped while reading a single pixel. Applies only to MIRI.
        - **nsample_total**: integer, total number of samples
        - **tffr**: integer, extra time factor.
- **warnings**: dictionary, contains all warnings generated by the exposure. Warnings can include:
    - **<instrument>_missing_instrument_aperture**: generated if the supplied configuration dictionary did not include the aperture information. The warning text will indicate what the aperture has been set to.
    - **partial_saturated**: indicates partial saturation of at least one pixel. The warning text will include the number of partially saturated pixels.
    - **full_saturated**: indicates full (unrecoverable) saturation of at least one pixel. The warning text will include the number of fully saturated pixels.
- **transform**: dictionary, containing the coordinate transform information that describes the image axes for the 2d and 3d data. Contains the following values:
    - **wave_refpix**: integer, the reference pixel of the wavelength arrays
    - **wave_refval**: floating point, the value (in microns) of wave_refpix
    - **wave_step**: floating point, interval (in microns) between successive points in the wavelength array.
    - **wave_size**: integer, the size of the wavelength axis, in pixels
    - **wave_min**: floating point, the minimum wavelength value (in microns) of the 2d and 3d data
    - **wave_max**: floating point, maximum wavelength (in microns) of the 2d and 3d data

    - **x_refpix**: integer, the reference pixel for the x axis in the 2d and 3d data
    - **x_refval**: floating point, the value (in arcseconds) of the x_refpix pixel of the x axis of the 2d and 3d data
    - **x_step**: the per-pixel size (in arcseconds) of the x axis of the 2d and 3d data
    - **x_size**: integer, the size of the x axis of the 2d and 3d data supplied in the report.
    - **x_min**: floating point, the minimum value (in arcseconds) of the x axis of the 2d and 3d data
    - **x_max**: floating point, the maximum value (in arcseconds) of the x axis of the 2d and 3d data.
    - **y_refpix**: integer, the reference pixel for the y axis in the 2d and 3d data
    - **y_refval**: floating point, the value (in arcseconds) of the y_refpix pixel of the y axis of the 2d and 3d data
    - **y_step**: floating point, the per-pixel size (in arcseconds) of the y axis of the 2d and 3d data
    - **y_size**: integer, the size of the y axis in the 2d and 3d data
    - **y_min**: floating point, the minimum value (in arcseconds) of the y axis in the 2d and 3d data
    - **y_max**: floating point, the maximum value (in arcseconds) of the x axis in the 2d and 3d data
    - **wave_det_refpix**: integer, the detector reference pixel in wavelength space (if applicable).
    - **wave_det_refval**: floating point, the value (in microns) of wave_refpix on the detector (if applicable)
    - **wave_det_step**: floating point, the distance in wavelength space between adjacent pixels on the detector (if applicable).
    - **wave_det_size**: integer, the length of the wavelength axis of the detector, in pixels
    - **wave_det_min**: floating point, the minimum wavelength value (in microns) of the detector
    - **wave_det_max**: floating point, the maximum wavelength value (in microns) of the detector
- **scalar**: dictionary, contains scalar values of interest from the observation. Includes the following values:
    - **exposure_time**: floating point, the same value as exposure_time in the '`information: exposure_specification`' part of the report
    - **aperture_size**: floating point, size of the extraction aperture in arcseconds.
    - **cr_ramp_rate**: floating point, expected cosmic rays per pixel per ramp
    - **reference_wavelength**: floating point, wavelength (in microns) used for determining scalar outputs
    - **extraction_area**: floating points, area (in square arcseconds) from which data is extracted for scalar outputs
    - **all_dithers_time**: floating point, total time (in seconds) spent during the observation during all exposures/dithers, whether on-target or off-target
    - **total_integrations**: integer, the same value as total_integrations in the '`information: exposure_specification`' part of the report
    - **saturation_time**: floating point, the same value as saturation_time in the '`information: exposure_specification`' part of the report
    - **measurement_time**: floating point, the same value as measurement_time in the '`information: exposure_specification`' part of the report
    - **fraction_saturation**: floating point, the fraction describing how close to saturation the brightest pixel on the detector is.
    - **detector_ngroups**: integer, the maximum number of groups that can be requested *before* the brightest pixel on the detector saturates.
    - **sn**: floating point, signal-to-noise ratio for the observation
    - **extracted_noise**: floating point, total noise counts extracted in the background region
    - **extracted_flux**: floating point, total flux extracted from the extraction aperture

    - **duty_cycle**: floating point, the same value as duty_cycle in the '`information: exposure_specification`' part of the report
    - **background**: floating point, flux of the background spectrum at the reference wavelength
    - **contamination**: floating point, flux from contamination by other (non-target) sources in the background region
    - **background_sky**: floating point, flux in the background extraction region contributed by the sky
    - **disperser**: string, the grating/prism/grism in use, if any
    - **filter**: string, the filter in use, if any
    - **y_offset**: floating point, the y offset of the target
    - **x_offset**: floating point, the x offset of the target
    - **background_area**: floating point, the background extraction area in square arcseconds
    - **background_total**: floating point, the background flux including both sky and contamination
    - **total_exposure_time**: floating point, the same value as total_exposure_time in the '`information: exposure_specification`' part of the report
- **1d**: dictionary, Contains 1d spectra representing various aspects of the observation, presented either as a function of wavelength or alongside the reference wavelength. In the descriptions below, names in italics are single values presented against the reference wavelength if the calculation is an 'imaging' calculation. Values include:
    - **fp**: count rate at the focal plane
    - ***extracted_contamination***: flux from contamination
    - ***total_flux***: combined target and background flux
    - **bg**: background flux
    - ***extracted_noise***: standard deviation of the extracted flux
    - **bg_rate**: background flux at the focal plane
    - ***extracted_bg_only***: extracted background flux not including contamination (if any)
    - ***n_partial_saturated***: number of partially saturated pixels
    - ***n_full_saturated***: number of fully saturated pixels
    - ***sn***: signal-to-noise ratio
    - **wave_pix**: single array containing the reference wavelength as its only value
    - ***extracted_flux***: total extracted flux from the target
    - ***extracted_flux_plus_bg***: total countrate including both target and background
    - **wave_calc**: single array containing the wavelengths over which the exposure was calculated
    - ***extracted_bg_total***: total extracted background flux
    - **target**: target flux as a function of wavelength
- **2d**: dictionary, contains 2d images of the on-detector observation. Values include
    - **snr**: 2d numpy array of floating point values, the signal-to-noise ratio for each pixel in the observation
    - **detector**: 2d numpy array of floating point values, the on-detector countrate for each pixel in the observation
    - **saturation**: 2d numpy array of floating point values, the detector saturation information. Pixels have the value of 0 (unsaturated), 1 (partial saturation), or 2 (full saturation).
    - **ngroups_map**: 2d numpy array of integer values, the maximum number of groups *before* a given pixel on the detector would saturate.
- **3d**: dictionary, contains 3d datacubes of the on-detector observations. Values include

    - **flux**: 3d numpy array of floating point values, contains flux as a function of wavelength for each pixel at each wavelength in pandeia's internal observation measurement.
    - **flux_plus_background**: 3d numpy array of floating point values, contains flux and background together for each pixel at each wavelength.
- **input**: dictionary, containing information about the calculation inputs. When running Pandeia from python, this should simply be a copy of the configuration dictionary (see Pandeia Configuration Dictionaries for more information on the configuration dictionary). As such, its contents will not be further discussed here.

Pandeia report fits object

Pandeia can also output a dictionary of fits objects, by outputting a report object (created by running "`perform_calculation(<configuration>, dict_report=False)`" and running the '`as_fits()`' method on the resulting Report object)

- **1d**: dictionary, with a fits hdulist for every item in the regular 1D report dictionary mentioned above. All of the fits headers except wave_calc and wave_pix will have two fields in the first header: wavelength (either wave_calc or wave_pix) and the property in question.
- **2d**: dictionary, with a fits hdulist for every item in the regular 2D report dictionary. All of the fits headers have WCS information extracted from the detector pixel grid. These headers are, for 2D spectroscopic modes, assumed to be linear wavelength scales; it may be necessary to use the 1D wave_calc value to generate accurate wavelengths.
- **3d**: dictionary, with a fits hdulist for every item in the regular 3D report dictionary. All of the fits headers have WCS information extracted from the detector pixel grid. These headers are assumed to be linear in wavelength scale; it may be necessary to use the 1D wave_calc values to generate accurate wavelengths.

Pandeia report object

The pandeia '`Report`' object contains the same data as the report dictionary (and can be converted into the report dictionary via its '`as_dict()`' method, or fits dictionary via its '`as_fits()`' method), but also contains some additional information.

Per-pixel background count rate

The Pandeia Report object stores the per-pixel background count rate in the internal property '`bg_pix`'. This contains a 2d array of the entire simulated region of the detector with the per-pixel sky background for each pixel, and could potentially be useful in determining background flux in order to compare observations when looking for the lowest possible background.

Interpreting downloads from the JWST ETC website

When running simulations on the JWST ETC website, you can download the result of a particular simulation. In addition to the input values used for the simulation (discussed in the Pandeia Quickstart, the download also contains many FITS files which provide information about the simulation, and which often correspond to the content of a Pandeia report.

Structure of a JWST ETC result download

JWST ETC downloads contain the following files and folders. Note that these files are also discussed on the JWST ETC Downloads page.

- backgrounds.fits: FITS file containing the spectrum used to determine background count rates
- cube (non-IFU observations)
    - cube_flux_plus_background.fits
    - cube_flux.fits
    - model (empty folder)
- cube (IFU observations)
    - cube_reconstructed.fits
    - cube_reconstructed_noise.fits
    - cube_reconstructed_snr.fits
    - cube_reconstructed_saturation.fits
    - model
        - cube_flux_n.fits (one for each IFU element, with the n replaced by the element number)
        - cube_flux_plus_background_n.fits (one for each IFU element, with the n replaced by the element number)
- image
    - image_detector.fits
    - image_saturation.fits
    - image_snr.fits
- input.json (discussed in the Pandeia Quickstart)
- lineplot
    - lineplot_bg_rate.fits
    - lineplot_bg.fits
    - lineplot_extracted_bg_only.fits
    - lineplot_extracted_bg_total.fits
    - lineplot_extracted_contamination.fits
    - lineplot_extracted_flux_plus_bg.fits
    - lineplot_extracted_flux.fits
    - lineplot_extracted_noise.fits
    - lineplot_fp.fits
    - lineplot_n_full_saturated.fits
    - lineplot_n_partial_saturated.fits
    - lineplot_sn.fits

- lineplot_target.fits
- lineplot_total_flux.fits
- lineplot_wave_calc.fits
- lineplot_wave_pix.fits

Accessing downloaded FITS files

If you have astroconda installed, FITS files are most easily accessed via the astropy module, in particular astropy.io.fits. Below is an example of obtaining data from a lineplot file, assuming that python is run from the base directory of the download:

access_download_fits_file

```
import astropy.io.fits as f

with f.open('lineplot/lineplot_extracted_bg_total.fits', 'r') as input_file:
    reference_wavelength = input_file[1].data[0]['WAVELENGTH']
    extracted_flux = input_file[1].data[0]['extracted_flux']
```

Correspondences with the Pandeia report dictionary

File	Extension	Equivalent Key
cube/cube_flux_plus_background.fits	0	report['3d']['flux_plus_background']
cube/cube_flux.fits	0	report['3d']['flux']
image/image_detector.fits	0	report['2d']['detector']
image/image_saturation.fits	0	report['2d']['saturation']
image/image_snr.fits	0	report['2d']['snr']
image/image_ngroups_map.fits	0	report['2d']['ngroups_map']
lineplot/lineplot_bg_rate.fits	1	report['1d']['bg_rate']
lineplot/lineplot_bg.fits	1	report['1d']['bg']
lineplot/lineplot_extracted_bg_only.fits	1	report['1d']['extracted_bg_only']
lineplot/lineplot_extracted_bg_total.fits	1	report['1d']['extracted_bg_total']
lineplot/lineplot_extracted_contamination.fits	1	report['scalar']['contamination']
lineplot/lineplot_extracted_flux_plus_bg.fits	1	report['1d']['extracted_flux_plus_bg']
lineplot/lineplot_extracted_flux.fits	1	report['1d']['extracted_flux']

lineplot/lineplot_extracted_noise.fits	1	report['1d']['extracted_noise']
lineplot/lineplot_fp.fits	1	report['1d']['fp']
lineplot/lineplot_n_full_saturated.fits	1	report['1d']['n_full_saturated']
lineplot/lineplot_n_partial_saturated.fits	1	report['1d']['n_partial_saturated']
lineplot/lineplot_sn.fits	1	report['1d']['sn']
lineplot/lineplot_target.fits	1	report['1d']['target']
lineplot/lineplot_total_flux.fits	1	report['1d']['total_flux']
lineplot/lineplot_wave_calc.fits	1	report['1d']['wave_calc']
lineplot/lineplot_wave_pix.fits	1	report['1d']['wave_pix']

Downloaded background file

The backgrounds.fits file included in the download contains a data table in its first extension. This table includes the following columns:

- wavelength: the wavelength in microns
- background: total background flux
- thermal: the thermal component of the total background flux
- straylight: the stray light component of the background flux
- infield: the sky component of the background flux

Sample code

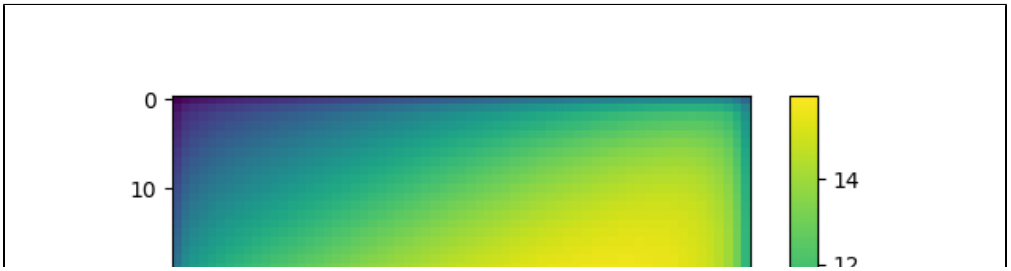
Displaying an image of the signal-to-noise ratio of a Pandeia imaging observation

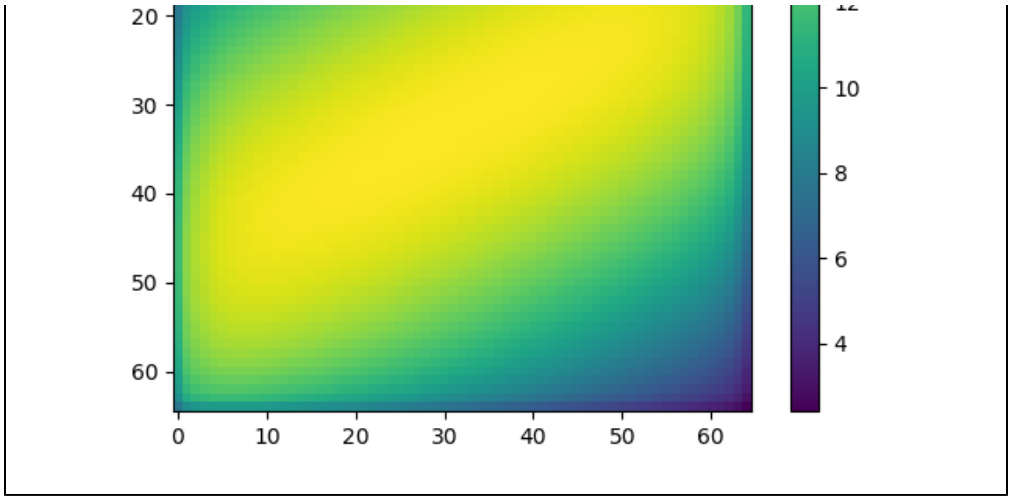
display_snr_image

```
import matplotlib.pyplot as plt

# The following section is only needed if the PYSYN_CDBS environment variable is not already set.
# The PYSYN_CDBS environment variable should point to the path of the CDBS data files
import os
location_of_cdbs = "/path/to/cdbs/files"
os.environ['PYSYN_CDBS'] = location_of_cdbs
```

```
# End section

# The following section is only needed if the pandeia_refdata environment variable is not already set
# The pandeia_refdata environment variable should point to the path of the pandeia reference data
import os
location_of_pandeia_refdata = "/path/to/pandeia/refdata"
os.environ['pandeia_refdata'] = location_of_pandeia_refdata
# End section
from pandeia.engine.calc_utils import build_default_calc
from pandeia.engine.perform_calculation import perform_calculation

# The following are parameters which can easily be changed
telescope = 'jwst'
instrument = 'nircam'
mode = 'sw_imaging'

# Source parameters
offsets = {'x': 0., 'y': 0.}
orientation = 27.
geometry = 'gaussian2d'
major_axis = 8.5 # arcseconds
minor_axis = 0.75 # arcseconds
name = 'Blackbody'
sed = 'blackbody'
temp = 50000.
bandpass = 'bessel,j'
magnitude = 18.

configuration = build_default_calc(telescope, instrument, mode)
scene = {}
scene['position'] = {'position_parameters': ['x_offset', 'y_offset', 'orientation']}
scene['position']['x_offset'] = offsets['x']
scene['position']['y_offset'] = offsets['y']
scene['position']['orientation'] = orientation
scene['shape'] = {'geometry': geometry, 'major': major_axis, 'minor': minor_axis, 'norm_method':
'integ_infinity', 'surf_area_units': 'arcsec^2'}
scene['spectrum'] = {'name': name, 'spectrum_parameters': ['sed', 'normalization']}
scene['spectrum']['sed'] = {'sed_type': sed, 'temp': temp}
scene['spectrum']['normalization'] = {'type': 'photsys', 'norm_fluxunit': 'abmag'}
scene['spectrum']['normalization']['bandpass'] = bandpass
scene['spectrum']['normalization']['norm_flux'] = magnitude
configuration['scene'][0] = scene

report = perform_calculation(configuration)

plt.imshow(report['2d']['snr'])
plt.colorbar()
plt.show()
```

Note that the above is being done in a very simplistic way, and that matplotlib, in particular, has many options that aren't on display here. See the matplotlib user's guide for many more details.





Displaying the signal-to-noise ratio of a spectrum as a function of wavelength

display_snr_spectrum

```
import matplotlib.pyplot as plt

# The following section is only needed if the PYSYN_CDBS environment variable is not already set.
# The PYSYN_CDBS environment variable should point to the path of the CDBS data files
import os
location_of_cdbs = "/path/to/cdbs/files"
os.environ['PYSYN_CDBS'] = location_of_cdbs
# End section

# The following section is only needed if the pandeia_refdata environment variable is not already set
# The pandeia_refdata environment variable should point to the path of the pandeia reference data
import os
location_of_pandeia_refdata = "/path/to/pandeia/refdata"
os.environ['pandeia_refdata'] = location_of_pandeia_refdata
# End section
from pandeia.engine.calc_utils import build_default_calc
from pandeia.engine.perform_calculation import perform_calculation

# The following are parameters which can easily be changed
telescope = 'jwst'
instrument = 'nirspec'
mode = 'fixed_slit'

# Source parameters
offsets = {'x': 0., 'y': 0.}
geometry = 'gaussian2d'
major_axis = 8.5 # arcseconds
minor_axis = 0.75 # arcseconds
name = 'G2V Star'
sed = 'phoenix'
key = 'g2v'
bandpass = 'sdss,r'
magnitude = 18.

configuration = build_default_calc(telescope, instrument, mode)
scene = {}
scene['position'] = {'position_parameters': ['x_offset', 'y_offset']}
scene['position']['x_offset'] = offsets['x']
```

```
scene['position']['y_offset'] = offsets['y']
scene['shape'] = {'geometry': 'point'}
scene['spectrum'] = {'name': name, 'spectrum_parameters': ['sed', 'normalization']}
scene['spectrum']['sed'] = {'sed_type': sed, 'key': key}
scene['spectrum']['normalization'] = {'type': 'photsys', 'norm_fluxunit': 'abmag'}
scene['spectrum']['normalization']['bandpass'] = bandpass
scene['spectrum']['normalization']['norm_flux'] = magnitude
configuration['scene'][0] = scene

report = perform_calculation(configuration)

disperser = configuration['configuration']['instrument']['disperser']
fig, ax = plt.subplots()
ax.plot(report['1d']['sn'][0], report['1d']['sn'][1])
ax.set(xlabel='Wavelength (micron)', ylabel='SNR', title='G2V Star observed with NIRISS {}'.format(disperser))
plt.show()
```