implot

August 16, 2005

Abstract

Plots FITS images using PGPLOT, with an optional overlay of celestial coordinates and source positions (which uses the WCS library).

1 Instruments/Modes

Instrument	Mode
EPIC	all modes which produce FITS images
OM	all modes which produce FITS images

Implot was originally designed for the pipeline, but with version 2.0 a lot more functionality was added, making it a useful interactive tool.

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

3.1 Quick start.

What you need to get started:

- A FITS image file.

That's it. However if you want to plot source marker overlays on the output image, the total list of what you will need is

- A FITS image file.

- A FITS-format source list, containing at least columns named `RA` and `DEC`, with the values in decimal degrees.

The final requirement is only if you have set `radiusstyle`='psf'. Then you will need in total

- A FITS image file.
- A FITS-format source list, containing columns `RA` and `DEC` (values in decimal degrees) and also `SCTS` and `BG_MAP`.
- A Calibration Constituents File or CCF (pointed to by SAS_CCF)

The requirement for the CCF is because in this case the task needs to get information on the Point Spread Function (PSF) from the CCF.

The simplest possible call to **implot** is as follows:

```
implot set=<image_name>
```

Entry-level for plotting sources is:

```
implot set=<image_name> srclisttab=<source_list_name>
```

3.2 Detailed description.

This task takes FITS image datasets, eg those arising from EPIC or OM observations, and produces image files in any of the file formats supported by the local version of PGPLOT. This can be set by the user via the parameter `device`. The default value of `device` is /XS (Xserver), but alternative outputs (depending on what drivers have been installed with PGPLOT) can be used, for example /XW for x-windows, /GIF, /JPEG, /PS for Postscript, and /CPS for Colour Postscript. The Data Products ICD specifies that graphics products should be in PNG format, but this may not be supported by available versions of PGPLOT. A temporary fix is to produce GIF images and convert these to PNG using a separate utility.

The FITS image is first scaled and (optionally) truncated in intensity and size. Either a colour-coded image or a contour map image can be produced, depending on the value of `imagestyle`.

The scaling or image transfer function is controled by the parameter `zscaletype`, which can have values 'linear', 'log' or 'sqrt'.

If it is desired to truncate the brightest and faintest values in the image, the user should set `withzclip`=yes (the default). The truncation values are submitted via the parameters `zhistolo` and `zhistohi`, each of which is restricted to the range 0 to 1. The way these work is perhaps best conveyed by an example. Suppose `zhistohi` is set to 0.85. The brightness values are sorted and the brightness level that separates the fainter 85% of the pixels is calculated. This level is then defined as the maximum brightness to be displayed, and all brighter pixels are set to this level. The working of `zhistolo` is similar but of course inverted.

A new feature in **implot** 2.0 is the ability to show only a subset of the image. The previous versions 'cut away' any blank border area, showing only those rows and columns of the image which contained at least 1 non-zero-valued pixel; this facility is retained, although it is now controllable via the parameter

`trimborder`. The default for this is 'yes' however and I imagine that the user could comfortably forget about it except in exceptional circumstances. The image may now be further truncated in x and y via the parameters `withxyclip`, `xfraclo`, `xfrachi`, `yfraclo` and `yfrachi`. If `withxyclip`=yes, the task shows (filling the whole field available) only that fraction of the image (minus its blank borders, if `trimborder` has been left at 'yes') between `xfraclo` and `xfrachi` etc is displayed. This amounts to a zoom facility.

The colour map (for colour-coded image output, `imagestyle`='image') is choosable via the parameter `colourmap`. Its valid values are as follows:

- -1: Inverted greyscale, ie faint=white, bright=black. This is useful for hardcopy where it may not be desirable always to use up toner in printing large areas of featureless black background.
- 1: Greyscale, ie faint=black, bright=white.
- 2: 'Rainbow.'
- 3: 'Heat.'
- 4: 'IRAF.'
- 5: 'AIPS.'
- 6: 'TJP.'
- 7: 'CGP.'

Note that 0 is not a valid colour map index.

The default colour map is number 7 which is a rainbow with the addition of black for the very lowest level, shading into white at the highest levels.

If `imagestyle`='contour' has been chosen the output is a contour map. The spacing between contours may be controlled in one of two ways, either by specifying that separation directly (`contourstyle`='separation', `contourseparation`) or by specifying the number of contours between mininum and maximum image values (`contourstyle`='number', `ncontours`).

Images may be annotated with various ancillary information, all of which can be controlled with input parameters. The available annotations are:

1. Frame and text information: each image can be labeled with the FITS keywords reporting the instrument, observation, exposure, target, date, and principal investigator, if the appropriate information can be found in the file. This facility is invoked via the parameter `withframe`.
2. Coordinate Grid: if selected with `gridstyle`='grid', a sky coordinates (RA, DEC) grid is plotted over the image; `gridstyle`='ticks' causes only the RA and Dec tick marks around the edge of the plot to be drawn. Note that, for these annotations to be possible, the image must contain the appropriate WCS keywords.
3. Annotated sources: if `withsrclisttab`=y (the default) and the appropriate source-list dataset+table (containing columns `RA` and `DEC`, with the values in decimal degrees) is specified using `srclisttab`, a small circle or ellipse is drawn around the position of each selected source. The interface is now fairly versatile: a subset of sources can be selected for plotting; the radii of the circles can be varied in several ways; and labels can be added to the circles. These facilities are described in more detail in the following subsections.

3.3 Source selection.

This is accomplished via the parameter `expression`. The user can supply a boolean selection expression to this parameter, involving (for example) names of columns in the source table. The full grammar of such expressions is described in the **selectlib** library.

An example of a situation in which some selection may be desirable is in the case in which the source list is the standard XMM **emldetect** pipeline product. These lists contain many rows corresponding to the same source position but with different values of the `ID_BAND` keyword. Inclusion of all the rows in the table will lead to plots which have many concentric source circles at each position. It is better to use a selection expression such as

```
expression='(ID_BAND==0)'
```

to limit the sources plotted to those with `ID_BAND` = 0 (or whatever other number is desired).

3.4 Maximum number of sources.

It may be desirable to apply (in addition to the source selection described above) some definite upper limit to the number of sources to be plotted - say, to plot only the first 50 brightest sources. Such a limit may be difficult or impossible to achieve by use of the `expression` parameter. For this reason, an `ncut` parameter has been provided. However, this parameter alone cannot provide full versatility. It is not sufficient just to specify the number of sources to keep: it is also necessary to determine which sources to keep. This means that the sources must be sorted into a series before the series can be truncated after the `ncutth` member. Parameters `withncut`, `ncutsortstyle` and `ncutsortexpression` have been provided to allow the sources to be sorted.

Parameter `ncutsortstyle` has the following allowed values:

- radiusup: sources are ordered in order of increasing radius (see subsection 3.5).
- radiusdown: sources are ordered in order of decreasing radius (see subsection 3.5).
- rownumber: sources are ordered in order of increasing row number in the source table.
- expr: for this value, the task looks in parameter `ncutsortexpression` for an expression involving column names. This expression is evaluated and the sources are then ordered in increasing order of the resulting value.

3.5 Source circle radii.

The task draws a circle of a certain radius at each selected source position. There are now three styles by which the radii may be specified. These are chosen via the parameter `radiusstyle` as follows.

3.5.1 Style 'user'.

With this choice the task sets all radii to the value of `userradius`. The units of this parameter are image pixels.

3.5.2 Style 'psf'.

This is only likely to be useful if the source list is a standard XMM EPIC source list product, since the task looks for columns with names `SCTS` and `BG_MAP`. Note also that the environment variable SAS_CCF must point to a valid CCF for this style to work.

For this setting of `radiusstyle`, the size of each radius is scaled from the brightness of the source. An elliptical locus is obtained at which the source rate per pixel (which is, remember, not truly point-like but spread out over the PSF) decreases to some given fraction of the background rate. The fraction is specified by the parameter `bkgfraction`. If `withellipse`='yes', this ellipse is drawn, but if `withellipse`='no', a circle of radius equal to the larger semi-axis is drawn.

It should be noted that the PSF used in this routine is only approximately correct. You should not attempt to extract quantitative information from the resulting circles/ellipses.

3.5.3 Style 'expr'.

In this style, the radius can be scaled from an arbitrary function of column values in the input source table. This is a two-stage process:

1. The arithmetic expression supplied to the parameter `radiusexpression` is evaluated to give a proto-radius value.
2. The proto-radius value is passed through the function

   ```
   radius = maxradius * max(1 + 0.5 * log10(proto-radius / max(proto-radius)), 0.2)
   ```

   The result is the radius in image pixels. Here 'maxradius' (in image pixels) is supplied via the parameter `maxradius`.

Some users find this a bit too 'foolproof' for complete satisfaction, so I may in the future either make it simpler (ie dispense with the second step above altogether) or at least allow the option for WYSIWYG setting of radii.

3.6 Source labels.

Labels are added to the sources when `withlabels`='yes'. The source of information about labels is controlled by the parameter `labelstyle`. This parameter works in a fashion identical to `ncutsortstyle` (see subsection 3.4). The accompanying expression parameter is `labelexpression`. This expression may be any arithmetic combination of source-list columns which yields a numeric result, or it may be the name of a string-valued column.

At present, only integer values are permitted as source labels, so if `labelstyle`='expr' the column indicated in `labelexpression` must be an integer-valued one. In future it is intended to allow any alphanumeric column value to be used. The parameter interface for labels may at that point be expanded or modified slightly.

The colour, text size and line thickness of the source circles and labels may be specified via the parameters `srccolour`, `labelsize` and `thickness`.

3.7 User-added sources.

In addition to the FITS source list, the user may supply further source particulars directly via the command line. This is controlled via the boolean `addusersources`. Information on the RA and Dec of the sources as well as their desired (alphanumeric) labels and colours is supplied via the list parameters `userras`, `userdecs`, `userlabels` and `usercolours`. Naturally, the order in which the information is supplied to each of these four parameters must be the same, and they must have the same number of list members.

To illustrate, the addition to the **implot** command line of

```
addusersources=yes userras='35.43 35.90' userdecs='-4.76 -4.71' userlabels='SNR_8 RR-32'
usercolours='1 2'
```

causes **implot** to add two extra source annotations to the plot at the RA/Dec pairs specified. User-added sources are denoted by crosses rather than circles, so the user additions here will appear as a white cross at RA=35.43, dec=-4.76 with (white) label SNR_8, and a red cross at RA=35.90, dec=-4.71 with (red) label RR-32.

4 Parameters

This section documents the parameters recognized by this task (if any).

Parameter	Mand	Type	Default	Constraints
set	yes	dataset		

Name of the image dataset to plot.

device	no	string	/xs	

PGPLOT device, e.g. /GIF /CPS (colour Postscript) or /XW (X-windows). These are not case-sensitive.

gridstyle	no	string	ticks	ticks—grid—none

Style of grid overlay.

imagestyle	no	string	image	image—contour

Whether to draw a colour-coded image or a contour plot.

zscaletype	no	string	linear	linear—log—sqrt

The image transfer function.

colourmap	no	integer	7	-2 to 7

Colour scale: -2=reverse-rainbow, -1=reverse-greyscale, 1=grey, 2=rainbow, 3=heat, 4=IRAF, 5=AIPS, 6=TJP, 7=Rainbow+white (default).

contourstyle	no	string	number	number—separation

Style of calculation of the contour separation.

ncontours	no	integer	5	`ncontours`>1

Number of contours (valid if `contourstyle`='number').

contourseparation	no	real	0.2	contourseparation>0

Separation of contours as a fraction of the distance between maximum and minimum (clipped) values of the image (valid if contourstyle='separation').

withzclip	no	boolean	yes	

Truncate brightest and faintest pixels?

zhistolo	no	real	0.0	0.0 ≤zhistolo≤ 1.0

Active only if withzclip=yes. The pixels are ordered in order of increasing brightness; the minimum image brightness is then defined to be that value such that only zhistolo (as a fraction of the total number) of the pixels are fainter than this. Fainter pixels are then set to equal this lower limiting value.

zhistohi	no	real	0.999	0.0 ≤zhistohi≤ 1.0

Active only if withzclip=yes. The pixels are ordered in order of increasing brightness; the maximum image brightness is then defined to be that value such that zhistohi (as a fraction of the total number) of the pixels are fainter than this. Brighter pixels are then set to equal this upper limiting value.

trimborder	no	boolean	yes	

Trim any blank (ie, zero- or null-valued) border from the displayed image?

withframe	no	boolean	yes	

Include the frame plus anciliary information to the side.

withxyclip	no	boolean	no	

Display only a part of the image?

xfraclo	no	real	0.0	0.0 ≤xfraclo≤ 1.0

Active only if withxyclip=yes. Suppose the size of the image in the x-direction (after any blank border has been removed) is P pixels. Only that part of the image starting at pixel xfraclo$\times P$ is displayed. Note that xfraclo must be smaller than xfrachi.

xfrachi	no	real	1.0	0.0 ≤xfrachi≤ 1.0

Active only if withxyclip=yes. Suppose the size of the image in the x-direction (after any blank border has been removed) is P pixels. Only that part of the image ending at pixel xfrachi$\times P$ is displayed. Note that xfraclo must be smaller than xfrachi.

yfraclo	no	real	0.0	0.0 ≤yfraclo≤ 1.0

Active only if withxyclip=yes. Suppose the size of the image in the y-direction (after any blank border has been removed) is Q pixels. Only that part of the image starting at pixel yfraclo$\times Q$ is displayed. Note that yfraclo must be smaller than yfrachi.

yfrachi	no	real	1.0	0.0 ≤yfrachi≤ 1.0

Active only if withxyclip=yes. Suppose the size of the image in the y-direction (after any blank border has been removed) is Q pixels. Only that part of the image ending at pixel yfrachi$\times Q$ is displayed. Note that yfraclo must be smaller than yfrachi.

withsrclisttab	no	boolean	yes	

Plot circles around source positions?

srclisttab	no	table	srclist.ds:SRCLIST	

Name of the dataset+table containing the source list. Note that the name should be in two parts, separated by a colon, these comprising the name of the dataset followed by the name of the table. If the

colon + table name is omitted, the first table in the dataset is used.

expression	no	string		

Source selection expression, possibly involving column names, keyword values or constants. See **selectlib** for the grammar. Only those sources for which the expression evaluates to TRUE are plotted. The default (an empty expression) is TRUE for all sources.

thickness	no	integer	1	1 ≤`thickness`≤ 5

The line thickness to use for source circles and label text.

srccolour	no	integer	7	0 ≤`srccolour`≤ 7

Colour index of the plotted circle and associated label text.

withncut	no	boolean	no	

`withncut` = 'yes' should be used if the user wishes to plot only a certain definite number of sources from the (filtered) source list. This number should then be supplied via parameter `ncut`. Note however that such a specification of the maximum number of sources to plot implies that the list of sources is ordered according to some rule. This ordering is accomplished by defining a function via the parameters `ncutsortstyle` and `ncutsortexpression`. The sources are sorted in increasing order of their value of this function and only the first `ncut` from the resulting sequence are plotted.

ncut	no	integer	30	0 ≤`ncut`

Maximum number of sources to plot. This parameter has no effect when `withncut` = 'no'.

ncutsortstyle	yes	string	'expr'	radiusup—radiusdown—expr—rownumber

This parameter controls how the source list is sorted before being truncated at number `ncut`. See the description of this parameter in subsection 3.4 but, briefly: if `ncutsortstyle` = 'expr', the task evaluates `ncutsortexpression`, sorts the sources in increasing order of the result, then plots only the first `ncut` of them.

ncutsortexpression	yes	string		

If `ncutsortstyle` = 'expr', the task evaluates `ncutsortexpression` and sorts the sources in increasing order of the result. The parameter `ncut` is then read and only the first `ncut` sources are plotted.

radiusstyle	no	string	user	expr—psf—user

This parameter controls how the radii of the source circles are scaled. If `radiusstyle`='psf', an ellipse is plotted about each source that represents the locus at which the intensity of the source's Point Spread Function (PSF) is equal to the background level. If `radiusstyle`='expr', the expression (involving in general keywords, column names etc) found in `radiusexpression` is evaluated and the radii scaled from that. See subsection 3.5.

userradius	no	real	5.0	0 <`userradius`

When `radiusstyle`='user', the present parameter is read and its value (in units of image pixels) is used for all source circle radii.

radiusexpression	yes	string		

This parameter specifies an arithmetical expression, possibly involving column names, keyword values or constants. The result is used to scale the radii of the plotted circles. The parameter is read when `radiusstyle`='expr'.

withellipse	no	boolean	no	

If `radiusstyle`='psf', specifies that sources should be indicated by an elliptical locus. Otherwise a circle with radius equal to the larger semiradius of the ellipse is used.

bkgfraction	no	real	1.0	`bkgfraction` > 0.0

If `radiusstyle`='psf', an ellipse is generated at the locus at which the source counts per pixel (obtained from the column `SCTS`) equal `bkgfraction` times the background counts per pixel (obtained from the column `BG_RATE`).

maxradius	no	real	15.0	1.0 ≤`maxradius`≤ 100.0

The maximum radius (in image pixels) of the plotted circles.

withlabels	no	boolean	no	

If 'yes', additional information is plotted to the right of the source markers. The source of this information is dictated by `labelstyle`.

labelsize	no	real	1.0	0.0 ≤`labelsize`

The font size to use for labels.

labelstyle	yes	string	expr	radiusup—radiusdown—expr—rownumber

This parameter controls how source labels are assigned. See the description of this parameter in subsection 3.6.

labelexpression	yes	string		

If `labelstyle` = 'expr', the task evaluates `labelexpression` and uses the result as the source label text. The expression may be either an arithmetic expression of columns which has a numerical result, or the name of a single column which can also be string-valued. That is, string-valued columns are not permitted in arithmetic expressions in parameter `labelexpression`.

addusersources	no	boolean	no	

Plot additional sources at user-supplied positions?

userras	yes	real list		0.0 ≤`userras`≤ 360.0

A list of Right Ascensions of the user-added sources. This and parameter `userdecs` are of type 'angle' described in the **param** task documentation.

userdecs	yes	real list		−90.0 ≤`userdecs`≤ 90.0

A list of Declinations of the user-added sources. This and parameter `userras` are of type 'angle' described in the **param** task documentation.

userlabels	no	string list		

A list of label text strings for the user-added sources.

usercolours	no	integer list	2	0 ≤`usercolours`≤ 7

A list of colour indices for the user-added sources. These are defined as follows:

- 0: black.
- 1: white.

- 2: red.
- 3: green.
- 4: blue.
- 5: cyan.
- 6: purple.
- 7: yellow.

5 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

badZclipLevels *(error)*
The maximum permitted image brightness is less than the minimum.

badPGPLOTdevice *(error)*
Invalid device for PGPLOT

badImageTransferId *(error)*
Invalid `zscaletype` value

badImageStyle *(error)*
Unrecognised value of `imagestyle`.

badZclipValues *(error)*
`withzclip`=yes and `zhistolo` > `zhistohi`.

badNcutSortStyle *(error)*
Unrecognised value of `ncutsortstyle`.

badRadiusStyle *(error)*
Unrecognised value of `radiusstyle`.

badLabelStyle *(error)*
Unrecognised value of `labelstyle`.

imageAllZero *(error)*
`trimborder`=yes but whole image is blank.

sizelessImageArray *(error)*
The image is of zero size.

badXclipValues *(error)*
`withxyclip`=yes and `xfraclo` > `xfrachi`.

badYclipValues *(error)*
`withxyclip`=yes and `yfraclo` > `yfrachi`.

badContourStyle *(error)*
Unrecognised value of `contourstyle`.

badImageStyle *(error)*
Unrecognised value of `imagestyle`.

badWcsType *(error)*
The WCS system is not RA—TAN/DEC–TAN. No other coordinate system is yet recognized by the task.

badWcsWorldUnit *(error)*
The WCS world unit is not degrees. No other unit is yet recognized by the task.

badGridStyle *(error)*
Unrecognised value of `gridstyle`.

pgcrvlErr *(error)*
Error in drawing the RA/Dec grid. Rerun with `gridstyle`=none.

noSourcesSelected *(error)*
No sources passed the filtering expression.

allRadiusRowsNull *(error)*
All the rows specified by `radiusexpression` are null-valued.

badInstrum *(error)*
Unrecognised value of instrumentId obtained from the image dataset.

badLabelColumnDataType *(error)*
The data type of the result of `labelexpression` has not been recognized.

badLabelsStyle *(error)*
Unrecognised value of `labelstyle`.

badColourMap *(error)*
Unrecognised (zero) value of `colourmap`.

badBrightnessRange *(warning)*
The minimum and maximum image brightness truncation levels are the same.
corrective action: Contours are not plotted if they were requested.

contourSeparationTooLarge *(warning)*
The contour separation specified by `contourseparation` results in <= 1 contour.
corrective action: None.

sourcesOutside *(warning)*
There are source positions outside the displayed image area. No cause for alarm.
corrective action: None.

6 Input Files

1. FITS image, in any of the data types output by **evselect**. The image must be the primary image of the FITS file.
2. A FITS source list. This table must have columns called `RA` and `DEC` (in decimal degrees). If `radiusstyle`='psf', it must also have columns called `SCTS` and `BG_MAP`.

7 Output Files

1. PGPLOT output file, depending on plot device selected (default name also depends on the format, e.g. pgplot.ps for Postscript, pgplot.gif for GIF.

8 Algorithm

The PGPLOT library routine PGIMAG is used to plot the image. The WCS library of Mark Calabretta is used to compute and plot the optional celestial coordinate grid (or tick-marks).

```
read parameters;

load the image;

find all null-valued pixels and set them to the minimum non-null image value;

if (--withzclip) {
  calculate the effective minimum and maximum image values effMin and effMax
  by histogram clipping;
} else {
  effMin = minval(image);
  effMax = maxval(image);
}

set up PGPLOT;

# Avoid irritating phenomenon in which background and foreground colours swap
# for ps/cps output, except within the image itself:
if (pgplot device is 'PS' or 'CPS') {
  if (labelcolour==0 || labelcolour==1) {
    labelcolour = 1 - labelcolour;
  }
  if (usercolours==0 || usercolours==1) {
    usercolours = 1 - usercolours;
  }
}

#.........................................................
# Plot the image:

calculate x and y pixel limits from --trimborder and --xfraclo etc;

call pgimag;

if (--withframe) {
  draw frame;

  write info at the side of the plot;

  draw colour wedge;
```

```
  if (--gridstyle eq 'none') {
    annotate frame with X and Y ticks;
  } else {
    annotate frame (using wcs keywords) with RA and DEC ticks;
  }
}

if (--gridstyle ne 'none') {
  draw RA/DEC grid;
}

if (withSrcList) {
#.....................................................
# Plot the sources:

  filter the source list using --expression;

  calculate source radii;

  if (--ncutsortstyle ne 'none') {
    sort the source list in order of the value of --ncutsortexpression;
  }

  truncate the sorted list at --ncut;

  calculate the source labels;

  foreach (remaining source) {
    calculate the image position of the source;

    plot the source circle;

    add the label;
  }

  if (--withframe) {
    write some stuff to the base and side of the plot;
  }
}
```

9 Comments

References