

fakeit: simulate observations of theoretical models

Produce spectra with simulated data.

Syntax: **fakeit** [<file spec>...]

where <file spec> ::= [<file number>] <file name>[{ranges}]... is similar to the syntax used in the **backgrnd**, **corfile**, and **response** command. The **fakeit** command is used to create a number of spectrum files, where the current model is multiplied by the response curves and then added to a realization of any background. Statistical fluctuations can be included. The integration time and correction norm are requested for each file. The file names input as command line arguments are used as background. The number of faked spectra produced is the maximum of the number of spectra currently loaded and the number of file specifications in the command line arguments. The special case **fakeit none** makes one fake spectrum for each spectrum loaded (or one fake spectrum if there are none loaded). See the examples below for a clearer description.

If a faked spectrum is based on a currently loaded spectrum, then by default the background, response, correction file, and numerical information are taken from the currently-defined data, unless a background file is specified on the command line in which case it becomes the background. The **fakeit none** case prompts for the rmf and arf filenames and sets the default numerical data to 1.0, except the correction norm, which is set to zero. If the output file is type II then the exposure time and correction scale factor will be the same for all spectra in the file.

For each output file, the user will be prompted for an output file name. If a background file is in use then fakeit will also simulate a new background for each spectrum. Background files are given the same names as output spectrum files but with `_bkg` appended to the end of the stem.

The simulated spectra automatically become the current data files. The ignore status is completely reset.

Statistical Issues:

The statistical fluctuations used to create the simulated spectra will depend on whether the current spectra have Poisson or Gaussian errors. If a spectrum file has a STAT_ERR column and the POISSERR keyword is set to false then xspec assumes Gaussian errors with sigma from the values in the column. Otherwise, errors are assumed to be Poisson based on the number of counts. Note that it is possible for the spectrum and background files to have different error types. For fakeit cases when there is no current file to use, Poisson errors are assumed.

Type I vs. Type II Output:

Fakeit determines whether to place its fake spectra and background data into type I or type II files based on the following rules.

If fake spectra are based on currently loaded spectra then the output files will have the same format as those loaded. For example: Assume 3 spectra are currently loaded, spectrum 1 from file `typeIdata.pha` and spectra 2 and 3 from file `typeIIdata.pha`. Then,

```
XSPEC12> fakeit
```

will produce 3 fake spectra in 2 output files with names prompted from the user. The first file will be type I, the second type II containing 2 spectra. The same is true for any background files produced.

If the user asks for more fake spectra to be created than the number of spectra currently loaded, for example by typing the following when the same 3 spectra above described are loaded:

```
XSPEC12> fakeit 5
```

then fake spectra 1-3 will be placed in the two files as before. For the additional fake spectra (4 and 5), fakeit uses the following rule: If any of the originally loaded spectra were in a type II file, then all of the additional fake spectra will be placed in 1 type II file. Otherwise, they will each be placed in a separate type I file. In this example, since a type II file was originally loaded (typeIIData.pha) when fakeit was called, spectra 4 and 5 will be placed together in a type II output file, in addition to the type I and type II files for the first 3 fake spectra.

If there are no currently loaded spectra all output files will be type I unless either of the following situations exist: 1. Any of the background files entered on the command line are type II, as indicated by row specifiers in brackets. 2. The first response file used clearly belongs to a format associated with type II data, such as SPI/Integral with its multiple RMF format (see section on SPI/Integral usage).

Overall, though the method of determining output format for additional spectra may seem quite complicated, it can be easily summed up: Fakeit will place all additional spectra and backgrounds (ie. those not based on already loaded data) in type I output files, unless it detects any evidence of type II file usage amongst the command line input, in which case it will produce type II output.

Note For SPI/Integral Format:

Since the SPI/Integral format builds its responses from a combination of multiple RMFs and ARFs, it must use a different scheme than the OGIP type I and II formats for storing RMF and ARF file location information. This information is stored in a FITS extension, named “RESPFILE_DB” , added to the PHA file. Therefore, when fakeit prompts the user for the location of the response file, simply enter the name of a FITS file which contains a RESPFILE_DB extension pointing to the RMFs and ARFs to be applied. When prompted for an ARF name, enter nothing.

The prompts will only appear for the first spectrum in the data set, and the ARFs will be assigned row by row 1 to 1 with the spectra. For example, if no data is currently loaded, to create 3 fake SPI spectra from the RMFs and ARFs named in the RESPFILE_DB extension of the file realSpiData.pha :

```
XSPEC12> fakeit 3
// ...(various prompts will follow)...
For fake spectrum #1 response file is needed:  realSpiData.pha
// ...and ancillary file:  <Ret>
// ...(more fakeit prompts)...
```

This will create 3 fake spectra, each making use of the same RMFs/ARFs, spectrum 1 using the first row of the ARFs, spectrum 2 using the second etc.

*** CAUTION – SPI/Integral ***

As currently implemented, the RESPFILE_DB method of storing ARF locations does not retain specific row information. The assumption is that the rows in the ARF correspond 1 to 1 with the rows in the spectral data extension. Therefore, much confusion can arise when the row numbers of the loaded spectra do not match that of the fake spectra. For example:

```
XSPEC12> data my_spi_data.pha{3-4}
```

```
// my_spi_data.pha contains a RESPFILE_DB table pointing to arf1.fits,
// arf2.fit, arf3.fits.
// ...(fit to some model(s))...
XSPEC12> fakeit
```

This will produce 2 fake spectra generated from the model*response operation, where the model has parameters based on a fit to the original spectra in rows 3 and 4 of my_spi_data.pha, which used ROWS 3 AND 4 of the 3 arf files for their own responses. However, the responses used above to generate the 2 fake spectra will use ROWS 1 AND 2 of the 3 arf files. This is necessary since the fake spectra will be placed in rows 1 and 2 of their fakeit output file.

Examples:

Type I files:

For each of these examples, assume 3 spectra are currently loaded, each in its own type I file, and that the second spectrum has a background file.

```
XSPEC12> fakeit
```

This will produce 3 fake spectra each in its own type I output file, and the user will be prompted for the file names. The response file information will come from each of the original spectra. If any response information is invalid, the user will then be prompted. A fake background file will be produced for the second spectrum.

```
XSPEC12> fakeit 4
```

Produces 4 fake spectra, the first 3 created as in the previous example. The fourth will be created with no background spectrum, and this user is prompted for response information.

```
XSPEC12> fakeit backa,,none 4
```

Produces 4 fake spectra. For the first spectrum, a fake background file will be generated from the file backa. The second uses its own background file as before. The third fake spectra will no longer use the response information from loaded spectrum 3, the user will be prompted instead, and its default numerical data will be reset to 1. The fourth spectrum will be created as in the previous example.

If no data is currently loaded:

```
XSPEC12> fakeit 2
```

Produces 2 fake spectra in separate type I files, unless the first user entered response file belongs to a format that is explicitly type II (ie. SPI/Integral).

Type II files:

Assume four spectra with no backgrounds have been loaded from one type II file:

```
XSPEC12> data original_type2_data.pha{5-8}
```

Then, after model(s) have been entered and a fit:

```
XSPEC12> fakeit
```

This will produce 4 fake spectra in rows 1 to 4 of one type II output file, with responses and arfs taken from the columns of original_type2_data.pha.

```
XSPEC12> fakeit ,,backb{1-3}
```

This produces 5 fake spectra in two type II output files, and 3 fake background spectra also placed in two type II output files:

The first 4 fake spectra are placed in one output file since that is how the 4 spectra they were based on were originally organized. The default numerical data for this file are taken from the original spectra. Fake spectra 3 and 4 now have backgrounds, based on `backb{1}` and `backb{2}` respectively. These will generate 2 fake background spectra, placed in rows 3 and 4 of the first output fake background file. Rows 1 and 2 of this file will just consist of zeros since the first 2 spectra have no backgrounds.

The fifth fake spectrum will be placed in the second type II PHA file. Response and numerical data will not be based on the existing loaded spectra. A fake background will be generated from `backb{3}` and placed in row 1 of the second type II fake background file.

Now assume no data is currently loaded:

```
XSPEC12> fakeit 2 backb{1}
```

2 fake spectra in one type II output file are produced, as is a corresponding fake background file with 2 rows. The fact that the user has entered a type II background file on the command line tells `fakeit` to produce type II output. The first fake spectrum will have no associated background, so row 1 in the fake background file will be all zeros. Row 2 will consist of the fake background generated from `backb{1}`.