URL: http://cxc.harvard.edu/ciao3.4/instrument.html
Last modified: December 2006

AHELP for CIAO 3.4

instrument

Context: sherpa

Jump to: Description Examples Bugs See Also

Synopsis

Defines an expression to be used for modeling the instrument in source or background data analysis. The command RESPONSE is equivalent.

Syntax

```
sherpa> {INSTRUMENT | RESPONSE} [{SOURCE | BACK}] [# [ID]] = <modelExpr>

where # may specify the number of the dataset (default dataset number
is 1). The modifiers SOURCE and BACK may be used to specify the
application of the instrument model stack to either the source or
background data only. If neither is specified, then the model is used
for both the source and background data. The ID modifier is used only
for the command INSTRUMENT BACK (see below), and even then if and only
if the Sherpa state object variable multiback is set to 1, i.e., if
more than one background dataset is to be associated with a single
source dataset. The ID modifier may be any unreserved string (e.g., A,
foo, etc.), i.e., a string that is not a parsable command.
```

Description

The model expression, <modelExpr>, is an algebraic combination of one or more of the following elements:

```
{<sherpa_modelname> | <sherpa_modelname>[modelname] |
 <modelname> }
```

along with numerical values. The following operators are recognized: + * ( ); however, the operators + and * do not have the same meaning that they do when combining source model components. See below for details. (See the CREATE command for more information on establishing model components.)

Note that:

- In CIAO 3.1 the definition of INSTRUMENT BACK is required for both filtering and fitting the data if either background file or background models have been defined. INSTRUMENT BACK is set automatically when the PHA source file is input to Sherpa, however it is deleted if the NEW background file is input for a given data set, thus the new INSTRUMENT BACK has to defined on the command line before filtering and fitting the data with the new background file.

- By default, if the instrument model expression includes a component that has not previously been established, Sherpa will prompt for the initial parameter values for that component. This prompting can be turned off using the PARAMPROMPT OFF command.
- For the specific case of PHA data, a RSP instrument model is automatically defined if the RESPFILE and/or ANCRFILE header keywords are present and point to existing RMF and ARF files. The instrument model stack is then set to include the RSP model.

To reset an instrument model stack, issue the command:

```
sherpa> {INSTRUMENT | RESPONSE} [<dataset range> | ALLSETS] =
```

Instrument models describe instrument characteristics, such as effective area, a detector's energy response, or a mirror's point−spread function. They are convolved with, e.g., a source model to compute the number of detected counts in each detector bin.

Instrument model stacks are thus fundamentally different from other model stacks in that the models they contain are not evaluated themselves, but are used to tranform (i.e., fold) amplitude arrays (y' = f(y,x), instead of y = f(x)). (In Sherpa, there are two types of transformations: multiplication by an array, and multiplication by a redistribution matrix.) Thus, as noted above, the instrument stack operators + and * do not have the same meaning as their model stack counterparts.

Instrument models bound by the * operator collectively take a photon spectrum y and fold it to a counts spectrum y'. The order of the models does not matter so long as only there is only one redistibutive model (e.g., RMF or PSF) in the set.

```
sherpa> farf1d[a](arf.fits)
sherpa> frmf1d[r](rmf.fits)
sherpa> instrument = a*r
```

Here, the photon spectrum y is multiplied by the ARF, then folded through the RMF. This instrument stack is equivalent to

```
sherpa> instrument = rsp[a](rmf.fits,arf.fits)
```

Sets of instrument models separated by the + operator each fold the same evaluated photon spectrum y, with the resulting group of counts spectra being summed.

```
sherpa> farf1d[a1](arf_order1.fits)
sherpa> farf1d[a2](arf_order2.fits)
sherpa> frmf1d[r1](rmf_order1.fits)
sherpa> frmf1d[r2](rmf_order2.fits)
sherpa> instrument = a1*r1 + a2*r2
```

Here, the photon spectrum y is folded through the combination a1*r1 to produce counts spectrum c1; y is also folded through the combination a2*r2 to produce counts spectrum c2. The overall counts spectrum is then c1+c2.

The two rules governing instrument stacks are:

- Instrument models sets separated by + operators must all do similar evaluations, e.g., instrument = arf + rmf*arf is not allowed.
- As mentioned above, instrument model sets separated by + operators (or or a single instrument model set) cannot include two or more redistributive models, e.g., instrument = rmf*rmf is not allowed.

Note that if one wants to do more complex operations (e.g., dividing one arf by another as part of the folding process), one can use S−Lang to do the preliminary dirty work (for this example, the array division; the new array can then be loaded into Sherpa as the "arf" via Sherpa/S−Lang module function load_arf).

For Sherpa version 3.0.2, support for "dummy" instruments and datasets has been added:

- If data have been input and the instrument stack contains only an ARF, a dummy RMF will be created that maps the ARF bins to the data bins, if possible.
- If data have not been input and the instrument stack contains only an ARF, both a dummy RMF and a dummy PHA dataset will be created. (This is useful when one simply wants, e.g., to visualize models along an energy or wavelength grid, without having to define a DATASPACE.)
- If data have been input that contain information about the energy or wavelength grid (such as is contained in, e.g., the BIN_LO and BIN_HI columns of a Chandra grating data file), then both a dummy ARF and RMF are created.

One may always overwrite the dummy instruments if they are not appropriate.

Also for Sherpa version 3.0.2, checks have been added that may lead to the instrument stack being deleted if a subsequent DATA or DATASPACE command is issued, if it appears that the models in the stack are incompatible with the input data. In future versions of Sherpa, the instrument stack may be deleted automatically in such situations to avoid analysis problems (i.e., one will always have to specify instrument stacks after inputting data).

Also note that there are several instrument−model−stack−related Sherpa/S−Lang module functions.

Example 1

Define an instrument model using specified input response files:

```
sherpa> INSTRUMENT 1 = RSP[instrumentA]
instrumentA.rmf parameter value [] example.rmf
instrumentA.arf parameter value [] example.arf
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
```

Example 2

Define an instrument model, inputting the response files individually:

```
sherpa> ERASE ALL
sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> INSTRUMENT 1 = FARF[iarf]*FRMF[irmf]
sherpa> iarf.arf = example.arf
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> irmf.rmf = example.rmf
```

Example 3

Define an instrument model using specified input response files:

```
sherpa> ERASE ALL
sherpa> RSP[instrumentA](example.rmf, example.arf)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT 1 = instrumentA
```

Example 4

Define an instrument model using specified input response files, including an encircled−energy arf (EEARF):

```
sherpa> ERASE ALL
sherpa> FARF1D[ieearf](example.eearf)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> RSP[instrumentA](example.rmf, example.arf)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT 1 = ieearf*instrumentA
```

Example 5

Define an instrument model using specified input response files, with a path:

```
sherpa> ERASE ALL
sherpa> INSTRUMENT 1 = RSP[instrumentA]("data/example.rmf",
        "data/example.arf")
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
```

In all of the above examples, since neither a SOURCE or BACK argument is specified, the same instrument model is established for both the source and background.

Example 6

Define different source and background instrument models using specified input response files (RMF and ARF only):

```
sherpa> ERASE ALL
sherpa> PARAMPROMPT ON
Model parameter prompting is on
sherpa> INSTRUMENT SOURCE 1 = RSP[instrumentAsrc]
instrumentAsrc.rmf parameter value [] example.rmf
instrumentAsrc.arf parameter value [] example.arf
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT BACK 1 = RSP[instrumentAbkg]
instrumentAbkg.rmf parameter value [] example_bkg.rmf
instrumentAbkg.arf parameter value [] example_bkg.arf
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
```

Example 7

Define different source and background instrument models using specified input response files (RMF and ARF only):

```
sherpa> ERASE ALL
sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> INSTRUMENT SOURCE 1 = RSP[instrumentAsrc]
sherpa> instrumentAsrc.rmf = example.rmf
sherpa> instrumentAsrc.arf = example.arf
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT BACK 1 = RSP[instrumentAbkg]
sherpa> instrumentAbkg.rmf = example_bkg.rmf
sherpa> instrumentAbkg.arf = example_bkg.arf
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
```

Example 8

Define different source and background instrument models using specified input response files (RMF and ARF only):

```
sherpa> ERASE ALL
sherpa> RSP[instrumentAsrc](example.rmf, example.arf)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT SOURCE 1 = instrumentAsrc
sherpa> RSP[instrumentAbkg](example_bkg.rmf, example_bkg.arf)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT BACK 1 = instrumentAbkg
```

Example 9

Define different source and background instrument models using specified input response files (RMF and ARF only):

```
sherpa> ERASE ALL
sherpa> INSTRUMENT SOURCE 1 = RSP[instrumentAsrc]("data/example.rmf",
        "data/example.arf")
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT BACK 1 = RSP[instrumentAbkg]("data/example_bkg.rmf",
        "data/example_bkg.arf")
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
```

Example 10

Automatically define instrument models using input response files specified in dataset header(s):

```
sherpa> ERASE ALL
sherpa> DATA data/example.pi
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:
  <directory_path>/example.rmf
ARF is being input from:
  <directory_path>/example.arf
Background data are being input from:
  <directory_path>/example_bkg.pi
WARNING: statistical errors specified in the PHA file.
         These are currently IGNORED.  To use them, type:
         READ ERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin
sherpa> SHOW
...
------------------------------
Defined analysis model stacks:
------------------------------

instrument source 1 = AutoReadResponse
instrument back 1 = AutoReadResponse

-----------------------------------
Defined instrument model components:
-----------------------------------

rsp1d[AutoReadResponse]
    Param   Type      Value        Min        Max                Units
    -----   ----      -----        ---        ---                -----
 1    rmf string: "/data/simteste/Testing/sherpaTest/data/example.rmf"
 2    arf string: "/data/simteste/Testing/sherpaTest/data/example.arf"
```

In this example, the same instrument model was automatically defined for both the source and background data, since the input data file header referenced the response files named example.rmf and example.arf.

Example 11

Define an instrument model using the point−spread function contained in the file psf.fits:

```
sherpa> ERASE ALL
sherpa> PARAMPROMPT ON
Model parameter prompting is on
sherpa> FPSF2D[ps1]
ps1.file parameter value ["none"] psf.fits
ps1.xsize parameter value [32]
ps1.ysize parameter value [32]
ps1.xoff parameter value [0]
ps1.yoff parameter value [0]
ps1.fft parameter value [1]
sherpa> INSTRUMENT = ps1
```

The source model will be convolved with the PSF provided in the psf.fits file.

Bugs

See the Sherpa bug pages online for an up−to−date listing of known bugs.

See Also

sherpa

autoest, background, create, create_model, createparamset, fit, freeze, get_defined_models, get_model_params, get_models, get_num_par, get_par, get_stackexpr, getx, gety, guess, integrate, is_paramset, jointmode, kernel, lineid, linkparam, mdl, modelexpr, modelstack, nestedmodel, noise, paramprompt, paramset, pileup, rename, run_fit, set_par, set_paramset, set_stackexpr, source, thaw, truncate, unlink

---

URL: http://cxc.harvard.edu/ciao3.4/instrument.html
Last modified: December 2006