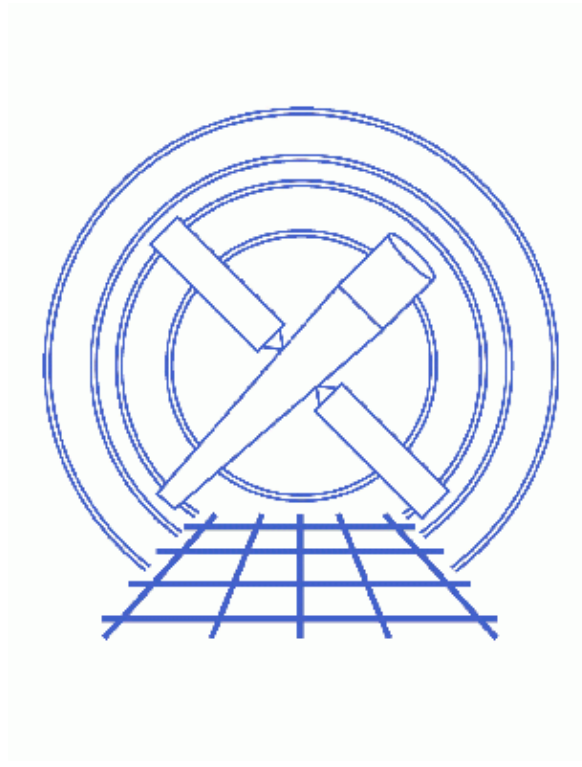


Fitting FITS Image Data



Sherpa Threads (CIAO 3.4)

Table of Contents

- **Getting Started**
 - ◆ Downloading the data
 - ◆ Downloading the sherpa_plotfns.sl script
 - ◆ Loading the script into Sherpa
- **Reading FITS Image Data into Sherpa**
 - ◆ Changing the default coordinate setting
 - ◆ Changing the default statistic
 - ◆ Changing the default fit method
- **Filtering Image Data**
 - ◆ Interactively filter the data
 - ◆ Define the filter on the command line
 - ◆ Using the S-Lang functions
- **Defining a Source Model**
- **Fitting the model**
 - ◆ Examining the residual image
 - ◆ Calculating errors on the fit parameters
- **Adding an extra component**
- **What difference does the integration setting make?**
- **Saving, restoring, and checking the Sherpa settings**
- **Summary**
- **History**
- **Images**
 - ◆ Image of the data
 - ◆ Filter region defined on the image
 - ◆ Image of the filter
 - ◆ Filtered data
 - ◆ Radial profile of the initial parameter values
 - ◆ Radial profile of component g1
 - ◆ Residuals of the best-fit model using: g1+bgnd
 - ◆ Radial profile of the best-fit model using: g1+bgnd
 - ◆ Residuals of the best-fit model using: g1+g2+bgnd

Fitting FITS Image Data

Sherpa Threads

Overview

Last Update: 1 Dec 2006 – reviewed for CIAO 3.4: no changes

Synopsis:

This thread models image data of the supernova remnant G21.5–0.9 using a source model expression that involves multiple model components. While the sample datafile used in this thread is available as [sherpa.tar.gz](#), the [Reading FITS Image Data into Sherpa](#) section shows how it was created.

Related Links:

- The [Fitting data in Sherpa](#) section of the "Introduction to *Sherpa*" thread describes the concepts used in *Sherpa* to describe the fitting process.

Proceed to the [HTML](#) or [hardcopy \(PDF: \[A4\]\(#\) / \[letter\]\(#\)\)](#) version of the thread.

Getting Started

Downloading the data

The data used in this thread is available in the [sherpa.tar.gz](#) file, as described in the "[Getting Started](#)" thread.

Downloading the sherpa_plotfns.sl script

The thread uses the `plot_rprofr()` function which is included as part of the [sherpa_plotfns.sl](#) script. The most recent version of `sherpa_plotfns.sl` is version 1.29 (02 Nov 2004):

```
unix% grep Id $ASCDS_CONTRIB/share/slsh/local-packages/sherpa_plotfns.sl
% $Id: sherpa_plotfns.sl,v 1.29 2004/11/02 15:59:14 dburke Exp $
```

Note that `$ASCDS_CONTRIB/share/slsh/local-packages/` is the default path in the standard CIAO scripts installation; see the [Scripts page](#) for more information. **Please check that you are using the most recent version before continuing.** If you do not have the script installed or need to update to a newer version, please refer to the [Scripts page](#).

Loading the script into Sherpa

The `sherpa_plotfns.sl` script is loaded into a *Sherpa* session with the `evalfile` function:

Fitting FITS Images – Sherpa

```
sherpa> () = evalfile("sherpa_plotfns.sl");
```

The version may also be found by viewing the `_sherpa_plotfns_version` or `_sherpa_plotfns_version_string` variables once the file is loaded:

```
sherpa> _sherpa_plotfns_version
1.29
sherpa> _sherpa_plotfns_version_string
1.29
```

If you wish the functions to always be available to *Sherpa* add the following line to your `~/ .sherparc` file:

```
() = evalfile("sherpa_plotfns.sl");
```

For more information on configuring *Sherpa*, see the [Customizing Sherpa with a Resource File](#) thread.

Reading FITS Image Data into Sherpa

Sample ObsID used: 1838 (ACIS–S, G21.5–0.9)

File types needed: `evt2`


First, an image of the region of interest is created:

```
unix% punlearn dmcoppy
unix% dmcoppy \
  "acisf01838N001_evt2.fits[sky=box(4059.25,4235.75,521.5,431.5)][bin x=:2,y=:2]" \
  image2.fits
```

We use a binning factor of 2 in order to reduce the number of pixels in the image, and hence shorten the time taken to fit the image. The actual choice of binning scale depends on the quality of the data and the models you wish to fit.

After starting a *Sherpa* session, the dataset is input using the `READ DATA` command. It can be shortened to just `DATA` and, once the image is read in, it can be displayed using the `IMAGE` command:

```
sherpa> DATA image2.fits
sherpa> IMAGE DATA
```

which creates [this image](#) . The default imager is `ds9`; see the [Using SAOImage ds9](#) CIAO thread for more information.

Changing the default coordinate setting

The default coordinate system for fitting images is that of `logical` (also called `image`) which uses the FITS convention for numbering pixels: the first pixel is centered at (1,1) with the bottom–left corner being at (0.5,0.5) and the top–right corner being at (1.5,1.5).

Although a simple system, it can be difficult when converting the best–fit parameters, such as source location, back to more–meaningful coordinate systems. For this reason we choose to use the `physical` coordinate system for fitting. This is done with either the `COORD` command or the `set_coord()` function:

```
sherpa> COORD physical
```

Changing the default statistic

When fitting Chandra or XMM–Newton imaging data it is almost certain that many pixels will only contain a small number of counts. For this reason we use the Cash statistic to fit the source model. This choice means that we can not subtract a background dataset but have to fit it instead – as we do below – or ignore the background signal.

```
sherpa> STATISTIC cash
```

You can try increasing the binning factor when creating the image – which *may* mean that you can use one of the chi–square statistics available in *Sherpa* – but this is likely to lose small–scale information present in the image.

Changing the default fit method

The default fitting method (Levenberg–Marquardt) is not well suited for use with the Cash statistic, so we choose to use the simplex optimiser instead. The Powell method is another choice but it tends to be slower than Simplex.

```
sherpa> METHOD simplex
```

Please consider using a variety of methods to ensure you have found the global best–fit rather than a local one.

The current *Sherpa* status may also be reviewed:

```
sherpa> SHOW

Optimization Method: Simplex
Statistic:           Cash

-----
Input data files:
-----


Data 1: image2.fits fits.
Total Size: 56376 bins (or pixels)
Dimensions: 2
Size: 261 x 216
Coordinate setting: physical
Total counts (or values): 32300
```

Filtering Image Data

Here we illustrate several ways of filtering image data within *Sherpa*.

Interactively filter the data

After the data has been displayed, use the "Region → Shape" menu in ds9 to choose a region shape; in this example, we use the `circle` shape. Left–click on the display to draw the desired shape; for further instructions on how to create regions in ds9, see the Using CIAO Region Files thread.

After the desired region size is set, as shown in Figure 2 , the region can be used to filter the data:



Fitting FITS Images – Sherpa

```
sherpa> IGNORE ALL
sherpa> NOTICE IMAGE
```

The NOTICE IMAGE command will include the specified pixels in the filter.

Define the filter on the command line

The same filter can be set on the command line with the region definition:

```
sherpa> IGNORE ALL
sherpa> NOTICE PHYSICAL "circle(4071,4250,135)"
```

One may also specify a file which contains the region definition:

```
sherpa> NOTICE PHYSICAL "region(sherpa.reg)"
```

Using the S-Lang functions

Sherpa provides several S-Lang functions which can be used to filter the data. The set_notice2d() and set_ignore2d() are S-Lang analogues of the NOTICE and IGNORE commands (the values printed to the screen – namely 0 and 1 – indicate that the routines worked correctly).

```
sherpa> sherpa_eval("ignore all")
0
sherpa> set_notice2d(1,"circle(4071,4250,135)","physical")
1
```

The set_filter() function allows you to use an array of values to specify which pixels are to be included. This allows you to filter your data in ways not expressible using the available region shapes. For example you could remove all pixels with a value less than 20.

The NOTICE help file has details on using filter expressions in other coordinate systems. The SHOW command will list the current set of filters as well as how many pixels are left after filtering.

```
sherpa> SHOW

Optimization Method: Simplex
Statistic:           Cash

-----
Input data files:
-----

Data 1: image2.fits fits.
Total Size: 56376 bins (or pixels)
Dimensions: 2
Size: 261 x 216
Coordinate setting: physical
Total counts (or values): 32300


Current filters for dataset 1:
ignore source 1 all
notice source 1 physical circle(4071,4250,135)
Noticed filter size: 14317 bins

Sum of data within filter: 25619
```

An image of the filter can be displayed in ds9 by saying

```
sherpa> IMAGE FILTER
```

Fitting FITS Images – Sherpa

The resulting image for the applied filter is [shown here](#) . The pixels which are displayed as white (and have a value of 1) are those that pass the filter and those in black (with a value of 0) are those that do not.

If you display an image following filtering, what will be displayed is the product of the data and filter mask, over the entire range of the input dataset. You may then need to zoom in, such as when the noticed region is relatively small compared with the full dataset. Therefore to display the filtered data, you just need to say:

```
sherpa> IMAGE DATA
```

as shown in [this image](#) .

Defining a Source Model


We wish to fit the image using a source model expression that involves multiple model components. By the end of the thread we will use two two-dimensional Gaussian functions together with a constant, but for now we begin with only one Gaussian. Since this is an example these choices are not physically motivated but just an empirical choice to describe the emission.

```
sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> SOURCE = GAUSS2D[g1]
sherpa> SHOW source
Source 1: g1
gauss2d[g1] (integrate: off)
  Param   Type      Value      Min      Max      Units
  -----
  1  fwhm  thawed      20       0.2    2000
  2  xpos  thawed  4068.5   3936.5  4206.5
  3  ypos  thawed  4249.5   4115.5  4385.5
  4  ellip frozen      0         0       0.999
  5  theta frozen      0         0       6.2832
  6  ampl  thawed    263      2.63   26300
```

Note that the default integrate setting for the GAUSS2D model is `off`. We will keep it turned off whilst investigating the best-fit solution since the fits will be faster.

Sherpa makes initial guesses for the parameters, but they may not be too sensible. The `IMAGE FIT` command will display a montage of the data, current model, and residuals that can be used to see how well the model describes the data. This can be hard to interpret, so we make use of the `plot_rprofr()` function provided by the [sherpa_plotfns.sl](#) script.

```
sherpa> plot_rprofr(0,150,5)
```


This plots a radial profile of the data and model (points and solid line respectively) in the top plot and a radial profile of the residual image in the bottom plot. The plot for the initial parameter values looks [like this](#) . The arguments in the function call refer to the minimum (0) and maximum (150) radii of the profile and the third value (5) is the bin width (the coordinate system matches that set by the `COORD` command). The center is found from the source component which contains parameters called `xpos` and `ypos` (here it is the `g1` model).

The initial values are way off, so we chose values that better fit the data (by trial and error). Note that this step can greatly reduce both the time needed to fit the data and the chance that the optimiser will end up stuck in a local minimum. We also chose to use a log-log display since the dynamic range along both axes is large.

```
sherpa> g1.ampl = 20
sherpa> g1.fwhm = 60
sherpa> set_log
```

Fitting FITS Images – Sherpa

```
sherpa> plot_rprofr(0,150,5)
```

The resulting plot  shows that the background needs to be included, and that it appears that a single gaussian component is not sufficient to account for both the extended emission and the core. We decide to consider the background component first and so add in a constant model to the source expression:

```
sherpa> SOURCE = g1 + CONST2D[bgnd]
sherpa> bgnd INTEGRATE OFF
sherpa> bgnd.c0 = 0.2
```

We turn off the integration setting on this component to match the gaussian component, and then set the background model to a level estimated (by eye) from the radial profile. Try "plot_rprofr(0,150,5)" to see the difference.

```
sherpa> SHOW SOURCE
Source 1: (g1 + bgnd)
gauss2d[g1] (integrate: off)
  Param  Type      Value      Min      Max      Units
-----  ----  -----  ---      ---      -----
  1  fwhm  thawed      60      0.2     2000
  2  xpos  thawed    4068.5   3936.5  4206.5
  3  ypos  thawed    4249.5   4115.5  4385.5
  4  ellip frozen      0         0      0.999
  5  theta frozen      0         0     6.2832
  6  ampl  thawed     20      2.63   26300
const2d[bgnd] (integrate: off)
  Param  Type      Value      Min      Max      Units
-----  ----  -----  ---      ---      -----
  1    c0  thawed     0.2         0      263
```

Fitting the model



We are now ready to fit the data:

```
sherpa> FIT
smp1x: v1.3
smp1x: initial statistic value = -4.32350E+04
smp1x: converged to minimum = -4.37074E+04 at iteration = 322
smp1x: final statistic value = -4.37074E+04
      g1.fwhm  58.8448
      g1.xpos  4070.5
      g1.ypos  4251.16
      g1.ampl  22.9853
      bgnd.c0  0.214633
```

As can be seen the best-fit values for the gaussian FWHM and amplitude and the background level are close to the values we guessed previously. Since we are using the PHYSICAL coordinate system the xpos, ypos, and fwhm parameters are given in the physical system. For this ACIS image this is the SKY coordinate system, which means the FWHM corresponds to $58.8 * 0.492 = 28.9$ arcseconds.

The fit can be visualised using 'IMAGE FIT' – which displays the data, model, and residual image. Here we are interested in the residuals and so do:

```
sherpa> IMAGE RESID
sherpa> plot_rprofr(0,150,5)
```

The resulting image looks  like this and the radial profile  like this. Although the gaussian does a reasonable job at describing the radial profile of the large-scale emission, the residuals do appear to be spatially correlated. This suggests that the model used here is not sufficient to describe *all* the structure in the source; however it will suffice for this example.

Examining the residual image

It is possible to write out the residual image – using the `WRITE RESID` command – for further analysis. For example, the resulting image could be smoothed using `aconvolve` to enhance any structure in the residuals.

Calculating errors on the fit parameters

Note that the statistic value is negative because we are using the Cash statistic – which is a maximum-likelihood statistic – rather than a chi-square statistic. This means that we can not use the `GOODNESS` command to get an "absolute" measure of the fit – i.e. how well the model actually fits the data. However, we can still use the `PROJECTION` command to estimate errors on these parameters:

```
sherpa> PROJECTION
Projection complete for parameter: g1.fwhm
Projection complete for parameter: g1.xpos
Projection complete for parameter: g1.ypos
Projection complete for parameter: g1.ampl
Projection complete for parameter: bgnd.c0

Computed for sherpa.proj.sigma = 1
```

Parameter Name	Best-Fit	Lower Bound	Upper Bound
g1.fwhm	58.8448	-0.259457	+0.252193
g1.xpos	4070.5	-0.178803	+0.180389
g1.ypos	4251.16	-0.177393	+0.17923
g1.ampl	22.9853	-0.230724	+0.23693
bgnd.c0	0.214633	-0.00512331	+0.00541841


See the [Statistics section](#) of the *Sherpa* threads for more information on calculating errors and confidence regions.

Adding an extra component

The residual image and profile of the fit show excess emission at the center of the source. We will add another gaussian component to try and account for this emission; the FWHM and amplitude were chosen to provide a qualitatively reasonable fit to the radial profile of the source (using `plot_rprof()`).


```
sherpa> SOURCE = gauss2d[g2] + g1 + bgnd
sherpa> g2.fwhm = 10
sherpa> g2.ampl = 100
sherpa> FIT
smp1x: v1.3
smp1x: initial statistic value = -4.75742E+04
smp1x: converged to minimum = -4.94900E+04 at iteration = 1480
smp1x: final statistic value = -4.94900E+04
      g2.fwhm  6.7109
      g2.xpos  4070.79
      g2.ypos  4249.31
      g2.ampl  215.17
      g1.fwhm  64.4178
      g1.xpos  4070.62
      g1.ypos  4251.52
      g1.ampl  17.1613
      bgnd.c0  0.188668
```

Fitting FITS Images – Sherpa

To see if the central component is elliptical – as suggested by the [earlier residual image](#)  – we "thaw" the ellipticity and position–angle parameters of the second gaussian. They are set to non–zero values to move them away from their minimum values (which helps the fit). We increase the [iters parameter](#) of the simplex optimisation method since this particular fit requires more than the default 2000 iterations.

```
sherpa> THAW g2.ellip g2.theta
sherpa> g2.ellip = 0.1
sherpa> g2.theta = 1
sherpa> simplex.iters = 3000
sherpa> fit
smp1x: v1.3
smp1x:  initial statistic value =    -4.95371E+04
smp1x:   converged to minimum =    -4.96677E+04 at iteration =    2364
smp1x:  final statistic value =    -4.96677E+04
      g2.fwhm  8.313
      g2.xpos  4070.74
      g2.ypos  4249.28
      g2.ellip  0.332344
      g2.theta  0.789361
      g2.ampl  215.838
      g1.fwhm  64.5789
      g1.xpos  4070.63
      g1.ypos  4251.54
      g1.ampl  17.0184
      bgnd.c0  0.188113

sherpa> plot_rprofr("g1",0,150,5)
```

The residual profile looks [like this](#) . Since there are now two components with `xpos` and `ypos` parameters (the "g1" and "g2" components), we now have to include the name of the component when calling `plot_rprofr()`.

To see how reliable the ellipticity and theta values are we calculate the one–sigma errors using the PROJECTION method:

```
sherpa> PROJECTION g2.ellip g2.theta
Projection complete for parameter: g2.ellip
Projection complete for parameter: g2.theta

Computed for sherpa.proj.sigma = 1
-----
Parameter Name      Best-Fit Lower Bound      Upper Bound
-----
g2.ellip            0.332344  -0.0206705      +0.0194327
g2.theta            0.789361  -0.0360199      +0.0349404
```

What difference does the integration setting make?

In the fits above we turned off the [integrate](#) setting of the CONST2D model (the default for the GAUSS2D model is off). With `integrate` set to off the models are evaluated at a single point in each pixel; this is the center of each bin. If set to on the model is evaluated across *all* the pixel – this may be analytic for simple cases such as the constant model or estimated numerically for more complex cases such as the gaussian model – which takes into account the variation of the model across the pixel but takes longer to evaluate, and hence fit.

If the model does not change significantly across the pixel then this difference is unlikely to make a difference, other than for the normalisation parameters. For this observation we obtained a FWHM of the second gaussian of close to 8 physical pixels, which is 4 logical (or image) pixels. To see if we are

Fitting FITS Images – Sherpa

under-sampling the image we try re-fitting the data after changing the integrate setting to on for all of the components. Since the pixel size is four physical pixels (due to the 2x2 binning used to create the image) we divide the model amplitudes by four (or close to this number) before re-fitting.

```
sherpa> g1 integrate on
sherpa> g2 integrate on
sherpa> bgnd integrate on
sherpa> g1.ampl = 4
sherpa> g2.ampl = 50
sherpa> bgnd.c0 = 0.05
sherpa> simplex.iters = 3000
sherpa> FIT
  smplx: v1.3
  smplx:  initial statistic value =    -4.95589E+04
  smplx:   converged to minimum =    -4.96666E+04 at iteration =    1564
  smplx:   final statistic value =    -4.96666E+04
          g2.fwhm  7.95973
          g2.xpos  4070.74
          g2.ypos  4249.28
          g2.ellip  0.371972
          g2.theta  0.787842
          g2.ampl  62.421
          g1.fwhm  64.5579
          g1.xpos  4070.63
          g1.ypos  4251.54
          g1.ampl  4.25945
          bgnd.c0  0.0470197
```

In this particular case the results for the second gaussian (g_2) do change slightly – with the most significant change being the increase in the ellipticity – whilst the values for the large-scale emission (g_1) are unaffected. Whether such changes are important depends on the science aims of your analysis and the quality of the data.

In this case we could also re-do the analysis using an image created using a binning factor of 1 instead of 2. This would take longer since there are more pixels, but we may not need to set the integrate setting to on since the pixel size is smaller. If you are concerned about such issues then the best suggestion is to try the fit using the various settings and see what difference it makes.

Saving, restoring, and checking the Sherpa settings

The SAVE command allows you to save the current state of *Sherpa* – e.g. the names of loaded data files, filter settings, defined models, and current parameter settings – to a file. This file can then be used to restore this state with the USE command. This allows you to continue your analysis at a later date, or try out different ideas – such as using a different optimiser or adding another component – without having to re-do all the fits. The [Introduction to Sherpa](#) thread contains a discussion of how to save and restore the *Sherpa* session.

The final overall status of this *Sherpa* session may be viewed as follows:

```
sherpa> SHOW

Optimization Method: Simplex
Statistic:           Cash

-----
Input data files:
-----
```

Fitting FITS Images – Sherpa

```
Data 1: image2.fits fits.
Total Size: 56376 bins (or pixels)
Dimensions: 2
Size: 261 x 216
Coordinate setting: physical
Total counts (or values): 32300

Current filters for dataset 1:
ignore source 1 all
notice source 1 physical "circle(4071,4250,135)"
Noticed filter size: 14317 bins

Sum of data within filter: 25619

-----
Defined analysis model stacks:
-----

source 1 = ((g2 + g1) + bgnd)

-----
Defined source/background model components:
-----

gauss2d[g1] (integrate: on)
  Param  Type      Value      Min      Max      Units
  -----
  1  fwhm  thawed    64.5579    0.2     2000
  2  xpos  thawed   4070.6299  3936.5  4206.5
  3  ypos  thawed   4251.5433  4115.5  4385.5
  4  ellip frozen      0          0        0.999
  5  theta frozen      0          0        6.2832
  6  ampl  thawed     4.2595    2.63    26300

const2d[bgnd] (integrate: on)
  Param  Type      Value      Min      Max      Units
  -----
  1    c0  thawed   4.702e-02    0        263

gauss2d[g2] (integrate: on)
  Param  Type      Value      Min      Max      Units
  -----
  1  fwhm  thawed     7.9597    0.2     2000
  2  xpos  thawed   4070.7413  3936.5  4206.5
  3  ypos  thawed   4249.28    4115.5  4385.5
  4  ellip thawed     0.372     0        0.999
  5  theta thawed     0.7878    0        6.2832
  6  ampl  thawed    62.421    2.63    26300
```

Summary

In this thread we have shown you how you can fit a two-dimensional model to your image data. As with fitting one-dimensional data, care must be taken to avoid reaching a local, rather than global, minimum. It is suggested that you fit your data using different optimisation methods – for instance the fit above could be re-done with the [POWELL](#) method – and to try different initial parameter values when fitting.

In the example above the fit was not very complex: possible additions would have been to [link](#) the centers of the two gaussians to be the same; simultaneously fit the same model to more than one dataset (e.g. for

Fitting FITS Images – Sherpa

multiple observations of a source or when analysing the data from the three imaging cameras in XMM–Newton); include an exposure map in the fit to account for instrumental features such as chip gaps and bad columns; or convolve the model by the PSF during fitting.

History

14 Dec 2004 updated for CIAO 3.2: script version and path

21 Dec 2005 reviewed for CIAO 3.3: no changes

01 Dec 2006 reviewed for CIAO 3.4: no changes

URL: <http://cxc.harvard.edu/sherpa/threads/spatial/>

Last modified: 1 Dec 2006

Image 1: Image of the data

Note that a logarithmic scaling has been chosen to bring out the faint emission of the source.

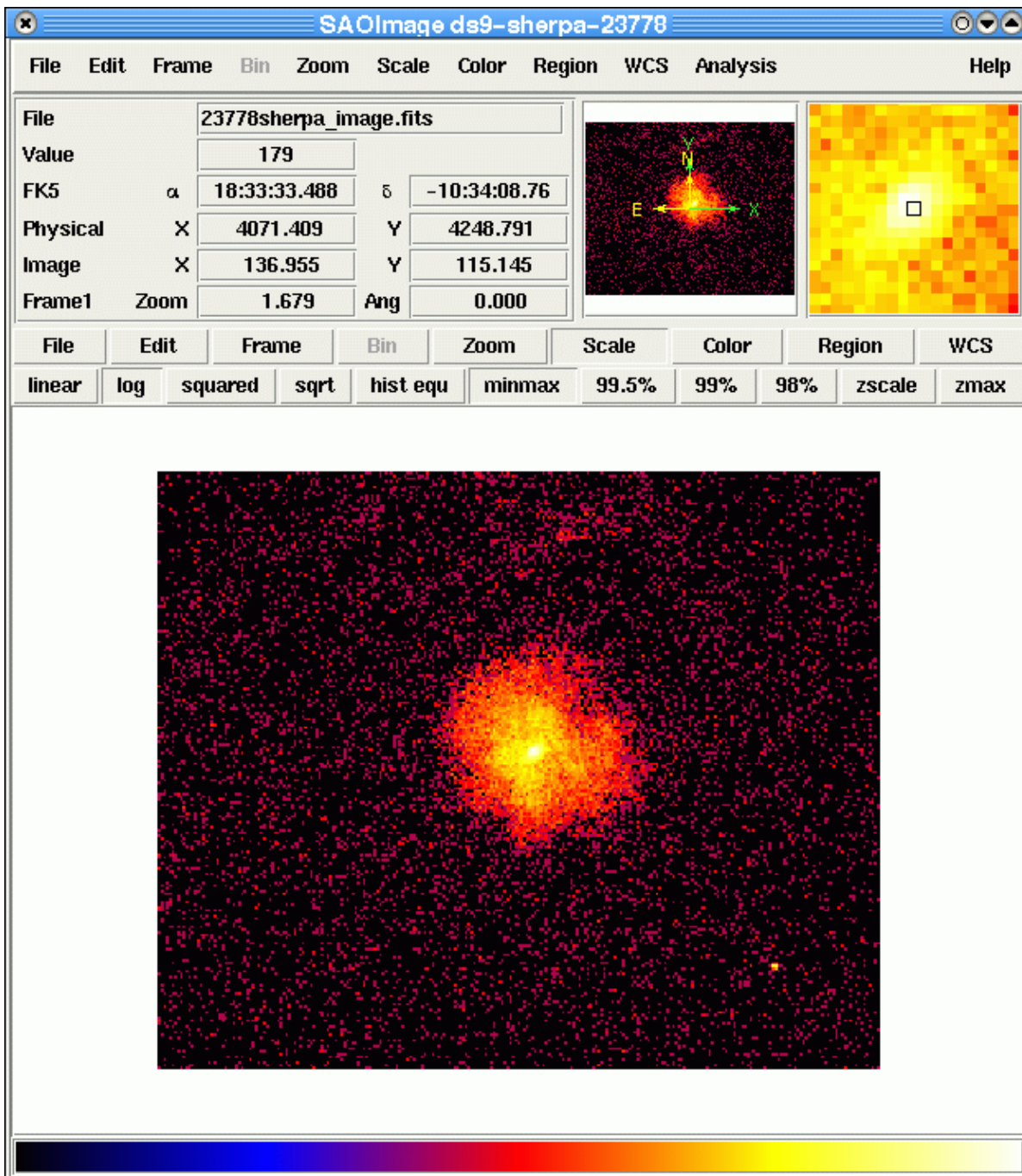


Image 2: Filter region defined on the image

The region displayed on the image is a circle with center at (4071,4250) and a radius of 135 (using the physical coordinate system). This region will be used to fit the source.

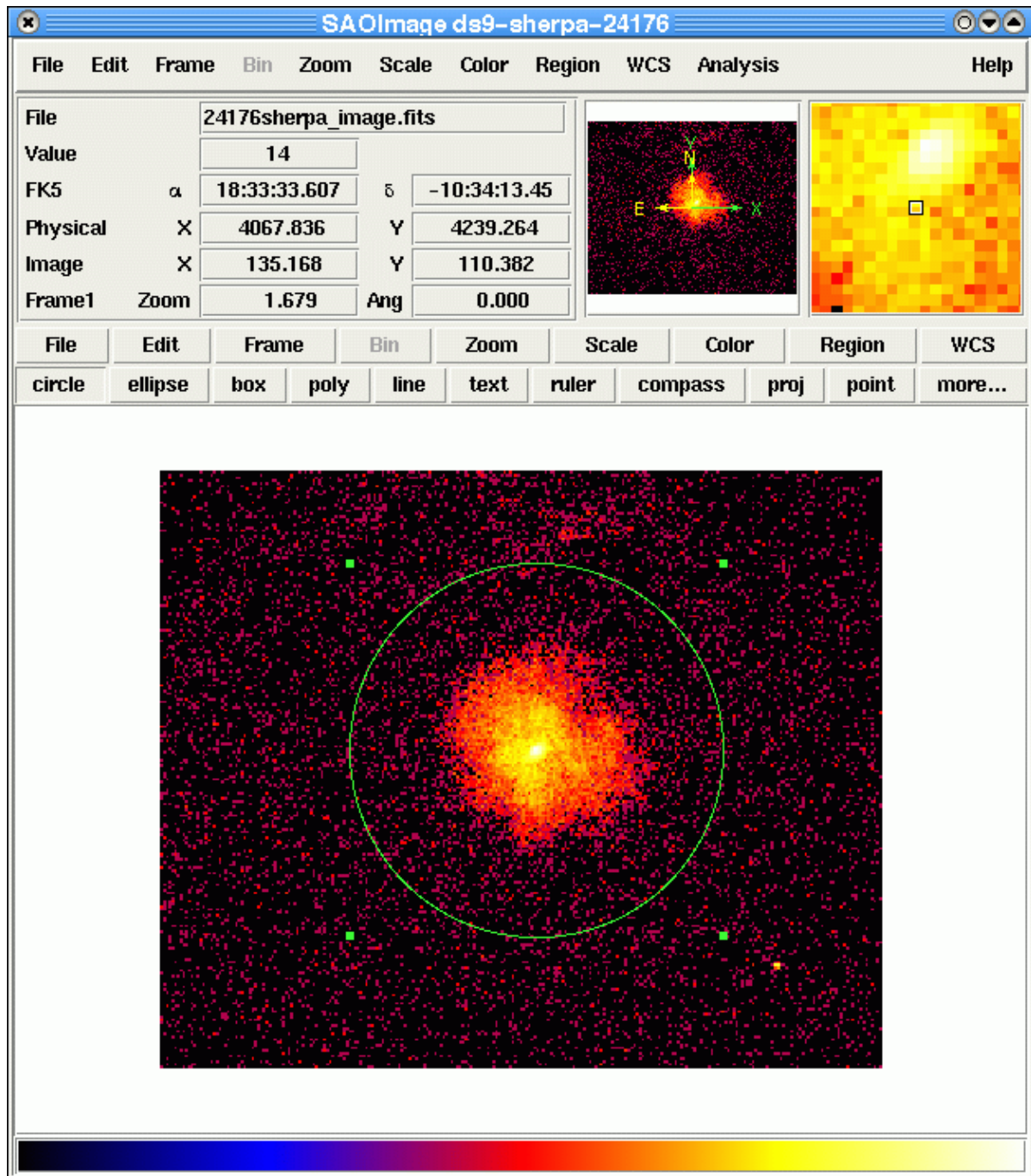


Image 2: Filter region defined on the image

Image 3: Image of the filter

This image shows those pixels that pass the filter (with a value of 1, colored white) and those that do not (value of 0, colored black).

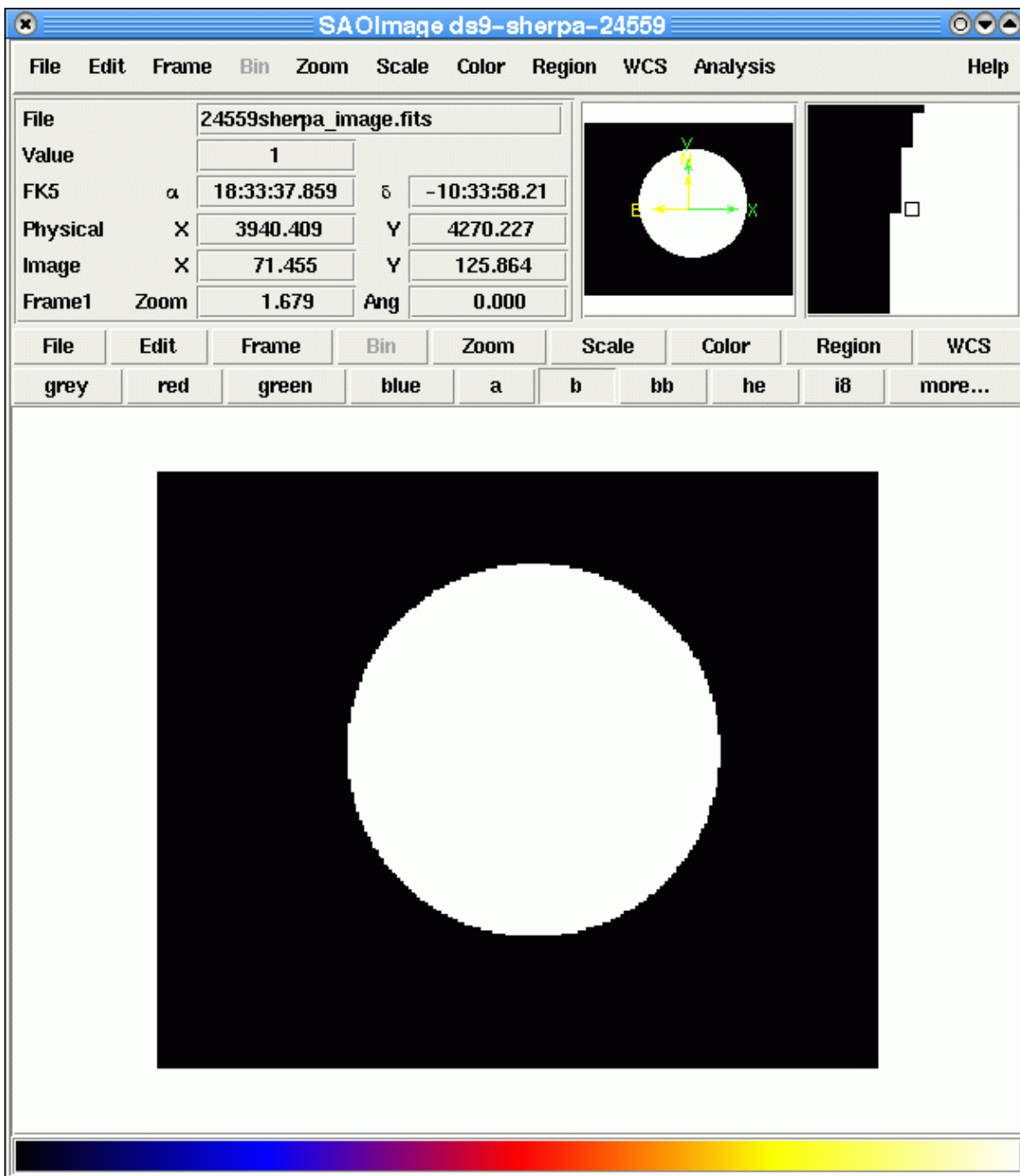


Image 4: Filtered data

This image shows the data that is to be fit (again using a logarithmic scaling to display the full dynamic range of the data). The whole image is shown, which can lead to large areas which have a value of 0 because they are excluded by the filter.

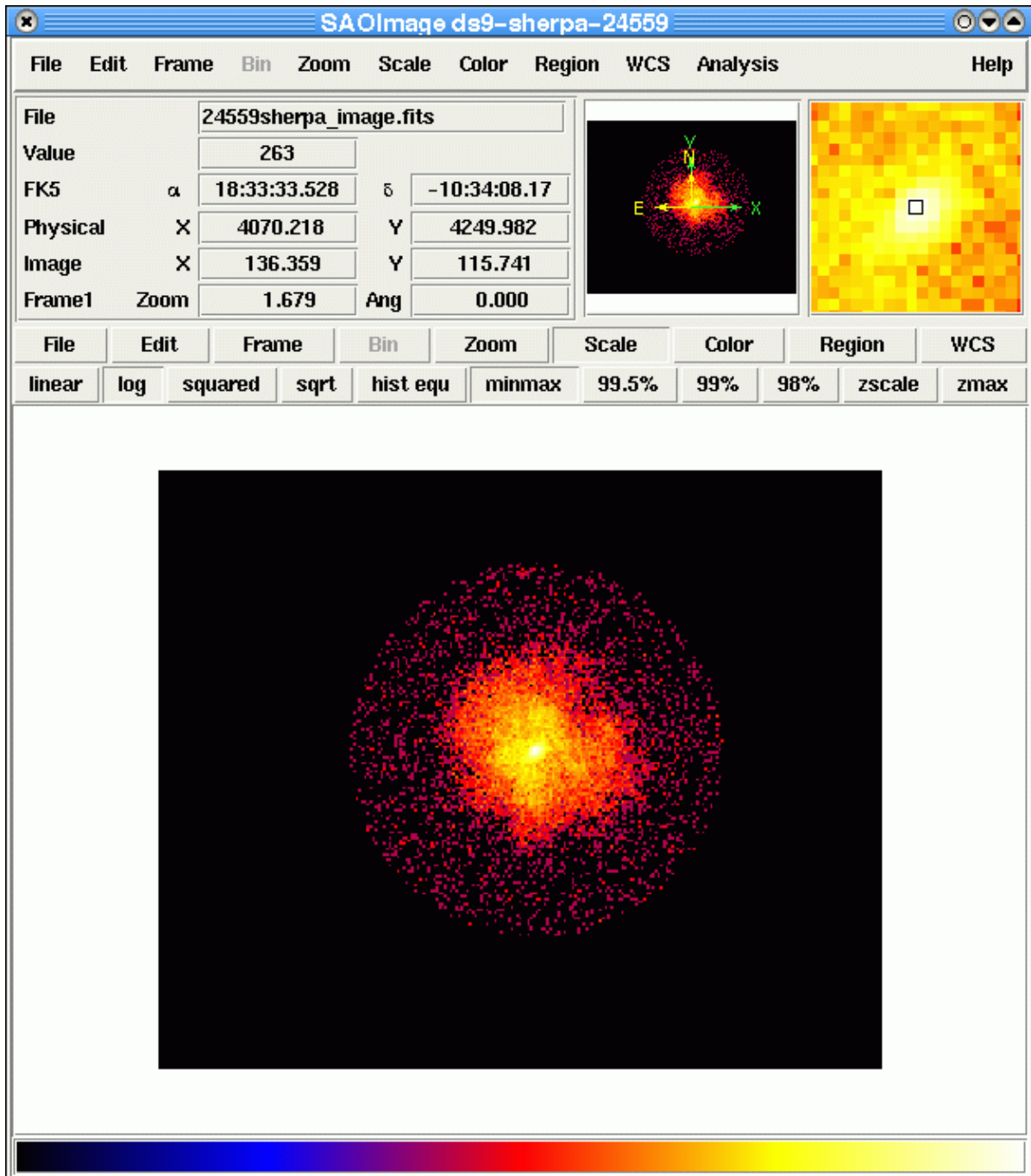


Image 5: Radial profile of the initial parameter values

The top plot shows the radial profile of the data (squares) and current model (red line). The bottom plot shows the radial profile of the residual image. The plot was created using the `plot_rprof_r()` function. It shows that the initial guesses made by *Sherpa* for the parameter values do not describe the data very well.

g1 @ 4068.50,4249.50

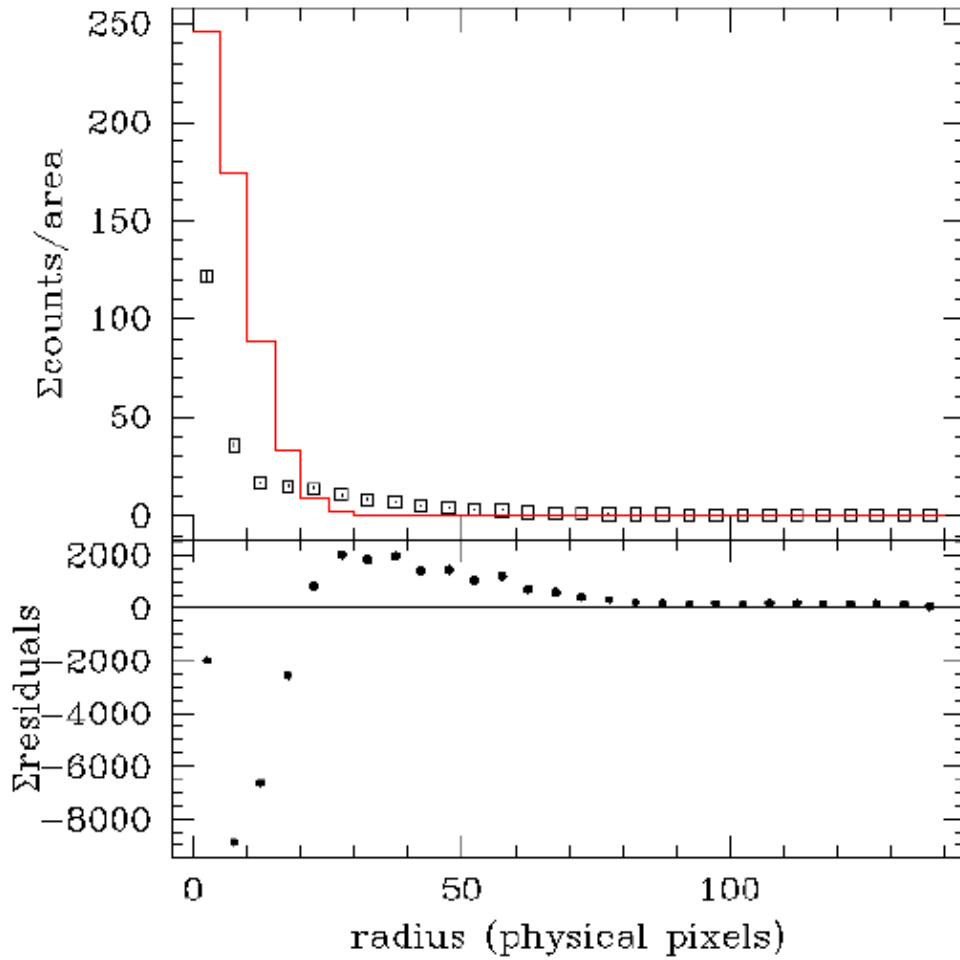


Image 6: Radial profile of component g1

The radial profile of the gaussian model, after manual tweaking of its parameters, suggests that a constant needs to be included to account for the background signal and another gaussian for the core emission. The axes have been changed (except for the ordinate of the residual plot) to use logarithmic scaling due to the large dynamic range of the data

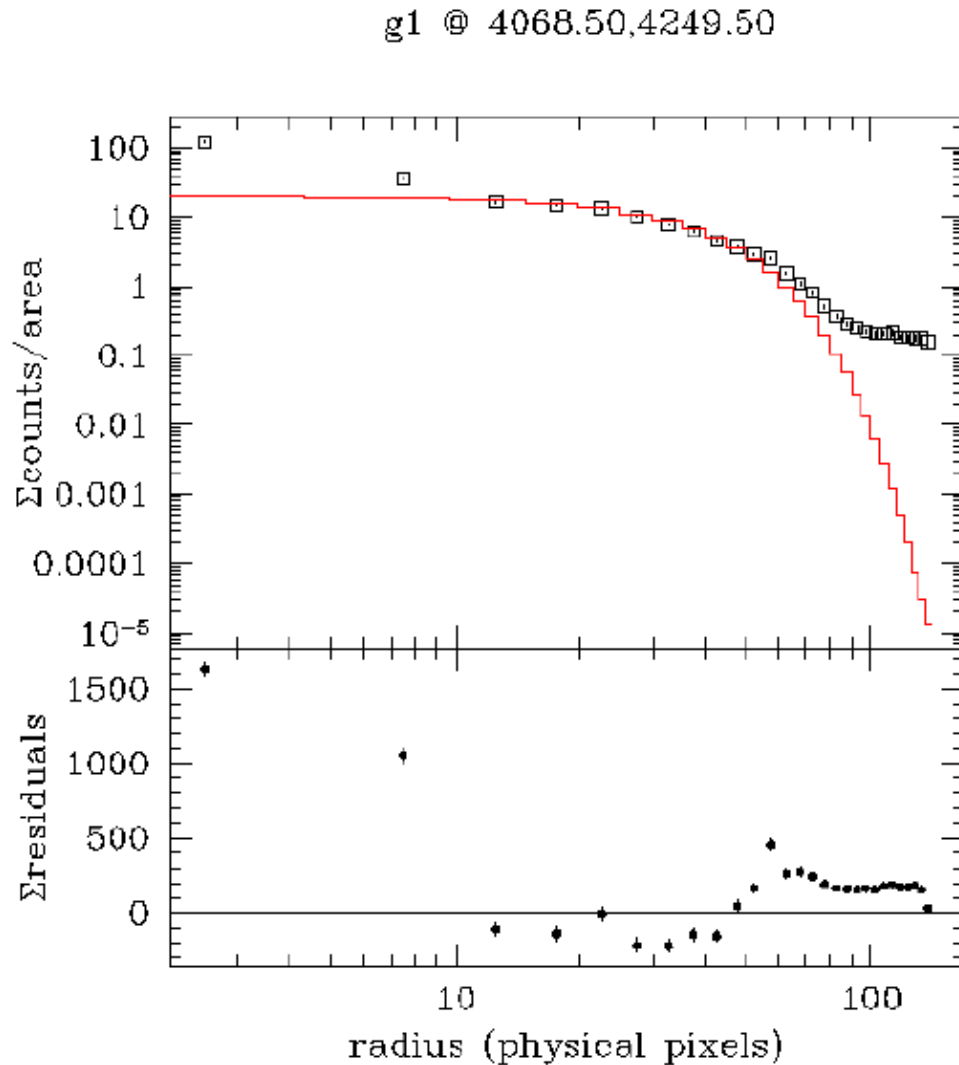


Image 7: Residuals of the best-fit model using: g1+bgnd

The residual image of the best-fitting two-dimensional gaussian plus constant model. The source is obviously more complex than this description: an excess in the core is visible as well as spatially-correlated residuals (e.g. regions of positive or negative pixels) at larger scales.

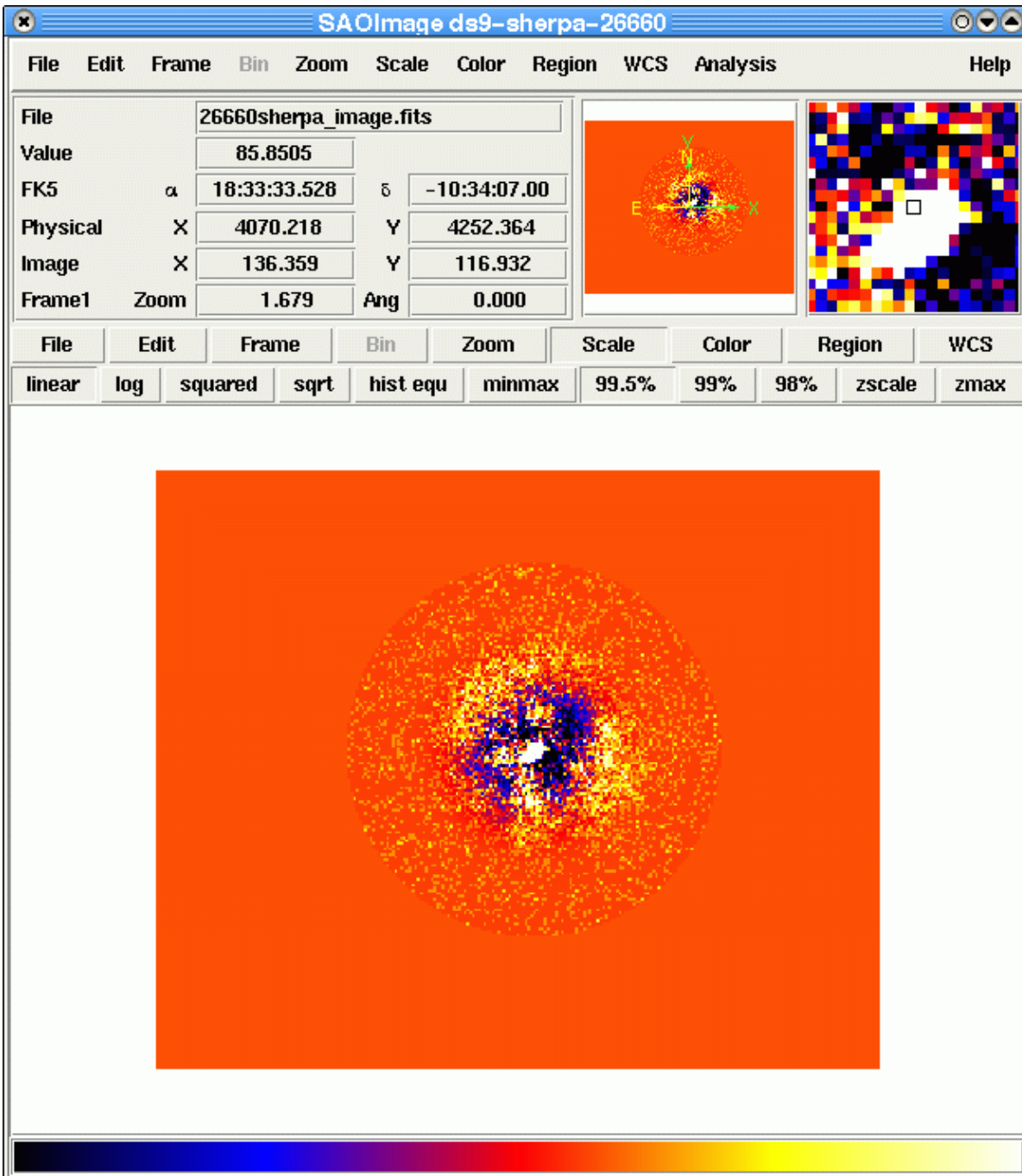
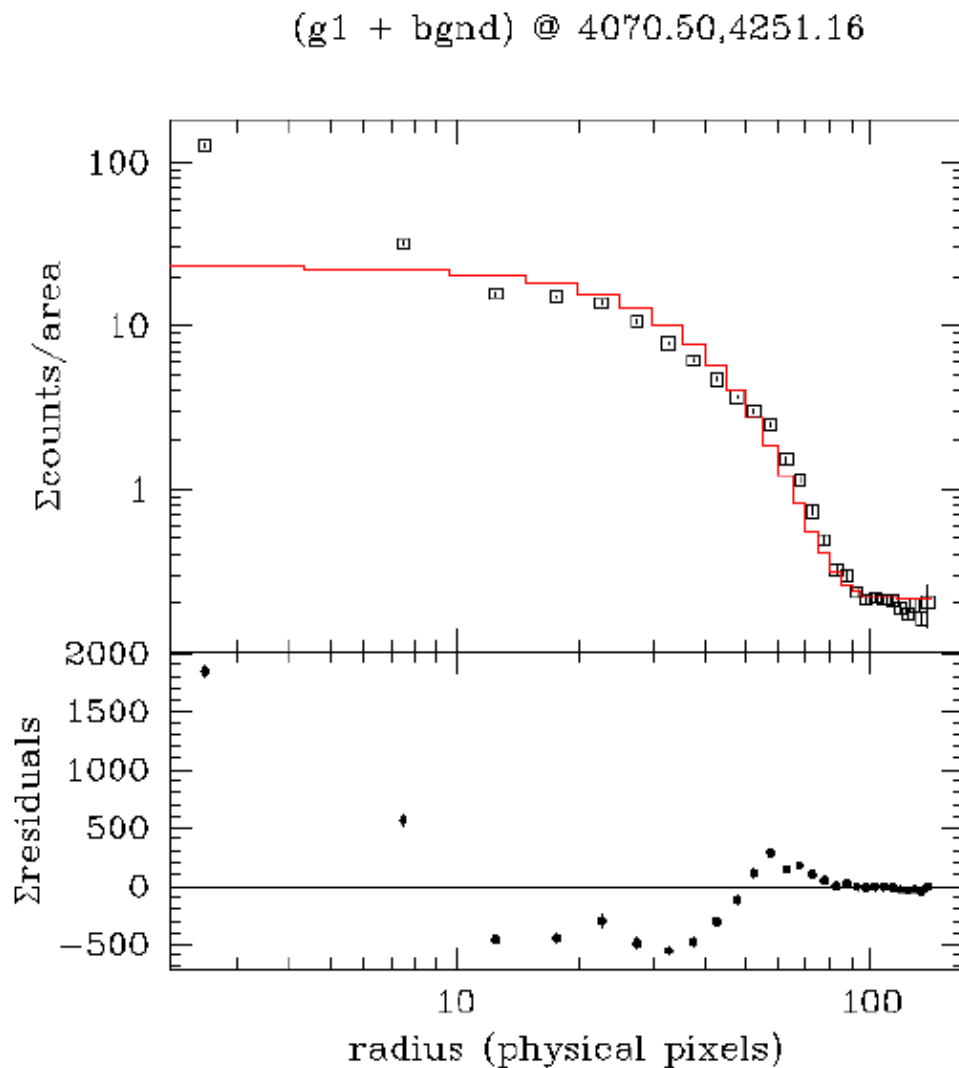


Image 8: Radial profile of the best-fit model using: g1+bgnd

The radial profile of the best-fitting gaussian plus constant model shows the core excess emission obvious in the [residual image of this fit](#). The model over-predicts the data (at radii between 10 and 50 pixels) and under-predicts at larger radii. This could be due to the non-gaussian nature of the emission (the correlated regions seen at large radii in the residual image) or it may be due to the core region biasing the fit parameters of the gaussian model.



The `plot_rprof()` function can be used to display the residuals in units of "sigma" – i.e. the residuals divided by the error – rather than counts, which can often be useful.

Image 9: Residuals of the best-fit model using: g1+g2+bgnd

Using two gaussians together with a constant provides a good description of the radial profile. However, we note that the residual image still shows correlated areas. You could add further model components to the fit to account for these regions, or decide to use a different set of functions to try and describe the emission.

$((g2 + g1) + \text{bgnd}) @ 4070.63, 4251.54$

