DBSP$_{DRP}$

Aug 18, 2021

Contents:

CHAPTER 1

Installing DBSP_DRP

Conda is the recommended pacakage manager used to install `DBSP_DRP`.

1.1 From Source

```
$ git clone https://github.com/finagle29/DBSP_DRP.git
$ cd DBSP_DRP
$ conda env create -f environment.yml
$ conda activate dbsp_drp
$ pip install -e .
```

This performs an editable install, which allows you to make modifications to the code and immediately see their effects. Importantly, this can be used in combination with `git` branches to test features in development.

1.2 Using pip

First download the provided environment.yml file

Now use the environment.yml file to create a conda environment with the required dependencies.

```
$ cd /path/to/Downloads
$ conda env create -f environment.yml
$ conda activate dbsp_drp
```

Now use `pip` to install DBSP_DRP

```
$ pip install git+https://github.com/finagle29/DBSP_DRP.git
```

CHAPTER 2

Post-Install

The telluric correction code provided by PypeIt relies on a large (5 GB) atmospheric model file (TellFit_Lick_3100_11100_R10000.fits), which can be downloaded here and must be installed into the `pypeit/data/telluric/atm_grids` directory of your PypeIt installation.

To determine the location of your PypeIt installation, open the Python interpreter and run

```
>>> import pypeit
>>> import os
>>> print(os.path.dirname(pypeit.__file__))
/Users/me/anaconda3/envs/dbsp_drp/lib/python3.7/site-packages/pypeit
```

An easier alternative is to download and run this script, which will perform the download and install it into the current PypeIt installation.

```
$ cd /path/to/Downloads
$ chmod +x download_tellfile
$ ./download_tellfile
```

If you have multiple PypeIt installations on the same machine, you can create a hard link from the one PypeIt installation to the others so you can reuse the atmospheric model file.

```
$ ln /path/to/stable/pypeit/data/telluric/atm_grids/TellFit_Lick_3100_11100_R10000.
↪fits /path/to/other/pypeit/data/telluric/atm_grids/TellFit_Lick_3100_11100_R10000.
↪fits
```

Make sure to use the same filename in both PypeIt installations. If you're not sure where your PypeIt installations are, run the previous Python snippet in each `conda` or `venv` environment you want to use `DBSP_DRP` in.

2.1 Testing your installation

Make sure your PypeIt installation was successful

```
$ run_pypeit -h
```

Run some built-in tests for DBSP_DRP, including verification that the quicklook script works

```
$ cd /path/to/DBSP_DRP
$ pytest .
```

CHAPTER 3

Using DBSP_DRP

After *Installing DBSP_DRP*, you are ready to reduce some DBSP data!

```
$ dbsp_reduce --help
usage: dbsp_reduce [-h] [-i] -r ROOT -d OUTPUT_PATH [-a {red,blue}] [-m]
               [--debug] [-j N] [-p PARAMETER_FILE] [-t] [-c]
               [--splicing-interpolate-gaps]

Automatic Data Reduction Pipeline for P200 DBSP

optional arguments:
    -h, --help            show this help message and exit
    -i, --no-interactive  Interactive file-checking?
    -r ROOT, --root ROOT  File path+root, e.g. /data/DBSP_20200127
    -d OUTPUT_PATH, --output_path OUTPUT_PATH
                            Path to top-level output directory.  Default is the␣
↪current working directory.
    -a {red,blue}, --arm {red,blue}
                            [red, blue] to only reduce one arm (null splicing)
    -m, --manual-extraction
                            manual extraction
    --debug               debug
    -j N, --jobs N        Number of processes to use
    -p PARAMETER_FILE, --parameter-file PARAMETER_FILE
                            Path to parameter file. The parameter file should be␣
↪formatted as follows:

                            [blue]
                            ** PypeIt parameters for the blue side goes here **
                            [red]
                            ** PypeIt parameters for the red side goes here **
                            EOF

                            The [red/blue] parameter blocks are optional, and their␣
↪order does not matter.
```

(continues on next page)

(continued from previous page)

```
    -t, --skip-telluric   Skip telluric correction
    -c, --null-coadd      Don't coadd consecutive exposures of the same target.
                            By default consective exposures will be coadded.
    --splicing-interpolate-gaps
                            Use this option to linearly interpolate across large gaps
                            in the spectrum during splicing. The default behavior is␣
↪to
                            only use data from one detector in these gaps, which␣
↪results
                            in a slightly noisier spliced spectrum.
```

The basic usage of DBSP_DRP is as follows:

```
$ dbsp_reduce -r /path/to/data/DBSP_YYYYMMDD/ -d /output/path/DBSP_YYYYMMDD_redux
```

When reducing a night of data on a machine with multiple CPU cores, it is highly recommended to add the `-j N` flag to use N jobs for the telluric correction.

Note: the default setting is to open a GUI that allows you to verify that the header data is correct for all of the files. Add the option `-i` or `--no-interactive` to turn this behavior off.

If you only want to reduce a subset of your data, you can select unwanted files in the header validation table GUI and right click to delete them from the current reduction run. Be sure to always keep the standard stars you need for fluxing in the data reduction.

If you want to only reduce the red arm or the blue arm, add the flag `--arm red` or `-a blue` for short.

If you want lots of intermediate debugging plots displayed to the screen, add the flag `--debug`.

If you want to check that your target traces have been correctly identified, and manually select them if they were missed, add the flag `--manual-extraction` or `-m` for short. Check out *Using the Manual Tracing and Manual Aperture GUIs* for more details.

If you want to fine-tune the reduction parameters, create a parameter file like so:

```
# params.cfg
[blue]
# PypeIt reduction parameters to control the blue side reduction
[reduce]
    [[skysub]]
        no_local_sky = True
    [[extraction]]
        use_2dmodel_mask = False
[red]
# PypeIt parameters to control the red side reduction
[reduce]
    [[skysub]]
        no_local_sky = True
    [[extraction]]
        use_2dmodel_mask = False
```

and use the option `-p params.cfg` or its longer form `--parameter-file params.cfg`.

See PypeIt Parameters for more details on the full set of available parameters.

CHAPTER 4

Using the Manual Tracing and Manual Aperture GUIs

4.1 Manual Tracing GUI

After the first round of reducing the red and/or blue sides, `dbsp_reduce` will open a matplotlib window to display the sky-subtracted spectra, along with any object traces that were automatically detected.

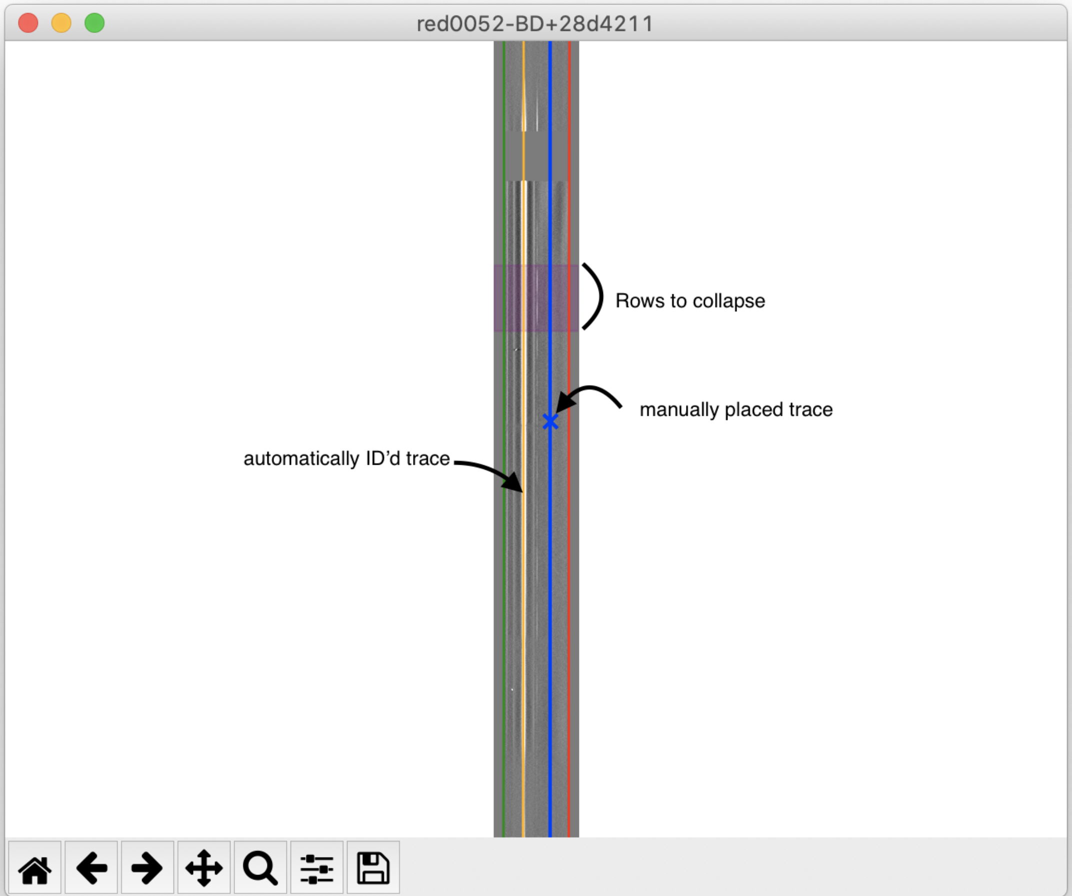
Using the left and right arrow keys, you can cycle through the spectra.

If your science target was not automatically detected, you can zoom in on the trace using the standard matplotlib zoom tool and then with your mouse over the trace, press the `m` key to mark that trace.

If you make a mistake, you can press `d` with your mouse over a previously-marked trace to delete the trace.

To adjust the region of the spectrum that will be collapsed to select apertures and background regions, press `c`, then left click and drag to highlight the region to be collapsed in purple.

After you close this window, for each frame you marked with a manual trace, a window will pop up with a Manual Aperture and Sky Selection GUI.



4.2 Manual Aperture and Sky Selection GUI

This GUI shows the collapsed flux, along with any automatically identified traces (in orange) and your manually placed traces (in blue) along with the FWHM of each shaded in a lighter color.

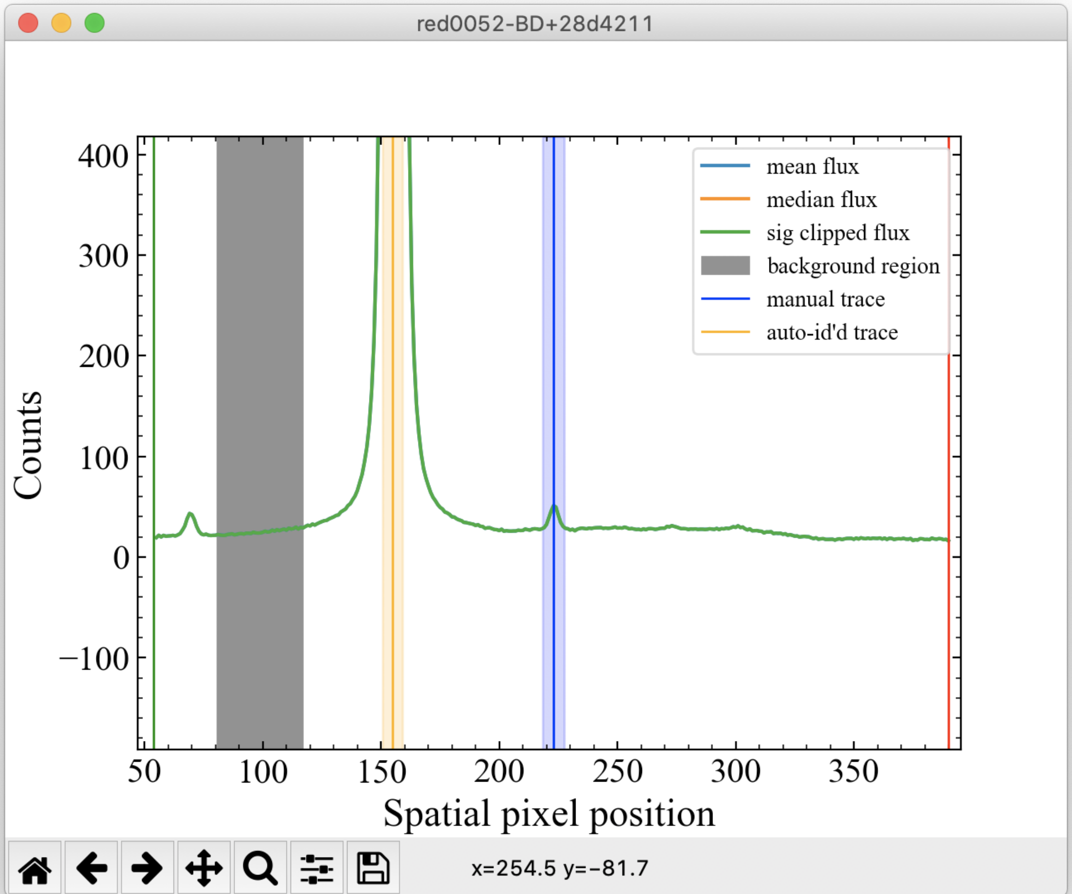
In this GUI, you can left click and drag your manual traces (in blue) to adjust their position.

Additionally, you can right click and drag the shaded FWHM regions to adjust their extent.

To mark background regions, press `b` and then left click and drag to mark background regions by shading them in gray.

You can delete a background region by holding your mouse over the shaded background regions and press `d` to delete.

Once you are finished adjusting manual traces/FWHMs and marking background regions, close the window to be shown the same GUI for the next object you marked a manual trace on.



4.2.1 Tips

Make sure to select background regions on either side of your target for the best sky subtraction.

If you are dealing with faint sources, it is a good idea to re-mark in blue any orange (automatically identified) traces in case parameter changes lose these objects.

CHAPTER 5

DBSP_DRP QA Outputs

The QA folder has two main QA pages: `MF_A.html` and `Extraction.html`.

`MF_A.html` is automatically generated by PypeIt and includes a number of QA plots regarding the calibration data, which currently only encompass the wavelength calibration. You can read in more detail about PypeIt's QA plots here

In `Extraction.html`, for each target, there is a page that shows various steps of the reduction for the red and blue sides. From left to right, the images are flat-fielded frame, sky model, sky-subtracted frame, sky-subtracted residuals, and sky- and object-subtracted residuals. Above each set of images, the raw filename, time of observation and the airmass is displayed. Slit edges are marked in red and green, object traces are marked in orange, and extraction FWHMs are highlighted in orange. If the extraction FWHM is very different from what you would expect, it is probably a good idea to use manual tracing on that target.

CHAPTER 6

DBSP Quicklook

During an observing run, in order to make time-sensitive decisions about what to observe, a quicklook script is provided.

6.1 Usage

```
$ dbsp_ql --help
usage: dbsp_ql [-h] [--no-show] fname

Quicklook for P200 DBSP

positional arguments:
  fname       file to take a quick look at, or else red/blue
              to just perform rough calibrations

optional arguments:
  -h, --help  show this help message and exit
  --no-show   Set this flag to suppress opening of plots
```

First, navigate to a directory you want the output data in.

```
$ cd /path/to/workdir
```

Then, with at least one arc frame and at least one flat frame in `/path/to/calibs` run

```
$ dbsp_ql /path/to/calibs/red
$ dbsp_ql /path/to/calibs/blue
```

to perform quick calibrations for the red and blue sides, respectively.

Then once you have a science frame (either in the same directory, or elsewhere) in `/path/to/science/data`, run

```
$ dbsp_ql /path/to/science/data/red0030.fits
$ dbsp_ql /path/to/science/data/blue0030.fits
```

This step should be very quick (about 15 seconds for the red side, 8 seconds for the blue side), and each command will pop up a ginga window for you to inspect the sky-subtracted frame. Also, a GUI will pop up to display the fluxed spectrum for each object identified in the frame.

The quicklook script produces PypeIt 2D Spectrum and 1D Spectrum files, described in *Data Reduction Outputs*.

CHAPTER 7

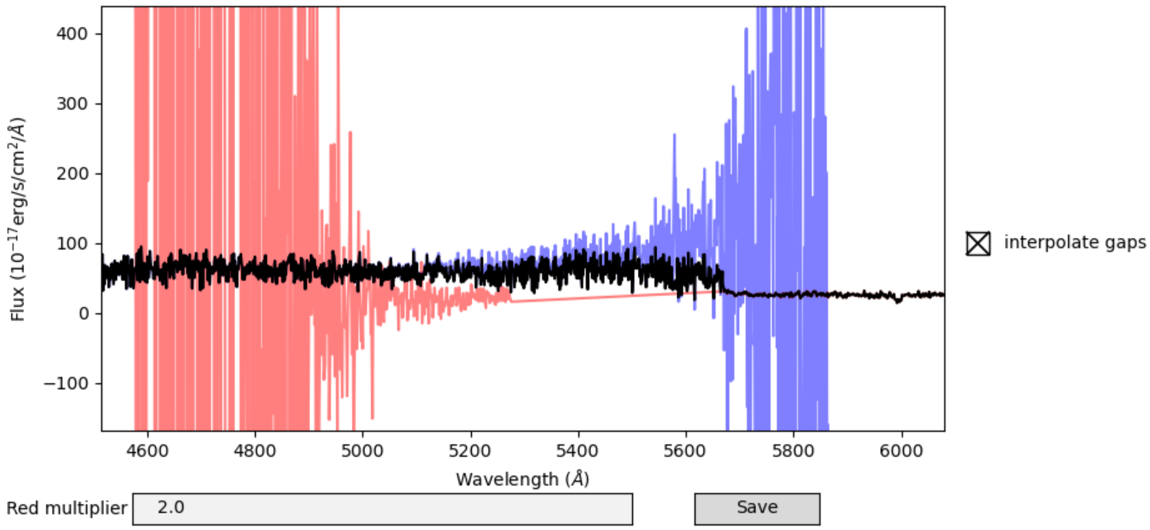
Adjusting splicing between arms

Typically the splicing between arms is quite good, and you only need to adjust it when the optimal extraction FWHM vary greatly between the red and blue sides. This can happen when an extended object is near another object on the slit. You can check the extraction FWHM visually by looking at the Extraction QA page.

```
$ dbsp_adjust_splicing --help
usage: dbsp_adjust_splicing [-h] fname [fname ...]

Red/Blue splicing adjustment for P200 DBSP

positional arguments:
  fname       Final data output from DBSP_DRP to adjust the splicing of. Can be one␣
↪or multiple files.

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

In this GUI, the blue and red spectra are plotted as well as the spliced spectrum. After using the usual Matplotlib tools to zoom and pan to the region of interest, you can adjust the red multiplier to multiply the red flux by a constant, and change if flux will be interpolated across detector gaps, as well as save your changes to the same file you opened. After you close the GUI, you will be presented with the next spectrum to adjust the splicing of.

The default splicing that DBSP_DRP does uses a red multiplier of 1, since the flux calibration provided by PypeIt is excellent.


400
300
200
100
0
−100
Flux (10^{-17}erg/s/cm^2/Å)
4600
4800
5000
5200
5400
5600
5800
6000
Wavelength (Å)
interpolate gaps
Red multiplier
2.0
Save

CHAPTER 8

Data Reduction Outputs

Assuming you ran DBSP_DRP with `dbsp_reduce -r $RAW_PATH -d $OUTPUT_PATH`, then in the `$OUTPUT_PATH/Science` directory you will find

PypeIt 2D Spectrum files `spec2d_redNNNN-target_DBSPr_obstimestamp.fits` described in more detail here. Briefly, these files hold flat-fielded 2d spectral images, as well as sky, noise, object, wavelength, and bad pixel mask images. These files can be visually inspected using the command `pypeit_show_2dspec SPEC2D_FILE`.

PypeIt 1D Spectrum files `spec1d_redNNNN-target_DBSPr_obstimstamp.fits` described in more detail here. Briefly, these files hold 1d spectra for each object that was traced and extracted from the raw frame. Each object's spectrum is stored in a separate extension of the FITS file, and the extension names are of the form `SPATNNNN-SLITMMMM-DET01` where the SPAT number describes the spatial pixel coordinate of the object and the SLIT number is only useful for spectrographs with multiple slits (and the DET number is only useful for spectrographs with multiple detectors for the same arm). These files can be visually inspected using the command `pypeit_show_1dspec --exten N SPEC1D_FILE` to view extension N of the file.

PypeIt 1D Coadd files `redNNNN-redNNNN_target_SPATNNNN.fits` described in more detail here. These files exist to separate out multiple objects on the same frame into their own file, and to coadd consecutive exposures of the same frame. These files can be visually inspected using the command `lt_xspec COADD_FILE`. The filename contains the first and last raw data file and the medial spatial pixel coordinate of the object. This was done to make the coadd filename of constant length, instead of scaling with the number of coadded frames, since all operating systems/file systems have maximum allowed filename lengths.

Telluric-corrected 1D Coadd files have `_tellcorr` appended to the base filename of the coadd file.

In the directory `$OUTPUT_PATH/spliced` are the spliced final data products. The final data product of DBSP_DRP is a FITS file named `target_char.fits` with structure described below, where `char` is a one letter designation of which object along the slit it is.

Table 1: table

Extension Number	Name	Header	Data
0	PRIMARY	Version info about DBSP_DRP, PypeIt, numpy and astropy	None
1	RED0031	Header from raw red side fits file	Fluxed (not telluric-corrected) red side spectrum
2	RED0032	Header from raw red side fits file	Fluxed (not telluric-corrected) red side spectrum
3	BLUE0032	Header from raw blue side fits file	Fluxed blue side spectrum
4	BLUE0032	Header from raw blue side fits file	Fluxed blue side spectrum
5	RED	Header from raw red side fits file	Fluxed (and telluric-corrected) red side spectrum
6	BLUE	Header from raw blue side fits file	Fluxed blue side spectrum
7	SPLICED	Empty	Final spliced spectrum

The header on the 0th extension also contains cards named `B_COADD` and `R_COADD` which contain the filename of the blue and red coadd files, respectively, that were spliced together. This is useful for determining which traces a particular final output file corresponds to. The 0th header also contains an `INTERP_GAPS` cards, noting whether or not detector gaps were interpolated over during splicing. If the splicing has been manually adjusted (see *Adjusting splicing between arms* for more details) then a `RED_MULT` card will also be present, recording the factor multiplied into the red coadd spectrum before splicing with the blue coadd.

If *n* red side files were coadded and *m* blue side files were coadded, then extensions 1 through *n* would contain the red side raw headers and fluxed spectrum from each individual file, extensions 1+*n* through 1+*n*+*m* would contain the blue side raw headers and fluxed spectra from each individual file, and the last 3 extensions are the red coadd, blue coadd, and final spliced spectrum.

If an object was not observed in the blue (red) then there will be no raw blue (red) frames, and the BLUE (RED) extension would still exist, but contain no data.

Each of the data tables contain columns for *wave*, *flux*, and *sigma*, with wavelength in Angstroms and both flux and sigma in units of 10^{-17}erg/s/cm^2/Ang.

CHAPTER 9

dbsp_drp

9.1 dbsp_drp package

9.1.1 Subpackages

dbsp_drp.tests package

Submodules

dbsp_drp.tests.test_splicing module

dbsp_drp.tests.test_table_edit module

Module contents

9.1.2 Submodules

9.1.3 dbsp_drp.coadding module

9.1.4 dbsp_drp.fix_headers module

9.1.5 dbsp_drp.fluxing module

9.1.6 dbsp_drp.gui_helpers module

9.1.7 dbsp_drp.manual_aperture module

9.1.8 dbsp_drp.manual_tracing module

9.1.9 dbsp_drp.p200_redux module

9.1.10 dbsp_drp.qa module

9.1.11 dbsp_drp.quicklook module

9.1.12 dbsp_drp.reduction module

9.1.13 dbsp_drp.splicing module

9.1.14 dbsp_drp.table_edit module

9.1.15 dbsp_drp.telluric module

9.1.16 Module contents

CHAPTER 10

Indices and tables

- genindex
- modindex
- search