Step−by−Step Guide to Estimating Errors and Confidence Levels

Sherpa Threads (CIAO 3.4)

Table of Contents

Step−by−Step Guide to Estimating Errors and Confidence Levels

Sherpa Threads

Overview

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

Synopsis:

This thread repeats the steps of the Estimating Errors and Confidence Levels thread, this time using the "native" *Sherpa* interface to the routines, rather than the functions provided by the `paramest.sl` script.

Read this thread if:

You want to estimate errors or confidence levels for parameters in a fit (to data of any dimensionality) and do not want to use the simple interface provided by the routines in the `paramest.sl` script.

Related Links:

- The ahelp files for the *Sherpa* routines: `COVARIANCE`, `UNCERTAINTY`, `PROJECTION`, `INTERVAL−UNCERTAINTY`, `INTERVAL−PROJECTION`, `REGION−UNCERTAINTY`, and `REGION−PROJECTION`.
- The "Estimating Errors and Confidence Levels" thread provides a greater level of description of the steps taken here.
- 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

All that is needed is to download the sherpa.tar.gz file, as described in the "Getting Started" thread.

Please review the "Estimating Errors and Confidence Levels" thread since it describes the following steps in greater detail than that presented below.

Find the best fit

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. We now set up the source model − an absorbed power law − and fit it:

```
sherpa> source = xswabs[abs] * powlaw1d[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. The GOODNESS command can be used to find out how well the model fits the data (since the statistic is a variant of Chi squared rather than the Cash formalism):

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

```
DataSet 1: 131 data points -- 128 degrees of freedom.
 Statistic value       = 83.2877
 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.

Errors on individual parameters (projection)

We will use the projection method to estimate 1 sigma errors on the `gamma` parameter of the powerlaw component. The `restore_proj()` routine is used to ensure that the fields of the `sherpa.proj` variable − which are used by the `PROJECTION` command − are reset to their default values.

```
sherpa> restore_proj
sherpa> projection p1.gamma
Projection complete for parameter: p1.gamma

Computed for sherpa.proj.sigma = 1
        ---------------------------------------------------------
        Parameter Name      Best-Fit Lower Bound     Upper Bound
        ---------------------------------------------------------
            p1.gamma         1.51851  -0.105572       +0.107951
```

If you want the 90% confidence limits on this parameter then you need to set the `sigma` field of the `sherpa.proj` variable to `1.6` (see the "Confidence Intervals" table in "`ahelp projection`" for the relationship between sigma and confidence level).

```
sherpa> list_proj
Parameter       Current         Default                        Description
-----------------------------------------------------------------------------
fast                  1               1    Switch to LM/simplex: 0(n)/1(y)
sigma                 1               1                     Number of sigma
sherpa> sherpa.proj.sigma = 1.6
sherpa> projection p1.gamma abs.nh
Projection complete for parameter: abs.nH
Projection complete for parameter: p1.gamma

Computed for sherpa.proj.sigma = 1.6
        ---------------------------------------------------------
        Parameter Name      Best-Fit Lower Bound     Upper Bound
        ---------------------------------------------------------
           abs.nH             2.4061  -0.240423       +0.260944
            p1.gamma         1.51851  -0.167618       +0.174267
```

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 call to `projection`.

To estimate errors on all the thawed parameters call `projection` with no parameter names. Since `sherpa.proj.sigma` is still set to 1.6 the following calculates the 90% confidence limits for all the thawed parameters:

```
sherpa> projection
Projection complete for parameter: abs.nH
```

```
Projection complete for parameter: p1.gamma
Projection complete for parameter: p1.ampl

Computed for sherpa.proj.sigma = 1.6
        ---------------------------------------------------------
        Parameter Name      Best−Fit Lower Bound     Upper Bound
        ---------------------------------------------------------
           abs.nH             2.4061  −0.240423       +0.260944
            p1.gamma         1.51851  −0.167618       +0.174267
            p1.ampl      0.000241434  −1.11634e−05    +1.14954e−05
```

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 `UNCERTAINTY` and `COVARIANCE` commands behave similarly, although the fields in the state object for the different methods are different.

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

Here we use `INTERVAL−PROJECTION` method 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> restore intproj
sherpa> sherpa.intproj.arange = 0
sherpa> sherpa.intproj.min = 1
sherpa> sherpa.intproj.max = 2
sherpa> list intproj
Parameter       Current         Default                        Description
----------------------------------------------------------------------------
fast                  1               1    Switch to LM/simplex: 0(n)/1(y)
expfac                3               3          Expansion factor for grid
arange                0               1              Auto−range: 0(n)/1(y)
min                   1               0                      Minimum value
max                   2               0                      Maximum value
log                   0               0             Log−spacing: 0(n)/1(y)
nloop                20              20              Number of grid points
sigma                 1               1                    Number of sigma
sherpa> intproj p1.gamma
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...
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 `INTERVAL−UNCERTAINTY` command behaves similarly, although the fields in the state object for the two methods are different.

How are two parameters correlated (region−projection)?

In this section we use the `REGION−PROJECTION` method of *Sherpa* to see whether the `p1.gamma` and `abs.nh` 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> sherpa.regproj.sigma = [1,1.6]
sherpa> list regproj
Parameter       Current         Default                        Description
------------------------------------------------------------------------------
fast                  1               1    Switch to LM/simplex: 0(n)/1(y)
expfac                3               3          Expansion factor for grid
arange                1               1              Auto−range: 0(n)/1(y)
min               [0,0]           [0,0]          Minimum values, each axis
max               [0,0]           [0,0]          Maximum values, each axis
log               [0,0]           [0,0]  Log−spacing: 0(n)/1(y), each axis
nloop           [10,10]         [10,10]   Number of grid points, each axis
sigma           [1,1.6]         [1,2,3]      Number of sigma, each contour
sherpa> regproj p1.gamma abs.nh
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 resulting plot looks like this.

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> sherpa.regproj.arange = 0
sherpa> sherpa.regproj.min = [1.2,2]
sherpa> sherpa.regproj.max = [1.9,2.8]
sherpa> sherpa.regproj.nloop = [21,21]
sherpa> regproj p1.gamma abs.nh
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 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 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 `REGION−UNCERTAINTY` command behaves similarly, although the fields in the state object for the two methods are different.

History

14 Jan 2005	reviewed for CIAO 3.2: no changes
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/confidence_manual/ Last modified: 1 Dec 2006

Image 1: Source spectrum

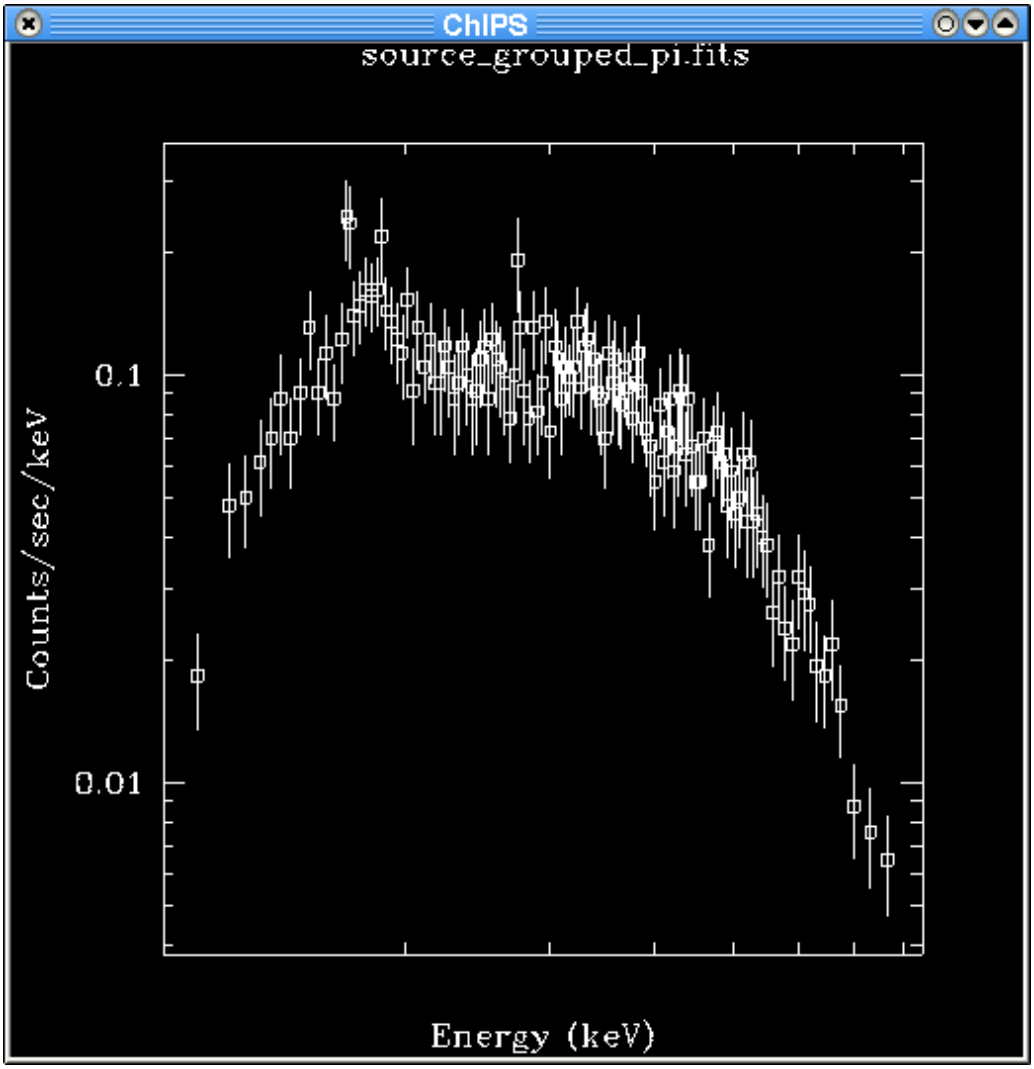


Image 2: Best fit model with residuals

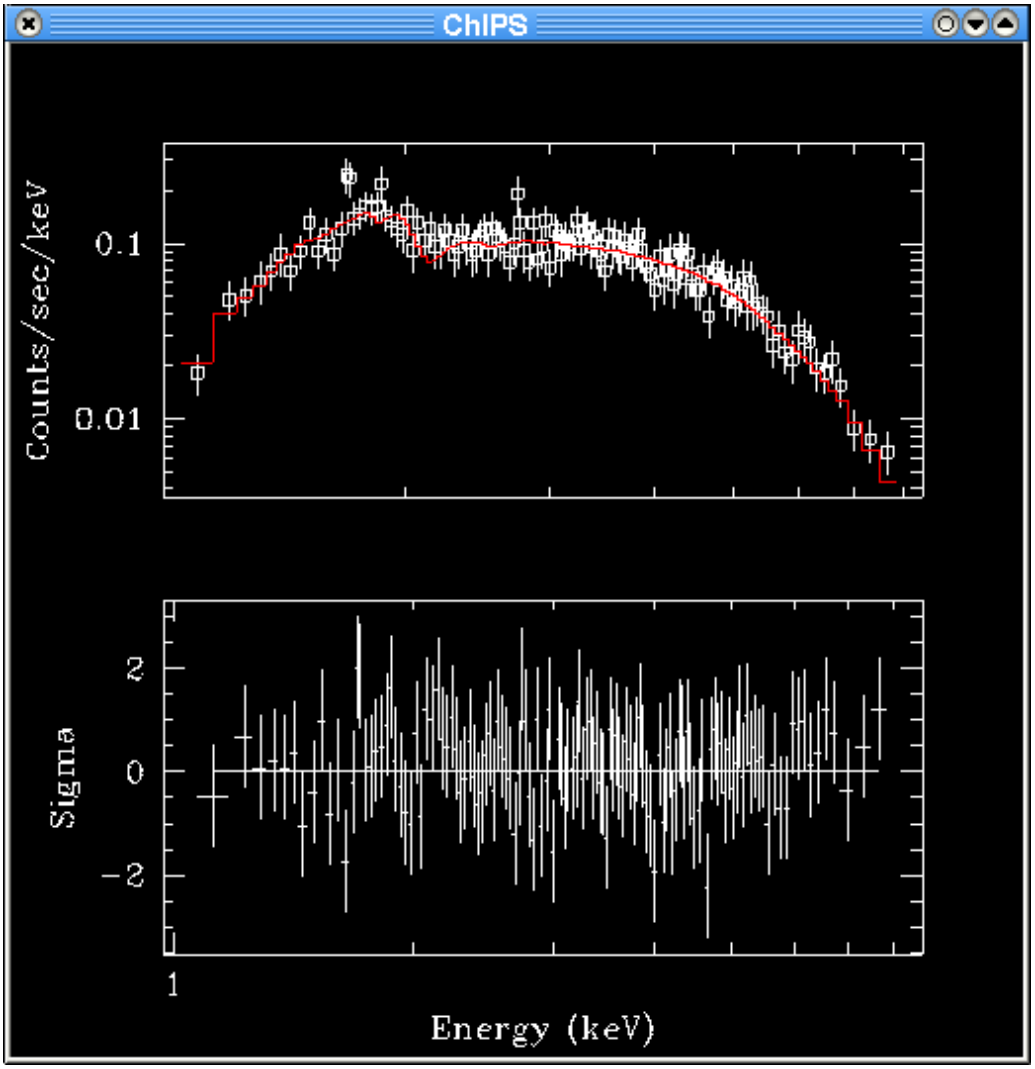


Image 3: Plot of interval−projection results

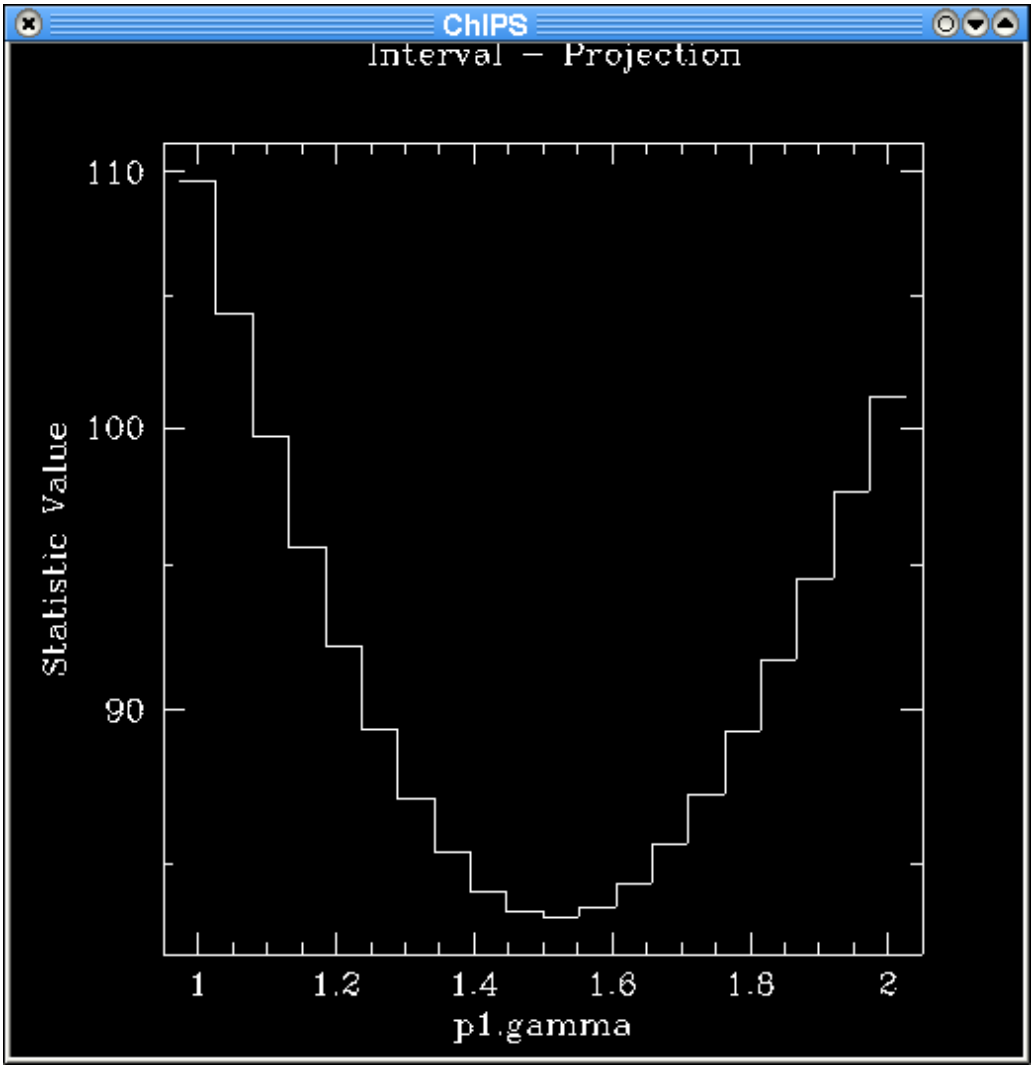
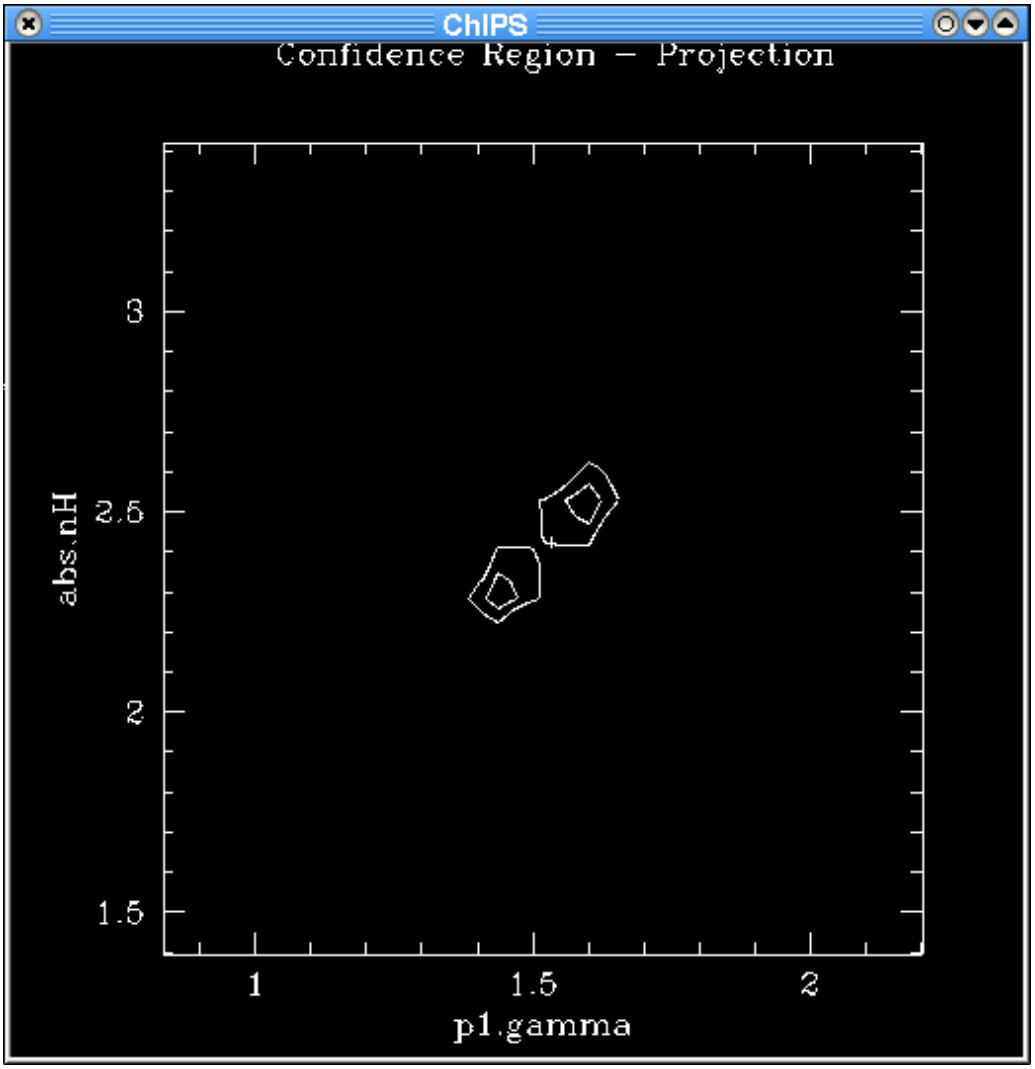


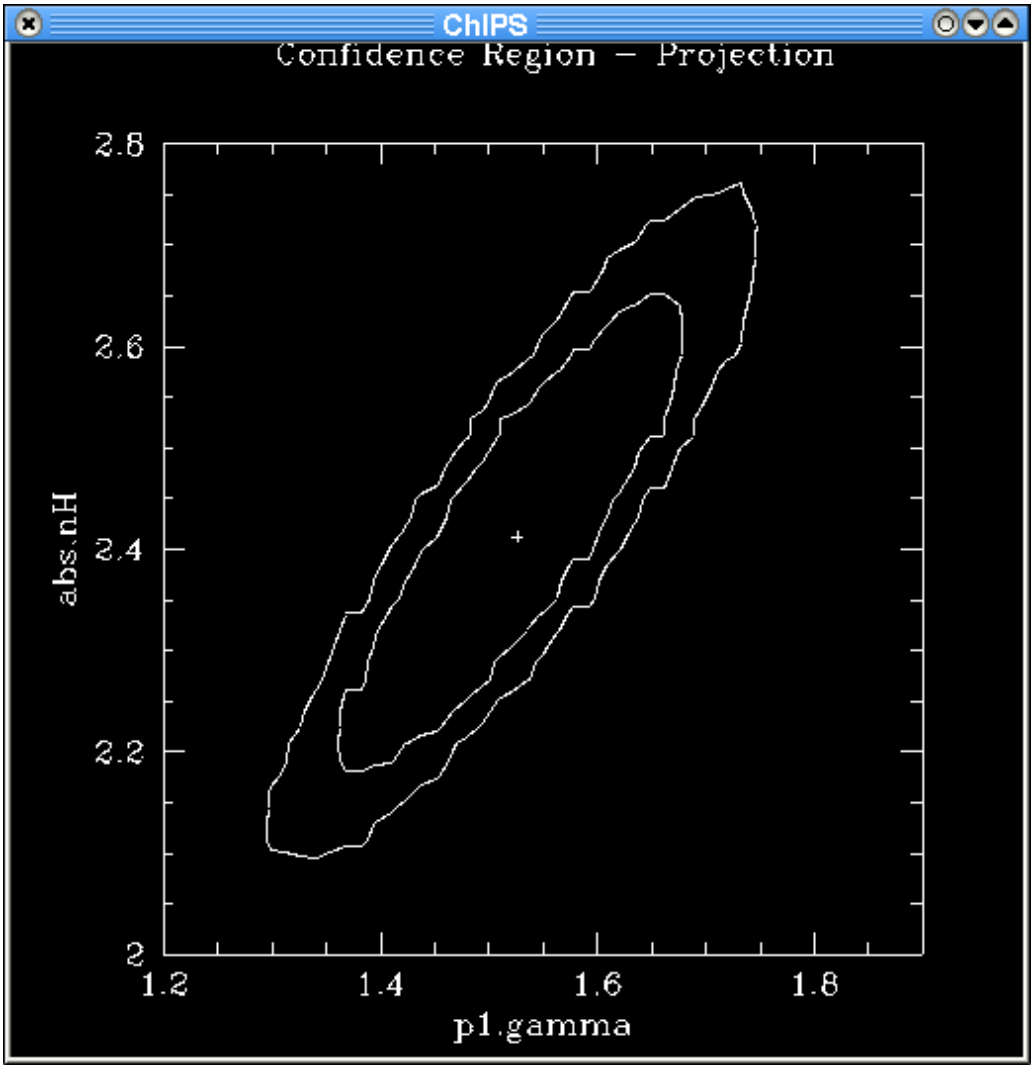
Image 4: Plot of region−projection results



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

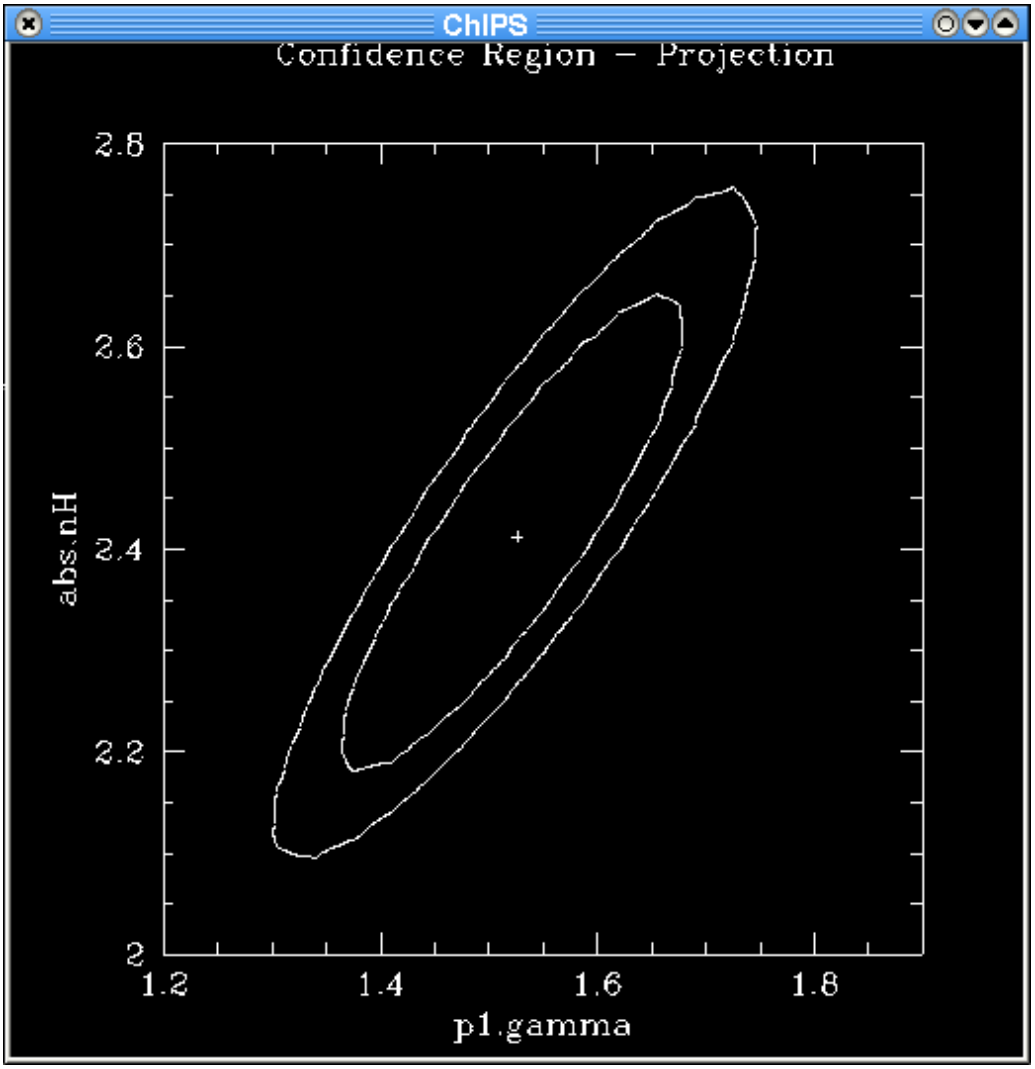
From the `PROJECTION` runs on the individual parameters we expect the 90% confidence range to be `1.3−1.8` and `2.1−2.7`; the automatically calculated range is larger than this which accounts for the poor quality of the contours. The plot needs to be re−evaluated using more points, and with a better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

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



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)



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