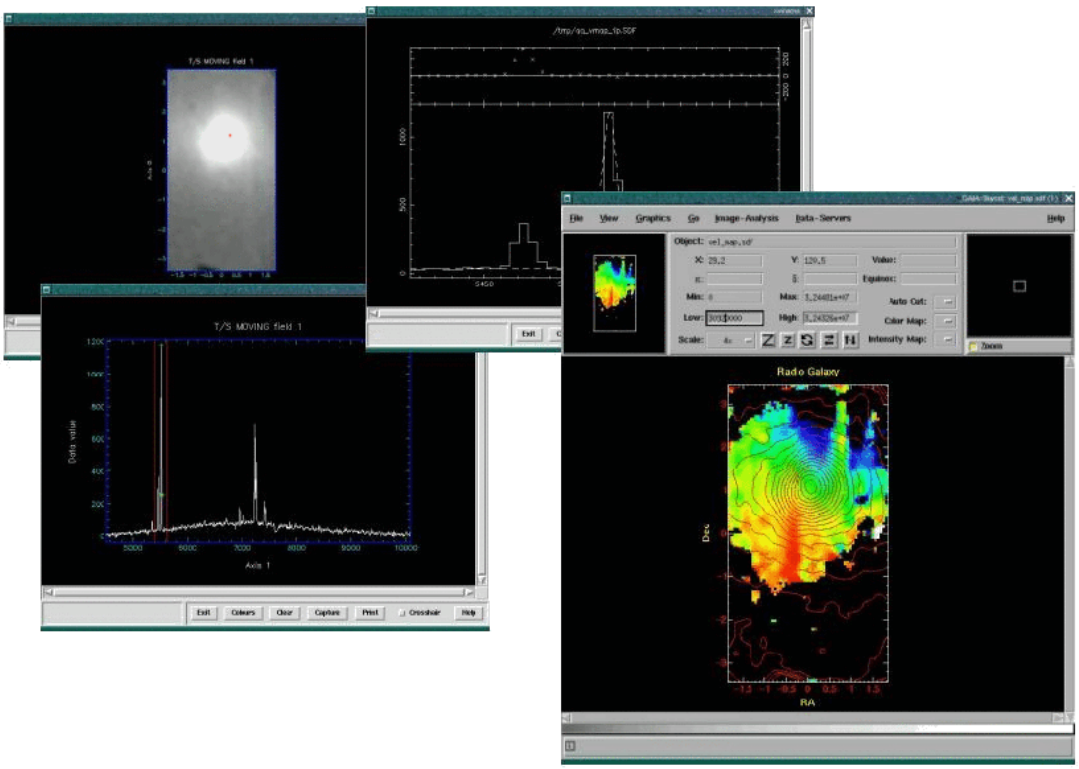
SUN/237.4

Starlink Project
STARLINK User Note 237.4

A. Allan & Malcolm J. Currie

2010 September 7

DATACUBE — An IFS datacube manipulation package
Version 1.3
User's Manual



Abstract

DATACUBE is a package which includes the IFU Data Product Cookbook (SC/16), and a collection of example shell scripts for IFS data cube manipulation.

Contents

Revision history

(1) 9th November 2000; Version 0.1 Original version (AA)

(2) 30th December 2000; Version 0.2 Auto-generated code prologs (AA)

(3) 2nd January 2001; Version 1.0 Release version (AA)

(4) 2005 October 20; Version 1.1 Removed applications, and improved scripts (MJC)

(5) 2006 March 10; Version 1.1 Add gridspec and velmoment, and used implementation status (MJC)

(6) 2008 July 2; Version 1.2 Added trendview and improved the introduction (MJC)

(7) 2010 July 13; Version 1.2 Added pvslice (MJC)

1 The Datacube Package

DATACUBE is a collection of C-shell scripts layered on top of various pieces of the Starlink Software Collection (SSC) for the analysis of spectral datacubes, such as from integral-field spectrometers. The scripts permit graphical selection of spatial and spectral regions and features using the cursor.

DATACUBE provides facilities:

- to view a 'white-light' image of the whole and/or part of the spectral range (**squash**, **passband**);
- to view a grid of channel maps of defined spectral width (**step**);
- to extract individual spectra (**ripper**);
- to extract an arbitrary position-spectral slice (**pvslice**);
- to compare spectra (**compare**, **stacker**);
- to average and compare groups of spectra (**multistack**);
- to plot a grid of spectra with spatial averaging (**gridspec**);
- to build a velocity map from an emission line by line fitting (**velmap**), or by intensity-weighted moments (**velmoment**), or by selecting a cube's voxels using the values of another velocity map as spectral co-ordinates (**mapbyvel**);
- to build an emission-line strength map (**peakmap**); and
- to plot multiple spectra from a cube overlaying fitted baselines and spectral feature mask (**trendview**).

The backbone of the DATACUBE Package is the IFU Data Product Cookbook (SC/16). This includes illustrated examples of the DATACUBE scripts, script programming tips, and lists of potentially useful applications to manipulate and analyse cubes.

The C-Shell approach was a deliberate design decision which was made for two main reasons: first, due to the still developing nature of the field it is difficult to pin down exactly visualisation tasks will be necessary, and the choice of `csh` means that the existing scripts can be easily modified by the end user to do exactly the job required; second, after examining the Starlink Software Collection (SSC) it was realised that a great deal of the functionality required to analyse IFU data already existed and it would be wasteful to re-invent the wheel by re-coding large chunks of software.

Some tasks once mature can, and should, be re-coded in FORTRAN or C, mainly for the speed and memory advantages that this will bring. For instance the **velmap** script as implemented is rather slow and memory intensive, it seems likely that this will be the first candidate to be re-coded in a lower-level language.

Hopefully, while I haven't managed to think of everything, there is enough ground work here so that the approach to solving your data visualisation or data cube manipulation problem is obvious, even if the package doesn't have a script or application to do exactly what you require.

While this document contains the detailed descriptions of the shell scripts included with the package, the "how to" information detailing their use can be found in SC/16.

I would welcome comments, criticisms, contributions and corrections to the package once people start to figure out what they want to do with this sort of data. Comments can be sent either to the authors at `aa@astro.ex.ac.uk` or `mjc@star.rl.ac.uk` or to the Starlink software librarian `starlink@jiscmail.ac.uk`.

2 Setting up the applications

After sourcing the Starlink `/star/etc/login` and `/star/etc/cshrc` files the `datacube` command will make the datacube package scripts available to you, the following message (or one much like it!) should appear

```
% datacube

  DATACUBE applications are now available -- (Version 1.2)
   Support is available by e-mailing starlink@jiscmail.ac.uk

       Type cubehelp for help on DATACUBE commands.
  Type 'showme sun237' to browse the hypertext documentation
  or 'showme sc16' to consult the IFU data product cookbook.

%
```

at this point you can test whether the datacube package has been correctly installed by typing

```
% datacube_test
```

which will run the test script for the package. If the script does not run correctly you should consult your Starlink system administrator.

It should be noted that there are a great many useful applications available in other Starlink packages that will assist you in dealing with your IFU data (see SC/16 for details), so you may, at the minimum, wish to setup the KAPPA applications with the `kappa` command.

```
% kappa

    KAPPA commands are now available -- (Version 1.12)

    Type kaphelp for help on KAPPA commands.
    Type 'showme sun95' to browse the hypertext documentation.

    NOTE, several applications have had major changes made to their
    parameter lists. See the 'Release Notes' section of SUN/95 for
    details.
%
```

Acknowledgments

In compiling this document I (AA) have again depended heavily on the help of many people in the IFS community. However, special thanks should go to Rachel Johnston, Jeremy Allington-Smith, James Turner and Frank Valdes for their co-operation and contributions.

A Descriptions of Individual Applications

compare
Compares multiple extracted spectra from a three-dimensional IFU NDF

Description:

This shell script reads a three-dimensional IFU NDF as input and presents you with a white light image of the cube. You can then select an *X*-*Y* position using the cursor. The script will extract and display this spectrum next to the white-light image. You can then select another *X*-*Y* position using the cursor, and the script will display this spectrum as well, allowing s comparison of the two.

Usage:

```
compare [-i filename]
```

Command-line Arguments :

- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA and FIGARO packages.

gridspec
Averages groups of neighbouring spectra of a three-dimensional IFU NDF and then plots these averaged spectra in a grid

Description:

This shell script reads a three-dimensional IFU NDF as input and if you request zooming the script presents you with a white-light image of the cube. You can then select the lower and upper spatial limits to plot using the cursor. You can instead supply an NDF section with the filename to define both spatial and spectral limits to plot.

The script averages spectra in the chosen region by specified compression factors in the spatial domain. It then displays the average spectra in a grid, where the exterior axes indicate the spatial co-ordinates of the averaged spectra, and the interior axes the data values against spectra co-ordinates.

Usage:

```
gridspec [-b string] [-i filename] [-z/+z]
```

Command-line Arguments :

- `-b` *string*
  The number of spectra to block average along the *X* and *Y* axes respectively. This should be a comma-separated list or a single number; the latter case applies the same compression factor to both spatial axes. The numbers must be positive integers. `[2]`
- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file.
- `-z`
  The script will automatically prompt to select a region to zoom before prompting for the region of interest. `[TRUE]`
  `+z`
  The program will not prompt for a zoom before requesting the region of interest. `[FALSE]`

Notes:

- The compression is trimmed, so that only compression-factor multiples of original pixels are included in the plot.
- The spatial averaging is aligned to obtain the expected number of pixels irrespective of the pixel origin of the input cube. Note that this may not be suitable if you wish to preserve alignment with another compressed dataset. See KAPPA:COMPAVE parameter ALIGN for more details.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

mapbyvel
Uses a velocity map to extract values from a velocity cube forming a new map

Description:

This shell script uses a velocity map of a spatial region to create a new velocity map. The spatial co-ordinates and velocity value at each pixel specify the co-ordinates of voxels in a spectral cube. The data value in the cube at this voxel becomes the data value in the output map.

While creating velocity maps is the expected usage for the script, it makes no specific tests that the Z co-ordinate is velocity, or even that the image value is velocity. In mathematical form the output image values are given by the following expression.

$$\mathrm{Output}(x, y) = \mathrm{Cube}(x, y, \mathrm{Image}(x, y))$$

One application is multi-spectral imaging in complex molecular clouds, where one optically thin spectral line provides a clearer mapping of the cloud and system velocities. Using this specie's velocity map traces the varying velocity, and thus data observed through different spectral lines can be compared, enabling the calculation of cloud parameters such as opacity.

Usage:

```
mapbyvel [-i filename] [-m filename] [-o filename]
```

Command-line Arguments :

- `-i` *filename*
  A three-dimensional NDF with spatial axes and a third velocity axis By default the script will prompt for the input cube.
- `-m` *filename*
  The name of a two-dimensional velocity map, where the data values are velocities. It should have the same spatial co-ordinates as the supplied cube. Such a map can be created using **velmap** or KAPPA:COLLAPSE with the Iwc or Comax estimators. By default the script will prompt for the input cube.
- `-o` *filename*
  The filename for the output NDF velocity map.

Notes:

- The script assumes X and Y axes are spatial and Z is spectral. Use KAPPA:PERMAXES to re-orient the cube if that is not the case.

- A further assumption is the cube's *Z*-axis co-ordinate matches the data values in the supplied image. If that is not the case, change the co-ordinate system with KAPPA:WCSFRAME and/or KAPPA:WCSATTRIB.
- It propagates the original variances from the cube, but does not account for errors in the input velocity map affecting the values extracted from the cube.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

multistack
Averages groups of spectra extracted from a three-dimensional IFU NDF and then plots these averaged spectra in a stack

Description:

This shell script reads a three-dimensional IFU NDF as input and presents you with a white-light image of the cube. You can then select a number of *X*-*Y* positions using the cursor. The script will then group these spectra creating an average spectrum for each group. It then displays the average spectra in a 'stack', where each group spectrum plotted offset vertically from the previous one in the stack.

Usage:

```
multistack [-g number] [-i filename] [-n number] [-o number] [-z/+z]
```

Command-line Arguments :

- `-g` *number*
  The number of spectra in a group.
- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file.
- `-n` *number*
  The number of groups to extract.
- `-o` *number*
  Offset between the spectra in the stack.
- `-z`
  The script will automatically prompt to select a region to zoom before prompting for the region of interest. `[TRUE]`
  `+z`
  The program will not prompt for a zoom before requesting the region of interest. `[FALSE]`

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

passband
Displays multiple passband images from a three-dimensional IFU NDF.

Description:

This shell script reads a three-dimensional IFU NDF as input and presents you with a white-light image of the cube. You can then select and X-Y position using the cursor. The script will extract and display this spectrum next to the white-light image. You can then select a spectral range using the cursor and the script will display a passband image of the cube in that spectral range.

Usage:

```
passband [-i filename] [-o filename] [-z/+z]
```

Command-line Arguments :

- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file.
- `-o` *filename*
  An output two-dimensional NDF of the passband image. By default the output will be displayed only.
- `-z`
  The script will automatically prompt to select a region to zoom before prompting for the region of interest. `[TRUE]`
  `+z`
  The program will not prompt for a zoom before requesting the region of interest. `[FALSE]`

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

peakmap
Builds a map of emission-line strength from a spectral-cube NDF.

Description:

This shell script reads a three-dimensional spectral-cube NDF and presents you with a white-light image of the cube. You can then select an *X*-*Y* position using the cursor. The script will extract and display this reference spectrum. You will then be prompted to specify various fitting parameters, such as the peak position, using the cursor. The script will then attempt to fit the emission line. The fit will be displayed and you are consulted regarding the goodness of fit. If you consider the fit to be good enough, the script will attempt to perform similar fits to all spectra within the cube, building a two-dimensional NDF image of the strength of the line. These will use the same initial parameters as the reference spectrum, unless option `-a` is selected. You may view this image drawn with a key (option `-d`), and overlay a contour plot (with a key) of the white-light image (option `-c`).

If you do not force the fit to be considered 'good' by using the `-f` command-line option, the script will offer the opportunity to manually refit the spectral feature for individual pixels, such as those that were unsuccessfully fitted by the automatic procedure. In this case, the line-strength map will be plotted and replotted after the new fit, regardless of the `-p` option.

Usage:

```
peakmap [-a] [-c number] [-ci index] [-f] [-i filename] [-l logfile]
        [-o filename] [-p] [-v] [-z/+z]
```

Command-line Arguments :

- `-a`
  Requests that each fit may be inspected then approved or re-fit, not just the initial reference fit. A re-fit will change the initial parameter guesses for subsequent fits, so it is recommended that you note the co-ordinates of spectra to re-fit and tackle these individually in the final manual re-fit stage. `[FALSE]`
- `-c` *number*
  Number of contours in the white-light image. Set to fewer than 1 means no contours are overlaid. `[15]`
- `-ci` *index*
  The palette colour index of the contours. It should be an integer in the range `0` to `15`. It is best to choose an index corresponding to white, or black or another dark colour to make the contours stand out from other elements of the plot. `0` is the background colour. KAPPA:GDSTATE will list the current palette colours. `[0]`
- `-f`
  Force the script to accept the first attempt to fit a Gaussian to the line. This is a dangerous option; if the fit is poor, or unobtainable the script may terminate abruptly

if it is forced to accept the fit. This will additionally suppress manual re-fitting of bad pixels at the end of the run of the script. [FALSE]

- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file.
- `-l` *filename*
  The name of an text log file containing the fitted Gaussian coefficients for each spatial pixel. The file is written as a Starlink Small Text List (STL) described in SUN/190). The STL file comprises a schema to locate and describe the columns, and store global properties; and a formatted table of the coefficients. The schema includes the units and a brief description of each column, and the name of the input NDF used. The table lists the Gaussian centre, peak height the FWHM, and integrated flux, each with its fitting error.
- `-o` *filename*
  The filename for the output NDF of the line-strength map.
- `-p`
  The script will plot the final image map to the current display as well as saving it to an NDF file. Additionally, it will overplot the white-light image as a contour map for comparison. [FALSE]
- `-v`
  The script will generate a VARIANCE array from the line fits and attach it to the peak-intensity-map NDF. [FALSE]
- `-z`
  The script will automatically prompt to select a region to zoom before prompting for the region of interest. [TRUE]
  `+z`
  The script will not prompt for a zoom before requesting the region of interest. [FALSE]

Notes:

- The line-strength image map display scales between the 15 and 98 percentiles. The map uses a false-colour lookup table.
- CURSA:CATCOPY may be used to convert the STL log file (see the `-l` option) to FITS format for analysis with the likes of TOPCAT, provided the STL has the `.txt` file extension. If you want just the tabulated data for your own favourite tool, the schema can be easily removed manually, or with **sed** excluding the lines up to and including the line beginning `BEGINTABLE`.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA and FIGARO packages.

pvslice
Extracts and displays a velocity-position slice from an (RA,Dec,vel) cube

Description:

This script extracts and displays a slice from a position-velocity cube. The slice need not be parallel to either spatial pixel axis.

The script displays a chosen spatial plane from the supplied cube in the left half of the current graphics device. You are then invited to select two spatial positions within the displayed plane using the cursor. A two-dimensional slice is then extracted from the cube that passes through the two selected spatial positions. The first axis in this slice measures spatial distance along the line, and the second axis is the velocity axis. This slice appears in the right-hand side of the current graphics device, and saved to the specified output NDF.

Usage:

```
pvslice -i filename -o filename [-ci index] [-p plane]
```

Command-line Arguments :

- `-ci` *index*
  The colour or colour index of the annotations on the left-hand display indicating the spatial location of the slice. An index should be a positive integer no more than 15, normally 2 or 3 to stand out from other elements of the plot. If absent the annotations will appear in red. `[]`
- `-i` *filename*
  The name of an existing three-dimensional (RA,Dec,vel) cube. The script will prompt for the input file if not supplied on the command line.
- `-o` *filename*
  The name of the NDF in which to store the velocity-position slice. The script will prompt for this NDF if it is not supplied via this option.
- `-p` *plane*
  Velocity or pixel index of the plane to display to enable cursor selection of the slice end points. To specify a velocity supply a floating-point value such as 2.0; for an index supply an integer. `[0]`

Examples:

```
pvslice -i orion_masked -o orion_pvmap
```

This extracts a user-selected plane from the cube NDF called orion_masked, and saves it to NDF orion_pvmap. The slice is shown to the right of the spatial image.

```
pvslice -i orion_masked -o orion_pvmap -p 1.5
```

As above but slice selection is from the velocity plane at 1.5 as opposed to the middle index.

Notes:

- The WCS in the returned NDF is somewhat complex; it has a degenerate pixel axis and a degenerate WCS axis. These could be removed using `ndfcopy trim`, but this would result in a WCS with no inverse transformation (from current to pixel), which gives problems when displaying it, *etc.* Alternatively, the AXIS frame can be used by doing `wcsframe frame=axis`.
- To avoid modification of the input cube, the script creates a temporary copy in the current directory, and deletes the copy upon completion.
- While the script is intended to make position-velocity slices from spectral cubes, the contents of the third axis is arbitrary.
- The script clears the graphics database for the current device. It then creates two FRAME pictures with labels a1 and a2. Within each of these DISPLAY creates FRAME and DATA pictures, the former holding annotated axes. On exit a2 is the current picture.
- The label for the spatial (first) axis in the slice plot is `"Offset from start"`.

Related Applications :

KAPPA: PROFILE; FIGARO: SLICE.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

ripper
Extracts a one-dimensional spectrum from a three-dimensional IFU NDF.

Description:

This shell script reads a three-dimensional IFU NDF datacube as input, presents you with a white-light image of the cube and allows you to select an *X*-*Y* position using the cursor. It then extracts (and optionally displays) the spectrum for that *X*-*Y* position.

Usage:

```
ripper [-i filename] [-o filename] [-p]
```

Command-line Arguments :

- `-i` *filename*
  be a three-dimensional NDF. By default the script will prompt for the input file.
- `-o` *filename*
  The filename for the output spectrum. By default the script will prompt for the name of the output file.
- `-p`
  The script will plot the extracted spectrum to the current display as well as saving it to an NDF file. `[FALSE]`

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

squash
Extracts a two-dimensional white-light image from a three-dimensional IFU NDF.

Description:

This shell script reads a three-dimensional IFU NDF as input and allows you to extract a specific spectral range from the cube to form a white-light image.

Usage:

```
squash [-i filename] [-l number] [-o filename] [-p] [-u number]
```

Command-line Arguments :

- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF, by default the script will prompt for the input file.
- `-l` *number*
  Lower spectral-axis bound of the region of interest.
- `-o` *filename*
  The filename for the output white-light or passband image. By default the script will prompt for the name of the output file.
- `-p`
  The script will plot the extracted image to the current display as well as saving it to an NDF file. `[FALSE]`
- `-u` *number*
  Upper spectral-axis bound of the region of interest.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

stacker
Plots a stack of spectra extracted from a three-dimensional IFU NDF.

Description:

This shell script reads a three-dimensional IFU NDF datacube as input and presents you with a white-light image of the cube. You can then select a number of *X*-*Y* position using the cursor. The script will then extract and display these spectra in a 'stack' with each spectrum plotted offset vertically from the previous one in the stack.

Usage:

```
stacker [-i filename] [-n number] [-o number] [-z/+z]
```

Command-line Arguments :

- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file.
- `-n` *number*
  Number of spectra to extract.
- `-o` *number*
  Offset between the spectra in the stack.
- `-z`
  The script will automatically prompt the user to select a region to zoom before prompting for the region of interest. `[TRUE]`
  `+z`
  The program will not prompt for a zoom before requesting the region of interest. `[FALSE]`

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

step
Steps through the each X-Y plane of a three-dimensional IFU NDF in the spectral direction using KAPPA:DISPLAY to display the output.

Description:

This shell script reads a three-dimensional IFU NDF as input and allows you to step through the datacube in the spectral direction in slices. The output goes to files and (optionally) to the screen.

Usage:

```
step [-i filename] [-l number] [-p] [-s number] [-u number]
```

Command-line Arguments :

- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file.
- `-l` *number*
  Lower spectral-axis bound of the region of interest.
- `-p`
  The script will plot the extracted images to the current display as well as saving it to an NDF file. `[FALSE]`
- `-s` *number*
  Spectral-axis step size for each passband chunk.
- `-u` *number*
  Upper spectral-axis bound of the region of interest.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

trendview
Plots multiple spectra from a cube overlaying fitted trends and mask

Description:

This displays whole or part of a spectral cube in a multiple line plot graphic of data values versus spectral co-ordinates via KAPPA:CLINPLOT. In addition to the raw data, fitted trends and masks generated by KAPPA:MFITTREND are overlaid in different colours for comparison and quality assessment. There is an option to also draw the residuals. The raw data appear in yellow, the fitted trends in darkgreen, the mask in red, and the residuals in blue.

The lower-left plot has annotated axes of data value against spectral co-ordinates.

For ease of use the baseline and mask files have defined names given by the name argument and suffix arguments. See the Notes.

Usage:

```
trendview [-d device] [-i filename] [-o offset] [-r] [-s suffix]
          [-y ytop[,ybot]]
```

Command-line Arguments :

- `-d`
  The device name. `[xwindows]`
- `-i` *filename*
  The name of the raw dataset. The specified file should be a three-dimensional NDF. By default the script will prompt for the input file. A section may be defined to home in either or both spatial and spectral regions.
- `-o`
  An offset to apply to the mask, whose good values are 0. In other words the data-value level at which to draw the mask. Its purpose is improve mask visibility; and to narrow the y plotting range, hence see the residuals between the raw data the fitted trends more clearly. `[0]`
- `-r`
  If present the residuals are plotted.
- `-s`
  The file suffix appended to the baseline and mask file names. `[""]`
- `-y` *ytop[,ybot]*
  A comma-separated list giving the upper and lower data-value limits of every spectrum plot in either order. A positive lower limit will be ignored if the offset (`-o`) is not positive too or when residuals are to be plotted (`-r`). A single value is deemed to be the upper limit, and the lower limit is defaulted.

  If this option is not specified, the limits derive from the range given by the 1 and 99 percentiles, extending by 5% of this range below its minimum and 2% above its

maximum. However, a positive lower limit may again be substituted by a default whenever the offset is not positive too, or residuals are to be plotted.

When the lower limit requires a default, it is set to negative one twentieth of the upper limit if there is no (`-o`) offset applied to the mask, otherwise it is the offset level minus a twentieth of the range between the upper limit and the offset. `[]`

Examples:

```
trendview -i ac9_trim -y 8
```

Plots the spectra in NDFs ac9_trim, ac9_trim_bsl, and ac9_trim_msk between data values -0.4 and 8.

```
trendview -i ac9_trim"(2~10,4~8,)" -y 0.5,10
```

As before but the plot range is -0.5 to 10, and display only displays a 10-by-8 spatial pixel region.

```
trendview -i ac9_trim"(2~10,4~8,-200:200)" -y 0.5,10
```

As the previous example but now only a part (-200 to 200) of the spectral range appears in each plot.

```
trendview -i ac9_trim -o _o4 -o 7
```

Plots the spectra in NDFs ac9_trim, ac9_trim_bsl_o4, and ac9_trim_msk_o4 between data values derived automatically percentiles in ac9_trim, with the mask line drawn at value 7.

```
trendview -i ac9_trim"(2~10,4~8,)" -r -y -3,8
```

As the second example but now the residuals are also plotted. The lower data-value limit on the plots is -3 to accommodate negative residuals.

Notes:

- The raw data is given by the `-i` argument; the fitted trends or baselines is called <filename>_bsl<suffix>; and the mask is called <filename>_msk<suffix>, where <filename> is value of the `-i` option without any section or file extension and <suffix> is the value of option `-s`.
- No key is plotted and the margins are narrow to maximise the plot region. However, these are not guaranteed to work with all aspect ratios of the graphics device and spatial pixels, and for example the title may be clipped or completely absent. This is unlikely to cause any hardship while used an exploratory tool. However, as is normal for a publication graphic, some adjustments of the style options may be required. See Descriptions of Plotting Attributes in SUN/95 to be used with STYLE parameter

in the CLINPLOT calls, and the margins can be enlarged through the MARGIN parameter.

Output :

A composite CLINPLOT plot showing the raw data, and the baseline fit and mask from MFITTREND.

Prior Requirements :

- The supplied NDFs should be spectral cubes with the spectral being the third.
- A large display area is recommended if the screen is used for display, for instance,
  ```
  xmake xwindows -width 1200 -height 900
  ```
  that mostly fills the screen. The actual width and and height in pixels will depend on your screen's dimensions.

  Also a suitable background colour is needed that will show all four loci for the hardwired colours. This should be of mid-intensity so that the light and dark curves will both be visible. For example,
  ```
  palentry 0 steelblue
  ```
  will do this for the original colour scheme.

velmap
Builds a velocity map of an emission line from a spectral-cube NDF by line fitting.

Description:

This shell script reads a three-dimensional spectral-cube NDF and presents you with a white-light image of the cube. You can then select an *X*-*Y* position using the cursor. The script will extract and display this reference spectrum. You will then be prompted to specify various fitting parameters, such as the peak position, using the cursor. The script will then attempt to fit the emission line. The fit will be displayed and you are consulted regarding the goodness of fit. If you consider the fit to be good enough, the script will attempt to perform similar fits to all spectra within the cube, building a two-dimensional NDF image of the velocity of the line. These will use the same initial parameters as the reference spectrum, unless option `-a` is selected. You may view this image drawn with a key (option `-d`), and overlay a contour plot (with a key) of the white-light image (option `-c`).

If you do not force the fit to be considered 'good' by using the `-f` command-line option, the script will offer the opportunity to manually refit the spectral feature for individual pixels, such as those that were unsuccessfully fitted by the automatic procedure. In this case the velocity map will be plotted and replotted after the new fit, regardless of the `-p` option.

Usage:

```
velmap [-a] [-c number] [-ci index] [-f] [-i filename] [-l filename]
       [-o filename] [-p] [-r number] [-s system] [-v] [-z/+z]
```

Command-line Arguments :

- `-a`
  Requests that each fit may be inspected then approved or re-fit, not just the initial reference fit. A re-fit will change the initial parameter guesses for subsequent fits, so it is recommended that you note the co-ordinates of spectra to re-fit and tackle these individually in the final manual re-fit stage. `[FALSE]`
- `-c` *number*
  Number of contours in the white-light image. Set to fewer than 1 means no contours are overlaid. `[15]`
- `-ci` *index*
  The palette colour index of the contours. It should be an integer in the range `0` to `15`. It is best to choose an index corresponding to white, or black or another dark colour to make the contours stand out from other elements of the plot. `0` is the background colour. KAPPA:GDSTATE will list the current palette colours. `[0]`
- `-f`
  Force the script to accept the first attempt to fit a gaussian to the line. This is a

dangerous option; if the fit is poor, or unobtainable the script may terminate abruptly if it is forced to accept the fit. Additionally this will supress manual re-fitting of bad pixels at the end of the run of the script. [FALSE]

- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file.
- `-l` *filename*
  The name of an text log file containing the fitted Gaussian coefficients for each spatial pixel. The file is written as a Starlink Small Text List (STL) described in SUN/190). The STL file comprises a schema to locate and describe the columns, and store global properties; and a formatted table of the coefficients. The schema includes the units and a brief description of each column, and the name of the input NDF used. The table lists the Gaussian centre, peak height the FWHM, and integrated flux, each with its fitting error.
- `-o` *filename*
  The filename for the output NDF of the velocity map.
- `-p`
  The script will plot the final image map to the current display as well as saving it to an NDF file. It will additionally overplot the white-light image as a contour map for comparison. [FALSE]
- `-r` *number*
  Rest-frame spectral unit of the line being fitted.
- `-s` *system* The co-ordinate system for velocities. Allowed values are:

  `"VRAD"` – radio velocity;
  `"VOPT"` – optical velocity;
  `"ZOPT"` – redshift; and
  `"VELO"` – relativistic velocity.

  If you supply any other value, the default is used. ["VOPT"]
- `-v`
  The script will generate a variance array from the line fits and attach it to the velocity-map NDF. [FALSE]
- `-z`
  The script will automatically prompt to select a region to zoom before prompting for the region of interest. [TRUE]
  `+z`
  The script will not prompt for a zoom before requesting the region of interest. [FALSE]

Notes:

- The velocity-map display scales between the 2 and 98 percentiles. The map uses a false-colour spectrum-like colour table so that low-velocity regions appear in blue and high-velocity regions appear in red.

- CURSA:CATCOPY may be used to convert the STL log file (see the `-l` option) to FITS format for analysis with the likes of TOPCAT, provided the STL has the `.txt` file extension. If you want just the tabulated data for your own favourite tool, the schema can be easily removed manually, or with **sed** excluding the lines up to and including the line beginning `BEGINTABLE`.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA and FIGARO packages.

velmoment
Builds a velocity map from a three-dimensional IFU NDF from the intensity-weighted spectral co-ordinates

Description:

This shell script processes a three-dimensional IFU NDF to form a velocity map.

If you request zooming the script first presents you with a white-light image of the cube. You can then select the lower and upper spatial limits to plot using the cursor. You can instead supply an NDF section with the filename to define both spatial and spectral limits to analyse, and from which to create the output velocity map. You may average spectra in the chosen region by specifying compression factors in the spatial domain.

The script then derives the intensity weighted co-ordinate of each spatially averaged spectrum, and converts the data units into a velocity. You may view this image drawn with a key (option `-d`), and overlay a contour plot of the white-light image (option `-c`).

Usage:

```
velmoment [-b string] [-c number] [-ci index] [-i filename] [-o filename]
          [-p] [-r number] [-s system] [-z/+z]
```

Command-line Arguments :

- `-b` *string*
  The number of spectra to block average along the *X* and *Y* axes respectively. This should be a comma-separated list or a single number; the latter case applies the same compression factor to both spatial axes. The numbers must be positive integers. `[1]`
- `-c` *number*
  Number of contours in the white-light image. Set to fewer than one means no contours are overlaid. `[15]`
- `-ci` *index*
  The palette colour index of the contours. It should be an integer in the range `0` to `15`. It is best to choose an index corresponding to white, or black or another dark colour to make the contours stand out from other elements of the plot. `0` is the background colour. KAPPA:GDSTATE will list the current palette colours. `[0]`
- `-i` *filename*
  The script will use this as its input file, the specified file should be a three-dimensional NDF. By default the script will prompt for the input file. If there are multiple spectral lines present, you should supply an NDF section after the name to restrict the spectral range analysed to a specific line and its environs.
- `-o` *filename*
  The filename for the output NDF of the velocity map.
- `-p`
  The script will plot the final image map to the current display as well as saving it to

an NDF file. Additionally it will over- plot the white-light image as a contour map for comparison. [FALSE]

- `-r` *number*
  Rest-frame spectral unit of the line being fitted.
- `-s` *system*
  The co-ordinate system for velocities. Allowed values are:

  `"VRAD"` – radio velocity;

  `"VOPT"` – optical velocity;

  `"ZOPT"` – redshift; and

  `"VELO"` – relativistic velocity.

  If you supply any other value, the default is used. `["VOPT"]`
- `-z`
  The script will automatically prompt to select a region to zoom before prompting for the region of interest. [TRUE]
  `+z`
  The program will not prompt for a zoom before requesting the region of interest. [FALSE]

Notes:

- The compression is trimmed, so that only compression-factor multiples of original pixels are included in the plot.
- The spatial averaging is aligned to obtain the expected number of pixels irrespective of the pixel origin of the input cube. Note that this may not be suitable if you wish to preserve alignment with another compressed dataset. See KAPPA:COMPAVE parameter ALIGN for more details.
- The velocity map display scales between the 2 and 98 percentiles. The map uses a false-colour spectrum-like colour table so that low-velocity regions appear in blue and high-velocity regions appear in red.
- If the cube is compressed spatially, so is the contour map.
- For NDFs in the UK data-cube format, where there is no SPECTRUM or DSBSPECTRUM Domain in the WCS Frames, the data are first collapsed in their native wavelengths in Ångstrom, then the pixel values are converted to VOPT using the simple formula $c(\lambda - \lambda_0)/\lambda_0$, where λ is the intensity-weighted wavelength, λ_0 is the rest-frame wavelength for the chosen spectral line, and c is the velocity of light in km s^{-1}.

Implementation Status:

This script invokes a collection of A-tasks from the KAPPA package.

B Release Notes—V1.1

B.1 New Commands

The following new scripts have been added:

gridspec Averages groups of spatially neighbouring spectra of a three-dimensional IFU NDF and then plots these averaged spectra in a grid.

velmoment Builds a velocity map from the collapsed intensity-weighted spectral co-ordinates. It supports spatial compression.

B.2 Modified Commands

The following applications have been modified:

- **compare** has a `-i` option for the input filename for non-interactive scripting.
- **multistack** and **stacker** plot ordinate limits now always make visible all the spectra, regardless of the chosen offset.
- **passband** does not dump WCS mappings in NDFCOPY. Ensured that the plots of spectra use the axis co-ordinate system. Removed unused `-r` option.
- **peakmap** now has `-c` option to specify the number of contour levels to plot.
- **step** now plots the image slices in an optimally shaped grid rather than full size in quick succession, and thereby permits comparison of the slices.
- **velmap** generalised the transformation of the measured line displacements from the rest-frame co-ordinate in the current spectral system into one of four velocity measurements using WCSTRAN. Added a SpecFrame to achieve this functionality for UK data-cube files. Also recognises the SKY-DSBSPECTRUM Domain.

  Added the `-s` option to define the co-ordinate system of the derived velocities. Ensured that the plots of spectra use the axis co-ordinate system. The script inquires the WCS for the rest-frame frequency to avoid unnecessary prompting.

  Changed lower percentile for display from 15 to 2. Use spectrum colour table. Added a velocity key.

  Additional `-c` option to specify the number of contours; this may be zero to prevent contours be overlaid. Added `-ci` option to specify the colour index of contours.

B.3 General changes

- The package applications COPYAXIS, GETBOUND, and PUTAXIS have been withdrawn. The scripts that invoked these now call KAPPA:SETAXIS (`setaxis ndf=? like=?`) KAPPA:NDFTRACE (access FLBND and FUBND output parameters), and SETAXIS in WCS mode (`setaxis ndf=? dim=? mode=wcs comp=Centre` for each dimension) respectively.

- The scripts have been generalised to spectral co-ordinate systems other than Wavelength in Ångstrom. Units are reported too.
- A bug displacing by +1 pixel negative pixel indices determined by the cursor has been fixed. It was present in most of the scripts.
- Spectrum plots now use the histogram style for greater clarity.
- Reset KAPPA:DISPLAY parameters in a few scripts so that features enabled in other scripts, like a key, do not bleed through.
- Tidied the scripts. Changes included the silent removal and creation of files, alphabetical ordering of the options, documentation of default values of options, corrections to grammar and punctuation, avoidance of :r, aligned output, command-line rubbish disposal, and tab removal.
- Accepts .sdf extension and/or NDF sections to be supplied through new internal routine `checkndf.csh`.
- Shortened many scripts through use of internal routine `getcurpos.csh` to obtain and mark cursor positions.

C Release Notes—V1.2

C.1 New Commands

The following new script has been added:

trendview Plots multiple spectra from a cube overlaying fitted trends and spectral-feature mask. It enables assessment of baseline subtraction and feature masking.

C.2 Modified Commands

The following applications have been modified:

- **peakmap** now has a `-a` option to inspect and approve the fit at each spatial pixel. There is also a `-l` option to log the Gaussian fit parameters and their errors in a Small Text List file.

  It inteprets a non-positive number of contours to mean no contour overlay required. There is a new `-ci` option to specify the colour index of contours.

  It permits case-insensitive responses to prompts except where a prompt requires a file name.

  A bug has been fixed that caused the peak-height error to be stored within the output NDF 's VARIANCE component, instead of the actual variances.

- **velmap** now has a `-a` option to inpsect and approve the fit at each spatial pixel. There is also a `-l` option to log the Gaussian fit parameters and their errors in a Small Text List file.

  It permits case-insensitive responses to prompts except where a prompt requires a file name.

  It now recognises DSBSPECTRUM as a valid spectral domain.

  A number of bugs were fixed: the velocity variance is stored in the output NDF VARIANCE (previously the velocity error had been supplied); the WCS Frame is reset before re-plotting in final individual-pixel refit loop (previously the wrong spectral co-ordinate could be plotted); and the logic for deciding whether to create or switch WCS Frames is disentangled enabling DSBSPECTRUM data to be used.

D Release Notes—V1.3

D.1 New Commands

The following new scripts have been added:

mapbyvel Form a new image from a cube's voxels using all the data values of a supplied image as the *Z* co-ordinate in the cube. The supplied image must span the same *X*-*Y* region as the cube. In practice the cube has RA, Dec, velocity axes and the input image is a velocity map.

pvslice Extracts and displays position-velocity slice from a (RA,Dec,vel) cube. The slice need not be parallel to either spatial pixel axis, and it is defined using the graphics cursor.

D.2 Modified Commands

The following applications have been modified:

- Internal script **checkndf**—used in all of the public scripts—now has new options:
  - `-d` to set the required dimensionality, 2 or 3, where previously only cubes where tested;
  - `-p` to set the prompt string for the required NDF (that may include spaces);
  - `-q` (for quiet) to prevent the reporting of the filename and its vital statistics.
- It has also has an improved NDF-validation method, which allows for NDFs within a container file or NDF extension.