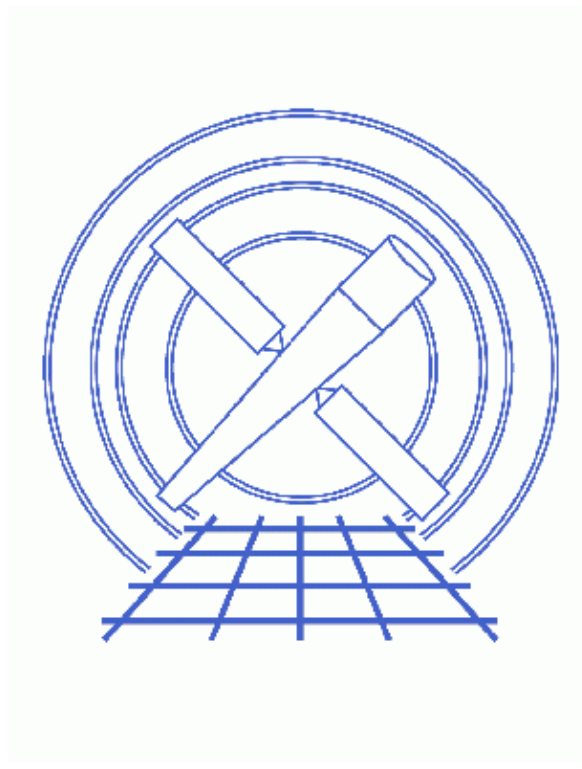


Estimating Errors and Confidence Levels



Sherpa Threads (CIAO 3.4)

Table of Contents

- ***Getting Started***
 - ◆ Downloading the data
 - ◆ Downloading the paramest package
 - ◆ Loading the paramest.sl script into Sherpa
- ***Find the best fit***
- ***Errors on individual parameters (projection)***
- ***How does the fit surface vary for a parameter (interval–projection)?***
- ***How are two parameters correlated (region–projection)?***
- ***Other Uses of the Script***
- ***History***
- ***Images***
 - ◆ Source spectrum
 - ◆ Best fit model with residuals
 - ◆ Plot of interval–projection results
 - ◆ Plot of region–projection results
 - ◆ Improved region–projection results (chips.mingridsize=50)
 - ◆ Improved region–projection results (chips.mingridsize=100)

Estimating Errors and Confidence Levels

Sherpa Threads

Overview

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

Synopsis:

The best-fit values of model parameters are just the beginning of the story; they are only useful if you can attach some idea of an error – or significance level – to them. *Sherpa* provides a number of routines for estimating the errors on a single parameter or for seeing how two parameters are correlated.

All the routines work in essentially the same way: the parameter space around the best fit location is searched until the fit statistic (i.e. chi squared or Cash) increases by a certain amount; the exact amount depends on what confidence level (e.g. in units of sigma or as a percentage) you want. The routines use different strategies for finding the change in fit statistic; as a rule of thumb the faster the method the less accurate the result!

The interface to the *Sherpa* routines provides great flexibility but is not simple to use for the occasional user. In this thread we provide a function that makes the routines simple to use; it is written in S-Lang, but you do not need to know this to use the script!

Read this thread if:

You want to estimate errors or confidence levels for parameters in a fit (to data of any dimensionality) without having to remember how to set the parameters for the *Sherpa* command.

Related Links:

- Section 15.6 ([Confidence Limits on Estimated Model Parameters](#)) of Numerical Recipes version 2 (please be aware of the [conditions of use](#) of the on-line version of Numerical Recipes).
- The ahelp files for the package: [ahelp_paramest](#).
- The ahelp files for the *Sherpa* routines: [COVARIANCE](#), [UNCERTAINTY](#), [PROJECTION](#), [INTERVAL-UNCERTAINTY](#), [INTERVAL-PROJECTION](#), [REGION-UNCERTAINTY](#), and [REGION-PROJECTION](#).
- The "[Step-by-Step Guide to Estimating Errors and Confidence Levels](#)" thread repeats this thread using the default *Sherpa* interface instead of the routines from `paramest.sl`.
- The "[Accessing fit results using S-Lang](#)" thread highlights some of the S-Lang functions that provide access to fit results and statistic values.

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 paramest package

The thread uses the `paramest.sl` script; for information about the script, consult the help file ("[ahelp paramest](#)"). The most recent version of `paramest.sl` is v1.12 (02 Nov 2004):

```
unix% grep Id $ASCDS_CONTRIB/share/slsh/local-packages/paramest.sl
% $Id: paramest.sl,v 1.12 2004/11/02 15:45:22 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](#).

If the script is correctly installed, you should also be able to say:

```
unix% paccess paramest
/home/username/cxcds_param/paramest.par
```

where we have assumed that `$PFILES` is set to

`"/home/username/cxcds_param/soft/ciao/param";` see "[ahelp parameter](#)" for details.

Loading the paramest.sl script into Sherpa

The `paramest.sl` script is loaded into a *Sherpa* session with the [evalfile](#) function:

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

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

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

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

Find the best fit

Before calculating the errors on fit parameters you have to find the best fit. The [Sherpa thread page](#) contains a number of threads showing you how to use the various methods and statistics in *Sherpa*; for this example we will use the default method ([LEVENBERG-MARQUARDT](#)) and the default statistic ([CHI_GEHRELS](#)) to fit a simple model to a binned PHA spectrum. Unlike most fits we neglect the background component here.

The source spectrum we use (`source_grouped_pi.fits`) has the `RESPFILE` and `ANCRFILE` set to point to the location of the [RMF](#) and [ARF](#) for the source, respectively. This means that the [instrument model](#) will be set up automatically when the file is loaded, as shown below.

```
unix% dmlist source_grouped_pi.fits header,clean | grep FILE | grep -v HISTORY
BACKFILE          none
CORRFILE          none
RESPFILE          rmf.fits
```

Errors & Confidence Levels – Sherpa

```
ANCRFILE          arf.fits
```

First we check the *Sherpa* settings:


```
sherpa> erase all
sherpa> show method
Optimization Method: Levenberg-Marquardt

      Name      Value      Min      Max      Description
      ----      -
1  iters      2000         1    10000  Maximum number of iterations
2  eps       1e-03      1e-09         1  Absolute accuracy
3  smplx         0         0         1  Refine fit with simplex (0=no)
4  smplxep     1      1e-04      1000  Switch-to-simplex eps factor
5  smplxit     3         1         20  Switch-to-simplex iters factor

sherpa> show statistic
Statistic:          Chi-Squared Gehrels
```

and then load in the data:


```
sherpa> data source grouped pi.fits
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ ERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin
RMF is being input from:
  /data/ciao/rmf.fits
ARF is being input from:
  /data/ciao/arf.fits
sherpa> ignore energy :0.5,8:
sherpa> set log
sherpa> lp data
```

The resulting plot  shows the source data that is to be fit.

Now we set up the source model – an absorbed power law – and fit it:

```
sherpa> source = xswabs[abs] * powlawld[p1]
abs.nH parameter value [0.1]
p1.gamma parameter value [1]
p1.ref parameter value [4]
p1.ampl parameter value [0.000149261]
sherpa> fit
LVMQT: V2.0
LVMQT: initial statistic value = 4583.05
LVMQT: final statistic value = 83.2873 at iteration 8
      abs.nH  2.4061  10^22/cm^2
      p1.gamma 1.51851
      p1.ampl  0.000241434

sherpa> sherpa.resplot.y log = 0
sherpa> lp 2 fit delchi
```

The resulting plot looks like this . We set "sherpa.resplot.y_log" to 0 to ensure that the Y axis of the residuals plot is in linear, not log, spacing (since the earlier call to set_log set all axes to be in log spacing). The best-fit parameter values are displayed at the end of the fit but can also be displayed using the list_par command:

```
sherpa> list_par

#      Name Type      Value Lnk Frz      Min      Max      Delta
1      abs.nH src      2.4061  0  0      1e-07      10      -1
2      p1.gamma src      1.5185  0  0      -10       10      -1
3      p1.ref  src       4      0  1       1       7.4971  -1
```

Errors & Confidence Levels – Sherpa

```
4          p1.ampl  src  0.00024143  0  0  1.4926e-06  0.014926  -1
5 AutoReadResponse.rmf inst"/data/ciao/rmf.fits"
6 AutoReadResponse.arf inst"/data/ciao/arf.fits"
```

Since we are using a chi-square statistic we can get an idea of how well the model fits using the `GOODNESS` command:

```
sherpa> goodness
Goodness: computed with Chi-Squared Gehrels

DataSet 1: 131 data points -- 128 degrees of freedom.
Statistic value      = 83.2873
Probability [Q-value] = 0.999225
Reduced statistic    = 0.650682
```

See the "[Accessing the FIT results](#)" section of the "Accessing fit results using S-Lang" thread for details of how to read these values into S-Lang variables.

We are now ready to estimate errors on the parameters. In the following sections we illustrate several of the routines that can be used in *Sherpa*.

Errors on individual parameters (projection)

We will use the `projection` method to estimate 1 sigma errors on the gamma parameter of the powerlaw component. To do this we use the `proj()` function provided by `paramest.sl`.

```
sherpa> proj("p1.gamma")
Parameter name:      Curr Val      Limits
p1.gamma            1.51851      -10      10

Report error at this number of sigma from the best-fit value (0:) (1):
Projection complete for parameter: p1.gamma

Computed for projection.sigma = 1

-----
Parameter Name      Best-Fit Lower Bound      Upper Bound
-----
p1.gamma            1.51851  -0.105572      +0.107951

The PROJECTION run has finished for p1.gamma.
The get_proj() command can be used to access the data
of this run.
```

When run the function displays the current best-fit value of the selected parameters – along with the allowed range of the parameter – and will then prompt you for values. The set of questions you are asked depends on what function you are calling, and correspond to the configuration parameters for that particular method. In this case we are using the `PROJECTION` method so we are asked for parameters corresponding to fields in the `sherpa.regproj` structure (as discussed in the `fastopt` parameter, the "fast" option is hidden and defaults to "yes").

If you want the 90% confidence limits on this parameter then re-run the function and answer 1.6 at the prompt, or set its value by giving it as an addition parameter to the `proj()` function:

```
sherpa> proj("p1.gamma abs.nh", "sigma=1.6")
Parameter name:      Curr Val      Limits
p1.gamma            1.51851      -10      10
abs.nh              2.4061      1e-07      10
```

Errors & Confidence Levels – Sherpa

```
Projection complete for parameter: abs.nH
Projection complete for parameter: pl.gamma

Computed for projection.sigma = 1.6
-----
Parameter Name      Best-Fit Lower Bound      Upper Bound
-----
abs.nH              2.4061  -0.240423  +0.260944
pl.gamma            1.51851 -0.167618  +0.174267

The PROJECTION run has finished for pl.gamma abs.nh.
The get_proj() command can be used to access the data
of this run.
```

We also asked for the error on the nH parameter of the absorption model. Note that the order of the parameters in the screen output matches that given by `list_par()` and not the order specified in the function call.

To estimate errors on all the thawed parameters call the function with the list of names set to " ":

```
sherpa> proj("")
Parameter name:      Curr Val      Limits
abs.nH              2.4061      1e-07      10
pl.gamma            1.51851     -10        10
pl.ampl             0.000241434 1.49261e-06 0.0149261

Report error at this number of sigma from the best-fit value (0:) (1.6):
Projection complete for parameter: abs.nH
Projection complete for parameter: pl.gamma
Projection complete for parameter: pl.ampl

Computed for sherpa.proj.sigma = 1.6
-----
Parameter Name      Best-Fit Lower Bound      Upper Bound
-----
abs.nH              2.4061  -0.240423  +0.260944
pl.gamma            1.51851 -0.167618  +0.174267
pl.ampl             0.000241434 -1.11634e-05 +1.14954e-05

The PROJECTION run has finished for abs.nH pl.gamma pl.ampl.
The get_proj() command can be used to access the data
of this run.
```

See the "[Accessing the PROJECTION results](#)" section of the "Accessing fit results using S-Lang" thread for details of how to read these values into S-Lang variables.

The `unc()` and `cov()` routines – which correspond to the UNCERTAINTY and COVARIANCE methods respectively – behave similarly.

How does the fit surface vary for a parameter (interval–projection)?

It can be useful to know how the fit statistic varies as the parameter of interest varies: if it is not approximately parabolic then error estimates may be meaningless. The INTERVAL-PROJECTION and INTERVAL-UNCERTAINTY methods produce such plots, using the projection and uncertainty methods respectively. The `paramest.sl` script provides two functions – `iproj()` and `iunc()` – that can be used

to call these routines.


Here we use `iproj()` to see how the fit statistic varies with the `gamma` parameter of the power law component. Since we already know that the 90% errors are approximately ± 0.2 we choose to set the axis range manually:

```
sherpa> iproj("pl.gamma")
Parameter name:      Curr Val      Limits
pl.gamma            1.51851      -10      10

Should the limits be calculated automatically? (yes): no
Minimum value for pl.gamma (-10:10) (0): 1
Maximum value for pl.gamma (-10:10) (0): 2
Should the X-axis be evaluated using logarithmic spacing? (no):
Number of points to evaluate along the X axis (1:) (20):
Interval-Projection: grid size set by user.
                    outer grid loop 20% done...
                    outer grid loop 40% done...
                    outer grid loop 60% done...
                    outer grid loop 80% done...

The INTERVAL-PROJECTION run has finished for pl.gamma.
The get_intproj() command can be used to access the data
of this run.

sherpa> ticks maj y 10
sherpa> ticks min y 5
sherpa> redraw
```

The resulting plot looks like this  (the calls to the `TICKS` command are to add extra numeric labels to the Y axis since the default settings for this plot are not too helpful). The "confidence intervals" table in "[ahelp projection](#)" list a range of common confidence levels and the corresponding change in chi-square values (i.e. the statistic value on the Y axis in this plot).

See the "[Accessing the INTERVAL-PROJECTION results](#)" section of the "Accessing fit results using S-Lang" thread for an example of how to convert this plot into one of delta Chi squared versus parameter value.

The `iunc()` routine – which corresponds to the [INTERVAL-UNCERTAINTY](#) method – behaves similarly.

How are two parameters correlated (region-projection)?

The two interesting parameters in this model are the slope of the powerlaw (`pl.gamma`) and the column density of absorbing material (`abs.nh`). In this section we use the `rproj()` function – which calls the [REGION-PROJECTION](#) command of *Sherpa* – to see whether the two parameters are correlated.

From our earlier run we know that the 90% errors on the two parameters – when evaluated *independently* – are approximately 1.3–1.8 (`gamma`) and 2.1–2.7 (`nH`). However we decide to let the routine calculate limits itself, and choose to display contours at the 1 and 1.6 sigma level (68.3% and 90% confidence levels).

```
sherpa> restore regproj
sherpa> rproj("pl.gamma abs.nh")
Parameter name:      Curr Val      Limits
pl.gamma            1.51851      -10      10
abs.nh              2.4061      1e-07      10


Should the limits be calculated automatically? (yes):
Multiply estimated grid limits by this factor (3):
```


Errors & Confidence Levels – Sherpa

```
Should the X-axis be evaluated using logarithmic spacing? (no):
Number of points to evaluate along the X axis (1:) (10):
Should the Y-axis be evaluated using logarithmic spacing? (no):
Number of points to evaluate along the Y axis (1:) (10):
Comma-separated list of contour levels in units of sigma (1,2,3): 1,1.6
Region-Projection: computing grid size with covariance...done.
                    outer grid loop 20% done...
                    outer grid loop 40% done...
                    outer grid loop 60% done...
                    outer grid loop 80% done...

Minimum: 83.2873
Levels are: 85.5833 87.7093

The REGION-PROJECTION run has finished for pl.gamma and abs.nh.
The get_regproj() command can be used to access the surface data
of this run.
```

The resulting plot looks like this . The "restore_regproj" call resets the fields of the sherpa.regproj variable to their default values.


The automatically-chosen limits have resulted in a poor-quality plot: there are not enough data points close to the best-fit location hence the contours do not accurately reflect the confidence region. The easiest way to change this is to re-run the function and increase the number of points; we also elect to use a smaller parameter range along both axes to reduce the amount of wasted computation.

```
sherpa> rproj("pl.gamma abs.nh")
Parameter name:      Curr Val      Limits
pl.gamma            1.51851      -10      10
abs.nh              2.4061       1e-07    10


Should the limits be calculated automatically? (yes): no
Minimum value for pl.gamma (-10:10) (0): 1.2
Maximum value for pl.gamma (-10:10) (0): 1.9
Minimum value for abs.nh (1e-07:10) (0): 2
Maximum value for abs.nh (1e-07:10) (0): 2.8
Should the X-axis be evaluated using logarithmic spacing? (no):
Number of points to evaluate along the X axis (1:) (10): 21
Should the Y-axis be evaluated using logarithmic spacing? (no):
Number of points to evaluate along the Y axis (1:) (10): 21
Comma-separated list of contour levels in units of sigma (1,1.6):
Region-Projection: grid size set by user.
                    outer grid loop 20% done...
                    outer grid loop 40% done...
                    outer grid loop 60% done...
                    outer grid loop 80% done...

Minimum: 83.2873
Levels are: 85.5833 87.7093

The REGION-PROJECTION run has finished for pl.gamma and abs.nh.
The get_regproj() command can be used to access the surface data
of this run.
```

The resulting plot looks like this . Although the results are much better the contours still do not appear smooth. We now try changing the `chips.mingridsize` value to see whether this will improve the appearance of the plot:

```
sherpa> store conf.tmp
sherpa> chips.mingridsize = 100
sherpa> restore conf.tmp
```

The resulting plot looks like this . The reason for using the STORE/RESTORE commands is because the contour plot needs to be re-created to pick up any change in the `chips.mingridsize` parameter; calling `redraw` is not enough. So this means either re-running the REGION-PROJECTION – which can take a lot of time – or using the *ChIPS* store file. Note that the file `conf.tmp` is left in the current working directory

How are two parameters correlated (region-projection)?

by this sequence of commands.

See the "[Accessing the REGION-PROJECTION results](#)" section of the "Accessing fit results using S-Lang" thread for details of how to read these values into S-Lang variables.

The `runc()` routine – which corresponds to the [REGION-UNCERTAINTY](#) method – behaves similarly.

Other Uses of the Script

In this thread we showed several of the routines available in `paramest.sl`. The complete list, given in Table 1, is explained in the [help file](#).

Table 1. Routines Available in `paramest.sl`

Routine	Sherpa Command
<code>rproj()</code>	REGION-PROJECTION
<code>runc()</code>	REGION-UNCERTAINTY
<code>iproj()</code>	INTERVAL-PROJECTION
<code>iunc()</code>	INTERVAL-UNCERTAINTY
<code>proj()</code>	PROJECTION
<code>unc()</code>	UNCERTAINTY
<code>covar()</code>	COVARIANCE

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

Image 1: Source spectrum

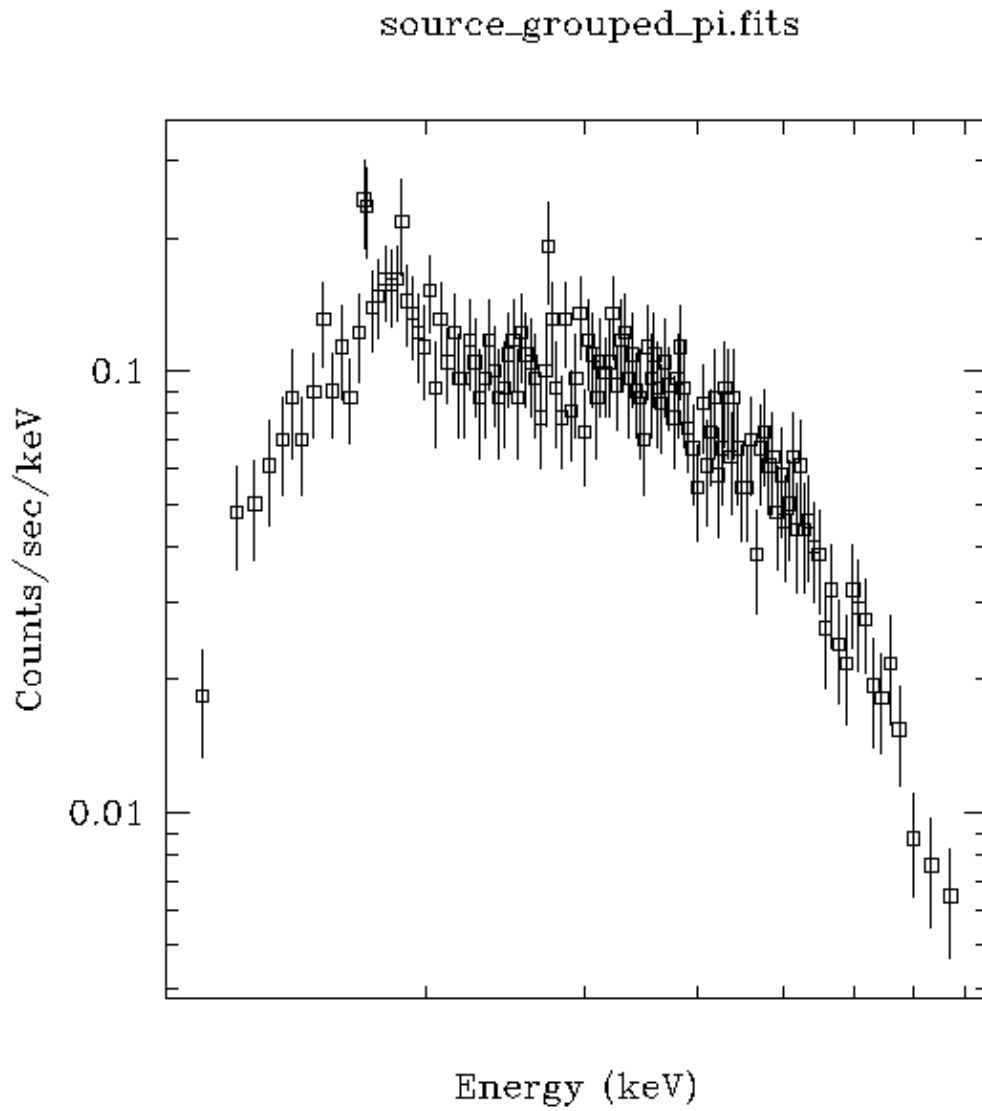


Image 2: Best fit model with residuals

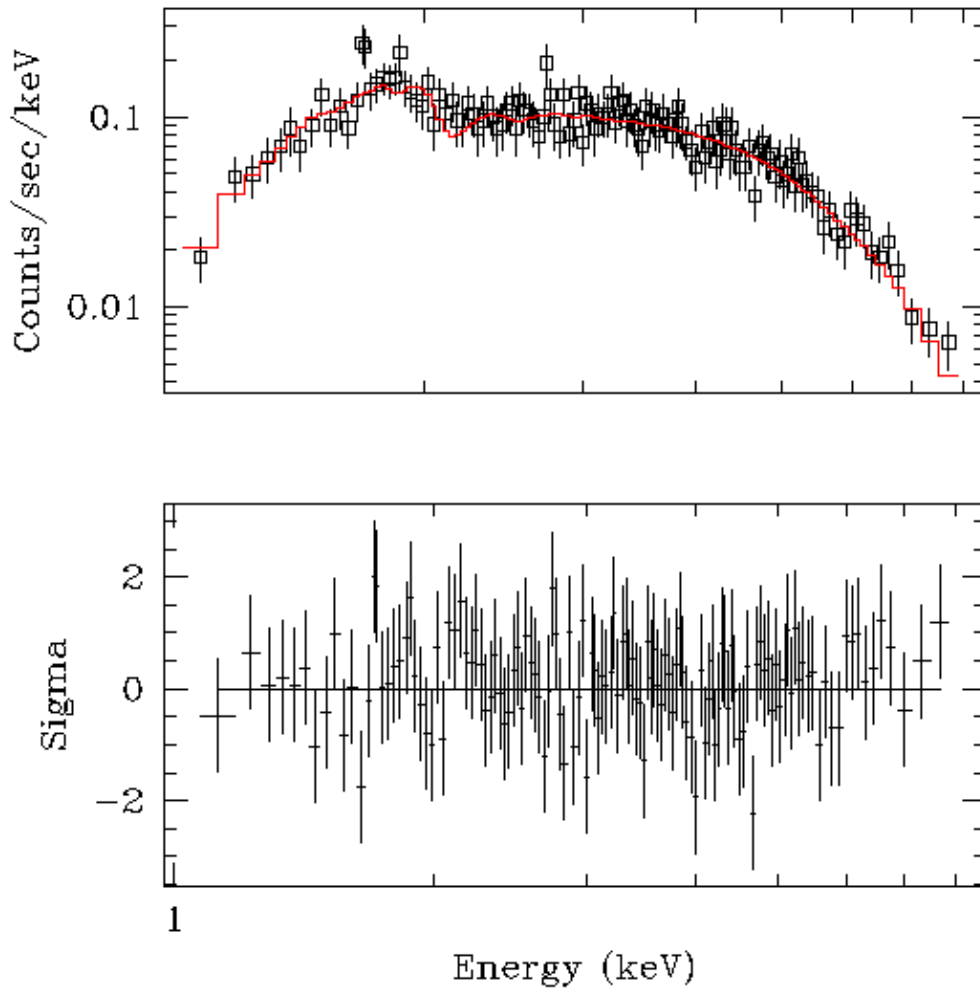


Image 3: Plot of interval–projection results

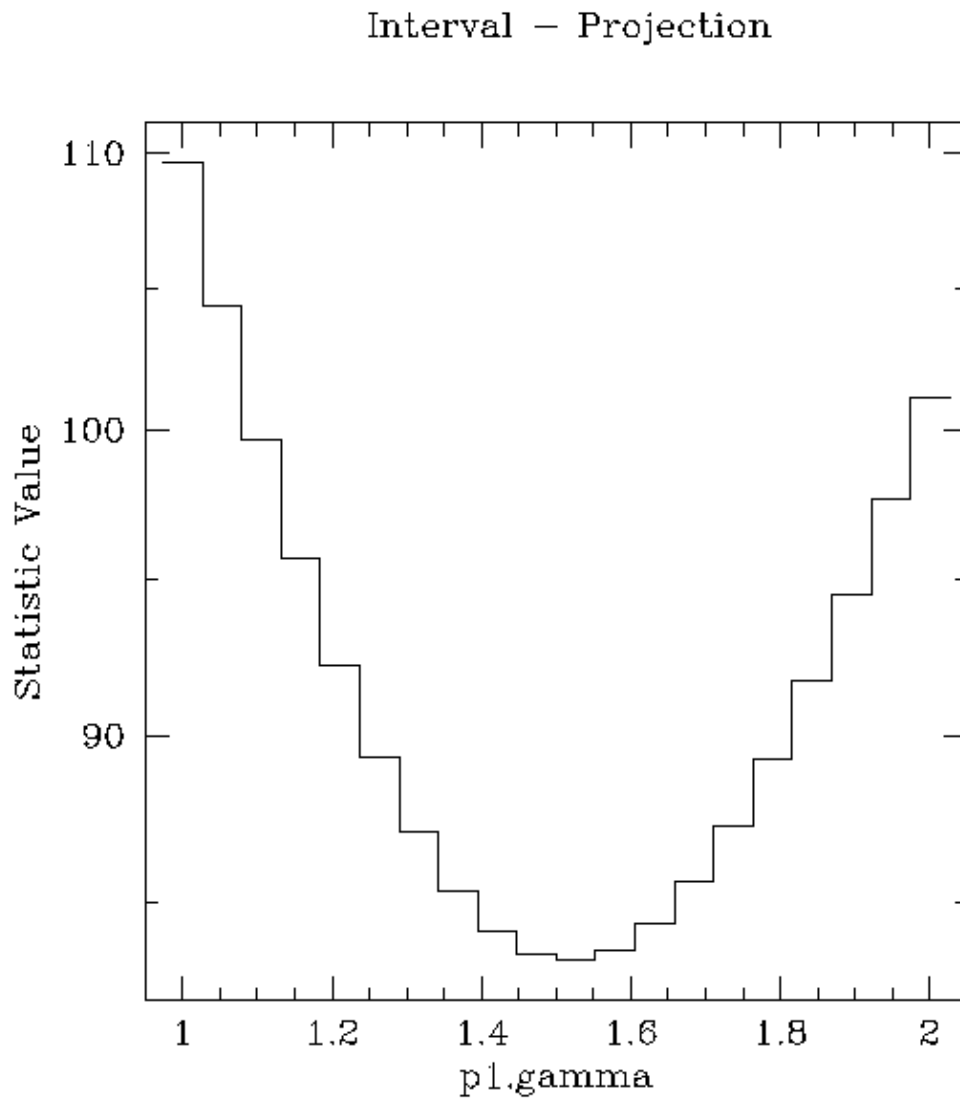
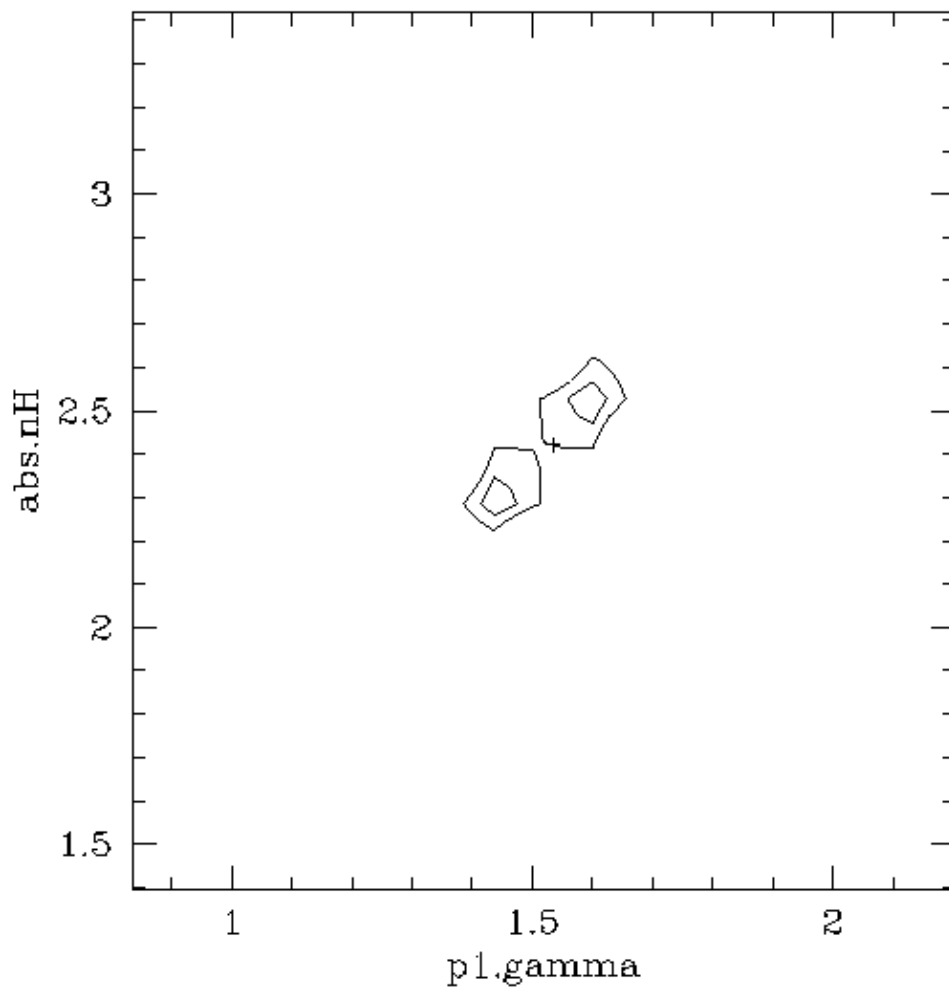


Image 4: Plot of region–projection results

Confidence Region – Projection

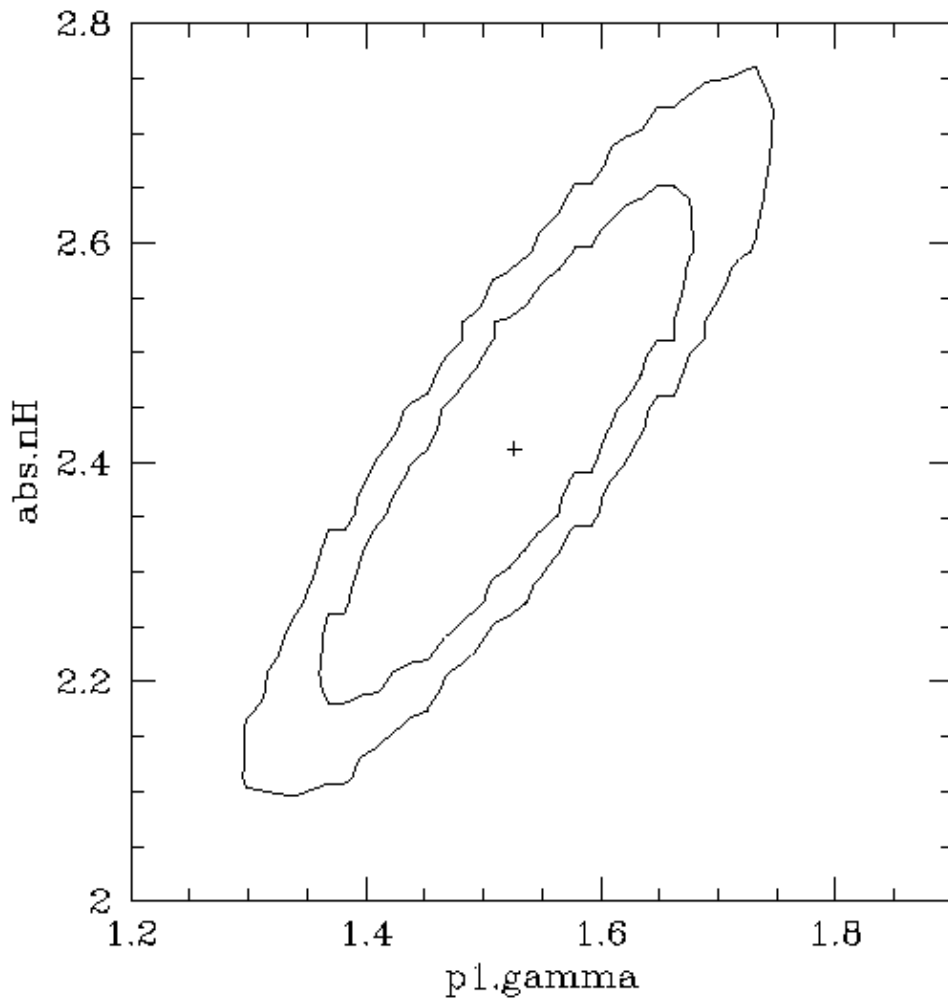


This plot shows the results of the `rproj()` call using the default values: 10 points were used along each axis and the range was calculated automatically. The two contours are drawn at the 68.3% and 90% confidence levels.

From the PROJECTION runs on the individual parameters we expect the 90% confidence range to be 1.3–1.8 and 2.1–2.7; the automatically calculated range is larger than this which accounts for the poor quality of the contours. The plot needs to be re-evaluated using more points, and with a better choice of the axes.

Image 5: Improved region–projection results (chips.mingridsize=50)

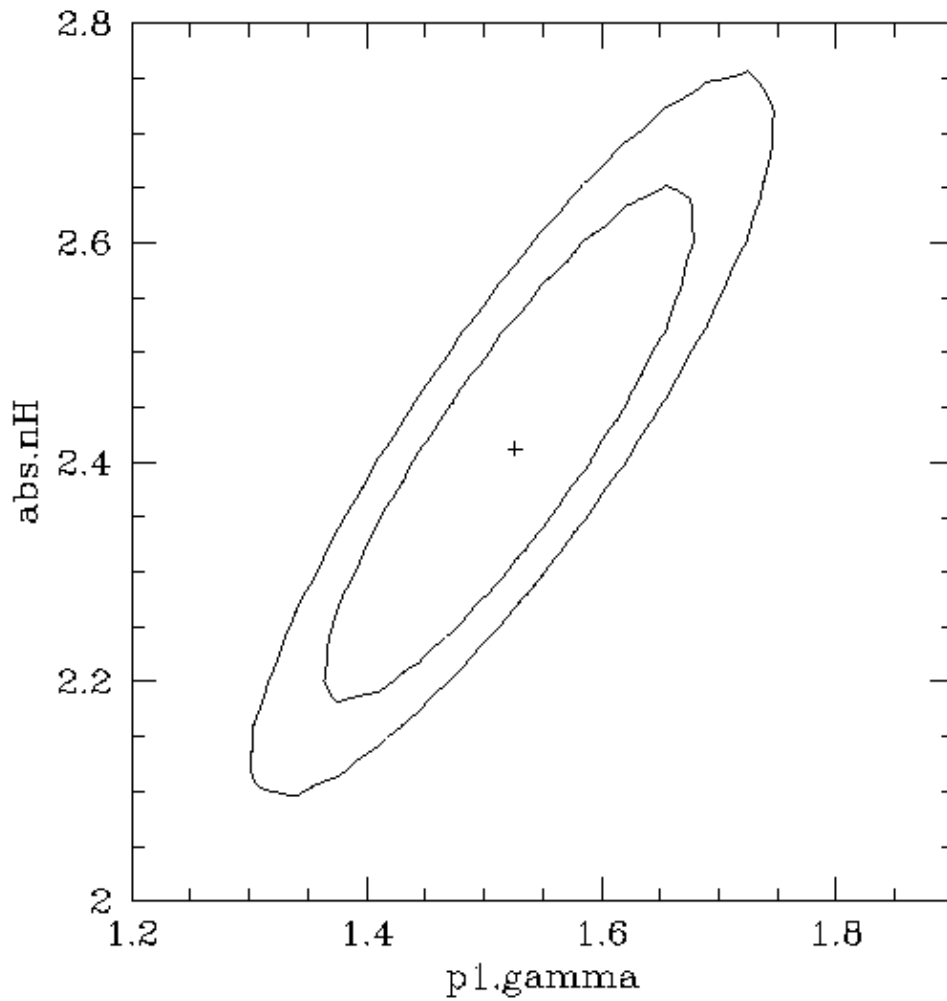
Confidence Region – Projection



The results are greatly improved by using more points along each axis and restricting the ranges of the two parameters used for the contour plot. However the contours still do not appear smooth.

Image 6: Improved region–projection results (chips.mingridsize=100)

Confidence Region – Projection



By increasing the `chips.mingridsize` field to 100 we have been able to create a sensible–looking contour plot.