AUSTRIA In-Kind Contribution: Data Reduction Software Project

DR03 Report: Architecture of Spectral Coaddition Recipe

12th November 2010

1 Introduction

This document reports on the architecture for the DR-03 sub-project. It is structured as follows: Sect. 2 contains a general description of the tasks and the requirements for the recipes. In Sect. 3, the algorithm is described, including a short summary of robust estimators of the mean and outlier rejection procedures. Finally, Sect. 4 describes the proposed implementation in the framework of a CPL recipe. The actual design of the algorithms was considered with priority for one-dimensional spectra.

2 Purpose

Spectra supply much of the basic information that astronomers use in examining celestial bodies. The goal of this sub-project is to improve the processing of spectra as outlined in the Statement of Work of the Austrian Data Reduction Project. In general, the project should apply to all ESO spectrographs covering the wavelength range from the optical to the infrared. The result will be CPL recipes for instrument independent coaddition or stacking procedures for one-dimensional spectra. The procedures must be able to handle the stacking of spectra obtained within the same observing block (OB) or from different OBs or different nights.

The observational data required for this project will be extracted from the ESO archive. For the moment, the science data used for tests and verification include one-dimensional slit spectra of point-like sources. These data are a set of UVES spectra obtained from the ESO Advanced Data Products archive.

This is a two steps procedure: In the first part we are about to build the most simple case as a prototype and then step by step we will implement additional features including re-gridding. As a first step, the global workflow defining the individual modules with the respective inputs and data products will be developed. The development will be based upon the Kepler scientific workflow tool[1] which is the framework for the Reflex-based ESO pipeline tool.

Currently, we are using as input data pipeline-calibrated one-dimensional spectra which are free of instrument signatures. We assume that the spectra are sampled on identical wavelengths grids and are

[1] `https://kepler-project.org`

flux calibrated so that no rebinning and no adjustment of sensitivity functions are required. The output data is the one-dimensional stacked spectrum.

3 Algorithm

Stacking strategies will be investigated for the following cases of overlapping spectra:

- Stacking of spectra originating from the same OB
- Stacking of spectra originating from different OBs (nights)
- Stacking of spectra with different spectral resolutions

In the case of different OBs (nights) the data sets may differ one from the other because of differences in seeing, night sky brightness, extinction due large differences in zenith distances, extinction due to variable photometric conditions, different integration times and inaccuracies of the flux calibration (instrumental response). Different stacking strategies will be investigated on basis of the observational parameters (relevant FITS keywords), available calibration data (night sky spectrum, instrument response function) and characteristics of the object.

3.1 Coaddition of spectra

The algorithm for co-adding will combine at each pixel position of the output spectrum, data values from all the input spectra, each one coming with an estimate of its variance. Robust estimators of the mean of a sample of data values and outlier rejection methods are described in the topical report *DR01 - Sky subtraction*. These methods can be used also here. For reference we summarize the algorithm and the relevant formulas:

Arithmetic mean: The samples are combined using the mean

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \,. \tag{1}$$

The variance can be also computed

$$\sigma_{\rm av}^2 = \frac{1}{N^2}\sum_{i=1}^{N} \sigma_i^2 \,. \tag{2}$$

The arithmetic mean is the most efficient estimator, but is also very sensitive to outliers.

Median: The median is computed as the 50th percentile of the ordered set of spectrum samples:

$$x_{\rm med} = x_{\frac{N+1}{2}} \quad \text{if } N \text{ is odd} \tag{3}$$

and

$$x_{\rm med} = \frac{1}{2}(x_{\frac{N}{2}} + x_{\frac{N}{2}+1}) \quad \text{if } N \text{ is even} \tag{4}$$

The sample variance is hard to estimate in this case so we use instead the asymptotic value for the ratio of the variances of mean and median called *asymptotic relative efficiency* of the median and defined as

$$\frac{\sigma^2(\text{Mean})}{\sigma^2(\text{Median})} = \frac{2}{\pi}. \tag{5}$$

For a Gaussian distribution, $\sigma^2(\text{Mean}) = 1/N$.

Min-max rejection: We use as estimator the following value

$$x_{\text{min-max}} = \frac{1}{N - N_{\text{low}} - N_{\text{high}}} \sum_{i=N_{\text{low}}+1}^{N-N_{\text{high}}} x_i \,. \tag{6}$$

Experimentally we expect the variance of the min-max rejection estimator to lie between those for the arithmetic mean of the full sample and the median.

Kappa-sigma clipping: In the $\kappa\sigma$ clipping algorithm, all values that deviate from the mean by more than κ standard deviations are rejected as outliers. Typically, $\kappa = 3$. Here the variance is therefore estimated as

$$\sigma^2_{\kappa\sigma} = \frac{1}{N^2_{\text{good}}} \sum_{\text{good}} \sigma_i^2 \,. \tag{7}$$

For $\kappa\sigma$ clipping to work reliably, it should use robust estimators of location and scale, i.e. median and the inter-quartile range, instead of the mean and standard deviation.

3.2 Representation of spectral coordinates in FITS

In most cases where the recipe is applied to data we expect that the one-dimensional input spectra are sampled on identical wavelength grids. However, the situation may arise that spectra will have to be rebinned before coaddition.

Raw spectra are typically sampled on a non-uniform grid in wavelength, given by the grism and the optical system of the instrument. For practical purposes, one often prefers spectra that are sampled on a linear grid with constant bin width or spectra that are sampled on a logarithmic grid. The latter case is particularly important for velocity or redshift measurements using cross-correlation techniques.

By default, the recipe will rebin all input spectra to the wavelength grid specified by the first frame in the SOF, regardless of whether this is linear, logarithmic or other. The recipe also allows the user to specify an output grid via a set of recipe parameters. Support is provided for linear and logarithmic binning in this case.

In this section, we summarize how spectral coordinates are represented by WCS keywords in FITS headers. The standard reference is Greisen et al. (2006).

The recipe creates one-dimensional fits files on output, i.e.

```
NAXIS = 1
```

The length of the spectrum in pixels is given as `NAXIS1`. Wavelengths are given in units of Ångstrom:

```
CUNIT1 = 'Angstrom'
```

The reference pixel is always assumed to be the first pixel:

```
CRPIX1 = 1.
```

Linear wavelength coordinates are defined by

$$\lambda_i = \lambda_0 + (i - i_0)\Delta. \tag{8}$$

The correspondence is

```
CRPIX1 = i0 = 1.
CRVAL1 = λ0
CD1_1  = Δ
CTYPE1 = 'WAVE    '
```

`Iraf` uses this convention but identifies it by `CTYPE='LINEAR'` and `DC-FLAG = 0` (Valdes 1993, 1999).

The recipe parameters for the specification of a linear wavelength grid are `--scale=lin`, `--lambda_start` (`CRVAL1`), `--disp` (dispersion `CD1_1`)[2] and `--npix` (`NAXIS1`).

For logarithmic wavelength coordinates, at least two conventions exist which are, unfortunately, not compatible.

`Iraf` uses a variant of Eq. (8) with decadic logarithms (Valdes 1993, 1999):

$$\log \lambda_i = \log \lambda_0 + (i - i_0) \log \Delta \tag{9}$$

or

$$\lambda_i = \Delta^i \lambda_0. \tag{10}$$

The correspondence to FITS keywords is

```
CRVAL1  = log λ0
CD1_1   = Δ
CTYPE1  = 'LINEAR  '
DC-FLAG = 1
```

The standard defined by Greisen et al. (2006) uses natural logarithms and corresponds to

$$\lambda_i = \lambda_0 \, \mathrm{e}^{\Delta(i-i_0)/\lambda_0} \tag{11}$$

where the logarithmic bin width is

$$\ln \lambda_{i+1} - \ln \lambda_i = \frac{\Delta}{\lambda_0}. \tag{12}$$

Comparison with $\mathrm{d}\ln\lambda = \mathrm{d}\lambda/\lambda$ shows that Δ represents the linear bin width (dispersion in Å/pixel) at the start of the wavelength range, λ_0.

The correspondence to FITS keywords is

```
CRVAL1 = λ0
CD1_1  = Δ
CTYPE1 = 'WAVE-LOG'
```

Both `CRVAL1` and `CD1_1` are given in units of Ångstrom.

We *provisionally* implement the standard defined by Greisen et al. (2006) in our recipe. This convention is understood by `wcslib` which is used by the wcs routines in `CPL`. Unfortunately, it appears that

[2] We do no use the alternative keyword `CDELT1`, although this is understood by the wcs functions in CPL.

this convention is not understood by `Iraf`. Conversely, `wcslib` seems to ignore the keyword `DC-FLAG` which distinguishes linear from logarithmic coordinates in the `Iraf` convention.

The recipe parameters for the specification of a linear wavelength grid are `--scale=log`, `--lambda_start` (`CRVAL1`), `--disp` (dispersion at λ_0, `CD1_1`) and `--npix` (`NAXIS1`).

It would be desirable to specify the wavelengths units in the fits header via `CUNIT1` as mentioned above. Unfortunately, at least the UVES pipeline does not specify this keyword. We have noticed that the conversion routine `cpl_wcs_convert` returns wavelengths in units of metres if `CUNIT1` is present, and in native units if it is not. In order to be consistent, we therefore have to drop `CUNIT1` from FITS headers created from recipe parameters.

3.3 Rebinning of spectra

This section outlines the main idea of the basic rebinning algorithm, which redistributes flux intensity from one sampling domain into another.

We assume that we are given the wavelength intervals $I_1 = (\lambda_0, \lambda_1), \ldots, I_n = (\lambda_{n-1}, \lambda_n)$, such that $\lambda_0 < \ldots < \lambda_n$, together with corresponding fluxes $\phi_1, \ldots, \phi_n$. We want to redistribute these fluxes into the given intervals $J_1 = (\lambda'_0, \lambda'_1), \ldots, J_m = (\lambda'_{m-1}, \lambda'_m)$, such that $\lambda'_0 < \ldots < \lambda'_m$. The resulting fluxes are denoted by $\psi_1, \ldots, \psi_m$.

Each interval J_j receives flux from all intervals I_i with which it overlaps. Specifically, the fraction of flux distributed from the interval I_i to the interval J_j is proportional to the length of their intersection $I_i \cap J_j$ according to the formula

$$\psi_j = \sum_i \frac{|I_i \cap J_j|}{|J_j|} \phi_i, \tag{13}$$

where $|I|$ denotes the length of the interval I. This method results in a local conservation of the flux.

4 Data handling and recipe specifications

For efficient data handling the pixel values from the input spectra are arranged in a row-stacked form using the CPL matrix data structure. Matrix columns correspond to constant wavelength values, the rows contain the individual spectra, On this matrices, we apply the algorithms and the mathematical processing and the final result will be in the last processing step saved again as a one-dimensional spectrum. For this procedure to work we use some function from the CPL release and some function written by the EsoSoft team.

The workflow of this data handling is as follows:

Within the recipe the spectra are obtained from the `framelist`, which is created from the given SOF file. Each spectrum is extracted as a `cpl_image` object. Further on, for each spectrum, we obtain the pointer to its data. The pointer is of type void. The data associated with this pointer is then type casted as double, and wrapped as `cpl_matrix`. On this matrix we perform the collapse over each column using one of the robust mean estimation algorithms described in the previous section to obtain the stacked spectrum. The stacked spectrum is then re-converted to the `cpl_image` format, and saved as a DFS compliant fits file.

As specified in the Statement of Work the software delivery will include CPL-based recipes at the level of an instrument-independent pipeline. The recipes can be executed using the ESO pipeline standard tools such as EsoRex or can be used as actors in the ESO Reflex Kepler-based workflow

engine. The code will be documented using `doxygen`[3]. The software will be developed following the standard ESO guidelines (VLT-SPE-ESO-19000-1618 document) on CPL 5.2.0 based functions. At present, the workflow contains the following modules:

- Processing of input data: data organizer for one-dimensional spectra to be stacked according to their acquisition history (single OB, different OBs, different nights)
- Conversion procedures from CPL-image format to matrices for algorithmic handling as described above
- Stacking algorithms following the methods described in the algorithmic part, including detection and rejection of cosmic rays and rejection of data based on seeing, image quality, or signal-to-noise criteria.

All modules include error propagation.

Further enhancements will be done by re-gridding for spectra from different OBs. The software delivery will include also validation reports on test data.

References

Greisen, E. W., Calabretta, M. R., Valdes, F. G., & Allen, S. L. 2006, A&A, 446, 747

Valdes, F. 1993, in ASP Conf. Ser. 52: Astronomical Data Analysis Software and Systems II, ed. R. J. Hanisch, R. J. V. Brissenden, & J. Barnes, 467–471

Valdes, F. 1999, `http://iraf.net/irafdocs/specwcs.ps.gz`

[3] `http://www.stack.nl/~dimitri/doxygen/`