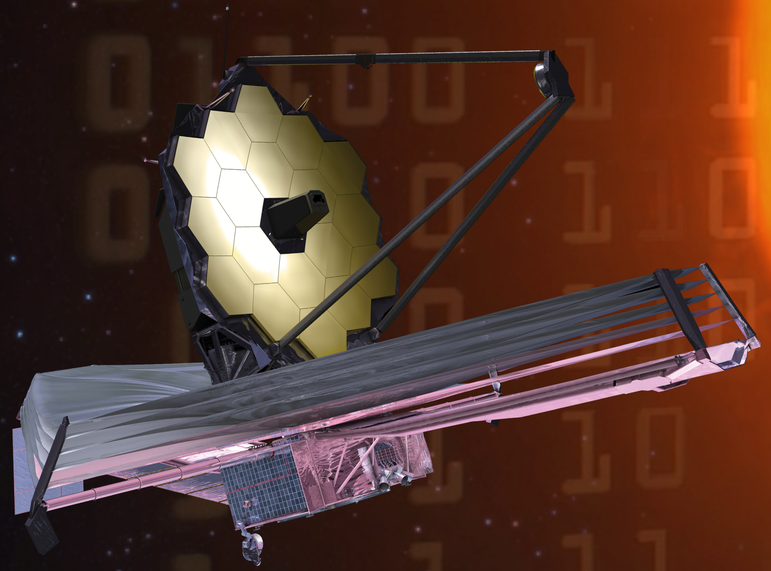


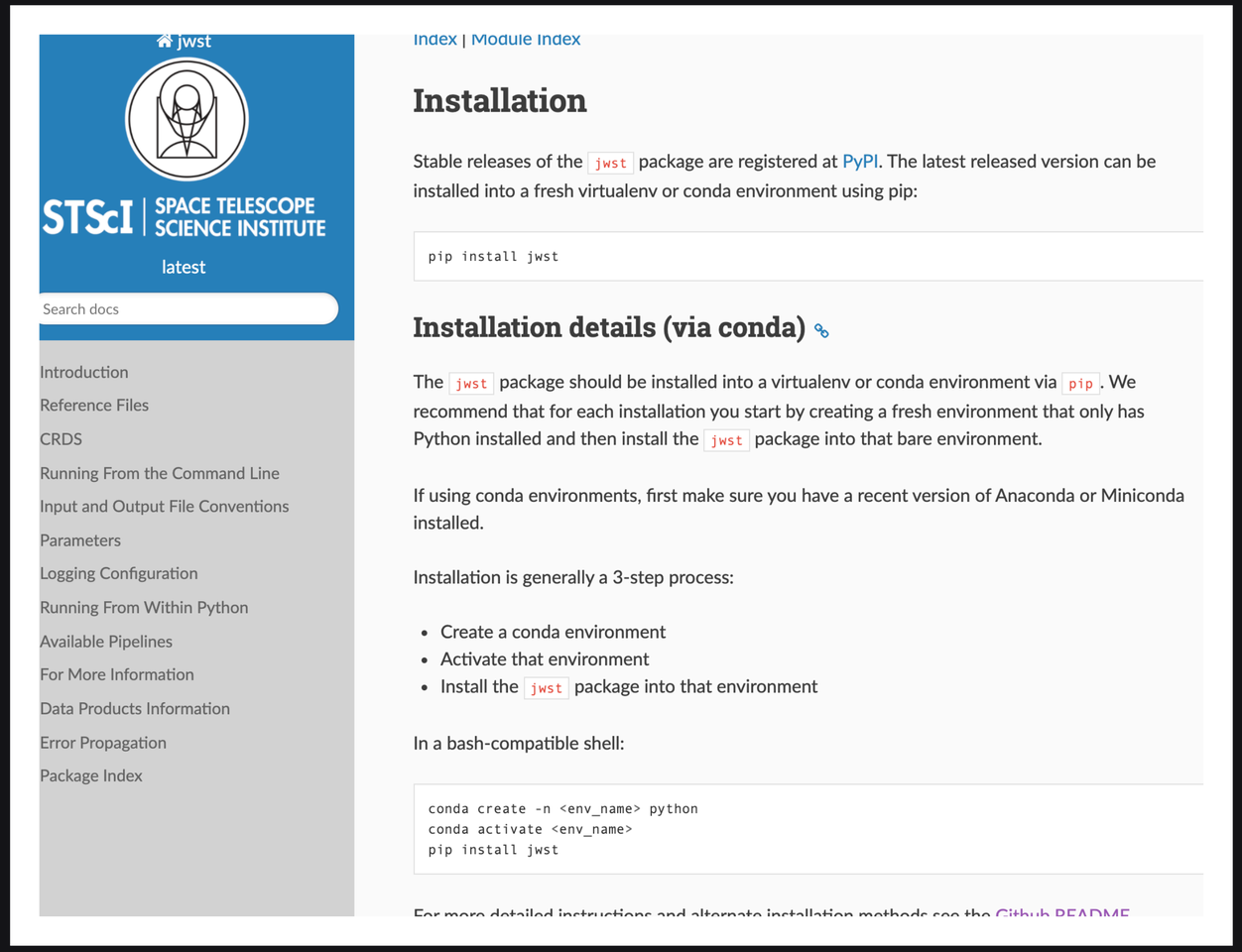
HOW TO ENGAGE WITH THE JWST CALIBRATION PIPELINE?

NÉSTOR ESPINOZA | STScI | SPACE TELESCOPE SCIENCE INSTITUTE

THE JWST DATA REDUCTION PIPELINE

THE JWST DATA REDUCTION PIPELINE

jwst-pipeline.readthedocs.io



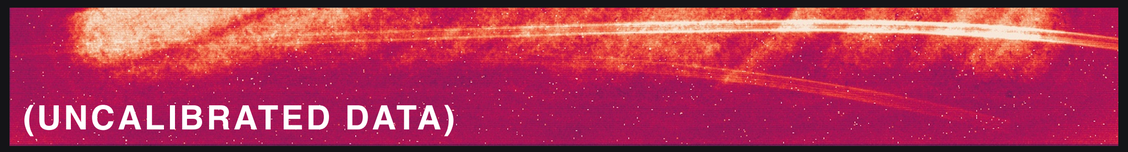
- Really, *a* JWST pipeline.
- Coded in **Python.** **Modular.** Open source.
- Algorithms tailored to particular instruments and modes.
- Has direct input from JWST instrument teams.
- Currently under active development at STScI.

*TSO: Time-Series Observation

THE JWST DATA REDUCTION PIPELINE

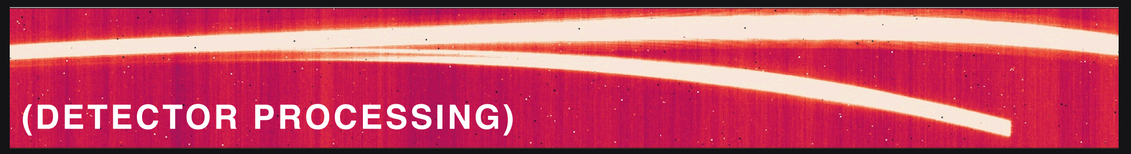
The calibration pipeline is divided into Stages. For *TSOs:

STAGE 0:

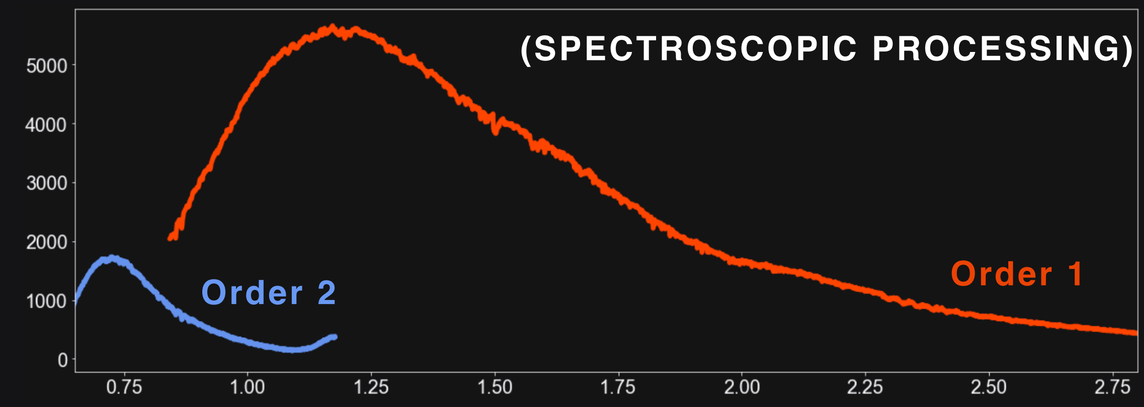


Simulated uncalibrated (*.uncal.fits) NIRISS/SOSS data

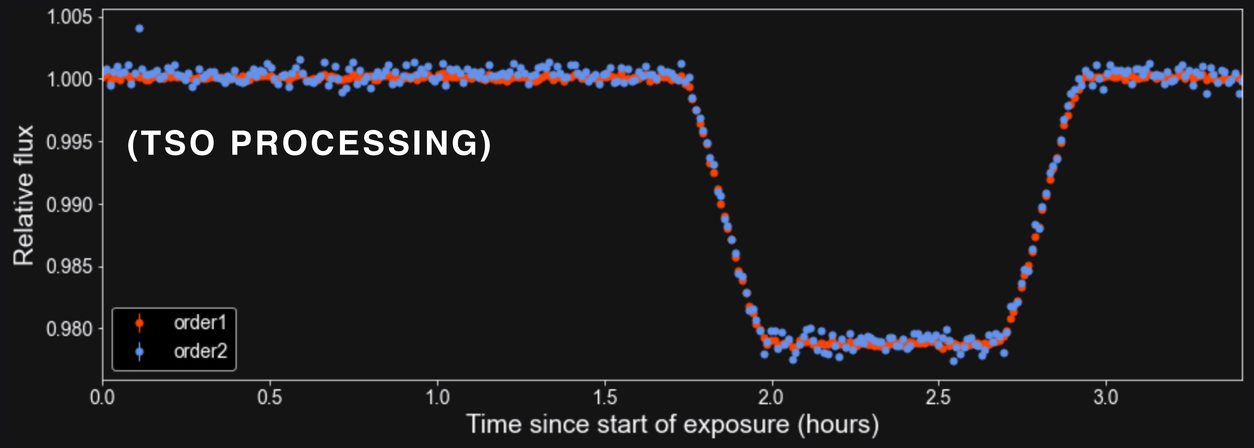
AFTER STAGE 1:



AFTER STAGE 2:



AFTER STAGE 3:



*TSO: Time-Series Observation

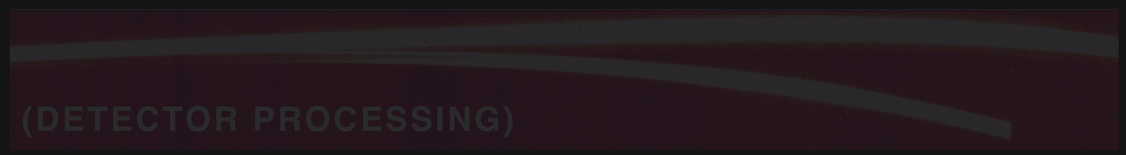
THE JWST DATA REDUCTION PIPELINE

The calibration pipeline is divided into Stages. For *TSOs:

STAGE 0:

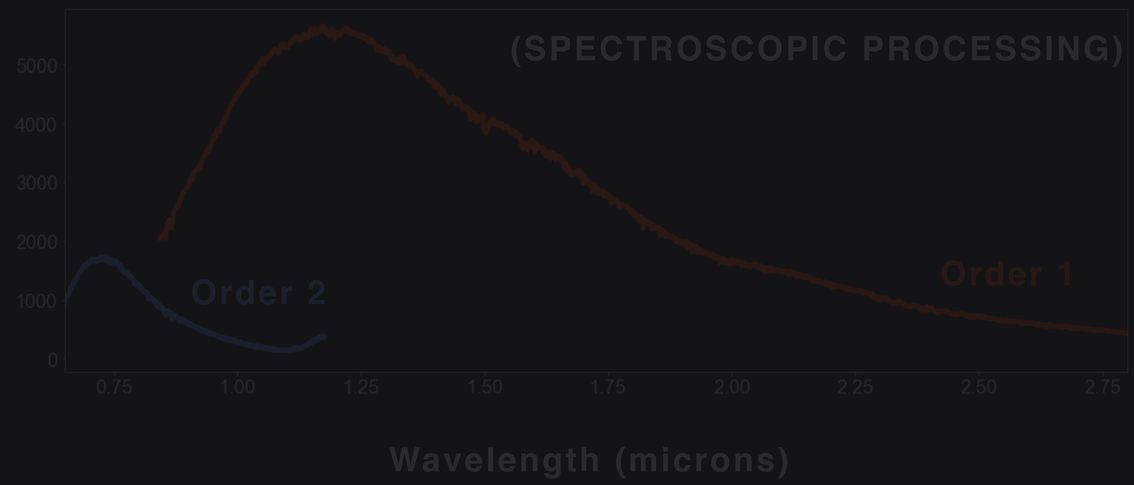
Simulated uncalibrated (*.uncal.fits) NIRISS/SOSS data

AFTER STAGE 1:



Pipeline allows you to run each stage in one line of code.

AFTER STAGE 2:



AFTER STAGE 3:

*TSO: Time-Series Observation

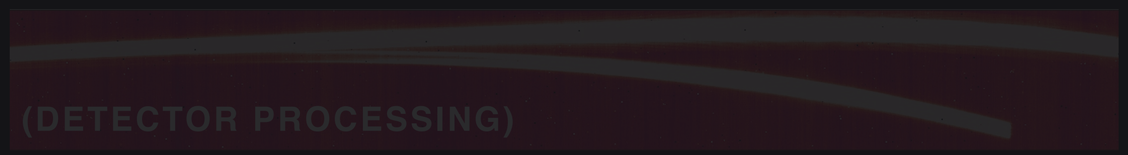
THE JWST DATA REDUCTION PIPELINE

The calibration pipeline is divided into Stages. For *TSOs:

STAGE 0:

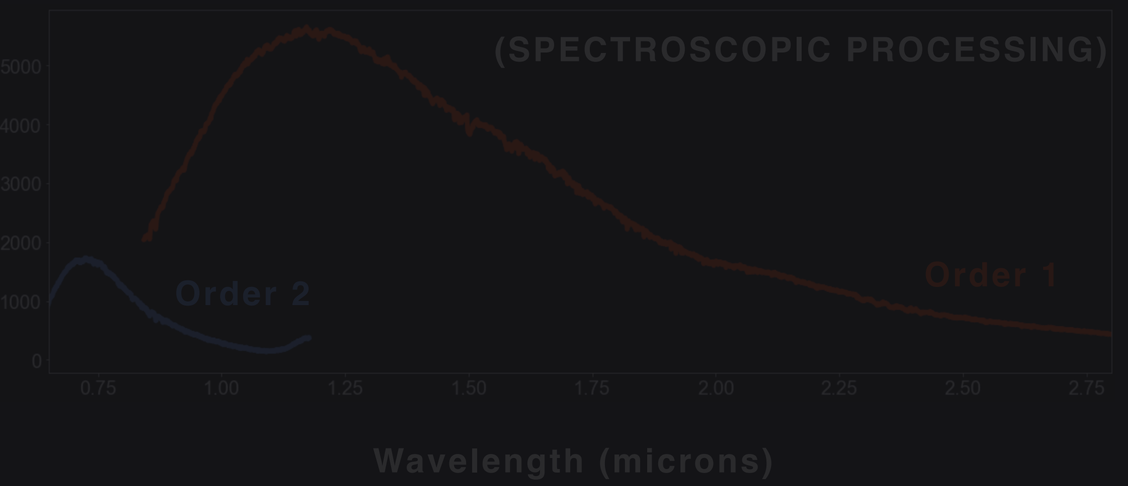
Simulated uncalibrated (*.uncal.fits) NIRISS/SOSS data

AFTER STAGE 1:



Pipeline allows you to run each stage in one line of code.

Here, however, I'll focus on how to run individual steps.



THE JWST DATA REDUCTION PIPELINE

Running the pipeline in a jupyter notebook

THE JWST DATA REDUCTION PIPELINE

Running the pipeline in a jupyter notebook

```
# Need to set these enviromental variables for this notebook to work properly:
%set_env CRDS_PATH $HOME/crds_cache
%set_env CRDS_SERVER_URL https://jwst-crds.stsci.edu

# Import all useful libraries:
import os
import numpy as np
import matplotlib.pyplot as plt

from astropy.io import fits

from jwst.pipeline import calwebb_detector1
from jwst.pipeline import calwebb_spec2
```

THE JWST DATA REDUCTION PIPELINE

Running the pipeline in a jupyter notebook

(JWST Calibration Reference Data System)

Needed to be able to get files from CRDS

```
# Need to set these enviromental variables for this notebook to work properly:
%set_env CRDS_PATH $HOME/crds_cache
%set_env CRDS_SERVER_URL https://jwst-crds.stsci.edu

# Import all useful libraries:
import os
import numpy as np
import matplotlib.pyplot as plt

from astropy.io import fits

from jwst.pipeline import calwebb_detector1
from jwst.pipeline import calwebb_spec2
```

Steps from Stages 1 and 2

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

Near-IR			MIRI		
Step	**Non-TSO**	**TSO**	**Step**	**Non-TSO**	**TSO**
group_scale	✓	✓	group_scale	✓	✓
dq_init	✓	✓	dq_init	✓	✓
saturation	✓	✓	saturation	✓	✓
ipc [1]			ipc		
superbias	✓	✓	firstframe	✓	
refpix	✓	✓	lastframe	✓	
linearity	✓	✓	linearity	✓	✓
persistence [2]	✓		rscd	✓	
dark_current	✓	✓	dark_current	✓	✓
			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

Near-IR			MIRI		
Step	**Non-TSO**	**TSO**	**Step**	**Non-TSO**	**TSO**
group_scale	✓	✓	group_scale	✓	✓
dq_init	✓	✓	dq_init	✓	✓
saturation	✓	✓	saturation	✓	✓
ipc [1]			ipc		
superbias	✓	✓	firstframe	✓	
refpix	✓	✓	lastframe	✓	
linearity	✓	✓	linearity	✓	✓
persistence [2]	✓		rscd	✓	
dark_current	✓	✓	dark_current	✓	✓
			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

The dq_init step of Stage 1 simply **populates data-quality flags** into our groups.

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

```
# Create output directory:
os.mkdir('pipeline_outputs_directory')

# Run step:
dq_results = calwebb_detector1.dq_init_step.DQInitStep.call('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits',
                                                            output_dir='pipeline_outputs_directory',
                                                            save_results=True)
```

Note that in your first run of the JWST pipeline, you will probably see much more outputs than the ones above. That's normal.

All right! The `DQInitStep` has went through successfully. What did it do? Let's compare the before and after:

```
data_before = fits.open('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits')
print('Before:')
print(data_before.info())

data_after = fits.open('pipeline_outputs_directory/WASP43_NIS_SOSS-seg002_CLEAR_dqinitstep.fits')
print('\nAfter:')
print(data_after.info())
```

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

```
# Create output directory:
os.mkdir('pipeline_outputs_directory')

# Run step:
dq_results = calwebb_detector1.dq_init_step.DQInitStep.call('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits',
                                                            output_dir='pipeline_outputs_directory',
                                                            save_results=True)
```

Stage (1 here)

Note that in your first run of the JWST pipeline, you will probably see much more outputs than the ones above. That's normal.

All right! The `DQInitStep` has went through successfully. What did it do? Let's compare the before and after:

```
data_before = fits.open('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits')
print('Before:')
print(data_before.info())

data_after = fits.open('pipeline_outputs_directory/WASP43_NIS_SOSS-seg002_CLEAR_dqinitstep.fits')
print('\nAfter:')
print(data_after.info())
```

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

```
# Create output directory:
os.mkdir('pipeline_outputs_directory')

# Run step:
dq_results = calwebb_detector1.dq_init_step.DQInitStep.call('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits',
                                                            output_dir='pipeline_outputs_directory',
                                                            save_results=True)
```

Stage (1 here)

Step

Note that in your first run of the JWST pipeline, you will probably see much more outputs than the ones above. That's normal.

All right! The `DQInitStep` has went through successfully. What did it do? Let's compare the before and after:

```
data_before = fits.open('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits')
print('Before:')
print(data_before.info())

data_after = fits.open('pipeline_outputs_directory/WASP43_NIS_SOSS-seg002_CLEAR_dqinitstep.fits')
print('\nAfter:')
print(data_after.info())
```

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

```
# Create output directory:
os.mkdir('pipeline_outputs_directory')

# Run step:
dq_results = calwebb_detector1.dq_init_step.DQInitStep.call('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits',
                                                            output_dir='pipeline_outputs_directory',
                                                            save_results=True)
```

Input (filename in this case)

Note that in your first run of the JWST pipeline, you will probably see much more outputs than the ones above. That's normal.

All right! The `DQInitStep` has went through successfully. What did it do? Let's compare the before and after:

```
data_before = fits.open('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits')
print('Before:')
print(data_before.info())

data_after = fits.open('pipeline_outputs_directory/WASP43_NIS_SOSS-seg002_CLEAR_dqinitstep.fits')
print('\nAfter:')
print(data_after.info())
```

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

```
# Create output directory:
os.mkdir('pipeline_outputs_directory')

# Run step:
dq_results = calwebb_detector1.dq_init_step.DQInitStep.call('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits',
                                                            output_dir='pipeline_outputs_directory',
                                                            save_results=True)
```

Input (filename in this case)

Pipeline, pls save result in a fits file

Output directory

Note that in your first run of the JWST pipeline, you will probably see much more outputs than the ones above. That's normal.

All right! The `DQInitStep` has went through successfully. What did it do? Let's compare the before and after:

```
data_before = fits.open('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits')
print('Before:')
print(data_before.info())

data_after = fits.open('pipeline_outputs_directory/WASP43_NIS_SOSS-seg002_CLEAR_dqinitstep.fits')
print('\nAfter:')
print(data_after.info())
```

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Let’s use the ***uncal.fits NIRISS** simulated data:

```
# Create output directory:
os.mkdir('pipeline_outputs_directory')

# Run step:
dq_results = calwebb_detector1.dq_init_step.DQInitStep.call('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits',
                                                            output_dir='pipeline_outputs_directory',
                                                            save_results=True)
```

```
2021-06-23 00:49:14,438 - stpipe.DQInitStep - INFO - DQInitStep instance created.
2021-06-23 00:49:15,967 - stpipe.DQInitStep - INFO - Step DQInitStep running with args ('WASP43_NIS_SOSS-seg002_CLEAR
_uncal.fits',).
2021-06-23 00:49:15,969 - stpipe.DQInitStep - INFO - Step DQInitStep parameters are: {'pre_hooks': [], 'post_hooks':
[], 'output_file': None, 'output_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': Fals
e, 'output_use_index': True, 'save_results': True, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_
dir': ''}
2021-06-23 00:49:18,632 - stpipe.DQInitStep - INFO - Using MASK reference file $HOME/crds_cache/references/jwst/niris
s/jwst_niriss_mask_0014.fits
2021-06-23 00:49:23,521 - stpipe.DQInitStep - INFO - Extracting mask subarray to match science data
2021-06-23 00:49:35,668 - stpipe.DQInitStep - INFO - Saved model in pipeline_outputs_directory/WASP43_NIS_SOSS-seg002
_CLEAR_dqinitstep.fits
2021-06-23 00:49:35,673 - stpipe.DQInitStep - INFO - Step DQInitStep done
```

Output is *_dqinitstep.fits

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Important: no need to give fits files as inputs. Can be results from previous steps. For instance:

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Important: no need to give fits files as inputs. Can be results from previous steps. For instance:

```
saturation_results = calwebb_detector1.saturation_step.SaturationStep.call(dq_results,
                                                                           output_dir='pipeline_outputs_directory',
                                                                           save_results=True)
```

```
2021-06-22 16:50:21,357 - stpipe.SaturationStep - INFO - SaturationStep instance created.
2021-06-22 16:50:21,660 - stpipe.SaturationStep - INFO - Step SaturationStep running with args (<RampModel(107, 6, 25
6, 2048) from WASP43_NIS_SOSS-seg002_CLEAR_dqinitstep.fits>,).
2021-06-22 16:50:21,661 - stpipe.SaturationStep - INFO - Step SaturationStep parameters are: {'pre_hooks': [], 'post_
hooks': [], 'output_file': None, 'output_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_mode
l': False, 'output_use_index': True, 'save_results': True, 'skip': False, 'suffix': None, 'search_output_file': True,
'input_dir': ''}
2021-06-22 16:50:21,704 - stpipe.SaturationStep - INFO - Using SATURATION reference file $HOME/crds_cache/references/
jwst/niriss/jwst_niriss_saturation_0011.fits
2021-06-22 16:50:21,819 - stpipe.SaturationStep - WARNING - Keyword CDP_REP_LIMITS does not correspond to an existing
DQ mnemonic, so will be ignored
2021-06-22 16:50:25,954 - stpipe.SaturationStep - INFO - Extracting reference file subarray to match science data
2021-06-22 16:50:28,422 - stpipe.SaturationStep - INFO - Detected 2486 saturated pixels
2021-06-22 16:50:28,690 - stpipe.SaturationStep - INFO - Detected 0 A/D floor pixels
2021-06-22 16:50:37,683 - stpipe.SaturationStep - INFO - Saved model in pipeline_outputs_directory/WASP43_NIS_SOSS-se
g002_CLEAR_saturationstep.fits
2021-06-22 16:50:37,684 - stpipe.SaturationStep - INFO - Step SaturationStep done
```

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline “hello world”: running a single step

Important: no need to give fits files as inputs. Can be results from previous steps. For instance:

```
saturation_results = calwebb_detector1.saturation_step.SaturationStep.call(dq_results,
                                                                            output_dir='pipeline_outputs_directory',
                                                                            save_results=True)
```

```
2021-06-22 16:50:21,357 - stpipe.SaturationStep - INFO - SaturationStep instance created.
2021-06-22 16:50:21,660 - stpipe.SaturationStep - INFO - Step SaturationStep running with args (<RampModel(107, 6, 25
6, 2048) from WASP43_NIS_SOSS-seg002_CLEAR_dqinitstep.fits>,).
2021-06-22 16:50:21,661 - stpipe.SaturationStep - INFO - Step SaturationStep parameters are: {'pre_hooks': [], 'post_
hooks': [], 'output_file': None, 'output_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_mode
l': False, 'output_use_index': True, 'save_results': True, 'skip': False, 'suffix': None, 'search_output_file': True,
'input_dir': ''}
2021-06-22 16:50:21,704 - stpipe.SaturationStep - INFO - Using SATURATION reference file $HOME/crds_cache/references/
jwst/niriss/jwst_niriss_saturation_0011.fits
2021-06-22 16:50:21,819 - stpipe.SaturationStep - WARNING - Keyword CDP_REP_LIMITS does not correspond to an existing
DQ mnemonic, so will be ignored
2021-06-22 16:50:25,954 - stpipe.SaturationStep - INFO - Extracting reference file subarray to match science data
2021-06-22 16:50:28,422 - stpipe.SaturationStep - INFO - Detected 2486 saturated pixels
2021-06-22 16:50:28,690 - stpipe.SaturationStep - INFO - Detected 0 A/D floor pixels
2021-06-22 16:50:37,683 - stpipe.SaturationStep - INFO - Saved model in pipeline_outputs_directory/WASP43_NIS_SOSS-se
g002_CLEAR_saturationstep.fits
2021-06-22 16:50:37,684 - stpipe.SaturationStep - INFO - Step SaturationStep done
```

THE JWST DATA REDUCTION PIPELINE

JWST pipeline advanced: running many steps

This allows you to do neat stuff:

THE JWST DATA REDUCTION PIPELINE

JWST pipeline advanced: running many steps

This allows you to do neat stuff:

THE JWST DATA REDUCTION PIPELINE

JWST pipeline advanced: running many steps

This allows you to do neat stuff:

Near-IR			MIRI		
Step	**Non-TSO**	**TSO**	**Step**	**Non-TSO**	**TSO**
group_scale	✓	✓	group_scale	✓	✓
dq_init	✓	✓	dq_init	✓	✓
saturation	✓	✓	saturation	✓	✓
ipc [1]			ipc		
superbias	✓	✓	firstframe	✓	
refpix	✓	✓	lastframe	✓	
linearity	✓	✓	linearity	✓	✓
persistence [2]	✓		rscd	✓	
dark_current	✓	✓	dark_current	✓	✓
			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

THE JWST DATA REDUCTION PIPELINE

JWST pipeline advanced: running many steps

This allows you to do neat stuff:

Near-IR			MIRI		
Step	**Non-TSO**	**TSO**	**Step**	**Non-TSO**	**TSO**
group_scale	✓	✓	group_scale	✓	✓
dq_init	✓	✓	dq_init	✓	✓
saturation	✓	✓	saturation	✓	✓
ipc [1]			ipc		
superbias	✓	✓	firstframe	✓	
refpix	✓	✓	lastframe	✓	
linearity	✓	✓	linearity	✓	✓
persistence [2]	✓		rscd	✓	
dark_current	✓	✓	dark_current	✓	✓
			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

THE JWST DATA REDUCTION PIPELINE

JWST pipeline advanced: running many steps

This allows you to do neat stuff:

Near-IR			MIRI		
Step	**Non-TSO**	**TSO**	**Step**	**Non-TSO**	**TSO**
group_scale	✓	✓	group_scale	✓	✓
dq_init	✓	✓	dq_init	✓	✓
saturation	✓	✓	saturation	✓	✓
ipc [1]			ipc		
superbias	✓	✓	firstframe	✓	
refpix	✓	✓	lastframe	✓	
linearity	✓	✓	linearity	✓	✓
persistence [2]	✓		rscd	✓	
dark_current	✓	✓	dark_current	✓	✓
			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

```
# Define steps we want to run our data through:
steps = [calwebb_detector1.dq_init_step.DQInitStep.call,
         calwebb_detector1.saturation_step.SaturationStep.call,
         calwebb_detector1.superbias_step.SuperBiasStep.call]

# Define input on the first iteration of the loop below:
calwebb_input = 'WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits'

for i in range(len(steps)):

    # Developers would not be very happy with me using several times the same variable as input/output,
    # but we are astronomers --- note calwebb_input in the first iteration is a string, then is an object:
    calwebb_input = steps[i](calwebb_input,
                             output_dir='pipeline_outputs_directory',
                             output_file='enmasse')
```

```
2021-06-23 01:16:59,294 - stpipe.DQInitStep - INFO - DQInitStep instance created.
2021-06-23 01:17:00,758 - stpipe.DQInitStep - INFO - Step DQInitStep running with args ('WASP43_NIS_SOSS-seg002_CLEAR
_uncal.fits',).
2021-06-23 01:17:00,759 - stpipe.DQInitStep - INFO - Step DQInitStep parameters are: {'pre_hooks': [], 'post_hooks':
[], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse', 'output_
dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': True, 'save
_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:04,171 - stpipe.DQInitStep - INFO - Using MASK reference file $HOME/crds_cache/references/jwst/niris
s/jwst_niriss_mask_0014.fits
2021-06-23 01:17:11,149 - stpipe.DQInitStep - INFO - Extracting mask subarray to match science data
2021-06-23 01:17:26,577 - stpipe.DQInitStep - INFO - Saved model in pipeline_outputs_directory/enmasse_dqinitstep.fit
s
2021-06-23 01:17:26,585 - stpipe.DQInitStep - INFO - Step DQInitStep done
2021-06-23 01:17:26,630 - stpipe.SaturationStep - INFO - SaturationStep instance created.
2021-06-23 01:17:27,800 - stpipe.SaturationStep - INFO - Step SaturationStep running with args (<RampModel(107, 6, 25
6, 2048) from enmasse_dqinitstep.fits>,).
2021-06-23 01:17:27,804 - stpipe.SaturationStep - INFO - Step SaturationStep parameters are: {'pre_hooks': [], 'post_
hooks': [], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse',
'output_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': Tru
e, 'save_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:27,851 - stpipe.SaturationStep - INFO - Using SATURATION reference file $HOME/crds_cache/references/
jwst/niriss/jwst_niriss_saturation_0011.fits
2021-06-23 01:17:27,989 - stpipe.SaturationStep - WARNING - Keyword CDP_REP_LIMITS does not correspond to an existing
DQ mnemonic, so will be ignored
2021-06-23 01:17:31,660 - stpipe.SaturationStep - INFO - Extracting reference file subarray to match science data
2021-06-23 01:17:34,589 - stpipe.SaturationStep - INFO - Detected 2486 saturated pixels
```

Neat, huh? Just for fun, let's see how our superbias-corrected groups look like:

```
uncal_data = fits.open('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits')
uncal_first_group_first_integration = uncal_data['SCI'].data[0,0,:,:]

corrected_data = fits.open('pipeline_outputs_directory/enmasse_superbiasstep.fits')
cal_first_group_first_integration = corrected_data['SCI'].data[0,0,:,:]
```

```
plt.figure(figsize=(15,5))

plt.subplot(211)
plt.title('uncal')
im = plt.imshow(uncal_first_group_first_integration)
im.set_clim(10000,15000)
```

THE JWST DATA REDUCTION PIPELINE

JWST pipeline advanced: running many steps

This allows you to do neat stuff:

Near-IR			MIRI		
Step	**Non-TSO**	**TSO**	**Step**	**Non-TSO**	**TSO**
group_scale	✓	✓	group_scale	✓	✓
dq_init	✓	✓	dq_init	✓	✓
saturation	✓	✓	saturation	✓	✓
ipc [1]			ipc		
superbias	✓	✓	firstframe	✓	
refpix	✓	✓	lastframe	✓	
linearity	✓	✓	linearity	✓	✓
persistence [2]	✓		rscd	✓	
dark_current	✓	✓	dark_current	✓	✓
			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

```
# Define steps we want to run our data through:
steps = [calwebb_detector1.dq_init_step.DQInitStep.call,
         calwebb_detector1.saturation_step.SaturationStep.call,
         calwebb_detector1.superbias_step.SuperBiasStep.call]

# Define input on the first iteration of the loop below:
calwebb_input = 'WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits'

for i in range(len(steps)):

    # Developers would not be very happy with me using several times the same variable as input/output,
    # but we are astronomers --- note calwebb_input in the first iteration is a string, then is an object:
    calwebb_input = steps[i](calwebb_input,
                             output_dir='pipeline_outputs_directory',
                             output_file='enmasse')
```

```
2021-06-23 01:16:59,294 - stpipe.DQInitStep - INFO - DQInitStep instance created.
2021-06-23 01:17:00,758 - stpipe.DQInitStep - INFO - Step DQInitStep running with args ('WASP43_NIS_SOSS-seg002_CLEAR
_uncal.fits',).
2021-06-23 01:17:00,759 - stpipe.DQInitStep - INFO - Step DQInitStep parameters are: {'pre_hooks': [], 'post_hooks':
[], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse', 'output_
dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': True, 'save
_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:04,171 - stpipe.DQInitStep - INFO - Using MASK reference file $HOME/crds_cache/references/jwst/niris
s/jwst_niriss_mask_0014.fits
2021-06-23 01:17:11,149 - stpipe.DQInitStep - INFO - Extracting mask subarray to match science data
2021-06-23 01:17:26,577 - stpipe.DQInitStep - INFO - Saved model in pipeline_outputs_directory/enmasse_dqinitstep.fit
s
2021-06-23 01:17:26,585 - stpipe.DQInitStep - INFO - Step DQInitStep done
2021-06-23 01:17:26,630 - stpipe.SaturationStep - INFO - SaturationStep instance created.
2021-06-23 01:17:27,800 - stpipe.SaturationStep - INFO - Step SaturationStep running with args (<RampModel(107, 6, 25
6, 2048) from enmasse_dqinitstep.fits>,).
2021-06-23 01:17:27,804 - stpipe.SaturationStep - INFO - Step SaturationStep parameters are: {'pre_hooks': [], 'post_
hooks': [], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse',
'output_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': Tru
e, 'save_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:27,851 - stpipe.SaturationStep - INFO - Using SATURATION reference file $HOME/crds_cache/references/
jwst/niriss/jwst_niriss_saturation_0011.fits
2021-06-23 01:17:27,989 - stpipe.SaturationStep - WARNING - Keyword CDP_REP_LIMITS does not correspond to an existing
DQ mnemonic, so will be ignored
2021-06-23 01:17:31,660 - stpipe.SaturationStep - INFO - Extracting reference file subarray to match science data
2021-06-23 01:17:34,589 - stpipe.SaturationStep - INFO - Detected 2486 saturated pixels
```

Neat, huh? Just for fun, let's see how our superbias-corrected groups look like:

```
uncal_data = fits.open('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits')
uncal_first_group_first_integration = uncal_data['SCI'].data[0,0,:,:]

corrected_data = fits.open('pipeline_outputs_directory/enmasse_superbiasstep.fits')
cal_first_group_first_integration = corrected_data['SCI'].data[0,0,:,:]
```

```
plt.figure(figsize=(15,5))

plt.subplot(211)
plt.title('uncal')
im = plt.imshow(uncal_first_group_first_integration)
im.set_clim(10000,15000)
```

THE JWST DATA REDUCTION PIPELINE

JWST pipeline advanced: running many steps

This allows you to do neat stuff:

Near-IR			MIRI		
Step	**Non-TSO**	**TSO**	**Step**	**Non-TSO**	**TSO**
group_scale	✓	✓	group_scale	✓	✓
dq_init	✓	✓	dq_init	✓	✓
saturation	✓	✓	saturation	✓	✓
ipc [1]			ipc		
superbias	✓	✓	firstframe	✓	
refpix	✓	✓	lastframe	✓	
linearity	✓	✓	linearity	✓	✓
persistence [2]	✓		rscd	✓	
dark_current	✓	✓	dark_current	✓	✓
			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

```
# Define steps we want to run our data through:
steps = [calwebb_detector1.dq_init_step.DQInitStep.call,
         calwebb_detector1.saturation_step.SaturationStep.call,
         calwebb_detector1.superbias_step.SuperBiasStep.call]

# Define input on the first iteration of the loop below:
calwebb_input = 'WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits'

for i in range(len(steps)):

    # Developers would not be very happy with me using several times the same variable as input/output,
    # but we are astronomers --- note calwebb_input in the first iteration is a string, then is an object:
    calwebb_input = steps[i](calwebb_input,
                             output_dir='pipeline_outputs_directory',
                             output_file='enmasse')
```

```
2021-06-23 01:16:59,294 - stpipe.DQInitStep - INFO - DQInitStep instance created.
2021-06-23 01:17:00,758 - stpipe.DQInitStep - INFO - Step DQInitStep running with args ('WASP43_NIS_SOSS-seg002_CLEAR
_uncal.fits',).
2021-06-23 01:17:00,759 - stpipe.DQInitStep - INFO - Step DQInitStep parameters are: {'pre_hooks': [], 'post_hooks':
[], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse', 'output_
dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': True, 'save
_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:04,171 - stpipe.DQInitStep - INFO - Using MASK reference file $HOME/crds_cache/references/jwst/niris
s/jwst_niriss_mask_0014.fits
2021-06-23 01:17:11,149 - stpipe.DQInitStep - INFO - Extracting mask subarray to match science data
2021-06-23 01:17:26,577 - stpipe.DQInitStep - INFO - Saved model in pipeline_outputs_directory/enmasse_dqinitstep.fit
s
2021-06-23 01:17:26,585 - stpipe.DQInitStep - INFO - Step DQInitStep done
2021-06-23 01:17:26,630 - stpipe.SaturationStep - INFO - SaturationStep instance created.
2021-06-23 01:17:27,800 - stpipe.SaturationStep - INFO - Step SaturationStep running with args (<RampModel(107, 6, 25
6, 2048) from enmasse_dqinitstep.fits>,).
2021-06-23 01:17:27,804 - stpipe.SaturationStep - INFO - Step SaturationStep parameters are: {'pre_hooks': [], 'post_
hooks': [], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse',
'output_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': Tru
e, 'save_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:27,851 - stpipe.SaturationStep - INFO - Using SATURATION reference file $HOME/crds_cache/references/
jwst/niriss/jwst_niriss_saturation_0011.fits
2021-06-23 01:17:27,989 - stpipe.SaturationStep - WARNING - Keyword CDP_REP_LIMITS does not correspond to an existing
DQ mnemonic, so will be ignored
2021-06-23 01:17:31,660 - stpipe.SaturationStep - INFO - Extracting reference file subarray to match science data
2021-06-23 01:17:34,589 - stpipe.SaturationStep - INFO - Detected 2486 saturated pixels
```

Neat, huh? Just for fun, let's see how our superbias-corrected groups look like:

```
uncal_data = fits.open('WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits')
uncal_first_group_first_integration = uncal_data['SCI'].data[0,0,:,:]

corrected_data = fits.open('pipeline_outputs_directory/enmasse_superbiasstep.fits')
cal_first_group_first_integration = corrected_data['SCI'].data[0,0,:,:]
```

```
plt.figure(figsize=(15,5))

plt.subplot(211)
plt.title('uncal')
im = plt.imshow(uncal_first_group_first_integration)
im.set_clim(10000,15000)
```

THE JWST DATA REDUCTION PIPELINE

JWST pipeline advanced: running many steps

This allows you to do neat stuff:

Near-IR			MIRI		
Step	**Non-TSO**	**TSO**	**Step**	**Non-TSO**	**TSO**
group_scale	✓	✓	group_scale	✓	✓
dq_init	✓	✓	dq_init	✓	✓
saturation	✓	✓	saturation	✓	✓
ipc [1]			ipc		
superbias	✓	✓	firstframe	✓	
refpix	✓	✓	lastframe	✓	
linearity	✓	✓	linearity	✓	✓
persistence [2]	✓		rscd	✓	
dark_current	✓	✓	dark_current	✓	✓
			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

```
# Define steps we want to run our data through:
steps = [calwebb_detector1.dq_init_step.DQInitStep.call,
         calwebb_detector1.saturation_step.SaturationStep.call,
         calwebb_detector1.superbias_step.SuperBiasStep.call]

# Define input on the first iteration of the loop below:
calwebb_input = 'WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits'

for i in range(len(steps)):

    # Developers would not be very happy with me using several times the same variable as input/output,
    # but we are astronomers --- note calwebb_input in the first iteration is a string, then is an object:
    calwebb_input = steps[i](calwebb_input,
                             output_dir='pipeline_outputs_directory',
                             output_file='enmasse')
```

```
2021-06-23 01:16:59,294 - stpipe.DQInitStep - INFO - DQInitStep instance created.
2021-06-23 01:17:00,758 - stpipe.DQInitStep - INFO - Step DQInitStep running with args ('WASP43_NIS_SOSS-seg002_CLEAR
_uncal.fits',).
2021-06-23 01:17:00,759 - stpipe.DQInitStep - INFO - Step DQInitStep parameters are: {'pre_hooks': [], 'post_hooks':
[], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse', 'output_
dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': True, 'save
_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:04,171 - stpipe.DQInitStep - INFO - Using MASK reference file $HOME/crds_cache/references/jwst/niris
s/jwst_niriss_mask_0014.fits
2021-06-23 01:17:11,149 - stpipe.DQInitStep - INFO - Extracting mask subarray to match science data
2021-06-23 01:17:26,577 - stpipe.DQInitStep - INFO - Saved model in pipeline_outputs_directory/enmasse_dqinitstep.fit
s
2021-06-23 01:17:26,585 - stpipe.DQInitStep - INFO - Step DQInitStep done
2021-06-23 01:17:26,630 - stpipe.SaturationStep - INFO - SaturationStep instance created.
2021-06-23 01:17:27,800 - stpipe.SaturationStep - INFO - Step SaturationStep running with args (<RampModel(107, 6, 25
6, 2048) from enmasse_dqinitstep.fits>,).
2021-06-23 01:17:27,804 - stpipe.SaturationStep - INFO - Step SaturationStep parameters are: {'pre_hooks': [], 'post_
hooks': [], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse',
'output_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': Tru
e, 'save_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:27,851 - stpipe.SaturationStep - INFO - Using SATURATION reference file $HOME/crds_cache/references/
jwst/niriss/jwst_niriss_saturation_0011.fits
2021-06-23 01:17:27,989 - stpipe.SaturationStep - WARNING - Keyword CDP_REP_LIMITS does not correspond to an existing
DQ mnemonic, so will be ignored
2021-06-23 01:17:31,660 - stpipe.SaturationStep - INFO - Extracting reference file subarray to match science data
2021-06-23 01:17:34,589 - stpipe.SaturationStep - INFO - Detected 2486 saturated pixels
2021-06-23 01:17:34,934 - stpipe.SaturationStep - INFO - Detected 0 A/D floor pixels
2021-06-23 01:17:41,741 - stpipe.SaturationStep - INFO - Saved model in pipeline_outputs_directory/enmasse_saturation
step.fits
2021-06-23 01:17:41,742 - stpipe.SaturationStep - INFO - Step SaturationStep done
2021-06-23 01:17:41,769 - stpipe.SuperBiasStep - INFO - SuperBiasStep instance created.
2021-06-23 01:17:42,368 - stpipe.SuperBiasStep - INFO - Step SuperBiasStep running with args (<RampModel(107, 6, 256,
2048) from enmasse_saturationstep.fits>,).
2021-06-23 01:17:42,370 - stpipe.SuperBiasStep - INFO - Step SuperBiasStep parameters are: {'pre_hooks': [], 'post_ho
oks': [], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse', 'o
utput_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': True,
'save_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:42,412 - stpipe.SuperBiasStep - INFO - Using SUPERBIAS reference file $HOME/crds_cache/references/jw
st/niriss/jwst_niriss_superbias_0120.fits
```

Neat, huh? Just for fun, let's see how our superbias-corrected groups look like:

NIRISS SOSS GR700XD/CLEAR simulated time-series observations

Contributors: Nestor Espinoza (nespinoza@stsci.edu); Joseph Filippazzo (jfilippazzo@stsci.edu), Erin M. May (erin.may@jhuapl.edu) and Kristin Sotzen (kristin.sotzen@jhuapl.edu)

About

This directory contains simulated JWST NIRISS/SOSS time-series observations, made with the latest version of `mirage`. It consists of a 320-integration, 6 groups per integration exposure covering a transit observation of WASP-43b. Detector effects such as 1/f, "weird" pixels (e.g., RC, dead pixels, etc.) and others have been included, as well as cosmic-rays, by `mirage` itself.

Products

There are three sets of products, all on different folders:

1. *The* `Uncalibrated` *products* are the raw, uncalibrated frames similar to the ones JWST will obtain. These are divided in two files -segments- which need to be calibrated separately. The first segment contains most of the integrations.
2. *The* `Stage 1 Outputs` *products* contain the so-called "rate-ints" products of the JWST Calibration Pipeline – that is, the ADUs per seconds at each integration for each pixel. These are obtained by fitting lines to all groups in each integrations (for details, see the description of the algorithm in the JWST Calibration Pipeline readthedocs).
3. *The* `Stage 2 Outputs` *products* contain the resulting products after running the `assign_wcs` step of Stage 2 of the pipeline. This is useful for extracting the wavelength solution from these products; no extracted spectra are provided, as the currently implemented algorithms for spectral extraction in the JWST Calibration Pipeline are inadequate for the curve nature of NIRISS/SOSS spectra.

Reference files

When running the JWST Calibration pipeline to reduce the uncalibrated products, the pipeline will choose the latest reference files to perform corrections from the JWST Calibration Reference Data System (CRDS; https://jwst-crds.stsci.edu/). However, these might not match the reference files used by the simulations and thus give rise to spurious systematics due to this mismatch. We list the reference files used to perform the simulations below:

- Reference file for the SuperBias step: `jwst_niriss_superbias_0017.fits`. The CRDS reference file can be overriden in the SuperBias step by doing, e.g., `calwebb_detector1.superbias_step.SuperBiasStep.call(..., superbias = "jwst_niriss_superbias_0017.fits")`.
- Reference file for the linearity step: `jwst_niriss_linearity_0013.fits`. Similarly, CRDS file can be override by doing, e.g., `calwebb_detector1.linearity_step.LinearityStep.call(..., override_linearity = "jwst_niriss_linearity_0013.fits")`.

			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	✓
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

```
2021-06-23 01:17:34,934 - stpipe.SaturationStep - INFO - Detected 0 A/D floor pixels
2021-06-23 01:17:41,741 - stpipe.SaturationStep - INFO - Saved model in pipeline_outputs_directory/enmasse_saturation
step.fits
2021-06-23 01:17:41,742 - stpipe.SaturationStep - INFO - Step SaturationStep done
2021-06-23 01:17:41,769 - stpipe.SuperBiasStep - INFO - SuperBiasStep instance created.
2021-06-23 01:17:42,368 - stpipe.SuperBiasStep - INFO - Step SuperBiasStep running with args (<RampModel(107, 6, 256,
2048) from enmasse_saturationstep.fits>,).
2021-06-23 01:17:42,370 - stpipe.SuperBiasStep - INFO - Step SuperBiasStep parameters are: {'pre_hooks': [], 'post_ho
oks': [], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse', 'o
utput_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': True,
'save_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:42,412 - stpipe.SuperBiasStep - INFO - Using SUPERBIAS reference file $HOME/crds_cache/references/jw
st/niriss/jwst_niriss_superbias_0120.fits
```

Neat, huh? Just for fun, let's see how our superbias-corrected groups look like:

NIRISS SOSS GR700XD/CLEAR simulated time-series observations

Contributors: Nestor Espinoza (nespinoza@stsci.edu); Joseph Filippazzo (jfilippazzo@stsci.edu), Erin M. May (erin.may@jhuapl.edu) and Kristin Sotzen (kristin.sotzen@jhuapl.edu)

About

This directory contains simulated JWST NIRISS/SOSS time-series observations, made with the latest version of `mirage`. It consists of a 320-integration, 6 groups per integration exposure covering a transit observation of WASP-43b. Detector effects such as 1/f, "weird" pixels (e.g., RC, dead pixels, etc.) and others have been included, as well as cosmic-rays, by `mirage` itself.

Products

There are three sets of products, all on different folders:

1. *The* `Uncalibrated` *products* are the raw, uncalibrated frames similar to the ones JWST will obtain. These are divided in two files -segments- which need to be calibrated separately. The first segment contains most of the integrations.
2. *The* `Stage 1 Outputs` *products* contain the so-called "rate-ints" products of the JWST Calibration Pipeline – that is, the ADUs per seconds at each integration for each pixel. These are obtained by fitting lines to all groups in each integrations (for details, see the description of the algorithm in the JWST Calibration Pipeline readthedocs).
3. *The* `Stage 2 Outputs` *products* contain the resulting products after running the `assign_wcs` step of Stage 2 of the pipeline. This is useful for extracting the wavelength solution from these products; no extracted spectra are provided, as the currently implemented algorithms for spectral extraction in the JWST Calibration Pipeline are inadequate for the curve nature of NIRISS/SOSS spectra.

Reference files

When running the JWST Calibration pipeline to reduce the uncalibrated products, the pipeline will choose the latest reference files to perform corrections from the JWST Calibration Reference Data System (CRDS; https://jwst-crds.stsci.edu/). However, these might not match the reference files used by the simulations and thus give rise to spurious systematics due to this mismatch. We list the reference files used to perform the simulations below:

- Reference file for the SuperBias step: `jwst_niriss_superbias_0017.fits`. The CRDS reference file can be overriden in the SuperBias step by doing, e.g., `calwebb_detector1.superbias_step.SuperBiasStep.call(..., superbias = "jwst_niriss_superbias_0017.fits")`.
- Reference file for the linearity step: `jwst_niriss_linearity_0013.fits`. Similarly, CRDS file can be override by doing, e.g., `calwebb_detector1.linearity_step.LinearityStep.call(..., override_linearity = "jwst_niriss_linearity_0013.fits")`.

			refpix	✓	✓
jump	✓	✓	jump	✓	✓
ramp_fitting	✓	✓	ramp_fitting	✓	
gain_scale	✓	✓	gain_scale	✓	✓

Stage 1 steps

```
2021-06-23 01:17:34,934 - stpipe.SaturationStep - INFO - Detected 0 A/D floor pixels
2021-06-23 01:17:41,741 - stpipe.SaturationStep - INFO - Saved model in pipeline_outputs_directory/enmasse_saturation
step.fits
2021-06-23 01:17:41,742 - stpipe.SaturationStep - INFO - Step SaturationStep done
2021-06-23 01:17:41,769 - stpipe.SuperBiasStep - INFO - SuperBiasStep instance created.
2021-06-23 01:17:42,368 - stpipe.SuperBiasStep - INFO - Step SuperBiasStep running with args (<RampModel(107, 6, 256,
2048) from enmasse_saturationstep.fits>,).
2021-06-23 01:17:42,370 - stpipe.SuperBiasStep - INFO - Step SuperBiasStep parameters are: {'pre_hooks': [], 'post_ho
oks': [], 'output_file': '/Users/nespinoza/Documents/niriss-commissioning/nis_comm/nis_034/datareduction/enmasse', 'o
utput_dir': 'pipeline_outputs_directory', 'output_ext': '.fits', 'output_use_model': False, 'output_use_index': True,
'save_results': False, 'skip': False, 'suffix': None, 'search_output_file': True, 'input_dir': ''}
2021-06-23 01:17:42,412 - stpipe.SuperBiasStep - INFO - Using SUPERBIAS reference file $HOME/crds_cache/references/jw
st/niriss/jwst_niriss_superbias_0120.fits
```

Neat, huh? Just for fun, let's see how our superbias-corrected groups look like:

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline: overriding reference files

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline: overriding reference files

This is easy to fix:

```
from astropy.utils.data import download_file

file_path = download_file('https://jwst-crds.stsci.edu/unchecked_get/references/jwst/jwst_niriss_superbias_0017.fits')
os.rename(file_path, 'jwst_niriss_superbias_0017.fits')
```

And let's now run the `superbias` step with this reference file:

```
sbias_corrected = calwebb_detector1.superbias_step.SuperBiasStep.call('pipeline_outputs_directory/enmasse_saturationst
                                                      output_dir='pipeline_outputs_directory',
                                                      output_file='good_reference_file',
                                                      override_superbias = "jwst_niriss_superbias_0017.fits")
```

All right! Let's see how we did:

THE JWST DATA REDUCTION PIPELINE

The JWST pipeline: overriding reference files

This is easy to fix:

```
from astropy.utils.data import download_file

file_path = download_file('https://jwst-crds.stsci.edu/unchecked_get/references/jwst/jwst_niriss_superbias_0017.fits')
os.rename(file_path, 'jwst_niriss_superbias_0017.fits')
```

And let's now run the `superbias` step with this reference file:

```
sbias_corrected = calwebb_detector1.superbias_step.SuperBiasStep.call('pipeline_outputs_directory/enmasse_saturationst
                                                       output_dir='pipeline_outputs_directory',
                                                       override_superbias = "jwst_niriss_superbias_0017.fits"
```

All right! Let's see how we did:

ALL THIS AND MORE

In [1]:
```
# Need to set these enviromental variables for this notebook to work properly:
%set_env CRDS_PATH $HOME/crds_cache
%set_env CRDS_SERVER_URL https://jwst-crds.stsci.edu

# Import all useful libraries:
import os
import numpy as np
import matplotlib.pyplot as plt

from astropy.io import fits

from jwst.pipeline import calwebb_detector1
from jwst.pipeline import calwebb_spec2
```

```
env: CRDS_PATH=$HOME/crds_cache
env: CRDS_SERVER_URL=https://jwst-crds.stsci.edu
```

```
2021-06-22 16:49:45,221 - stpipe - WARNING - /Users/nespinoza/miniconda2/envs/newen/lib/python3.7/site-packages/photutils/detection/findstars.py:35: AstropyDeprecationWarning: _StarFinderKernel was moved to the photutils.detection._utils module. Please update your import statement.
  AstropyDeprecationWarning)
```

Interacting with the JWST pipeline --- an Introduction

Author: Néstor Espinoza (nespinoza@stsci.edu) | **Date of last update:** June 22, 2021

1. Introduction

In the following notebook, we will perform some data analysis using the JWST pipeline and our own scripts, in order to showcase how flexible and modular the pipeline is. Above, we have imported the `calwebb_detector1` and `calwebb_spec2` classes, which contain functions from Stage 1 (detector processing) and Stage 2 (spectroscopic processing), respectively.

To execute this notebook, you will need to download the `Uncalibrated` data from NIRISS/SOSS here. In particular, let's use the `WASP43_NIS_SOSS-seg002_CLEAR_uncal.fits` file --- the second segment of data. Save it wherever you are running this notebook, and the cells below should run without issues.

2. Running Stage 1 steps individually:

SUMMARY

- **JWST PIPELINE IS MODULAR.** You can basically run any step from any stage on a given fits file as long as the headers are compliant with what the pipeline expects. This allows you to build your own pipeline using the steps you like!

SUMMARY

- **JWST PIPELINE IS MODULAR.** You can basically run any step from any stage on a given fits file as long as the headers are compliant with what the pipeline expects. This allows you to build your own pipeline using the steps you like!

- **PAY ATTENTION TO PIPELINE OUTPUTS.** The outputs of the pipeline are very useful to debug and/or understand what is going on behind the scenes (e.g., what reference file is being used, which parameters the pipeline is using).

SUMMARY

- **JWST PIPELINE IS MODULAR.** You can basically run any step from any stage on a given fits file as long as the headers are compliant with what the pipeline expects. This allows you to build your own pipeline using the steps you like!

- **PAY ATTENTION TO PIPELINE OUTPUTS.** The outputs of the pipeline are very useful to debug and/or understand what is going on behind the scenes (e.g., what reference file is being used, which parameters the pipeline is using).

- **THE PIPELINE ALLOWS TO MODIFY PARAMETERS AT WILL.** Be it reference files, or parameters that define a given step, all of them can be modified from the default ones.