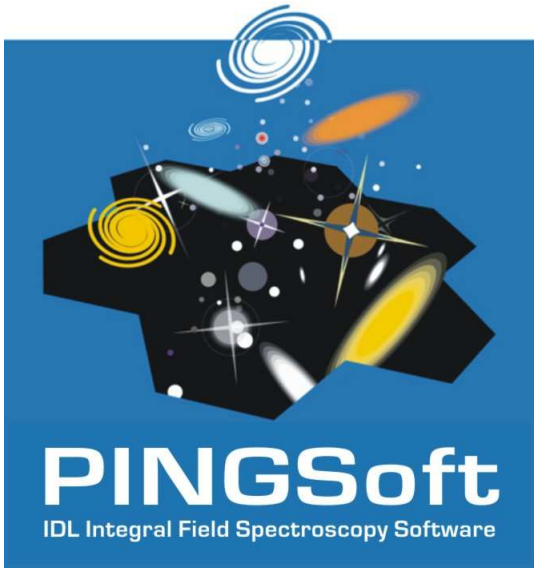

PINGSoft 2 User's Guide

F. Fabián Rosales-Ortega

Departamento de Física Teórica, Universidad Autónoma de Madrid, Spain
Instituto Nacional de Astrofísica, Óptica y Electrónica, Mexico

frosales@inaoep.mx

November, 2012

Contents

1 What is PINGSoft?

The PINGS software, or PINGSoft, is a set of IDL routines designed to visualise, manipulate, and analyse integral field spectroscopy (IFS) data regardless of the original instrument and spaxel shape (Rosales-Ortega, 2011). The routines were originally developed for PINGS: the PPAK IFS Nearby Galaxies Survey project (Rosales-Ortega et al., 2010).

PINGSoft 2 is a relatively major upgrade with respect to the first version: the overall functionality and layout have been improved, while the command syntax has been simplified. This version includes new routines that offer powerful spatial and spectral visualisation of the data, improved extraction routines, and new analysis tools. PINGSoft is optimised for a fast visualisation rendering, it supports 3D cube and RSS formats, it is able to run on practically any computer platform with minimal library requirements, and is adapted to work natively with the CALIFA survey data.

1.1 New features

- A new Graphical User Interface (widget) for interactive visualisation of the spaxels and spectra of a 3D cube or RSS file.
- The data can be convolved with a full set of narrow and broad-band filters for visualization and/or analysis purposes. The filter used to visualize the data is shown on the spectral window.
- Elliptical apertures for spectra extraction are now supported (for any ellipticity, size and PA).
- Radial binning extraction with either fixed bins or based on a S/N floor.
- Spectra extraction and integration based on a user-given mask.
- Conic or hyperbolic aperture extractions for any PA, size and angle.
- Spectra integration based on S/N on continuum and/or emission line features.
- Voronoi binning based on the method developed by Cappellari & Copin (2003).
- Intrinsic velocity field correction using a wavelength cross-correlation.
- Furthermore, the PINGSoft routines can now read 3D cubes or RSS files indistinctively. The syntax is much simpler, e.g. to load a RSS file the user only needs to include the name of the FITS file (and omit the position table) if the format is the following: `OBJECT.fits`, `OBJECT.pt.txt`, and both files reside on the same directory.

IMPORTANT:

If you find this code useful for your research please acknowledge the use of PINGSoft in your publications:

Rosales-Ortega (2011) NewAstron 16, 220

2 Installation

In this document we assume that the user will be installing PINGSoft in a Linux/Mac UNIX-based computer via a terminal window running IDL, and that the software will be called within an IDL-running terminal in command-based mode.

1. Unpack the TAR file containing the PINGSoft 2 library (`pingsoft_v2.tar.gz`). This will create a directory named `pingsoft/`, with all the codes of the distribution and additional subdirectories. Include this directory in your `$IDL_PATH`.

2. PINGSoft requires the ***latest*** versions of: a) the NASA IDL Astronomy Library; b) the set of routines created by David W. Fanning, known as the Coyote Library, both available at:

   - http://idlastro.gsfc.nasa.gov
   - http://www.idlcoyote.com/documents/programs.php

   respectively, these additional libraries should be included in your `$IDL_PATH`.

3. Optionally, for the `voronoi_3D.pro` routine the user will need the Voronoi Binning IDL source code method by Cappellari & Copin 2003, available at: http://www-astro.physics.ox.ac.uk/~mxc/idl/#binning

4. Finally, you need to define a `$PINGSOFT_PATH` system variable, pointing to the location where the `pingsoft/` directory resides, e.g. for a Bash shell:

```
# ---- PINGSoft path
export PINGSOFT_PATH=${HOME}/idl/pingsoft
```

This is all you need to install PINGSoft, to check if the installation was done correctly, open a **new** terminal, run IDL and type:

```
IDL> check_pingsoft
```

If the installation is correct, you should see the following window:

Otherwise, the script will prompt if you miss any library or the `$PINGSOFT_PATH` is not properly set.

2.1 IDL Graphic Settings

PINGSoft uses by default a white background and a combination of certain X-device parameters and colour tables, contrary to the default IDL graphic device settings (e.g. with black as background colour). These graphic settings are defined in the `pingsoft_display.pro` routine. If you want to come back to your original display settings you need to exit IDL and log in again, without invoking any PINGSoft routine.

2.2 Mouse cursor problems in Mac OSX

Some Mac OSX users might find problems using properly the mouse cursor in IDL. This is related with some conflicts between the X11 settings and the default window manager. In particular for Snow Leopard, the user needs to install and use XQuartz instead of the default X11 application (in Lion Mountain X11 is not even installed by default). XQuartz is available at:

http://xquartz.macosforge.org/

Most of the cursor problems (mouse hanging while `cursor` command is active) can be solved by going to the X11/XQuartz Preferences, Windows, then check "Click-through Inactive Windows". Additional information on how to solve the mouse cursor problem can be found here:

http://tinyurl.com/mouse-idl-mac
http://www.idlcoyote.com/misc_tips/maccursor.html

3 Supported formats

PINGSoft supports 3D cubes and Row Stacked Spectra (RSS) FITS files, all routines can read 3D cubes or RSS files indistinctively (no need to transform 3D cubes to the RSS format or to indicate the data type). Furthermore, PINGSoft supports natively all data formats and FITS extensions of the CALIFA survey.

All PINGSoft routines follow a very simple syntax, 3D cubes can be loaded by simply entering the FITS file name (both `.fits` and `.fits.gz` extensions are supported). To load a RSS file the user only needs to include the name of the FITS file, and omit the position table if the file name format is the following:

FITS RSS file: `OBJECT.fits` (or `OBJECT.fits.gz`)
Position table: `OBJECT.pt.txt`

and both files reside on the same directory (see below for an explanation of the RSS format). If the IFS data to visualise/analyse are in the data formats explained below, then PINGSoft should work smoothly and the visualisation/manipulation of the data should be straightforward.

3.1 The 3D cube format

In the case of 3D cubes, the first two data axes should correspond to the spatial axes, the Right Ascension (RA) is assumed to be in first data-axis, Declination (Dec) in the second data-axis. The third axis corresponds to the dispersion axis. The spaxel shape should be squared in a continuous (contiguous) grid. PINGSoft assumes that the spaxel size/resolution is contained in the `CDELT1` FITS header entry, in units of arcsec. The dispersion vector is built from the values in `CRVAL3`, `CDELT3`, `CRPIX3`, and `NAXIS3`, supported units are Angstroms, microns and milli-arcseconds. Negative `CRPIX3` values are supported.

In the case that the World Coordinate System (WCS) is included in the FITS header, PINGSoft assumes that the reference Right Ascension value is stored in the `CRVAL1` entry, and the reference Declination in the `CRVAL2` value, both must be in degrees. The reference spatial position coordinates (from which RA-Dec offsets are calculated) is assumed to be in the `CRPIX1` and `CRPIX2` values.

3.2 The RSS format

The Row Stacked Spectra or RSS format consists in a 2D FITS image in which the X-axis corresponds to the dispersion axis, and the other one corresponds to a given spatial ordering of the spectra determined by a corresponding position table, i.e. the N-row in the Y-axis corresponds to the spectrum at the position $(X_N, Y_N) \equiv (\Delta\mathrm{RA}_N, \Delta\mathrm{Dec}_N)$ from a (0, 0) reference point (in arcseconds), which is the N entry of the `ASCII` file position table. The number of rows on the RSS FITS file is equal to the total number of spectra of the IFS data, i.e. the spectra are *stacked one on top of each other* in the Y-axis. The format of the input position table should be the following:

```
C       1.34       0         0
1     -13.936      0.000     1
2      15.686      3.019     1
3       0.000     12.071     1
4     -13.936     12.077     1
5      13.935     12.077     1
6     -10.452      6.038     1
...
```

where the first line determines the size and shape of the spaxel (see below), and the following entries correspond to: $\mathtt{ID}_{\#}$ $\mathtt{X}_{\mathrm{offset}}$ $\mathtt{Y}_{\mathrm{offset}}$ `flag`, where $\mathtt{ID}_{\#}$ is the spectrum number identification (integer value) and `flag` might be any numerical value (used sometimes for internal quality control). The size and shape of the spaxels is determined by the **first two entries** of the first line of the position table, the first character should be either a `C` or `S`, which corresponds to a **c**ircular (e.g. fibre, PPAK) or **s**quare (e.g. PMAS, VIMOS) spaxel, and the second entry should be a floating number corresponding to the radius or side length respectively, e.g.:

`C`	`1.34`	Circular spaxel, with a radius 1.34 arcsec, i.e. PPAK
`S`	`0.67`	Square spaxel, with sides of length 0.67 arcsec, i.e. VIMOS

see the position tables in the `$PINGSOFT_PATH/pos_tables` directory for more examples.

In addition to the format restrictions mentioned above, the RSS file should be wavelength calibrated and the spectral information should be included in the FITS header, namely the `CRPIX1`, `CRVAL1` and `CDELT1` values.

3.3 Default supported instruments

Some IFU instruments are supported by default by PINGSoft, in which case no additional position tables are needed. PINGSoft can recognise the default instruments from the FITS header. The default instruments/configurations are:

- PMAS: single pointing, 0.5, 0.75 & 1.0 arcsec configurations.
- PPAK: central bundle (331 fibres), central + sky bundles (382 fibres) & standard dithered configuration (993 fibres).
- VIMOS: single pointing, all resolutions in both configurations (40×40 & 80×80, 0.33 & 0.67 arcsec), and HR dithered "super-cube" (square 4 pointing dither pattern, (44 × 44 spaxels, $29.7'' \times 29.7''$).
- INTEGRAL-WHT: SB1, SB2, and SB3 standard bundles.

See Sec. 5 for more information on these instruments. Additionally, the p3d software for IFS data reduction (Sandin et al., 2010) includes fiber position tables in the PINGSoft format for most instruments supported by p3d[1]. PINGSoft is known to work also with SINFONI data.

[1] Stored at: `$p3d_path/data/tables/pingsoft_postables/`

4 The PINGSoft integral field spectroscopy software

All PINGSoft routines are called via command lines in a terminal running IDL. The syntax and online help for any program can be obtained by entering the name of the procedure without any parameter or keyword, With the exception of the visualisation widget: `view_ifs, /help`.

Additionally, in the PINGSoft webpage you can download the `pingsoft_examples/` directory which includes some 3D cubes and RSS example files. I recommend the user to download this example data and follow the instructions in the `README.pro` file in order to get a first insight of the main PINGSoft routines. All the example commands used in this document can be found in the `README.pro` file of the `pingsoft_examples/` directory.

Visualisation

view_ifs

This routine provides a spatial and spectral interactive visualisation widget for 3D cubes and RSS IFS files. If the command is simply entered in the IDL terminal:

```
IDL> view_ifs
```

it prompts for an input FITS file using a dialog window. Otherwise, the input file can be passed directly to the command as the first parameter:

```
IDL> view_ifs, 'OBJECT.fits'
```

The widget will be displayed automatically if the input file is a 3D FITS cube. If the input FITS is a RSS file, the program will look in the same directory for a position table named `OBJECT.pt.txt` and will launch the widget if the file exists. If this is not the case, the program will exit with an error. The user can define the name of the corresponding position table using the `PT` parameter:

```
IDL> view_ifs, 'OBJECT.fits', PT='PosTable.txt'
```

The `view_ifs` widget is shown in Fig. 1, displaying the 3D cube file `ngc4625.rscube.fits` included in the `pingsoft_examples/` directory. The widget displays two main panels, on the right a visualisation of the spatial distribution of spaxels (or field-of-view, FoV). The color-scaling corresponds to a narrow/broad-band image of a transmission filter convolved with the data at a given wavelength [2] (shown as a dotted-curve in the spectral window). The spatial units are assumed to be arcseconds in a standard North (up) East (left) configuration. The left panel shows the spectrum of the spaxel corresponding to the position of the mouse, the wavelength range is extracted from the information on the FITS header. The corresponding spaxel position, ID[3] and offsets are shown on the top of the spectral window. Optionally, if the WCS is included in the FITS header, the RA and Dec are also shown in sexagesimal and degree units, a mouse LEFT-click prints the same spaxel information on the IDL terminal where the program was called.

USAGE: At first glance, the usage of the `view_ifs` widget may seem "tricky", but it is easy to get used to: when the widget is launched, the user can explore spatially and spectrally the IFS data but the widget options will be *inactive*. To have access to the widget options, the user needs to RIGHT-click with the mouse over the FoV panel. When the widget options are changed the spectral explorer will become active again and the widget options will be unavailable. To active/deactivate the widget options the user only needs to RIGHT-click over the FoV to switch between the explorer ON/OFF options. The *status* bar above the FoV panel (below the object name) will indicate whether the explorer is active or not.

[2] Default: Hα narrow-band filter of 80 Å FWHM with central wavelength at 6547 Å, if this wavelength is outside the spectral range, the filter is shifted to the mean wavelength.

[3] In the IDL format, i.e. starting at zero

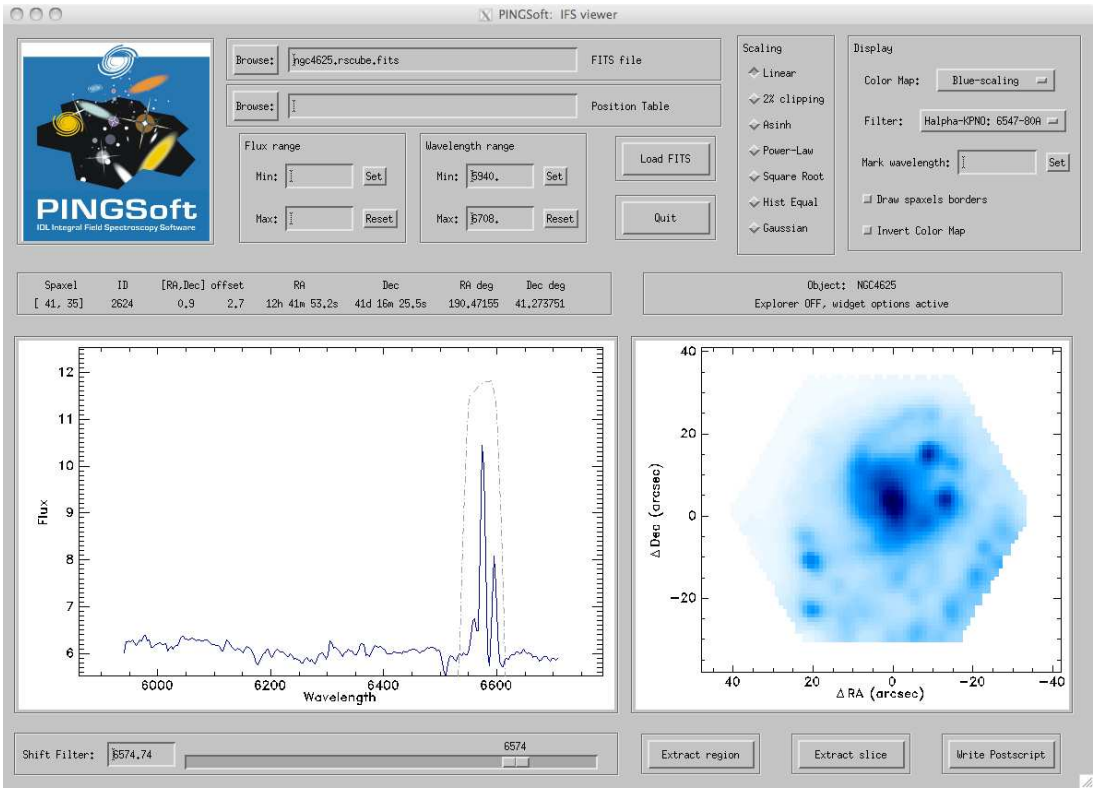

Figure 1: Screen shot of the visualisation widget launched by the `view_ifs` command, displaying the 3D cube of the galaxy NGC 4625 included in the `pingsoft_examples/` directory.

Several options are available to visualise the data, including different intensity scalings, color maps, a set of different narrow and broad-band filters in the optical to generate the visualisation in the FoV panel, the choice to define the flux intensity and spectral ranges, to drawing the contour of the spaxels, to invert the color-map, etc. Note that the central wavelength of the filter used to display the data can be shifted to any position along the spectral range, either by using the slider or by setting the wavelength in the corresponding field. New FITS (position tables) files can be loaded using the corresponding fields at the top-center of the widget, and by pressing the "Load FITS" button.

Extract region: By pressing this button, all the subsequent LEFT-clicks over the FoV panel will mark and select the spaxels to be extracted. When the program is terminated (by RIGHT-click) the following files are created:

```
     Extracted RSS:   OBJECT_rss.fits    (Extracted RSS of the selected spaxels)
    Position table:   OBJECT_rss.pt.txt  (Position table of the new RSS file)
Integrated ASCII:   OBJECT_integ.txt   (Integrated spectrum in ASCII format)
            FITS:   OBJECT_integ.fits  (Integrated spectrum in FITS format)
      Postscript:   OBJECT_integ.eps   (Postscript image of the integrated spectrum)
     IDL indices:   OBJECT_index.txt   (IDL indices of the selected spaxels)
```

shown in the IDL terminal window, while the spectral panel will show the integrated spectrum of the selected spaxels.

Extract slice: Pressing this button will invoke the `extract_filter` command with the filter and central wavelength parameters as the current values displayed in the widget. This will create a FITS image file called `OBJECT_slice.fits` as reported in the status bar, additional information will be shown in the IDL terminal window. WARNING: This option is only available for 3D cubes and RSS with a rectangular-contiguous grid.

Write Postscript: Pressing this button will create an encapsulated Postscript image (`OBJECT_FoV.eps`) of the FoV panel with the current display options of the widget. The name of the file will be reported in

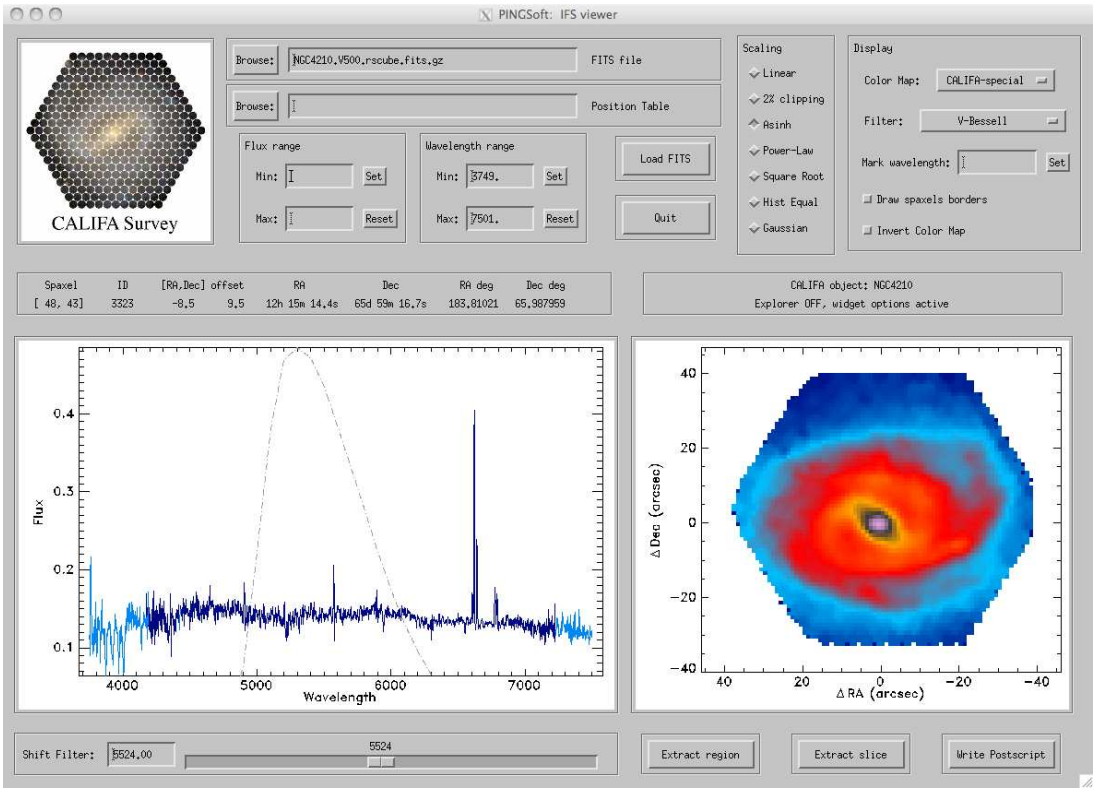

Figure 2: Screen shot of the visualisation widget for a CALIFA datacube, the bad-pixels are displayed in a light-blue color in the spectral panel.

the status bar and terminal window.

Mark wavelength: Use this field to enter one or several wavelengths at which a vertical line will be drawn in the spectral panel. This option is useful when trying to identify features at known wavelengths. The input formats can be of the form:

```
Mark wavelength: 6563
                 5007, 6563
                 5895*1.002          (e.g. known redshifts)
                 [4310,5876]*1.0015
```

CALIFA data: When a CALIFA data file is loaded with `view_ifs`, the display color is changed to the CALIFA-special color map, the routine identifies automatically the different FITS extensions (HDUs) of the CALIFA format (for both RSS and 3D cube versions). The BADPIX extension is shown in the spectral panel simultaneously with the flux data, the bad-pixels are displayed in a light-blue color as shown in Fig. 2.

Size and resolution: The `view_ifs` widget may not display properly for screens with resolutions lower than 1400×800. In this case, the user can modify by hand the size of the widget to fit their own screen resolution by editing the first entries in the `widget_param.pro` routine.

`view_3D`

This routine is the command-line version of `view_ifs`, it provides a 2D interactive visualisation of the spaxels and spectra of a 3D cube or a RSS file and its corresponding position table. However, the visualisation is performed in two standard IDL windows (i.e. a lighter visualisation option). All display and interactive options are similar to `view_ifs` (with the exception of the "Extract slice" option), the command is terminated by RIGHT-click on the FoV window.

A mouse MIDDLE-button click is equivalent to the "Extract region" button in the widget, it prompts in the IDL terminal for a PREFIX used to generate the new series of files, all the subsequent LEFT-clicks over the FoV window will mark and select the spaxels to be extracted. The extraction is performed by RIGHT-click on the FoV window, the extracted files are displayed in the terminal window, while the spectral window shows the integrated spectrum of the selected spaxels.

Additional features:

1. The `view_3D` routine accepts the `/LARGE` keyword, which displays a much larger FoV window.
2. The user can specifying the FITS extension to read using the `EXTENSION` keyword
3. An optional output IDL structure can be obtained when spaxels are manually selected.
4. The `_EXTRA` structure keyword can be set for user's defined specific graphics output, both for the IDL window or Postscript output, e.g. `_EXTRA={title:'IFS cube',xrange:[-60,50]}`

Calling sequence:

```
view_3D, 'OBJECT.fits' [, OUT.str, PT='Ptable.txt', EXTENSION=extension, $
                       MIN_FLUX=min_flux, MAX_FLUX=max_flux, LMIN=lam_min, LMAX=lam_max, $
                       FILTER=filter, BAND=band, CT=ct, VLINE=vline, FONT=font, $
                       /CLIP, /GAMMA, /LOG, /ASINH, /SQRT, /HISTOGRAM, /GAUSSIAN, $
                       /PS, /DRAW, /LARGE, /NOBAND, _EXTRA={extra} ]

INPUTS:

        'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

OPTIONAL KEYWORDS:

              OUT.str:  Output IDL structure (when spaxels are manually selected).

      PT='Ptable.txt':  Name of the position table in ASCII format for an input RSS file
                        in (North-East configuration).
                        NOTE: compulsory if not included in the default instruments/setups
                        or when the name is not in the 'OBJECT.pt.txt' format.

            EXTENSION:  Non-negative scalar integer specifying the FITS extension to read.
                        For example, specify EXTENSION = 1 to read the first FITS extension.

         MIN/MAX_FLUX:  Minimum/maximum flux in the spectral window to be plot, if
                        not set these are floating values.

            LMIN/LMAX:  Defines the wavelength range on the spectral window,
                        if not set values are taken from the FITS header.

               FILTER:  Internal number of the narrow or broad-band filter used to
                        display the data. Available filters and corresponding
                        numbers can be obtained by typing:  IDL> pingsoft_filters
                        Default: 1 (Halpha KPNO-NOAO - CWL: 6547A  FWHM: 80)

                 BAND:  Central wavelength of the narrow or broad-band used to
                        display the data, i.e. shifts the band to the position
                        defined by the user (if within the spectral range).
                        Defaults: nominal central wavelength of the
                        corresponding filter (mean wavelength if outside the range).

                VLINE:  Either a floating value or a vector of floating
                        numbers containing the lambda value at which a single or
                        several vertical lines will be drawn for reference
                        purposes, equivalent to "Mark wavelength" in the VIEW_IFS widget,
                        e.g. VLINE=6500 or VLINE=[4200,5400,6700].

                   CT:  IDL Color Table used to display the data, (default ct=1, BLUE/WHITE).

                  /PS:  Writes an encapsulated Postscript file of the spaxels visualisation.

                 FONT:  Postscript IDL font to be used when /PS is set. Default: 12 (Helvetica)

                /DRAW:  Draws the contours of the spaxels.
```

```
        /LARGE:  Displays a larger window for the spatial distribution
                 of spaxels (right window).

       /NOBAND:  The narrow/broad band is not drawn in the spectral window.

        _EXTRA:  Structure with the _EXTRA tags for user's defined graphics output,
                 e.g. _EXTRA={title:'IFS cube'}

Intensity Scalings:

      Default:  LINEAR, displays the range of intensities using a linear min/max scaling.

        /CLIP:  A histogram stretch, with a 2% of pixels clipped at both the top and bottom.

       /GAMMA:  Displays the range of intensities using a Power-law (gamma) scaling.

         /LOG:  Displays the range of intensities using a logarithmic scaling.

       /ASINH:  Displays the range of intensities using an inverse hyperbolic sine function scaling.

        /SQRT:  Displays a linear stretch of the square root histogram of the image values.

   /HISTOGRAM:  Displays a linear stretch of the histogram equalized image histogram.

    /GAUSSIAN:  The scaling is performed by applying a Gaussian normal function to the image histogram.
```

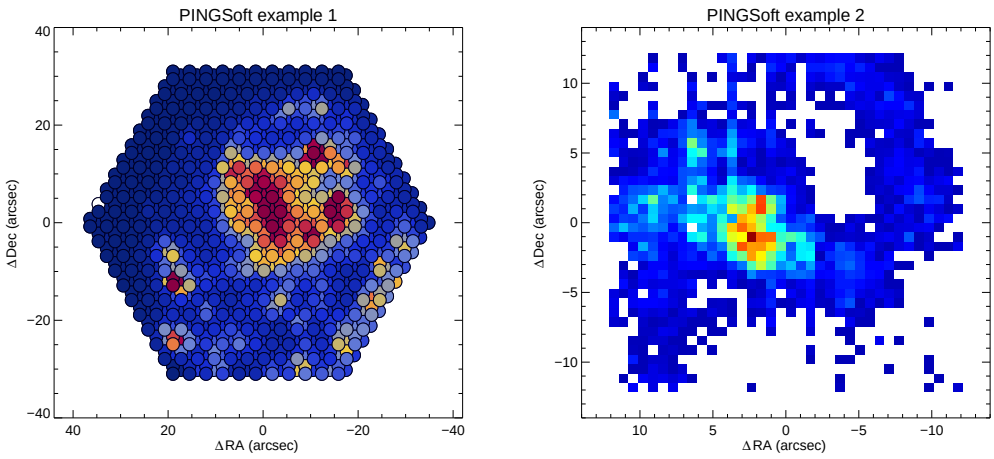

Figure 3: Postscript outputs examples of the `view_3D` routine

Examples:

To visualise the RSS file `IRAS06295.VIMOS.fits` with position table named `IRAS06295.VIMOS.pt.txt` (both included in `pingsoft_examples/`) limiting the intensity and wavelength range on the spectral window, drawing two vertical lines at lambda 6200 and 6700, using an inverse hyperbolic sine function scaling:

```
view_3d, 'IRAS06295.fits',  min=-1, max=6, lmin=6200, lmax=6800, vline=[6200,6700], /asinh
```

To create a Postscript image of the `ngc4625.dither.fits` RSS file, with the spaxels drawn, the PINGSoft-special colour table, a 2% clipping scaling, using a narrow-band Hα 20 Å filter, and a special title (shown in Fig. 3):

```
view_3d, 'ngc4625.dither.fits', ct=44, filter=2, /clip, /draw, /ps, _extra={title:'PINGSoft example 1'}
```

To create a Postscript image of `IRAS06295.rscube.fits`, with rainbow colour table, and using a B-Johnson (1965) filter and a special title (shown in Fig. 3):

```
view_3d, 'IRAS06295.rscube.fits', ct=33,  filter=3, /ps, _extra={title:'PINGSoft example 2'}
```

Spectra extraction

The following section describes the extraction routines of PINGSoft. Given the similarity of the inputs and/or optional keywords of these routines, only the relevant parameters for each program will be explained in this document. For more information see the online help (by typing the command line in the IDL terminal). Note also that, by default, all routines opens a FoV visualisation as `view_3D`, showing the simulated aperture and extracted spaxels. On the other hand, if the `/PS` keyword is set, a Postscript image of the FoV is generated. The `/PLOT` and `/PS` keywords are mutually exclusive. All routines have similar display options as in `view_3D`.

`extract_region`

Extracts the spectra of spaxels/regions selected by hand. This is a direct implementation of the "Extract region" button of the `view_ifs` widget or the MIDDLE-click option of `view_3D`. The script opens a similar visualisation as `view_3D`, where the user can select spaxels by successive LEFT-clicks on the right window. The same output files are created as in the previous examples. Similar display options as in `view_3D`.

Calling sequence:

```
extract_region, 'OBJECT.fits' [, OUT.str, PT='PosTable.txt', PREFIX='prefix', $
                               EXTENSION=extension, (+ all visual options of VIEW_3D) ]

        PREFIX:  String with the prefix of the output files, default: 'region'
```

Example:

Extract spaxels from the `ngc4625.331.fits` (single PPAK exposure), with prefix '`TEST` and display the extraction using `view_3D` (example shown in Fig. 4):

```
extract_region, 'ngc4625.331.fits', prefix='TEST'

view_3d, 'TEST_rss.fits', /draw
```

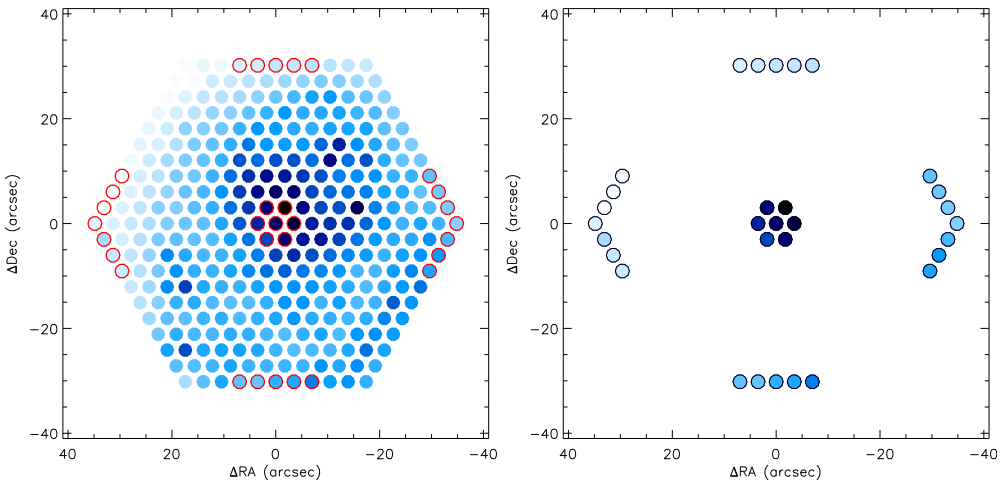

Figure 4: Examples of spaxel extraction using the `extract_region` command. On the left, a single frame of NGC 4625 showing the selected spaxels outlined in red. On the right, visualisation of the selected spaxels by using the `view_3d` command on the generated RSS and position table files.

extract_aperture

Extracts the spectra within an elliptical or circular aperture. The user can define a circular aperture of any physical dimension (in arcsec) centered at a given position from the reference point. The program then extracts the spaxels which centers fall within the aperture, generating a new RSS and position table files, the integrated spectrum in ASCII, FITS and EPS format. Similar display options as in `view_3D`.

CALLING SEQUENCE:

```
extract_aperture, 'OBJECT.fits', x0, y0, semi_major, semi_minor, PA
                  [, out_str, PREFIX='prefix', PT='PosTable.txt', EXTENSION=extension,
                  /PLOT, /PS, FONT=font, _EXTRA={extra},
                  (+ all visual options of VIEW_3D) ]

    'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

           x0/y0:  RA/Dec offsets (in arcsec) of the central position of the aperture to be simulated.

      semi_major:  Semi-major axis of the simulated elliptical aperture (in arcsec)

      semi_minor:  Semi-minor axis of the simulated elliptical aperture (in arcsec)

              PA:  Position angle of the major-axis, measured counterclockwise from the North.

OPTIONAL KEYWORDS

          PREFIX:  String with the prefix of the output files. Default: input positions

           /PLOT:  Display the visualisation and the simulated aperture.

             /PS:  Writes an encapsulated Postscript file of the spaxels visualisation
                   and the simulated aperture.
```

Examples:

A circular aperture of 15 arcsec radius (semi-major = semi-minor), centered at (-2,3), with graphic visualisation, power-law scaling and outlined spaxels (left-panel Fig. 5):

```
extract_aperture, 'ngc4625.dither.fits', -2, 3, 15, 15, 0, /gamma, /draw, /plot, $
                                  _extra={title:'PINGSoft: circular extraction'}
```

An elliptical aperture of 8 arcsec in major axis, 4 arcsec in minor axis, centered at (2,0), with an inclination of 45 degrees, with defined prefix, using the `GREEN/WHITE` colour table (right-panel Fig. 5):

```
extract_aperture, 'IRAS06295.fits', 2, 0, 8, 4, 45, /plot, ct=8, prefix='elipse', $
                                  _extra={title:'PINGSoft: elliptical extraction'}
```

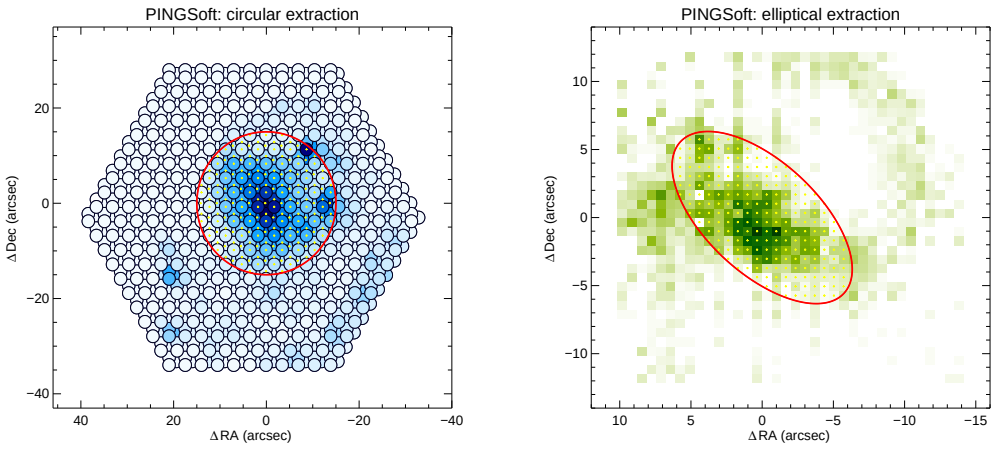

Figure 5: Examples of spaxel extraction using the `extract_aperture` command.

extract_radial

Extracts radial average spectra within consecutive elliptical rings from a reference point, based on either fixed bins or S/N floor. Taking as a reference the input central coordinates the program extracts and co-adds the spectra within successive rings of either 1) a given width in arcsec, or 2) an adaptive size depending on a input S/N target, creating in both cases an integrated spectrum in each of the bins, which are then stored in a RSS file. The initial and final radius for the extraction can be defined. The program also generates a "Radial reference table" with the information of the bin size and integration radius, as displayed in the terminal. This position table can be used to visualise the radial extracted spectra using the `view_3D` command. Similar display options as in `view_3D`.

S/N based extraction: In the case of adaptive bin size based on S/N floor, the S/N is calculated using a "standard" definition, e.g. S/N = μ/σ of the flux band chosen. If the user sets the `/PROXY` keyword, the S/N is calculated using an *optional* S/N definition: S/N = $\mu/\sqrt{(\sigma)}$ which is equivalent to a *density S/N*, i.e. a *PROXY* for the surface brightness of the object. Note that the user can change the calculation of the S/N by editing the `s2n_rat` function at the beginning of the program.

WARNING: The S/N based extraction is **highly** unstable, the S/N depends heavily on the spectral band, width and type of data.

Calling sequence:

```
extract_radial, 'OBJECT.fits', x0, y0, epsilon, PA, step
                [, RMIN=rmin, RMAX=rmax, S2N=s2n, LAM_S2N=lam_s2n, WIDTH=width, /PROXY,
                OUT.str, PT='pos_table.txt', PREFIX='prefix', EXTENSION=extension,
                (+ all visual options of VIEW_3D) ]

 'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

         x0/y0:  RA/Dec offsets (in arcsec) of the central position of the aperture to be simulated.

       epsilon:  Eccentricity of the elliptical apertures (circle=0,  ellipse= < 1)

            PA:  Position angle of the major-axis, measured counterclockwise from the North.

          step:  Radial bin size in arcsec (minimum bin size in case of S/N floor).

OPTIONAL KEYWORDS

          RMIN:  Initial extraction radius (in arcsec), default: 0

          RMAX:  Final extraction radius (in arcsec), default: maximum FoV radius

           S2N:  Floating value of the target S/N for each radial bin.
                 If set, the S/N is calculated for each bin at the minimum radial size
                 set by the 'step' argument, if the S/N is below this value, the radius
                 will grow by 1 arcsec step until the S/N target is achieved.
                 The S/N is calculated using the "standard" definition Mean(flux)/StdDev(flux)

       LAM_S2N:  Central wavelength of the band at which the S/N will be calculated.
                  If not set, default: mean(lambda)

         WIDTH:  Width of the band at which the S/N will be calculated, default: 100

        /PROXY:  The S/N is calculated using an "optional" S/N definition:
                 Mean(flux)/sqrt(StdDev(flux)), which is equivalent to a "density S/N"
                 and it is a PROXY for the surface brightness of the object.
```

Examples:

Radial extraction on a VIMOS field with circular rings (i.e. PA=0, epsilon=0) of 3 arcsec, centered at the origin coordinates (0,0), starting at 2 arcsec from the origin, up to a maximum radius of 12 arcsec:

```
extract_radial, 'IRAS06295.fits', 0, 0, 0, 0, 3, rmin=2, rmax=12
```

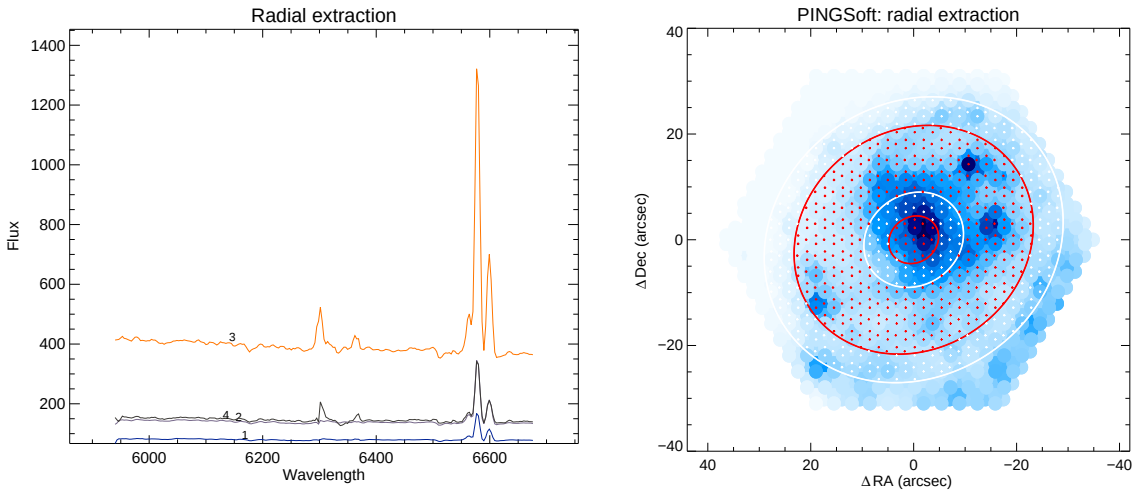

Figure 6: Example of a radial extraction using the `extract_radial` routine.

Radial extraction on NGC 4625 of elliptical apertures with an eccentricity of 0.5, PA=120 degrees, based on a S/N target of 80, calculated at 6100Å, up to a maximum radius of 30 arcsec (see Fig. 6):

```
extract_radial, 'ngc4625.dither.fits', 0, 0, 0.5, 120, 5, rmax=30,  s2n=80, lam_s2n=6100
```

extract_slit

Extracts the spectra within a rectangular aperture, resembling a long-slit observation. The user can define a rectangular aperture of any physical dimensions (in arcsec) centered at a given position from the reference point (in arcsec) and with a given orientation (defined by the PA). The program then extracts the spaxels which fall within the aperture, generating a new RSS and position table files and the integrated spectrum in ASCII and FITS format. Similar display options as in `view_3D`.

Calling sequence:

```
extract_slit, 'OBJECT.fits', x0, y0, length, width, PA
                   [, out_str, PREFIX='prefix', PT='PosTable.txt', EXTENSION=extension,
                   /PLOT, /PS, FONT=font, _EXTRA={extra},
                   (+ all visual options of VIEW_3D) ]

    'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

            x0/y0:  RA/Dec offsets (in arcsec) of the central position of the slit to be simulated.

     length/width:  Physical dimensions of the simulated slit (in arcsec). For the PA convention, length > width.

               PA:  Position angle of the slit, measured counterclockwise from the North.
```

Examples:
A slit of 50×5 arcsec, centered at (-3,0), with a PA=20 degrees, graphic visualisation using Power-law scaling and outlined spaxels:

```
extract_slit, 'ngc4625.dither.fits', -3, 0, 50, 5, 20, /gamma, /draw
```

A slit of 70×5 arcsec, centered at (5,8), with a PA=72 degrees, graphic visualisation using a 2% clipping scaling, with the filter centered at 6700Å using the `BGTY` color table (see left-panel of Fig. 7):

```
extract_slit, 'NGC5947.rscube.fits', 5, 8, 70, 5, 72, /clip, band=6700, ct=4, $
                                /ps, _extra={title:'PINGSoft: slit extraction'}
```

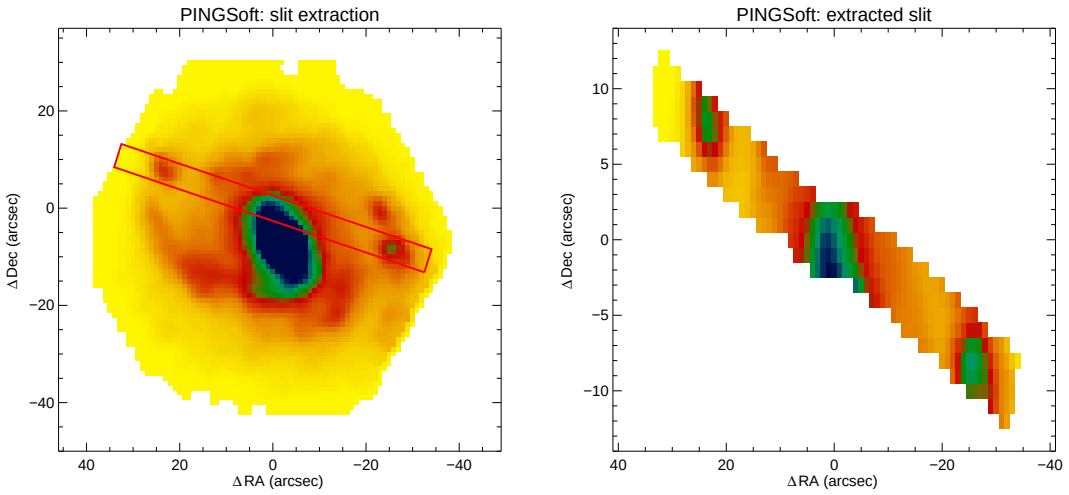

Figure 7: Example of spectra extraction using the `extract_slit` command on NGC 5947 (left) and resulting RSS extracted file (right)

extract_cone

Extracts the spectra within a region defined by a hyperbolic cone. The user can define a circular cone of any physical dimensions (in arcsec) centered at a given position from the reference point. The input parameters are the angle between the major axis and either asymptote of the simulated cone, the PA of the major axis of the cone, measured counterclockwise from the North, and optionally, an `epsilon` parameter controlling the curvature of the vertex, as shown in Fig. 8. The program then extracts the spaxels which centers fall within the cone, generating a new RSS and position table files and the integrated spectrum in ASCII and FITS format. Similar display options as in `view_3D`.

Calling sequence:

```
extract_cone, 'OBJECT.fits', x0, y0, theta, PA  [, EPSILON=epsilon,
              [, OUT.str, PREFIX='prefix', PT='PosTable.txt', EXTENSION=extension,
              /PLOT, /PS, FONT=font, _EXTRA={extra},
              (+ all visual options of VIEW_3D) ]

'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

        x0/y0:  RA/Dec offsets (in arcsec) of the position of vertex of the cone to be simulated.

        theta:  Angle between the major axis and either asymptote of the simulated cone.

           PA:  Position angle of the major axis of the cone, measured counterclockwise from the North.

OPTIONAL KEYWORDS:

      EPSILON:  Floating parameter greater-equal than 1 controling the curvature of the vertex.
```

Examples:

A conic extraction centered at (-15,20), with a PA=140 and theta=20 degrees, for two different epsilon values: 0 and 20, as shown in Fig. 8:

```
extract_cone, 'ngc4625.dither.fits', -15, 20, 20, 140, epsilon=0

extract_cone, 'ngc4625.dither.fits', -15, 20, 20, 140, epsilon=20
```

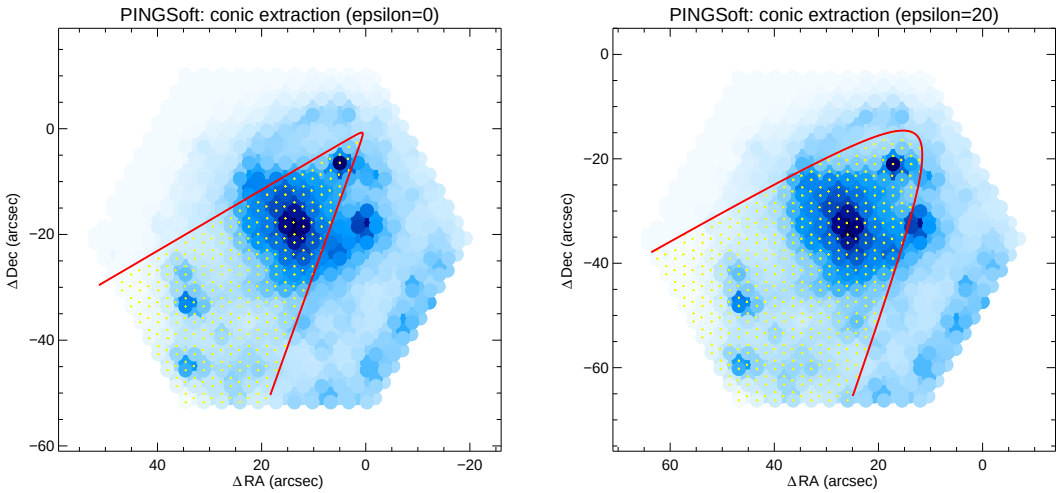

Figure 8: Example of spectra extraction using the `extract_cone` command on NGC 4625, using two different `epsilon` values.

extract_mask

Extracts the spectra based on a user's given mask or segmentation map. The program works in two different ways depending on whether the input is a "mask" (i.e. a FITS image with values 0 and 1) or a "segmentation map" (i.e. a FITS image with integer values ranging from 0,1,2,...,N). In the first case, the routine simply applies a mask extraction to the IFS data, i.e. those regions in the mask image equal to 0 are discarded, while regions with values equal to 1 are preserved, and a new cube is created in which the discarded spaxels are zeroed. In the second case, for each region in the segmentation map with values different than 0, the command integrates in the IFS cube the spectra for all spaxels with the same integer value on the segmentation map, and appends the integrated spectrum to a new RSS file. The final output is a RSS file with the same number of entries as the non-zero integer values of the segmentation map, plus a Pseudo-position table for visualisation. Additionally, the program writes a new 3D cube with the regions segmented as the input segmentation map, in which each spaxel corresponds to the integrated spectrum at that particular region. In the segmentation map case, the program displays the FoV and the segmented regions on an IDL window.

Note the script works only with regular-grid data, i.e. continuous 3D cubes. The 2D mask/segmentation map and 3D cube must have the same X-Y dimensions.

Calling sequence:

```
extract_mask,  'OBJECT.fits', 'mask.fits' [, PREFIX='prefix',
               /NORMALIZE, /SEGMENT, /PS, /SILENT, EXTENSION=extesion
               (+ all visual options of VIEW_3D) ]


    'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

      'mask.fits':  String of the 2D mask or segmentation map FITS file.

OPTIONAL KEYWORDS

           PREFIX:  String with the prefix of the output files. Default: 'mask'

        EXTENSION:  Non-negative scalar integer specifying the FITS extension to read.
                    For example, specify EXTENSION = 1 to read the first FITS extension.

       /NORMALIZE:  If set, the integrated spectrum is normalized by the number
                    of spaxels used to integrate the segmentd region.

         /SEGMENT:  If set, the on-screen or Postscript visualisation of the
                    segmented regions uses a special (nicer?) color-coding
```

```
/NUMBER:  If set, the segmentation integer numbe is plot on top of
          the region

/SILENT:  Omits printing output information on the terminal

    /PS:  Writes an encapsulated Postscript file of the segmented regions

 _EXTRA:  Structure with the _EXTRA tags for user's defined graphics output,
          e.g. _EXTRA={title:'IFS cube'}
```

Examples:

Using the 2D FITS segmentation map `ngc4625.seg.fits` corresponding to regions with Hα emission (H II regions) obtained with HIIExplorer (Sánchez et al., 2012), we extract the integrated spectra within each region, using two different output visualisations options (see Fig. 9):

```
extract_mask, 'ngc4625.rscube.fits', 'ngc4625.seg.fits', prefix='ngc4625', ct=6, filter=2, /clip, /ps

extract_mask, 'ngc4625.rscube.fits', 'ngc4625.seg.fits', prefix='ngc4625', /segment, /number, /ps
```

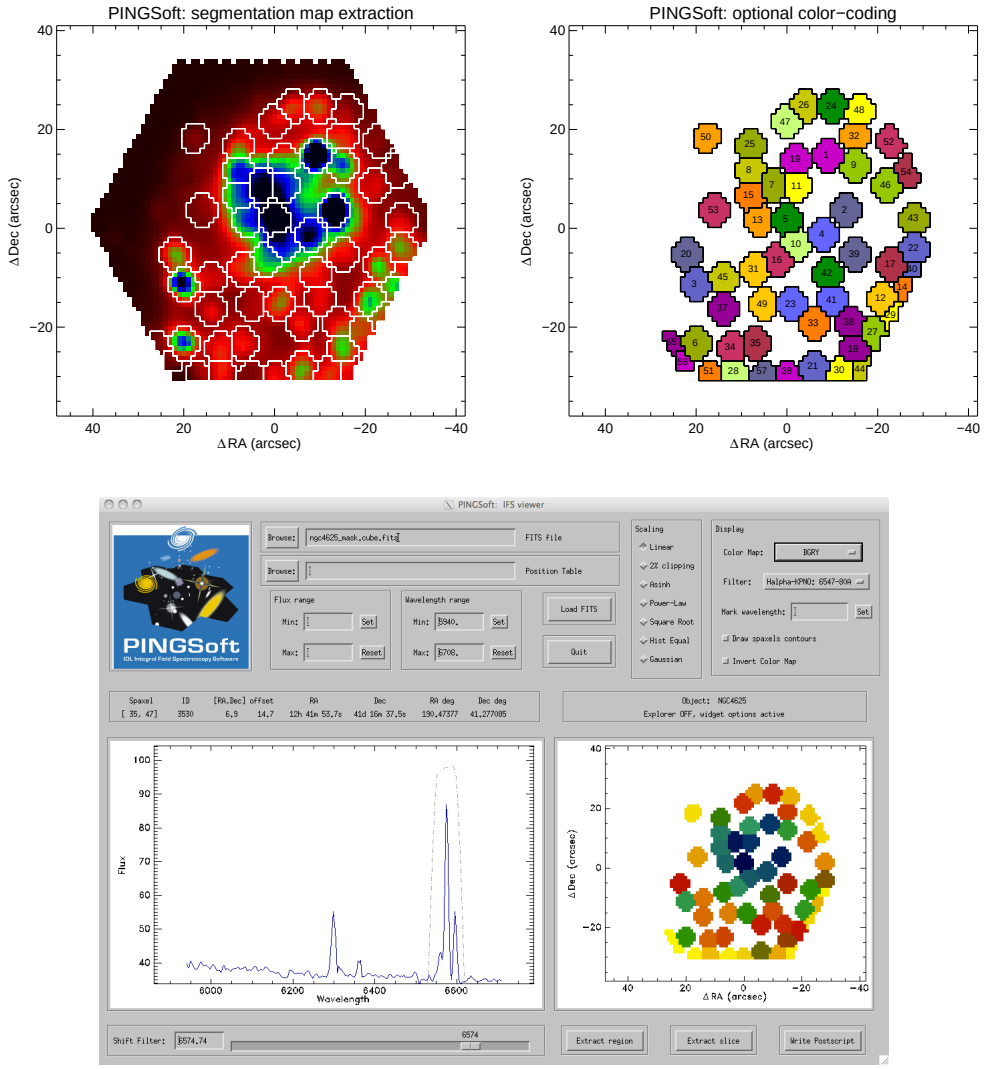

Figure 9: Extracted regions using a 2D segmentation map and the `extract_mask` command on the IFS cube of NGC 4625. *Top-left*: normal visualisation. *Top-right*: using the `/SEGMENT` and `/NUMBER` keywords. *Bottom*: segmented 3D cube output as seen by `view_ifs`.

integrate_3D

Integrates the spectra within a 3D cube or RSS file into a single spectrum. The program generates an ASCII and a FITS file of the integrated spectrum. By default, all the spectra in the 3D cube or RSS file are considered for the integration. This can be changed by setting the `MASK` (3D cube) or `BAD_ROWS` (RSS) keywords. In the first case, the mask should be a 2D FITS image with values 0 and 1, those regions in the mask image equal to 0 are discarded, while regions with values equal to 1 are considered for the spectra integration. The 2D mask image must have the same X-Y dimensions as the input cube. In the second case, the string passed to `BAD_ROWS` should be an ASCII file containing the indices of the spaxels to be neglected. As an output, the integrated spectrum is displayed on screen, and a encapsulated Postscript file is created, unless the `/SILENT` keyword is set.

Calling sequence:

```
integrate_3D, 'OBJECT.fits' [, OUT.str, PREFIX='prefix', BAD_ROWS='bad_rows.txt',
                            MASK='mask.fits', NORMALIZE=normalize, /SILENT ]

    'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

 OPTIONAL KEYWORDS:

          OUT.str:  Name of the optional output structure.

           PREFIX:  String with the prefix of the output files.

             MASK:  String of a 2D FITS image with values 0 and 1, those regions in
                    the mask image equal to 0 are discarded, while regions with values
                    equal to 1 are considered for the spectra integration. The 2D mask
                     image must have the same X-Y dimensions as the input cube.

         BAD_ROWS:  String of an ASCII file with the indices of rows to be discarded
                    if the input is a RSS file (indices in IDL format).

        NORMALIZE:  Value of the wavelength where the flux is to be normalized.

          /SILENT:  Omits printing output information on the terminal

    NOTE: ASCII and FITS output in input units (10^-16 erg s^-1 cm^-2 A^-1 for CALIFA)
```

Examples:

Integrate all the spectra within the RSS file `ngc4625.dither.fits`, normalizing to 6000 Å and excluding the spectra with indices in the ASCII file `bad_index.txt'`:

```
integrate_3d, 'ngc4625.dither.fits', norm=6000, bad_rows='bad_index.txt'
```

Integrate all spaxels within the 3D cube `NGC5947.rscube.fits`, using the 2D image mask `NGC5947_mask.fits`, generated with the s2n_ratio_3D command (see below), which corresponds to all regions with S/N greater or equal to NN, and saving the results in an anonymous structure:

```
integrate_3d, 'NGC5947.rscube.fits', out, mask='NGC5947_mask.fits'

 Files written

   Integrated ASCII:   integ_3D.txt
               FITS:   integ_3D.fits
         Postscript:   integ_3D.eps

       Mask applied:   NGC5947_mask.fits

 IDL structure saved:

   LAMBDA          DOUBLE    Array[501]
   FINTEG          FLOAT     Array[501]
```

Data products and analysis

In this section we introduce routines that generate data products different than extracted spectra, or apply some level of analysis to the IFS data.

extract_filter

Generates a 2D FITS image after convolving the 3D data with a narrow or broad-band filter. The command creates a 2D FITS image slice of the IFS data using either one of the PINGSoft default transmission curves or a user's defined filter. It works **only** with continuous-regular grids. By default, the program opens a FoV visualisation as `view_3d`, showing the output image scaling and in a second window, it displays the integrated spectrum of the IFS data, showing the convolved band. The central wavelength of the filter can be shifted to any within the spectral range using the `BAND` keyword. For narrow band filters, the continuum at ± 100 Å can be subtracted using the `/SUBTRACT` option. The user can supply its own filter by setting the `USER` keyword, with the string of the ASCII file containing the transmission filter in the same format as the PINGSoft filters. Similar display options as in `view_3d`.

NOTE: expect weird orientations/results with RSS data.

Calling sequence:

```
extract_filter,  'OBJECT.fits', filter [, OUT.str, EXTENSION=extension, PREFIX='prefix',
                                BAND=band, USER=user, /SUBTRACT, /PS, PT=pt, _EXTRA=extra,
                                (+ all graphic options as VIEW_3D) ]

   'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

          filter:  Integer value of the default PINGSoft filter to be used (see below)

OPTIONAL KEYWORDS

          PREFIX:  String with the prefix of the output files.

            BAND:  Floating value to which the central wavelength of the broad/narrow
                   band filter will be shifted.

            USER:  String of the ASCII file containing the transmission filter,
                   must be in the same format as the PINGSoft filters.

       /SUBTRACT:  Subtracts the adjacent continuum at +/- 100 Ang, works only with
                   Halpha filters (1,2)

             /PS:  Writes an encapsulated Postscript file of the 2D convolved image.
```

The following table shows the transmission filters included as default in PINGSoft, the table includes the filters name, source, central wavelength and internal integer values associated to each filter, the individual files can be found in the `$PINGSOFT_PATH/filters` directory. Fig. 10 show a diagram of these filters.

```
 Available PINGSoft FILTERS:

   1: Halpha KPNO-NOAO   | 6547 AA  80-FWHM       11: B-Bessell (1990)   | 4520 AA
   2: Halpha CTIO-SINGS  | 6586 AA  20-FWHM       12: V-Bessell          | 5524 AA
                                                  13: R-Bessell          | 6535 AA
   3: B-Johnson (1965)   | 4400 AA
   4: V-Johnson          | 5500 AA                14: B-CTIO-SINGS       | 4350 AA
   5: R-Johnson          | 7000 AA                15: V-CTIO-SINGS       | 5550 AA
                                                  16: R-CTIO-SINGS       | 6500 AA
   6: u-SDSS-III         | 3551 AA
   7: g-SDSS-III         | 4686 AA                17: B-KPNO-Harris      | 4357 AA
   8: r-SDSS-III         | 6166 AA                18: V-KPNO-Harris      | 5375 AA
   9: i-SDSS-III         | 7480 AA                19: R-KPNO-Harris      | 6425 AA
  10: z-SDSS-III         | 8932 AA
```

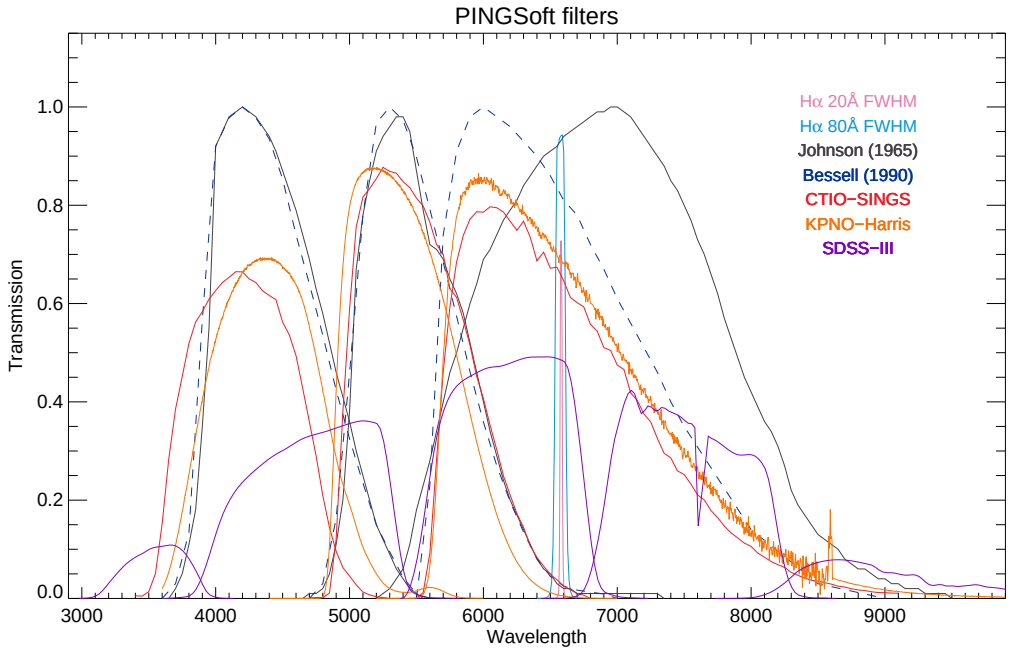

Figure 10: Default transmission filters included in PINGSoft, individual files can be found in the `$PINGSOFT_PATH/filters` directory.

Examples:

Obtain a V-band 2D FITS image of `NGC5947.rscube.fits`, using a Bessell (1990) filter, but shifted to 6400 Å, and applying a 2% clipping scaling to a Postscript output:

```
extract_filter, 'NGC5947.rscube.fits', 12, band=6400, prefix='NGC5947_V', /clip, /ps
```

Obtain a Hα narrow band image of `ngc4625.rscube.fits` using the CTIO-SINGS filter (20Å FWHM) with continuum subtracted at $\pm$100 Å using the `/SUBTRACT` option, and saving an output structure `out`:

```
extract_filter, 'ngc4625.rscube.fits', 2, out, /subtract
```

vfield_3D

Calculates the intrinsic velocity field in 3D data using a wavelength cross-correlation. Taking a user's given reference spaxel and a spectral window containing an emission line within that spectrum, the command applies a cross-correlation in wavelength with respect to the reference template and corrects the IFS data by the calculated shift in lambda. The spaxel reference entry can be either an integer number with the ID value of the reference spectrum (in IDL format) or a two element vector [x,y], with the pixel coordinates (offsets) of the reference spectrum (only for 3D cubes). The procedure writes a velocity-field corrected data file, a 2D image velocity map[4], and shows on screen a histogram of the velocity shifts with respect to the reference spectrum (in km/s). Additional information is printed on the terminal. If the `/PS` keyword is set, the program writes an encapsulated Postscript file of the velocity shift histogram. All variables can be recovered within a written binary file and an optional anonymous structure.

WARNING: The program has been tested using only strong emission lines as spectral templates, expect strange behaviour with absorption or weak features. Note that for AGN/LINER kind of spectra, the program can be confused if [N II] emission is of the same intensity (or greater) than Hα.

[4] Only if the input FITS file is a 3D cube.

CALLING SEQUENCE:

```
       vfield_3D, 'OBJECT.fits', spaxel_ref, eline_ref [, OUT.str, VMAX=vmax, /PS,
                  LMIN=lmin, LMAX=lmax, CONT1=cont1, CONT2=cont2, /FORCE_FIT
                  PREFIX=prefix, PT=pt, EXTENSION=extension ]

   'OBJECT.fits':  String of the wavelength calibrated RSS or 3Dcube FITS file

      spaxel_ref:  Either, an integer number with the ID value of the reference spectrum
                   (as seen by VIEW_3D, in IDL format) or a two element vector [x,y],
                   with the pixel coordinates of the reference spectrum (only for 3D cubes).

       eline_ref:  Wavelength guess value of the emission line to use in order to perform
                   the cross-correlation, e.g. Halpha 6563*(1+z)

 OPTIONAL KEYWORDS

         OUT.str:  Name of the optional output structure (including OUT.vfield,
                   i.e. the shift applied to the spectra with respect to the reference spectrum).

            VMAX:  Maximum velocity (km/s) with respect to the reference spectrum for which
                   a wavelength shift is applied. Spectra with larger values are not shifted.
                   Default: 500 km/s

        LMIN/MAX:  min/max wavelength for the cross-correlation, default: eline +/- 100 Ang

         CONT1/2:  Continuum bands used to normalize the spectra, default: eline +/- 80 Ang

             /PS:  Writes an encapsulated Postscript file of the wavelength shift histrogram

      /FORCE_FIT:  Forces a Gaussian fit to each spectra before calculating the cross-correlation
                   (use with care, slower and somewhat inefficient)
```

Example:

Let's consider the VLT-VIMOS observation of the LIRG IRAS F06295-1735: the user can note the intrinsic velocity field of the object by moving the mouse cursor over the FoV and noticing the shift of the Hα emission line across the field with respect to the reference vertical line (with the aid of the following command):

```
view_3d, 'IRAS06295.rscube.fits', /clip, lmin=6650, lmax=6750, band=6700, filter=2, vline=6702
```

We calculate and correct for this shift by applying a cross-correlation in wavelength, using as a reference the spaxel [15,15], and the Hα emission line at the observed wavelength 6700 Å:

```
vfield_3d, 'IRAS06295.rscube.fits',  [15,15], 6700, out, /ps

 PINGSoft:  vfield_3D
 ============================

              Reference spaxel:  [15,15]

           Mean velocity shift:  -20.1318

          Sigma velocity shift:  100.093

 Velocity field corrected FITS:  IRAS06295_vfield.fits

             Velocity map FITS:  IRAS06295_vel.fits

             Binary file saved:  IRAS06295_vfield.sav

             Histogram PS file:  IRAS06295_vfield.eps
```

The procedure writes the velocity-field corrected cube `IRAS06295_vfield.fits`, the histogram of the velocity shifts with respect to the reference spectrum (in km/s) and a Hα velocity map of the object as shown in Fig. 11.

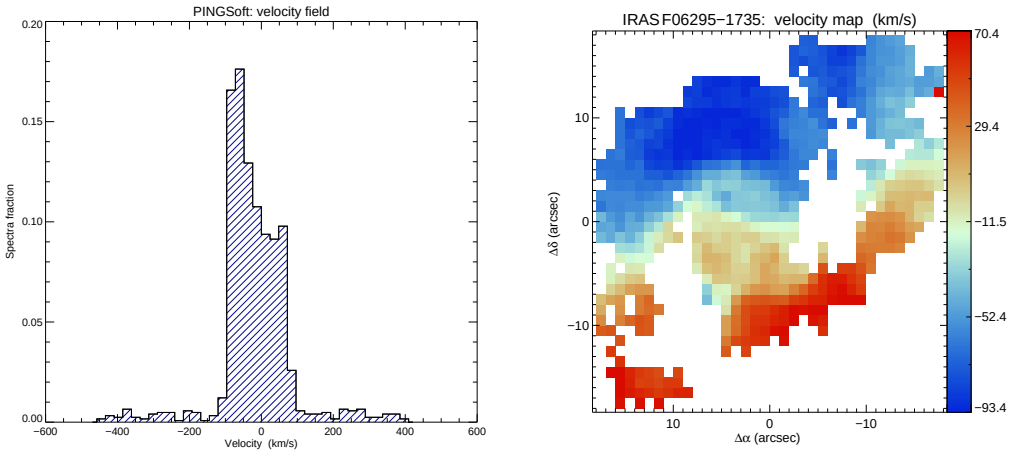

Figure 11: Velocity field histogram and Hα velocity map of IRAS F06295-1735, both derived with the `vfield_3D` command.

The user can verify that all the spectra is aligned to the reference spectrum zero-velocity in velocity-field corrected file:

```
view_3d, 'IRAS06295_vfield.fits', /clip, lmin=6650, lmax=6750, band=6700, filter=2, vline=6702
```

The output structure contains the values `out.VFIELD` and `out.MAPV` corresponding to the velocity field and velocity map respectively.

s2n_ratio_3D

Calculates the continuum and emission-line S/N of the IFS data, and allows to interactively extracts spectra based on a S/N floor. The program calculates the S/N ratio on a pre-defined continuum and emission-line region per each spaxel. The user has to provide: 1) the central (observed) wavelength of a (featureless) continuum band (used to calculate the $(S/N)_{cont}$), and 2) a central (observed) wavelength of an emission-line feature band (used to calculate the $(S/N)_{eline}$).

When launched, the program displays the integrated spectrum of the input IFS data, and shows the continuum and emission-line spectral regions defined by the input parameters (see Fig. 12); in the emission-line case, the graph also displays the pseudo-continuum adjacent bands used to calculate $(S/N)_{eline}$. If the default values does not satisfy the user, the band widths and central wavelengths of the pseudo-continuum adjacent bands can be set with the `WCONT`, `WIDTH` and `PSEUDO` keywords (see below). The command prompts for confirmation, if the current ranges are accepted, the program calculates the S/N for each case and displays a new window with four panels (see Fig. 12): the left-column shows the S/N as a function of spaxel position for the *continuum* (top) and the *emission-line* (bottom), on each case, a red-horizontal line marks a guess 1-σ S/N threshold, spaxels above that threshold are displayed in the FoV panels on the right-column. At this point, the user can interactively change the spaxel threshold and/or the *continuum* S/N calculation (see below), the program prompts for new threshold input values and updates the corresponding window.

If the user accepts and proceeds with the extraction, a whole series of files are created, including a RSS file with the spaxels above the S/N threshold, the integrated spectrum of the selected spaxels (in ASCII, FITS and EPS format), a table with the calculated signal, noise, S/N ratio and IDL index for each selected spaxel, a S/N map an extraction mask (0/1 values) in FITS format, for the both the continuum and emission-line cases. Additionally, the final selected spaxels and S/N maps are shown on screen.

S/N calculation: The S/N ratio is calculated using the function `s2n_rat_fun`. In the case of the emission-line, the value is calculated following the prescriptions in Sec. 2 of Rosales-Ortega et al. (2012).

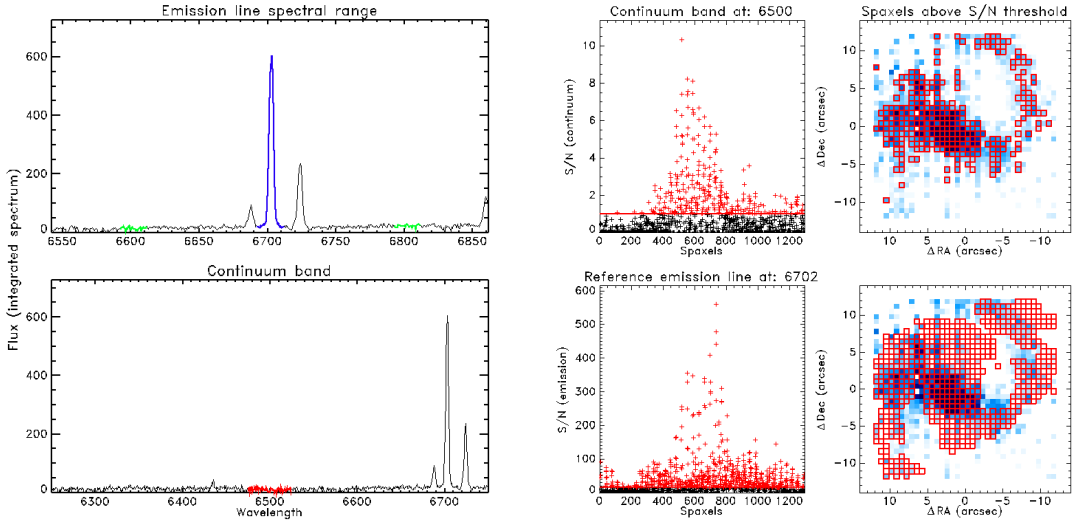

Figure 12: *Left*: First displayed window of the `s2n_ratio_3D` command. showing the integrated spectrum of the input IFS data, and the continuum and emission-line spectral regions defined by the input parameters. *Right*: Second displayed window showing the S/N as a function of spaxel position for the continuum (top) and emission-line (bottom) cases.

In the case of the continuum, we use a *standard* definition , i.e. the ratio of the average flux level value (signal) to the standard deviation of the signal (noise), any systematic slope in the spectral window is corrected by fitting a linear model to detrend the data before calculating the `stddev`. This "working" S/N definition assumes that the standard deviation is dominated by noise instead of real spectral features, and therefore it is important in practice to select a "clean" spectral region for its calculation. However, the user can select (either interactively or *a priori*) two additional ways of calculating the S/N in the continuum case:

1. `/SQRTN`, the continuum noise is equal to `sqrt[stddev(flux)]`, i.e. the squared-root of a detrended standard deviation of the signal. This definition may result useful when the noise is strongly non Poissonian, if the structure of the signal in the spaxels is not optimally weighted and/or if there are strong gradients in the S/N.

2. `/PROXY`, in this case the signal is calculated as the sum of the flux (instead of the mean) and the noise is equal to `sqrt(signal)`. This definition represents a "density" S/N, and follows the surface brightness of the object at the chosen spectral band.

These two options are NOT formal definitions of S/N and should NOT be used as so, but instead as *proxies* of the signal in the IFS data which can be useful for extracting relevant regions.

Interactivity: The interactivity can be avoided by using: 1) the `/SKIP` parameter, which skips the wavelength range confirmation and prints minimum output information on-screen; and 2) by setting the `S2N_MIN` keyword, which is two-entries vector `[s2n_cont,s2n_eline]` containing the S/N threshold values, when this parameter is set, the script extracts automatically those spaxels above the input values for both samples without further interaction.

Calling sequence:

```
s2n_ratio_3d, 'OBJECT.fits', lam_cont, lam_eline  [, OUT.str, S2N_MIN=[s2n_cont,s2n_eline],
              WIDTH=width, WCONT=wcont, PSEUDO=[lam_pseudo1,lam_pseudo2], /SQRTN, /PROXY, /SKIP,
              PREFIX=prefix, PT='Postable.txt', EXTENSION=extension,  _EXTRA={extra},
              (+ all graphic options as VIEW_3D) ]

 'OBJECT.fits':  String of the wavelength calibrated RSS or 3Dcube FITS file

      lam_cont:  Central (observed) wavelength of the (featureless) continuum band
                 to calculate the (S/N)_cont
```

```
      lam_eline:  Central (observed) wavelength of the emission line feature band
                  to calculate the (S/N)_eline

 OPTIONAL KEYWORDS

         OUT.str:  Name of the optional output structure.

         S2N_MIN:  A two-entries vector [s2n_cont,s2n_eline] containing the S/N threshold values.

                   NOTE: when this parameter is included, the script extracts automatically those
                         spaxels above the input values for both samples without interaction.

           WIDTH:  Width (in lambda units) of the spectral regions (elines and pseudo-continuum bands)
                   from where the (S/N)_eline is obtained. Default: 50 lamba units.

           WCONT:  Width (in lambda units) of the continuum band from where the (S/N)_cont is obtained
                   Default: 50 units.

          PSEUDO:  A two-entries vector [lam_pseudo1,lam_pseudo2] containing the central wavelenghts
                   of the pseudo-continuum adjacent bands used to calculate (S/N)_eline.
                   Default: [lam_eline-100,lam_eline+100]

          /SQRTN:  When this keyword is set, the continuum noise is equal to sqrt(sigma). This may be useful
                   when the noise is strongly non Poissonian, if the structure of the signal in the spaxels
                   is not optimally weighted and/or if there are strong gradients in the S/N.

          /PROXY:  When this keyword is set, the signal is calculated as the sum of the flux
                   (instead of the mean) and the noise is equal to sqrt(signal). This definition
                   represents a "density" S/N, and follows the surface brightness of the object at the
                   chosen spectral band.

           /SKIP:  Skips wavelength range confirmation and prints minimum output information on-screen
```

Outputs:

```
 PINGSoft:  S/N ratio extraction
 ================================================

 Files written:

     CONTINUUM sample
     -------------------

          Extracted RSS:   PREFIX.cont_rss.fits
         Position table:   PREFIX.cont_rss.pt.txt
         FoV Postscript:   PREFIX.cont_FoV.eps
        Integrated FITS:   PREFIX.cont_integ.fits
                  ASCII:   PREFIX.cont_integ.txt
             Postscript:   PREFIX.cont_integ.eps
        S/N table ASCII:   PREFIX.cont_s2n.txt
           S/N map FITS:   PREFIX.cont_s2n.fits
        Extraction mask:   PREFIX.cont_mask.fits

     EMISSION sample
     -------------------

          Extracted RSS:   PREFIX.eline_rss.fits
         Position table:   PREFIX.eline_rss.pt.txt
         FoV Postscript:   PREFIX.eline_FoV.eps
        Integrated FITS:   PREFIX.eline_integ.fits
                  ASCII:   PREFIX.eline_integ.txt
             Postscript:   PREFIX.eline_integ.eps
        S/N table ASCII:   PREFIX.eline_s2n.txt
           S/N map FITS:   PREFIX.eline_s2n.fits
        Extraction mask:   PREFIX.eline_mask.fits

       Output structure:   PREFIX.str.sav
        Plots PNG image:   PREFIX.plots.png

      If not PREFIX set, then PREFIX='s2n_ratio'
```

Examples:

1. We want to calculate the S/N for the galaxy IRAS F06295-1735 in the "featureless" continuum band centered at 6500 Å and width 50 Å, and on the Hα emission line, at the approximate observed wavelength 6702 Å and width 20 Å (see left-panel of Fig. 12):

```
s2n_ratio_3d, 'IRAS06295.rscube.fits', 6500, 6702, width=20, wcont=50, /clip
```

When the program prompts for new S/N thresholds, we choose S/N=1 for the continuum case (standard definition) and S/N=15 for the emission-line case, and proceed with the extraction (right-panel Fig. 12). The program writes a whole set of data products, in particular, the `s2n_ratio.eline_mask.fits` was applied to the velocity field 2D FITS image obtained in the example of `vfield_3D` (`IRAS06295_vel.fits`) in order to mask and produce the right-panel of Fig. 11.

2. Fig. 13 shows the difference between the "conventional" S/N definition and the *proxy* to S/N which can be used with `s2n_ratio_3D`. The top row shows the S/N measured in the conventional manner (mean/stddev) for the galaxy NGC 7466, using the spectral window centered at 5725 Å, with a width of 100 Å, the red symbols corresponds to those above a S/N threshold of 5, which cover a substantial fraction of the area of the galaxy. The bottom row shows the same S/N calculation, but using the S/N "proxy" as defined above, in this case the S/N threshold is 1. Note that the S/N "proxy" seems to better follow the morphology of the targets above a given S/N threshold (i.e. the signal surface brightness, which is important for *a-posteriori* Voronoi binning, see below). Although this trick improves S/N, the user must bear in mind that it is NOT a true S/N-based calculation.

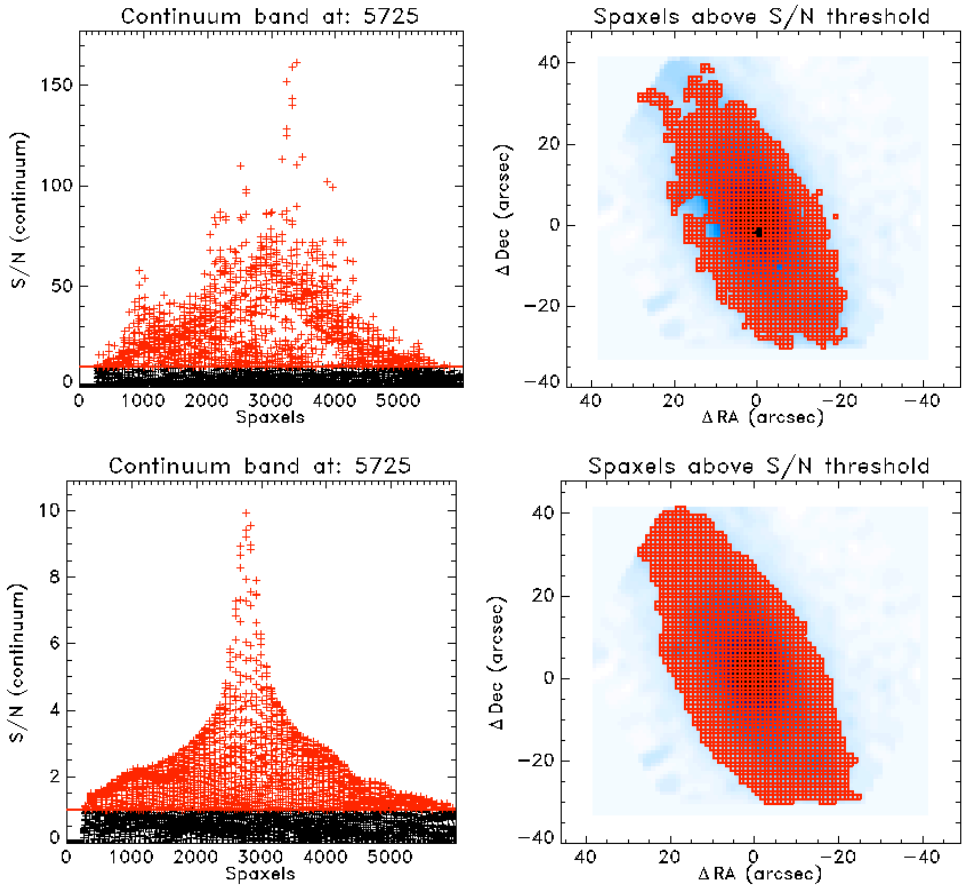

Figure 13: *Top*: S/N per spaxel in the galaxy NGC 7466 calculated using the "conventional" definition. *Bottom*: Same as above, but using the `/PROXY` keyword as explained in the text.

voronoi_3D

Applies the Voronoi tessellation method to bin the IFS data to a target signal-to-noise ratio per bin. This program is a wrapper of the `voronoi_2D_binning` code by Cappellari & Copin (2003) which performs an adaptive spatial binning of IFS data. The Voronoi Binning method optimally solves the problem of preserving the maximum spatial resolution of general two-dimensional data, given a constraint on the minimum S/N ratio. In order to work properly, this routine needs the `voronoi_2D_binning` code to be in the `$IDL_PATH`. The source code can be found at:

http://www-astro.physics.ox.ac.uk/~mxc/idl/#binning

In addition, this program MUST BE used with the output of `s2n_ratio_3D` (or use a similar input format).

The `voronoi_2D_binning` code requires as an input the spatial position (x,y), the signal and noise (as separate values) in order to perform the binning, note that the code does not care what is the proxy used to calculate the "signal" or "noise". The signal and noise can be calculated by `s2n_ratio_3D` for all the spaxels within a IFS data file (as described above, see example below), and the information can be passed as input for this Voronoi binning script (either considering all the spaxels or only those above a certain threshold, using any of the available S/N "proxies"). In the case of `s2n_ratio_3D`, the outputs used by `voronoi_3D` can be either the "`S/N table ASCII`" (passed through the `ASCII_S2N` keyword) or the output IDL structure (passed through the `STR_S2N` keyword, see below), either for the continuum or the emission-line case.

The most important input parameter for the Voronoi binning is the "target S/N", i.e. the minimum required S/N that each bin must posses after the tessellation. Therefore, the main `voronoi_3D` inputs are: 1) the input IFS data file, 2) the S/N information, and 3) the target S/N. The program then invokes the `voronoi_2D_binning` script, prints the binning/iterations information on the terminal and displays a window which shows on the top-panel the FoV of the spatial binning and on the bottom the S/N-per-bin as a function of radius (see Fig. 14). At this point, the S/N-binning target can be changed interactively or the user can proceed with the extraction. In the latter case, the script writes a set of output files and displays on the screen the FoV of the input IFS data overlaid with the binning tessellation (where all visual options of `view_3D` can be applied). The output files include: a) a RSS file with the integrated spectra of each Voronoi bin (the coordinates of the associated Pseudo-Position table correspond to the centre-of-mass of the bin); b) a 2D FITS segmentation map, corresponding to the Voronoi bins; and 3) a full segmented 3D cube. An optional Postscript file of the FoV tessellation can also be included.

Important note on S/N and Voronoi binning:

DO NOT TRUST the S/N-per-bin reported by the `voronoi_2D_binning` code (specially if the input signal/noise was derived using the "conventional" S/N definition). Several tests have shown that the *real* S/N goes as the signal-to-ratio value reported by the code per bin, divided by the squared-root of the number of pixels in the bin: $\mathrm{S/N} = (\mathrm{S/N}_{\mathrm{Capellari-bin}}) \ / \ (\sqrt{N}_{\mathrm{pixels-per-bin}})$. The best strategy is to recalculate the S/N of the output Voronoi-binned spectra (e.g. using `s2n_ratio_3D`), update the target S/N in `voronoi_3D` and proceed iteratively until the "real" S/N-per-bin corresponds to the user's target S/N.

On the other hand, either for a S/N-floor extraction, or spatially adaptive-binning, we strongly suggest to use a proxy that goes with the inverse of the square of the signal in order to trace the morphology and extract the relevant regions (note that the noise $\sim$ sqrt(signal) recipe implies in practice that binning will proceed till we reach a certain flux level). Use this information as input for the Cappellari's code and then recalculate the S/N of the output, Voronoi-binned spectra using the "conventional" S/N definition if you want to report S/N-per-bin in your data (see examples below).

Calling sequence:

```
voronoi_3D, 'OBJECT.fits', S2N_target, ( ASCII_S2N='ascii_s2n' -OR- STR_S2N=str_s2n),
            [ /NORMALIZE, /SEGMENT, /PS, PREFIX=prefix, EXTENSION=extension, PT='PosTable.txt',
            /NO_CVT, /WVT, PIXSIZE=pixelsize, (voronoi_2d_binning.pro options)
            (+ all display options as in VIEW_3D) ]
```

```
    'OBJECT.fits':  String of the wavelength calibrated 3D cube or RSS FITS file.

       S2N_target:  Minimum required S/N that each bin must posses after the tessellation

and EITHER:

        ASCII_S2N:  String with the name of the "S/N table ASCII" output of S2N_RATIO_3D
                    (or a table with a similar format) containing the spatial position (x,y),
                    the signal and noise (as separate values) in order to perform the bining.
OR

          STR_S2N:  Variable of the output IDL structure of S2N_RATIO_3D with the same input
                    information as described above.

OPTIONAL KEYWORDS

       /NORMALIZE:  If set, the integrated spectrum is normalized by the number
                    of spaxels within the Voronoi bin.

         /SEGMENT:  If set, the on-screen or Postscript visualisation of the
                    segmented regions uses a special, nicer color-coding

             /PS:  Writes an encapsulated Postscript file of the segmented regions

VORONOI_2D_BINNING options:

          /NO_CVT:  Set this keyword to skip the Centroidal Voronoi Tessellation
                    (CVT) step (vii) of the algorithm in Section 5.1 of Cappellari & Copin (2003).

            /WVT:  When this keyword is set, the routine bin2d_cvt_equal_mass is
                    modified as proposed by Diehl & Statler (2006, MNRAS, 368, 497).

          PIXSIZE:  Optional pixel scale of the input data.
```

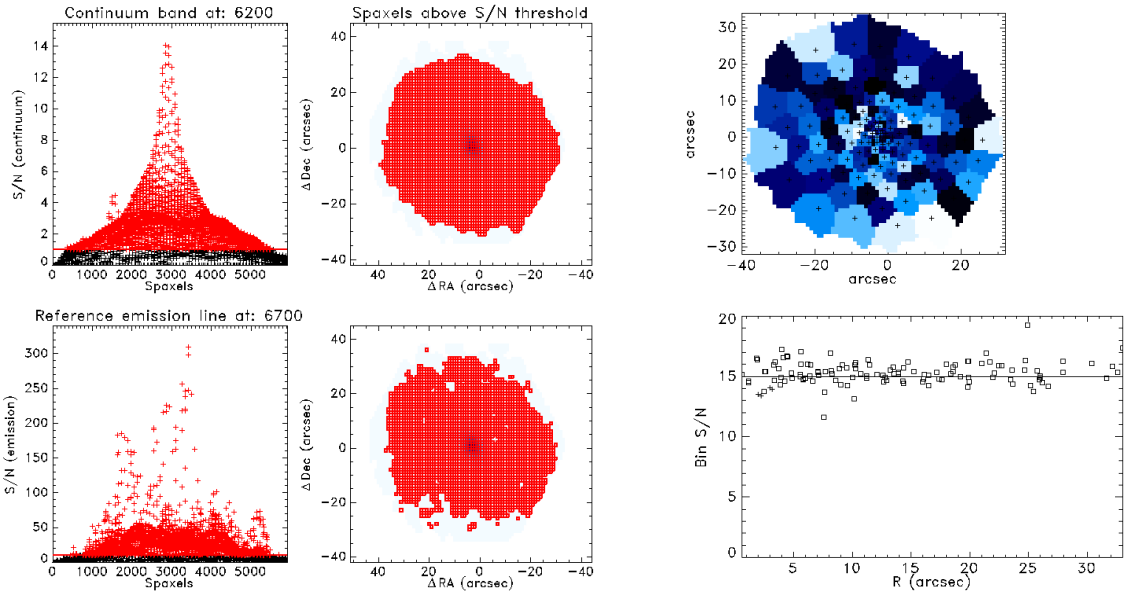

Figure 14: On the left-panel, S/N per spaxel for the IFS cube of NGC 5947 calculated using the `s2n_ratio_3D` and the `/PROXY` keyword, the output information corresponding to the *continuum* case with a S/N threshold of 1 is passed as input to the `voronoi_3D` command, which generates a Voronoi binning with a target S/N of 15 (right-panel).

Outputs:

```
If PREFIX is set:

          Segmentation map:  PREFIX_voronoi.seg.fits
  Extracted spectra in RSS:  PREFIX_voronoi.rss.fits
     Pseudo-position table:  PREFIX_voronoi.rss.pt.txt
    Full segmented 3D cube:  PREFIX_voronoi.cube.fits
  Postscript segmented FoV:  PREFIX_voronoi.FoV.eps      (only if /PS set)

   If PREFIX is not set, PREFIX=OBJECT
```

Examples:

1. We want to apply the Voronoi binning tessellation to the IFS cube `NGC5947.rscube.fits` using a featureless continuum band. For doing so, we calculate the S/N-per-spaxel at 6200 Å within a 200 Å width band, using the `/PROXY` option of the `s2n_ratio_3D` routine (left-panel of Fig. 14):

```
s2n_ratio_3d, 'NGC5947.rscube.fits', 6200, 6700, wcont=200, /skip, s2n=[1,10], /proxy
```

In this example, we only consider spaxels with S/N-proxy in the continuum greater-equal than 1. We then apply the `voronoi_3D` script using the "`S/N table ASCII`" output of `s2n_ratio_3D` as input information, setting a target S/N of 15. Fig. 15 shows the output, full-segmented Voronoi-binned cube `NGC5947_voronoi.cube.fits` as seen by `view_ifs`:

```
voronoi_3d, 'NGC5947.rscube.fits', 15, ASCII_S2N='s2n_ratio.cont_s2n.txt', /segment
```

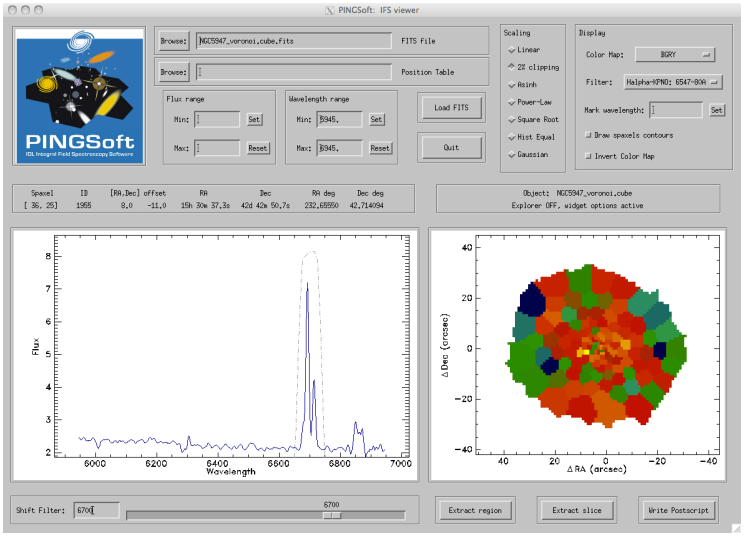

Figure 15: Widget visualisation of the output Voronoi-binned cube described above.

2. This example shows the difference between the S/N-per-bin calculated by the `voronoi_2D_binning` script and the *real* S/N as derived by a conventional definition. The top-panels of Fig. 16 shows the S/N-per-spaxel of the CALIFA galaxy UGC 12054 calculated on a continuum band, the standard definition of S/N was used (mean/stddev), with a threshold of S/N=5. Therefore, if we want to perform a Voronoi binning the target-S/N should be above this value.

We then apply the `voronoi_2D_binning` code with a target S/N of 30, a typical value for a good spectrum to apply e.g. stellar synthesis analysis. The bottom-panels of Fig. 16 shows the Voronoi binning and the output statistics. First, we can note that the bins are very small and are practically the same size as the original spaxels. The S/N vs. radius plot shows that the majority of the spaxels are around S/N ∼30 with a certain dispersion, plus some outliers with higher S/N, as expected.

We then calculate the S/N of the output binned spectra using exactly the same definition and parameters as before. Fig. 17 shows the S/N calculation for all the bins within the new Voronoi cube. By

definition, as the target S/N was set to 30, all the bins should be around or above that value (as shown in the bottom-panel of Fig. 16). However, we notice that only the central bins are above the S/N target (in red) and that the S/N of most of the bins are below the target value. In order to reach a standard S/N ∼30 from the binned spectra, we should use an input target-S/N in the `voronoi_2D_binning` script of ∼100.

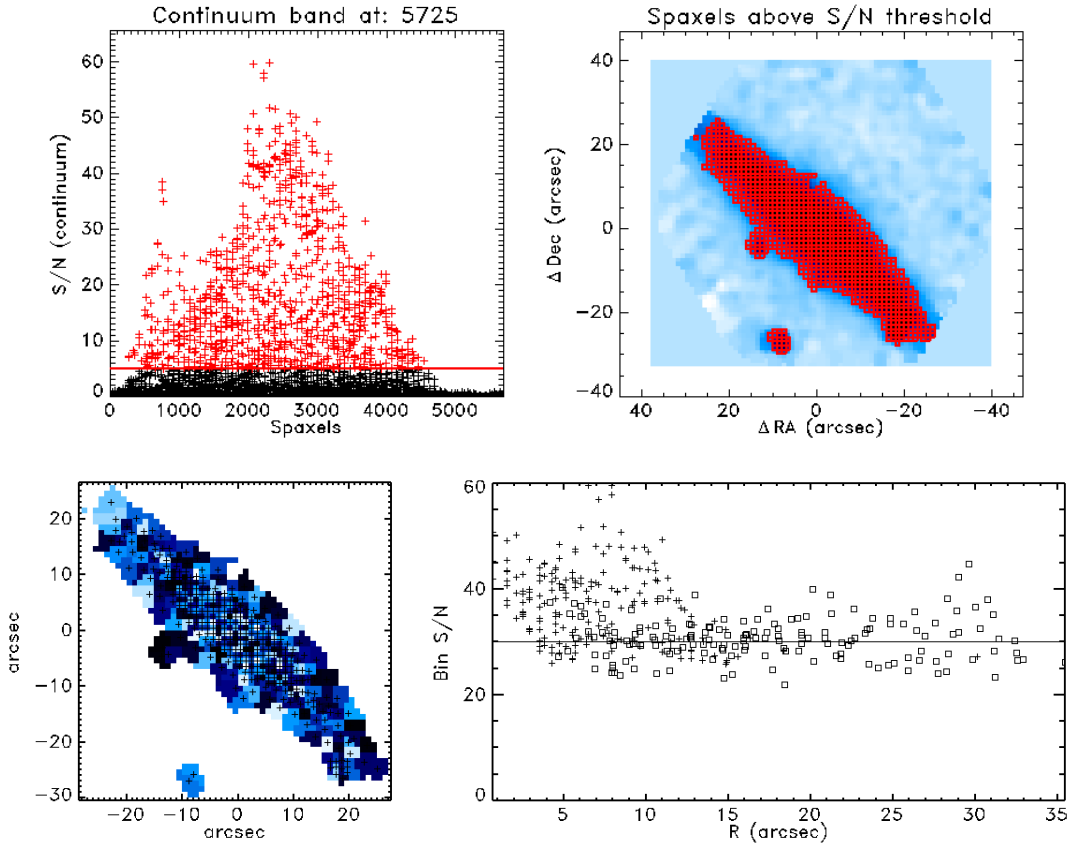

Figure 16: *Top*: S/N-per-spaxel of the CALIFA galaxy UGC 12054 using the conventional S/N definition on the continuum. *Bottom*: Voronoi binning of the same object using a target-S/N of 30.

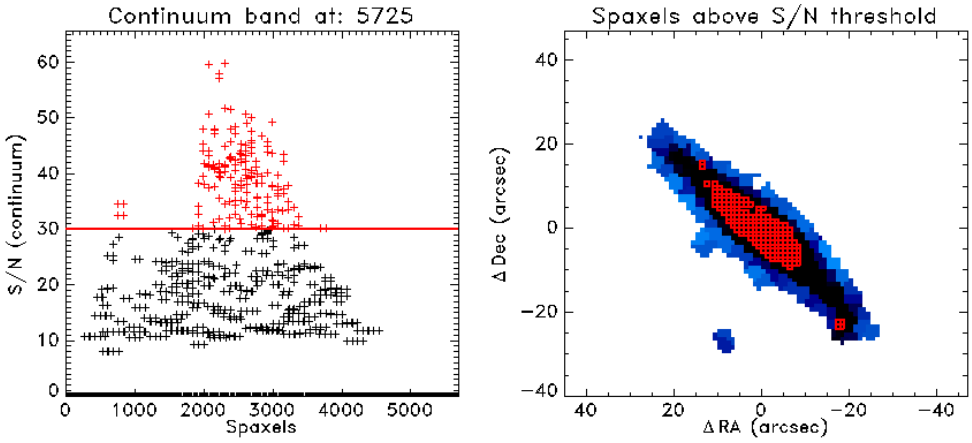

Figure 17: S/N-per-spaxel of the Voronoi-binned cubed of UGC 12054 using the same definitions and parameters as before. Note that only a small fraction of the bins have a *real* S/N equal to the target one.

s2n_optimize

Extracts spectra interactively based on a S/N continuum and emission line optimization method by Rosales-Ortega et al. (2012) using cumulative integrated spectra.

The optimization method is performed by first calculating the $(S/N)_{continuum}$ and $(S/N)_{emission}$ for each of the spaxels (preferably in a velocity-corrected 3D cube) following the same prescriptions as in `s2n_ratio_3D`. The spaxels are then sorted into two groups (continuum and emission-line), one in decreasing order of the $(S/N)_c$ value and the other in decreasing order of the $(S/N)_{em}$. The script creates an integrated spectrum for each of the spaxels subsample by adding the spectra in the sorted order as described before. In each step, the procedure calculates the S/N in the continuum and/or emission line feature in the new generated spectrum. The rationale of this criterion is the following: spectra of good quality (high S/N) should help to enhance the S/N of the integrated spectrum in each of the two groups. However, if the inclusion of a new spectrum (spectra) decreases the S/N of the integrated spectrum with respect to the values found in previous steps, then the individual S/N of the spectrum (spectra) for which this turn-off is found marks the tentative threshold value (or range) sought for high quality spectra within an IFS cube.

When launched, the program displays the integrated spectrum of the input IFS data, and shows the continuum and emission-line spectral regions defined by the input parameters; in the emission-line case, the graph also displays the pseudo-continuum adjacent bands used to calculate $(S/N)_{em}$. If the default values does not satisfy the user, the band widths and central wavelenghts of the pseudo-continuum adjacent bands can be set with the `WCONT`, `WIDTH` and `PSEUDO` keywords. The command prompts for confirmation, if the current ranges are accepted, the program calculates the S/N for each case and displays a new window with six panels: The left-column shows the "statistical" S/N of the continuum (top) and emission (bottom) features sorted in decreasing order as a function of spaxel position. The blue vertical-line shows the position at which S/N=1. The middle-column panels show the cumulative S/N of the integrated spectra in the same sorted order as the left-panels. The red vertical-line shows the position of the maximum S/N. Spaxels to the left of this threshold are displayed in the FoV panels on the right-column. Statistical information is printed on the terminal. At this point, the user can interactively change the spaxel threshold and/or the continuum S/N calculation (as explained in in `s2n_ratio_3D`), the program prompts for new threshold input values and updates the corresponding window. If the user accepts and proceeds with the extraction, a whole series of files are created with the same format as in `s2n_ratio_3D`. The final selected spaxels and S/N maps are shown on screen.

Calling sequence:

```
s2n_optimize, 'OBJECT.fits', lam_cont, lam_eline  [, OUT.str, WIDTH=width, WCONT=wcont,
              PSEUDO=[lam_pseudo1,lam_pseudo2], /SQRTN, /PROXY, /SKIP,
              PREFIX=prefix, PT='Postable.txt', EXTENSION=extension,
             (+ all graphic options as VIEW_3D) ]

'OBJECT.fits':  String of the wavelength calibrated RSS or 3Dcube FITS file

    lam_cont:  Central (observed) wavelength of the (featureless) continuum band
               to calculate the (S/N)_cont

   lam_eline:  Central (observed) wavelength of the emission line feature band
               to calculate the (S/N)_eline

   + Similar keywords as S2N_RATIO_3D
```

Examples:

Using the same example as for the `s2n_ratio_3D` case, we now use the S/N optimization method to extract integrated spectra based on a continuum band at 6200 Å and on the Hα emission line:

```
s2n_optimization, 'NGC5947.rscube.fits', 6200, 6700
```

IFS manipulation

split_califa

Extracts the FITS extensions of the CALIFA survey data, writing individual files with the appropriate headers for 3D visualisation.

CALLING SEQUENCE:

```
split_califa, 'OBJECT.fits'  [, PREFIX='prefix']
```

Outputs:

```
              Data:  PREFIX.DATA.fits
            Errors:  PREFIX.ERROR.fits
        Bad pixels:  PREFIX.BADPIX.fits
    Error scaling:  PREFIX.ERRWEIGHT.fits  (only 3D cubes)
Position table:  PREFIX.POSTABLE.fits  (only RSS files, binary table)
```

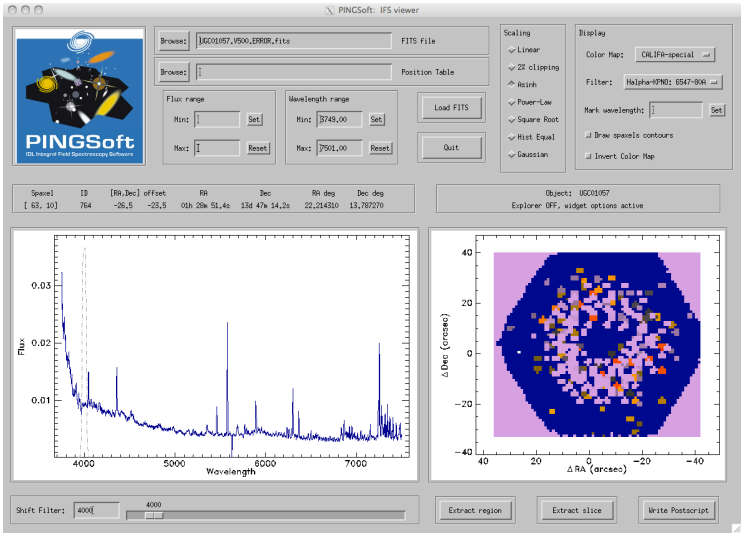

Figure 18: `ERROR` FITS extension cube of the CALIFA galaxy UGC 1057 extracted with `split_califa` as seen by `view_ifs`.

cube2rss

Generates a RSS file and a associated position table from a 3D cube FITS file. The script transforms a 3D cube FITS file into a Row-Stacked-Spectra FITS file and generates the (continuous) position table of the RSS according to the original geometry of the cube. The size, resolution and reference pixel of the Field-of-View are deduced from the 3D FITS header (e.g. `NAXIS1, CDELT1, CRPIX1`, etc.). The geometry which is assumed defines the reference pixel (1,1) at the bottom-right corner of the X-Y projection, and the (N,M) pixel at the top-left of the image.

Supported geometries include:

1. The pixel index increases horizontally, from right to left on the same row (e.g. PPAK, CALIFA).
2. The index increases vertically, from bottom to top on the same column (e.g. VIMOS).

The default is set to 1, other geometries can be easily incorporated. If the FITS header includes a reference pixel and WCS, this information is considered in order to create the associated position table (in the `view_3D` format). If the /CALIFA keyword is set, the program needs only the name of the input 'OBJECT.rscube.fits' file in order to generates the RSS FITS with all the appropriate parameters and the position table in the standard North-East configuration.

Calling sequence:

```
    cube2rss, 'input_3Dcube.fits' [, GEOMETRY=geometry, PREFIX='prefix', TRIM_RED=trim_red,
                                     /PPAK, /SILENT ]

    'input_3Dcube.fits':  3D cube FITS input file.

 OPTIONAL KEYWORDS:

          TRIM_RED:  Number of pixels to be removed at the red end of each
                     spectrum (useful for cosmetic purposes).

             /PPAK:  If set, creates the position table for the PPAK instrument in
                     the right standard North-East orientation.
```

Miscellaneous functions

This section contains some useful functions that can be used as stand-alone programs for general purposes in astronomy and IFS. Some of them are subroutines of main PINGSoft programs.

`redshift_3D`

Returns the redshift of the input IFS data measured from the integrated spectra.

Calling sequence:

```
  result = redshift_3D( 'OBJECT', [ lam_rest,  WIDTH=width, LAM_Z=lam_z, /SILENT ])

   'OBJECT.fits':  String of the IFS 3D cube, RSS FITS or 1-D ASCII file

           WIDTH:  Width of the spectral band centered at lam_rest where
                   the reference emission-line will be sought for the
                   redshift determination.

           LAM_Z:  output, lam_rest at measured redshift.
```

Defaults: `lam_rest` = 6563 Å (i.e. Hα), `WIDTH` = 100 Å. Note potential problems if the intensity of [N II] is approximate or higher than Hα (e.g. LINERS/AGNs).

`offset2radec`

Transforms small angle offsets in arcsec from a reference point to equatorial coordinates. The value of Right Ascension is an approximation, valid within 1 degree.

Calling sequence:

```
    result = offset2radec(RA_ref, Dec_ref, X_off, Y_off, /SEXAG)

     RA_ref/Dec_ref:  coordinates of the reference point in DEGREES

        X_off/Y_off:  offset from the reference point in ARCSEC

             /SEXAG:  Output in sexagesimal units (default: degrees).

     OUTPUT: a vector with 2 numbers containing the coordinates in DEGREES:
```

```
            result = [ RA, Dec ]

            if the /SEXAG parameter is set, then a vector with 6 numbers:

            result = [ RA_hr,RA_min,RA_sec, Dec_deg,Dec_min,Dec_sec ]
```

Example:

Calculate the equatorial coordinates of an offset $(\Delta\mathrm{RA}, \Delta\mathrm{Dec}) = (15, 23)$ in arcsec from the reference point (0,0) with coordinates RA=23.501865, Dec=15.528838:

```
offset = offset2radec( 23.501865, 15.528838, 15, 23)
```

radec2offset

Transforms equatorial coordinates to small angle offsets from a given reference point.

Calling sequence:

```
  result = radec2offset(RA_ref, Dec_ref, RA_off, Dec_off, /PLOT)

      RA_ref/Dec_ref:  Coordinates of the reference point

      RA_off/Dec_off:  Coordinates of the offset from the reference point

                /PLOT:  Plots a diagram of the offset from the reference point

      NOTE: All coordinates must be either in DEGREES or in SEXAGESIMAL notation,
            in the latter case, each entry must be a vector with 3 entries, i.e.

              RA = [hr,min,sec]
             Dec = [deg,min,sec]

    OUTPUT: a vector with 4 numbers, containing the X,Y-offsets in ARCSEC,
            the first 2 numbers using the RA approximation for small angles
            with the right symbol for a NE configuration.
            The second pair of offsets are the exact values using the general
            formula, but probably with a wrong symbol.

            result = [ X-approx,Y-approx, X-exact,Y-exact ]
```

Example:

```
offset = radec2offset( 23.501865, 15.528838, 23.506190, 15.535227, /plot)
```

set_value2D

Replaces the values in a 2D array of an index vector found with the IDL `WHERE()` function.

Calling sequence:

```
 result = set_value2D( image, index, value )

    image:  Variable name of the 2D array where values will be replaced.

    index:  Index vector produced by the WHERE() function.

    value:  Value to be replaced within the 2D array at the index locations.

   result:  New 2D array with the values substituted.
```

Example:

```
IDL> image = congrid(bindgen(3,3),9,9)

IDL> print, image
    0   0   0   1   1   1   2   2   2
    0   0   0   1   1   1   2   2   2
    0   0   0   1   1   1   2   2   2
    3   3   3   4   4   4   5   5   5
    3   3   3   4   4   4   5   5   5
    3   3   3   4   4   4   5   5   5
    6   6   6   7   7   7   8   8   8
    6   6   6   7   7   7   8   8   8
    6   6   6   7   7   7   8   8   8

IDL> index = where(image eq 4)

IDL> new = set_value2d(image,index,9)

IDL> print, new
    0   0   0   1   1   1   2   2   2
    0   0   0   1   1   1   2   2   2
    0   0   0   1   1   1   2   2   2
    3   3   3   9   9   9   5   5   5
    3   3   3   9   9   9   5   5   5
    3   3   3   9   9   9   5   5   5
    6   6   6   7   7   7   8   8   8
    6   6   6   7   7   7   8   8   8
    6   6   6   7   7   7   8   8   8
```

set_value3D

Replaces vectors in a 3D array of an index vector found with the IDL `WHERE()` function.

Calling sequence:

```
 result = set_value3D( cube, index, vector )

     cube:  Variable name of the 3D array where vectors will be replaced

    index:  Index vector produced by the WHERE() function
            (based on a 2D slice of the cube)

   vector:  Vector to be replaced within the 3D array at the index locations.

   result:  New 3D array with the vectors substituted.
```

Example:

```
IDL> cube = bindgen(3,3,3)

IDL> print, cube
  0   1   2
  3   4   5
  6   7   8

  9  10  11
 12  13  14
 15  16  17

 18  19  20
 21  22  23
 24  25  26

; Index vector produced by the WHERE() function
; based on a 2D slice of the cube:

IDL> index = where(cube[*,*,0] gt 5)
```

```
IDL> vector = [0,0,0]

IDL> new = set_value3d(cube,index,vector)

IDL> print, new
  0   1   2
  3   4   5
  0   0   0

  9  10  11
 12  13  14
  0   0   0

 18  19  20
 21  22  23
  0   0   0
```

Retired routines

Scripts belonging to previous PINGSoft versions located at the `retired/` directory, although this routines are no longer supported, they might be useful for RSS data manipulation.

`read_rss`: Reads a RSS FITS file and stores the data into an IDL vector.

`extract_rss`: Extracts a section of consecutive rows in a RSS FITS file and creates a new one.

`merge_rss`: Merges a list of RSS files into a single RSS file.

`merge_ptable`: Concatenates a list of position table files into a single one for mosaicking purposes.

`shift_ptable*`: Shifts the reference point or applies an offset to a given position table.

`get_new_pt`: Generates a new position table based on an index of selected spaxels.

`show_hdr`: Shows on screen the header of a FITS file, which can be written to an ASCII file.

`write_hdr`: Adds or updates an entry in the header of a FITS file.

`copy_hdr`: Copies the header of one FITS file to another.

`write_wcs`: Adds or updates the WCS (World Coordinate Systems) entries in a FITS header.

* This is a subroutine of PINGSoft, found in the parent directory.

5 Additional notes

5.1 Intensity scaling

The default colour-scale of the visualisation is obtained by sampling the range of intensities within the chosen narrow band into an *ad hoc* colour dynamic range of 255 values (i.e. equal to the number of values in a given IDL colour table). However, very different ranges of intensities are expected depending of the object, spectral range and signal-to-noise of the observations. Given that the main purpose of this routine is to visualise easily the IFS data, several intensity scaling functions are available to the user in order to improve the contrast and the identification of spectral features: a full linear (min/max) sampling, a power-law (gamma) function, a logarithmic transformation, and an inverse hyperbolic sine scaling. For a full explanation of the scaling functions and these parameters see the `Coyote` documentation of the `cgImgScl` routine.

5.2 PMAS users

The automatic identification of the position table for the default instruments/setups is based on the number of rows in the RSS file. However for PMAS, all three resolutions yield a RSS file with the same number of spectra. If the user intends to use the PINGSoft default position tables for PMAS, he/she needs to define the instrument's resolution in the FITS header. For that purpose, apply the appropriate command to the science RSS file, i.e.

```
write_hdr, 'pmas_object.fits', 'PMASR', 0.50, 'PMAS resolution', format='f4.2'
write_hdr, 'pmas_object.fits', 'PMASR', 0.75, 'PMAS resolution', format='f4.2'
write_hdr, 'pmas_object.fits', 'PMASR', 1.00, 'PMAS resolution', format='f4.2'
```

Then you can use directly:

```
view_rss, 'pmas_object.fits'
```

5.3 VIMOS users

Same as above, the user needs to define the instrument's resolution in the FITS header by setting the appropriate `VIMOSR` entry:

```
write_hdr, 'vimos_object.fits', 'VIMOSR', 0.33, 'VIMOS resolution', format='f4.2'
write_hdr, 'vimos_object.fits', 'VIMOSR', 0.67, 'VIMOS resolution', format='f4.2'
```

6 PINGSoft quick list

Visualisation

view_ifs: Provides a 2D spatial and spectral interactive visualisation widget for 3D cubes and RSS IFS files.

view_3D: Command-line version of `view_ifs`, it provides an interactive visualisation of the spaxels and spectra of a 3D cube or a RSS file.

Spectra extraction

extract_region: Extracts the spectra of regions selected by hand.

extract_aperture: Extracts the spectra within an elliptical or circular aperture.

extract_radial: Extracts radial average spectra within consecutive elliptical rings from a reference point, based on either fixed bins or S/N floor.

extract_slit: Extracts the spectra within a rectangular aperture, resembling a long-slit observation.

extract_cone: Extracts the spectra within a region defined by a hyperbolic cone.

extract_mask: Extracts the spectra based on a user's given mask or segmentation map.

integrate_3D: Integrates the spectra within a 3D cube or RSS file into a single spectrum.

Data products and analysis

extract_filter: Generates a 2D FITS image after convolving the 3D data with a narrow or broad-band filter.

vfield_3D: Calculates the intrinsic velocity field in 3D data using a wavelength cross-correlation.

s2n_ratio_3D: Calculates the continuum and emission-line S/N of the IFS data, and allows to interactively extracts spectra based on a S/N floor.

voronoi_3D: Applies the Voronoi tessellation method to bin the IFS data to a target signal-to-noise ratio per bin.

s2n_optimize: Extracts spectra interactively based on a S/N continuum and emission line optimization method using cumulative integrated spectra.

IFS manipulation

split_califa: Extracts the FITS extensions of the CALIFA survey data, writing individual files with the appropriate headers for 3D visualisation.

cube2rss: Generates a RSS file and a associated position table from a 3D cube FITS file.

Miscellaneous functions

redshift_3D: Returns the redshift of the input IFS data measured from the integrated spectra.

offset2radec: Transforms small angle offsets in arcsec from a reference point to equatorial coordinates.

radec2offset: Transforms equatorial coordinates to small angle offsets from a given reference point.

set_value2D: Replaces the values in a 2D array of an index vector found with the IDL `WHERE()` function.

set_value3D: Replaces vectors in a 3D array of an index vector found with the IDL `WHERE()` function.

References

Cappellari M., Copin Y., 2003, MNRAS, 342, 345

Rosales-Ortega F. F., 2011, NewA, 16, 220

Rosales-Ortega F. F., Arribas S., Colina L., 2012, A&A, 539, 73

Rosales-Ortega F. F., Kennicutt R. C., Sánchez S. F., Díaz A. I., Pasquali A., Johnson B. D., Hao C. N., 2010, MNRAS, 405, 735

Sánchez S. F., Rosales-Ortega F. F., Marino R. A., Iglesias-Páramo J., Vílchez J. M., Kennicutt R. C., Díaz A. I., Mast D., et al., 2012, A&A, 546, 2

Sandin C., Becker T., Roth M. M., Gerssen J., Monreal-Ibero A., Böhm P., Weilbacher P., 2010, A&A, 515, 35