

model: define a theoretical model

Define the form of the theoretical model to be fit to the data.

```
model [<source num>:<name>] [<delimiter>] <component1> <delimiter>  
<component2> <delimiter>... <componentN> [ <delimiter>]
```

```
model [?]
```

```
model [<name>|unnamed] none
```

```
model clear
```

```
model <name>|unnamed active|inactive
```

```
rmodel [<source num>:]<spec num> <response function>|none
```

where <delimiter> is some combination of (, +, *,), and <componentJ> is one of the model components known to XSPEC. The optional name must be preceded by a source number followed by a colon. To specifically refer to the default model use the string unnamed. Descriptions of these models may be accessed by typing `help models` at the prompt.

The source argument and name, if present, assign that model to be used with one of the sources found to be in the spectrum during the data pipelining. These 2 parameters allow one to simultaneously analyze multiple models, each assigned to their own responses. The model will be referred to the channel space using a response corresponding to that source number. To create a model for a source number higher than 1, a detector response must first exist for that number. See the examples below and the **response** command for more information about using multiple sources. This ability to assign multiple models both generalizes and replaces the XSPEC11 method of using `'/b'` to specify background models.

After the model is loaded, if there are data present the model is attached through the instrumental response to the spectra to be fitted, as in XSPEC11. Unlike XSPEC11, however, if there are no data loaded the model will be attached to a default diagonal dummy response. The parameters of that dummy response (energy range, number of flux points, linear/logarithmic intervals) can be set by the user in the Xspec.init file using the DUMMY setting. Thus any model can be plotted in energy or wavelength space as soon as it has been defined.

The model components are of various types depending on what they represent and how they combine with other models additive, multiplicative, convolution, pile-up, and mixing models. Each component may have one or more parameters that can be varied during the fit (see the `newpar` command writeup).

Additive model components are those directly associated with sources, such as power laws, thermal models, emission lines, etc. The net effect of two independent additive models is just the sum of their individual emissivities.

Multiplicative model components do not directly produce photons, but instead modify (by an energy-dependent multiplicative parameter) the spectrum produced by one or more additive components. Examples of multiplicative models are photoelectric absorption models, edges, absorption lines, etc.

Convolution models components modify the spectrum as a whole, acting like operators rather than simply applying bin by bin multiplication factors. An example of a convolution model is a gaussian smoothing with energy dependent width. Thus, when using convolution models, the ordering of components is in general significant (see below under [syntax rules](#)).

The pile-up model is similar to the operation of the convolution models. The only difference is that the flux is multiplied by the effective area on input and divided by the same factors on output.

Mixing model components implement two-dimensional transformations of model spectra. The data are divided into regions by assigning them to 2 or more datagroups, and the transformation “mixes” the flux among the regions. An example is the project (projection) model, which assumes that the regions are 3-dimensional ellipsoidal shells in space, and projects the flux computed from the other components onto 2-dimensional elliptical annuli.

A list of all the currently installed models is given in response to the command

```
model ? (the ‘?’ is not actually required)
```

(this will leave the current model in use).

The new command variants have the following uses:

model [<name>] none

removes the model of name <name> if given. Without the <name> argument, the command removes the unnamed “default” model, which is of course the XSPEC11 behavior.

model clear

removes all models

model <name>|unnamed active|inactive

makes the model named <name> active (fit to data) or inactive. Inactive models are tied to a dummy (unit diagonal) response. Making a model assigned to a given source active makes any previous model assigned to that source inactive. Note that to make the default unnamed model active or inactive refer to it by the string unnamed.

See the commands `delcomp`, `addcomp` and `editmod` for details on how to modify the current model without having to enter a completely new model.

rmodel [<source num>:]<spec num> <response function>|none

assigns or removes a response function to the response belonging to <source num> of spectrum <spec num>. Currently the only available <response function> in XSPEC is `gain`, which makes **rmodel** redundant with the **gain** command usage:

```
gain fit [<source num>:]<spec num>
```

Syntax Rules

Model components are combined in the obvious algebraic way, with + separating additive models, * separating multiplicative models, and parentheses to show which additive models the multiplicative models act on. The * need not be included next to parentheses, where it is redundant. Also, if only one additive model is being modified by one or more multiplicative models, the required brackets may be replaced by a *. In this case the additive model must be the last component in the grouping. Thus

$$M_1*(A_1+A_2) + M_2*M_3(A_3) + M_4*A_4 + A_5$$

is a valid model, where the M 's signify multiplicative models and the A 's additive models.

The old style syntax for entering models (versions 9.02 and earlier) is not supported in version 12 and will return a syntax error.

XSPEC12's recursive lexical analyzer and expression parser allows, in principle, infinite nesting depth. It has been tested to 3 levels of parentheses, although it should be said that this new behavior is a by-product of the design rather than fulfilling an important need. Thus, expressions such as

$$M_1*(A_1 + A_2*(A_3 + M_2*M_3*(A_6 + A_7))) + C_1*(A_8 + A_9*(A_{10} + M_2*A_6))$$

are supported.

The model expression is analyzed on entry and syntax errors, or undefined models, will return control to the prompt with an error message. XSPEC12's model definition algorithm treats expressions delimited by '+' signs that are not within parentheses as separate "Component Groups". The Component Group comprises a list of components of the different types, and these are in turn calculated and then combined to produce an internal "Sum Component". These Sum Components from each such component group are then added to produce the output model (note that if there is an overall component – for example, a convolution or mixing component – then all of the model will be contained inside one Component Group).

The syntax rules that are checked for are as follows:

Expression must not begin with a "*"

A "*" must be preceded and followed by words or a brace (redundant braces are removed).

A standalone component must be additive. A standalone component is defined as a single component model or a single component at the beginning (end) of the expression followed (preceded) by a "+", or in the middle of the expression delimited by 2 "+" signs.

A convolution component must not appear at the end, or followed by a closing brace.

A mixing model component must appear first in the expression and apply to all components (thus a model including a mixing component always has one Component Group).

When using convolution components, the order in which they are applied is in general significant. For example, the two models

$$C_1*M_1(A_1+A_2) \quad \text{and} \quad M_1*C_1(A_1+A_2)$$

are not necessarily equivalent (here the C's represent convolution models). The way XSPEC handles the ordering of components is by first computing the spectrum for the additive components of a given additive group (A_1+A_2 in the above example). It then applies all multiplicative or convolution components in the additive group from right to left in the order they appear in the model formula.

N.B. Beginning with v12.5.0, convolutions no longer have to precede the source. Parentheses may also be used to specify convolution precedence, so the following two examples are not equivalent:

$$C_I * M_I(A_1+A_2) \quad \text{and} \quad (C_I * M_I)(A_1+A_2)$$

Mixing models are a special case. The mixing transformation is applied to the entire model once the combination into a single Sum Component has been executed. Note that since XSPEC12 can have multiple models applied to a given spectrum, the mixing transformation can nevertheless be applied to only one of the models being fit. This will be relevant, for example, for the case where the background is fitted by a separate model

Examples

Note that po (= powerlaw) and ga (= gauss) are additive models, and that wabs and phabs (different photoelectric absorption screens) are multiplicative models.

```
XSPEC12> model po
// The single component po (powerlaw) is the model.
XSPEC12> model po+ga
XSPEC12> model (po+ga)wabs
XSPEC12> model phabs(po+ga)
XSPEC12> model wa(phabs(po)+ga)
XSPEC12> model wa po phabs ga //error: old syntax
XSPEC12> model wa*phabs*po
XSPEC12> model (po+po)phabs
//Note that though the first and second components are the same
// form, their parameters are varied separately.
XSPEC12> model phabs*wa(po)
```

A complex (and almost certainly unphysical) example is the following:

```
XSPEC12>model wa(po+pha(peg+edge(disk+bbod)))const + pla(pos+hr*step) + not*gau
```

Applying multiple models:

Assume 3 spectra are loaded, each with a single response (source 1 by default).

```
XSPEC12> model wa(po)
// The unnamed model wa(po) will apply to all 3 spectra, accordingly
// multiplied by each spectrum's response.
XSPEC12> response 2:2 new_resp.pha 2:3 another_new_resp.pha
// Additional responses assigned to source number 2 for spectra 2 and 3.
XSPEC12> model 2:second_mod ga
// The model "second_mod" will now apply to source 2, and is therefore
// multiplied by new_resp.pha and another_new_resp.pha for spectra 2
// and 3 respectively.
XSPEC12> model second_mod inactive
// "second_mod" will no longer apply to spectra 2 and 3, though they
// retain responses for source 2.
```

```
OR
XSPEC12> response 2:2 none
XSPEC12> response 2:3 none
// No responses exist for source number 2, second_mod is
// rendered inactive.
```