Simulating 1−D Data: the Sherpa FAKEIT Command

Sherpa Threads (CIAO 3.4)

Table of Contents

Simulating 1−D Data: the Sherpa FAKEIT Command

Sherpa Threads

Overview

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

Synopsis:

Sherpa can be used to simulate 1−D data, using the `FAKEIT` command. This thread provides detailed examples.

Proceed to the HTML or hardcopy (PDF: A4 | letter) version of the thread.

Getting started

Please follow the "Sherpa Threads: Getting Started" thread.

Introduction to FAKEIT

The `FAKEIT` command creates a simulated 1−D dataset calculated using a previously defined source model expression and a previously defined instrument model. Poisson deviates are then added to the modeled data. If a background file is supplied, then the backscale correction is used, and background is added to the simulated data before Poisson noise is added.

Note that all *Sherpa* library models can be used in the simulations. In addition, models created and implemented by a user may be included. Individual models can be combined into composite models, and parameters can be linked across the model components.

Note also that both the source and instrument models ***must*** be predefined. Currently, only instrument models that are defined using redistribution matrix (RMF) and ancillary (ARF) 1−D spectral files may be used with `FAKEIT`.

There are several steps involved in creating simulations:

1. Defining an instrument model using a redistribution matrix file and an ancillary response file.
2. Defining a background by reading a background data file or specifying background model.
3. Setting `FAKEIT` parameters (`TIME` and `BACKSCALE`).
4. Defining a source model expression.
5. Running the simulation using `FAKEIT`.
6. Defining the model normalization for the simulated data, if desired.

7. Writing the simulated data to output files.

This thread illustrates each of these steps with an example. This thread will also illustrate use of the `FAKEIT` command when a PHA data file has been previously read. In addition, the thread will show how a dataset placed into memory by `FAKEIT` can be plotted and fit. The thread concludes with an additional overall example.

Defining an Instrument Model

The default instrument response files for 1−D simulations of Chandra data include the redistribution matrix (RMF) and the ancillary (ARF) files in standard FITS format. These files can be created with the `mkrmf` and `mkarf` tools, or with a script as described in the Using specextract to Extract ACIS Spectra and Response Files thread.

The following *Sherpa* command defines an instrument model using the specified input response files:

```
sherpa> RSP[instrumentA](dataA_rmf.fits, dataA_arf.fits)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
```

Note that this also sets the name of the instrument model to `instrumentA`:

```
sherpa> SHOW instrumentA
rsp1d[instrumentA]
    Param   Type  Value
    −−−−−   −−−−  −−−−−
 1    rmf string: "dataA_rmf.fits" (N_E=1180,N_PHA=512)
 2    arf string: "dataA_arf.fits" (N_E=1180)
```

Now the instrument model must be assigned:

```
sherpa> INSTRUMENT = instrumentA
```

Define a Background

A background model or background data can be used in the simulations. Note that if PHA data are input as a background file, TIME and BACKSCALE parameters will have default settings corresponding to the values of the header keywords EXPTIME and BACKSCAL. These may of course be changed.

Background data and/or models are treated as follows in FAKEIT:

1. If a background model stack is defined, it is evaluated on the source data grid, and the resulting background amplitudes are added to the source amplitudes (taking into account differences in exposure time and backscale). Faked data are then sampled given the sum. If background data exist, they are not altered. If the source dataset was background−subtracted prior to the command FAKEIT being issued, it will not be background−subtracted afterwards.
2. If no background model stack is defined, and the data are background−subtacted, then the source model stack is evaluated directly, and the new, faked data are background−subtracted. Note that subsequently issuing an `UNSUBTRACT` command is unwise, because the unsubtracted data will not be integer counts data.

3. If no background model stack is defined, and the data are not background−subtracted, then the source model stack is evaluated directly, and the (properly scaled) background data are added to the faked data.

Reading a background data file

A PHA−type background file is read so that the exposure time and backscale information may be taken into account in the simulations. The exposure time and backscale from the background file is only used for apropriate scaling of the input background data.

```
sherpa> BACK dataA_bkg.pha
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
```

The instrument model must be explicitly set for the background as well:

```
sherpa> INSTRUMENT back = instrumentA
```

The `SHOW` command gives the current *Sherpa* status:

```
sherpa> SHOW

Optimization Method: Levenberg−Marquardt
Statistic:           Chi−Squared Gehrels

−−−−−−−−−−−−−−−−−
Input data files:
−−−−−−−−−−−−−−−−−


  Background 1: dataA_bkg.pha pha.
  Total Size: 512 bins (or pixels)
  Dimensions: 1
  Total counts (or values): 2220
  Exposure: 108675.66 sec
  Count rate: 0.020 cts/sec
  Backscal: 0.044189

  The data are NOT background subtracted.


−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Defined analysis model stacks:
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

instrument source 1 = instrumentA
instrument back 1 = instrumentA

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Defined instrument model components:
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

rsp1d[instrumentA]
    Param   Type  Value
    −−−−−   −−−−  −−−−−
 1    rmf string: "dataA_rmf.fits" (N_E=1180,N_PHA=512)
 2    arf string: "dataA_arf.fits" (N_E=1180)
```

There is no dataset listed (only background) since we have not yet simulated any data. After `FAKEIT` has been run, the information will be put in this empty dataset (*i.e.* dataset number 1).

Define a Background Model Expression

If a background model expression is defined then the background model amplitude will be added to the source amplitudes (taking into account differences in exposure time and backscale). Faked data are then sampled given the sum. A background model is defined in a standard way with the model name given in the bracket. In this example, the background source model is a combination of 1−D polynomial and Gaussian models:

```
sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> BACKGROUND = polynom1d[bkgA]+gauss1d[bkgB]
sherpa> bkgA.c0=6.4e−5
sherpa> bkgA.c1=2.3e−5
sherpa> bkgA.c3=1.4e−05
sherpa> bkgB.pos=0.057
sherpa> bkgB.ampl=0.003
sherpa> show

..cut..
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Defined analysis model stacks:
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−


instrument source 1 = instrumentA
bg 1 = (bkgA + bkgB)
instrument back 1 = instrumentA

..cut..
```

Setting FAKEIT Parameters

The default values for the `FAKEIT` parameters are:

```
sherpa> SHOW FAKEIT
Fakeit exposure time: 1 seconds.
Fakeit backscale: 1
```

Next, these parameter values are set to the desired values for the simulation:

```
sherpa> FAKEIT TIME = 33483.2
sherpa> FAKEIT BACKSCALE = 0.044189
sherpa> SHOW FAKEIT
Fakeit exposure time: 33483.2 seconds.
Fakeit backscale: 0.044189
```

Note that in this example the simulated data and the background data will have the same backscale value, which means that the area of the simulated source will be the same as in the supplied background file. If the backscale is set to 1 (the default value), then the scaling of the area is assumed based on backscale from the header of the background file.

Defining a Source Model Expression

The source model expression is defined in the standard way, with the model component's assigned name given in square brackets:

```
sherpa> SOURCE = POWLAW1D[modelA]
```

The *Sherpa* model library and model syntax are described in detail in the *Sherpa* Reference Manual.

The model parameters can then be set as desired:

```
sherpa> modelA.gamma=2

sherpa> SHOW modelA
powlaw[modelA]  (integrate: on)
    Param   Type      Value        Min        Max                 Units
    -----   ----      -----        ---        ---                 -----
 1  gamma thawed          2        -10         10
 2    ref frozen          1    -1e+120     1e+120
 3   ampl thawed          1          0     1e+120
```

Running the Simulation Using FAKEIT

A simple command runs the simulation:

```
sherpa> FAKEIT
```

The `SHOW` command can then be used to check the *Sherpa* status:

```
sherpa> SHOW

Optimization Method: Levenberg−Marquardt
Statistic:           Chi−Squared Gehrels

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

Data 1: fake pha.
Total Size: 512 bins (or pixels)
Dimensions: 1
Total counts (or values):    4.986073e+06
Exposure: 33483.20 sec
Count rate: 148.913 cts/sec
Backscal: 0.044189

(etc)
```

The default filename for the simulated dataset is "`fake pha.`". This file is ***not*** saved to disk until the command `WRITE DATA` is given (see the Writing The Simulated Data To Output Files section).

Defining the Model Normalization for the Simulated Data

Simulated data should have a normalization which agrees, for example, with the observed source flux. If the flux of the simulated model is known, it can be easily converted to the model parameter values. This requires simulating the data with the default parameters, calculating the flux, and then rescaling the model parameters. We first describe the steps and then use the S−lang functions to demonstrate how efficiently renormalize the model. S−lang allows to simplify the simulations scripts as describe below.

Assuming that we have just simulated data using default parameters and modelA.ampl=1, so we next calculate the flux:

```
sherpa> EFLUX (2:10)
Flux for source dataset 1: 2.57862e−09 ergs/cm**2/s
```

To find the normalization, given a source flux of say 1.0e−13 `ergs cm`$^{-2}$ `s`$^{-1}$, we divide the true source flux (1.0e−13) by the calculated flux for the above default source model (2.57862e−09); the normalization (`modelA.ampl`) was 1 `photon keV`$^{-1}$ `cm`$^{-2}$ `s`$^{-1}$:

```
sherpa> 1.0e−13 / 2.57862e−09
3.87804e−05
```

So, the normalization of the source model may be adjusted as follows:

```
sherpa> modelA.ampl = 3.87804e−05
```

and `FAKEIT` is run again:

```
sherpa> FAKEIT
sherpa> FLUX (2:10)
Flux for source dataset 1: 1.55122e−05 photons/cm**2/s
sherpa> EFLUX (2:10)
Flux for source dataset 1: 9.99999e−14 ergs/cm**2/s
```

Or, if the count rate of the source in a given instrument is known in advance (*e.g.* it can be obtained from `PIMMS`, or from previous observations), then a simple scaling can be used to adjust the normalization of the source model.

For example, say the known count rate is 0.4 cts/sec. From the first `FAKEIT` run (or a repeat run, with `ampl` returned to 1.0), we note the non−normalized count rate:

```
sherpa> modelA.ampl = 1.0
sherpa> FAKEIT

sherpa> SHOW
.
.
Data 1: fake pha.
Total Size: 512 bins (or pixels)
Dimensions: 1
Total counts (or values):    4.979089e+06
Exposure: 33483.20 sec
Count rate: 148.704 cts/sec
Backscal: 0.044189
.
.
```

Using this non−normalized count rate (148.704 `cts/sec`), the following scaling factor is obtained:

```
sherpa> 0.4 / 148.704
0.00268991
```

So, the normalization of the source model may be adjusted as follows:

```
sherpa> modelA.ampl = 0.00268991
```

and `FAKEIT` is run again:

```
sherpa> FAKEIT

sherpa> SHOW
.
.
Data 1: fake pha.
Total Size: 512 bins (or pixels)
Dimensions: 1
Total counts (or values): 78869
Exposure: 33483.20 sec
Count rate: 2.355 cts/sec
Backscal: 0.044189
.
.
```

Defining Model Normalization using Sherpa S−lang Module Functions

All the steps required to set model normalization can be efficiently done using S−lang Module functiona `get_eflux`, `get_par`, and `set_par`:

```
sherpa> modflux=get_eflux(1,[2,10])

sherpa> print(modflux)
dataset          =  1
range            =  Double_Type[2]
comp             =  NULL
value            =  6.93625e−12
units            =  ergs/cm**2/s

sherpa> obsflux=1.e−13

sherpa> foo=get_par("modelA.ampl")
sherpa> print(foo)
name             =  modelA.ampl
model            =  powlaw
type             =  src
value            =  0.00268991
min              =  0
max              =  1e+120
delta            =  −1
units            =  NULL
frozen           =  0
linked           =  0
linkexpr         =  NULL

sherpa> foo.value=foo.value*(obsflux/modflux.value)
sherpa> print(foo)
name             =  modelA.ampl
model            =  powlaw
type             =  src
value            =  3.87804e−05
min              =  0
```

```
max              =  1e+120
delta            =  −1
units            =  NULL
frozen           =  0
linked           =  0
linkexpr         =  NULL

sherpa> ()=set_par(foo)

sherpa> show model
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Defined source/background model components:
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

poly1d[bkgA]  (integrate: on)
    Param   Type      Value         Min         Max                 Units
    −−−−−   −−−−      −−−−−         −−−         −−−                 −−−−−
 1     c0 thawed    6.4e−05           0 1.7309e−04
 2     c1 frozen    2.3e−05 −1.183e−03 1.1826e−03
 3     c2 frozen          0  −8.08e−05   8.08e−05
 4     c3 frozen    1.4e−05           0 1.7309e−04
 5     c4 frozen          0           0 1.7309e−04
 6     c5 frozen          0           0 1.7309e−04
 7     c6 frozen          0           0 1.7309e−04
 8     c7 frozen          0           0 1.7309e−04
 9     c8 frozen          0           0 1.7309e−04
10 offset frozen          0 −2.761e−02     14.664

gauss1d[bkgB]  (integrate: on)
    Param   Type      Value         Min         Max                 Units
    −−−−−   −−−−      −−−−−         −−−         −−−                 −−−−−
 1   fwhm thawed     5.6254 5.6254e−02   562.5416
 2    pos thawed    5.7e−02 2.7614e−02     14.664
 3   ampl thawed      3e−03 1.7309e−06 1.7309e−02

powlaw[modelA]  (integrate: on)
    Param   Type      Value         Min         Max                 Units
    −−−−−   −−−−      −−−−−         −−−         −−−                 −−−−−
 1  gamma thawed          2         −10          10
 2    ref frozen          1    −1e+120     1e+120
 3   ampl thawed  3.878e−05           0     1e+120
```

Writing the Simulated Data to Output Files

The simulated data are not automatically saved to disk. The user may choose to write the data as either PHA or ASCII. The PHA header will contain exposure time and backscale values, as well as paths to the RMF, ARF, and background files. However, the ASCII data file will only contain the data points (channel number, counts). Since all of the simulation parameters are lost in an ASCII data file, it is advisable to save the simulated data in PHA format.

This is done with the `WRITE` command:

```
sherpa> WRITE DATA dataA_sim1.pha PHA
Write X−Axis: Bin  Y−Axis: Counts
```

This creates a PHA file with columns of channel and counts. This file may then be grouped using `dmgroup` if necessary.

To save the simulated data to an ASCII data file:

```
sherpa> WRITE DATA dataA_sim1.dat ASCII
Write X−Axis: Energy (keV)  Y−Axis: Flux (Counts/sec/keV)
```

Using FAKEIT with a PHA File

The FAKEIT command will operate whether or not the user has previously read data from a PHA file. If the user has previously read data from a PHA file:

- An instrument model may have been automatically defined if the response files are properly referenced in the header of the data file.
- If the data file's backfile keyword supplies a background file, then it will be used in the simulations.
- Since input PHA files also usually contain the exposure time and backscale keywords pertinent to an observation, this information will also be used by `FAKEIT`.
- The input dataset will be overwritten by the simulated dataset created by `FAKEIT`.

For example, erase all current *Sherpa* settings (or `EXIT` and begin a new session), and then input a PHA data file:

```
sherpa> ERASE ALL
sherpa> SHOW

Optimization Method: Levenberg−Marquardt
Statistic:           Chi−Squared Gehrels

sherpa> DATA dataA.pha
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: using systematic errors specified in the PHA file.
Background data are being input from:
  /sherpa/fakeit/dataA_bkg.pha
RMF is being input from:
  /sherpa/fakeit/dataA_rmf.fits
ARF is being input from:
  /sherpa/fakeit/dataA_arf.fits
```

Loading this PHA data file resulted in an instrument model being automatically defined (using RMF and ARF files), and a background data file being automatically input. And, from the `SHOW FAKEIT` command we see that the `FAKEIT` parameters have also been properly set automatically:

```
sherpa> SHOW FAKEIT
Fakeit exposure time: 33483.2 seconds.
Fakeit backscale: 0.0441895
```

That is, input of this PHA data file has automatically completed the first three steps involved in creating simulations:

- Defining an instrument model using a redistribution matrix file and an ancillary response file.
- Reading a background data file.
- Setting `FAKEIT` parameters (`TIME` and `BACKSCALE`).

The following commands will complete the remaining steps:

- Defining a source model expression.
- Running the simulation using `FAKEIT`.
- Defining the model normalization for the simulated data, if desired.
- Writing the simulated data to output files.

Important note: If your input PHA data are grouped, the simulated data produced by `FAKEIT` will also be grouped. (The grouping *cannot* be removed with `UNGROUP`.) When you write the simulated data to a file via `WRITE DATA ... PHA`, the channels in the output file represent the *grouped* channels. However, no grouping information is written to the file, so when you read it back into *Sherpa*, the bin energies are incorrect. Currently, there is no workaround for this problem, so `WRITE ... PHA` should be used only with *ungrouped* PHA data.

```
sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> SOURCE = POWLAW1D[modelA]
sherpa> modelA.gamma=2
sherpa> FAKEIT
FAKEIT: The current background data have been added to the faked spectrum.
```

Note that the `FAKEIT` command replaces the input dataset with the simulated data:

```
sherpa> SHOW
.
.
Data 1: fake pha.
Total Size: 79 bins (or pixels)
Dimensions: 1
Total counts (or values): 1407
Exposure: 33483.25 sec
Count rate: 0.042 cts/sec
Backscal: 0.044189
.
.
```

The results may be written out to a file:

```
sherpa> WRITE DATA dataA_sim2.pha PHA
Write X−Axis: Bin  Y−Axis: Counts
sherpa> WRITE DATA dataA_sim2.dat ASCII
Write X−Axis: Energy (keV)  Y−Axis: Flux (Counts/sec/keV)
```

Plotting and Fitting Simulated Data

The simulated dataset may be plotted in the same way as any other dataset:

```
sherpa> LPLOT DATA
```

Figure 1 shows the resulting plot.

The simulated dataset may be also filtered and fit in the same way as any other dataset. For example, we next filter the simulated dataset, to include only data between 0.5 and 8.0 keV:

```
sherpa> IGNORE ENERGY :0.5,8:
sherpa> LPLOT DATA
```

Figure 2 shows the resulting plot. Then, we fit the simulated dataset using the same source model expression that had been used to create it:

```
sherpa> SUBTRACT
sherpa> FIT
 LVMQT: V2.0
 LVMQT: initial statistic value = 20.1631
 LVMQT: final statistic value = 19.0384 at iteration 5
        modelA.gamma  2.01041
        modelA.ampl  0.00014407

sherpa> LPLOT FIT
```

Figure 3 shows the resulting plot.

We here examine the fit using the *Sherpa* command `COVARIANCE`:

```
sherpa> COVAR

Computed for sherpa.cov.sigma = 1
        --------------------------------------------------------
        Parameter Name      Best−Fit Lower Bound     Upper Bound
        --------------------------------------------------------
        modelA.gamma         2.01041  −0.134556       +0.134556
        modelA.ampl       0.00014407  −1.12255e−05    +1.12255e−05
```

This quick method to calculate one−sigma parameter ranges, using `COVAR`, may underestimate the errors for a complex parameter space. The much slower but more appropriate `PROJECTION` algorithm should be used in such cases.

Simulating Multiple Datasets

One may simulate multiple datasets within the same *Sherpa* session. Each dataset is assigned a number.

For example, we next simulate a second dataset:

```
sherpa>   # Defining An Instrument Model
sherpa> RSP[instrumentB](dataB_rmf.fits, dataB_arf.fits)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT 2 = instrumentB
sherpa> SHOW INSTRUMENT 2
Instrument 2: rsp1d[instrumentB]
    Param   Type  Value
    -----   ----  -----
 1    rmf string: "dataB_rmf.fits" (N_E=1499,N_PHA=1024)
 2    arf string: "dataB_arf.fits" (N_E=1499)
```

Note that in order to simulate dataset number 2, there must be an instrument model number 2 defined. Similarly, a background dataset number 2 is entered, and the `FAKEIT` parameters for dataset number 2 are set:

```
sherpa> BACK 2 dataB_bkg.pha
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 BERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin
```

```
sherpa> INSTRUMENT back 2 = instrumentB

sherpa> SHOW FAKEIT 2
Fakeit exposure time: 1 seconds.
Fakeit backscale: 1

sherpa> FAKEIT 2 TIME = 27358.61
sherpa> FAKEIT 2 BACKSCALE = 0.0001
sherpa> SHOW FAKEIT 2
Fakeit exposure time: 27358.6 seconds.
Fakeit backscale: 0.0001
```

As with the instrument model, a source model number 2 is next defined, and finally the second dataset is simulated:

```
sherpa>  # Defining A Source Model Expression
sherpa> SOURCE 2 = xsphabs[modelB1]*POWLAW1D[modelB2]
sherpa> modelB1.nH = 0.025
sherpa> modelB2.gamma = 1.7
sherpa> modelB2.ampl = 1.0
sherpa> SHOW SOURCE 2
Source 2: (modelB1 * modelB2)
xsphabs[modelB1]  (XSPEC model name: phabs)  (integrate: off)
    Param   Type      Value        Min        Max                 Units
    −−−−−   −−−−      −−−−−        −−−        −−−                 −−−−−
 1     nH thawed    2.5e−02      1e−07         10    10**22 atoms/cm**2
powlaw[modelB2]  (integrate: on)
    Param   Type      Value        Min        Max                 Units
    −−−−−   −−−−      −−−−−        −−−        −−−                 −−−−−
 1  gamma thawed        1.7        −10         10
 2    ref frozen          1    −1e+120     1e+120
 3   ampl thawed          1          0     1e+120

sherpa> FAKEIT 2
FAKEIT: The current background data have been added to the faked spectrum.
```

This second simulated dataset may be normalized in the same manner as the first simulated dataset (dividing the known count rate of say 0.4, with the non−normalized count rate, to obtain a scaling factor; 0.4 / 1130.409 = 0.000353854):

```
sherpa> SHOW
.
.
Data 2: fake pha.
Total Size: 1024 bins (or pixels)
Dimensions: 1
Total counts (or values):    3.092874e+07
Exposure: 27358.61 sec
Count rate: 1130.494 cts/sec
  Backscal: 1e−04
.
.

sherpa> 0.4 / 1130.494
0.000353828
sherpa> modelB2.ampl = 0.000353828
sherpa> FAKEIT 2
FAKEIT: The current background data have been added to the faked spectrum.
```

And finally, the second simulated dataset is plotted:

```
sherpa> LPLOT DATA 2
```

The second simulated dataset may also be written to output data files:

```
sherpa> WRITE DATA 2 dataB_sim1.pha PHA
Write X−Axis: Bin  Y−Axis: Counts
sherpa> WRITE DATA 2 dataB_sim2.dat ASCII
Write X−Axis: Energy (keV)  Y−Axis: Flux (Counts/sec/keV)
```

To plot both simulated datasets at once:

```
sherpa> IGNORE 2 ENERGY :0.5,8:
sherpa>   # Plotting Both Simulated Datasets
sherpa> LPLOT 2 DATA 1 DATA 2
```

Figure 4 shows the resulting plot.

To exit *Sherpa*:

```
sherpa> EXIT
Goodbye.
unix%
```

Using a Sherpa Script to Run Simulations

Multiple simulations with count−rate normalization

The following script does several FAKEIT simulations using the count rate for normalization; this is different than the previous examples which all use the flux instead. Note that this type of a script can be included in a more complex script with fake_time, energy and cnt_rate as parameters.

The response files used in the script − `aciss_aimpt_cy06.rmf` and `aciss_aimpt_cy06.arf` − can be downloaded from the ACIS Cycle 06 RMFs/ARFs CALDB page.

```
sherpa> $cat multi_sim.shp
# Create a fake spectrum with a count rate of <cnt_rate> in the energy
# range <energy>, with a total exposure time of <fake_time>.

  fake_time = 100000
  energy    = [0.5, 2]
  cnt_rate  = 0.02

# Define instrument response and source model
  paramprompt off
  INSTRUMENT 1 = RSP[acis]("aciss_aimpt_cy06.rmf", "aciss_aimpt_cy06.arf")
  source 1 = pow[pow1]
  pow1.gamma = 1.9

# Define structure to allow manipulation of the pow1.ampl parameter
# value in S−lang, and give an initial guess for fakeit
  pow_ampl = get_par("pow1.ampl")
  pow_ampl.value = 1e−5

# Run fakeit once to define an initial dataset.  This is needed for
# the set_exptime() S−lang command to function
  fakeit
  ()=set_exptime(fake_time)

# Run fakeit once to get model counts for initial amplitude, then
# normalize to get desired count rate
  ()=set_par(pow_ampl)
  fakeit
  pow_ampl.value *= cnt_rate / (get_mcounts_sum(,energy).value / fake_time)
  ()=set_par(pow_ampl)
  fakeit

# Verify correct counts in fake dataset and plot data
  print("Count rate in specified energy range is")
  print(get_dcounts_sum(,energy).value / fake_time)
  lp data
```

Running this script:

```
sherpa> use multi_sim.shp
Model parameter prompting is off
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
Count rate in specified energy range is
0.01961
```

creates Figure 5 .

Simulating hot plasma emission

In this script, emission from a hot plasma is simulated with two different models. Note that the normalization is obtain within the script. The results of both simulations are then compared:

```
sherpa> $more thread.overallexample
RSP[acis](dataB_rmf.fits, dataB_arf.fits)
INSTRUMENT = acis
FAKEIT TIME = 5000
PARAMPROMPT OFF
SOURCE = xswabs[abs1]*xsmekal[m1]
abs1.1 = 0.03
m1.1 = 4.0
m1.norm = 0.025
FAKEIT
# Renormalize to the 2−10 keV flux of 1.e−13 erg/s/cm2
modflux1=get_eflux(1,[2,10])
obsflux1=1.e−13
foo1=get_par("m1.ampl")
foo1.value=foo1.value*(obsflux1/modflux1.value)
()=set_par(foo1)
FAKEIT
WRITE DATA sim_meka.pha PHA

INSTRUMENT 2 = acis
FAKEIT 2 TIME = 5000
SOURCE 2 = abs1 * xsapec[ap1]
ap1.kT = 4.0
ap1.norm = 0.025
FAKEIT 2
# SHOW
# Renormalize to the 2−10 keV flux of 1.e−13 erg/s/cm2
modflux2=get_eflux(2,[2,10])
obsflux2=1.e−13
```

```
foo2=get_par("ap1.ampl")
foo2.value=foo2.value*(obsflux2/modflux2.value)
()=set_par(foo2)
WRITE DATA 2 sim_apec.pha PHA
EXIT
```

This script can be run from the *Unix* command line as follows:

```
unix% sherpa thread.overallexample >&! thread.overallexample.out
```

The `thread.overallexample.out` file will then contain the screen output generated by the commands in the script.

History

14 Jan 2005	updated for CIAO 3.2: minor changes to screen output
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/fakeit/ Last modified: 1 Dec 2006

Image 1: The simulated dataset

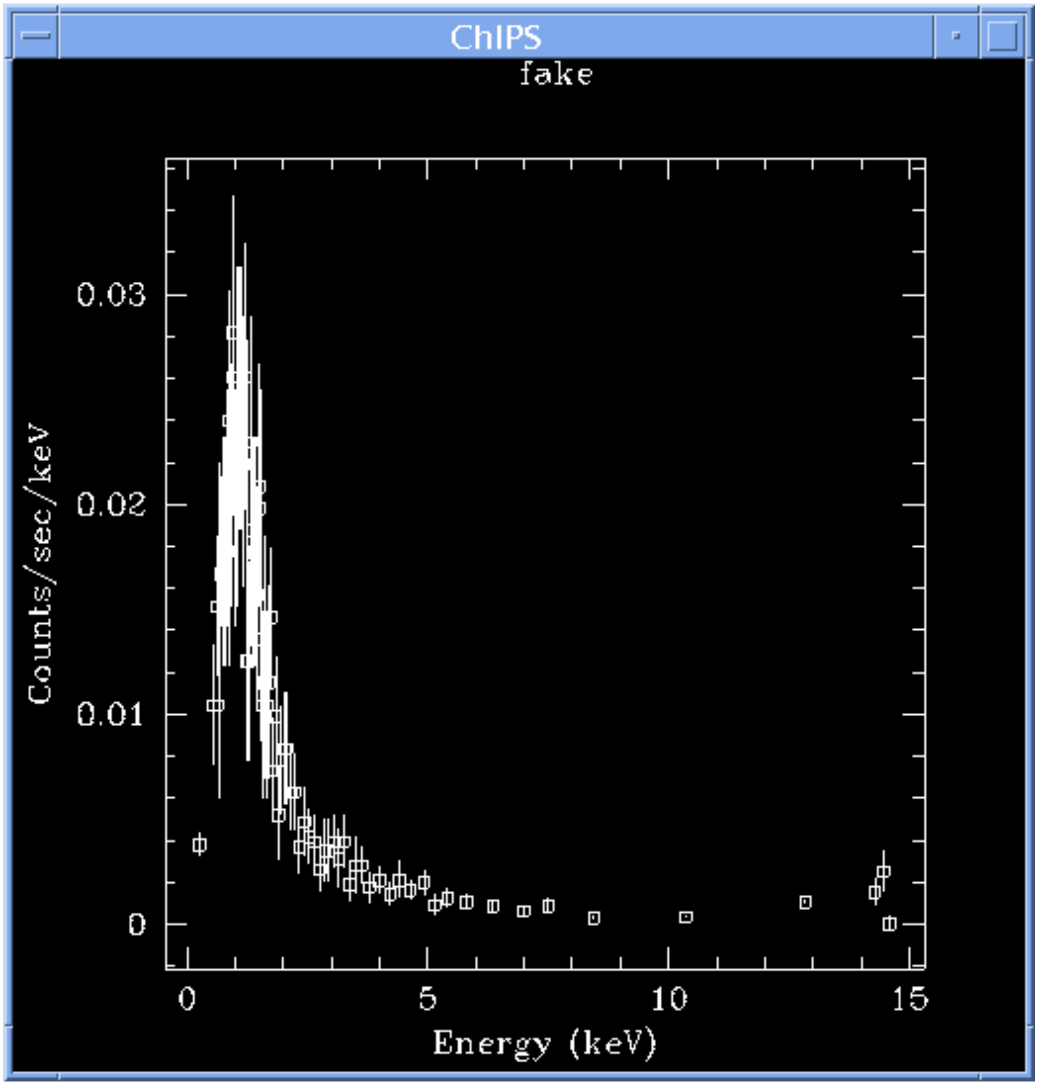


Image 2: Energy range to be fit

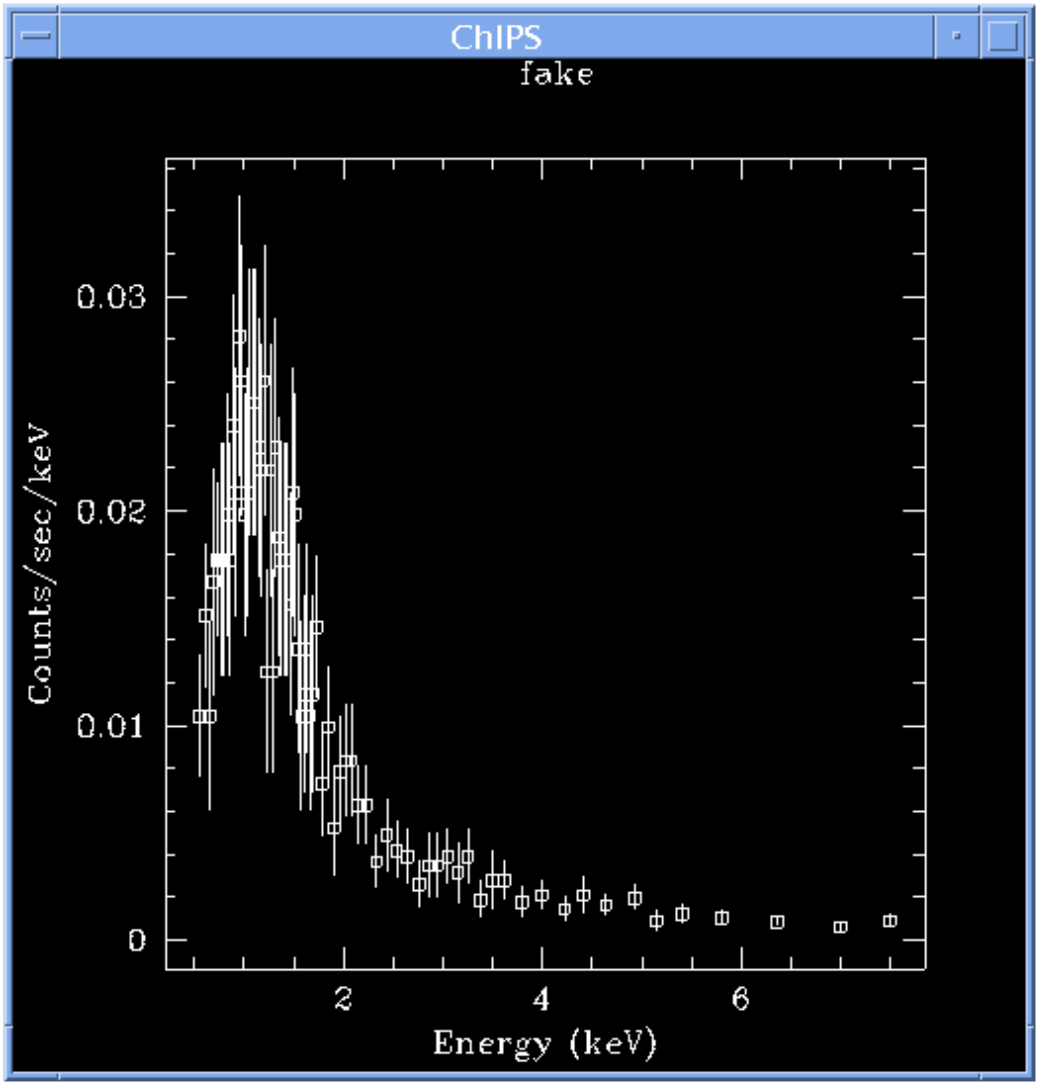


Image 3: Fit to simulated dataset

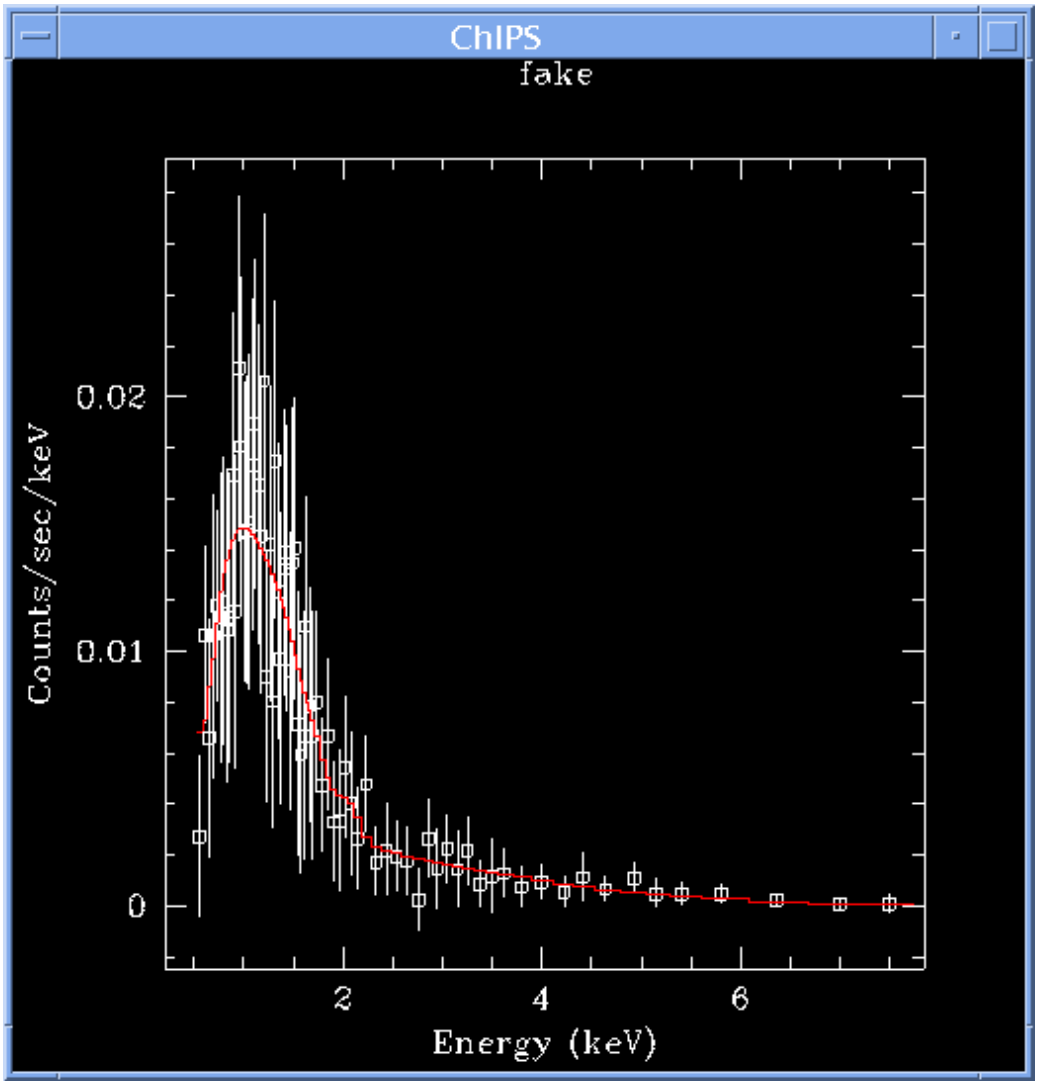


Image 4: Using more than one simulation at a time

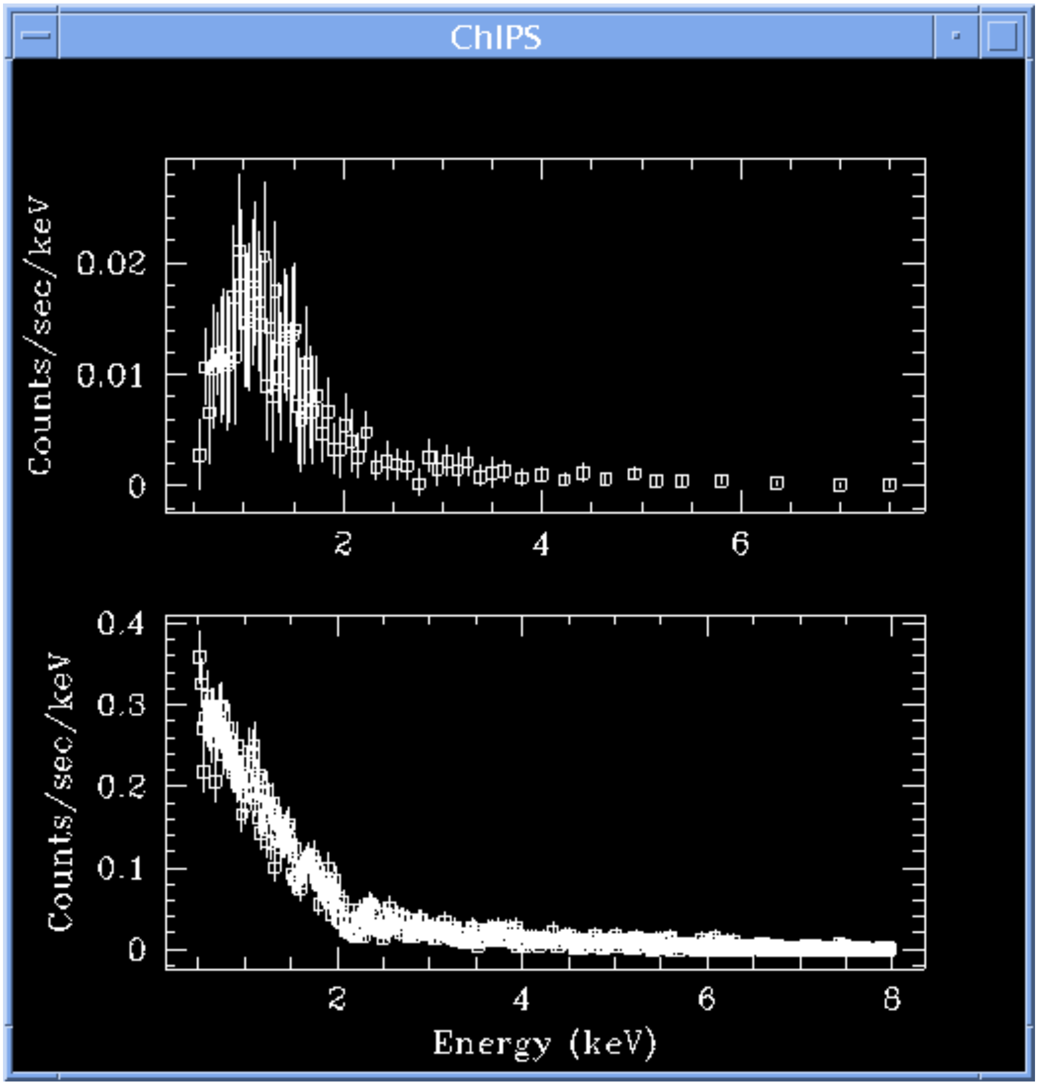
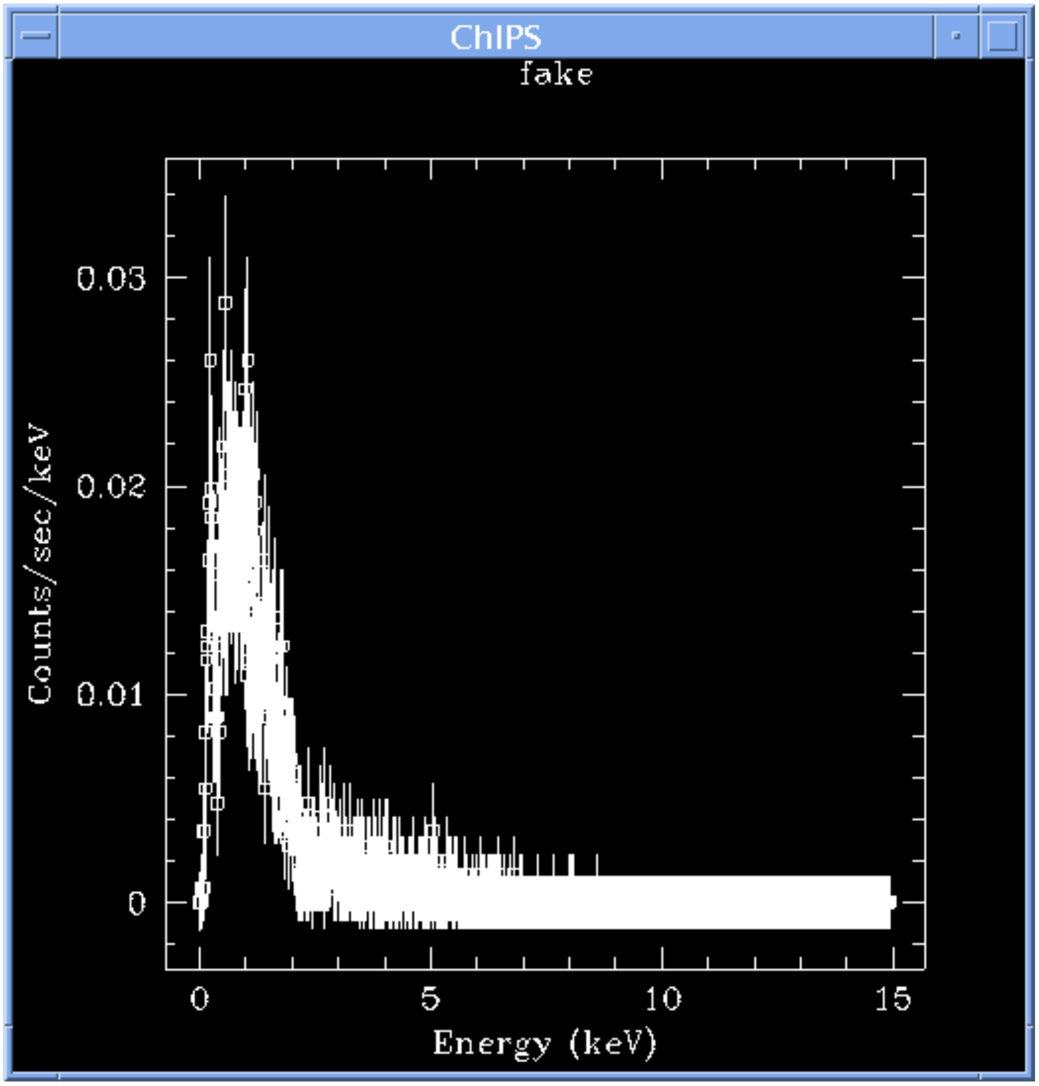


Image 5: Plot created by multi_sim.shp


ChIPS
fake
Counts/sec/keV
Energy (keV)