OSPEX ANY SPECFILE Input

Back to Main OSPEX page

The OSPEX ANY SPECFILE method was implemented in November 2014 to allow you to analyze any type of input data file by supplying your own software readers. This should be especially useful for new data types that haven't been explicitly implemented in OSPEX (yet), but that you would like to analyze. The idea is that you supply two routines - readers for a spectrum and response file - and tell OSPEX the base name of those readers. When you set the spectrum input file in OSPEX to a file that it doesn't recognize, it will call your reader routines.

Setup

1. You create one or two routines: one to read the input spectrum file, and, if necessary, another to read the detector response file (input and output required is described below).
2. The names of those routines must be xxx_data and xxx_drm where xxx is chosen by you.
3. At the command line, set the ospex control parameter spex_file_reader to a string corresponding to xxx, the base name of those two routines.

   ```
   o->set, spex_file_reader='xxx'
   ```

4. Using the GUI or the command line, set the input spectrum file, and if necessary, the input response file. If using the command line:

   ```
   o->set,spex_specfile='yourspectrumfile.yyy'
   o->set, spex_drmfile='yourdrmfile.yyy'
   ```

5. Proceed as usual in OSPEX to analyze the data.

xxx_data Routine Requirements

The procedure definition of your xxx_data routine should look like this:

PRO xxx_data, FILES=files, data_str=data_str, ERR_CODE=err_code, ERR_MSG=err_msg, _REF_EXTRA=_ref_extra

The xxx_data routine must have the following keyword arguments:

FILES - (INPUT) Scalar or vector string of file names to read
DATA_STR - (OUTPUT) Structure containing info read from file (described below)
ERR_CODE - (OUTPUT) 0 means success reading file(s). 1 means failure
ERR_MSG - (OUTPUT) string containing error message if any. '' means no error.
_REF_EXTRA - (INPUT) - Only needed if you want to pass arguments through to another routine.

The DATA_STR structure returned by the xxx_data routine must contain these fields:

```
data_str = { $
START_TIME: start_time, $ start time of file in anytim format
END_TIME: end_time, $ end time of file in anytim format
RCOUNTS: rcounts, $ [nenergy,ntime] count data
ERCOUNTS: ercounts, $ [nenergy,ntime] error in count data
```

UT_EDGES: ut_edges, $ [2,ntime] edges of time bins in anytim format
UNITS: units, $ units of rcounts, usually 'counts'
AREA: area, $ effective detector area in cm^2
LTIME: ltime, $ [nenergy,ntime] livetimes
CT_EDGES: ct_edges, $ [2,nenergy] edges of energy bins in keV
DATA_NAME: data_name, $ string name of data, e.g. 'RHESSI'
TITLE: title, $ string title of data, e.g. 'RHESSI SPECTRUM'
RESPFILE: respfile, $ string name of corresponding SRM file, or blank string
  OR [nedges_out, nedges_in] numeric array of DRM values
  OR structure containing fields drm ([nedges_out, nedges_in]), edges_in, and edges_out
DETUSED: detused, $ string of detector names
ATTEN_STATES: atten_states, $ [ntime] array of filter states, or scalar -1 if doesn't apply
DECONVOLVED: deconvolved, $ If 1, original data is already photons
PSEUDO_LIVETIME: pseudo_livetime, $ if 1, livetime isn't a real livetime
XYOFFSET: xyoffset } [2] x,y location of source on Sun

xxx_drm Routine Requirements

The xxx_drm routine is needed only if the RESPFILE value returned by the xxx_data routine did not contain the DRM values (i.e. RESPFILE was not numeric or a structure). In that case, RESPFILE was either the name of the DRM file or a blank string. If you used the GUI to set the spectrum file, and RESPFILE was the name of the DRM file, it will automatically be set for you. If you're using the command line, whether RESPFILE was the name of the file or a blank string, you must explicitly set it via the spex_drmfile parameter.

The procedure definition of your xxx_drm routine should look like this:

PRO xxx_drm, DRM_STR, FILE=file, SFILE=sfile, ERR_CODE=err_code, ERR_MSG=err_msg

The xxx_drm routine must have the following keyword arguments:

FILE - (INPUT) string of file name to read
ERR_CODE - (OUTPUT) 0 means success reading file. 1 means failure.
ERR_MSG - (OUTPUT) string containing error message if any. Blank string means no error.

and the following positional argument:

DRM_STR - (OUTPUT) Structure containing info read from DRM file (must contain these fields):

drm_str = { $
EDGES_OUT: edges_out, $ [2,nedges_out] count energy edges in keV
PH_EDGES: ph_edges, $ [2,nedges_in] photon energy edges in keV
AREA: area, $ detector area in cm^2
DRM: drm, $ [nedges_out,nedges_in,nfilter] DRM matrix for each filter state
SEPDETS: sepdets, $ 1 if separate detectors
DATA_NAME: data_name, $ string name of data, e.g. 'RHESSI'
FILTER: atten_state, $ [nfilter] value of filter states, or scalar -1 if doesn't apply
DETUSED: detused } string of detectors used, e.g. '1F 3F 4F 5F 6F 8F 9F'

Example

There is an example of using this new 'ANY' input method in the SSW OSPEX directory. The routines stx_read_data.pro and stx_read_drm.pro were written to read simulated STX data. To use them, you would do the following:

```
o->set, spex_file_reader='stx_read'
o->set, spex_specfile='stx_spectrum_20190101_0000.fits'   ; name of your spectrum file
o->set, spex_drmfile='stx_srm_20190101_0000.fits'         ; name of your response file
```

or if you're using the OSPEX GUI:

```
o->set, spex_file_reader='stx_read'
```

then use the GUI to navigate to your spectrum input file.

Kim Tolbert
Last Modified: 28 September 2015