Pypelt Documentation

Release 0.11.1dev

Cooke, Prochaska, and Associates

Sep 29, 2019

Contents

PypeIt is a Python based data reduction pipeline (DRP) written oringinally for echelle spectroscopy and since expanded to low-resolution spectrometers. This documentation details the code, how to run it, and what it produces.

CHAPTER 1

Getting Started

1.1 Installing PypeIt

This document will describe how to install PypeIt.

1.1.1 Installing Dependencies

Though we have tried to keep the number of dependencies low, there are a few packages that need to be installed (various python packages and linetools).

In general, we recommend that you use Anaconda for the majority of these installations.

Detailed installation instructions are presented below:

Python Dependencies

PypeIt depends on the following list of Python packages.

We recommend that you use Anaconda to install and/or update these packages.

- python version 3.6 or later
- numpy version 1.15.4 or later
- astropy version 3.1 or later
- scipy version 1.1 or later
- matplotlib version 3.0 or later
- numba version 0.39.0 or later (optional - some speed ups, inc. wavecal)
- PyQT5 version 5 (needed for linetools)
- h5py version 2.7 (for data I/O)
- yaml – You may need to install pyyaml

- configobj – version 5.0.6 or later
- scikit-learn – version 0.20 or later
- IPython – version 7.2.0 or later

If you are using Anaconda, you can check the presence of these packages with:

```
conda list "^python$|numpy|astropy$|scipy$|matplotlib|numba|PyQT|ginga|yaml|h5py"
```

If the packages have been installed, this command should print out all the packages and their version numbers.

If any of the packages are out of date, they can be updated with a command like:

```
conda update scipy
```

The following packages need to be installed by cloning from GitHub:

- ginga JXP's fork of Ginga
- linetools Linetools is a package designed for the analysis of 1-D spectra.

Do **not** use pip install for these.

To remind you, install via GitHub with a sequence like:

```
git clone https://github.com/profxj/ginga
cd ginga
python setup.py install
```

This will push the code into your Python distribution.

1.1.2 Installing PypeIt

We recommend that you install PypeIt with *pip*:

```
pip install pypeit
```

Nuff said. If you have not yet satisfied all the requirements, PypeIt will fail when you first attempt to run it. You can grab all of them (except *ginga*) by doing:

```
pip install -r path/requirements.txt
```

where path is to wherever *pip* installed the code. Or you can download the requirements.txt file and run on it directly.

1.1.3 Tests

In order to assess whether PypeIt has been properly installed, we suggest you run the following tests:

1. Ensure run_pypeit works

Go to a directory outside of the PypeIt directory (e.g. your home directory), then type run_pypeit.:

```
cd
run_pypeit -h
```

2. Run the PypeIt unit tests

If you cloned the Repo (i.e., not PyPI), then you can run the standard tests by doing:

```
python setup.py test
```

3. Try the test suite – ONLY FOR DEVELOPERS

Ask for help if you really want to do this.

We have provided a suite of tests that you can download and run via this Repo: TestSuite

It can be installed as follows:

```
# we suggest installing this in the directory above PypeIt
git clone https://github.com/pypeit/PypeIt-development-suite.git
```

To run the test:

```
cd PypeIt-development-suite
./pypeit_test all
```

Note: pypeit_test can also take the argument kast instead of all.

The test takes a while to run but should run without issue if all the packages have been properly installed.

1.1.4 PIP

For the developers, see pyp_twine for details on how to push a new build to PyPI.

1.2 Code Flow

This describes the standard code flow of PypeIt.

1.2.1 ARMS

Multi-slit and longslit reductions.

Step	Class/module	Internals	Outputs	QA
Setup	PypeItSetup	fitstbl	keck_lris_red_setup_A.fits	
		setup_dict	setup_files/keck_lris_red_2018-Jun-19.setups	
			keck_lris_red_setup_A.pypeit	
Bias	BiasFrame	msbias	MasterBias_A_02_aa.fits	
ArcImg	ArcImage	msarc	MasterArc_A_02_aa.fits	
Bad Pixel Mask	BPMImage	msbpm		
Pixel location	—	pixlocn		
Trace Slits	TraceSlit	tslits_dict	MasterTrace_A_02_aa.fits.gz	Slit_Trace_A_02_aa.png
			MasterTrace_A_02_aa.json	
1D Wave Calib	WaveCalib	wv_calib	MasterWaveCalib_A_02_aa.json	Arc_1dfit_A_02_aa_S0000.png
Wave Tilts	WaveTilts	mstilts	MasterTilts_A_02_aa.fits	Arc_tilts_A_02_aa_S0000.png
Pixel flat	FlatField	mspixflat-nrm	MasterFlatField_A_02_aa.fits	
Process Science	ScienceImage	sciframe	spec2d_basename.fits	
Global skysub	ScienceImage	global_sky		
Find objects	ScienceImage	tracelist		
Extraction	ScienceImage	specobjs		
Flexure	arflex	flex_list		basename_flex_sky.png
				basename_flex_corr.png
Heliocentric	arwave	velcorr		
Flux	FlexSpec	sensfunc	spec1d_basename.fits	

CHAPTER 2

Running PypeIt

PYPEIT HOWTO

2.1 PypeIt Parameters

PypeIt allows you to customize its execution without having to change the code directly.

Although not ubiquitous, most optional arguments of PypeIt's algorithms are contained within the `pypeit.par.pypeitpar.PypeItPar` superset. See the *Current PypeItPar Parameter Hierarchy* below for the current structure of a `pypeit.par.pypeitpar.PypeItPar` instance.

More importantly, each instrument served provides its own default values for `pypeit.par.pypeitpar.PypeItPar` as defined by its *default_pypeit_par* method; e.g., `pypeit.spectrographs.shane_kast.ShaneKastSpectrograph.default_pypeit_par()`. Users can alter these parameters via the PypeIt file, see *PypeIt Reduction File*. Only those parameters that the user wishes to be different from the default *used for their specified instrument* need to be includes in the PypeIt file.

PypeIt uses the configobj class to parse the user supplied arguments. The syntax is important and the nesting of the parameter changes must match the *Current PypeItPar Parameter Hierarchy*. Examples of **'How to change parameters using the PypeIt file'_** are given below.

2.1.1 Current PypeItPar Parameter Hierarchy

PypeItPar Keywords

`[rdx]`: *ReducePar Keywords*

`[calibrations]`: *CalibrationsPar Keywords*

`[[biasframe]]`: *FrameGroupPar Keywords*

`[[[process]]]`: *ProcessImagesPar Keywords*

`[[darkframe]]`: *FrameGroupPar Keywords*

[[[process]]]: *ProcessImagesPar Keywords*

[[arcframe]]: *FrameGroupPar Keywords*

[[[process]]]: *ProcessImagesPar Keywords*

[[tiltframe]]: *FrameGroupPar Keywords*

[[[process]]]: *ProcessImagesPar Keywords*

[[pixelflatframe]]: *FrameGroupPar Keywords*

[[[process]]]: *ProcessImagesPar Keywords*

[[pinholeframe]]: *FrameGroupPar Keywords*

[[[process]]]: *ProcessImagesPar Keywords*

[[traceframe]]: *FrameGroupPar Keywords*

[[[process]]]: *ProcessImagesPar Keywords*

[[standardframe]]: *FrameGroupPar Keywords*

[[[process]]]: *ProcessImagesPar Keywords*

[[flatfield]]: *FlatFieldPar Keywords*

[[wavelengths]]: *WavelengthSolutionPar Keywords*

[[slits]]: *TraceSlitsPar Keywords*

[[tilts]]: *WaveTiltsPar Keywords*

[scienceframe]: *FrameGroupPar Keywords*

[[process]]: *ProcessImagesPar Keywords*

[scienceimage]: *ScienceImagePar Keywords*

[flexure]: *FlexurePar Keywords*

[fluxcalib]: *FluxCalibrationPar Keywords*

PypeItPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.PypeItPar`

Key	Type	Options	Default	Description
rdx	pypeit.par.pypeitpar.ReducePar		*ReducePar Keywords*	PypIt reduction rules.
calibrations	pypeit.par.pypeitpar.CalibrationsPar		*CalibrationsPar Keywords*	Parameters for the calibration algorithms
scienceframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for the science observations
scienceimage	pypeit.par.pypeitpar.ScienceImagePar		*ScienceImagePar Keywords*	Parameters determining sky-subtraction, object finding, and extraction
flexure	pypeit.par.pypeitpar.FlexurePar		*FlexurePar Keywords*	Parameters used by the flexure-correction procedure. Flexure corrections are not performed by default. To turn on, either set the parameters in the 'flexure' parameter group or set 'flexure = True' in the 'rdx' parameter group to use the default flexure-correction parameters.
fluxcalib	pypeit.par.pypeitpar.FluxCalibrationPar		*FluxCalibrationPar Keywords*	Parameters used by the flux-calibration procedure. Flux calibration is not performed by default. To turn on, either set the parameters in the 'fluxcalib' parameter group or set 'fluxcalib = True' in the 'rdx' parameter group to use the default flux-calibration parameters.

ReducePar Keywords

Class Instantiation: `pypeit.par.pypeitpar.ReducePar`

Key	Type	Options	Default	Description
spectrograph	str	gemini_gnirs, keck_deimos, keck_lris_blue, keck_lris_red, keck_lris_red_longonly, keck_nires, keck_hires_red, keck_hires_blue, mmt_binospec, keck_nirspec_low, shane_kast_blue, shane_kast_red, shane_kast_red_ret, tng_dolores, wht_isis_blue, vlt_xshooter_uvb, vlt_xshooter_vis, magellan_fire, magellan_mage, vlt_xshooter_nir, gemini_gmos_south_ham, gemini_gmos_north_e2v, gemini_gmos_north_ham, lbt_mods1r, lbt_mods1b, lbt_mods2r, lbt_mods2b, vlt_fors2		Spectrograph that provided the data to be reduced. Options are: gemini_gnirs, keck_deimos, keck_lris_blue, keck_lris_red, keck_lris_red_longonly, keck_nires, keck_hires_red, keck_hires_blue, mmt_binospec, keck_nirspec_low, shane_kast_blue, shane_kast_red, shane_kast_red_ret, tng_dolores, wht_isis_blue, vlt_xshooter_uvb, vlt_xshooter_vis, magellan_fire, magellan_mage, vlt_xshooter_nir, gemini_gmos_south_ham, gemini_gmos_north_e2v, gemini_gmos_north_ham, lbt_mods1r, lbt_mods1b, lbt_mods2r, lbt_mods2b, vlt_fors2
detnum	int, list			Restrict reduction to a list of detector indices
sortroot	str			A filename given to output the details of the sorted files. If None, the default is the root name of the pypeit file. If off, no output is produced.
calwin	int, float		0	The window of time in hours to search for calibration frames for a science frame
scidir	str		Science	Directory relative to calling directory to write science files.
qadir	str		QA	Directory relative to calling directory to write quality assessment files.
redux_path	str		/Users/westfall/Work/packages/pypeit/doc	Path to folder for performing reductions.
ignore_bad_headers	bool		False	Ignore bad headers (NOT recommended unless you know it is safe).

CalibrationsPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.CalibrationsPar`

Key	Type	Options	Default	Description
caldir	str		Masters	Directory relative to calling directory to write master files.
setup	str			If masters='force', this is the setup name to be used: e.g., C_02_aa . The detector number is ignored but the other information must match the Master Frames in the master frame folder.
trim	bool		True	Trim the frame to isolate the data
badpix	bool		True	Make a bad pixel mask? Bias frames must be provided.
biasframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for the bias correction
darkframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for the dark-current correction
arcframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for the wavelength calibration
tiltframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for the wavelength tilts
pixelflatframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for the field flattening
pinholeframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for the pinholes
traceframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for images used for slit tracing
standardframe	pypeit.par.pypeitpar.FrameGroupPar		*FrameGroupPar Keywords*	The frames and combination rules for the spectrophotometric standard observations
flatfield	pypeit.par.pypeitpar.FlatFieldPar		*FlatFieldPar Keywords*	Parameters used to set the flat-field procedure
wavelengths	pypeit.par.pypeitpar.WavelengthSolutionPar		*WavelengthSolutionPar Keywords*	Parameters used to derive the wavelength solution
slits	pypeit.par.pypeitpar.TraceSlitsPar		*TraceSlitsPar Keywords*	Define how the slits should be traced using the trace ?PINHOLE? frames
tilts	pypeit.par.pypeitpar.WaveTiltsPar		*WaveTiltsPar Keywords*	Define how to tract the slit tilts using the trace frames

FlatFieldPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.FlatFieldPar`

Key	Type	Options	Default	Description
method	str	bspline, skip	bspline	Method used to flat field the data; use skip to skip flat-fielding. Options are: None, bspline, skip
frame	str		pixelflat	Frame to use for field flattening. Options are: "pixelflat", or a specified calibration filename.
illumflatten	bool		True	Use the flat field to determine the illumination profile of each slit.
spec_samp_fine	int, float		1.2	bspline break point spacing in units of pixels for spectral fit to flat field blaze function.
spec_samp_coarse	int, float		50.0	bspline break point spacing in units of pixels for 2-d bspline-polynomial fit to flat field image residuals. This should be a large number unless you are trying to fit a sky flat with lots of narrow spectral features.
spat_samp	int, float		5.0	Spatial sampling for slit illumination function. This is the width of the median filter in pixels used to determine the slit illumination function, and thus sets the minimum scale on which the illumination function will have features.
tweak_slits	bool		True	Use the illumination flat field to tweak the slit edges. This will work even if illumflatten is set to False
tweak_slits_thresh	float		0.93	If tweak_slits is True, this sets the illumination function threshold used to tweak the slit boundaries based on the illumination flat. It should be a number less than 1.0
tweak_slits_maxfrac	float		0.1	If tweak_slit is True, this sets the maximum fractional amount (of a slits width) allowed for trimming each (i.e. left and right) slit boundary, i.e. the default is 10% which means slits would shrink or grow by at most 20% (10% on each side)

WavelengthSolutionPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.WavelengthSolutionPar`

Key	Type	Options
reference	str	arc, sky, pixel
method	str	simple, semi-brute, basic, holy-grail, reidentify, full_templ
echelle	bool	
ech_fix_format	bool	
ech_nspec_coeff	int	
ech_norder_coeff	int	
ech_sigrej	int, float	
lamps	list	
nonlinear_counts	float	
sigdetect	int, float, list, ndarray	
fwhm	int, float	
reid_arxiv	str	
nreid_min	int	
cc_thresh	float, list, ndarray	
cc_local_thresh	float	
nlocal_cc	int	
rms_threshold	float, list, ndarray	
match_toler	float	

Key	Type	Options
`func`	str	
`n_first`	int	
`n_final`	int, float, list, ndarray	
`sigrej_first`	float	
`sigrej_final`	float	
`wv_cen`	float	
`disp`	float	
`numsearch`	int	
`nfitpix`	int	
`IDpixels`	int, float, list	
`IDwaves`	int, float, list	
`medium`	str	`vacuum`, `air`
`frame`	str	`observed`, `heliocentric`, `barycentric`
`nsnippet`	int	

TraceSlitsPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.TraceSlitsPar`

Key	Type	Options	Default	Description
function	str	polynomial, legendre, chebyshev	legendre	Function use to trace the slit center. Options are: polynomial, legendre, chebyshev
medreject	int		0	Median-smoothing iterations to perform on sqrt(trace) image before applying to Sobel filter, which detects slit/order edges.
number	int		-1	Manually set the number of slits to identify (>=1). 'auto' or -1 will automatically identify the number of slits.
trim	tuple		0, 0	How much to trim off each edge of each slit. Each number should be 0 or positive
maxgap	int			Maximum number of pixels to allow for the gap between slits. Use None if the neighbouring slits are far apart or of similar illumination.
maxshift	int, float		0.15	Maximum shift in trace crude. Use a larger number for more curved slits/orders.
pad	int		0	Integer number of pixels to consider beyond the slit edges.
sigdetect	int, float		20.0	Sigma detection threshold for edge detection
min_slit_width	float		6.0	If a slit spans less than this number of arcseconds over the spatial direction of the detector, it will be ignored. Use this option to prevent the alignment (box) slits from multislit reductions, which typically cannot be reduced without a significant struggle.
add_slits	str, list			Add one or more user-defined slits. The syntax to define a slit to add is: 'det:spec:spat_left:spat_right' where det=detector, spec=spectral pixel, spat_left=spatial pixel of left slit boundary, and spat_righ=spatial pixel of right slit boundary. For example, '2:2000:2121:2322,3:2000:1201:1500' will add a slit to detector 2 passing through spec=2000 extending spatially from 2121 to 2322 and another on detector 3 at spec=2000 extending from 1201 to 1500.
rm_slits	str, list			Remove one or more user-specified slits. The syntax used to define a slit to remove is: 'det:spec:spat' where det=detector, spec=spectral pixel, spat=spatial pixel. For example, '2:2000:2121,3:2000:1500' will remove the slit on detector 2 that contains pixel (spat,spec)=(2000,2121) and on detector 3 that contains pixel (2000,2121).
diffpolyorder	int		2	Order of the 2D function used to fit the 2d solution for the spatial size of all orders.
single	list		[]	Add a single, user-defined slit based on its location on each detector. Syntax is a list of values, 2 per detector, that define the slit according to column values. The second value (for the right edge) must be greater than 0 to be applied. LRISr example: setting single = -1, -1, 7, 295 means the code will skip the user-definition for the first detector but adds one for the second. None means no user-level slits defined.
sobel_mode	str	nearest, constant	nearest	Mode for Sobel filtering. Default is 'nearest' but the developers find 'constant' works best for DEIMOS.
pcaexttap	list		0, 0	The number of extra orders to predict in the negative (first number) and positive (second number) direction. Must be two numbers in the list and they must be integers.
smash_range	list		0.0, 1.0	Range of the slit in the spectral direction (in fractional units) to smash when searching for slit edges. If the spectrum covers only a portion of the image, use that range.
traceimpoly	int		5	Order of legendre polynomial fits to slit/order boundary traces.
mask_frac_thresh	float		0.6	Minimum fraction of the slit edge that was *not* masked to use in initial PCA.

WaveTiltsPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.WaveTiltsPar`

Key	Type	Options	Default	Description
`idsonly`	bool		False	Only use the arc lines that have an identified wavelength to trace tilts
`tracethresh`	int, float, list, ndarray		20.0	Significance threshold for arcs to be used in tracing wavelength tilts. This can be a single number or a list/array providing the value for each slit
`sig_neigh`	int, float		10.0	Significance threshold for arcs to be used in line identification for the purpose of identifying neighboring lines.The tracethresh parameter above determines the significance threshold of lines that will be traced, but these lines must be at least nfwhm_neigh fwhm away from neighboring lines. This parameter determines the significance above which a line must be to be considered a possible colliding neighbor. A low value of sig_neigh will result in an overall larger number of lines, which will result in more lines above tracethresh getting rejected
`nfwhm_neigh`	int, float		3.0	Required separation between neighboring arc lines for them to be considered for tilt tracing in units of the the spectral fwhm (see wavelength parset where fwhm is defined)
`maxdev_tracefit`	int, float		0.2	Maximum absolute deviation (in units of fwhm) for the legendre polynomial fits to individual arc line tilt fits during iterative trace fitting (flux weighted, then gaussian weighted)
`sigrej_trace`	int, float		3.0	Outlier rejection significance to determine which traced arc lines should be included in the global fit
`spat_order`	int, float, list, ndarray		3	Order of the legendre polynomial to be fit to the the tilt of an arc line. This parameter determinesboth the orer of the *individual* arc line tilts, as well as the order of the spatial direction of the2d legendre polynomial (spatial, spectral) that is fit to obtain a global solution for the tilts across theslit/order. This can be a single number or a list/array providing the value for each slit
`spec_order`	int, float, list, ndarray		4	Order of the spectral direction of the 2d legendre polynomial (spatial, spectral) that is fit to obtain a global solution for the tilts across the slit/order. This can be a single number or a list/array providing the value for each slit
`func2d`	str		`legendre2d`	Type of function for 2D fit
`maxdev2d`	int, float		0.25	Maximum absolute deviation (in units of fwhm) rejection threshold used to determines which pixels in global 2d fits to arc line tilts are rejected because they deviate from the model by more than this value
`sigrej2d`	int, float		3.0	Outlier rejection significance determining which pixels on a fit to an arc line tilt are rejected by the global 2D fit

FrameGroupPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.FrameGroupPar`

Key	Type	Options	Default	Description
`frametype`	str	`pinhole`, `arc`, `dark`, `pixelflat`, `standard`, `tilt`, `bias`, `trace`, `science`	`science`	Frame type. Options are: pinhole, arc, dark, pixelflat, standard, tilt, bias, trace, science
`useframe`	str		`science`	A master calibrations file to use if it exists.
`number`	int		0	Used in matching calibration frames to science frames. This sets the number of frames to use of this type
`exprng`	list		None, None	Used in identifying frames of this type. This sets the minimum and maximum allowed exposure times. There must be two items in the list. Use None to indicate no limit; i.e., to select exposures with any time greater than 30 sec, use exprng = [30, None].
`process`	pypeit.par.pypeitpar.ProcessImagesPar		*ProcessImagesPar Keywords*	Parameters used for basic image processing

ProcessImagesPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.ProcessImagesPar`

Key	Type	Options	Default	Description
overscan	str	polynomial, savgol, median, none	savgol	Method used to fit the overscan. Options are: polynomial, savgol, median, none
overscan_par	int, list		5, 65	Parameters for the overscan subtraction. For 'polynomial', set overscan_par = order, number of pixels, number of repeats ; for 'savgol', set overscan_par = order, window size ; for 'median', set overscan_par = None or omit the keyword.
match	int, float		-1	(Deprecate?) Match frames with pixel counts that are within N-sigma of one another, where match=N below. If N < 0, nothing is matched.
combine	str	mean, median, weightmean	weightmean	Method used to combine frames. Options are: mean, median, weightmean
satpix	str	reject, force, nothing	reject	Handling of saturated pixels. Options are: reject, force, nothing
sigrej	int, float		20.0	Sigma level to reject cosmic rays (<= 0.0 means no CR removal)
n_lohi	list		0, 0	Number of pixels to reject at the lowest and highest ends of the distribution; i.e., n_lohi = low, high. Use None for no limit.
sig_lohi	list		3.0, 3.0	Sigma-clipping level at the low and high ends of the distribution; i.e., sig_lohi = low, high. Use None for no limit.
replace	str	min, max, mean, median, weightmean, maxnonsat	maxnonsat	If all pixels are rejected, replace them using this method. Options are: min, max, mean, median, weightmean, maxnonsat
lamaxiter	int		1	Maximum number of iterations for LA cosmics routine.
grow	int, float		1.5	Factor by which to expand regions with cosmic rays detected by the LA cosmics routine.
rmcompact	bool		True	Remove compact detections in LA cosmics routine
sigclip	int, float		4.5	Sigma level for rejection in LA cosmics routine
sigfrac	int, float		0.3	Fraction for the lower clipping threshold in LA cosmics routine.
objlim	int, float		3.0	Object detection limit in LA cosmics routine
bias	str	as_available, force, skip	as_available	Parameter for bias subtraction. as_available: Bias subtract if bias frames were providedforce: Require bias subtraction, i.e., break if bias frames were not providedskip: Skip bias subtraction even if bias frames were provided

ScienceImagePar Keywords

Class Instantiation: `pypeit.par.pypeitpar.ScienceImagePar`

Key	Type	Options	Default	Description
bspline_spacing	int, float		0.6	Break-point spacing for the bspline sky subtraction fits.
boxcar_radius	int, float		1.5	Boxcar radius in arcseconds used for boxcar extraction
trace_npoly	int		5	Order of legendre polynomial fits to object traces.
global_sky_std	bool			Global sky subtraction will be performed on standard stars. This should be turnedoff for example for near-IR reductions with narrow slits, since bright standards canfill the slit causing global sky-subtraction to fail. In these situations we go straight to local sky-subtraction since it is designed to deal with such situations
sig_thresh	int, float		10.0	Significance threshold for object finding.
maxnumber	int		10	Maximum number of objects to extract in a science frame. Use None for no limit.
sn_gauss	int, float		4.0	S/N threshold for performing the more sophisticated optimal extraction which performs a b-spline fit to the object profile. For S/N < sn_gauss the code will simply optimal extractwith a Gaussian with FWHM determined from the object finding.
find_trim_edge	list		5, 5	Trim the slit by this number of pixels left/right before finding objects
std_prof_nsigma	float		30.0	prof_nsigma parameter for Standard star extraction. Prevents undesired rejection.
model_full_slit	bool		False	If True local sky subtraction will be performed on the entire slit. If False, local sky subtraction will be applied to only a restricted region around each object. This should be set to True for either multislit observations using narrow slits or echelle observations with narrow slits
no_poly	bool		False	Turn off polynomial basis (Legendre) in global sky subtraction
manual	list			List of manual extraction parameter sets
sky_sigrej	float		3.0	Rejection parameter for local sky subtraction

FlexurePar Keywords

Class Instantiation: `pypeit.par.pypeitpar.FlexurePar`

Key	Type	Options	Default	Description
method	str	boxcar, skip slitcen, skip		Method used to correct for flexure. Use skip for no correction. If slitcen is used, the flexure correction is performed before the extraction of objects (not recommended). Options are: None, boxcar, slitcen, skip
maxshift	int, float		20	Maximum allowed flexure shift in pixels.
spectrum	str		/Users/westfall/Work/packages/pypeit/pypeit/data/sky_spec/paranal_sky.fits	Archive sky spectrum to be used for the flexure correction.

FluxCalibrationPar Keywords

Class Instantiation: `pypeit.par.pypeitpar.FluxCalibrationPar`

Key	Type	Options	Default	Description
balm_mask_wid	float		5.0	Mask width for Balmer lines in Angstroms.
std_file	str			Standard star file to generate sensfunc
std_obj_id	str, int			Specifies object in spec1d file to use as standard. The brightest object found is used otherwise.
sensfunc	str			FITS file that contains or will contain the sensitivity function.
extinct_correct	bool		True	If extinct_correct=True the code will use an atmospheric extinction model to extinction correct the data below 10000A. Note that this correction makes no sense if one is telluric correcting and this shold be set to False
telluric_correct	bool		False	If telluric_correct=True the code will grab the sens_dict['telluric'] tag from the sensfunc dictionary and apply it to the data.
star_type	str			Spectral type of the standard star (for near-IR mainly)
star_mag	float			Magnitude of the standard star (for near-IR mainly)
multi_det	list			List of detector numbers to splice together for multi-detector instruments (e.g. DEIMOS) They are assumed to be in order of increasing wavelength And that there is *no* overlap in wavelength across detectors (might be ok if there is)
telluric	bool		False	If telluric=True the code creates a synthetic standard star spectrum using the Kurucz models, the sens func is created setting nresln=1.5 it contains the correction for telluric lines.
poly_norder	int		5	Polynomial order for sensfunc fitting
polycorrect	bool		True	Whether you want to correct the sensfunc with polynomial in the telluric and recombination line regions

2.1.2 Instrument-Specific Default Configuration

The following provides the changes to the global default parameters provided above for each instrument. That is, if one were to include these in the PypeIt file, you would be reproducing the effect of the *default_pypeit_par* method specific to each derived `pypeit.spectrographs.spectrograph.Spectrograph` class.

KECK DEIMOS

Alterations to the default parameters are:

```
[rdx]
    spectrograph = keck_deimos
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 2
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
```

(continues on next page)

(continued from previous page)

```
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = None, 30
        [[[process]]]
            combine = median
            sig_lohi = 10.0, 10.0
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 3
        exprng = None, 30
    [[standardframe]]
        number = 1
    [[wavelengths]]
        lamps = ArI, NeI, KrI, XeI
        nonlinear_counts = 56360.1
        match_toler = 2.5
        n_first = 3
    [[slits]]
        sigdetect = 50.0
        trace_npoly = 3
[scienceframe]
    exprng = 30, None
    [[process]]
        sigclip = 4.0
        objlim = 1.5
[flexure]
    method = boxcar
```

KECK LRISb

Alterations to the default parameters are:

```
[rdx]
    spectrograph = keck_lris_blue
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = None, 30
    [[pinholeframe]]
```

(continues on next page)

(continued from previous page)

```
        exprng = 999999, None
    [[traceframe]]
        number = 3
        exprng = None, 30
    [[standardframe]]
        number = 1
    [[wavelengths]]
        method = full_template
        lamps = NeI, ArI, CdI, KrI, XeI, ZnI, HgI
        nonlinear_counts = 56360.1
        sigdetect = 10.0
        rms_threshold = 0.2
        match_toler = 2.5
        n_first = 3
    [[slits]]
        sigdetect = 30.0
[scienceframe]
    exprng = 29, None
[flexure]
    method = boxcar
```

KECK LRISr

Alterations to the default parameters are:

```
[rdx]
    spectrograph = keck_lris_red
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = None, 30
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 3
        exprng = None, 30
    [[standardframe]]
        number = 1
    [[wavelengths]]
        lamps = NeI, ArI, CdI, KrI, XeI, ZnI, HgI
        nonlinear_counts = 49806.6
        sigdetect = 10.0
```

(continues on next page)

(continued from previous page)

```
        rms_threshold = 0.2
    [[slits]]
        sigdetect = 50.0
    [[tilts]]
        tracethresh = 25
        maxdev_tracefit = 1.0
        spat_order = 4
        spec_order = 7
        maxdev2d = 1.0
        sigrej2d = 5.0
[scienceframe]
    exprng = 29, None
    [[process]]
        sigclip = 5.0
        objlim = 5.0
[scienceimage]
    bspline_spacing = 0.8
[flexure]
    method = boxcar
```

KECK LRISr

Alterations to the default parameters are:

```
[rdx]
    spectrograph = keck_lris_red
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = None, 30
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 3
        exprng = None, 30
    [[standardframe]]
        number = 1
    [[wavelengths]]
        lamps = NeI, ArI, CdI, KrI, XeI, ZnI, HgI
        nonlinear_counts = 70731.92549999998
        sigdetect = 10.0
        rms_threshold = 0.2
```

(continues on next page)

(continued from previous page)

```
    [[slits]]
        sigdetect = 50.0
    [[tilts]]
        tracethresh = 25
        maxdev_tracefit = 1.0
        spat_order = 4
        spec_order = 7
        maxdev2d = 1.0
        sigrej2d = 5.0
[scienceframe]
    exprng = 29, None
    [[process]]
        sigclip = 5.0
        objlim = 5.0
[scienceimage]
    bspline_spacing = 0.8
[flexure]
    method = boxcar
```

KECK NIRES

Alterations to the default parameters are:

```
[rdx]
    spectrograph = keck_nires
[calibrations]
    [[biasframe]]
        useframe = none
    [[darkframe]]
        exprng = 20, None
    [[arcframe]]
        number = 1
        exprng = 20, None
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
    [[traceframe]]
        number = 5
    [[standardframe]]
        number = 1
        exprng = None, 20
    [[flatfield]]
        illumflatten = False
    [[wavelengths]]
        method = reidentify
        echelle = True
        ech_norder_coeff = 6
        ech_sigrej = 3.0
        lamps = OH_NIRES
        nonlinear_counts = 760000.0
```

(continues on next page)

(continued from previous page)

```
        fwhm = 5.0
        reid_arxiv = keck_nires.fits
        rms_threshold = 0.2
        n_final = 3, 4, 4, 4, 4
    [[tilts]]
        tracethresh = 10.0
[scienceframe]
    exprng = 20, None
    [[process]]
        satpix = nothing
        sigclip = 20.0
[scienceimage]
    bspline_spacing = 0.8
```

KECK NIRSPEC

Alterations to the default parameters are:

```
[calibrations]
    [[biasframe]]
        exprng = None, 2
    [[darkframe]]
        exprng = None, 5
    [[arcframe]]
        number = 1
        exprng = 1, None
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = 0, None
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 5
        exprng = 0, None
    [[standardframe]]
        number = 1
        exprng = None, 5
    [[wavelengths]]
        lamps = OH_R24000
        rms_threshold = 0.2
    [[slits]]
        sigdetect = 200.0
    [[tilts]]
        tracethresh = 10.0
[scienceframe]
    exprng = 1, None
```

SHANE KASTb

Alterations to the default parameters are:

```
[rdx]
    spectrograph = shane_kast_blue
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        exprng = None, 61
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = 0, None
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 5
        exprng = 0, None
    [[standardframe]]
        number = 1
        exprng = 1, 61
    [[wavelengths]]
        method = full_template
        lamps = CdI, HgI, HeI
        nonlinear_counts = 49806.6
        rms_threshold = 0.2
        match_toler = 2.5
        n_first = 3
    [[tilts]]
        maxdev_tracefit = 0.02
        spec_order = 5
        maxdev2d = 0.02
[scienceframe]
    exprng = 61, None
[flexure]
    method = boxcar
    spectrum = /Users/westfall/Work/packages/pypeit/pypeit/data/sky_spec/sky_kastb_
↪600.fits
```

SHANE KASTr

Alterations to the default parameters are:

```
[rdx]
    spectrograph = shane_kast_red
```

(continues on next page)

(continued from previous page)

```
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        exprng = None, 61
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = 0, None
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 5
        exprng = 0, None
    [[standardframe]]
        number = 1
        exprng = 1, 61
    [[wavelengths]]
        lamps = NeI, HgI, HeI, ArI
        nonlinear_counts = 49806.6
[scienceframe]
    exprng = 61, None
[flexure]
    method = boxcar
```

SHANE KASTr

Alterations to the default parameters are:

```
[rdx]
    spectrograph = shane_kast_red_ret
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        exprng = None, 61
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
```

(continues on next page)

(continued from previous page)

```
    [[pixelflatframe]]
        number = 3
        exprng = 0, None
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 3
        exprng = 0, None
    [[standardframe]]
        number = 1
        exprng = 1, 61
    [[wavelengths]]
        lamps = NeI, HgI, HeI, ArI
        nonlinear_counts = 91200.0
[scienceframe]
    exprng = 61, None
[flexure]
    method = boxcar
```

TNG DOLORES

Alterations to the default parameters are:

```
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 0.1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 3
    [[standardframe]]
        number = 1
[scienceframe]
    exprng = 1, None
```

WHT ISISb

Alterations to the default parameters are:

```
[rdx]
    spectrograph = wht_isis_blue
```

(continues on next page)

(continued from previous page)

```
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        exprng = None, 120
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        [[[process]]]
            combine = median
            sig_lohi = 10.0, 10.0
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 3
    [[standardframe]]
        number = 1
        exprng = None, 120
    [[wavelengths]]
        method = simple
[scienceframe]
    exprng = 90, None
```

VLT XShooter_UVB

Alterations to the default parameters are:

```
[rdx]
    spectrograph = vlt_xshooter_uvb
[calibrations]
    [[biasframe]]
        number = 5
    [[arcframe]]
        number = 1
        [[[process]]]
            overscan = median
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
    [[traceframe]]
        number = 3
        [[[process]]]
```

(continues on next page)

(continued from previous page)

```
            overscan = median
    [[standardframe]]
        number = 1
    [[wavelengths]]
        method = reidentify
        echelle = True
        ech_norder_coeff = 5
        ech_sigrej = 3.0
        lamps = ThAr_XSHOOTER_UVB
        nonlinear_counts = 55900.0
        reid_arxiv = vlt_xshooter_uvb1x1_iraf.json
        rms_threshold = 0.5
    [[slits]]
        maxshift = 0.5
        sigdetect = 8.0
[scienceframe]
    useframe = overscan
```

VLT XShooter_VIS

Alterations to the default parameters are:

```
[rdx]
    spectrograph = vlt_xshooter_vis
[calibrations]
    [[biasframe]]
        number = 5
    [[arcframe]]
        number = 1
        [[[process]]]
            overscan = median
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
    [[traceframe]]
        number = 3
        [[[process]]]
            overscan = median
    [[standardframe]]
        number = 1
    [[flatfield]]
        tweak_slits_thresh = 0.9
    [[wavelengths]]
        method = reidentify
        echelle = True
        ech_sigrej = 3.0
        lamps = ThAr_XSHOOTER_VIS
        nonlinear_counts = 56360.1
        fwhm = 11.0
        reid_arxiv = vlt_xshooter_vis1x1.fits
        cc_thresh = 0.5
```

(continues on next page)

(continued from previous page)

```
        cc_local_thresh = 0.5
        rms_threshold = 0.5
        n_final = 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3
    [[slits]]
        maxshift = 0.5
        sigdetect = 8.0
        trace_npoly = 8
    [[tilts]]
        tracethresh = 15
        spec_order = 5
[scienceframe]
    useframe = overscan
[scienceimage]
    bspline_spacing = 0.5
    model_full_slit = True
```

VLT XShooter_NIR

Alterations to the default parameters are:

```
[rdx]
    spectrograph = vlt_xshooter_nir
[calibrations]
    [[biasframe]]
        useframe = none
        number = 5
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
    [[traceframe]]
        number = 3
    [[standardframe]]
        number = 1
    [[flatfield]]
        illumflatten = False
        tweak_slits_thresh = 0.9
    [[wavelengths]]
        method = reidentify
        echelle = True
        ech_nspec_coeff = 5
        ech_norder_coeff = 5
        ech_sigrej = 3.0
        lamps = OH_XSHOOTER
        nonlinear_counts = 172000.0
        sigdetect = 10.0
        fwhm = 5.0
        reid_arxiv = vlt_xshooter_nir.fits
        cc_thresh = 0.5
```

(continues on next page)

(continued from previous page)

```
        cc_local_thresh = 0.5
        rms_threshold = 0.25
    [[slits]]
        maxshift = 0.5
        sigdetect = 120.0
        trace_npoly = 8
    [[tilts]]
        tracethresh = 25.0
        maxdev_tracefit = 0.04
        maxdev2d = 0.04
[scienceframe]
    useframe = none
    [[process]]
        satpix = nothing
        sigclip = 20.0
[scienceimage]
    bspline_spacing = 0.8
    trace_npoly = 8
    global_sky_std = False
    model_full_slit = True
```

GEMINI-N GNIRS

Alterations to the default parameters are:

```
[rdx]
    spectrograph = gemini_gnirs
[calibrations]
    [[biasframe]]
        useframe = none
        [[[process]]]
            overscan = none
    [[darkframe]]
        [[[process]]]
            overscan = none
    [[arcframe]]
        number = 1
        [[[process]]]
            overscan = none
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            overscan = none
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = None, 30
        [[[process]]]
            overscan = none
    [[pinholeframe]]
        [[[process]]]
            overscan = none
    [[traceframe]]
        number = 5
```

(continues on next page)

(continued from previous page)

```
        exprng = None, 30
        [[[process]]]
            overscan = none
    [[standardframe]]
        number = 1
        exprng = None, 30
        [[[process]]]
            overscan = none
    [[flatfield]]
        illumflatten = False
        tweak_slits_thresh = 0.9
    [[wavelengths]]
        method = reidentify
        echelle = True
        ech_nspec_coeff = 3
        ech_norder_coeff = 5
        ech_sigrej = 3.0
        lamps = OH_GNIRS
        nonlinear_counts = 106500.0
        reid_arxiv = gemini_gnirs.fits
        cc_thresh = 0.6
        rms_threshold = 1.0
        n_final = 1, 3, 3, 3, 3, 3
    [[slits]]
        maxshift = 0.5
        sigdetect = 50.0
    [[tilts]]
        tracethresh = 5.0, 10, 10, 10, 10, 10
        sig_neigh = 5.0
        nfwhm_neigh = 2.0
[scienceframe]
    exprng = 30, None
[scienceimage]
    bspline_spacing = 0.8
    global_sky_std = False
    sig_thresh = 5.0
    model_full_slit = True
    no_poly = True
```

GEMINI-S GMOS-S

Alterations to the default parameters are:

```
[calibrations]
    [[biasframe]]
        number = 5
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
```

(continues on next page)

(continued from previous page)

```
        number = 5
        [[[process]]]
            combine = median
            sig_lohi = 10.0, 10.0
    [[traceframe]]
        number = 3
    [[standardframe]]
        number = 1
    [[wavelengths]]
        lamps = CuI, ArI, ArII
        rms_threshold = 0.4
    [[slits]]
        trace_npoly = 3
```

GEMINI-N GMOS-N

Alterations to the default parameters are:

```
[calibrations]
    [[biasframe]]
        number = 5
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        [[[process]]]
            combine = median
            sig_lohi = 10.0, 10.0
    [[traceframe]]
        number = 3
    [[standardframe]]
        number = 1
    [[wavelengths]]
        lamps = CuI, ArI, ArII
        rms_threshold = 0.4
    [[slits]]
        trace_npoly = 3
```

GEMINI-N GMOS-N

Alterations to the default parameters are:

```
[calibrations]
    [[biasframe]]
        number = 5
    [[arcframe]]
        number = 1
        [[[process]]]
```

(continues on next page)

(continued from previous page)

```
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        [[[process]]]
            combine = median
            sig_lohi = 10.0, 10.0
    [[traceframe]]
        number = 3
    [[standardframe]]
        number = 1
    [[wavelengths]]
        lamps = CuI, ArI, ArII
        rms_threshold = 0.4
    [[slits]]
        trace_npoly = 3
```

MAGELLAN FIRE

Alterations to the default parameters are:

```
[rdx]
    spectrograph = magellan_fire
[calibrations]
    [[biasframe]]
        useframe = overscan
    [[darkframe]]
        exprng = 20, None
    [[arcframe]]
        number = 1
        exprng = 20, None
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
    [[traceframe]]
        number = 5
    [[standardframe]]
        number = 1
        exprng = None, 60
    [[wavelengths]]
        echelle = True
        ech_sigrej = 3.0
        lamps = OH_XSHOOTER
        nonlinear_counts = 20000.0
        rms_threshold = 0.2
    [[slits]]
        maxshift = 0.5
```

(continues on next page)

(continued from previous page)

```
        sigdetect = 50
    [[tilts]]
        tracethresh = 10, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30,␣
↪30, 30, 30, 30, 30, 10
[scienceframe]
    exprng = 20, None
    [[process]]
        satpix = nothing
        sigclip = 20.0
```

MAGELLAN magellan_mage

Alterations to the default parameters are:

```
[rdx]
    spectrograph = magellan_mage
[calibrations]
    [[biasframe]]
        number = 5
    [[darkframe]]
        exprng = 20, None
    [[arcframe]]
        number = 1
        exprng = 20, None
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
    [[traceframe]]
        number = 3
    [[standardframe]]
        number = 1
        exprng = None, 20
    [[wavelengths]]
        method = reidentify
        echelle = True
        ech_sigrej = 3.0
        lamps = ThAr_MagE
        nonlinear_counts = 64879.65
        reid_arxiv = magellan_mage.fits
        cc_thresh = 0.5
        cc_local_thresh = 0.5
        rms_threshold = 0.2
    [[slits]]
        maxshift = 3.0
        sigdetect = 10.0
    [[tilts]]
        tracethresh = 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10
[scienceframe]
    exprng = 20, None
    [[process]]
```

(continues on next page)

(continued from previous page)

```
        satpix = nothing
        sigclip = 20.0
[scienceimage]
    find_trim_edge = 4, 4
```

KECK HIRES_R

Alterations to the default parameters are:

```
[rdx]
    spectrograph = keck_hires_red
[calibrations]
    [[biasframe]]
        number = 5
    [[arcframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
    [[traceframe]]
        number = 3
    [[standardframe]]
        number = 1
        exprng = None, 600
    [[wavelengths]]
        echelle = True
        ech_sigrej = 3.0
        lamps = ThAr
        nonlinear_counts = 56360.1
        rms_threshold = 0.25
    [[slits]]
        maxshift = 0.5
        sigdetect = 600.0
[scienceframe]
    exprng = 600, None
    [[process]]
        satpix = nothing
        sigclip = 20.0
```

LBT MODS1R

Alterations to the default parameters are:

```
[rdx]
    spectrograph = lbt_mods1r
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
```

(continues on next page)

(continued from previous page)

```
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        exprng = None, 60
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = 0, None
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 5
        exprng = 0, None
    [[standardframe]]
        number = 1
        exprng = 1, 200
    [[wavelengths]]
        lamps = OH_MODS
        nonlinear_counts = 64879.65
        fwhm = 10.0
        rms_threshold = 1.0
        n_first = 1
    [[slits]]
        sigdetect = 300
    [[tilts]]
        maxdev_tracefit = 0.02
        spat_order = 5
        spec_order = 5
        maxdev2d = 0.02
[scienceframe]
    exprng = 200, None
```

LBT MODS1B

Alterations to the default parameters are:

```
[rdx]
    spectrograph = lbt_mods1b
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        exprng = None, 60
        [[[process]]]
            sigrej = -1
```

(continues on next page)

(continued from previous page)

```
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = 0, None
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 5
        exprng = 0, None
    [[standardframe]]
        number = 1
        exprng = 1, 200
    [[wavelengths]]
        lamps = XeI, ArII, ArI, NeI, KrI
        nonlinear_counts = 64879.65
        rms_threshold = 0.2
        n_first = 1
    [[slits]]
        sigdetect = 300
    [[tilts]]
        maxdev_tracefit = 0.02
        spec_order = 5
        maxdev2d = 0.02
[scienceframe]
    exprng = 200, None
```

LBT MODS2R

Alterations to the default parameters are:

```
[rdx]
    spectrograph = lbt_mods2r
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        exprng = None, 60
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = 0, None
    [[pinholeframe]]
        exprng = 999999, None
```

(continues on next page)

(continued from previous page)

```
    [[traceframe]]
        number = 5
        exprng = 0, None
    [[standardframe]]
        number = 1
        exprng = 1, 200
    [[wavelengths]]
        lamps = OH_MODS
        nonlinear_counts = 64879.65
        fwhm = 10.0
        rms_threshold = 1.0
        n_first = 1
    [[slits]]
        sigdetect = 300
    [[tilts]]
        maxdev_tracefit = 0.02
        spec_order = 5
        maxdev2d = 0.02
[scienceframe]
    exprng = 200, None
```

LBT MODS2B

Alterations to the default parameters are:

```
[rdx]
    spectrograph = lbt_mods2b
[calibrations]
    [[biasframe]]
        number = 5
        exprng = None, 1
    [[darkframe]]
        exprng = 999999, None
    [[arcframe]]
        number = 1
        exprng = None, 60
        [[[process]]]
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        exprng = 0, None
    [[pinholeframe]]
        exprng = 999999, None
    [[traceframe]]
        number = 5
        exprng = 0, None
    [[standardframe]]
        number = 1
        exprng = 1, 200
    [[wavelengths]]
        lamps = XeI, ArII, ArI, NeI, KrI
```

(continues on next page)

(continued from previous page)

```
        nonlinear_counts = 64879.65
        rms_threshold = 0.2
        n_first = 1
    [[slits]]
        sigdetect = 300
    [[tilts]]
        maxdev_tracefit = 0.02
        spec_order = 5
        maxdev2d = 0.02
[scienceframe]
    exprng = 200, None
```

VLT FORS2

Alterations to the default parameters are:

```
[rdx]
    spectrograph = vlt_fors2
[calibrations]
    [[biasframe]]
        number = 5
        [[[process]]]
            overscan = median
    [[darkframe]]
        [[[process]]]
            overscan = median
    [[arcframe]]
        number = 1
        [[[process]]]
            overscan = median
            sigrej = -1
    [[tiltframe]]
        number = 1
        [[[process]]]
            overscan = median
            sigrej = -1
    [[pixelflatframe]]
        number = 5
        [[[process]]]
            overscan = median
    [[pinholeframe]]
        [[[process]]]
            overscan = median
    [[traceframe]]
        number = 3
        [[[process]]]
            overscan = median
    [[standardframe]]
        number = 1
        [[[process]]]
            overscan = median
    [[flatfield]]
        illumflatten = False
        tweak_slits_thresh = 0.9
    [[wavelengths]]
```

(continues on next page)

(continued from previous page)

```
        lamps = HeI, ArI
        sigdetect = 10.0
        rms_threshold = 0.25
    [[slits]]
        maxshift = 0.5
        sigdetect = 50.0
        trace_npoly = 3
    [[tilts]]
        tracethresh = 25.0
[flexure]
    method = boxcar
```

2.2 PypeIt Cookbook

This document gives an overview on how to run PypeIt, i.e. minimal detail is provided. Notes on *Installing PypeIt* are found elsewhere.

We now also provide a set of Slides that provide a more visual step-by-step. Find them here at the PYPEIT HOWTO These should be considered to contain the most up-to-date information.

The following outlines the standard steps for running PypeIt on a batch of data. There are alternate ways to run these steps, but non-experts should adhere to the following approach.

2.2.1 Outline

Here is the basic outline of the work flow. The following is for one instrument in one working directory.

1. Organize/Prepare your data

- Identify folder(s) with raw images
- The raw images can be gzip compressed although the Python FITS reader works much more slowly on gzipped files
- We will refer to that folder as RAWDIR

2. Run the pypeit_setup *without* the –custom option to handle instrument setup.

   Inputs are the path to the raw data with the data prefix (e.g. lrisb) and then one of the PypeIt-approved *Instruments* (e.g. keck_lris_blue, shane_kast_red). Here is an example:

   ```
   pypeit_setup -r /full_path/RAWDIR/lrisb  -s keck_lris_blue
   ```

   This does the following:

   - Generates a setup_files/ folder that holds two files
   - Generates a dummy PypeIt reduction file within the folder [ignore it]
   - Generates a .sorted file which lists files sorted by setup

   You should scan the output WARNING messages for insufficient calibration files (e.g. missing arc frames)

3. Inspect the sorted-file to confirm the expected instrument configuration(s)

- If needed, add more files to your RAWDIR
- If you do, repeat Step 2 above

4. Run pypeit_setup *with* the –custom option

   This produces one folder per setup and a custom *PypeIt Reduction File*. Here is an example of the call:

   ```
   pypeit_setup -r /full_path/RAWDIR/lrisb  -s keck_lris_blue -c=all
   ```

   This generates one folder per setup and a unique *PypeIt Reduction File* file in each folder.

5. Prepare the custom *PypeIt Reduction File* for reducing a given setup

- Enter one of the setup folders (e.g. kast_lris_blue_A)
- Modify the custom *PypeIt Reduction File* as needed
  - trim/add calibration files
  - edit frametypes
  - Modify user-defined execution parameters

6. Run the reduction (described in *Running PypeIt*)

- *run_pypeit* PypeIt_file
- Hope for the best. . . :)

7. Examine QA (*PypeIt QA*)

- When an exposure is fully reduced, a QA file (PDF) is generated in the QA folder
- Examine the output as described in the *PypeIt QA* documentation

8. Examine spectra

- Examine the extracted 1D spectra with *pypeit_show_1dspec*
- Examine the extracted 2D spectra with *pypeit_show_2dspec*

9. Flux
10. Coadd (see *Coadd 1D Spectra*)
11. Repeat steps 5-10 for additional setups, as desired

2.3 PypeIt Reduction File

2.3.1 Overview

The primary file which informs the PypeIt data reduction pipeline is referred to as the PypeIt reduction file and it has a .pypeit extension. This can be generated from PypeIt scripts (*recommended*) or by hand if you are sufficiently familiar with the code.

This document provides guidance on generating and modifying the file.

We *recommend* that you generate a unique PypeIt file for each instrument setup (modulo detectors) or for each mask. It is possible that you will need to modify the settings for different gratings, etc. It will also enable you to more easily customize the associated calibration files to process.

2.3.2 Types

For reference, we distinguish between several types of PypeIt files.

Instrument PypeIt file

For each instrument being reduced in a working folder, the top-level PypeIt file is referred to as an *instrument* PypeIt file. It is intended to be used to generate the instrument setups file and custom PypeIt files for the full reductions.

The standard naming for the instrument PypeIt file is:

```
instrument_date.pypeit
e.g., lris_blue_2016-Nov-23.pypeit
```

Custom PypeIt file

When one performs the full reduction on a set of files for a given setup, the *custom* PypeIt file is used. We refer to it as custom because it may be significantly customized for the specifc instrument configuration and/or target.

While it is possible for a custom PypeIt files to be used on more than one setup grouping, it is not recommended.

A typical naming scheme is by setups, e.g.:

```
lris_blue_setup_A.pypeit
```

although specifying by instrument configuration:

```
kast_blue_600_4310_d55.pypeit
```

or target:

```
kast_blue_3C273.pypeit
```

may be preferable.

2.3.3 pypeit_setup

By default, the pypeit_setup script will generate a set of custom .pypeit files, one per instrument configuration. These will have names like:

```
lris_blue_setup_A.pypeit
```

This is the default because we expect that most users wish to reduce at one time the full set of exposures taken with the same instrument configuration. Of course, one can create other custom .pypeit files.

2.3.4 By Example

For reference, there are existing PypeIt files in PypeIt development suite. The PypeIt development suite is recommended for download (see *Installing PypeIt*), and the relevant PypeIt files are located in:

```
PypeIt-development-suite/pypeit_files/
```

You should be able to find one that matches your instrument.

2.3.5 Line by line

This section describes the various sections of a .pypeit file. In principle, you can use the following description to build a .pypeit file from scratch. This is **not** recommended. The following documentation is mainly for guiding modifications to an existing PypeIt file.

Create a .pypeit file. Name it anything you want, but for example, it's useful to have: the instrument name, the grating or grism used, the dichroic, etc. For example, we could call our PypeIt file 'lris_blue_long_600_4000_d560.pypeit', for our data was collected on LRIS's blue arm, in long slit mode, using the 600/4000 grism and d560 dichroic.

You can make any comments in your PypeIt file with a pound sign:

```
# This is a comment line
```

We *recommend* you separate the main blocks of the .pypeit file with comments.

The first thing to include are changes to the default settings related to running PypeIt. The only one required is to set the name of the spectrograph:

```
run spectograph name_of_your_spectrograph
```

We do recommend including several others, and the .pypeit files made by the pypeit_pypfiles script includes most of the following. Here are ones that one typically sets:

```
# Change the default settings
run ncpus 1                       # number of CPUs to use; can also negative integers,
                                  so -1 means all but one CPU
run spectrograph lris_blue        # the spectrograph (+arm, if necessary) this set of␣
↪data is from;
                                  see README for list of available instruments
output verbosity 2                     # level of screen output; 0 = no output, 1 = low␣
↪level of output;
                                  2 = output everything
output overwrite True                # overwrite any existing output files?
output sorted lris_blue_long_600_4000_d560     # name of output files
```

bias

If you have no bias frames and/or wish to subtract the bias with the overscan region, then set the following:

```
bias useframe overscan
```

Setup block

If a Setup is defined here, the value (e.g. "A" or "D") will be used instead of starting from the default "A" value. But *only* if there is a single Setup in the PypeIt file.

Data block

By Files

This is the recommended approach when performing the full run (as opposed to *pypeit_setup*).

By Path Only

Next, tell PypeIt where your raw data lives! One specifies the full path and may use wild cards to include a set of files. If the data are compressed, include that extension. Multiple entries are allowed

Here is an example:

```
# Read in the data
data read
 /Users/path/to/your/raw/data/*.fits
data end
```

If you wish to skip individual files, you can specify these without the complete path, e.g.:

```
skip LB.20160406.17832.fits
```

These will be ignored as if they didn't exist.

Spect block

Then, give PypeIt some information about your raw data. For example, PypeIt only accepts calibration files if they were created within a time window of the science frame of interest. You can set your own time window here. PypeIt also requires a certain number of each type of calibration file to be matched with the science frame, and here you can set what you want the minimum to be:

```
spect read
 #fits calwin 1000.     # calibration window; default window is 12 hrs;
                        here it is changed to 1000. hrs
 pixelflat number 1       # number of pixel flats needed for data reduction
 bias number 3          # number of bias frames; note that in this case,
                        PypeIt will combine the 3 biases into a master bias
 arc number 1           # number of arcs
 trace number 1         # number of trace frames
spect end
```

In addition to the basic calibration settings above, you may wish to redefine the frametype of a given file. Here are some examples:

```
spect read
 set bias     b150910_2036.fits.gz
 set bias     b150910_2037.fits.gz
 set bias     b150910_2038.fits.gz
 set pixelflat  b150910_2051.fits.gz
 set trace    b150910_2051.fits.gz
 set standard b150910_2083.fits.gz
spect end
```

Whole enchilada

With that, the most basic PypeIt file looks something like this:

```
# Change the default settings
run ncpus 1
run spectrograph lris_blue
```

(continues on next page)

(continued from previous page)

```
output verbosity 2
output overwrite True
output sorted lris_blue_long_600_4000_d560

# Read in the data
data read
 /Users/path/to/your/raw/data/*.fits
data end

spect read
 #fits calwin 1000.

 pixelflat number 1
 bias number 3
 arc number 1
 trace number 1
spect end
```

You can now run PypeIt with this .pypeit settings file! See how in *Running PypeIt*.

2.4 Calibration Check

2.4.1 Overview

We *strongly recommend* that one perform a calibration check with the .pypeit file before proceeding to run the reduction. This verifies that the number of desired calibration files exist and allows the user to examine how the code will group calibration files with science frames. It does **not** check the sanctity of the files nor process the calibrations in any manner.

2.4.2 calcheck

The procedure is simple. Add the following line to your .pypeit file:

```
run calcheck True
```

You must also verify that your .pypeit file does **not** include this line:

```
run setup True   # Cannot be set for calcheck or full reduction
```

Either set 'run setup' to False, comment it out, or remove it altogether.

You may then run PypeIt, e.g.:

```
run_pypeit kast_blue_setup_A.pypeit
```

The code will exit with error if there are insufficient calibration frames. Otherwise, it will exit after organizing the files and will produce a new .group file for your inspection.

You should confirm that the correct number of science and exposure standard files have been identified.

2.4.3 Settings

PypeIt identifies calibration files that are closest in time to every individual science frame. You can place an upper limit on the time window that PypeIt uses to search for calibrations but setting the keyword:

```
fits calwin 12.0
```

which will search for calibrations that were taken within +/-12 hours from a science frame. See docs on *calwin* for a further discussion.

The primary settings you need to specify at this stage are:

1. The number of calibration files required of each frametype
2. Over-ride any frametype designations, as necessary.
3. Modify the method(s) for bias subtraction, flat fielding etc.

For the second issue, see modifying_frametype.

For the first issue see below:

calwin

When associating calibration files to a given science frame, PypeIt will restrict to data within a window in time. This is specified by a *calwin* parameter which has a default value of 12 (hours) for most instruments. One can turn off this restriction by setting the value to 0 in the *Spect block*:

```
fits calwin 0
```

This is the default for *Taking Calibrations for LRISb* and may become the default for all instruments.

Calib number

The user can specify and/or over-ride defaults for the number of calibration frames required by adding a series of lines (or edit the existing ones) in the *Spect block* of the .pypeit file. One line per calibration frametype, as desired. Here is a block one might use for LRISb:

```
# Spect
spect read
 arc number 1
 trace number 5
 bias number 10
 standard number 1
 pixelflat number 3
spect end
```

When a positive, non-zero value is used, the code will require that there be that many calibration frames for each science frame reduced. And, PypeIt will restrict to precisely that many calibration files.

If you wish to use *at least* an input number of frames (and more if they exist), then specify the calibration number with a negative integer value, e.g.:

```
pixelflat number 5
arc number 1
trace number -5
bias number -5
standard number -1
```

2.5 Running PypeIt

This document describes the process to run the reduction. It assumes:

1. You have already properly inspected and fussed with your setups (setup)
2. You have entered one of the setup sub-folders
3. You have *calcheck* on the custom *PypeIt Reduction File* and edited it as needed
4. You have double checked that neither *run calcheck* nor *run setup* are set to True in your custom *PypeIt Reduction File*

See the *PypeIt Cookbook* for additional details.

2.5.1 run_pypeit

The main script to run the PypeIt reduction is *run_pypeit*. It should have been installed in your Python path. Here is its usage:

```
usage: run_pypeit [-h] [-v VERBOSITY] [-m] [-d] [--debug_arc] pypeit_file

##  PypeIt : The Python Spectroscopic Data Reduction Pipeline v0.7.0.dev0
##
##  Available pipelines include:
##   armed, arms
##  Available spectrographs include:
##   isis_blue, lris_blue, kast_red, lris_red, kast_blue
##  Last updated: 07Feb2017

positional arguments:
  pypeit_file            PypeIt reduction file (must have .pypeit extension)

optional arguments:
  -h, --help            show this help message and exit
  -v VERBOSITY, --verbosity VERBOSITY
                        (2) Level of verbosity (0-2)
  -m, --use_masters     Load previously generated MasterFrames
  -d, --develop         Turn develop debugging on
  --debug_arc           Turn wavelength/arc debugging on
```

Of these, only –use_masters is likely to be frequently used by the standard user. This flag will reload masters from the hard-drive if they exist.

Advanced users may run with –develop to have additional logging output provided.

CHAPTER 3

Data Products

3.1 Outputs

PypeIt, despite a pipeline for data *reduction*, is capable of generating an inordinate amount of data products. These pages document the various data products and the means to control the output. A full description of the naming system is described *here*.

3.1.1 Contents

Output Naming

There is no standard for naming data reduction output products in Astronomy, nor even common practices. PypeIt follows its own schema.

A naming system must provide unique names (to avoid overwriting files) but one also desires a format that is both compact and informative. Our approach is a compromise between these competing requirements/desires.

Source Files

This section describes the components of file naming for observed sources (including standard stars).

Prefix

The file type is indicated by its prefix, a short label. The following Table lists all formats for the *Compact* output format of PypeIt. We describe each and include the likely suffix(es).

Prefix	Format	Suffix
spec1D	multi-extension FITS; one binary FITS table per extracted object	.fits
spec2D	multi-extension FITS; one 2D array per spectral image	.fits
qa	Series of figures assessing data reduction and data quality	.pdf

Instrument

The second label indicates the instrument. Here are the set of currently supported instruments in PypeIt:

Instr	Telescope	Short Description	Web Page
kastb	Lick Shane 3m	blue camera of the Kast dual-spectrometer	KastWebSite
kastr	Lick Shane 3m	red camera of the Kast dual-spectrometer	KastWebSite
lrisb	Keck I	blue camera of the LRIS spectrometer	LRISWebSite

Date and Time

By including the UT observing date and time to the nearest second, we believe the filename is now unique. The UT date + time are drawn from the Header and refer to the start of the observation, if there are multiple time stamps. Other DRPs (e.g. LowRedux) have tended to use the Frame number as the unique identifier. We have broken with that tradition: (1) to better follow archival naming conventions; (2) because of concerns that some facilities may not include the frame number in the header; (3) some users may intentionally or accidentally generate multiple raw files with the same frame number.

The adopted format is:

```
YYYYMMMDDTHHMMSS
e.g. 2015nov11T231402
```

A typical filename may then appear as:

```
spec1D_lrisb_2011nov11T231402.fits
```

Source Identifiers

PypeIt reduces each detector separately and associates identified slits and objects to that detector. Therefore, sources are uniquely identified by a combination of these source-id-values. If requested (*Explode*), the Spec1D files can be exploded to yield one FITS file per source. In this case, the filenames are appended by the source identifiers:

```
_DetID_SlitID_ObjID
```

A complete filename may then appear as:

```
spec1D_lrisb_2011nov11T231402_02_783_423.fits
```

For sanity sake, files that are exploded in this manner are placed into their own folders named by the instrument and timestamp.

Calibration Files

The following section describes the components of file naming for calibrations.

Reduction Files

Several file are generated in preparing to run the full reduction and when running PypeIt. These are distinsuished by extension:

Spec1D Output

A primary data product for PypeIt are 1D, calibrated spectra for extracted sources. The most fundamental spectrum may be described by two arrays: flux, wavelength. These together with an error array are the minimal output for even the Quick reduction mode. There are, however, several methods of extraction, calibration, etc. which yield various data products.

Arrays

To allow the inclusion of multiple combinations of arrays, the standard format in PypeIt for spec1D output per object is a binary FITS table. The types of spectral arrays that may be outputted are:

Type	Default Unit	Description	Comments
WAVE	Angstrom	Calibrated wavelenth of each pixel	Vacuum, heliocentric corrected
COUNTS/FLUX	e^- or f_λ	Integrated across the spatial profile	Not normalized by exposure time
VAR/FVAR	$(e^-)^2$ or $(f_\lambda)^2$	Variance in the counts	0 or negative values indicate masked pixels
MASK	–	Bit-wise mask values	See ref for a description
SKY	e^-/pixel or μ	Sky model spectrum	
TRACE	pixel	Best centroid of the object along the detector	

Extractions

Because there are several modes of extraction in PypeIt, there may be multiple outputs of the spectral arrays. These are then prefixed by the extraction mode.

Extraction Mode	Description
BOXCAR	Top-hat extraction around the trace. The precise window used is defined by the BOXCAR_APERTURE, in pixels.
OPTIMAL	Standard Horne algorithm for extraction using the fitted spatial profile. An estimate of this profile is given by OBJ_FWHM

Therefore, the integrated counts for a boxcar extraction are given by the BOXCAR_COUNTS array with variance BOXCAR_VAR.

Additional Parameters

In addition to the spectral arrays, a number of measurements are included in the binary FITS tables. This includes identifiers for the object, which may locate the object on the detector. A complete listing is now given:

Keyword	Type	Description
DET_ID	int	Detector Identifier
SLIT_ID	int	Slit Identifier; given in fractional units of the detector
OBJ_ID	int	Object Identifier; given in fractional units of the slit
RAW_FILE	str	Name of the raw data file

Format

HDF5

PypeIt will generate a single HDF5 file for each science exposure. The HDF5 file contains the groups: header, meta, boxcar and optimal. Each group has its respective datasets:

Group	Description
Meta	Meta is an astropy Table of N rows, corresponding to the N objects/spectra extracted from the exposure. The table contains the RA, DEC, object ID, slit ID, detector number, science index, FWHM (spatial resolution in arcseconds), resolution (spatial resolution in lambda/Dlambda), and xslit.
Header	Header contains the original header information as saved on the telescope.
Boxcar	Boxcar contains N datasets, corresponding to the N objects/ spectra extracted via boxcar extraction.
Optimal	Optimal contains N datasets, correspodning to the N objects/ spectra extracted via optimal extraction. If one of the N objects were not extracted optimally, its dataset will still exist, but be empty.

FITS

If one uses the default *Compact* mode for outputs, a single multi-extension FITS file will be generated that contains the binary FITS tables for each extracted source. To ease access to the individual tables, the FITS header contains the following cards:

Header Card	Type	Example	Description
NOBJ	int	2	Number of extracted sources
ID_####	int	02334223	ID for the source (DET_ID, SLID_ID, OBJ_ID)
S2N_####	float	3.23	Median S/N of of the spectrum

In addition, a reproduction of nearly the entire Header from the raw FITS file is provided, modulo the header cards that describe the data type and size (e.g. NAXIS).

Spec2D Output

During the data reduction proceess, PypeIt creates a series of 2D spectral images prior to extraction of 1D spectra. And, of course, several of these 2D images may have greater value for analysis than the 1D spectra. For each on-source exposure, PypeIt outputs a series of these images, with the number set by the *Reduction Mode*. The following table describes the possible products:

2D Spec Type	Description	Written?
ivar	Inverse variance image; sky+detector	standard, full
mask	Mask image	full
objmodel	Model of the object(s) flux	full
processed	Bias-subtracted, flat-fielded image	full
residual	Residual image; data-model	full
sensitivity	Sensitivity image for fluxing (surface brightness)	full
skysub	Sky-subtracted, processed image	standard, full
skymodel	Model of sky emission on detector	full
waveimg	Wavelength image. Vacuum, flexure, and helio-centric corrected	standard, full

3.1.2 Standard Products

There are four standard types of output products generated by PypeIt. These separate calibrations from spectra and QA plots. Each type is designated by a unique prefix in the filename:

Output Type	Prefix	Description
1D Spectra	spec1d	1D arrays and meta data associated with extracted 1D spectra
Object info	objinfo	ASCII table listing several object attributes
2D Spectra	spec2d	2D arrays related to sources (e.g. sky-subtracted image)
Calibration	MasterFrame	Calibration images, fits, meta files, etc.
Reduction	N/A	Files that guide or describe the reduction
QA	qa	Quality assurance figures

3.1.3 Reduction Mode

PypeIt can be run in several modes of reduction, which demark the level of sophistication (e.g. quick and dirty vs. MonteCarlo) and also the amount of output written to disk. See ReductionModes for a full description of these. The Table below briefly summarizes the standard outputs that are generated by each mode. More detail is given in the documentation describing each type of output products.

Mode	Type	Outputs
Quick	1D Spectra	boxcar (counts), meta
	Object info	all
	2D Spectra	none
	Calibration	meta
	QA	S/N
Minimal	1D Spectra	optimal (fluxed), meta
	2D Spectra	ivar, skysub, waveimg
	Calibration	meta
	QA	flexure, S/N, slits, tilts, tracing, wavelength
Full	1D Spectra	boxcar, optimal, meta
	2D Spectra	ivar, mask, objmodel, processed, residual, skymodel, skysub, waveimg
	Calibration	meta, bias, illumination image, pixel flat, sensitivity fit and image, slit files, tilts image, wavelength fits
	QA	flexure, object profile, S/N, slits, tilts, tracing, wavelength

3.1.4 Compactness

There are two modes for writing the output files which differ in the number of files written.

Compact

Write the fewest files possible, generally one per each of the *Standard Products*. This is the PypeIt default for all *Reduction Mode*.

Explode

Write approximately one file per reduction product.

[Describe how to turn this on]

3.1.5 Organization

[Describe directory structure here.]

3.2 PypeIt QA

As part of the standard reduction, PypeIt generates a series of Quality Assurance (QA) files. This documentation describes the typical outputs, in the typical order that they appear. The basic arrangement is that individual PNG files are created and then a set of HTML files are generated to organize viewing of the PNGs.

3.2.1 HTML

When the code completes (or crashes out), a set of HTML files are generated in the QA/ folder. There is one HTML file per MasterFrame set and one HTML file per science exposure. Example names are MF_A_01_aa.html and J124_J1247-0337_LRISr_2017Mar20T140044.html.

Quick links are provided to allow one to jump between the various files.

3.2.2 Calibration QA

The first QA PNG files generated are related to calibration processing. There is a unique one generated for each setup and detector and (possibly) calibration set.

Generally, the title describes the type of QA and the sub-title indicates the user who ran PypeIt and the date of the processing.

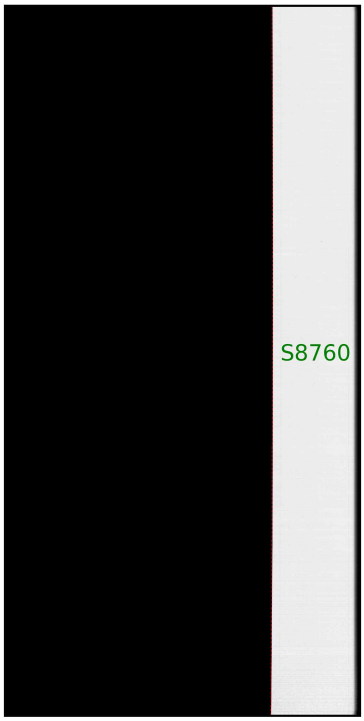
Slit Edge QA

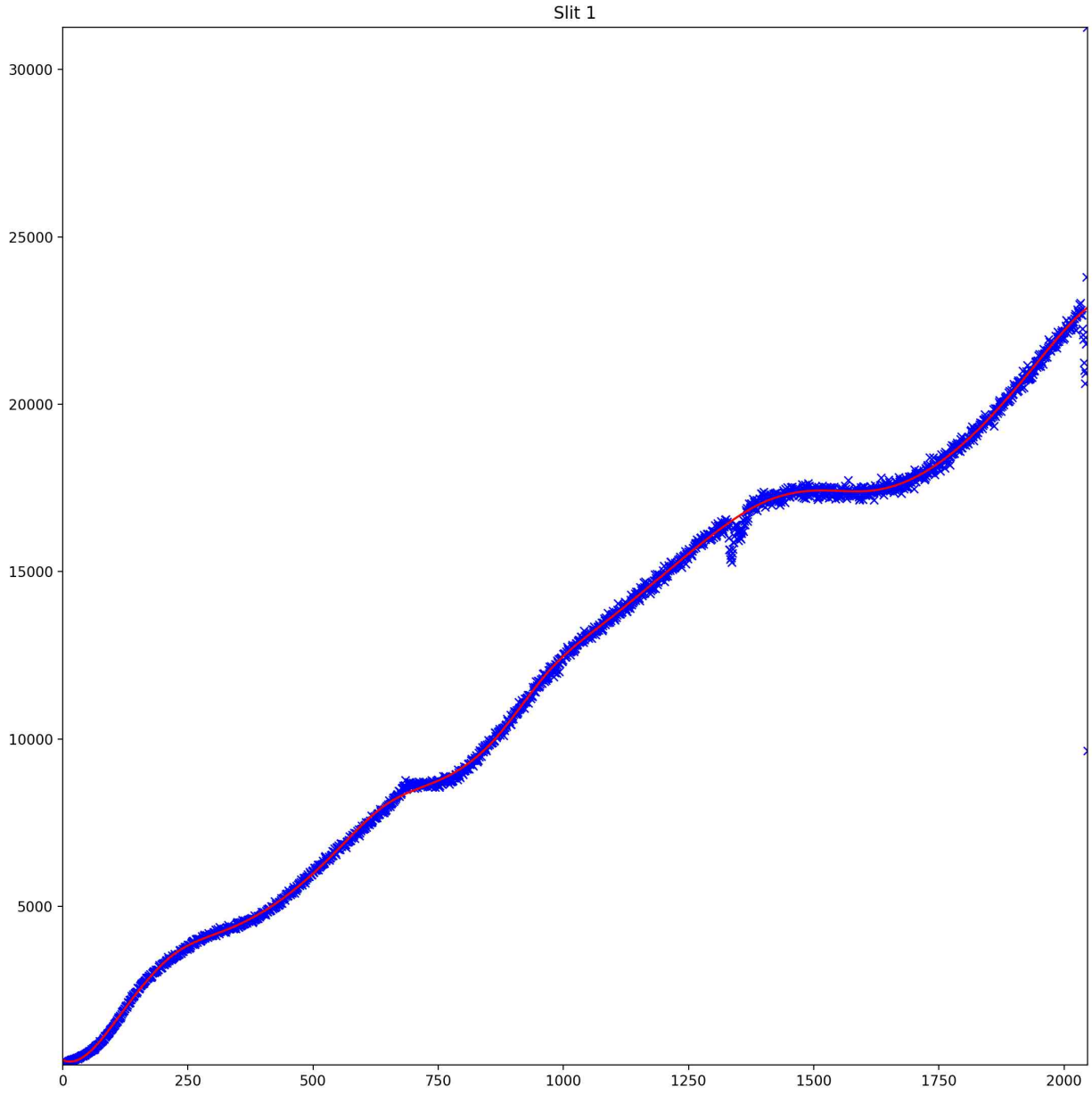
The first output is a plot showing the flat image of the given detector. The left/right slit edges are marked with red/cyan dashed lines. The slit name is labelled in green and the number indicates the position of the slit center on the detector mapped to the range [0-10000]. Here is an example:

Blaze QA

This page shows the blaze function measured from a flat-field image, and the fit to this function. There should be good correspondence between the two. Here is an example:

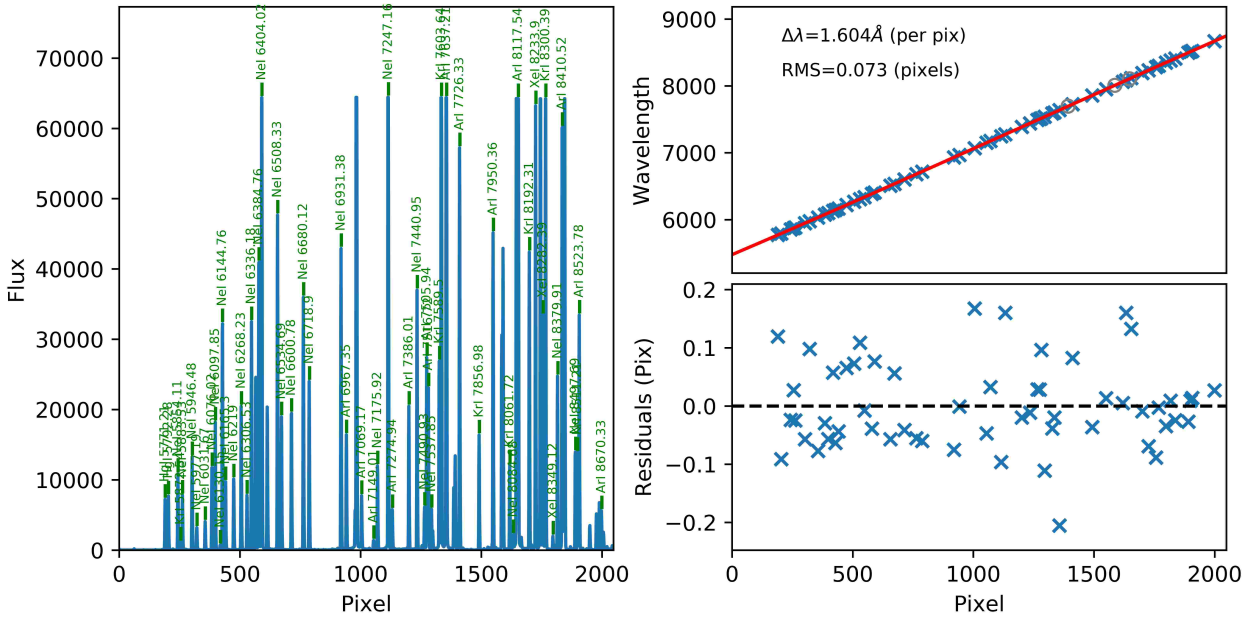
Trace of the slit edges D01
xavier_2017-Jun-11-T07h49m34s




Slit 1
30000
25000
20000
15000
10000
5000
0
250
500
750
1000
1250
1500
1750
2000

Wavelength Fit QA

This page shows the arc spectrum with labelled arc lines in the left panel and the fit and residuals to the fit in the right panels. Good solutions should have RMS < 0.1 pixels. Here is an example:

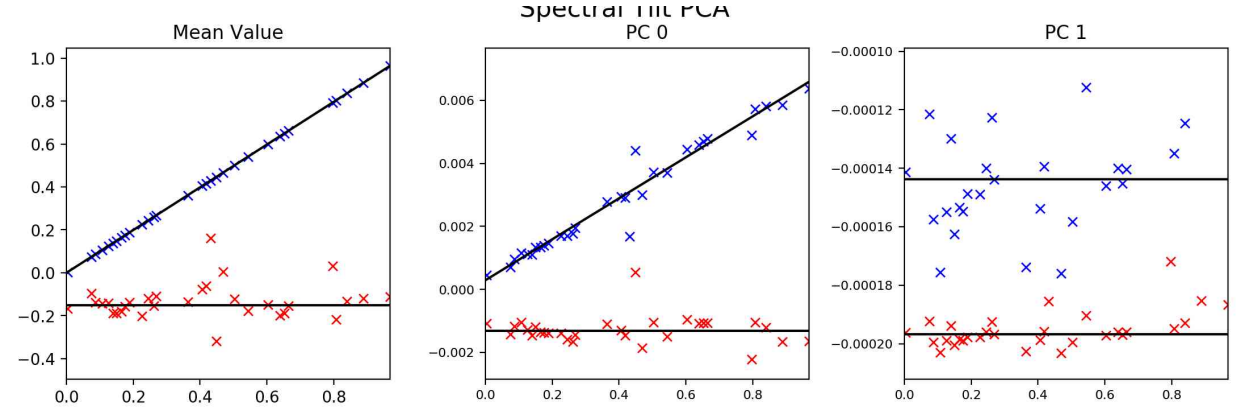


Spectral Tilts QA

There are generally a series of PNG files describing the analysis of the tilts of the arc lines.

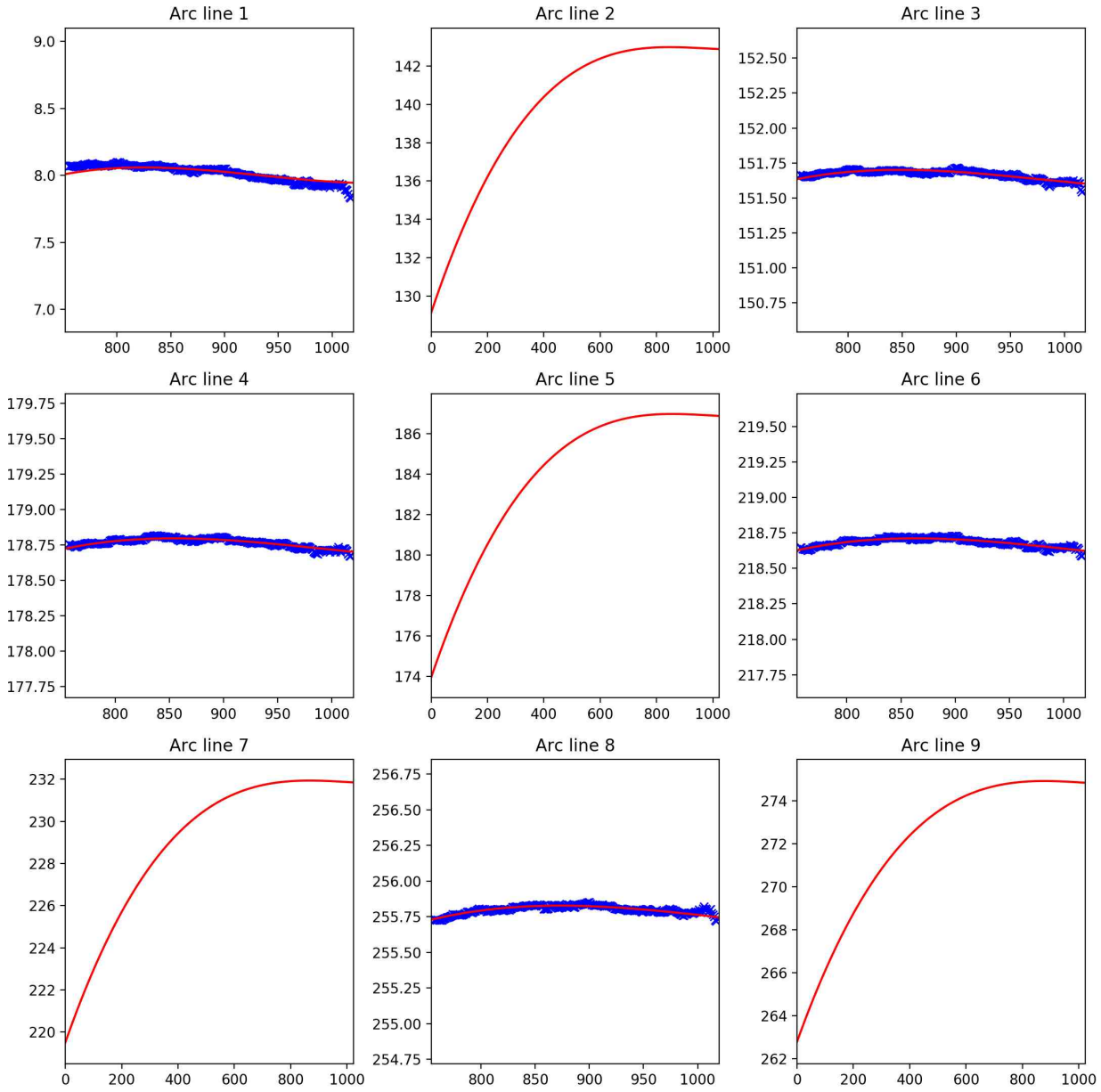
Arc Tilt PCA

One page should show fits to the PCA components describing the arcline tilt fits. One hopes for good models to the data (blue crosses; red crosses indicate lines ignored in the analysis) in the first two panels, and that the values for PC1 are small. Here is an example:



Arc Tilt Plots

There are then a series of PNG files showing the arc lines across the detector (blue) and associated fits (red). Here is an example page:

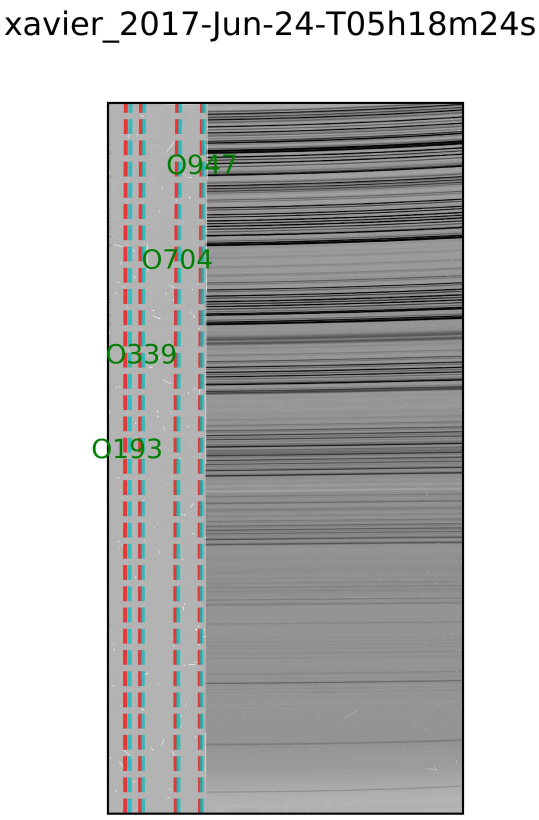


3.2.3 Exposure QA

For each processed, science exposure there are a series of PNGs generated, per detector and (sometimes) per slit.

Object Trace QA

An image of the sky-subtracted slit is displayed. Overlayed are the left/right (red/cyan) edges of the extraction region for each object. These are also labeled by the object ID value where the 3-digit number is the trace position relative to the slit, ranging from 0-1000. Here is an example:



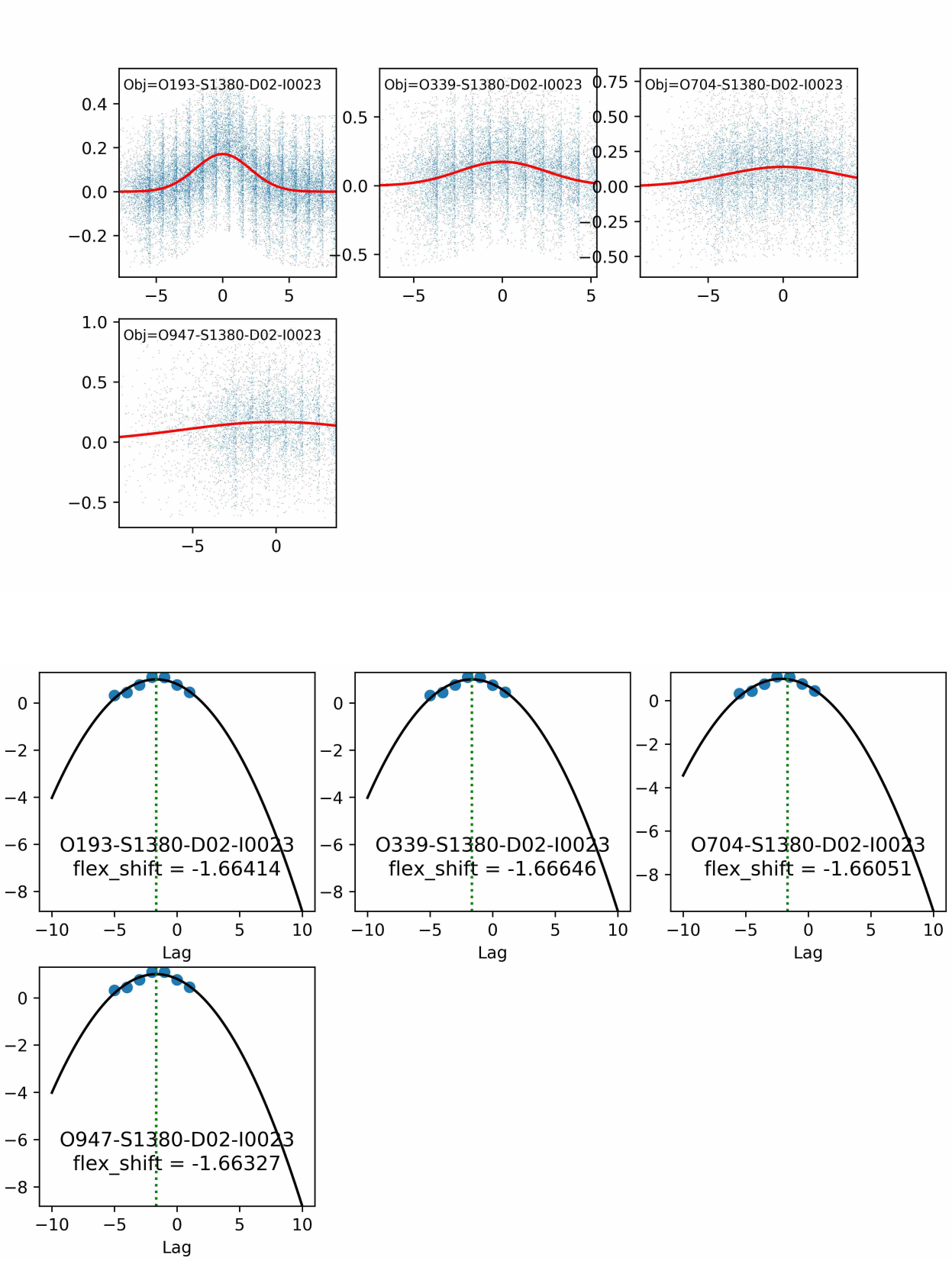
Object Profile QA

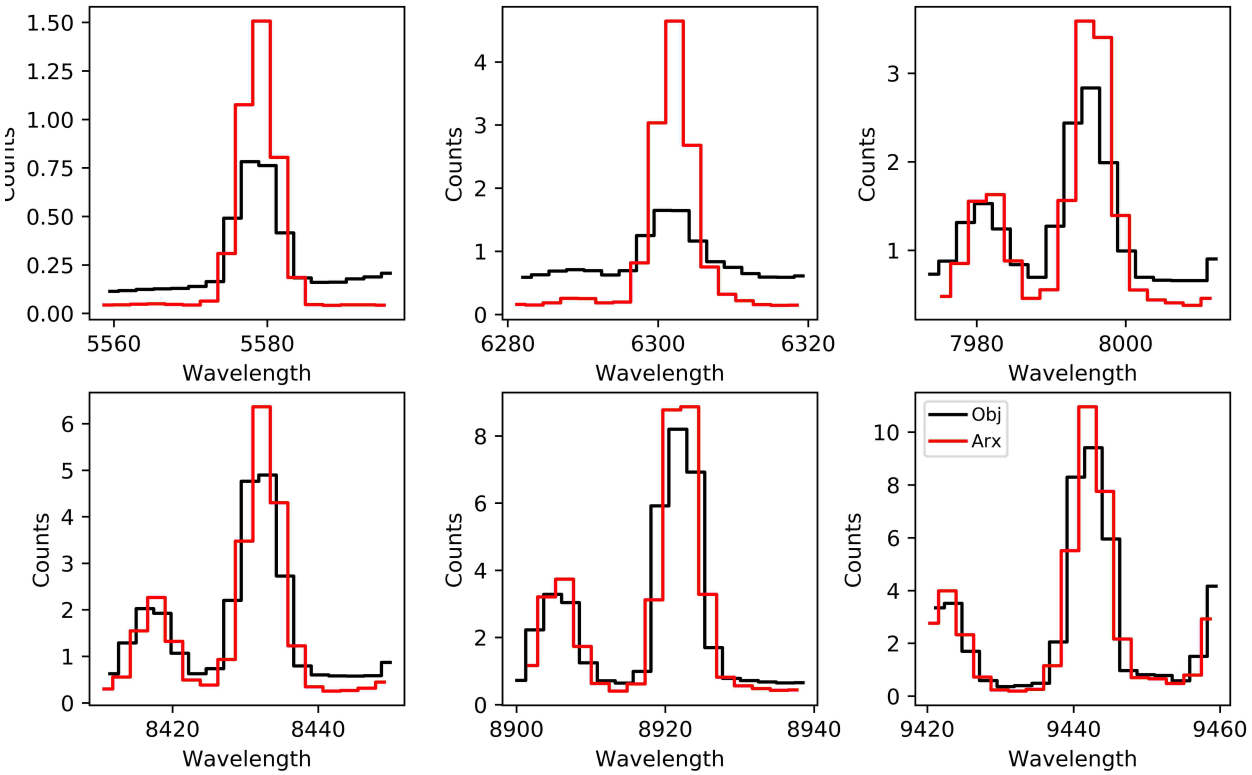
For all of the objects in a given slit where optimal extraction was performed the spatial profile and the fit are displayed. The x-axis is in units of pixels. Here is an example:

Flexure QA

If a flexure correction was performed (default), the fit to the correlation lags per object is shown and the adopted shift is listed. Here is an example:

There is then a plot showing several sky lines for the analysis of a single object (brightest) from the data compared against an archived sky spectrum. These should coincide well in wavelength. Here is an example:


Obj=O193-S1380-D02-I0023
Obj=O339-S1380-D02-I0023
Obj=O704-S1380-D02-I0023
Obj=O947-S1380-D02-I0023
O193-S1380-D02-I0023
flex_shift = -1.66414
O339-S1380-D02-I0023
flex_shift = -1.66646
O704-S1380-D02-I0023
flex_shift = -1.66051
O947-S1380-D02-I0023
flex_shift = -1.66327
Lag


Counts
Wavelength
Obj
Arx

CHAPTER 4

Calibrations

4.1 Calibration Overview

This document gives an overview of the calibration steps of PypeIt. There is additional documentation for several of the more complex steps.

4.1.1 Sequence of Events

Here is the sequence of events:

Step	Products	Description
datasec	datasec image	2D image describing the detector pixels for analysis

4.1.2 datasec

In this step, PypeIt parses the instrument settings file (or user input) to establish the region on each detector for analysis. The overscan section is also established and is included in the datasec if one exists.

A 2D image defining the datasec pixels is generated and stored internally (in _datasec).

Standard

The standard approach to defining the datasec is to set these in the instrument settings file. It is necessary to generate one set per amplifier as each of these may have distinct properties (e.g. readnoise, gain).

Here are the values for Kast blue:

```
det01 numamplifiers 2                # Number of amplifiers
det01 datasec01 [:,0:1024]
det01 oscansec01 [:,2049:2080]
```

(continues on next page)

(continued from previous page)

```
det01 datasec02 [:,1024:2048]
det01 oscansec02 [:,2080:2111]
det01 gain 1.2,1.2                    # Inverse gain (e-/ADU)
det01 ronoise 3.7,3.7                 # Read-out noise (e-)
```

LRIS

The FITS file writing for LRIS is sufficiently complex that the datasec definition (and file loading) is guided by a custom method: arlris.read_lris()

4.2 Bias Subtraction

4.2.1 Overview

The code can perform bias subtraction using input bias frames or by analyzing and subtracting off an estimate from the overscan region(s). The default for ARMLSD is to use bias frames and require that several be provided. A future implementation will combine the two approaches.

4.2.2 Methods

Subtract Bias Frame

This method combines the set of input bias frames and subtracts the resulting MasterBias from all other frames. This is the default in ARMLSD when bias frames are provided. It can be explicitly enforced by adding:

```
bias useframe bias
```

to the PypeIt reduction file.

Subtract Overscan

Analyze the overscan region and subtract a replicated version of the result from the full image. This method is applied by adding:

```
bias useframe overscan
```

to the PypeIt reduction file.

Overscan Algorithms

SavGol, Polynomial

4.3 Slit Tracing

One of the first and most crucial steps of the pipeline is to auto-magically identify the slits (or orders) on a given detector. This is a challenging task owing to the wide variety in:

- the number of slits/orders,
- the separation between slits/orders (if any)
- the varying brightness of flats across the detector

Developing a single algorithm to handle all of these edge cases (pun intended) is challenging if not impossible. Therefore, there are a number of user-input parameters that one may need to consider when running PypeIt (see below).

Underlying the effort is the TraceSlits class which can be used to load the Master frame output for tracing (a FITS and a JSON file).

4.3.1 Algorithm

Here is the flow of the algorithms.

1. A Sobolev S/N image is generated from the trace flat image
2. edge detection: An initial set of edges are derived from the Sobolev according to the trace-slit-threshold.
3. match edges: An algorithm is performed to match edges into slits for the first time.
4. trace (crudely) the slits: Each slit edge is traced with the trace_crude algorithm and snippets of edges are (hopefully) merged
5. PCA: A PCA analysis is performed of the well-traced edges found thus far. This is then used to rectify the Sobolev images and search for additional edges.
6. synchronize: Slit edges are synchronized primarily to pick up missing edges
7. trim: Trimming of small slits is performed

4.3.2 Open Issues

1. Bad columns yield fake edges. These should be masked out by the pipeline using the instrument-specific bad pixel mask.
2. Overlapping slits are notoriously difficult to detect. One may need to add/subtract individual slits on occasion.

4.3.3 Reduction Mode

Longslit

If you have only one slit per detector, you may wish to specify the *Number of Slits* as 1.

Multislit

Deriving all of the slits in a mask exposure is challenged by overlapping slits, slits that run to the detector edge, bad columns, etc. Our testing with DEIMOS and LRIS masks is thus far recovering ~95% of the slits.

It is highly recommended that you inspect the warning messages during slit tracing and then pause the code to inspect the MasterTrace output using the *pypeit_chk_edges* script.

We now summarize the PypeIt parameters that are occasionally varied to improve slit tracing.

One parameter to consider first is the trace-slit-threshold which sets the minimum S/N ratio in the Sobolev filter image for an edge to be detected. You may inspect these edges with the *pypeit_chk_edges* and –show=edgearr. The left edges in the Sobolev are the white regions in this image and the black regions (negative values) are the right edges.

The green/red traces show the left/right edges detected from this image; these are *not* the final traces. Inspect the positive/negative values of the edges in the Sobolev image and lower/raise trace-slit-threshold accordingly.

If your spectra span only a modest fraction (~50%) of the detector in the spectral direction, you may need to: (1) Reduce the value of trace-slit-mask_frac_thresh and maybe also: (2) Modify the range for smashing the Sobolev image with trace-slit-smash_range.

Add User Slits

The code may be instructed to add slits at user-input locations. The syntax is is a list of lists, with each sub-list having syntax (all integers): det:y_spec:x_spat0:x_spat1 For example:

```
[calibrations]
  [[slits]]
    add_slits = 2:2000:2121:2322,3:2000:1201:1500
```

The above will add one slit on detector 2 with left/right edge at 2121/2322 at row 2000. The shapes of the slit will be taken from the ones that are nearest.

See the PypeIt-HOWTO-slits slides for further details.

Remove Slits

The code may be instructed to remove slits at user-input locations. The syntax is a list of lists, with each sub-list having syntax (all integers): det:y_spec:x_spat For example:

```
[calibrations]
  [[slits]]
    rm_slits = 2:2000:2121,3:2000:1500
```

This will remove any slit on det=2 that contains x_spat=2121 at y_spec=2000 and similarly for the slit on det=3.

See the PypeIt-HOWTO-slits slides for further details.

Echelle

The primary difference currently between multi-slit and echelle is that the latter analyzes the left and right edges separately during the PCA algorithm.

4.3.4 Scripts

pypeit_chk_edges

PypeIt includes a simple script to show the processed Trace image and the slit/order edges defined by the algorithm. These are displayed in a Ginga viewer. Here is an example call:

```
pypeit_chk_edges MF_keck_lris_blue/MasterTrace_A_1_01
```

If debugging poor performance, you can show other outputs from intermediate steps in the process with the –show command:

```
--show=edgeearr  # Shows the edges derived early on from the Sobolev image
--show=xset      # Shows the edges derived after the mslit_tcrude() method
--show=siglev    # Shows the Sobolev S/N image
```

4.3.5 Trace Slit Settings

The following are settings that the user may consider varying to improve the slit tracing.

Number of Slits

Ironically, one of the more challenging slit configurations to automatically identify is a single slit. In part this is often because at least one edge of the slit butts against the detecor giving no image gradient. And also because only a small portion of the detector may be illuminated by this 'long' slit.

Therefore, when reducing long slit data, it may be a good idea to explicitly tell PypeIt that there is only 1 slit to be identified. You can set this using the keyword:

```
[calibrations]
  [[slits]]
    number=1
```

You can also use this variable to specify the number of slits that should be detected. Note, that this feature works best when you have well-defined and uniformly illuminated slits (usually the case with cross-dispersed data, for example).

Detection Threshold

The detection threshold for identifying slits is set relatively low to err on finding more than fewer slit edges. The algorithm can be fooled by scattered light and detector defects. One can increase the threshold with the *sigdetect* parameter:

```
[calibrations]
  [[slits]]
    sigdetect = 30.
```

Then monitor the number of slits detected by the algorithm.

Presently, we recommend that you err on the conservative side regarding thresholds, i.e. higher values of sigdetect, unless you have especially faint trace flat frames.

On the flip side, if slit defects (common) are being mistaken as slit edges then *increase* sigdetect and hope for the best.

Fraction Threshold

In an interemediate step, the mslit_tcrude() method, the edges defined thus far are traced across the detector with the trace_crude method. A PCA analysis of these is then performed on those edges which spanned at least mask_frac_thresh of the detector in the spectral direction. The default value is 0.6 which may be too large for some instruments (e.g. LRISb with the 300 grism). Consider lowering the value, especially if the code raised a warning on too few edges for the PCA:

```
[calibrations]
  [[slits]]
    mask_frac_thresh = 0.45
```

You may also need to adjust the trace-slit-smash_range parameter.

Smash Range

One of the final steps in slit/order definition is to identify edges by smashing a rectified version of the Sobolev image. The default is to smash the entire image, but if the spectra are primariliy in a subset of the image one should consider modifying the default parameter. This is frequently the case for low-dispersion data, e.g. LRISb 300 grism spectra (which has a different default value). Modify it as such:

```
[calibrations]
  [[slits]]
    smash_range = 0.5,1.
```

4.3.6 Slit Profile

DEPRECATED

With relatively short slits (often the case with multiobject or echelle data), the sky background is determined from relatively few pixels towards the edge of the slit, where the flux from a uniformly illuminated slit tends to roll off. To correct for this effect, PypeIt models the spatial slit profile of a trace frame (i.e. a flatfield with the same slit length as the science slit). The relevant set of parameters that determine the fit properties are given by:

```
reduce slitprofile perform False
reduce flatfield method bspline
reduce flatfield params [n]
```

where n in the last line should be an integer or floating point number.

The default setting is to not calculate the slit profile. To turn on this functionality, the argument of the first line above can be set to True. If the calculation is performed, the second line sets the method that should be used to determine the spatial slit profile.

At this stage, PypeIt only supports the value 'bspline', where the knot spacing is set by the third line above. If the argument of reduce flatfield params is n >= 1, PypeIt will place a knot at every n pixels. Otherwise, if n < 1, PypeIt will place a knot at every k pixels, where k=n*N and N is the total number of pixels in the spectral direction. The number of knots in the spatial direction is set automatically by PypeIt, to be twice the number of pixels along the slit. Thus, the user only has the ability to change the number of knots in the spectral direction (i.e. the blaze function). If the spatial slit profile is not calculated, the blaze function will still be calculated using the 'reduce flatfield' settings listed above.

4.3.7 Tips on Trace Flat Frames

The slit edges are traced using a "trace" frame. If neighboring slits are very close together, you can use a "pinhole" frame to trace the slit centroid.

In the current version of PypeIt, pinhole frames are only used for echelle data reduction. Pinhole frames are usually an exposure of a quartz lamp through a very short (pinhole) slit. Thus, neighboring slit edges of a pinhole frame should be well separated.

Trace frames, on the other hand, usually have the same slit length as the science frame. In cases where neighboring slits are very close together, it is necessary to first define the slit centroid using a pinhole frame, and the slit edges are defined using a trace frame by "expanding" the slits, by giving the following keyword argument:

```
trace slits expand True
```

This has been developed for the APF primarily.

4.3.8 For Developers

One of the ways the edge-finding algorithm is fooled is via chip defects, e.g. bad columns. It is therefore valuable to mask any such known features with the bad pixel mask when one introduces a new instrument (or detector).

4.4 Wavelength Calibration

4.4.1 Basic Algorithms

These notes will describe the algorithms used to perform wavelength calibration in 1D (i.e. down the slit/order) with PypeIt. The basic steps are:

1. Extract 1D arc spectra down the center of each slit/order
2. Load the parameters guiding wavelength calibration
3. Generate the 1D wavelength fits

The code is guided by the WaveCalib class, partially described by this WaveCalib.ipynb Notebook.

For the primary step (#3), we have developed several algorithms finding it challenging to have one that satisfies all instruments in all configurations. We now briefly describe each and where they tend to be most effective. Each of these is used only to identify known arc lines in the spectrum. Fits to the identified lines (vs. pixel) are performed with the same, iterative algorithm to generate the final wavelength solution.

Holy Grail

This algorithm is based on pattern matching the detected lines with that expected from the lamps observed. It has worked well for the low dispersion spectrographs and has been used to generate the templates needed for most of the other algorithms. It has the great positive of requiring limited developer effort once a vetted line-list for the observed lamps has been generated.

However, we have found this algorithm is not highly robust (e.g. slits fail at ~5-10% rate) and it struggles with high dispersion data (e.g. ThAr lamps). At this stage, we recommend it be used primarily by the Developers to generate template spectra.

Reidentify

Following on our success using archived templates with the LowRedux code, we have implemented an improved version in PypeIt. Each input arc spectrum is cross-correlated against one or more archived spectra, allowing for both a shift and a stretch.

Archived spectra that yield a high cross-correlation score are used to identify arc lines based on their recorded wavelength solutions.

This algorithm is optimal for fixed-format spectrographs (e.g. X-Shooter, ESI).

Full Template

This algorithm is similar to *Reidentify* with two exceptions: (i) there is only a single template used (occasionally one per detector for spectra that span across multiple, e.g. DEIMOS); (ii) IDs from the input arc spectrum are generally performed on snippets of the full input array. The motivation for the latter is to reduce non-linearities that are not well captured by the shift+stretch analysis of *Reidentify*.

We recommend implementing this method for multi-slit observations, long-slit observations where wavelengths vary (e.g. grating tilts). We are likely to implement this for echelle observations (e.g. HIRES).

4.4.2 Common Failure Modes

Most of the failures should only be in MultiSlit mode or if the calibrations for Echelle are considerably different from expectation.

As regards Multislit, the standard failure modes of the *Full Template* method that is now preferred are:

1. The lamps used are substantially different from those archived.
2. The slit spans much bluer/redder than the archived template.

In either case, a new template may need to be generated. If you are confident this is the case, raise an Issue.

4.4.3 Possible Items to Modify

FWHM

The arc lines are identified and fitted with ane expected knowledge of their FWHM (future versions should solve for this). A fiducial value for a standard slit is assume for each instrument but if you are using particularly narrow/wide slits than you may need to modify:

```
[calibrations]
  [[wavelengths]]
    fwhm=X.X
```

in your PypeIt file.

4.4.4 Line Lists

Without exception, arc line wavelengths are taken from the **'NIST database <http://physics.nist.gov/PhysRefData'_**, *in vacuum*. These data are stored as ASCII tables in the *arclines* repository. Here are the available lamps:

Lamp	Range (A)	Last updated
ArI	3000-10000	21 April 2016
CdI	3000-10000	21 April 2016
CuI	3000-10000	13 June 2016
HeI	2900-12000	2 May 2016
HgI	3000-10000	May 2018
KrI	4000-12000	May 2018
NeI	3000-10000	May 2018
XeI	4000-12000	May 2018
ZnI	2900-8000	2 May 2016
ThAr	3000-11000	9 January 2018

In the case of the ThAr list, all of the lines are taken from the NIST database, and are labelled with a 'MURPHY' flag if the line also appears in the list of lines identified by Murphy et al. (2007) MNRAS 378 221

4.4.5 By-Hand Calibration

If the automatic algorithm is failing (heaven forbid; and you should probably raise an Issue on PypeIt if you are sure it isn't your fault), you can input a set of pixel, wavelength values as a crutch in your .pypeit setup file. Here is the recommended approach:

1. Run PypeIt with –debug_arc on. This will force the code to stop inside ararc.py
2. Print the pixel values to the screen
    - (Pdb) tcent
3. Plot the arc spectrum.
    - (Pdb) plt.plot(yprep)
    - (Pdb) plt.show()
4. Compare that spectrum with a known one and ID a few lines. Write down. Better be using vacuum wavelengths
5. Add pixel values and wavelengths to your .pypeit file, e.g.
    - arc calibrate IDpixels 872.062,902.7719,1931.0048,2452.620,3365.25658,3887.125
    - arc calibrate IDwaves 3248.4769,3274.905,4159.763,4610.656,5402.0634,5854.110

4.4.6 Flexure Correction

By default, the code will calculate a flexure shift based on the extracted sky spectrum (boxcar). See *Flexure Correction* for further details.

4.4.7 Wavelength Frame

PypeIt offers several frames of reference that can used for the wavelength scale. The first choice is whether you would like the data to be calibrated to air or vacuum wavelengths. This option is controlled by the argument:

```
reduce calibrate wavelength air
```

where the default value is to calibrate to vacuum. You can also specify 'pixel', which will save the pixel values instead of the wavelength values (i.e. a wavelength calibration will not be performed). The calibration follows the Ciddor schema (Ciddor 1996, Applied Optics 62, 958).

You can also choose if you want the wavelength scale corrected to the heliocentric (Sun-centered), barycentric (Solar system barycentre), or topocentric (telescope centered). None is also an option, but this defaults to topocentric. This option is governed by the command:

```
reduce calibrate refframe barycentric
```

where the default value is a heliocentric wavelength scale. More details are provided in heliocorr.

4.4.8 Developers

Full Template

The preferred method for multi-slit calibration is now called *full_template* which cross-matches an input sepctrum against an archived template. The latter must be constructed by a Developer, using the core.wavecal.templates.py module. The following table summarizes the existing ones (all of which are in the data/arc_lines/reid_arxiv folder):

Instrument	Setup	Name
keck_deimos	600ZD grating, all lamps	keck_deimos_600.fits
keck_deimos	830G grating, all lamps	keck_deimos_830G.fits
keck_deimos	1200G grating, all lamps	keck_deimos_1200G.fits
keck_lris_blue	B300 grism, all lamps	keck_lris_blue_300_d680.fits
keck_lris_blue	B400 grism, all lamps?	keck_lris_blue_400_d560.fits
keck_lris_blue	B600 grism, all lamps	keck_lris_blue_600_d560.fits
keck_lris_blue	B1200 grism, all lamps	keck_lris_blue_1200_d460.fits
keck_lris_red	R400 grating, all lamps	keck_lris_red_400.fits
keck_lris_red	R1200/9000 , all lamps	keck_lris_red_1200_9000.fits
shane_kast_blue	452_3306 grism, all lamps	shane_kast_blue_452.fits
shane_kast_blue	600_4310 grism, all lamps	shane_kast_blue_600.fits
shane_kast_blue	830_3460 grism, all lamps	shane_kast_blue_830.fits

See the Templates Notebook or the core.wavecal.templates.py module for further details.

One of the key parameters (and the only one modifiable) for *full_template* is the number of snippets to break the input spectrum into for cross-matchging. The default is 2 and the concept is to handle non-linearities by simply reducing the length of the spectrum. For relatively linear dispersers, nsinppet=1 may frequently suffice.

For instruments where the spectrum runs across multiple detectors in the spectral dimension (e.g. DEIMOS), it may be necessary to generate detector specific templates (ugh). This is especially true if the spectrum is partial on the detector (e.g. the 830G grating).

4.4.9 Validation

See the iPython Notebook under test_suite for a comparison of the wavelength solution for PypeIt vs. LowRedux.

4.5 Wavelength Tilts

4.5.1 Overview

To construct a wavelength image that assigns a wavelength value to every pixel in the science frame, one must measure the tilts of the arc lines (or sky lines) across the slits/orders.

This process is organized by the WaveTilts class which is primarily a wrapper to methods in the artracewave.py module. Here is the code flow:

1. Extract an arc spectrum down the center of each slit/order
2. Loop on slits/orders

i. Trace the arc lines (fweight is the default)

ii. Fit the individual arc lines

iii. 2D Fit to the offset from pixcen

iv. Save

See this WaveTilts Notebook for some examples.

4.5.2 QA

The code will output a residual plot of the 2D fit to offsets. It should be possible to achieve an RMS < 0.05 pixels.

4.5.3 Scripts

pypeit_chk_tilts

This script displays several aspects of the tilts solution on the Arc frame. Here is the usage:

```
usage: pypeit_chk_tilts [-h] [--slit SLIT] option setup

Display MasterArc image in a previously launched RC Ginga viewer with tilts

positional arguments:
  option       Item to show [fweight, model, tilts, final_tilts]
  setup        setup (e.g. A_01_aa)

optional arguments:
  -h, --help   show this help message and exit
  --slit SLIT  Slit/Order [0,1,2..] (default: None)
```

And here is an example or two:

```
pypeit_chk_tilts fweight A_01_aa --slit 0
pypeit_chk_tilts model A_01_aa --slit 0
pypeit_chk_tilts tilts A_01_aa --slit 0
```

These will displace in a RC Ginga window.

4.5.4 Settings

IdsOnly

Limit tilt analysis to only the arc lines identified in 1D wavelength solution:

```
trace slits tilts idsonly True
```

This is critical when using instrument with a significant number of ghosts (e.g. LRISb).

Threshold

Minimum amplitude of an arc line for analysis. The default is 1000 (counts). You may wish to lower this parameter to include more lines, especially if you are short on lines near the spectral edges of the slit/order, e.g.:

```
trace slits tilts trthrsh 400.
```

We may eventually tune this parameter for the various instruments.

Order

Order of the function (default is Legendre) that is fit to each arc line across the slit/order. Very long slits will likely require order=3 or higher, e.g.:

```
trace slits tilts order 3
```

The default is 1 which may be raised.

4.6 Flat fielding

4.6.1 Overview

PypeIt corrects pixel-to-pixel variations using input pixelflat frames or by loading a pre-made master pixelflat. The default approach is to use pixel flat frames and require that several be provided.

4.6.2 Methods

If you are confident that pixel-to-pixel variations do not need to be corrected for your data, you can turn off the flat fielding correction with the argument:

```
reduce flatfield perform False
```

Alternatively, set this argument to 'True' (the default option) to perform the correction. To load a predefined file, use the command:

```
reduce flatfield useframe filename
```

where filename is the name of the file to be used for the flatfield correction. Alternatively, this command also accepts 'pixelflat' or 'trace' in place of 'filename'. Recall that a trace frame is typically an exposure of a quartz lamp through the same slit as the science expsoure, and a pixelflat frame is typically an exposure of a quartz lamp through a slit that is longer than that taken for the science frame.

If you opt to use a set of flat frames that you have taken for the flat field correction, the current implementation normalizes the combined input frames with a bspline:

```
reduce flatfield method bspline
```

Each method takes a set of parameters, which are supplied with the keyword:

```
reduce flatfield params [20]
```

bspline

The bspline method takes a single parameter which, if >= 1, corresponds to the spacing between knots in the spectral direction, in units of pixels. If the supplied parameter value is less than 1, PypeIt assumes that this represents a fraction of the pixels in the spectral direction, and will use this as the knot spacing. The default value is 0.05.

4.6.3 Blaze information

The blaze functions that are derived from one of the methods listed above are saved by PypeIt. If desired, you can perform a simple 2D PCA on the blaze models. This step is only recommended (but not necessary) for echelle data reduction, where the blaze functions of neighbouring slits are quite similar. A 2D PCA will not be performed if the argument of the following keyword is set to zero:

```
reduce flatfield 2dpca 0
```

A number greater than zero will result in a PCA fit to the blaze functions. The argument of this keyword sets the number of principal components to use when reconstructing the blaze functions.

4.7 Fluxing

4.7.1 Overview

Fluxing is done after the main run of PypeIt using a separate input file modeled after the PypeIt file. This file sets the main parameters of the run and guides the process. See *Example File* for a complete example file.

The top of the file sets fluxing parameters. The spectrograph must *always* be set:

```
[rdx]
  spectrograph = vlt_fors2
```

See the FluxCalib ParSet documentation for other parameters that guide generation of the sensitivity function or the fluxing operation.

WARNING: The code only allows for a select set of standards.

Sensitivity Function

If you wish to generate a sensitivity function from an input standard star file, then you need to set std_file and sensfunc:

```
[fluxcalib]
   std_file = spec1d_STD_vlt_fors2_2018Dec04T004939.578.fits
   sensfunc = bpm16274_fors2.fits
```

The former specifies the spec1d spectrum file produced by PypeIt for the standard star. The latter specifies the output file name, which will be overwritten if need be.

Fluxing

To flux one or more spec1d files, generate a *flux read*, e.g.:

```
flux read
  spec1d_UnknownFRBHostY_vlt_fors2_2018Dec05T020241.687.fits FRB181112_fors2_1.fits
  spec1d_UnknownFRBHostY_vlt_fors2_2018Dec05T021815.356.fits FRB181112_fors2_2.fits
  spec1d_UnknownFRBHostY_vlt_fors2_2018Dec05T023349.816.fits FRB181112_fors2_3.fits
flux end
```

The first entry of each row is the spec1d file to be fluxed and the second provides the output filename. One separates the two entries by *a single space*!

4.7.2 Flux Spec Script

It may be preferential to flux the spectra after the main reduction (i.e. run_pypeit). PypeIt provides a script to guide the process. Here is the usage:

```
pypeit_flux_spec FRB181112.flux -h
usage: pypeit_flux_spec [-h] [--debug] [--plot] [--par_outfile] flux_file

Parse

positional arguments:
  flux_file      File to guide fluxing process

optional arguments:
  -h, --help     show this help message and exit
  --debug        show debug plots?
  --plot         Show the sensitivity function?
  --par_outfile  Output to save the parameters
```

The parameters used to guide the process are written to par_outfile (default = fluxing.par) and –plot will generate a simple plot of the sensitivity function.

4.7.3 FluxSpec Class

The guts of the flux algorithms are guided by the FluxSpec class. See the FluxSpec.ipynb Notebook on GitHub (in doc/nb) for some usage examples, although we recommend that most users use the *Flux Spec Script*.

4.7.4 Example File

Here is a complete example file:

```
# User-defined fluxing parameters
[rdx]
   spectrograph = vlt_fors2
[fluxcalib]
   balm_mask_wid = 12.
   #std_file = spec1d_STD_vlt_fors2_2018Dec04T004939.578.fits
   sensfunc = bpm16274_fors2.fits

flux read
  spec1d_UnknownFRBHostY_vlt_fors2_2018Dec05T020241.687.fits FRB181112_fors2_1.fits
  spec1d_UnknownFRBHostY_vlt_fors2_2018Dec05T021815.356.fits FRB181112_fors2_2.fits
  spec1d_UnknownFRBHostY_vlt_fors2_2018Dec05T023349.816.fits FRB181112_fors2_3.fits
flux end
```

Note the std_file is commented out to avoid remaking the sensitivity function.

4.7.5 Sensitivity Function

PypeIt uses the CALSPEC calibration database, which can be found at http://stsci.edu/hst/observatory/crds/calspec.html for flux calibrations, specifically, generating the sensitivity function (see also standards).

The sensitivity function is generated by dividing the standard star's flux, which is loaded in by PypeIt from CALSPEC, by the standard star's counts per second. This is then multiplied to the science object's counts per second to yield a fluxed science spectrum.

The sensitivity function is written to disk as a YAML file in the MasterFrames folder with prefix MasterSensFunc. There is only one file per setup (not per detector). If one has a previous file, this can be placed in the MasterFrames folder to be loaded (one must turn on MasterFrame usage, e.g. with the -m flag on run_pypeit).

4.7.6 Fluxing Output

Science

The resulting fluxed science spectrum, f_λ, is given in units of $10^{-17}\,\mathrm{ergs/s/cm^2/Angstrom}$ and is stored in the 'box_flam' extension of the extracted 1D spectrum. If an optimal extraction was successful, there also exists an 'opt_flam' extension in the 1D spectrum.

Standard

The 1D extracted standard spectrum is also saved as an output of the fluxing routine. The counts and fluxed standard spectrum are available in the 'box_counts' and 'box_flam' extensions, respectively. The fluxed spectrum saved here is the fluxed standard, using the sensitivity function generated from itself (rather than the archived fluxed standard star loaded from CALSPEC), and can be examined and compared to the expected f_λ as a sanity check.

4.7.7 Troubleshooting

Problem with bspline knot

Things sometimes go wrong the fluxing and it commonly has to do with the bspline algorithm. If you reach a stop in the code with a message that says "Problem with bspline knot" there are a couple things to check:

- There are instances where there isn't data between the knots. You can change the knot spacing by including the following in your .pypeit file under the Reduce block:

```
reduce skysub bspline everyn NUM
```

  where you adjust NUM.

- If your observation of the standard star is taken with a setup that goes beyond the wavelength range of the version in data/standards/calspec.
- If the wavelength solution is really bad it can manifest as problem in bspline knot. If the issue isn't the spacing or wavelength coverage check the QA files to see if there is an issue in the wavelength solution. If this is the case, check the *Wavelength Calibration* page for Troubleshooting or open an issue on the GitHub repo.

CHAPTER 5

Instruments

5.1 Instruments

5.1.1 Overview

Below we describe all of the spectrographs that may be reduced by PypeIt. We also provide any suggested tips for customizing the PypeIt file.

PypeIt Name	Telescope	Instrument
shane_kast_blue	Lick 3m	Kast dual spectrometer; blue camera
shane_kast_red	Lick 3m	Kast dual spectrometer; red camera
shane_kast_red_ret	Lick 3m	Kast dual spectrometer; red reticon
keck_lris_blue	Keck	LRIS spectrometer; blue camera
keck_lris_red	Keck	LRIS spectrometer; red camera
keck_lris_red_longonly	Keck	LRIS spectrometer; red camera windowed
keck_nirspec_low	Keck	NIRSPEC spectrometer; low-dispersion
keck_deimos	Keck	DEIMOS spectrometer
gemini_gnirs	Gemini	GNIRS spectrometer
wht_isis_blue	WHT	ISIS spectrometer; blue camera?
vlt_fors2	VLT	FORS2 spectrometer; only a few gratings
vlt_xshooter_uvb	VLT	X-Shooter spectrometer; UVB camera
vlt_xshooter_vis	VLT	X-Shooter spectrometer; VIS camera
vlt_xshooter_nir	VLT	X-Shooter spectrometer; NIR camera
tnb_dolores	TNG	DOLORES (LRS) spectrograph; LR-R

5.1.2 Kast

5.1.3 LRIS

See the *Keck LRIS* specific notes for more.

5.1.4 DEIMOS

See the *Keck DEIMOS* specific notes for more.

5.1.5 X-Shooter

See the xshooter specific notes for more.

5.2 Keck DEIMOS

5.2.1 Overview

This file summarizes several instrument specific settings that are related to the Keck/DEIMOS spectrograph.

5.2.2 Deviations

Here are the deviations from the default settings for DEIMOS (set in the settings.keck_deimos file):

```
settings trace slits sigdetect 50.0
settings trace slits number -1
settings trace slits tilts params 1,1,1
settings trace slits tilts method spca
settings trace slits pca params [3,2,1,0]
settings trace slits polyorder  3
settings trace slits sobel mode nearest
settings trace slits fracignore  0.02   # 0.02 removes star boxes of 40pix size or␣
↪less (and any real ones too!)
settings bias useframe overscan
settings pixelflat combine method median
settings pixelflat combine reject level [10.0,10.0]
```

These are tuned to the standard calibration set taken with DEIMOS. Note that the *fracignore* setting is designed to remove alignment star boxes from the analysis. If you have real slits which are the same size (or smaller) they too will be eliminated.

5.3 Keck LRIS

5.3.1 Overview

This file summarizes several instrument specific settings that are related to the Keck/LRIS spectrograph.

5.3.2 Longslit

If reducing data with a longslit, we recommend that you specify that only a single slit is desired, i.e.:

```
trace slits number 1
```

See *Number of Slits* for further details.

5.3.3 Taking Calibrations for LRISb

Default Settings

Here are the deviations from the default settings for LRISb:

```
settings trace dispersion direction 0
settings trace slits tilts method spca
settings trace slits tilts params 1,1,1
settings trace slits pca params [3,2,1,0]
settings trace slits sigdetect 30.0        # Good for Twilight flats; faint dome␣
↪flats might fail miserably..
```

The last setting is fine for a relatively bright frame taken on the twilight sky, but we suspect a faint dome flat on the blue side will require a lower sigdetect (and is likely to be very challenging overall).

Internal flats, meanwhile, may be too bright and need to be tested.

Pixel Flat

It is recommend to correct for pixel-to-pixel variations using a slitless flat. If you did not take such calibration frames or cannot process them, you may wish to use an archival. If so, copy the file into your MasterFrame folder (should be named MF_lris_blue and you may need to create it yourself) and set the following in the _reduce-block of the PypeIt file:

```
reduce flatfield useframe MF_lris_blue/PypeIt_LRISb_pixflat_B600_2x2_17sep2009.fits.gz
```

5.3.4 Taking Calibrations for LRISr

Default Settings

Here are the deviations from the default settings for LRISr:

```
settings trace slits sigdetect 50.0   # Good for relatively bright dome flats
settings trace slits pca params [3,2,1,0]
```

Known issues

Multi-slit

The code may identify a 'ghost' slit in empty detector real estate if your mask does not fill most of the field. Be prepared to ignore it.

5.4 Magellan Mage

5.4.1 Overview

This file summarizes several instrument specific settings that are related to Magellan/Mage.

5.4.2 Short slits

There are several issues related to the very short slits of Magellan/Mage (34 pixels or 10" unbinned).

Find Objects

To have enough slit to 'properly' find objects, we restrict the find_trim_edge parameter, i.e.:

```
par['scienceimage']['find_trim_edge'] = (4,4)    # Slit is too short to trim 5,5␣
↪especially with 2x binning
```

For spatial binning, we recommend you to further reduce this by the binning factor.

CHAPTER 6

Object Algorithms

6.1 Object Finding

This document describes how the code identifies objects within the slits/orders.

6.1.1 Overview

Object identification is a challenging process to code, especially to allow for a large dynamic range between bright continuum sources and faint emission line sources. Our general philosophy has been to err on the faint side, i.e. detect sources aggressively with the side-effect of including false positives.

6.1.2 Algorithms

Each of the algorithms described below attempt to identify the peak location of objects in the slit and then defines a left and right edge for each source. The codes also define background regions for sky subtraction.

standard

The standard algorithm performs the following steps:

1. Rectify the sky-subtracted frame
2. Smooth this 2D image
3. Perform sigma clipping (median stat) down the wavelength dimension to further reject CRs. This may eliminate bright emission lines.
4. Smash the 2D image along the spectral dimension, to get a 1D array that represents the spatial profile of the exposure.
5. Perform an initial search for objects by fitting a low-order polynomial to the spatial profile and associate objects with pixels that are deviant with that fit.

6. Estimate the scatter in the slit array and then define all 5 sigma, positive excursion as objects (with 3 sigma edges).
7. Eliminate any objects within a few percent of the slit edge. Parameterized by *trace object xedge*.
8. Determine edges and background regions for each object.
9. Optional: Restrict to maximum number of input objects, ordered by flux.

nminima

The image is rectified and smashed along the spectral dimension as in the steps above. Then the following steps are performed:

1. The 1D array is smoothed by a Gaussian kernel of width *trace object nsmooth* (default=3).
2. Keep all objects satisfying the threshold criterion. The default is to compare against the scatter in the sky background. One can keep objects relative to the brightest object (NOT YET IMPLEMENTED).
3. Eliminate any objects within a few percent of the slit edge. Parameterized by *trace object xedge*.
4. By default, the code restricts to a maximum of 8 objects.
5. Determine edges and background regions for each object.

By-hand

6.1.3 Parameters

The following parameters refer to the prefix of *trace object* and refer to options for finding the object(s) in a slit.

Parameter	Algorithm	Options	Description
find	N/A	standard,nminima	Algorithm to use for finding objects
nsmooth	nminima	int; default=3	Parameter for Gaussian smoothing when the nminima algorithm is used
xedge	Any	float; default=0.03	Ignore any objects within xedge of the edge of the slit. One may lower this value to recover an object very close to the edge.

6.2 Object Tracing

This document describes how the code traces each object found within a slit.

6.2.1 Parameters

The following table describes parameters related to tracing the objects down the slit. These parameters also follow a prefix of *trace object*.

Parameter	Options	Description
order	int; default=2	Order of the polynomial function to be used to fit the object trace in each slit.
function	polynomial,legendre, chebyshev	Function to be used to trace the object in each

6.3 Coadd 1D Spectra

This document will describe how to combine the 1D spectra from multiple exposures of the same object.

PypeIt currently only offers the coadding of spectra in 1D and must be done outside of the data reduction pipeline, i.e. PypeIt will *not* coadd your spectra as part of the data reduction process.

The current defaults use the Optimal extraction and fluxed data.

6.3.1 Coadd 1dspec

The primary script is called *pypeit_coadd_1dspec* and takes an input YAML file to guide the process. Here is the usage:

```
wolverine> pypeit_coadd_1dspec -h
usage: pypeit_coadd_1dspec [-h] [--debug] infile

Script to coadd a set of spec1D files and 1 or more slits and 1 or more
objects. Current defaults use Optimal + Fluxed extraction. [v1.1]

positional arguments:
  infile      Input file (YAML)

optional arguments:
  -h, --help  show this help message and exit
  --debug     Turn debugging on
```

Turning on debugging will generate a series of diagnostic plots and likely hit as set_trace in the code.

6.3.2 Input File

The information PypeIt's coadder uses is contained within a .yaml file. At the most basic level, the file must include the names of the files to be coadded, and a series of dicts, labeled by 'a', 'b', 'c', etc., each of which has a PypeIt object identifier string (used to ID the object) and the name of an output file. Here is an example case:

```
'spectrograph': 'shane_kast_blue'
'filenames': ['spec1d_1.fits', 'spec1d_2.fits', 'spec1d_3.fits']
'a':
    'object': 'O503-S4701-D01-I0035'
    'outfile': 'tmp.hdf5'
```

The default behavior of the coadder is to use one object identifier string for all the files to be coadded. There are hard coded tolerance values in PypeIt (10 for the object identifier string and 50 for the slit identifier string) that work to find the same object across all the specified files. However, if the object changes positions along the slit over the exposures (e.g., you dithered while observing the object) this might not be the best way to coadd since the object identifier string

could be very different from exposure to exposure. For this case, there is functionality to specifiy an object identifier string for each specified file. The .yaml file would look like this:

```
'spectrograph': 'shane_kast_blue'
'filenames': ['spec1d_1.fits', 'spec1d_2.fits', 'spec1d_3.fits']
'a':
    'object': ['O290-S1592-D02-I0002', 'O457-S1592-D02-I0003
    ', 'O626-S1592-D02-I0004']
    'outfile': 'tmp.hdf5'
```

There is only one object to be coadded in each data frame. The 'object' tag is a object identifier string containing the object's relative location in the slit (here, 503 with 1000 the right edge), the slit ID which is relative on the detector (4701), the detector number (01), and the science index (0035), in one of the files.

One can also set local parameters for coadding. Common keywords for coadding algorithms are listed below (*More Keywords*).

The list of object identifiers in a given spec1d file can be output with the pypeit_show_1dspec script, e.g.:

```
pypeit_show_1dspec filename.fits --list
```

These can also be recovered from the object info files in the Science/folder (one per exposure).

The coadding algorithm will attempt to match this object identifier to those in each data file, within some tolerance on object and slit position. 'outfile' is the filename of the coadded spectrum produced.

6.3.3 Spectral Parameters

By default, the algorithm will combine the optimally extracted, fluxed spectra from each exposure. You may modify the extraction method, e.g.:

```
'extract': 'box'
```

and/or specify whether the spectrum is fluxed:

```
'flux': False
```

Note that these parameters must be outside of the 'a', 'b', 'c', etc. dicts or else they will have no effect.

6.3.4 Flux Scaling

Each entry can include a *scale* dict that will be used to scale the flux of the coadded spectrum using an input filter and magnitude. Here is an example:

```
'a':
    'object': ['SPAT0119-SLIT0000-DET01', 'SPAT0159-SLIT0000-DET01', 'SPAT0079-
↪SLIT0000-DET01']
    'outfile': 'FRB181112_fors2.fits'
    'scale': {'filter': 'DES_r', 'mag': 21.73, 'mag_type': 'AB', 'masks': [[0., 6000.
↪]]}
```

The call here will convolve the coadded spectrum with the DES r-band filter, and then scale the flux to give an AB magnitude of 21.73. Furthermore, the spectral wavelengths less than 6000 Ang are masked in the analysis.

Filters

Here is the set of ingested filters:

DES_g, DES_r, DES_i DES_z, DES_Y

6.3.5 Cosmic Ray Cleaning

By default, the script will attempt to identify additional, lingering cosmic rays in the spectrum. The algorithm employed depends on the number of input spectra. Note that most of the challenges associated with the coadding are related to CR identification, especially for cases of only two input spectra.

The main parameters driving the CR algorithms are described in *Cosmic Ray*.

Two Spectra

While it is possible to clean a significant fraction of any lingering CR's given 2 exposures, results are mixed and depend on the S/N ratio of the data and the presence of strong emission lines. We have now implemented three approaches, described below.

The default is *bspline* which is likely best for low S/N data. The algorithm may be modified with the cr_two_alg parameter.

diff

This algorithm compares the difference between the spectra and clips those that are *cr_nsig* away from the standard deviation.

ratio

Similar to *diff* above, but the ratio is also compared. This may be the best algorithm for high S/N data with strong emission lines.

bspline

A b-spline is fit to all of the pixels of the 2 spectra. By default, a breakpoint spacing of 6 pixels is used. Very narrow and bright emission lines may be rejected with this spacing and a lower value should be used (see *Cosmic Ray*). Of course, lowering the spacing will increase the likelihood of including cosmic rays. This algorithm is best suited for lower S/N spectra.

Three+ Spectra

For three or more spectra, the algorithm derives a median spectrum from the data and identifies cosmic rays or other deviant pixels from large deviations off the median.

6.3.6 Additional Coadding Parameters

You can adjust the default methods by which PypeIt coadds spectra by adding a dict named 'global' or a 'local' dict in the object block:

```
'spectrograph': 'shane_kast_blue'
'filenames': ['spec1d_1.fits', 'spec1d_2.fits', 'spec1d_3.fits']
'global':
    'wave_grid_method': 'velocity'
'a':
    'object': 'O503-S4701-D01-I0035'
    'outfile': 'tmp.hdf5'
    'local':
        'otol': 10
```

The adjustable parameters and options are:

Wavelength Rebinning

Parameter	Option	Description
wave_grid_method	default: concatenate	create a new wavelength grid onto which multiple exposures are rebinned after first concatenating all wavelength grids
–	velocity	create a new wavelength grid of constant km/s. Default is to use the median velocity width of the input spectrum pixels but a value 'v_pix' can be provided
–	pixel	create a new wavelength grid of constant Angstrom specified by the input parameter 'A_pix'

Flux Scaling

Parameter	Option	Description
scale_method	default: auto	scale the flux arrays based on the root mean square value (RMS) of the S/N^2 value for all spectra; if this RMS value is less than the minimum median scale value, no scaling is applied. If the RMS value is greater than the minimum but smaller than the maximum median scale value, the applied method is the median, as described below
–	hand	scale the flux arrays using values specified by the user in the input parameter 'hand_scale'. Must have one value per spectrum
–	median	scale the flux arrays by the median flux value of each spectra

Cosmic Ray

Parameter	Option	Description
cr_everynint; default=6		For CR cleaning of 2 spectra, this sets the spacing of the b-spline break points. Use a lower number to avoid clipping narrow emission/absorption lines, e.g. 4
cr_nsig	float; default=7.	Number of sigma which defines a CR
cr_two_algstr; default=bspline		Algorithm to adopt for cleaning only 2 spectra

More Keywords

Here are other keywords that one may wish to set for individual objects:

Keyword	Method	Type	Description
otol	arspecobj.mtch_obj_to_objects()	int	Tolerance for matching object ID number

6.3.7 Running the Coadd Code

Once you have this .yaml file set up, you can coadd your 1d spectra by running the command:

```
pypeit_coadd_1dspec name_of_yaml_file.yaml
```

The coadder will also produce a quality assurance (QA) file named 'root_of_outfile.pdf'. In the left panel, the QA shows the chi- squared residuals of the coadded spectrum, and in the right panel, the coadded spectrum (in black) is plotted over the original spectra.

CHAPTER 7

Documentation

7.1 Flexure Correction

This document will describe how a flexure correction is performed for each 1D spectrum extracted in PypeIt.

7.1.1 Overview

By default, the code will calculate a flexure shift based on the extracted sky spectrum (boxcar). A cross-correlation between this sky spectrum and an archived spectrum is performed to calculate a single, pixel shift. This is then imposed on the wavelength solution with simple linear interpolation.

The general approach is to compare the sky model from the observation with an archived sky model. Generally, by default, the Paranal sky spectrum is used, as derived from the SDSS codes. The default is different for Kast blue and LRIS blue where sky_kastb_600.fits and sky_LRISb_600.fits are respectively used (see *Alternate sky models* for all sky models).

7.1.2 Algorithm

The basic algorithm may be summarized as follows:

1. Identify the overlapping wavelength range between data and archived sky.
2. Rebin the archived sky spectrum onto the overlapping wavelength range.

3. Smooth the sky spectrum to the resolution of the data, if the archive has higher spectral resolution (preferred). 4. Normalize each spectrum to unit average sky counts 5. Subtract a bspline continuum from each 6. Perform a cross-correlation 7. Fit the cross-correlation with a parabola to find center 8. Apply shift

7.1.3 Usage

By default in ARMLSD, a flexure correction is performed on the boxcar extraction of the sky. This may be disabled by the following setting in the .pypeit file:

```
reduce flexure spectrum None
```

One can alternatively use the optimal extraction (if it is performed) with:

```
reduce flexure spectrum optimal
```

By default, the maximum shift allowed in pixels is 20. If you suspect a higher shift is required (e.g. results are poor), you may increase the default (e.g. to 50 pixels):

```
reduce flexure maxshift 50
```

7.1.4 Alternate sky models

You may find that the default sky models are not the best suited for your data.There is a script that allows the user to plot the extracted sky spectrum for their data against any of the sky models in the PypeIt archive. To use this script:

```
pypeit_compare_sky <Name of 1D spectrum> <Name of sky model>
```

As noted above, the Paranal sky model is the default reference. Presently, we are finding that the sky spectrum at Mauna Kea (measured with LRIS) is sufficiently variable and dark that a robust solution is challenging. Fair results are achieved by using the instrument-specific sky spectra in the LowRedux package. The best practice currently is to use the one that best matches as an optional parameter

You can use a different sky model than the default by placing the following line under the ‘’Reduce” block in your .pypeit file:

```
reduce flexure spectrum <Name of sky model>
```

The models supplied with PypeIt are,

7.1.5 Other

An alternate algorithm (activated with: reduce flexure spec slit_cen) measures the flexure from a sky spectrum extracted down the center of the slit. This is then imposed on the wavelength image so that any extractions that follow have a flexure correction already applied. Thus far, this algorithm has given poorer results than the default.

7.2 Frame Type

7.2.1 Overview

Every raw data file ingested by PypeIt is automatically assigned one or more frametype values. This is to separate calibration files, science frames, etc. The assignments are guided by criteria given in the default settings file for each spectrograph (e.g. setttings.kast_blue). One should not modify the default files but if you have a suggestion for improvement consult with the PypeIt authors.

7.2.2 Definitions

Here are the frametype values allowed and adopted in PypeIt:

Frame-type	Description
arc	Spectrum of one or more calibration arc lamps
bias	Bias frame; typically a 0s exposure with the shutter closed
dark	Dark frame; typically a >0s exposure to assess dark current (shutter closed)
pinhole	Spectrum taken through a pinhole slit (i.e. a very short slit length), and is used to define the centre if a slit (currently, this frame is only used for echelle data reduction). Often this is an exposure using a flat lamp, but one can in principle use a standard star frame too (or a science frame if the spectrum is uniform).
pixelflat	Spectrum taken to correct for pixel-to-pixel detector variations Often an exposure using a flat lamp, but for observations in the very blue, this may be on-sky
science	Spectrum of one or more science targets
standard	Spectrum of spectrophotometric standard star PypeIt includes a list of pre-defined standards
trace	Spectrum taken to define the slit edges and correct for illumination variations across the slit. Often this is an exposure using a flat lamp, but for observations in the very blue, this may be on-sky. The slit length of a trace frame should be the same as the science slit.
unknown	File could not be automatically identified by PypeIt

It is possible, and for flats common, that a frame can be assigned more than one frametype. .. _modify_frametype:

7.2.3 Auto-typing

PypeIt will, be default, attempt to auto identify the image type based on Header information. For each instrument, there are hard-coded conditions in the settings.instrument file that guide the process. Here are the conditions for a *trace* frame with the Shane Kast blue camera:

```
trace check condition1␣
↪lampstat01=on|lampstat02=on|lampstat03=on|lampstat04=on|lampstat05=on
trace check condition2 exptime>0  # Required for bias
```

The syntax uses "|" and "&" for logic and the strings refer to short-hand strings that were taken from the FITS header. It is unlikely that anyone other than a developer will wish to modify any of these conditions.

Regarding science vs. standard star typing (perhaps the most challening aspect), the code takes any source that satisfies the standard conditions to be a standard if it lies with 20arcmin of the PypeIt approved list of standards.

A file that satisfies all conditions of being a *bias* frame yet has an exposure time exceeding the minimum value for the detector is typed as a *dark*.

7.2.4 Modifying a frametype

data block

If your PypeIt reduction file includes the file-by-file listing of frames to analyze, you can edit the frametype directly in the appropriate column. The values in the .pypeit file will over-ride any assessed by the code. This is the recommend approach for standard users.

spect block

One can specify one or more frametype's for any file in the .pypeit file. Include one or more lines in the *Spect block* with syntax *set frametype filename*, e.g.:

```
set pixelflat b150910_2051.fits.gz
```

This will over-ride the automatic assignment by PypeIt.

7.3 Instrument Settings

This document will detail aspects of the instrument settings files used in PypeIt.

These are mainly notes for the lead developers.

7.3.1 Generating a new settings file

Here is a quick cookbook of the steps involved:

- Update Mosaic properties (e.g. lon, lat)
- Update Detector properties * RN, GAIN are hard-coded to match detector
- Update checks (note: white spaces are removed in this check) * CCD name * You must check NAXIS is 2 in "checks to perform".
- Update Keyword identifiers

Examine the base set of keywords in the data/settings/settings.basespect file and update for the instrument as necessary. Here are some of the standard ones:

```
keyword target 01.OBJECT                # Header keyword for the name given by the␣
↪observer to a given frame
keyword idname 01.OBSTYPE               # The keyword that identifies the frame type␣
↪(i.e. bias, flat, etc.)
keyword time 01.MJD-OBS                 # The time stamp of the observation (i.e.␣
↪decimal MJD)
keyword date 01.DATE-OBS                # The date of the observation (in the format␣
↪YYYY-MM-DD  or  YYYY-MM-DDTHH:MM:SS.SS)
keyword equinox None                    # The equinox to use
keyword ra 01.RA                        # Right Ascension of the target
keyword dec 01.DEC                      # Declination of the target
keyword airmass 01.AIRMASS              # Airmass at start of observation
keyword naxis0 01.NAXIS2                # Number of pixels along the zeroth axis
keyword naxis1 01.NAXIS1                # Number of pixels along the first axis
keyword exptime 01.EXPTIME              # Exposure time keyword
```

- Update FITS properties
  - timeunit refers to the format of the time KEYWORD (e.g. mjd)
  - We should give a few examples here
- Fiddle with rules for Image type ID. Below are some helpful guidelines
  - Again, check the settings.basespect file first
  - Common check or match rules to update include

    * arc match decker any – One frequently uses a narrow slit for arcs
    * xxx match dispangle |<=## – Add if your disperser has a variable angle
  - If a keyword is specified in science/pixflat/blzflat/trace/bias/arc frames it must also appear in the Keyword identifiers list.
  - If a keyword value contains only some interesting value, you can split the keyword value using the '%,' notation. For example, suppose you have the string 10:50:23.45, and you're interested in the 50 for a match condition, you would use '%' to indicate you want to split the keyword value, ':' indicates the delimiter text, '1' indicates you're interested in the 1st argument (0-indexed), '<60' is an example criteria. Each of these should be specified in this order, separated by commas, so the final string would be: %,:,1,<60 If you want to split on multiple delimiters, separate them with a logical or operator. For example, if you want to split a string at the characters ':' and '.', you would use the expression %,:|.,1,<60
  - If the text 'I' appears in the match condition, the absolute value will be taken. For example '|<=0.05' means that a given keyword's value for a calibration frame must be within 0.05 of a science frame's value, in order to be matched.
  - If a keyword's value contains spaces, replace all spaces with one underscore.
  - If the header contains two keyword's of the same name, only the value of the first one will be recognised.
- Run
- Add arc solution * set debug['arc'] = True in run_pypeit
- Add extinction file if a new observatory * Add file in data/extinction * Edit README
- Add test suite

7.4 Internals

7.4.1 Overview

This file contains notes related to the internal workings of PypeIt.

7.4.2 Objects

filesort

This *dict* whose keys are the various *Frame Type* used in PypeIt and the items are arrays of the indices of the frames with that type.

Generated by arsort.sort_data()

setup_dict

This highly nested *dict* organizes the various setups used in the input set of data files. The top-level keys, which define the setup, are simple labels: A, B, C, ...

The next set of keys are:

- '–': which holds a *dict* defining properties of the setup (e.g. dichroic)
- '01': a *dict* holding detector specific info (e.g. binning)
- 'aa', 'ab', 'ac', etc: are *dict*'s containing lists of filenames as a function of frame type

Here is an example (as output to the .setups file):

```
A:
  --:
    dichroic: d55
    disperser: {angle: None, name: 600/4310}
    slit: {decker: 0.5 arcsec, slitlen: None, slitwid: None}
  '01': {binning: None, det: 1, namp: 2}
```

Usually generated by arsetup.instr_setup()

setup_ID

The setup_ID is then commbines the keys of the *setup_dict*, e.g. **A_01_aa**

CHAPTER 8

For Developers

8.1 PypeIt scripts

PypeIt is packaged with several scripts that should have been installed directly into your path (e.g. ~/anaconda/bin).

8.1.1 Pipeline Scripts

pypeit_setup

This setups files for data reduction. See setup for details

run_pypeit

This is the main executable for PypeIt. See *Running PypeIt* for details.

8.1.2 Inspecting Data

The following scripts are inspecting the data products produced by PypeIt.

pypeit_show_1dspec

Wrapper around the linetools XSpecGUI. Grabs a single 1D spectrum from the PypeIt spec1d output and runs:

```
unix> pypeit_show_1dspec -h
 usage: pypeit_show_1dspec [-h] [--list] [--exten EXTEN] [--extract EXTRACT] [--obj␣
↪OBJ] file

 Parse
```

(continues on next page)

(continued from previous page)

```
 positional arguments:
   file           Spectral file

 optional arguments:
   -h, --help         show this help message and exit
   --list             List the extensions only?
   --exten EXTEN      FITS extension
   --obj OBJ          Object name in lieu of extension, e.g. O424-S1466-D02-I0013
   --extract EXTRACT  Extraction method. Default is boxcar. ['box', 'opt']
```

pypeit_show_2dspec

This script displays the sky-subtracted 2D image for a single detector in a Ginga RC viewer. It also overlays the slits and any objects extracted. It should be called from the reduction directory, i.e. above the Science folder where the spec2d image is located. Here is the usage:

```
unix> pypeit_show_2dspec -h
usage: pypeit_show_2dspec [-h] [--list] [--det DET] file

Display spec2d image in a Ginga viewer

positional arguments:
  file        PypeIt spec2d file

optional arguments:
  -h, --help  show this help message and exit
  --list      List the extensions only? (default: False)
  --det DET   Detector (default: 1)
```

The script can be called multiple times to load multiple detectors into one Ginga viewer.

pypeit_view_fits

This is a wrapper to the Ginga image viewer. It is a bit of a kludge in that it writes a dummy tmp.fits file to the harddrive and sends that into Ginga. The dummy file is deleted afterwards.:

```
unix> pyp_view_fits -h
usage: pyp_view_fits [-h] [--list] [--raw_lris] [--exten EXTEN] file

positional arguments:
  file           FITS file

optional arguments:
  -h, --help     show this help message and exit
  --list         List the extensions only? (default: False)
  --raw_lris
  --exten EXTEN  FITS extension (default: None)
```

8.1.3 Data Processing Scripts

pypeit_coadd_1dspec

See *Coadd 1D Spectra* for further details.

8.1.4 Calibration Scripts

pypeit_arcid_plot

Generate a PDF plot from a MasterFrame_WaveCalib.json file. This may be useful to ID lines in other data.:

```
unix> pypeit_arcid_plot -h
usage: pypeit_arcid_plot [-h] wave_soln title outfile

positional arguments:
  wave_soln   MasterWaveSoln file [JSON]
  title       Title for the plot
  outfile     Output PDF file

optional arguments:
  -h, --help  show this help message and exit
```

pypeit_lowrdx_pixflat

Convert a LowRedux pixel flat into a PypeIt ready file:

```
unix> pypeit_lowrdx_pixflat -h
usage: pypeit_lowrdx_pixflat [-h] lowrdx_file new_file

positional arguments:
  lowrdx_file  LowRedux Pixel Flat FITS file
  new_file     PypeIt FITS file

optional arguments:
  -h, --help   show this help message and exit
```

8.2 Code Flow

8.2.1 Overview

8.2.2 Setup

Flow

Below is the code flow for the *pypeit_setup* script. The following are nearly all function names or object methods. The module name is typically the first item, e.g. arparse.init is a method in arparse.py. Here goes:

```
├── pypeit_setup
|   ├── pyputils.make_pypeit_file(pypeit_file, spectrograph, dfnames)
|   ├── run_pypeit.parser(options)
|   ├── pypeit.PypeIt(args)
|   |   ├── load_input(pypeit_file)
|   |   |  ++ generates pyp_dict
|   |   ├── arparse.get_argflag_class()
|   |   |  ++ generates argf Class
|   |   ├── argf.init_param()
|   |   ├── plines = argf.load_lines(parlines)
```

(continues on next page)

(continued from previous page)

```
|   |   ├── argf.set_paramlist(plines)
|   |   ├── arparse.get_spect_class()
|   |   |  ++ generates spect Class
|   |   ├── spect.load_file(base=True)  # default
|   |   ├── spect.set_paramlist(lines)
|   |   ├── spect.load_file()  # instrument specific
|   |   ├── spect.set_paramlist(lines)
|   |   ├── spect.set_paramlist(plines)   # using pyp_dict
|   |   ├── spect.load_lines(spclines)      # command line
|   |   ├── argf.save()
|   |   ├── spect.save()
|   |   ├── arparse.init(argf, spect)  # Saved into settings.
|   |   ├── # fitsdict created
|   |   ├── fitsdict = arload.load_headers(datlines)
|   |   |   ├── # Checks on FITS files
|   |   |   ├── settings.spect['check'][ch]
|   |   ├── # Flip dispersion direction (if needed)
|   ├── ARMLSD  # This is formally below PypeIt, but I want to reduce the indentation␣
↪here
|   |   ├── armbase.SetupScience(fitsdict)
|   |   |   ├── filesort = arsort.sort_data(fitsdict)
|   |   |   |   ├── find_standard_file()
|   |   |   |  ++ Generates filesort dict
|   |   |   ├── arsort.match_science(fitsdict, filesort)
|   |   |   |  ++ Written to settings.spect[ftag]['index']
|   |   |   ├── arsciexp.ScienceExposure(i, fitsdict)
|   |   |   |  ++ Generates sciexp list of ScienceExposure objects
|   |   |  ++ setup_dict generated
|   |   |   ├── arsort.instr_setup(sciexp, kk+1, fitsdict, setup_dict)
|   |   |   |  ++ Generates setupIDs
|   |   |  ++ group_dict generated
|   |   |   ├── arsort.write_sorted(group_dict, setup_dict)
|   |   |   ├── arsort.write_setup(setup_dict)
```

Items

Items created and carried around:

```
filesort
fitsdict
settings.spect
settings.argf
setup_dict
group_dict
sciexp
```

8.3 New Spectrograph

Here are notes on how to build a new spectrograph from scratch or to add a new mode.

8.3.1 Entirely New

1. Build a new name_of_spectrograph.py file in pypeit.spectrograph
2. Fuss with the Detector object; one per detector - Set datasec, oscansec in the *raw* frame, i.e. as viewed on Ginga - Or generate a custom reader if these are variable
3. Set custom parameters

Near-IR

If this is a near-IR instrument, you may wish to turn off overscan subtraction. See Gemini_GNIRS for an example.

8.4 pypeit package

8.4.1 Subpackages

pypeit.core package

Subpackages

pypeit.core.wavecal package

Submodules

pypeit.core.wavecal.autoid module

pypeit.core.wavecal.defs module

pypeit.core.wavecal.fitting module

pypeit.core.wavecal.kdtree_generator module

pypeit.core.wavecal.patterns module

pypeit.core.wavecal.templates module

pypeit.core.wavecal.waveio module

pypeit.core.wavecal.wvutils module

Module contents

Submodules

pypeit.core.arc module

8.4.2 Submodules

8.4.3 Module contents

CHAPTER 9

Contents

9.1 pypeit package

9.1.1 Subpackages

pypeit.core package

Submodules

pypeit.core.combine module

pypeit.core.extract module

pypeit.core.flat module

pypeit.core.fsort module

pypeit.core.load module

pypeit.core.pca module

pypeit.core.plot module

pypeit.core.pydl module

pypeit.core.trace_slits module

9.1.2 Submodules

pypeit.waveimage module

9.1.3 Module contents

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Index