MAUVE Data Reduction Cookbook

Version	Date	Authors
0.1	28 Apr 2023	Amelia Fraser-McKelvie, Adam Watts
1.0	12 Jul 2023	Barbara Catinella, Luca Cortese
1.1	31 Jul 2023	Luca Cortese – Updated alignment and trouble-shooting sections
1.2	26 April 2024	Luca Cortese – Included check on PMP, conda environment version and satellite trail fix instructions. Updated naming convention for script, and trouble-shooting section.

Introduction

This cookbook is designed to guide the user through the reduction of MAUVE observations, from raw data to fully calibrated, aligned, and mosaicked cubes. MAUVE data are taken with the MUSE wide-field integral field spectrograph at the ESO VLT and include between 1 and 9 individual pointings per galaxy. Our observing mode is to take 4 object and 1 sky exposures per observing block (OB). The MUSE raw data files are composed of 24 FITS extensions, one for each IFU (Fig. A1).

MUSE data reduction is done on the magpipe machine at Data Central and is based on EsoRex (ESO Recipe Execution Tool). We use `pymusepipe`, which is a wrapper for EsoRex developed by Eric Emsellem for PHANGS-MUSE data reduction, and adapted by Amelia Fraser-McKelvie, Adam Watts and Eric Emsellem for the reduction of MAUVE and GECKOS data.

Useful references:

- EsoRex manual
- Emsellem et al. 2022: The PHANGS-MUSE survey
- PHANGS release README file, describing parts of the data reduction flow.

Log into Data Central and magpipe

- Log into Data Central using your credentials: https://desktops.datacentral.org.au/guacamole/
- Click magpipe, this opens the “Login to dccompute2” widget.
- Login using your DC username and our magpipe password: M@uve1sJu$tPurple – leave Session as is (Xorg).

Set up your working environment

Open a terminal. You might get an error (Failed to open directory “<your username>”) when opening the file manager or a terminal – ignore it.

Go to your home directory and make sure that the bash has the needed information (paths, etc.). Note that you can use the tab to complete the commands.

- `emacs .bashrc &`
- on the terminal: `diff .bashrc /data/home/bcatinella/.bashrc`
- attach the block of lines that you see on screen at the bottom of your .bashrc file (see Fig. 1).
- Test that everything works: opening a new terminal shows (PLPenv2) at the prompt.

It is also convenient to define a few useful aliases. You can create a file .bash_aliases that is automatically read by .bashrc. On the terminal in your home directory:

- `cp /data/home/bcatinella/.bash_aliases .`
- Open it with emacs (or another editor) and change as appropriate.

```
## Added on May 4 2023 -- from /data/home/amelia_fm/.bashrc
# This is the stuff to point the terminal to where python and pymusepipe are installed
PATH="/cloud/teamdata/mauve/soft/esorex/bin:$PATH"
PATH="/cloud/teamdata/mauve/soft/local/bin:$PATH"
PATH="/cloud/teamdata/mauve/soft/spacepylot-master:$PATH" #For spacepylot
# >>> conda initialise >>>
#!! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/cloud/teamdata/mauve/soft/miniconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/cloud/teamdata/mauve/soft/miniconda3/etc/profile.d/conda.sh" ]; then
        . "/cloud/teamdata/mauve/soft/miniconda3/etc/profile.d/conda.sh"
    else
        export PATH="/cloud/teamdata/mauve/soft/miniconda3/bin:$PATH"
    fi
fi
unset __conda_setup
#<<< conda initialise <<<
conda activate PLPenv2
```

Fig. 1: bash set up for MAUVE data reduction.

Check the version of `pymusepipe`

Different version of `pymusepipe` are installed on magpipe in different `conda` environments to avoid issues with versioning etc. As such, before starting the data reduction, you need to make sure that you are loading the correct conda environment. For MAUVE, the current options are:

- PLPenv2 – `pymusepipe v.2.24.7`
- PLPenv4 – `pymusepipe v.2.28.2`

If in doubt, always check the document at this link, where every updates of `pymusepipe` and `conda` environments are documented.

At the time of the last update of these instructions (26/04/2024), the default version for MAUVE is 2.28. If you are not in the right conda environment, **we strongly recommend that you update your `.bashrc`**

Just replace the last line in your `.bashrc` file (see Fig. 1 – i.e., conda activate PLPenv2) with

```
conda activate PLPenv4
```

If, for some reasons, you do not want to change your `.bashrc` file, you will need to type

```
conda activate PLPenv4
```

every time you open a new terminal to launch `pymusepipe`.

File system overview

Main directories for MAUVE data reduction:

- `/cloud/teamdata/mauve/` – main project directory
- `/cloud/teamdata/mauve/data/muse/` – our MUSE OBs
- `/cloud/teamdata/mauve/data/img/` – optical imaging needed for flux calibration (`SDSS/`) and alignment (`LEGACY/`)
- `/cloud/teamdata/mauve/reduction_directory_template/` – template directory for data reduction. You will create your own directory below.
- `/cloud/teamdata/mauve/red_v1/` – current version of reduction scripts.

Download MAUVE OBs to magpipe

Detailed instructions on this step can be found here.
Update the spreadsheet here.

Set up your data reduction directory

First, set up your own data reduction directory (using the available template) and rename files and directories to correspond to the galaxy that you are going to reduce. In this example, the user is "barbara" and the galaxy to process is NGC 4064, which has 2 OBs. We will create a "red_test_barbara" directory.

- `cd /cloud/teamdata/mauve/`
- `cp -r reduction_directory_template red_test_barbara`

For brevity, we will refer to `/cloud/teamdata/mauve/red_test_barbara` as `[dr_home]`.
This directory contains 3 subdirectories: `Config/`, `Data/` and `Scripts/`.

- Update "root" within the `Config/rc_mauve.dic` file with the correct data reduction directory:

```
(PLPenv2) bcatinella@dccompute2:/cloud/teamdata/mauve/red_test_barbara$ more Config/rc_mauve.dic
musecalib /cloud/teamdata/mauve/soft/esorex/calib/muse-2.8.7/
musecalib_time /soft/ESO/MUSE/astrocal/
root /cloud/teamdata/mauve/red_test_barbara/Data/MAUVE/
str_dataset OB
str_pointing P
ndigits 3
```

Fig. 2: `rc_mauve.dic` configuration file.

Next, we need to fix the data structure under `[dr_home]/Data/MAUVE/Muse/` by renaming the `galaxyID` directory, creating additional OB00X subfolders if there is more than 1 OB (we have 2 in this example), and creating symbolic links to the directories that contain the MUSE data (check the `/cloud/teamdata/mauve/data/muse/` directory for the OB names):

- `cd /cloud/teamdata/mauve/red_test_barbara/Data/MAUVE/Muse`
- `cp -r galaxyID NGC4064`
- `cd NGC4064`
- `rmdir OB001/Raw` [this is currently an empty directory, but needs to be a sim-link]
- `cp -r OB001 OB002`
- `ln -s /cloud/teamdata/mauve/data/muse/NGC4064-1 OB001/Raw`
- `ln -s /cloud/teamdata/mauve/data/muse/NGC4064-2 OB002/Raw`

Lastly, copy the SDSS and LEGACY r-band images of your galaxy to your `/Data/MAUVE/Imaging/` directory:

- `cp /cloud/teamdata/mauve/data/img/SDSS/NGC4064_SDSS_r.fits /cloud/teamdata/mauve/red_test_barbara/Data/MAUVE/Imaging/`
- `cp /cloud/teamdata/mauve/data/img/LEGACY/NGC4064_LEGACY_R.fits /cloud/teamdata/mauve/red_test_barbara/Data/MAUVE/Imaging/`

Set up the data reduction script

Navigate to the `[dr_home]/Scripts/python` directory and rename the template data reduction script to reflect the galaxy being reduced:

- `cd /cloud/teamdata/mauve/red_test_barbara/Scripts/python`
- `cp redpmp2_28_v1_template.py redpmp2_28_v1_NGC4064.py`

Note that the naming of the template includes the version of `pymusepipe` for which the template has been designed for. The template director will contain only the last version (i.e., the one you should use). In case you need to go back to previous versions of the script, you want to check differences, or need to run an old version of the data reduction, you should check the files in this folder:

`/cloud/teamdata/mauve/scripts/pmp/`

Here you will find all the versions of the DR scripts used by MAUVE and a README file highlighting the major differences.

Now open the python script with emacs or another editor and update the following three fields: `galaxyID`, `galaxy_dict`, and `run` (these are found near the top, after the library import lines and the filter definitions):

- Change the `galaxyID` field to the ID of the galaxy to be reduced, in this case NGC4064 (no spaces). Note that this is also the ID used for the SDSS/Legacy images, so, they need to be consistent (check the image names).
- Update `galaxy_dict` to reflect the number of OBs you want to reduce. Note the format of this line: `galaxy_dict = {galaxyID:['Muse', {1:1, 2:1}]}`. You only change the last part, which refers to the OBs. Use: `{1:1}` for 1 OB, `{1:1, 2:1}` for 2 OBs, `{1:1, 2:1,`

`3:1}` for 3 OBs and so on. It is unclear what the second index is for. Note that the OB order is set during the download procedure.

- Update `run` to indicate the data reduction directory that you are using.

```
## These are the things you can change
## But all you should really have to is the galaxy ID and dict
galaxyID = "NGC4064"
## "Muse dictionary gives OB numbers: dataset num in the galaxy folder. set it to OBnum:1 for all your OBs in the folder"
galaxy_dict = {galaxyID:['Muse', {1:1, 2:1}]}
# Run
run = "red_test_barbara"
```

Fig. 3: data reduction script setup.

Start reducing the data!

Data reduction flow overview

We are now ready to process the data! Before running the script, it is useful to have an idea of what the data reduction steps involve (see also Appendix B):

- **Step 1**: basic calibrations (bias, dark, flat, twilight, wavelength calibration, LSF, geometry, astrometry, standard star), applied to offset sky and object. This step runs EsoRex to the end of `muse_scibasic` (see Fig. B1) and creates pre-alignment images (see Fig. B3).
- **Step 2 (align)**: alignment with respect to reference image, based on LEGACY r-band imaging. This step is interactive in the terminal.
- **Manual alignment**. Check alignment solutions and improve iteratively. When happy, save offsets to table.
- **Check_alignment.** Option for (subsequent) independent check of alignment solution.
- **Step 2 (flux)**: flux and background rescaling based on SDSS r-band imaging.
- **Step 3**: sky subtraction and resampling to common reference WCS grid; creation of individual cubes (one per OB).
- **Step 4**: creation of final mosaic.

The data reduction steps illustrated above are all performed by running the `reduce_v1_NGC4064.py` script multiple times, switching on the right “step” every time. The relevant lines are found at the bottom of the script (see Fig. 4).

```
if __name__ == "__main__":
    # step1()
    # step2_align()
    # check_align()
    # step2_flux()
    # step3()
    # step4()
```

Fig. 4: data reduction script setup.

DR step 1: basic calibrations

This first step performs basic calibrations (bias, dark, flat, twilight, wavelength calibration, LSF, geometry, astrometry, standard star) and creates pre-alignment images. It requires no user intervention and takes at least 2h per OB:

- `cd /cloud/teamdata/mauve/red_test_barbara/Scripts/python`
- Open `reduce_v1_NGC4064.py` and uncomment "`step1()`". Save the file.
- In a terminal on that same directory, start python and run the script:
  - `ipython`
  - `%run -i reduce_v1_NGC4064.py`

This creates a lot of files and tables under `[dr_home]/Data/MAUVE/Muse/NGC4064/` and its various subdirectories. Note in particular:

- pre-alignments images (4 per OB, since each OB includes 4 object exposures; see Fig. 5) under `Alignment/`
- calibration images under `OB00x/Master/`
- calibration and pre-alignment tables under `OB00x/Astro_tables/`.

Before proceeding to the next step, make sure to do a few quick checks following the instructions on our QC cookbook.

```
(PLPenv2) bcatinella@dccompute2:/cloud/teamdata/mauve/red_test_barbara/Data/MAUVE/Muse/NGC4064$ ls *
Alignment:
IMAGE_FOV_prealign_OB001_SDSS_r_2023-03-26T02:52:02_0001.fits
IMAGE_FOV_prealign_OB001_SDSS_r_2023-03-26T02:52:02_0002.fits
IMAGE_FOV_prealign_OB001_SDSS_r_2023-03-26T02:52:02_0003.fits
IMAGE_FOV_prealign_OB001_SDSS_r_2023-03-26T02:52:02_0004.fits
IMAGE_FOV_prealign_OB002_SDSS_r_2023-03-27T03:22:49_0001.fits
IMAGE_FOV_prealign_OB002_SDSS_r_2023-03-27T03:22:49_0002.fits
IMAGE_FOV_prealign_OB002_SDSS_r_2023-03-27T03:22:49_0003.fits
IMAGE_FOV_prealign_OB002_SDSS_r_2023-03-27T03:22:49_0004.fits

Combined:
Cubes  Esorex_log  Log  Pipe_products  Sof

OB001:
Astro_tables  Cube     Esorex_log  Log   Master  Pipe_products  Reduced  Sof
Config        Cubewcs  Figures     Maps  Object  Raw            Sky      Std

OB002:
Astro_tables  Cube     Esorex_log  Log   Master  Pipe_products  Reduced  Sof
Config        Cubewcs  Figures     Maps  Object  Raw            Sky      Std
(PLPenv2) bcatinella@dccompute2:/cloud/teamdata/mauve/red_test_barbara/Data/MAUVE/Muse/NGC4064$
```

Fig. 5: overview of data reduction directory structure.

DR step 2 (align): automatic (pre)-alignment

***Important**. If the galaxy you are reducing has been already reduced before, and you are happy with the alignment solution, you do not need to go through the manual alignment. Jump to the next step to see how to incorporate the alignment solution obtained in a previous reduction.*

This step performs an automatic alignment with respect to the LEGACY r-band reference image. The goal is to match the astrometry of each individual exposure with the r-band image, to obtain an absolute astrometric solution that optimizes the mosaicking. This is done using the `myalign` module in `pymusepipe`. This step is interactive in the terminal, and takes only a couple of minutes per OB to run:

- Open `reduce_v1_NGC4064.py`, comment "`step1()`" and uncomment "`step2_align()`". Save.
- `%run -i reduce_v1_NGC4064.py`

At this point, the optical flow has run and created an alignment solution and the `pymusepipe` alignment module has been returned to the ipython terminal. PNG files of the aligned frames have been created under `Alignment/AlignFigures/`. We will use these next.

A few things to note:

- You will see a few `MusePipeWarning` messages on the terminal (see red text in Fig. 6), which you can safely ignore.
- You might get a system error ("*ICE default IO error handler doing an exit()*"). If this happens, you need to delete a couple of files in your home directory (see Troubleshooting section below), restart python and rerun step 2.

```
In [1]: %run -i reduce_v1_NGC4064.py
# MusePipeInfo === Initialising MusePipe for Target NGC4064 ===
# MusePipeInfo Initialise Pipe for Target = NGC4064    / Dataset 001
# MusePipeInfo Time_astrometry option will now be set to False
# MusePipeInfo Input MUSE mode = WFM-NOAO-N
# MusePipeInfo Found one ASTROMETRY file M.MUSE.2021-04-26T11:17:55.186.fits
# MusePipeInfo Found one GEOMETRY file M.MUSE.2021-04-26T11:05:30.766.fits
# MusePipeWarning Checkmode is True: the MUSE Mode will be checked.
# MusePipeWarning All Raw files which do NOT have musemode = WFM-NOAO-N and are mode specific (e.g., Flat field) will be masked.
# MusePipeWarning If you wish otherwise, set checkmode to False [but this may impact the data reduction].
# MusePipeInfo Initialise Pipe for Target = NGC4064    / Dataset 002
# MusePipeInfo Time_astrometry option will now be set to False
# MusePipeInfo Input MUSE mode = WFM-NOAO-N
# MusePipeInfo Found one ASTROMETRY file M.MUSE.2021-04-26T11:17:55.186.fits
# MusePipeInfo Found one GEOMETRY file M.MUSE.2021-04-26T11:05:30.766.fits
# MusePipeWarning Checkmode is True: the MUSE Mode will be checked.
```

Fig. 6: step 2: pre-alignment.

DR step 2B: manual alignment

The alignment solution created above does not always deliver the sub-spaxel accuracy needed for our mosaics, due to the lack of bright point sources (Emsellem+ 2022), so the next task is to fine-tune the alignment by hand (note that 1 MUSE pixel = 0.2 arcsec). This is the most time-consuming part of the data reduction. Give yourself plenty of time for this step as it might take quite a bit of tuning before you are happy with the end result. Since this procedure is somewhat subjective, we recommend that only a couple of people do this for the full data set, if possible.

First, inspect the result of the alignment for each frame:

- Open a File Manager window and navigate to `[dr_home]/Data/MAUVE/Muse/NGC4054/Alignment/AlignFigures`
- Under View → Arrange Items → Sort by Name
- Open all figures named `opflow_beforeafter_*.png` (not `opflow_beforeafter_frac*.png`).

Fig. 7 shows an example (frame 000). The left image shows an overlay of the reference image (SDSS), with the MUSE cube subtracted *before* alignment. The right image shows the same, but *after* the optical flow. Check the right image closely, to see if it has aligned correctly. This is an interesting example -- the 5 stars in the bottom half of the frame seem to first order reasonably well aligned (they are well subtracted), but the brightest one on the top left is not. It is already clear that we cannot align the bright one without misaligning the others; it turns out that the brightest star has proper motion and should not be used as a reference.

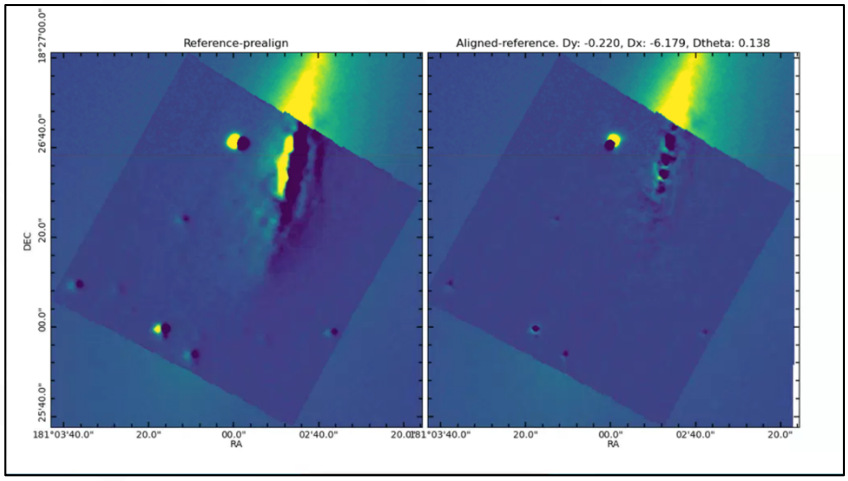


Fig. 7: inspection of `opflow_beforeafter_000.png`.

Even if the alignment looks OK, we need to check it more carefully and improve it to sub-spaxel accuracy. Start by running the `myalign` module with no offsets. This needs to be done for each frame, and the solution recorded on our DR spreadsheet. Let's start with the first frame (0):

- At the prompt in the ipython terminal type:
  `myalign.offset_and_compare_ima(0,extra_pixel=[0,0],threshold=3)`

Note that the shift given by `extra_pixel=[x,y]` is relative to the *original solution*, not the last time it was run for this frame (i.e., the changes are not additive between multiple runs). The `threshold` parameter seems to regulate the level (and thus the number) of contours. Increase it by a lot (e.g., move it to 20 or 50) to get more contours. It is unclear how to change the rotation.

The output on the terminal is shown in Fig. 8. This module also creates and pops up 4 plots (see Fig. 9) – the most useful ones are the contour plot (bottom right) and the scatter plot (top right). The better the alignment, the smaller the scatter.

```
In [2]: myalign.offset_and_compare_ima(000,extra_pixel=[0,0],threshold=3)
# MusePipeInfo [#001/i=000] --- Regridding Image  IMAGE_FOV_prealign_OB001_SDSS_r_2023-03-26T02:52:02_0001.fits ---
# MusePipeInfo        Offset   [PIXELS]:   6.1793   0.2202  /  [ARCSEC]:  -1.2359  0.0440
# MusePipeInfo        Rotation [DEGREE]:  -0.1379
# MusePipeInfo Renormalising the data as: Normalised = 9.3987e-01 * (-1.1050e+00 + MUSE)

In [3]:
```

Fig. 8: running `myalign.offset_and_compare_ima()`.

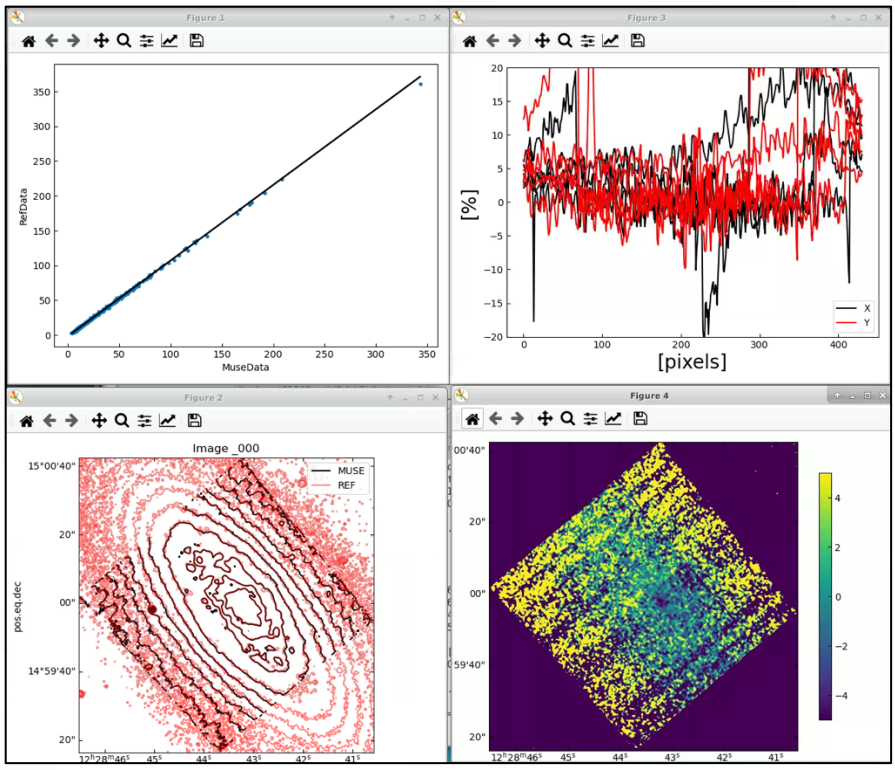


Fig. 9: plots created by `myalign.offset_and_compare_ima()`.

Now explore the contour plot in detail – zoom in a lot to see how well the red contours overlay with the black (Fig. 10, top). Use point sources (unless they have proper motion) if present and try to optimize near the region where the frames will join in the mosaic (if possible). If there are no point sources, try to match the contours (this is trickier, of course). Estimate x, y offsets in arcsec from the image, and use

the pixel scale information on the terminal (see Fig. 8) to compute the approximate offsets in pixels for `myalign` (note that the offsets are applied to the red contours, not the black).

It is possible that the best solution for the contours, may not always be good for a star, in particular if the star is far from the centre of the frame. If so, consider adding a rotation component. Indeed, unless the star has significant proper motion, it should be possible to find a good solution for both star and contours.

In this case, applying a small offset in both directions improves the alignment slightly ((Fig. 10, bottom):

- At the prompt in the ipython terminal type:
  `myalign.offset_and_compare_ima(0,extra_pixel=[0.5,1],extra_rotation=0, threshold=3)`

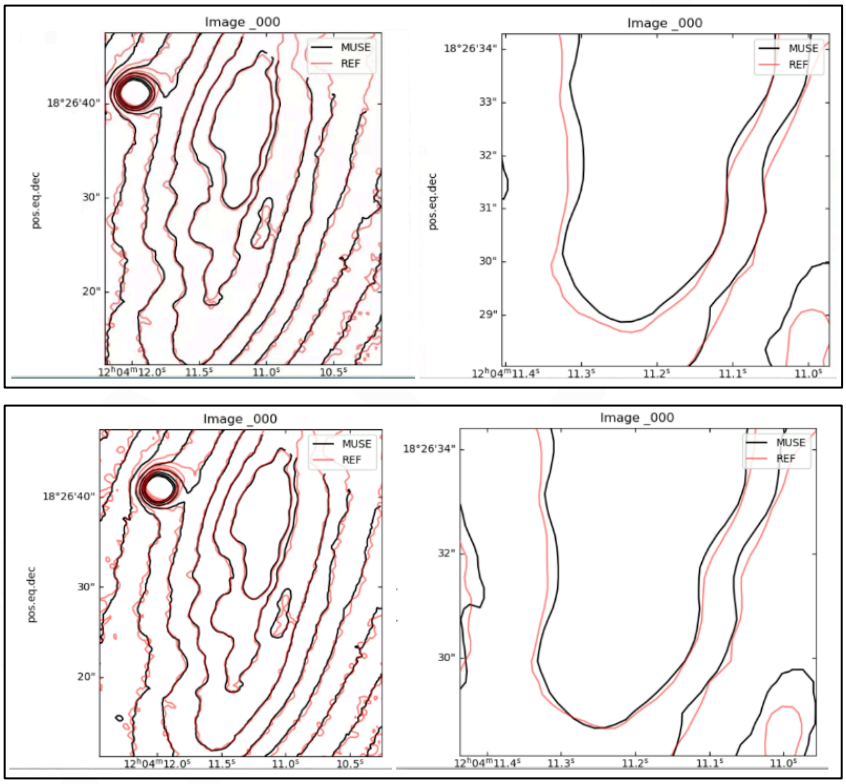


Fig. 10: manual alignment. Original (top) vs improved (bottom). Note that the bright point source in the left panels should not be used for alignment (proper motion).

```
In [4]: myalign.offset_and_compare_ima(000,extra_pixel=[0.5,1],threshold=3)
# MusePipeInfo [#001/i=000] --- Regridding Image  IMAGE_FOV_prealign_OB001_SDSS_r_2023-03-26T02:52:02_0001.fits ---
# MusePipeInfo        Offset   [PIXELS]:   6.6793   1.2202  /  [ARCSEC]:  -1.3359  0.2440
# MusePipeInfo        Rotation [DEGREE]:  -0.1379
# MusePipeInfo Renormalising the data as: Normalised = 9.3444e-01 * (-1.0679e+00 + MUSE)
```

Fig. 11: applying small offsets with `myalign.offset_and_compare_ima()`.

- When you are happy with the solution, record the offsets and rotation on our DR spreadsheet.
- Move to the next frame and do the same.
- Finally, save the table by running the following command (run it twice since the first time the table does not exist): `myalign.save_fits_offset_table(overwrite="True")`

  Note that, even if you save this table after optimizing the first frame, `myalign.offset_and_compare_ima()` does not read it. In other words, if you check the new offsets for frame 0, these are not updated (see Fig. 12). The only way to re-define the 0,0 position is to run `myalign.transfer_extra_to_guess()`. But this should almost never be needed.

```
In [10]: myalign.offset_and_compare_ima(000,extra_pixel=[0.5,1],threshold=3)
# MusePipeInfo [#001/i=000] --- Regridding Image  IMAGE_FOV_prealign_OB001_SDSS_r_2023-03-26T02:52:02_0001.fits ---
# MusePipeInfo        Offset   [PIXELS]:   6.6793   1.2202  /  [ARCSEC]:  -1.3359  0.2440
# MusePipeInfo        Rotation [DEGREE]:  -0.1379
# MusePipeInfo Renormalising the data as: Normalised = 9.3444e-01 * (-1.0679e+00 + MUSE)

In [11]: myalign.save_fits_offset_table(overwrite="True")

In [12]: myalign.offset_and_compare_ima(000,extra_pixel=[0,0],threshold=3)
# MusePipeInfo [#001/i=000] --- Regridding Image  IMAGE_FOV_prealign_OB001_SDSS_r_2023-03-26T02:52:02_0001.fits ---
# MusePipeInfo        Offset   [PIXELS]:   6.1793   0.2202  /  [ARCSEC]:  -1.2359  0.0440
# MusePipeInfo        Rotation [DEGREE]:  -0.1379
# MusePipeInfo Renormalising the data as: Normalised = 9.3987e-01 * (-1.1050e+00 + MUSE)
```

Fig. 12: saving the offset table after the first frame is aligned does not affect `myalign.offset_and_compare_ima()`.

Once you are happy with the result, please update the DR excel with the final offsets. You find the spreadsheet at [this link].

If you are not running the alignment the first time, but simply want to check the best solution (for example is someone else has done the alignment), please **DO NOT** re-run `step2_align()`. Instead, check that your script has a `check_align()` option (see Fig. 12a). If not, add it to the script and then comment `step2_align()`, uncomment `check_align()` and then re-run the DR script. Once the current offset are printed out, you can proceed testing the alignment as described above.

```
def check_align():
    global myalign
    myalign = AlignMuseDataset(folder_reference=refFold, folder_muse_images=alignFold,
                               folder_offset_table=tabFold, filter_name=alignFilter,
                               filter_suffix=alignFilter,
                               ref_unit= refUnitAlign,
                               pivot_lambda = pivotLambdaAlign,
                               folder_output_table=oTabFold, name_reference=imAlignRef,
                               firstguess='fits', name_offset_table=offsetTableName)


if __name__ == "__main__":
    # step1()
    # step2_align()
    check_align()
    # step2_flux()
    # step3()
    # step4()
```

Fig. 12a: function to check alignment solution without re-running everything

DR step 2: alignment using a pre-identified solution

With the survey progressing, it will happen that you need to re-reduce the data but are happy with the final alignment solution you have found before. If so, there is a way to automatically use what you have recovered in previous data reduction efforts.

First, identify the folder where the offset table you want to use is stored. For example, let's assume you want to use the solution you found in the `red_v2` run for a new reduction that is happening in the `red_v3` folder.

- Go in the Alignment tables folder for your current reduction:
  - `cd /cloud/teamdata/mauve/red_v3/Data/MAUVE/Alignment_tables/`
- Copy here the offset table you want to use (e.g., for galaxy NGC4064)
  - `cp /cloud/teamdata/mauve/red_v2/Data/MAUVE/Alignment_tables/NGC4064_offset_table_red_v2.fits .`
- Rename the table to be relevant for the current data reduction
  - `mv NGC4064_offset_table_red_v2.fits NGC4064_offset_table_red_v3.fits`

Then,

- Open `reduce_v1_NGC4064.py`, comment "`step1()`" and uncomment `check_align()`
- In the python terminal: `%run -i reduce_v1_NGC4064.py`

You should see appearing the results of the alignment, and you should check that these match those recorded on the spreadsheet at this link during the previous reduction.

If all is good you should proceed to the flux rescaling step. Please keep in mind that, generally, while the alignment solution from previous reductions can be used, it is good to always re-do the flux rescaling.

DR step 2 (flux): flux rescaling

This step uses the SDSS images to determine the rescaling values for sky background and flux (see Appendix C for more details).

- Open `reduce_v1_NGC4064.py`, comment "`step2_align()`" and uncomment "`step2_flux()`". Save.
- In the python terminal: `%run -i reduce_v1_NGC4064.py`

At this point, the optical flow is re-run using the offset obtained in the previous step and background rescaling and flux normalisation factors are computed. The normalization factors are printed out to the terminal (Fig. 13).

- Check the value of the normalization factor for each frame. This should be close to 1… and *definitely* not negative. A negative value indicates that there is an issue with the MUSE-reference flux comparison.

```
# MusePipeInfo Normalisation factors
# MusePipeInfo Image # : InitFluxScale     SlopeFit       NormFactor       Background
# MusePipeInfo Image #001/i=000:  1.000000e+00   9.404216e-01     9.404216e-01     -1.193129e+00
# MusePipeInfo Image #002/i=001:  1.000000e+00   9.455295e-01     9.455295e-01     -6.933543e-01
# MusePipeInfo Image #003/i=002:  1.000000e+00   9.435129e-01     9.435129e-01     9.047140e-01
# MusePipeInfo Image #004/i=003:  1.000000e+00   9.549765e-01     9.549765e-01     9.562511e-01
# MusePipeInfo Image #005/i=004:  1.000000e+00   9.986123e-01     9.986123e-01     -3.570764e+00
# MusePipeInfo Image #006/i=005:  1.000000e+00   1.031810e+00     1.031810e+00     -1.338833e+00
# MusePipeInfo Image #007/i=006:  1.000000e+00   1.005664e+00     1.005664e+00     3.951631e-01
# MusePipeInfo Image #008/i=007:  1.000000e+00   1.022960e+00     1.022960e+00     8.449155e-01
```

Fig. 13: flux rescaling.

- If all is good, save the table by running:
  `myalign.save_fits_offset_table(overwrite="True")`

Before proceeding to the next step, make sure to do a few quick checks following the instructions on our QC cookbook.

DR step 3: creation of individual cubes

Now everything should be aligned and rescaled in flux, and you are ready to finalise the reduction! This step involves a slightly readjustment of the sky subtraction, resampling to a common reference WCS grid, and creation of individual cubes (one per OB), after taking the alignment into account. It takes ~2h per OB.

- Open `reduce_v1_NGC4064.py`, comment "`step2_flux()`" and uncomment "`step3()`". Save.
- In the python terminal: `%run -i reduce_v1_NGC4064.py`

The products are saved in the `[dr_home]/Data/MAUVE/Muse/NGC4054/Combined/Cubes/` directory (Fig. 14). Note that there are 2 r-band images (one for each combined OB) that can be used for a second check on the alignment. Please, follow the instructions in our QC cookbook.

```
(PLPenv2) bcatinella@dccompute2:/cloud/teamdata/mauve/red_test_barbara/Data/MAUVE/Muse/NGC4064/Combined/Cubes$ ls
fullrefwcs_NGC4064_DATACUBE_FINAL.fits                  NGC4064_IMAGE_FOV_WCS_OB002_SDSS_r_2023-03-27T03:22:49_0002.fits
NGC4064_DATACUBE_FINAL_P001.fits                        NGC4064_IMAGE_FOV_WCS_OB002_SDSS_r_2023-03-27T03:22:49_0003.fits
NGC4064_DATACUBE_FINAL_P002.fits                        NGC4064_IMAGE_FOV_WCS_OB002_SDSS_r_2023-03-27T03:22:49_0004.fits
NGC4064_DATACUBE_FINAL_WCS_OB001_2023-03-26T02:52:02_0001.fits  NGC4064_IMAGE_FOV_WCS_OB002_white_2023-03-27T03:22:49_0001.fits
NGC4064_DATACUBE_FINAL_WCS_OB001_2023-03-26T02:52:02_0002.fits  NGC4064_IMAGE_FOV_WCS_OB002_white_2023-03-27T03:22:49_0002.fits
NGC4064_DATACUBE_FINAL_WCS_OB001_2023-03-26T02:52:02_0003.fits  NGC4064_IMAGE_FOV_WCS_OB002_white_2023-03-27T03:22:49_0003.fits
NGC4064_DATACUBE_FINAL_WCS_OB001_2023-03-26T02:52:02_0004.fits  NGC4064_IMAGE_FOV_WCS_OB002_white_2023-03-27T03:22:49_0004.fits
NGC4064_DATACUBE_FINAL_WCS_OB002_2023-03-27T03:22:49_0001.fits  NGC4064_mask_DATACUBE_FINAL_P001.fits
NGC4064_DATACUBE_FINAL_WCS_OB002_2023-03-27T03:22:49_0002.fits  NGC4064_mask_DATACUBE_FINAL_P002.fits
NGC4064_DATACUBE_FINAL_WCS_OB002_2023-03-27T03:22:49_0003.fits  NGC4064_mask_IMAGE_FOV_P001_white.fits
NGC4064_DATACUBE_FINAL_WCS_OB002_2023-03-27T03:22:49_0004.fits  NGC4064_mask_IMAGE_FOV_P002_white.fits
NGC4064_IMAGE_FOV_P001_Cousins_R.fits                   NGC4064_PIXTABLE_REDUCED_OB001_2023-03-26T02:52:02_0001.fits
NGC4064_IMAGE_FOV_P001_white.fits                       NGC4064_PIXTABLE_REDUCED_OB001_2023-03-26T02:52:02_0002.fits
NGC4064_IMAGE_FOV_P002_Cousins_R.fits                   NGC4064_PIXTABLE_REDUCED_OB001_2023-03-26T02:52:02_0003.fits
NGC4064_IMAGE_FOV_P002_white.fits                       NGC4064_PIXTABLE_REDUCED_OB001_2023-03-26T02:52:02_0004.fits
NGC4064_IMAGE_FOV_WCS_OB001_SDSS_r_2023-03-26T02:52:02_0001.fits  NGC4064_PIXTABLE_REDUCED_OB002_2023-03-27T03:22:49_0001.fits
NGC4064_IMAGE_FOV_WCS_OB001_SDSS_r_2023-03-26T02:52:02_0002.fits  NGC4064_PIXTABLE_REDUCED_OB002_2023-03-27T03:22:49_0002.fits
NGC4064_IMAGE_FOV_WCS_OB001_SDSS_r_2023-03-26T02:52:02_0003.fits  NGC4064_PIXTABLE_REDUCED_OB002_2023-03-27T03:22:49_0003.fits
NGC4064_IMAGE_FOV_WCS_OB001_SDSS_r_2023-03-26T02:52:02_0004.fits  NGC4064_PIXTABLE_REDUCED_OB002_2023-03-27T03:22:49_0004.fits
NGC4064_IMAGE_FOV_WCS_OB001_white_2023-03-26T02:52:02_0001.fits   refwcs_NGC4064_DATACUBE_FINAL.fits
NGC4064_IMAGE_FOV_WCS_OB001_white_2023-03-26T02:52:02_0002.fits   refwcs_NGC4064_DATACUBE_FINAL_P001.fits
NGC4064_IMAGE_FOV_WCS_OB001_white_2023-03-26T02:52:02_0003.fits   refwcs_NGC4064_DATACUBE_FINAL_P002.fits
NGC4064_IMAGE_FOV_WCS_OB001_white_2023-03-26T02:52:02_0004.fits   refwcs_NGC4064_IMAGE_FOV_white.fits
NGC4064_IMAGE_FOV_WCS_OB002_SDSS_r_2023-03-27T03:22:49_0001.fits
(PLPenv2) bcatinella@dccompute2:/cloud/teamdata/mauve/red_test_barbara/Data/MAUVE/Muse/NGC4064/Combined/Cubes$
```

Fig. 14: combined cubes from step 3 of the data reduction script.

Satellite trails

Some time it will happen that one (or more) exposures within one OB are contaminated by trails from satellites. You will generally notice this during the alignment stage of the reduction. If this is the case, this is the time to mask them before the final mosaic is created.

The procedure is relatively simple:

- In your `../Combined/Cubes/ folder` copy the fix_satellite script
  - `cp /cloud/teamdata/mauve/scripts/pmp/fix_satellite_OB00X_000X.py .`

- Rename the script to keep track of the OB and individual exposure you are fixing: e.g., if you are fixing the third exposure of OB002
  - `mv fix_satellite_OB00X_000X.py  fix_satellite_OB002_0003.py`

The script will look like the figure below.

```
# Fixing trails ###################
# import needed library
from pymusepipe.mpdaf_pipe import MuseCube

### call datacube for the frame that needs to be masked
c = MuseCube(filename="NGC4501_DATACUBE_FINAL_WCS_OB006_2023-05-21T02:19:38_0003.fits")

# Then record the coordinates of the two extremes (p and q) ## p is y and q is x
pq1 = [202, 30]
pq2 = [289, 394]
# Creating the mask with these pq's
c.mask_trail(pq1=pq1, pq2=pq2, width=10, margin=2)
### check that a tmask cube has been created


# Check that your mask is in the correct position
nc = MuseCube(filename="tmaskNGC4501_DATACUBE_FINAL_WCS_OB006_2023-05-21T02:19:38_0003.fits")
ima = nc[1500:1550,:,:].sum(axis=0)
ima.plot()
```

Fig. 15: Satellite trails masking script.

We suggest not to run it as a script, but to paste every command on a python terminal to make sure every step is done correctly.

- Replace the filename in the `c` variable as the name of the fits cube you need to mask.
- Identify the coordinates in pixels of the two extremes of the trail and update `pq1` and `pq2`.
- Then, cut and paste in a python terminal all the lines up to the `c.mask_trail` command.
- Check that a `tmask` file has been created.
- Update the definition of the `nc` variable in the script with the `tmask` created.
- Run the last 3 lines of the script to check that the trail has been properly masked.
- This will create a figure like the one below.
- If not, fix play with the pq1, pq2, widths and margin coordinates to improve the situation.

Once all is good, you can move to the next step.

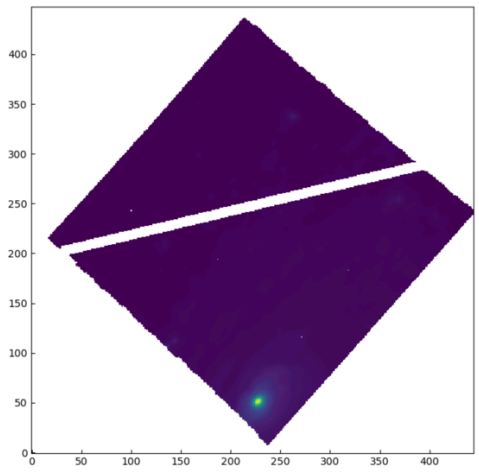


Fig. 16: Plot created to check correct masking of satellite trail.

DR step 4: create the final mosaic

The last step consists of combining the individual cubes (one per OB), already resampled to a common grid, into the final mosaic. It takes 0.5 h for mosaicking 2 OBs.

- Open `reduce_v1_NGC4064.py`, comment "`step3()`" and uncomment "`step4()`". Save.
- `%run -i reduce_v1_NGC4064.py`

The final result is saved as:
`[dr_home]/Data/MAUVE/Muse/NGC4054/Combined/Cubes/fullrefwcs_NGC4064_DATACUBE_FINAL.fits` (see Fig. 14).

Please make sure to do the last few quick checks following the instructions on our QC cookbook.

Congratulations, you're done!!

Troubleshooting

We document here some common errors and how to address these.

- **ICE default IO error handler doing an exit()**. Sometimes step 2 (pre-alignment) crashes with this error (see Fig. T1). This is a system error. To fix:
  - Open a new terminal and do a `ls -a` in your home directory to see the hidden files.
  - Delete the `.ICEauthority` and `.Xauthority` files from your home directory.
  - Close the ipython terminal that you were using, start a new one, and rerun whatever process made it crash.

Fig. T1: ICE default IO error.

- **Session management error, authentication rejected**. This happened during the manual alignment (see Fig. T2) but did not seem to affect anything.

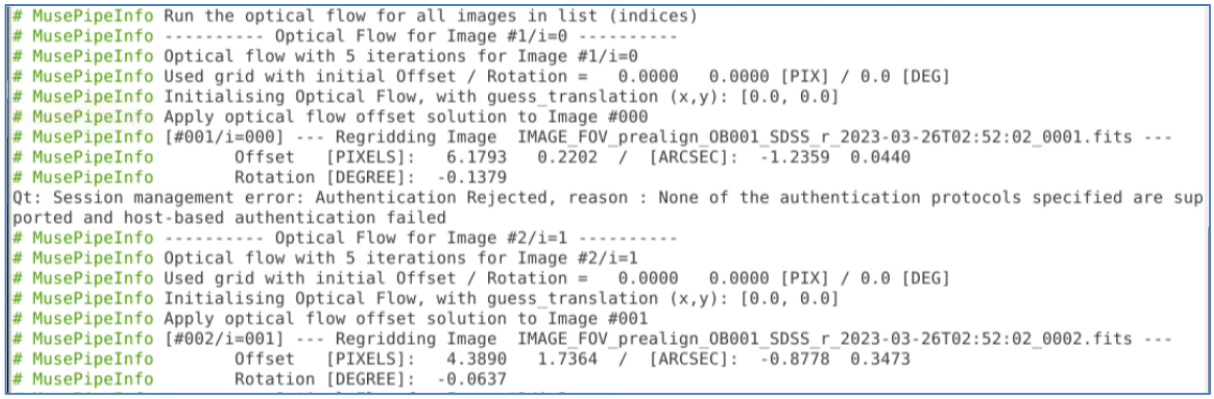


Fig. T2: Session management error.

- **LSF failing**. It is possible that for one (or more) calibrations the `muse_lsf` task will fail. If other calibrations are available and the LSF is obtained correctly, then all should be fine as pymusepipe will only use the correct calibrations for the following steps. In this case, you will realise that something has happened only when looking at the .err log files (see Fig.T3).

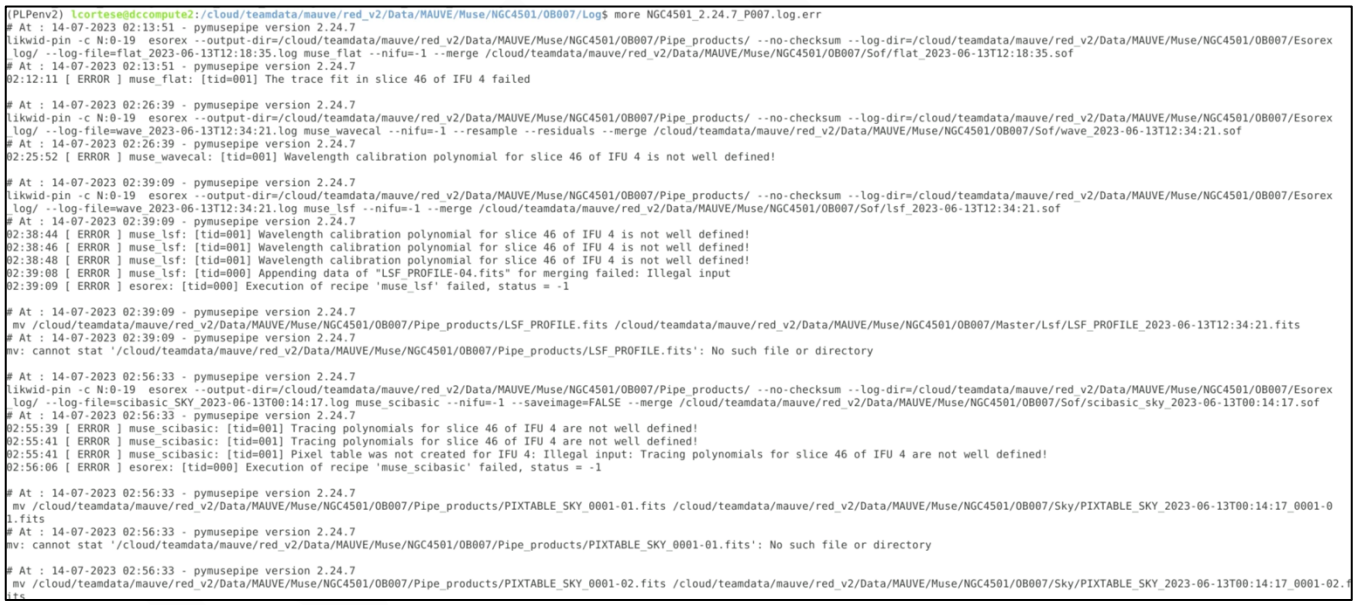
Fig. T3: Extract of `.err` log showing that the `muse_lsf` task failed.

- **Major crash in the data reduction**. It is possible that pymusepipe will crash. This happens rarely but, when it does, it is important to figure out why and try to find ways to solve the issue. Here an example on how we fixed the only time (so far) in which the script crashed. In Fig. T4, you can see what will happen if pymusepipe crashes.

```
    172             self._open_filelike(fileobj, mode, overwrite)

/cloud/teamdata/mauve/soft/miniconda3/envs/PLPenv2/lib/python3.7/site-packages/astropy/io/fits/file.py in _open_filename(self, filename, mode, overwrite)
    556
    557         if not self._try_read_compressed(self.name, magic, mode, ext=ext):
--> 558             self._file = fileobj_open(self.name, IO_FITS_MODES[mode])
    559             self.close_on_error = True
    560

/cloud/teamdata/mauve/soft/miniconda3/envs/PLPenv2/lib/python3.7/site-packages/astropy/io/fits/util.py in fileobj_open(filename, mode)
    388     """
    389
--> 390     return open(filename, mode, buffering=0)
    391
    392

FileNotFoundError: [Errno 2] No such file or directory: '/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Object/IMAGE_FOV_prealign_SDSS_r_2023-06-13T00:14:17_0001.fits'
```

Fig. T4: When pymusepipe crashes.

Your first step is to go under '`Log`' folder for the OB you are reducing (e.g., `NC4501/OB007/Log`) and look at the `.err` log. Go carefully back in time and try to identify when the first issue started. Please note that the script may crash at a different task than the one creating the issue in the first place. For example, in this case the crash happened at the `scibasic` step, but the problem started at `muse_flat`, where the `trace fit` failed (Fig. T5).



```
(PLPenv2) lcortese@dccompute2:/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Log$ more NGC4501_2.24.7_P007.log.err
# At : 14-07-2023 02:13:51 - pymusepipe version 2.24.7
likwid-pin -c N:0-19  esorex --output-dir=/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/ --no-checksum --log-dir=/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Esorex
_log/ --log-file=flat_2023-06-13T12:18:35.log muse_flat --nifu=-1 --merge /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Sof/flat_2023-06-13T12:18:35.sof
# At : 14-07-2023 02:13:51 - pymusepipe version 2.24.7
02:12:11 [ ERROR ] muse_flat: [tid=001] The trace fit in slice 46 of IFU 4 failed

# At : 14-07-2023 02:26:39 - pymusepipe version 2.24.7
likwid-pin -c N:0-19  esorex --output-dir=/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/ --no-checksum --log-dir=/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Esorex
_log/ --log-file=wave_2023-06-13T12:34:21.log muse_wavecal --nifu=-1 --resample --residuals --merge /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Sof/wave_2023-06-13T12:34:21.sof
# At : 14-07-2023 02:26:39 - pymusepipe version 2.24.7
02:25:52 [ ERROR ] muse_wavecal: [tid=001] Wavelength calibration polynomial for slice 46 of IFU 4 is not well defined!

# At : 14-07-2023 02:39:09 - pymusepipe version 2.24.7
likwid-pin -c N:0-19  esorex --output-dir=/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/ --no-checksum --log-dir=/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Esorex
_log/ --log-file=wave_2023-06-13T12:34:21.log muse_lsf --nifu=-1 --merge /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Sof/lsf_2023-06-13T12:34:21.sof
# At : 14-07-2023 02:39:09 - pymusepipe version 2.24.7
02:38:44 [ ERROR ] muse_lsf: [tid=001] Wavelength calibration polynomial for slice 46 of IFU 4 is not well defined!
02:38:46 [ ERROR ] muse_lsf: [tid=001] Wavelength calibration polynomial for slice 46 of IFU 4 is not well defined!
02:38:48 [ ERROR ] muse_lsf: [tid=001] Wavelength calibration polynomial for slice 46 of IFU 4 is not well defined!
02:39:08 [ ERROR ] muse_lsf: [tid=000] Appending data of "LSF_PROFILE-04.fits" for merging failed: Illegal input
02:39:09 [ ERROR ] esorex: [tid=000] Execution of recipe 'muse_lsf' failed, status = -1

# At : 14-07-2023 02:39:09 - pymusepipe version 2.24.7
 mv /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/LSF_PROFILE.fits /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Master/Lsf/LSF_PROFILE_2023-06-13T12:34:21.fits
# At : 14-07-2023 02:39:09 - pymusepipe version 2.24.7
mv: cannot stat '/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/LSF_PROFILE.fits': No such file or directory

# At : 14-07-2023 02:56:33 - pymusepipe version 2.24.7
likwid-pin -c N:0-19  esorex --output-dir=/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/ --no-checksum --log-dir=/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Esorex
_log/ --log-file=scibasic_SKY_2023-06-13T00:14:17.log muse_scibasic --nifu=-1 --saveimage=FALSE --merge /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Sof/scibasic_sky_2023-06-13T00:14:17.sof
# At : 14-07-2023 02:56:33 - pymusepipe version 2.24.7
02:55:39 [ ERROR ] muse_scibasic: [tid=001] Tracing polynomials for slice 46 of IFU 4 are not well defined!
02:55:41 [ ERROR ] muse_scibasic: [tid=001] Tracing polynomials for slice 46 of IFU 4 are not well defined!
02:55:41 [ ERROR ] muse_scibasic: [tid=001] Pixel table was not created for IFU 4: Illegal input: Tracing polynomials for slice 46 of IFU 4 are not well defined!
02:56:06 [ ERROR ] esorex: [tid=000] Execution of recipe 'muse_scibasic' failed, status = -1

# At : 14-07-2023 02:56:33 - pymusepipe version 2.24.7
 mv /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/PIXTABLE_SKY_0001-01.fits /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Sky/PIXTABLE_SKY_2023-06-13T00:14:17_0001-0
1.fits
# At : 14-07-2023 02:56:33 - pymusepipe version 2.24.7
mv: cannot stat '/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/PIXTABLE_SKY_0001-01.fits': No such file or directory

# At : 14-07-2023 02:56:33 - pymusepipe version 2.24.7
 mv /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Pipe_products/PIXTABLE_SKY_0001-02.fits /cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Sky/PIXTABLE_SKY_2023-06-13T00:14:17_0001-02.f
its
```

Fig. T5: Extract of `.err` log.

Then, check in the `.log` if the same task was successful for another set of calibrations. In other words, check if you have another `esorex` run pointing to (for this case) another `flatNN.sof` file that was successful. If this is the case, your best bet is to get rid of the problematic calibrations and re-run the script for that particular OB.

Here is how to do so.

- Go to the `Sof` folder and have a look at the `.sof` file used when the task crashed (see Fig. T6 for an example)

```
(PLPenv2) lcortese@dccompute2:/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Sof$ more flat_2023-06-12T12:51:30.sof
/cloud/teamdata/mauve/soft/esorex/calib/muse-2.8.7/badpix_table.fits BADPIX_TABLE
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T12:52:17.881.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T12:53:40.161.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T12:55:02.825.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T12:56:25.286.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T12:57:47.660.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T12:59:09.830.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T13:00:32.137.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T13:01:54.690.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T13:03:17.222.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T13:04:39.894.fits.fz FLAT
/cloud/teamdata/mauve/data/muse/NGC4501-7/MUSE.2023-06-12T13:06:02.285.fits.fz FLAT
/cloud/teamdata/mauve/red_v2/Data/MAUVE/Muse/NGC4501/OB007/Master/Bias/MASTER_BIAS_2023-06-12T11:34:46.fits MASTER_BIAS
```

Fig. T6: Example of `.sof` file.

- Identify all the problematic calibrations (in this case the flats)
- Go in the raw data folder for the OB you are working on magpipe:
  `e.g., cd /cloud/teamdata/mauve/data/muse/NGC4501-7`
- Create new folder and go there:
  ```
  mkdir dont_use_cals
  cd dont_use_cals
  ```
- Create a script to move all the problematic calibrations in this folder
  e.g., `sublime remove_bad_calib_NGC4501-7.sh` (see Fig. T7[1])

```
(PLPenv2) lcortese@dccompute2:/cloud/teamdata/mauve/data/muse/NGC4501-7/dont_use_cals$ more remove_bad_calib_NGC4501-7.sh
mv ../MUSE.2023-06-13T12:19:24.665.fits.fz .
mv ../MUSE.2023-06-13T12:20:46.929.fits.fz .
mv ../MUSE.2023-06-13T12:22:09.509.fits.fz .
mv ../MUSE.2023-06-13T12:23:32.021.fits.fz .
mv ../MUSE.2023-06-13T12:24:54.300.fits.fz .
mv ../MUSE.2023-06-13T12:26:16.592.fits.fz .
mv ../MUSE.2023-06-13T12:27:38.917.fits.fz .
mv ../MUSE.2023-06-13T12:29:01.285.fits.fz .
mv ../MUSE.2023-06-13T12:30:23.569.fits.fz .
mv ../MUSE.2023-06-13T12:31:45.914.fits.fz .
mv ../MUSE.2023-06-13T12:33:08.286.fits.fz .
```

Fig. T7: Example of script to remove problematic calibration files.

- Once the file is saved
  ```
  chmod +x remove_bad_calib_NGC4501-7.sh
  ./remove_bad_calib_NGC4501-7.sh
  ```

Now you are ready to try to re-run the script.
Just remember to:

- Create a reduction script for only the problematic OB
- Check whether the same calibrations are used for other OBs. If so, you'll need to remove them also from the other raw data folders.

[1] Please note that the list of files in Fig. T6 and T7 does not match. This simply because I (Luca) forgot to take a screenshot of the problematic `.sof` file before re-running the script and the original file was deleted.

- **Issue with DR script set-up**. It is possible (but very unlikely) that the script crashes as soon as it is launched and that the error resembles more a “python” error than a pymusepipe one. An example is illustrated below.

```
# MusePipeWarning Checkmode is True: the MUSE Mode will be checked.
# MusePipeWarning All Raw files which do NOT have musemode = WFM-NOAO-N and are mode specific (e.g., Flat field) will be masked.
# MusePipeWarning If you wish otherwise, set checkmode to False [but this may impact the data reduction].
# MusePipeInfo End of Pipe initialisation
# MusePipeInfo Will use image_registration for image regridding.
# MusePipeInfo Trying to create /cloud/teamdata/mauve/red_luca_vign/Data/MAUVE/Muse/IC3392/Alignment/AlignHeaders folder... Folder already exists, doing noth
ing.
# MusePipeInfo Trying to create /cloud/teamdata/mauve/red_luca_vign/Data/MAUVE/Muse/IC3392/Alignment/AlignFigures folder... Folder already exists, doing noth
ing.
# MusePipeWarning By default, rotation angles given in initial offset table will be used if they exist. If not, all initial rotation angles will be set to 0.
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/cloud/teamdata/mauve/red_luca_vign/Scripts/python/reduce_v1_IC3392.py in <module>
    158 if __name__ == "__main__":
    159         # step1()
--> 160          step2_align()
    161         # step2_flux()
    162         # step3()

/cloud/teamdata/mauve/red_luca_vign/Scripts/python/reduce_v1_IC3392.py in step2_align()
    104                             pivot_lambda = pivotLambdaAlign,
    105                             folder_output_table=oTabFold, name_reference=imAlignRef,
--> 106                             firstguess=None, name_offset_table=offsetTableName)
    107         myalign.run_optical_flow(provide_header=True)
    108         myalign.transfer_extra_to_guess()

~/.conda/envs/PLPtestv1/lib/python3.7/site-packages/pymusepipe/align_pipe.py in __init__(self, name_reference, folder_reference, folder_muse_images, name_mus
e_images, sel_indices_images, median_window, subim_window, dynamic_range, border, hdu_ext, chunk_size, firstguess, **kwargs)
    675             self.conversion_factor = get_conversion_factor(self.ref_unit,
    676                                                            self.muse_unit,
--> 677                                                            equivalency=self.equivalency)
    678         else:
    679             self.ref_unit = None

~/.conda/envs/PLPtestv1/lib/python3.7/site-packages/pymusepipe/align_pipe.py in get_conversion_factor(input_unit, output_unit, equivalency)
    208         else:
    209             return input_unit.unit.to(output_unit,
--> 210                                       equivalencies=equivalency) * input_unit.value
    211     else:
    212         return input_unit.unit.to(output_unit) * input_unit.value

~/.conda/envs/PLPtestv1/lib/python3.7/site-packages/astropy/units/core.py in to(self, other, value, equivalencies)
   1132         else:
   1133             return self._get_converter(Unit(other),
-> 1134                                        equivalencies=equivalencies)(value)
   1135
   1136     def in_units(self, other, value=1.0, equivalencies=[]):

~/.conda/envs/PLPtestv1/lib/python3.7/site-packages/astropy/units/core.py in convert(v)
    986         def make_converter(scale1, func, scale2):
    987             def convert(v):
--> 988                 return func(_condition_arg(v) / scale1) * scale2
    989             return convert
    990

~/.conda/envs/PLPtestv1/lib/python3.7/site-packages/astropy/units/equivalencies.py in iconverter(x)
    205
    206     def iconverter(x):
--> 207         return x / (wav.to_value(si.AA, spectral()) ** 2 / c_Aps)
    208
    209     def converter_f_nu_to_nu_f_nu(x):

AttributeError: 'float' object has no attribute 'to_value'

In [2]:
```

Fig. T8: Example of error due to issues in set-up of pmp script.

This could be due to minor changes to the pipeline you are now aware of and should **ONLY** happen if you are running the pipeline for the very first time. In this particular case, the issue was that the original script did not specify the unit of the pivot wavelengths, something required in new versions of the pipeline. If this happens:

- Take a screenshot of the error
- Try to identify where the issue is
- Get in touch with Luca/Barbara/Eric and share a copy of your script for assessment.

Appendixes

Appendix A. MUSE data

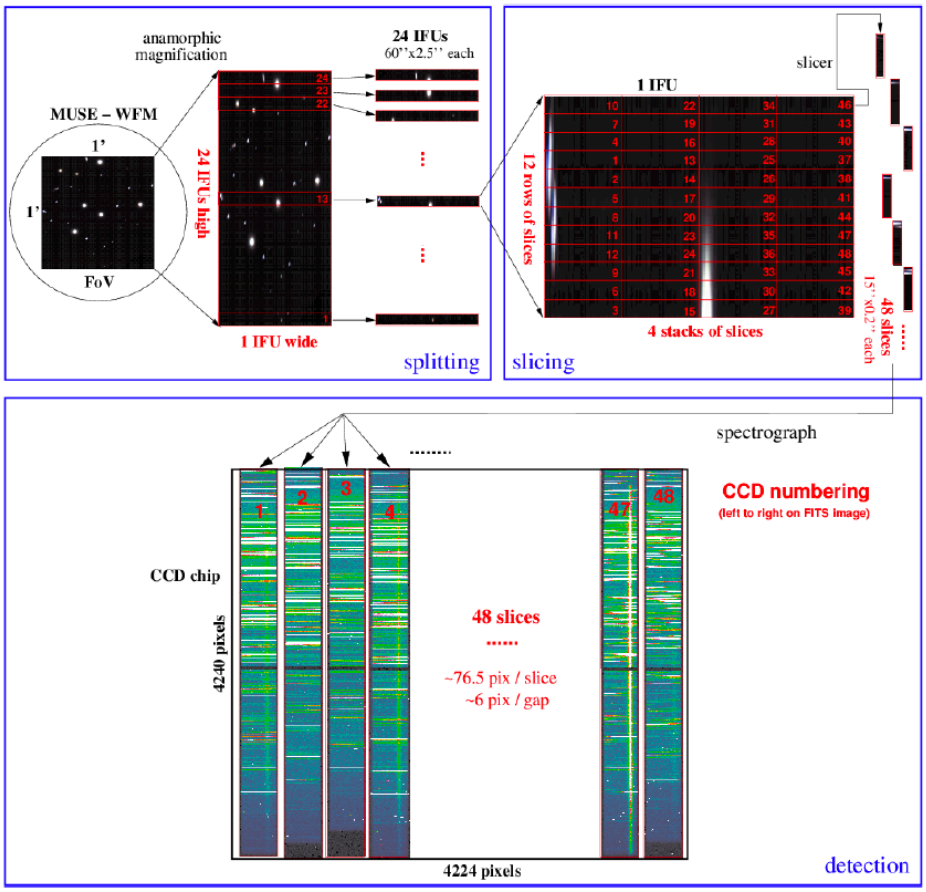


Figure 2.1: Graphical representations of the splitting and slicing procedures in the MUSE instrument. The example shown is for wide-field mode; narrow-field mode operates in the same way with a scaled-down field size. Note that the sizes given are approximate, the real data does not exactly cover a square region on the sky.

Fig. A1: MUSE data, from EsoRex manual.

Appendix B. EsoRex and `pymusepipe`

Fig. B1 and B2 illustrate the EsoRex standard data flow (from the EsoRex manual). Fig. B3 shows the data reduction flow of PHANGS-MUSE based on `pymusepipe`, a wrapper of EsoRex written by Emsellem.

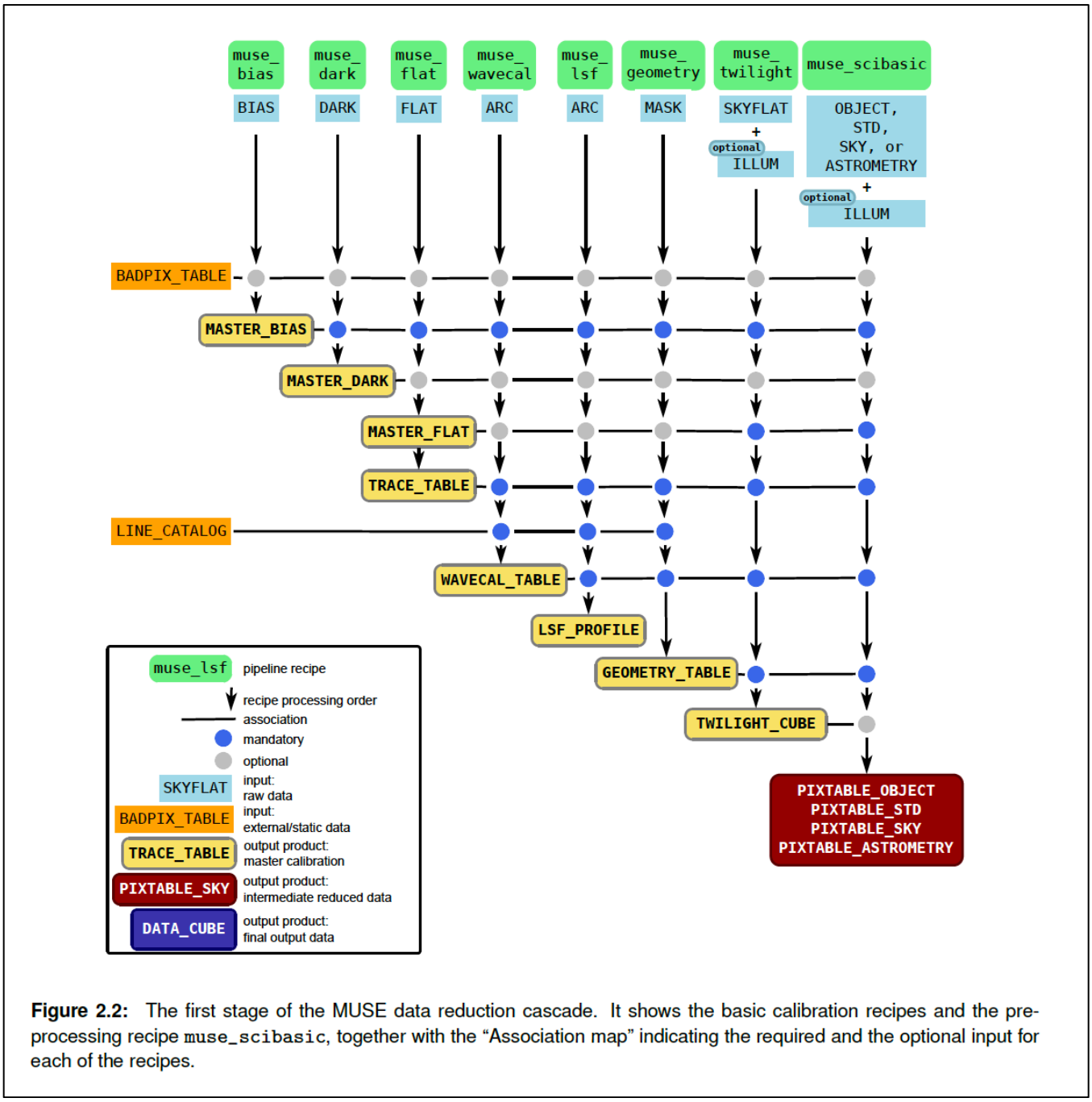


Figure 2.2: The first stage of the MUSE data reduction cascade. It shows the basic calibration recipes and the pre-processing recipe `muse_scibasic`, together with the "Association map" indicating the required and the optional input for each of the recipes.

Fig. B1: EsoRex data reduction, first stage (basic calibrations), from EsoRex manual.

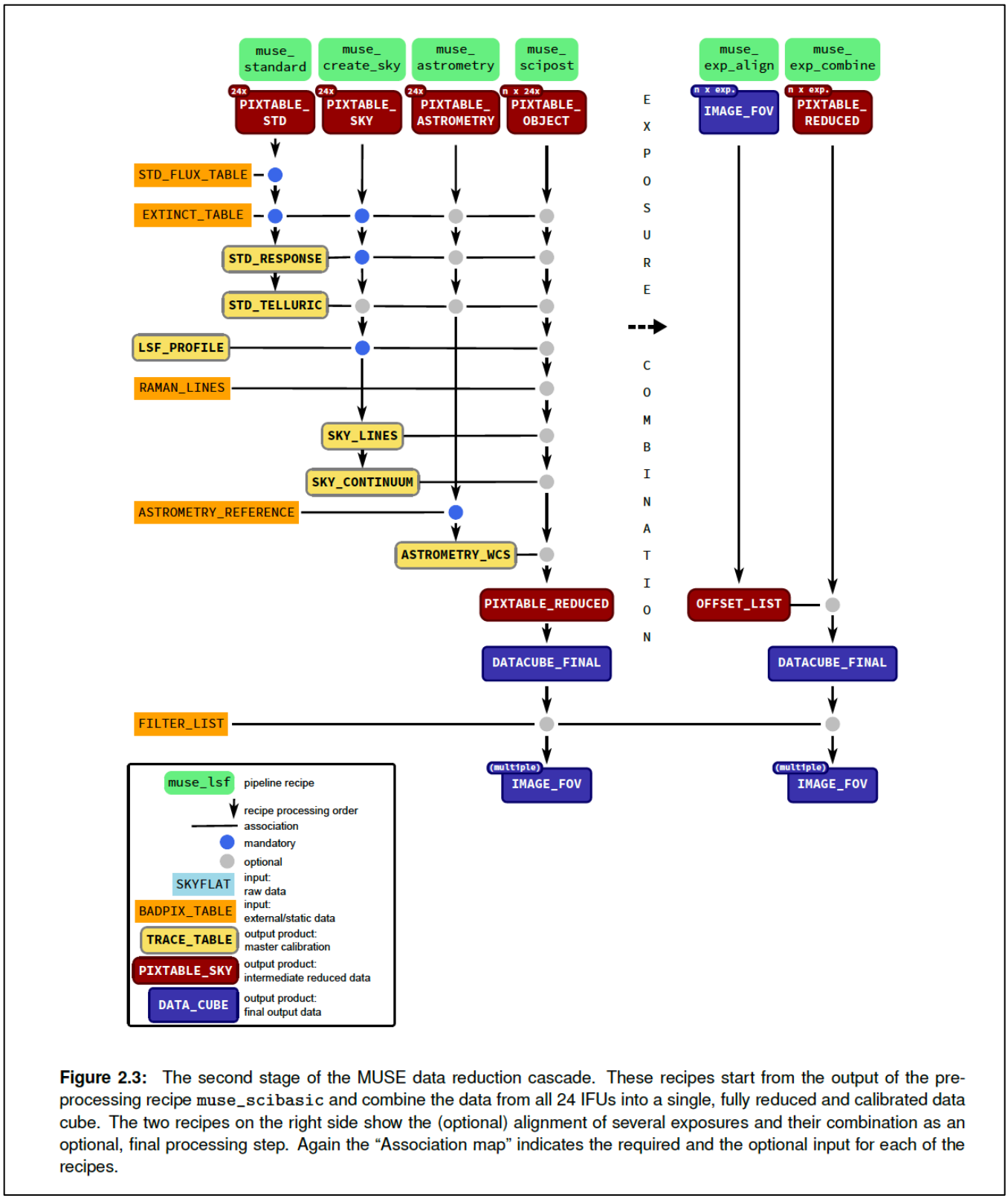


Figure 2.3: The second stage of the MUSE data reduction cascade. These recipes start from the output of the pre-processing recipe `muse_scibasic` and combine the data from all 24 IFUs into a single, fully reduced and calibrated data cube. The two recipes on the right side show the (optional) alignment of several exposures and their combination as an optional, final processing step. Again the "Association map" indicates the required and the optional input for each of the recipes.

Fig. B2: EsoRex data reduction, second stage (combining all IFUs), from EsoRex manual.

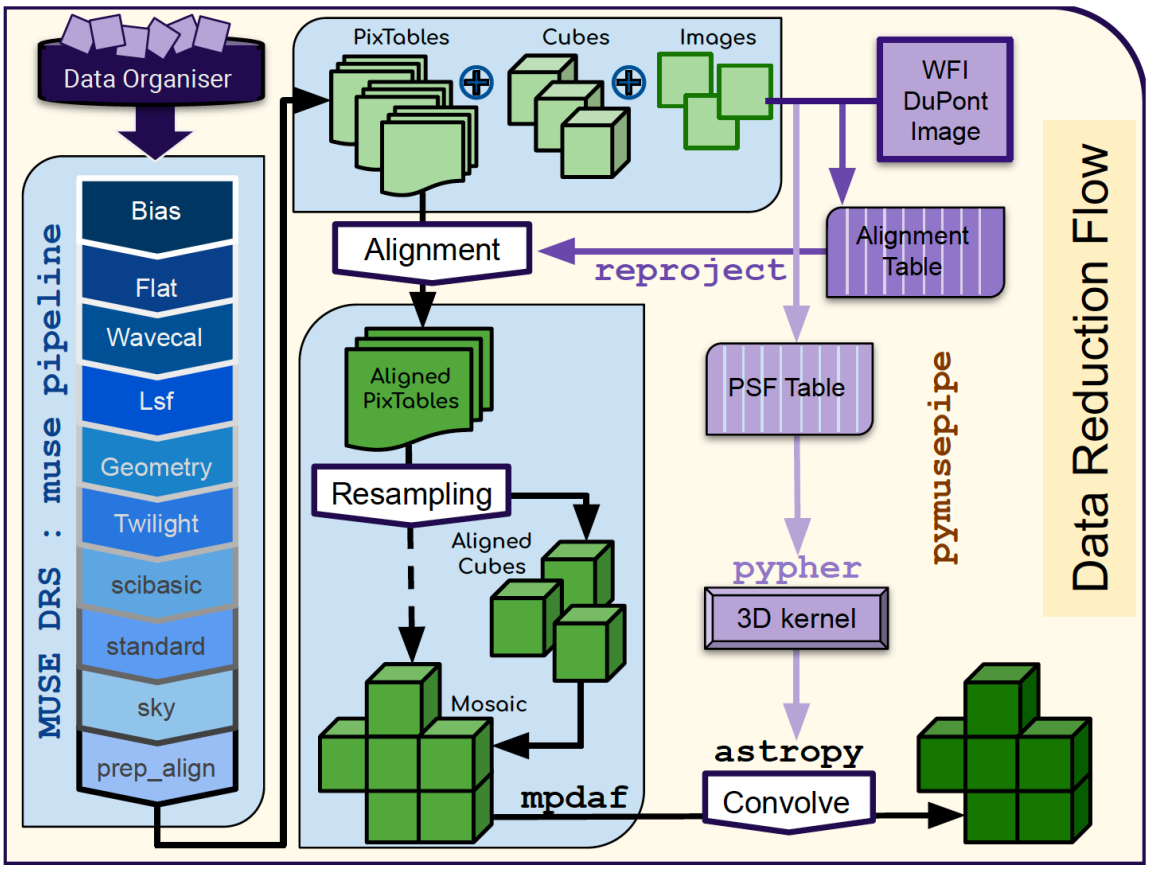


Fig. 7. Schematic data flow representing the PHANGS-MUSE data reduction part (DRP) of the pipeline. `pymusepipe` is used as the global process wrapper. The data flow includes a data organiser that addresses the raw data files, followed by standard MUSE data reduction steps (via `MUSE DRS`), which leads to the first reduced 'Pixel Tables' (PixTables) and cubes. These are aligned using WFI or Direct CCD reference images (via `reproject`). Aligned individual PixTables are then either resampled on a common grid or directly mosaicked. Optional convolution is performed using the framework from `mpdaf` and `astropy`, while `pypher` provides the required kernel cube.

Fig. B3: PHANGS-MUSE data reduction flow using `pymusepipe` (on which the MAUVE data reduction is based; from Emsellem et al. 2022).

Appendix C. Re-scaling procedure

Extract from Emsellem et al. 2022, explaining the details of the procedure described in `step2_flux()`.

$$\begin{aligned} \mathrm{Flux}^R(x,y) &= a \times \mathrm{MUSE}^R_1(x,y) \\ &= a \times \left(\mathrm{MUSE}^R_{\mathrm{raw}}(x,y) + \mathrm{Sky} - \mathrm{Sky}_1\right) + b, \end{aligned} \tag{1}$$

where Sky is a constant representing the true sky background for that specific exposure, Sky_1 is another constant representing the actual value removed during the initial sky subtraction process, and a and b are constants representing a linear regression representation of the $\mathrm{Flux}_{R-\mathrm{band}}$ versus the MUSE reconstructed image. A perfect sky subtraction and normalisation would lead to $a = 1$ and $b = 0$. We then use the fitted a value as a normalisation correction, and b to fix the sky contribution by applying $\mathrm{Sky} = \alpha \times \mathrm{Sky}_1$ where $\alpha = 1 - b/(a \cdot \mathrm{Sky}_1)$. Hence, knowing a and b as well as Sky_1, the value of the sky continuum integrated within the reference image filter, we derive a correction for the sky normalisation that yields a linear regression where $b = 0$. The `pymusepipe` package implements this approach as an option, using the recorded linear regression a and b values. The regression itself is performed via an orthogonal distance regression (ODR) comparing the reference and MUSE reconstructed images after noise filtering and binning: we use bins of 15×15 spaxels ($3'' \times 3''$) to minimise the impact of unresolved structure in the comparison.

We find that the distribution of the scaling factors a over the full set of PHANGS-MUSE exposures is well fit by a skew-normal distribution with location 0.99, scale of 0.06 and shape parameter α (related to the skewness) of 1.5 (the best Gaussian fit has a mean of 1.03 and sigma of 0.046). Only 28 (respectively, 27) exposures out of 676 have scaling factors lower than 0.9 (respectively, higher than 1.2). The distribution of background values (b) resembles a Gaussian function centred on 0 with a FWHM of about 1.4 (in units of 10^{-20} erg cm^{-2} s^{-1} Å^{-1}), with a small tail towards positive values. It is important to note that the sky re-normalisation only acts within the R-band filter, assuming that the reference image is background free. Since the reference