MUSEpack Documentation

Release 1.1.1dev20210128

Peter Zeidler

Oct 19, 2021

Contents:

CHAPTER 1

Introduction

MUSEpack is a python package written to support the data analyzes from Integral Field Units, specifically tailored to use datasets of the Multi Unit Spectroscopic Explorer (MUSE) mounted at UT4 of the VLT (Bacon et al. 2010).

The main purpose of MUSEpack is to measure stellar and gas radial velocities to an accuracy of $1 - 2\,\mathrm{km\,s\,s^{-1}}$ without the need for spectral template libraries using `radial_velocities.RV_spectrum`. With strong stellar absorption lines or gas emission lines and a high-resolution spectral line library such as the NIST Atomic Spectra Database it is possible to create templates using the observed spectra, which can be cross correlated with the spectra using a Monte Carlo method. For a detailed description and the citation of the code we refer to Zeidler et al. 2019.

On this website we introduce the individual python classes and modules of *MUSEpack*. In *Examples* we present detailed instructions on how to use the main modules and classes of *MUSEpack*. For any suggestions and bug reports please create a pull request in GitHub https://github.com/pzeidler89/MUSEpack or send an email to peterzeidler89@gmail.com.

1.1 Installing MUSEpack

To install MUSEpack download the latest repository from https://github.com/pzeidler89/MUSEpack MUSEpack is a Python3 only package and will not be supported for any version less than 3.6.

1.1.1 Additional python packages

To fully run all modules the following **non-standard** modules need to be installed:

- pyspeckit
- ppxf
- pysynphot

1.1.2 Additional 3rd party software

To properly run *MUSEreduce.musereduce* the ESO's MUSE data reduction pipeline v2.8.1 has to be installed. The latest version including the manual and further instructions can be found under: https://www.eso.org/sci/software/pipelines/muse/.

1.2 MUSEreduce

MUSEreduce.musereduce is an easy-to-use python Class used as wrapper for the VLT/MUSE data reduction pipeline and does not replace the core functionalities of the pipeline provided by ESO. In order to function properly with this version of *MUSEreduce.musereduce* we recommend to install the pipeline version v2.8.1 found under: https://www.eso.org/sci/software/pipelines/muse/. This version of *MUSEreduce.musereduce* has not been tested with other version of the data reduction pipeline.

To run *MUSEreduce.musereduce* all of the raw data must be stored in a folder named *user_path/raw/OB_ID*, where *OB_ID* must be a unique name for each individual OB. It is not mandatory (but recommended) to use the same nomenclature as as in the fits header. The script has to point to *user_path* by setting the keyword `rootpath` of the `json` config file to the parent directory.

class MUSEreduce.**musereduce**(*configfile*, *debug=False*)

Kwargs:

configfile [`str`] A `json` configfile for musereduce, where all the parameters are set.

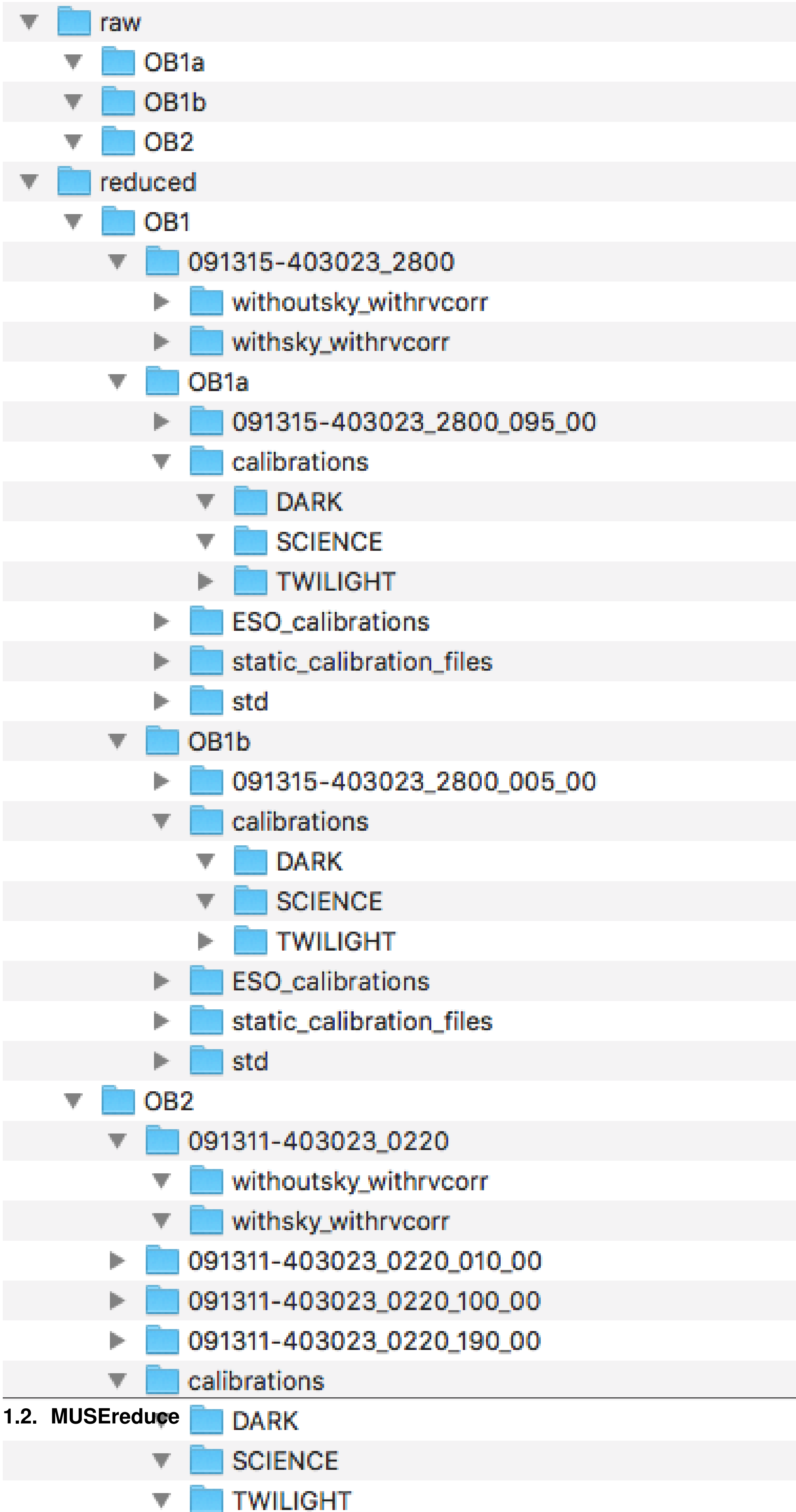
debug [`bool`, (optional), default: `None`] `True`: *MUSEreduce.musereduce* runs in debug mode and ESORex will not be executed. All products must be available. It can be used for testing the creation of folder, and creating the `.sof` files

execute()

This method executes wrapper and starts the data reduction process set in the `json` config file.

1.2.1 The basic folder structure

If `auto_sort_data` = `True` in the config file and *MUSEreduce.musereduce* is executed, the basic folder structure is created in the `rootpath`. As example, we consider that three OBs were observed: OB1a, OB1b, and OB2. OB1a and OB1b are two dither positions (with rotation angles of 5 and 95 deg and an exposure time of 2800s) of the same pointing OB1, while OB2 consists of three dither postions observed in one OB (with rotation angles of 10, 100, and 190 deg and an exposure time of 220s). The folder structure looks the following:


raw
OB1a
OB1b
OB2
reduced
OB1
091315-403023_2800
withoutsky_withrvcorr
withsky_withrvcorr
OB1a
091315-403023_2800_095_00
calibrations
DARK
SCIENCE
TWILIGHT
ESO_calibrations
static_calibration_files
std
OB1b
091315-403023_2800_005_00
calibrations
DARK
SCIENCE
TWILIGHT
ESO_calibrations
static_calibration_files
std
OB2
091311-403023_0220
withoutsky_withrvcorr
withsky_withrvcorr
091311-403023_0220_010_00
091311-403023_0220_100_00
091311-403023_0220_190_00
calibrations
DARK
SCIENCE
TWILIGHT

Each *master* OB (*OB1*, *OB2*) has its own folder in the *reduced* folder. Pointing *OB1* consists of the two OBs *OB1a* and *OB1b*. Each OB has the following folders:

- Each individual exposure (e.g., *091315-402023_2800_005_00*) following the structure: **RADEC_EXPTIME_ROTANGLE_COUNTER**. **RADEC** are the coordinates in sexagesimal, **EXPTIME** is the exposure time in seconds and the **ROTANGLE** is the rotation angle in degrees. The **COUNTER** is needed if there are two exposures with the same configuration.
- Each individual pointing (e.g., *091315-402023_2800*) following the structure: **RADEC_EXPTIME**.
- The *calibrations* contain *DARK*, *TWILIGHT*, and *SCIENCE*, in case the calibration files are created from the raw calibration files provided by ESO Therefore, the calibration steps must be executed.
- The *ESO_calibrations* is the folder, into which the reduced calibration files delivered by ESO (if available) are copied.
- The *static_calibration_files*, is the folder, into which the statics (part of the MUSE data reduction pipeline installation) are copied.
- The folder *std* will contain the reduced data of the standard star.

Note: The calibration files and the standard star are the same per individual OB and are only needed once.

1.2.2 The config file

The `json` configuration file is needed to run *MUSEreduce.musereduce*. This file contains all of the data reduction setup. The `json` configuration file can be found in the main directly of MUSEpack and can be downloaded `here`.

```
{
  "global": {
    "pipeline_path": "path_to_pipeline/muse/2.8.1/",
    "mode": "WFM-NOAO",
    "withrvcorr": true,
    "auto_sort_data": true,
    "using_specific_exposure_time": false,
    "dither_multiple_OBs": false,
    "n_CPU": -1,
    "rootpath": "path_to_data_folder",
    "OB": "",
    "OB_list": []
  },
  "calibration":{
    "execute": true,
    "using_ESO_calibration": true,
    "renew_statics": false,
    "dark": false,
    "create_sof": true
  },
  "sci_basic":{
    "execute": true,
    "execute_std": true,
    "skyreject": "15.,15.,1",
    "skylines": "5577.339,6300.304",
    "create_sof": true
  },
```

(continues on next page)

(continued from previous page)

```
  "std_flux":{
    "execute": true,
    "create_sof": true
  },
  "sky":{
    "execute": true,
    "modified": false,
    "sky_field": "auto",
    "fraction": 0.05,
    "ignore": 0.05,
    "method": "model",
    "create_sof": true
  },
  "sci_post":{
    "execute": true,
    "autocalib": "none",
    "subtract_sky": true,
    "raman": false,
    "create_sof": true
  },
  "dither_collect":{
    "execute": true,
    "user_list": []
  },
  "exp_align":{
    "execute": true,
    "srcmin": 5,
    "srcmax": 80,
    "create_sof": true
  },
  "exp_combine":{
    "execute": true,
    "weight": "exptime",
    "create_sof": true
  }

}
```

The file `config file` is structured the following. If keywords are directly controlling toggles of the MUSE data reduction pipeline their naming is identical.

config

Object Properties

- **global** (*global*) – global parameters affecting the data reduction process
- **calibration** (*calibration*) – parameters that affect the calibration steps of the MUSE data reduction pipeline
- **sci_basic** (*sci_basic*) – the preprocessing of the science exposures
- **std_flux** (*std_flux*) – the flux calibration
- **sky** (*sky*) – the sky subtraction
- **sci_post** (*sci_post*) – the postprocessing step of the data reduction
- **dither_collect** (*dither_collect*) – collecting the individual data cubes and pixel tables to combine them to one pointing

- **exp_align** (*exp_align*) – aligning the individual data cubes and pixel tables to one common WCS
- **exp_combine** (*exp_combine*) – combining the individual data cubes and pixel tables to one final science product

global

Object Properties

- **pipeline_path** (*string*) – The absolut path to the MUSE data reduction pipeline installation folder.
- **mode** (*string*) – The observation mode the data was obtained with. (*WFM-NOAO*, *WFM-AO*, *NFM-AO*)
- **withrvcorr** (*bool*) – bariocentric correction. Needs to be turned off, if one wants run an own wavelength calibration (*true*, *false*, *default='true'*)
- **auto_sort_data** (*bool*) – The raw data is sorted and the calibration files are assigned based on their header information. If `True`, the file lists *ID_DAR.list*, *ID_SCI.list*, and *ID_TWI.list* are created. If these have to be altered manually (e.g., using different calibration files), we recommend to run it first with `auto_sort_data` = `True`, then make the changes accordingly and from this point on set `auto_sort_data` = `False`. (*false*, *true*, *default=true*)
- **using_specific_exposure_time** (*float*) – The user can choose to only reduce a specific exposure time, if the same OB contains multiple exposures with different exposure times (e.g., long and short exposures)
- **dither_multiple_OBs** (*bool*) – Each OB is normally limited to a total exposure time of one hour. Therefore, one pointing may be distributed via multiple OBs. If `dither_multiple_OBs` = `True` it is possible to dither exposures from multiple OBs. In this case one must provide an `OB_list`. (*false*, *true*, *default=false*)
- **n_CPU** (*int*) – The number of CPUs used to reduce the data. If set to -1 all available cores are used. (*default=-1*)
- **rootpath** (*string*) – The absolut path in which the *raw* folder is located and in which the *processed* folder will be created.
- **OB** (*string*) – The name of the OB that shall be reduce. It must identical to the OB_ID given in the *raw* folder.
- **OB_list** (*list*) – If `dither_multiple_OBs` = `True` one must give a `string` list of OB_IDs, which will be dithered in the end. The calibration and data reduction runs on each individual OB to take into account different calibration files. (*default=[]*)

calibration

Object Properties

- **execute** (*bool*) – Must be set `True` if this step should be executed (*false*, *true*, *default=true*)
- **using_ESO_Calibration** (*bool*) – If the user wants to use the ESO calibration files (recommended if available) instead of running the calibration themselves (BIAS, DARK, FLAT, WAVECAL, LSF, TWILIGHT) it must be set to `True` (*false*, *true*, *default=true*)
- **renew_statics** (*bool*) – Set to `True` if the static calibration files should be copied again from the MUSE data reduction pipeline folder. This is necessary if the installed version of the pipline changes and one wants to obtain the latest static calibrations (*false*, *true*, *default=false*)

- **dark** (*bool*) – Set to `True` if one wants to also use the DARK files from the calibration. (*false*, *true*, *default=false*)
- **create_sof** (*bool*) – Must be set `True` if one wants to create a new *.sof* file. In case the user wants to change or create the *.sof* file manually, than it must be set to `False` since it will be overwritten otherwise (*false*, *true*, *default=true*)

sci_basic

Object Properties

- **execute** (*bool*) – Must be set `True` if this step should be executed (*false*, *true*, *default=true*)
- **execute_std** (*bool*) – Must be set `True` if the STD star should be reduced (*false*, *true*, *default=true*)
- **sky_reject** (*string*) – The sigma clipping parameters for the Gaussian fit to each sky emission line: *high sigma clipping limit* (`float`), *low sigma clipping limit* (`float`), *number of iterations* (`int`). For a more detailed description we refer to the MUSE data reduction pipeline manual. (*default='15., 15., 1'*)
- **skylines** (*string*) – The sky lines used to calibrate the wavelength offset in each IFU. the default are two lines but more strong skylines may be provided. For a more detailed description we refer to the MUSE data reduction pipeline manual. (*default='5577.339, 6300.304'*)
- **create_sof** (*bool*) – Must be set `True` if one wants to create a new *.sof* file. In case the user wants to change or create the *.sof* file manually, than it must be set to `False` since it will be overwritten otherwise (*false*, *true*, *default=true*)

std_flux

Object Properties

- **execute** (*bool*) – Must be set `True` if this step should be executed (*false*, *true*, *default=true*)
- **create_sof** (*bool*) – Must be set `True` if one wants to create a new *.sof* file. In case the user wants to change or create the *.sof* file manually, than it must be set to `False` since it will be overwritten otherwise (*false*, *true*, *default=true*)

sky

Object Properties

- **execute** (*bool*) – Must be set `True` if this step should be executed (*false*, *true*, *default=true*)
- **modified** (*bool*) – If set `True` the modified sky subtraction will be executed. This method will prevent the over subtraction of emission lines that are both emitted from the Earth's atmosphere (tellurics) and the target (e.g., HII regions). For a detailed description of this method we refer to Zeidler et al. 2019. If `modified` = `True` the `method` should be *subtract_model*. Other methods will **not** lead to an error but they may lead to wrong results. (*false*, *true*, *default=false*)
- **sky_field** (*string*) – Determines if a sky observation (if available) or the science observation itself is used to determine the background contamination (tellurics). If set to *auto* the pipeline will check if there are sky observations available and use the closest one in time to the science exposures. If the `skyfield` = *object*, the science exposure will be used. (*'auto'*, *'object'*, *default='auto'*)

- **fraction** (*float*) – The fraction of the image used to be considered as sky. Must be between 0. and 1. For more details we refer to the MUSE data reduction pipeline handbook. (*[0., 1.[*, *default=0.05*)
- **ignore** (*float*) – The fraction of the image ignored for sky consideration. Must be between 0. and 1. For more details we refer to the MUSE data reduction pipeline handbook. (*[0., 1.[*, *default=0.05*)
- **method** (*string*) – The method how the determined sky spectrum is fitted to each spaxel to subtract the sky background. Should be set to 'subtract-model' in case of `modified = True`. For more details we refer to the MUSE data reduction pipeline handbook. (*'model'*, *'subtract-model'*, *'simple'*, *'none'*, *default='model'*)
- **create_sof** (*bool*) – Must be set `True` if one wants to create a new *.sof* file. In case the user wants to change or create the *.sof* file manually, than it must be set to `False` since it will be overwritten otherwise (*false*, *true*, *default=true*)

sci_post

Object Properties

- **execute** (*bool*) – Must be set `True` if this step should be executed (*false*, *true*, *default=true*)
- **autocalib** (*string*) – This may execute the flux autocalibration between the different IFUs for empty fields. If `user` a AUTOCAL_FACTORS.fits table has to be provided by the user. (*'none'*, *'deepfield'*, *'user'*, *default=none*)
- **subtract_sky** (*bool*) – The user can decide to not run any sky subtraction. All telluric lines will be included in the final datacube. this may be useful if one wants to do their own wavelength calibration. (*false*, *true*, *default=true*)
- **raman** (*bool*) – If laser guide stars are used raman scattering in the atmosphere may be visible in the final data cubes. If set to `True` the Raman lines are removed. For more details we refer to the MUSE data reduction pipeline handbook. (*false*, *true*, *default=false*)
- **create_sof** (*bool*) – Must be set `True` if one wants to create a new *.sof* file. In case the user wants to change or create the *.sof* file manually it must be set to `False` since it will be overwritten otherwise (*false*, *true*, *default=true*)

dither_collect

Object Properties

- **execute** (*bool*) – Must be set `True` if this step should be executed (*false*, *true*, *default=true*)
- **user_list** (*list*) – A user list with the identifiers of the dither positions, in case only a subset shall be collected and copied to the final folder to be used for the combination. If left empty all dither positions are used to create the final data cube. (*default=[]*)

exp_align

Object Properties

- **execute** (*bool*) – Must be set `True` if this step should be executed (*false*, *true*, *default=true*)
- **create_sof** (*bool*) – Must be set `True` if one wants to create a new *.sof* file. In case the user wants to change or create the *.sof* file manually, than it must be set to `False` since it will be overwritten otherwise (*false*, *true*, *default=true*)

exp_combine

Object Properties

- **execute** (*bool*) – Must be set True if this step should be executed (*false*, *true*, *default=true*)
- **weight** (*string*) – The method how the fluxes in each dither position sre weighted. For more details we refer to the MUSE data reduction pipeline handbook. (*'exptime'*, *'fwhm'*, *'none'*, *'header'*, *default=exptime*)
- **create_sof** (*bool*) – Must be set True if one wants to create a new *.sof* file. In case the user wants to change or create the *.sof* file manually, than it must be set to False since it will be overwritten otherwise (*false*, *true*, *default=true*)

1.2.3 Utility methods

The following methods are routines to support the *MUSEreduce.musereduce*, needed to execute the various data reduction steps as it is suggested in the MUSE data reduction pipeline. They are documented here for completness.

MUSEreduce.**_get_filelist**(*self*, *data_dir*, *filename_wildcard*)

This module collects the necessary file lists from folders.

Args:

data_dir [str] The directory where the files are located.

filename_wildcard [str] The filenames which should be collected

MUSEreduce.**_call_esorex**(*self*, *exec_dir*, *esorex_cmd*)

This module calls the various ESOrex commands and gives it to the terminal for execution.

Args:

exec_dir [str] The directory where ESOrex should be executed.

esorex_cmd [str] The ESOrex command that needs to be executed

MUSEreduce.**_sort_data**(*self*)

This module reads the headers of all of the raw data and sorts it accordingly. It also assigns the correct calibration files to the exposures

MUSEreduce.**_bias**(*self*, *exp_list_SCI*, *exp_list_DAR*, *exp_list_TWI*, *create_sof*)

This module calls *ESORex'* muse_bias

Args:

exp_list_SCI [list] The list of all associated science files including their category read from the fits header

exp_list_DAR [list:] The list of all associated dark files including their category read from the fits header

exp_list_TWI [list:] The list of all associated twilight files including their category read from the fits header

create_sof [bool:] True: bias.sof is created and populated

False: bias.sof is not created and needs to be provided by the user

MUSEreduce.**_dark**(*self*, *exp_list_SCI*, *exp_list_DAR*, *create_sof*)

This module calls *ESORex'* muse_dark

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

exp_list_DAR [`list`:] The list of all associated dark files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_flat`**(*self*, *exp_list_SCI*, *exp_list_TWI*, *create_sof*)
This module calls *ESORex'* `muse_flat`

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

exp_list_TWI [`list`:] The list of all associated twilight files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_wavecal`**(*self*, *exp_list_SCI*, *exp_list_TWI*, *create_sof*)
This module calls *ESORex'* `muse_wavecal`

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

exp_list_TWI [`list`:] The list of all associated twilight files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_lsf`**(*self*, *exp_list_SCI*, *exp_list_TWI*, *create_sof*)
This module calls *ESORex'* `muse_lsf`

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

exp_list_TWI [`list`:] The list of all associated twilight files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_twilight`**(*self*, *exp_list_SCI*, *exp_list_TWI*, *create_sof*)
This module calls *ESORex'* `muse_twilight`

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

exp_list_TWI [`list`:] The list of all associated twilight files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_science_pre`**(*self*, *exp_list_SCI*, *create_sof*)

This module calls *ESORex'* `muse_scibasic`

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_std_flux`**(*self*, *exp_list_SCI*, *create_sof*)

This module calls *ESORex'* `muse_standard`

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_sky`**(*self*, *exp_list_SCI*, *create_sof*)

This module calls *ESORex'* `muse_sky`

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_modified_sky`**(*self*, *exp_list_SCI*, *create_sof*)

This module calls *ESORex'* `muse_sky` with the modified continuum and line subtraction as described in Zeidler et al. 2019.

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

`MUSEreduce.`**`_scipost`**(*self*, *exp_list_SCI*, *create_sof*, *OB*)

This module calls *ESORex'* `muse_scipost`

Args:

exp_list_SCI [`list`] The list of all associated science files including their category read from the fits header

create_sof [`bool`:] `True`: `bias.sof` is created and populated

`False`: `bias.sof` is not created and needs to be provided by the user

OB [`str`:] The specific `OB` to be reduced.

MUSEreduce.**_dither_collect**(*self*, *exp_list_SCI*, *OB*)

This module collects the individual dither exposures for one OB to be combined in the steps: *MUSEreduce._exp_align* and *MUSEreduce._exp_combine*

Args:

exp_list_SCI [list] The list of all associated science files including their category read from the fits header

create_sof [bool:] True: `bias.sof` is created and populated

False: `bias.sof` is not created and needs to be provided by the user

OB [str:] The specific `OB` to be reduced.

MUSEreduce.**_exp_align**(*self*, *exp_list_SCI*, *create_sof*, *OB*)

This module calls *ESORex'* `muse_exp_align`

Args:

exp_list_SCI [list] The list of all associated science files including their category read from the fits header

create_sof [bool:] True: `bias.sof` is created and populated

False: `bias.sof` is not created and needs to be provided by the user

OB [str:] The specific `OB` to be reduced.

MUSEreduce.**_exp_combine**(*self*, *exp_list_SCI*, *create_sof*)

This module calls *ESORex'* `muse_exp_combine`

Args:

exp_list_SCI [list] The list of all associated science files including their category read from the fits header

create_sof [bool:] True: `bias.sof` is created and populated

False: `bias.sof` is not created and needs to be provided by the user

1.2.4 History

New in version 0.1.0: Executes the MUSE reduction pileline in the correct order

New in version 0.1.1: introducing only one calibration file folder per OB

New in version 0.1.2: choosing the illumination file closest to the observation

New in version 0.1.3: selecting the files for the different master file creations

New in version 0.1.4: minor corrections

New in version 0.1.5: looping order changed. each module loops by itself

New in version 0.1.6: always checking if calibration files already exist

New in version 0.1.7: Choosing between ESO calibrations and own calibrations

New in version 0.1.8: User can choose specific exposure time to reduce

New in version 0.2.0: exposures spread via different OBs for one pointing is supported. To do so: run the script normally for each OB including `muse_scipost`. After all are processed then run `muse_exp_align` and `muse_exp_combine`. AO observations are supported. To reduce multiple OBs in one run, set rootpath to *manual*.

New in version 0.2.1: user can choose the number of CPUs used

New in version 0.2.2: sky subtraction can be modified, so that individual elements can be excluded

New in version 0.2.3: use *json* file as input

New in version 0.2.4: additional parameters added: *skyreject* and *skysubtraction*. The set parameters and modules are shown in the output

New in version 0.2.5: bug fixes

New in version 0.3.0: using the correct *ILLUM* file for the *STD* reduction in the `muse_sci_basic` routine, `muse_sci_basic` now separated for *STD* and *OBJECT* reduction.

New in version 0.3.1: one can select if the sof file are created automatically or provided by the user

New in version 0.4.0: supports now *pipeline v2.4.2* and the *NFM-AO* added: pipeline_path choosing if darks may be used only reduces *STD* once per OB general use of external *SKY* fields collecting the files for `muse_exp_combine` in an independent step `muse_exp_align` is an independent step now

New in version 0.4.1: new file names to correct a problem where data gets replaced in the `muse_scipost` routine if you reduce the data with and without sky

New in version 0.4.2: one can now change the ignore and fraction parameters in the *json* file

New in version 0.4.3: one can auto remove and rewrite the statics

New in version 0.4.4: changed the sky subtraction keyword the user can give now individual names for the different dither exposures: Does currently not work with multiple OBs or multiple pointings per OB

New in version 0.5.0: rewriting `MUSEreduce.musereduce` to a class and pep-8 style. *DEBUG* keyword added. Wrapper can be executed without running `esorex` but needs to be used with already existing reduced data.

New in version 0.5.1: added *skymethod*

New in version 0.5.2: `MUSEreduce.musereduce` can now handle if the exposures for one pointing are distributed via multiple OBs with multiple exposures in each OB.

New in version 1.0: The release version as originally published in Zeidler et al. 2019.

New in version 1.0.2: It is possible to omit the data reduction of the standard star in `muse_scibasic` by setting the *execute_std* to `False`. More skylines can be added in the `muse_scibasic` module but adding additional wavelengths in Angstrom to *skylines*.

New in version 1.1.0: `MUSEreduce.musereduce` supports now *pipeline v2.8.1*. A legacy version for *pipeline v2.4.2* was created in a separate release. The keywords *autocalib* and in `sci_oost` was added. The naming convention in `dither_collect` was changed so that it matches the convention **RADEC_EXPTIME_ROTANGLE_COUNTER**.

New in version 1.1.1: The key words *srcmin* and *srcmax* were added to *config.json* to be used in `exp_align`.

1.3 Spectral line fitting and radial velocity measurements

The main purpose of MUSEreduce is the spectral line fitting in order to create template spectra from observations, which are used to measure the radial velocities of stars and gas, especially in the absence of working spectral template libraries, e.g., for pre-main-sequence stars in the optical.

The `radial_velocities.RV_spectrum` class and the spectral fitting module `line_fitter` are the heart of this package using pyspeckit as basic spectral fitting routine and ppxf for the spectral cross-correlation. Together with a Monte Carlo bootstrap technique, it is possible to measure radial velocities with an accuracy of $1-3\,\mathrm{km\,s^{-1}}$ in the absence of any template library. For a detailed description on how this technique works, we refer to Zeidler et al. 2019.

1.3.1 Spectral line fitter

This module is the work horse for fitting spectral lines from an input catalog to a given 1D spectrum. The line fitting itself is performed using the module pyspeckit.

pyspeckit uses an iterative method to fit the spectral lines and the continuum simultaneously. *line_fitter* automatically adjusts the input parameters to pyspeckit in the manner to optimize the fitting (*continuum order* and *wavelegth limits*)

The *line_fitter* determines whether the fit fullfils the set parameters and was succesfull. The *line_fitter* also allows to fit blends by fixing a maximum ratio between the primary line (the one the user is interested in) and the secondary line (blend). This ensures that the blend does not become the dominant line.

In order to accomodate hyper-velocity stars an option is given that the wavelength limits are automatically adjust for each iteration n based on the solution $n-1$ of the primary line taking into account $\Delta\lambda/\lambda$.

If the spectra with high radial velocities, or if the object is red shifted an initial guess of the radial velocity needs to be provided. This may be done by the *Kwargs rv_sys* of the *radial_velocities.RV_spectrum*. This is important if the shift is much larger than the wavelength limits, where the automatic adjustment fails. Additionally, the more accurate this *rv_sys* is provided the faster the fit converges.

`line_fitter.`**`line_fitter`**(*self*, *linecat*, *line_idx*, *niter*, *input_resid_level*, *max_contorder*, *max_ladjust*, *adjust_preference*, *input_continuum_deviation*, *llimits*, *max_exclusion_level*, *blends*, *autoadjust*, *fwhm_block*)

This is module fits the spectral lines using pyspeckit. It automatically determines the goodness of fit and decides the best solution for the line and continuum fit. It handles blended lines in a way a maximum lines ratio can not be exceeded to not make the weaker blend the dominant lines. Additionally, the limits in wavelength range in iteration n can be adjusted automatically based on iteration n to accomodate for larger wavelength shifts.

In this way it is possible to fit spectral lines using a line catalog as only input.

Args:

linecat [`numpy.array()`] Array with the spectral lines and their wavelengths

line_idx [`str`] Name of the primary line

niter [`int`] Number of iterations

input_resid_level [`float`] The maximum MAD for the fit residuals for a succesfull fit

max_contorder [`int`] The maximum polynominal order of the continuum to have

max_ladjust [`str`] The maximum number of wavelength range adjustments in steps of 5 Angstrom

adjust_preference [`str`] contorder: continuum order is adjusted first

wavelength: wavelength range is adjusted first

input_continuum_deviation [`float`] by how much the continuum is allowed to deviate from a running median estimate. This is set to prevent lines mimicking a continuum

llimits [`list`] the limits for the wavelength fit as set in `ppxf`

max_exclusion_level [`float`] The exclusion level for lines to be excluded from the next baseline estimate as set in `pyspeckit`

blends [`ascii`-file or `None`] A file with primary lines that contain blends to provide a maximum amplitude ratio of the primary and the blend to prevent that the blend becomes the dominant line in the fit.

autoadjust [`bool`] `True`: the wavelength limits `llimit` will be adjusted to the fit of the previous iteration. All other wavelength range are adjusted accordingly taking into account the proper velocity corrected shift $\Delta\lambda/\lambda$. This is especially important to detect hyper-velocity stars.

False: no adjustment to the limits done

fwhm_block [bool:obj:] True: The minimum fwhm of the voigt profiles of the fitted lines is the instrument's dispersion

False: The minimum fwhm of the voigt profiles of the fitted lines is zero

Returns:

line_idx [str] Name of the primary line

temp_l [float] fitted wavelength of the primary line

temp_a [float] fitted amplitude of the primary line

temp_sl [float] fitted Lorentzian gamma of the primary line

temp_sg [float] fitted Gaussian sigma of the primary line

spec_select_idx [numpy.array()] indices of the used part of the spectrum

template_f [numpy.array()] template spectrum shifted to the rest-frame

continuum [numpy.array()] the fitted continuum

lstart [float] first used wavelength bin (might differ from input if it was adjusted during fitting)

lend [float] last used wavelength bin (might differ from input if it was adjusted during fitting)

contorder [int] The order of the polynominal used for the continuum

fit_f [array] the fitted spectrum

significance [float] the line strength over the continuum

fit_failed [bool] True: if the fit failed for some reason. This line will be excluded from further analyses

fit_f_highres [float] the fitted spectrum and the oversampling resolution

spec_select_idx_highres [numpy.array()] indices of the used part of the over sampled spectrum

template_f_highres [numpy.array()] over sampled template spectrum shifted to the rest-frame

continuum_highres [numpy.array()] the fitted over sampled continuum

Warning: The automatic handling of absorption and emission lines implemented in vers. 0.1.2 has not been tested yet.

1.3.2 Radial Velocities

This is the main class for measuring the radial velocities. A basic example is descibed in *Examples*. A detailed description can be found in Zeidler et al. 2019.

Throughout the document the **primary line** is the spectral line of interest for which the line fitting will be executed. **Secondary lines** are spectral lines in blends.

A detailed demonstration on how to use the radial velocity fitter is provided in *Examples* including template files.

The radial velocity fitter

class radial_velocities.**RV_spectrum**(*spec_id*, *spec_f*, *spec_err*, *spec_lambda*, *loglevel='INFO'*, *templatebins=100000*, *specbinsize=1.25*, *dispersion=2.4*, *linetype='absorption'*, *rv_sys=0.0*)

This class contains the 1D spectrum, as well as all fitting parameters and output values including the radial velocity catalog.

Args:

spec_id [str] unique identifier of the spectrum

spec_f [numpy.array()] flux array of the spectrum in erg/s/cm:math:^2/Angstrom

spec_err [numpy.array()] flux uncertainty array of the spectrum

spec_lambda [numpy.array()] wavelength_array of the spectrum in Angstrom

Kwargs:

loglevel [str (optional, default: INFO)] DEBUG: all functions run on a single core to obtain proper output in the correct order for proper debugging.

templatebins [float (optional, default: 100000)] Number of wavelength bins for the oversampled spectrum used to fit the spectral lines.

specbinsize [float (optional, default: 1.25)] The spectral bin size. The default is set to fit the MUSE dataset

dispersion [float (optional, default: 2.4)] The dispersion of the spectrograph. The default is set to the nominal MUSE instrument dispersion

linetype: str (optional, default: absorption) absorption: All spectral lines are absorption lines

emission: All spectral lines are emission lines

both : spectral lines can be absorption or emission lines. **This mode has not been tested yet !!!**

rv_sys [float (optional, default: 0)] systematic RV shift in km/s Should be provided for large velocity offsets or redshifted objects

catalog(*initcat=None*, *save=False*, *load=None*, *printcat=False*)

This module handles the catalog that holds the fit results of the primary lines.

Kwargs:

initcat [ascii (optional, default: None)] ascii file containing the primary lines format: name, lambda, start, end

save [bool (optional, default: False)] True: The catalog is written to a file.

load [str (optional, default: None)] Loads a catalog from catalog file.

printcat [bool (optional, default: False)] True: The catalog is printed to the terminal

clean()

This modules resets all of the output. This **must** be executed before repeating the spectral fit without re-initiating the class.

line_fitting(*input_cat*, *line_idxs*, *niter=5*, *n_CPU=-1*, *resid_level=None*, *max_contorder=2*, *max_ladjust=4*, *adjust_preference='contorder'*, *input_continuum_deviation=0.05*, *llimits=[-2.0, 2.0]*, *max_exclusion_level=0.3*, *blends=None*, *autoadjust=False*, *fwhm_block=False*)

Initializing the line fitting by calling *line_fitter*

Args:

input_cat: `numpy.array()` input spectral line catalog with wavelengths in Angstrom

line_idxs: `numpy.array()` `str` `numpy.array()`: of the line names for which RV fits should be performed, must be identical to "name" in init_cat

Kwargs:

niter [`int` (optional, default: 5)] The maximum number of iteration if convergence is not reached

n_CPU [`float` (optional, default: -1)] Setting the number of CPUs used for the parallelization. If set to -1 all available system resources are used. Maximum number of CPUs is the number of spectral lines the fit is performed on.

resid_level: `float` or `None` (optional, default: None) The maximum MAD for the fit residuals for a succesfull fit

max_contorder [`int` (optional, default: 2)] The maximum polynominal order of the continuum

max_ladjust [`int` (optional, default: 4)] Ahe maximum number of wavelength range adjustments in steps of 5 Angstrom

adjust_preference: `str` (optional, default: contorder) `contorder`: continuum order is adjusted first

`wavelength`: wavelength range is adjusted first

input_continuum_deviation `float` (optional, default: 0.05) Fraction by how much the continuum is allowed to deviate from a running median estimate. This is set to prevent lines mimicking a continuum

llimits: `list` (optional, default: [-2., 2.]) the limits for the wavelength fit as set in ppxf

max_exclusion_level `float` (optional, default: 0.3) The exclusion level for lines to be excluded from the next baseline estimate as set in pyspeckit

blends: `ascii` or `None` (optional, default: None) A file with primary lines that contain blends to provide a maximum amplitude ratio of the primary and the blend to prevent that the blend becomes the dominant line in the fit.

autoadjust `bool` (optional, default: `False`) `True`: the wavelength limits `llimit` will be adjusted to the fit of the previous iteration. All other wavelength range are adjusted accordingly taking into account the proper velocity corrected shift $\Delta\lambda/\lambda$. This is especially important to detect hyper-velocity stars.

fwhm_block `bool` (optional, default: `False`) `True`: The minimum fwhm of the voigt profiles of the fitted lines ais the instrument's dispersion. This prevents unphysical lines.

`False`: The minimum fwhm of the voigt profiles of the fitted lines is zero.

`plot`(*oversampled=False*)

Plots the spectrum and the regions of the primary lines to a file including the spectral fit and the template.

Kwargs:

oversampled [`bool` (optional, default: `False`)] `True`: The oversampled spectral fit and the template are used in the plot.

`rv_fit`(*guesses, niter=10000, line_sigma=3, n_CPU=-1, line_significants=5, RV_guess_var=0.0*)

The module runs the radial velocity fit using ppxf and the *Monte Carlo radial velocity determination*.

Args:

guesses [`numpy.array()`] The initial guesses for the the radial velocity fit guesses in the form [RV,sepctral_dispersion]

Kwargs:

niter [`int` (optional, default: 10000)] number of iterations to bootstrap the spectrum

line_sigma: `int` (optional, default: 3): sigma for the RV clipping for the individual lines

n_CPU [`float` (optional, default: -1)] Setting the number of CPUs used for the parallelization. If set to -1 all available system resources are used. Maximum number of CPUs is the number of spectral lines the fit is performed to.

line_significants: `int` (optional, default: 5) The sigma-level for the spectral line to be above the continuum in order to be considered *valid*

RV_guess_var [`float` (optional, default: 0)] The maximum variation the RV guess will be varied using a uniform distribution.

`rv_fit_peak`(*line_sigma=3*, *line_significants=5*)

This module determines the radial velocity solely on the fitted peaks of the spectral lines

Kwargs:

line_sigma [`int` (optional, default: 3)] The sigma-level for the sigma clipping before determining peak radial velocity measurement

line_significants: `int` (optional, default: 5) The sigma-level for the spectral line to be above the continuum in order to be considered *valid*

Monte Carlo radial velocity determination

This is the module that performs the Monte Carlo boot strapping to measure radial velocities. It is automatically called by the *`radial_velocities.RV_spectrum.rv_fit`* attribute of *`radial_velocities.RV_spectrum`* class and there is no need to repeat this step manually. Nevertheless, we show this part of the code since it may be useful for other applications to measure RVs.

`ppxf_MC.`**`ppxf_MC`**(*log_template_f*, *log_spec_f*, *log_spec_err*, *velscale*, *guesses*, *nrand=100*, *degree=4*, *goodpixels=None*, *moments=4*, *vsyst=0*, *sigma=5*, *spec_id=None*, *RV_guess_var=0.0*, *n_CPU=-1*)

This module runs the Monte Carlo ppxf runs, which is needed for the RV measurements. Most of the input parameters are similar to the standard ppxf parameters (see Cappellari and Emsellem 2004) for a more detailed explanation).

Args:

log_template_f [`numpy.array()`] The logarithmically binned template spectrum

log_spec_f [`numpy.array()`] The logarithmically binned source spectrum

log_spec_err [`numpy.array()`] The logarithmically source spectrum uncertainties

velscale: `float` The velocity scale of the source spectrum

guesses [`numpy.array()`] The initial guesses for the the radial velocity fit guesses in the form [RV,sepctral_dispersion]

Kwargs:

nrand [`int` (optional, default: 5)] The maximum number of iteration if convergence is not reached

degree [`int` (optional, default: 4)] The degree of the additive polynomial to fit offsets in the continuum. A low order polynominal may be needed to get better results

goodpixels [`numpy.array()` (optional, default: 5)] A `numpy.array()`: of pixels that are used by ppxf to fit the template to the source spectrum

moments [`int` (optional, default: 4)] The moments to be fit (v, sigma, h3, h4)

vsyst [`float` (optional, default: 0.)] A systematic velocity. This may be needed if the system move at high velocities compared to the rest frame. If the guess of vsyst is good, the fit runs faster and more stable.

sigma [`int` (optional, default: 5)] The sigma used to clip outliers in the histogram determination.

spec_id [`str` (optional, default: None)] An ID number of the source spectrum. This becomes handy when fitting many individual sources because the output files will be named with the ID.

RV_guess_var [`float` (optional, default: 0)] The maximum variation the RV guess will be varied using a uniform distribution.

n_CPU [`int` (optional, default: -1)] The number of cores used for the Monte Carlo velocity fitting. If n_CPU=-1 than all available cores will be used.

1.3.3 History

New in version 0.1.0: working radial velocity fitting package

New in version 0.1.0: no RV calculation for lines that are below the significance level

New in version 0.1.0: instrument dispersion added as keyword

New in version 0.1.1: moved to pep-8

New in version 0.1.2: now handles absorption and emission lines. **Not tested yet, though**

New in version 0.1.3: During the boot strap procedure, the initial guesses of the velocities are varied between certain limits, to ensure more stability to the fit in the case of an "unlucky choice of initial parameters. Added Kwarg to `RV_spectrum.rv_fit` is *RV_guess_var*, which is by default 0. It describes the min/max variation of the initial RV guess for each fit.

New in version 1.0: The release version as originally published in Zeidler et al. 2019.

New in version 1.1: Introducing the *Kwargs rv_sys*, which should be used for large initial RV offsets from rest frame or for redshifted spectra. *rv_sys* should provide an estimate of that shift.

1.4 Cubes

`cubes` is a module that contains a collection of support functions for the analyses of data cubes, specifically MUSE data cubes.

Todo: A more detailed description will follow soon.

`cubes.`**`linemaps`**(*input_fits*, *path=None*, *elements=None*, *wavelengths=None*)

This module is intended to create linemaps of specified lines/elements

Args:

input_fits [`str`] The fully reduced datacube Pampelmuse has been run on

Kwargs:

path [`str` (optional, default: current directory)] I/O path

element [obj:*list* (optional)] list of elements the linemaps shall be produced

wavelength [`list` (optional)] list of wavelength for givene elements, optional, must be given if *elements* is given

`cubes.`**`mosaics`**(*input_list*, *name*, *path=None*)

This module is intended to create mosaics of specified lines/elements. linemaps should have been created beforehand using the `linemaps` module

Args:

input_list [`list`] The list of specific linemaps to be used to mosaic

name [`str`] Name of the created mosaic

Kwargs:

path: `str` (optional, default: current directory) I/O path

`cubes.`**`pampelmuse_cat`**(*ra*, *dec*, *mag*, *filter*, *idx=None*, *path=None*, *sat=0.0*, *mag_sat=None*, *ifs_sat=None*, *mag_limit=None*, *regsize=0.5*)

This modules uses input parameters to create a catalog that is compatible with pampelmuse

Args:

ra [`float`] RA coordinates of the stars

dec [`float`] Dec coordinates of the stars

mag [`float`] magnitudes of the stars

filter [`str`] filter used for the magnitudes

Kwargs:

idx [`float` (optional, default][counting up from 1)] catalog index of the stars

sat [:obj‘float‘ (optional, default: 0)] value assigned to saturated sources in the catalog

mag_sat [`float` (optional, default: `None`)] magnitude that replaces saturated sources in the catalog

path [`str` (optional, default: current directory)] path of the output file

mag_limit [`float` (optional, default: `None`)] the magnitude at which the output catalog should be truncated

regsize [`float` (optional, default: :float:0.5)] the size of the regions in arcsec

`cubes.`**`wcs_cor`**(*input_fits*, *offset_input*, *path=None*, *offset_path=None*, *output_file=None*, *out_frame=None*, *in_frame=None*, *correct_flux=False*, *spec_folder='stars'*, *spec_path=None*, *correctiontype='shift'*)

Args:

inputfits [`str`] The fully reduced datacube, whose WCS has to be corrected

offset_input [`str`] The prm file produced by Pampelmuse or the OFFSET_LIST.fits file from the `exp_align` routine

Kwargs:

path [`str` (optional, Default: current directory)] I/O path

offset_path [`str` (optional, default: current directory)] I/O path of prm file

output_file [`str` (optional, default: input file name +_cor)] outputfile name

output_frame [`str` (optional, default][input frame)] coordinate frame of the output cube in case one want to change. Is always set to default : input frame if corrections are not based on a .prm file

in_frame [`str` (optional, default: input frame)] coordinate frame of the output cube in case it cannot be determined from the header information or it has to be manually changed

correct_flux [`bool` (optional, default: `False`)] If set `True` the fluxes of the data cube will be corrected to match the input catalog to correct for calibration offsets. If the input file is a prm-file: This step is only recommended if the input fluxes can be trusted. CUBEFIT and GETSPECTRA have to executed again using the corrected data cube to correct the prm file and to extract the corrected spectra. If the input file is a ESO OFFSET-LIST.fits file: The fluxes will be scaled based on the 'FLUX_SCALE' entries.

spec_folder [`str` (optional, default: `spectra`)] The folder name, in which the the extracted stellar spectra are stored. This is only needed if correct_flux=:obj:*True* and the corrections are based on a .prm file

spec_path [`str` (optional, default: current directory)] I/O path of the `spec_folder`. This keyword is only need if the corrections are based on a .prm file

correctiontype [`str` (optional, default: `shift`)] the type of distortion correction

`full`: the full 2D CD matrix, only possible if based on a .prm file

`shift`: shift in XY only.

1.4.1 History

New in version 0.1.0: The functions *`cubes.wcs_cor()`*, *`cubes.pampelmuse_cat()`*, *`cubes.linemaps()`*, and *`cubes.mosaics()`* were added.

New in version 0.1.0: *`cubes.wcs_cor()`* also can be used with *the OFFSET_LIST.fits* file produced by ESORex

1.5 Utility module

This modules contains a variety of functions to support *MUSEpack*. Many of these functions may be useful for other purposes.

Todo: The documentation of all of the utility modules will follow soon

`utils.`**`initial_guesses`**(*self, lines, blends=None, linestrength=100.0, llimits=[-2.0, 2.0]*)

Creates the initial guesses for the line fitter

Args:

lines [`numpy.array()`] central wavelengths of the spectral lines

Kwargs:

linestrength [`float` (default: 100)] initial guess for the line strength

blends [`str` (optional)] A file containing the a list of blended lines in the format: ** List is coming soon**

return:

guesses [`list`] lists containing the guesses

limits [`list`] lists containing the limits

limited [`list`] lists containing the limited

`utils.`**`update_parinfo`**(*self*, *guesses*, *llimits*, *line_idx*, *blends*, *parinfo*, *autoadjust*, *fwhm_block*)
Updates the parinfo file, created by pyspeckit.

Args:

guesses [`numpy.array()`] The initial guesses for the the radial velocity fit guesses in the form [RV,sepctral_dispersion]

llimits [`list`] the limits for the wavelength fit as set in `ppxf`

line_idx [`str`] Name of the primary line

blends [`ascii`-file or `None`]

A file with primary lines that contain blends to provide a maximum amplitude ratio of the primary and the blend to prevent that the blend becomes the dominant line in the fit.

parinfo: **`dict`** the parinfo file created by pyspeckit, which contains the fitted parameters for all input lines

autoadjust [`bool`] `True`: the wavelength limits `llimit` will be adjusted to the fit of the previous iteration. All other wavelength range are adjusted accordingly taking into account the proper velocity corrected shift $\Delta\lambda/\lambda$. This is especially important to detect hyper-velocity stars.

`False`: no adjustment to the limits done

fwhm_block [`bool:obj:`] `True`: The minimum fwhm of the voigt profiles of the fitted lines is the instrument's dispersion

`False`: The minimum fwhm of the voigt profiles of the fitted lines is zero

1.5.1 History

New in version 0.1.0: module created

New in version 0.1.1: moved to pep-8

New in version 0.1.2: now handles absorption and emission lines emission not tested yet, though

New in version 0.1.3: The `util.Line_clipping` was adjusted in how the two outliers are clipped before the MAD is calculated. It now keeps the $N-2$-lines that have the smaller deviation from each other.

New in version 0.1.4: adding *rv_sys*, to compensate for larger systematic RV shifts or redshifts for the line and RV fitter. The `util.lambda_rv_shift` was introduced

1.6 Examples

1.6.1 Measuring stellar radial velocities

In this tutorial we show the user how to create a spectral measure the stellar radial velocity. All files in including the `example` code may be downloaded by using the individual links or as a zip-archive `here`. The `stellar spectrum` used in this example contains the PampelMuse extracted spectrum of a member pre-main-sequence member star of the young massive star cluster Westerlund 2 showing a pronounced CaII-Triplet, which we will use to measure the RV. The user can also see that the Balmer lines are not well extracted, which is caused by the high nebular emission lines (Zeidler et al. 2018, Zeidler et al. 2019) demonstrating that this does not influence the radial velocity fit.

The procedure

1. Reading in the necessary data

   As a first step we must read the necessary data. This includes the spectrum, the `spectral line library`, and the `line list` of the spectral lines that are used to measure the radial velocity. Since the CaII-Triplet lines are blended with the Paschen Series, we also have to include a catalog of the `blends`.

```python
from astropy.io import ascii
from astropy.io import fits
import numpy as np
from MUSEpack.radial_velocities import RV_spectrum

spec_hdu = fits.open('star.fits')
lcat = ascii.read('line_library.dat')['wavelength']
blends = 'blends.dat'
target_lines = 'target_lines.list'
```

2. Creating the flux, flux uncertainty, and wavelength arrays, as well as the line lists In this step all the necessary arrays are created from the input files. The header information of the stellar spectrum is used to create the wavelength array. The flux is stored in the first extension while the flux uncertainties are stored in the second extension of the `stellar spectrum`.

   Note: The MUSE fluxes are stored in $10^{20}\,\mathrm{erg\,s^{-1}\,cm^{-2}\,A^{-1}}$. Therefore, we need to multiply the flux and uncertainty array with 10^{-20}

```python
spec_head = spec_hdu[0].header
spec_wave = np.arange(spec_head['NAXIS1']) * spec_head['CDELT1'] + spec_head['CRVAL1']
spec_data = spec_hdu[0].data * 10e-20
spec_error = spec_hdu[1].data * 10e-20
```

3. We are now initiating the spectral instance. The ID of the star will be *star_1*. We initiate the instance in INFO mode.

```python
spfit = RV_spectrum('star_1', spec_data, spec_error, spec_wave, loglevel='INFO')
```

4. As next step we need to initiate the catalog populated with the absorption lines, with which we want to measure the radial velocity. This catalog will contain all the fitted parameters and may be called at any point calling `spfit.cat`

```python
spfit.catalog(initcat=target_lines)
```

At this point the spectral instance is fully initiated and we can start working with it.

5. We now run the spectral fit using all available cores. The line indices may be read directly from the instance's catalog. A log file named *star_1.log* will be automatically created

```python
spfit.line_fitting(lcat, spfit.cat.index, resid_level=0.05,\
max_contorder=2, niter=10, adjust_preference='contorder', n_CPU=-1,\
max_ladjust=2, max_exclusion_level=0.3, blends=blends,\
llimits=[-2., 2.], autoadjust=True, fwhm_block=True,\
input_continuum_deviation=0.05)
```

6. We will now plot the fitted spectrum to a file called *star_1_plot.png*. The fitted spectrum will be oversampled.

```
spfit.plot(oversampled=True)
```

At this point the spectral instance contains the spectrum as well as the fitted template. The next steps will execute the radial velocity determination.

7. First we determine the radial velocity purely based on the line peaks. The measured peak velocity and its uncertainty may be called at any time using `spfit.rv_peak` and `spfit.erv_peak`, respectively.

```
spfit.rv_fit_peak(line_sigma=3, line_significants=5)
```

8. Now we measure the radial velocities using the Monte Carlo bootstrap method. As initial guess we use the peak velocity `spfit.rv_peak`. We sample the spectrum 20000 times and use all available CPUs. The measured radial velocity and its uncertainty may be called at any time using `spfit.rv` and `spfit.erv`, respectively.

```
spfit.rv_fit([spfit.rv_peak, 0.], niter=20000, n_CPU=-1)
```

9. The measured radial velocities (per line and for the star) are added to the spectral instance. We may now write the fitted parameters to an ASCII file. Additionally we may print the radial velocity of the star measure with the peaks only and with the the full spectrum.

```
spfit.catalog(save=True, printcat=True)
```

The results

After successfully executing the script the radial velocity of *star_1* is $13.90 \pm 1.949\,\mathrm{km\,s^{-1}}$ based on two out of the three absorption lines. The radial velocity based on the peaks only is $15.43 \pm 1.144\,\mathrm{km\,s^{-1}}$ and agrees well with the cross correlation.

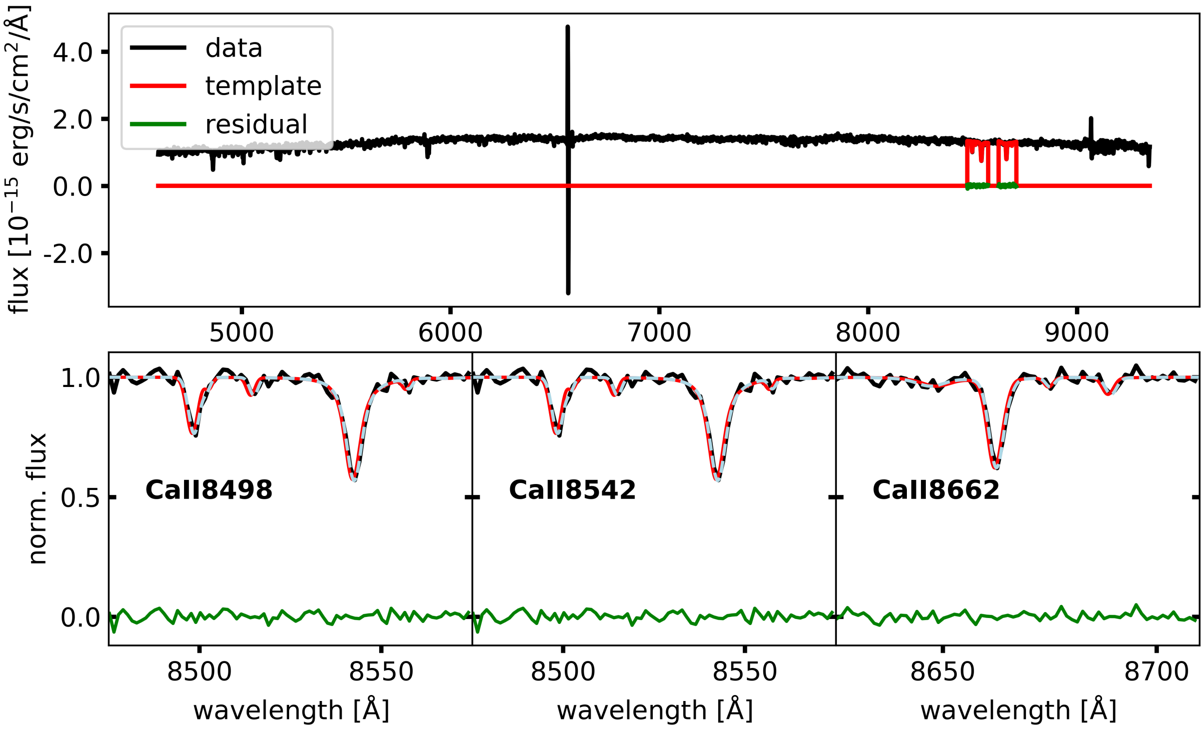
Note: The user should keep in mind that the radial velocity fit is a statistical Monte Carlo process so the final result may fluctuate slightly. All tests showed that these fluctuations are magnitudes smaller than the intrinsic uncertainty caused by a the limited spectral resolution and S/N.

The final results for each individual line, also documented in the *star_1.cat* file, are the following:

	l_lab	l_start	l_end	l_fit	a_fit	sg_fit	sl_fit	cont_order	RV	eRV	used	significance
CaII8498	8498.02	8475.0	8575.0	8498.528526	-3.17e-16	1.59	0.00	2	19.67	3.797		17.628082
CaII8542	8542.09	8475.0	8575.0	8542.529694	-1.14e-15	1.29	1.63	2	14.58	2.395	x	63.591358
CaII8662	8662.14	8625.0	8710.0	8662.552815	-5.64e-16	1.58	0.33	2	13.84	2.709	x	30.051289

The CaII8498 lines was discarded because it did not fulfill the 3σ criterion. The user may loosen the *line_sigma* parameter in the `radial_velocities.RV_spectrum.rv_fit` module to adjust the exclusion limits.

The following figure shows the line fit printed to *star_1_plot.png*



1.6.2 Additional tutorials

Todo: We will add more tutorials in the near future…

1.7 Credits

We ask the user of *MUSEpack* to cite the following paper(s) when using the software package: Zeidler et al. 2019.

We would like to thank B. James for her patience for being the first user of the *MUSEpack* and, especially, `MUSEreduce.musereduce`, which highly improved the user-friendliness of the package and tracing errors in the code

We thank N. Lützgendorf for helping us developing the Monte Carlo module `ppxf_MC`.

We thanks N. Miles for his continuous support and patience to creat the documentation.

1.8 Disclaimer

CHAPTER 2

References

- Zeidler, P., Nota, A., Sabbi, E., Luljak, P., McLeod, A. F., Grebel, E. K., Pasquali, A., Tosi, M. 2019, AJ, 158, 201
- Bacon, R., Accardo, M., Adjali, L., et al. 2010, Proc. SPIE, 7735, 773508

Python Module Index

Index

Symbols

C

E

I

J

L

M

P

R

U

W