Extracting Light Curves and Spectra

Introduction

Extracting clean, useable light curves and spectra is one of the main objectives of RXTE data reduction. In broad terms, it involves choosing:

1. **Which subset of the data to extract**, that is, which time and channel ranges, which PCA anodes or HEXTE clusters, etc. These choices sometimes depend on the results of a previous extraction.

2. **How to bin the data**, that is, whether to extract a light curve, spectrum or both, how large to make the bins, whether to exclude partially filled bins, etc.

Practically, extraction involves running one of two ftools, depending on the kind of data: **saextrct** for science array data and **seextrct** for science event data. How to run these ftools, known as the extractors, is described in detail below: each question asked by the two ftools is explained and examples are given.

One supplementary aspect of extraction deserves its own description and is covered in the the section that follows, namely, the various ways in which time selections can be made.

Finally, before extracting data from a given configuration, it’s a good idea to review just what options the data format actually allows. PCA Standard 1 data, for example, come in only one spectral bin, so extracting a spectrum would only yield the mean count rate. The reduction options are described in the *PCA Issues* chapter, in the section on EDS configurations.

Time Selections

In practical terms, the final step of selecting data based on time entails entering a set or sets of start and stop times into the extractors. This simple matter is covered below in the two sections about saextrct and seextrct. Here, however, we describe the first step of creating these start and stop times.

Generally, creating start and stop times involves using some "other" information. How you do this falls into three categories:

1. **From times you already know.** For example, suppose you want to integrate a spectrum over the same period as an optical observation, the start and stop times of which you know.

2. **From the plot of a light curve.** For example, suppose you have extracted a light curve from Standard 1 PCA data and want to extract a HEXTE spectrum from an interestingly high-flux portion of the light curve.

3. **From the values of columns in a FITS file (i.e. making a GTI file).** For example, suppose you want to extract only those data corresponding to times when the value of the satellite elevation, as contained in the filter file, is greater than 10 degrees.

Creating Start and Stop Times with xTime

Creating Start and Stop Times from Light Curves

This subsection presumes, of course, that you've already extracted a light curve. It appears here, rather than after the sections on running the extractors, for the sake of completeness.

Light curves created by the extractors are in FITS and usually have columns called `TIME`, `RATE`, `ERROR` and `FRACEXP`. This can be verified using the ftool **flcol** which prints to the screen the column names in a FITS file.

To plot a light curve, simply run the ftool **fplot**. The program prompts the user for the column containing the x-axis and then for the columns to be used on the y-axes. It calls the same PGPLOT subroutines as Xspec and Xronos. Here's an example:

```
rufus [121] [day] x1728: fplot data4_soft.lc
Name of X Axis Parameter[error][] TIME
Name of Y Axis Parameter[error] up to 8 allowed[] RATE[ERROR]
Lists of rows[-]
Device: /XWindow, /XTerm, /TK, /PS, etc[/XW]
Any legal PLT command[]
```

At the `PLT>` prompt, you can enter various commands to change the appearance of the plot. After examining the plot, make a note of the start and stop times you're interested in and quit the program.

The times displayed on the plot are relative to the value of the `TSTART` keyword in the header of the light curve. Since the extractors work in SCCS (spacecraft clock seconds), the start and stop times from the light curve have to be converted. The Perl script **timetrans** will do this. Simply type "timetrans" and then enter the ranges from the plot. The resulting output file containing the converted times can be entered into the extractors (with an @ sign) at the prompt Input time intervals t1-t2, t3-t4 in seconds.

Creating GTI

Creating GTI files based on filter files is covered in the *Screening* chapter. Here's a briefer, alternative example: a GTI file generated from a light curve containing the start and stop times corresponding to when the count rate in the light curve is between 10,000 and 20,000 counts per second. As in the case of filter files, the ftool **maketime** is used to generate the GTI file:

```
rufus [123] [day] x1728: maketime
Name of FITS file and [ext#][] data7.lc
Name of output FITS file[] medium.gti
Selection Expression[] rate.gt.10000.and.rate.lt.20000
Column containing HK parameter names[NAME]
Column containing HK parameter values[VALUE]
Column containing HK parameter times[TIME] Time
Flag, yes if HK format is compact[yes] no
```

The times in the GTI file, which you can examine with the ftool fdump, are relative to the value of `TSTART` in the header. The GTI file itself is applied by the extractors in response to the question Input GTI file to be AND'd with INFILE.

Extracting Science Array Data

The best way to explain how to use saextrct is by example. In what follows, a typical saextrct run is shown. Inputs are explained under the links which take you to a series of notes at the end of the example.

After the example and explanatory notes comes a list of various "reduction scenarios" motivated by scientific priorities and accompanied by the corresponding saextrct inputs - scenarios such as "creating a light curve with 0.125-second bins from Binned Burst Catcher data in the 2-6 keV band."

Example: Creating a Light Curve from PCA Standard-2 Data

Here, we'll use saextrct to create a light curve with 32-second bins from PCA Standard-2 data. We have already run: a) XDF to create a list (sdt2.list) of Standard-2 data files, and b) maketime to create a GTI file excluding elevations greater than 10 degrees (elv_gt_10.gti). No other selections are applied. In addition:

- User inputs are shown in boldface.
- Default inputs, where appropriate, are included in square brackets and are entered with a carriage return (not shown explicitly).
- For full documentation about saextrct, type **fhelp saextrct**.

The corresponding screen dialogue is as follows:

```
rufus [58] [day] redux: saextrct

Running SAEXTRCT version 3.5.2
==============================================
Input file name or @file-of-filenames:[] @std2.list
Input GTI files to be OR'd with INFILE (APPLY):[APPLY]
Input GTI file to be AND'd with INFILE:[-] elv_gt_10.gti
Root name for output file:[sac] pca_32s
Accumulate (ONE) or (MANY) Spectral/Light Curves:[ONE]
Name of TIME column:[TIME]
Name of COLUMNS to be accumulated (GOOD):[GOOD]
Input the binsize in seconds, use 0.1 etc. if nec (INDEF):[INDEF] 32
Minimum acceptable fractional exposure (INDEF):[INDEF]
Chose print option, LIGHTCURVE, SPECTRUM, or BOTH:[BOTH] lightcurve
Type of binning for LIGHTCURVE: (SUM, RATE, MEAN):[RATE]
Type of binning for SPECTRUM (SUM, RATE, MEAN):[SUM]
Maximum acceptable intensity for Light Curve (INDEF):[INDEF]
Maximum acceptable intensity for Spectrum (INDEF):[INDEF]
Column titles for SUM written as header Keywords:[-]
Column titles for MEAN written as header Keywords:[-]
Starting time for summation (INDEF):[INDEF]
Ending time for summation (INDEF):[INDEF]
Input time intervals t1-t2,t3-t4 in seconds (INDEF):[INDEF]
Minimum energy bin to include in Spectra (INDEF):[INDEF]
Maximum energy bin to include in Spectra (INDEF):[INDEF]
Input channels to be retained 1-2,3-4 (INDEF):[INDEF]
Input channels for each bin 1-5,6-256 (INDEF):[INDEF]
Perform a dryrun on files without processing any data? (Yes):[no]
```

Explanation of the Various Inputs in the Example above

- **Input file name:** The name of a single input science array file *or*, when prefaced with "@", the name of an ASCII file containing a list of several science array files. In the above example, the file std2.list was generated by XDF and contains the lines:

  ```
  /data/kaah5/xtegof/day/herx1/P00023/00023-02-01-00/pca/FS4a_3eb7080-3eb7fd0
  /data/kaah5/xtegof/day/herx1/P00023/00023-02-01-00/pca/FS4a_3eb8860-3eb922d
  ```

  At present, up to 100 files can be listed in this way. Please note that the list should end with a carriage return.

- **Input GTI files to be OR'd:** Whether to apply the GTI information in the science array files themselves. For practical purposes, there are two alternative inputs to this prompt: "APPLY" (the default) instructs saextrct to apply the GTI in the GTI extensions of each of the input files. These GTI are "OR'ed" with GTI input at the next prompt. A blank or "-" instructs saextrct not to apply the GTI.

- **Input GTI file to be AND'd:** The name of a FITS GTI file containing your selected good times, e.g. as constructed from a filter file using maketime. In this example, elv_gt_10.gti has GTI corresponding to when the elevation of the satellite was 10 degrees above the horizon. The current modus operandi of this type of file is:

- More than one file may be entered by using "@" followed by the name of a list of up to 100 GTI files, one per line, ending with a carriage return. The times contained in these files will be AND'ed.

  We recommend, however, if you want to combine more than one FITS GTI file, that you should use the ftool **mgtime** to merge the individual FITS GTI files into a single file.

- The GTI will be OR'ed with those in the input science array files, if applied.
- If your time selections comprise simply a start time and an end time, you can enter them instead at the timemin and timemax prompts.

- **Root name for output file:** Here, since "pca_std2" is entered, the extracted lightcurve and spectrum will be called "pca_std2.lc" and "pca_std2.pha", respectively.

- **Accumulate (ONE) or (MANY):** You can decide whether you want the output lightcurve and spectrum to contain either one column of binned events (combined from data extracted from the input columns prompted for below) or many columns (one per input column). Our default of one column is the more usual, but the choice depends on what you want to do with downstream software. Both xronos and xspec can handle many-column files. But of course, many-column files occupy more disk space.

- **Name of TIME column:** Self-explanatory. But note that unlike some other ftools, saextrct is insensitive to case: even though the column is called `Time`, the default of `TIME` is acceptable.

- **Name of COLUMNS to be accumulated:** Most science array files have more than one column containing valid science data (HEXTE Archive files, for example, have five). The default "GOOD", which we entered here, will cause saextrct to extract data from all the valid columns. If, instead, you want to extract data from a subset of columns, you can either list the columns at the prompt (using up to 80 characters) or input an ASCII file listing the columns, one per line, preceded with "@". For example, the list of columns in PCA Standard-2 files corresponding to the top layer of PCU0-2 comprises:

  ```
  X1LSpecPcu0
  X1RSpecPcu0
  X1LSpecPcu1
  X1RSpecPcu1
  X1LSpecPcu2
  X1RSpecPcu2
  ```

  To see the column names in any FITS file, run the ftool **flcol**. Please note that the list should end with a carriage return.

- **Input the binsize in seconds:** Self-explanatory - here we've entered 32 seconds. But the binsize is not a trivial matter:

  - If you enter a binsize smaller than the intrinsic resolution of the data (of the first data file, to be precise), the binsize will be reset to that value. The intrinsic time resolution of data in, say, column *N* is given by the value of the `TIMDEL` keyword in the header of the `XTE_SA` extension - *however, this value is superseded by the* CDLTN *keyword if present.*

  - Binsizes and time stamps have a minimum time resolution of 0.00000001 seconds (0.01 microseconds). This is because 8 of the 16 available digits in double precision are taken up by the number of seconds in Mission Elapsed Time. As a result, binsizes cannot be smaller than 0.01 microseconds nor be specified with an accuracy requiring more than 8 digits after the decimal point.

- **Minimum acceptable fractional exposure:** Allows the option for light curves of accumulating partially filled bins. By default, no partially filled bins are included in the output light curve. If, say, the minimum acceptable fractional exposure is set to 0.75, then all bins more than three quarters full will be included. Please note that if you do include partially filled bins, you should make sure that your analysis software, like Xronos, can interpret the FRACEXP column in the output light curve. Note, too, that this option is only active when the type of binning is "RATE".

- **Chose print option:** Specifies whether to output a light curve, a spectrum or both (the default). Since light curve files can be large, you might want to specify "spectrum" on those occasions when you want only a spectrum.

- **Type of binning for LIGHTCURVE:** This option determines how the binned values in light curves are calculated: "RATE" (the default) divides the total accumulated counts by the binsize; "SUM" simply gives the total accumulated counts in each bin; "MEAN" gives the mean value of counts in each bin, i.e. takes the average of the intrinsic bins in each time bin.

- **Type of binning for SPECTRUM:** This option determines how the binned values in spectra are calculated: "SUM" (the default) gives the total accumulated counts per channel in each bin; "RATE" divides the total accumulated counts per channel by the temporal binsize; "MEAN" gives the mean value of counts per channel in each bin, i.e. takes the average of the intrinsic bins in each time bin.

- **Maximum acceptable intensity for Light Curve:** Allows you to exclude from the output light curve bins above a threshold. The default is not to set any threshold.

- **Maximum acceptable intensity for Spectrum:** Allows you to exclude from the output spectrum bins above a threshold. The default is not to set any threshold.

- **Column titles for SUM written as header Keywords:** Does not apply to RXTE data.

- **Column titles for MEAN written as header Keywords:** Does not apply to RXTE data.

- **Starting time for summation:** Gives the option of setting a later time for the beginning of the extraction than the default - specified with "INDEF" - which is to start with the first valid time stamp (before GTI are applied, that is). Times must be given in absolute time, i.e. in seconds with an offset of TSTART + TIMEZERO, the values of which are in the header of the data file.

- **Ending time for summation:** Gives the option of setting an earlier time for the end of the extraction than the default - specified with "INDEF" - which is to end with the last valid time stamp (before GTI are applied, that is). Times must be given in absolute time, i.e. in seconds with an offset of `TSTART + TIMEZERO`, the values of which are in the header of the data file.

- **Input time intervals t1-t2, t3-t4:** Gives the option of making time selections directly at the prompt *or* in an ASCII file, e.g. as derived from a light curve using timetrans. Regarding this option:

  - Times must be given in absolute time, i.e. in seconds with an offset of `TSTART + TIMEZERO`, the values of which are in the header of the data file.

  - If entering time ranges directly at the prompt, the format should be, e.g.

    ```
    63460012.89412300-63460012.89412400,63460012.89412412-63460012.89412512
    ```

  - If entering time ranges in a file, the filename should be preceded by "@" and file format should be, e.g.

    ```
    63460012.89412300 63460012.89412400
    63460012.89412412 63460012.89412512
    ```

    where the last character is a carriage return.

  - The time selections will be combined with those in the FITS GTI file, if present.

  - The time selections will be OR'ed with those in the input science array files, if applied.

  - If your time selections comprise simply a start time and an end time, you can enter them instead at the timemin and timemax prompts.

- **Minimum energy bin to include in Spectra:** Allows the user to set the energy channel at which extraction begins. Please note that channels are specified here with respect to the full 256-channel PCA or 256-channel HEXTE pass bands. Be careful when dealing with input data files in which the channels are binned. For example, in PCA Standard-2 files, the first five channels (0-4) are combined into one: setting the minimum energy bin to 3 would cause this first combined bin to be included in the extraction because it encompasses unbinned energy channel 3. To convert binned channels to unbinned channels, use the Perl script **chantrans**.

  Although the prompt refers to spectra, this option can be used to restrict the pass band for a light curve.

- **Maximum energy bin to include in Spectra:** Allows the user to set the energy channel at which extraction stops. Please note that channels are specified here with respect to the full 256-channel PCA or 256-channel HEXTE pass bands. Be careful when dealing with input data files in which the channels are binned. For example, in PCA Standard-2 files, the last six channels (250-255) are combined into one: setting the minimum energy bin to 250 would cause this last combined bin to be included in the extraction because it encompasses unbinned energy channel 250. To convert binned channels to unbinned channels, use the Perl script **chantrans**.

  Although the prompt refers to spectra, this option can be used to restrict the pass band for a light curve.

- **Input channels to be retained:** Allows the user to specify the channels in the input files from which data are to be extracted. Please note:
  - This option is best used for light curves, since it does not cause any rebinning of the output spectrum (cf. the prompt Input channels for each bin below). For example, specifying channels 5-25, 45-70 and 76-180 for PCA Standard 2 data would lead to the output of a spectrum with 129 channels (the resolution of the configuration) but with zero counts in those channels outside the specified ranges.
  - Channels are specified here with respect to the full 256-channel PCA or 256-channel HEXTE pass bands. Be careful when dealing with input data files in which the channels are binned. For example, in PCA Standard-2 files, channels 54-135 are binned by a factor of 2 while channels 136-237 are binned by a factor of 3: setting channel ranges 54-134 and 138-237 would cause data to be extracted from channels 54-135 and 136-237 because channel 134 cannot be separated from channel 135 and channel 138 cannot be separated from channels 136 and 137. To convert binned channels to unbinned channels, use the Perl script **chantrans**.
  - If entering channel ranges directly at the prompt, the format should be, e.g. 5-25, 45-50, 76-180
  - If entering channels in a file, the filename should be preceded by "@" and file format should be, e.g.

    ```
    5 25
    45 70
    76 180
    ```

    where the last character is a carriage return. The Perl script **chantrans** can be used to generate such a list.

- **Input channels for each bin:** Allows the user to specify how input channels are to be binned into output channels. This option has two main practical uses:

  1. **Combining input data files with different channel binning.** Provided the input channels and output channels are compatible, it is possible to combine differently binned input files and extract a single light curve and/or spectrum.
  2. **Rebinning an output spectrum.** Provided the input channels and output channels are compatible, it is possible to produce an output spectrum with coarser binning than the input data. In effect, this is like running the ftool **rbnpha**.

  In addition, please note

  - Channels are specified here with respect to the full 256-channel PCA or 256-channel HEXTE pass bands. Be careful when dealing with input data files in which the channels are binned, especially if your input files are binned differently. Make sure that the output bins can include all the valid input bins from all the files. To convert binned channels to unbinned channels, use the Perl script **chantrans**.

- If entering channel ranges directly at the prompt, the format should be, e.g. 5-25, 45-50, 76-180.

- If entering channels in a file, the filename should be preceded by "@" and file format should be either like that of chantrans output, e.g.

```
5 25
45 70
76 180
```

where the last character is a carriage return. Or it can be in the form of the `CPIX` keyword (without the `CONTINUE` keywords, e.g., for the `SpecDet0` (the ninth) column in HEXTE Archive data files:

```
1CPIX9  = '0~63;2,64~127;4,128~255;8'
```

- **Perform a dryrun:** A dry run differs from the real thing in two respects: the on-screen error and warning messages are more extensive, and no output files are produced. Obviously, a dry run is a good idea if you are unsure of what your inputs will produce or if you want to find out why it did not produce what you expected.

Data Reduction Scenarios: Using Saextrct

The following "real-life" scenarios are intended to show the range of saextrct's data reduction capabilities. For the sake of brevity, only the inputs that differ from those in the above example are described.

- **PCA extractions**
  - Light curve: Standard-2, 64-s bins, 2-6 keV
  - Spectrum: pcabackest file, top layer only

- **HEXTE extractions**
  - Source and background spectra: Archive mode, Cluster-0, all detectors

---

Light curve: PCA Standard-2, 64-s bins, 2-6 keV

Creating light curves for specific energy bands is a common task in X-ray astronomy. The key first step is to determine what channels correspond to the desired energy band. For the PCA, this is best done via an extracted spectrum, as the instrument has undergone several high-voltage gain changes since launch, i.e. the energy-to-channel conversion depends on epoch:

1. First, use saextrct to extract a spectrum from your Standard-2 data files (just one file will do). Defaults can be entered for all inputs, if you like, or you can use a "good" spectrum, i.e. one that

has been extracted with all your desired GTI and other selections.

2. Generate the corresponding response matrix with pcarsp.

3. Read the spectral file into xspec with the "data" command (pcarsp automatically writes the name of the response file into the header of the spectral file).

4. Issue the xspec command "ignore 0.0-2.0 6.0-**" which causes the channels corresponding to 0.0-2.0 keV and above 6.0 keV to be excluded from fitting (the ignore command interprets integers as channels, reals as energies in keV).

5. Issue the xspec command "show" and look in the screen output for a line like:

```
  Noticed channels     6 to    19
```

   which explains that for this particular Standard-2 spectral file, 2-6 keV corresponds to channels 6-19. Quit xspec.

6. Remembering that saextrct requires original channels, we have to convert this channel range from the 129-channel Standard-2 binning to the full 256-channel binning. The script chantrans will do this. In this case, 6-19 corresponds to 10-23.

7. Finally, we can now run saextrct to extract the light curve. The channel range, 10-23, corresponding to 2-6 keV should be entered at the prompt:

```
Input channels to be retained 1-2,3-4 (INDEF):[INDEF] 10-23
```

Spectrum: pcabackest file, top layer only

The ftool pcabackest models the instrumental background of the PCA and produces output files that look in many respects like a Standard-2 data file. With these files as input, saextrct can be used to extract background light curves or, as in this example, a background spectrum from the top layer of all five PCU:

1. First run pcabackest to generate an model background datafile, choosing the option of separating the detector layers.

2. Run the ftool flcol on the file generated by pcabackest, (pca_back.fits, say) to list the columns.

3. From this list, create the file called, say, top_layer.cols, containing the lines:

```
X1LSpecPcu0
X1RSpecPcu0
X1LSpecPcu1
X1RSpecPcu1
X1LSpecPcu2
X1RSpecPcu2
X1LSpecPcu3
```

```
X1RSpecPcu3
X1LSpecPcu4
X1RSpecPcu4
```

Make sure the file ends with a carriage return.

4. Run saextrct, using pca_back.fits as the input file and entering top_layer.cols at the prompt:

```
Name of COLUMNS to be accumulated (GOOD):[GOOD] @top_layer.cols
```

Spectra, source and background: HEXTE Archive, all detectors

Before running HEXTE science array files through saextrct, the data have to be first filtered by cluster position, i.e. separated into on-source and off-source parts. As explained in the footnote, this is done using the ftool fselect. Then, assuming that you've produced two sets of files - one on-source, one off-source - proceed as follows:

1. Verify the names of the columns containing the spectral data by running the ftool flcol on one of the Archive mode data files. The columns, one per detector, are:

```
SpecDet0
SpecDet1
SpecDet2
SpecDet3
```

which should be entered at the prompt:

```
Name of COLUMNS to be accumulated (GOOD):[GOOD] SpecDet0 SpecDet1 SpecDet2 SpecI
```

2. Run saextrct on the on-source files, calling the output spectrum, e.g., herx1_on.pha.

3. Run saextrct on the off-source files, calling the output spectrum, e.g., herx1_off.pha.

4. Adjust the exposure of the off-source spectrum. This is necessary because the time spent by the cluster moving between on-source and off-source positions (2 seconds) is not accounted for in the data files, i.e. the structure of the files implies that on-source and off-source rows each contain 16 seconds of data. To make the adjustment:

    1. First verify the cluster dwell time by running fdump on one of the original HEXTE archive mode files to see how often, in terms of the 16-s rows, the cluster changes position:

```
rufus [78] [day] redux: fdump prhead=no /data/kaah5/xtegof/day/herx1/P00023
3-02-01-00/hexte/FS52_3eb6dcb-3eb7570
Name of optional output file[STDOUT]
Names of columns[] ClstrPosition
Lists of rows[-]


         ClstrPosition
         Count
      1       5
```

```
2       0
3       1
4       0
5       5
6       0
.       .
.       .
.       .
```

Here, we see that the cluster position changes from 0 (on-source for 1.5-degree rocking) to 1 or 5 (+1.5-degree & -1.5-degree offsets, respectively) every row, i.e. every 16 seconds. This means that each off-source row really contains 16 - 2 x 2 = 12 seconds of data, and, consequently, the total exposure for a spectrum extracted from off-source data in this case is overestimated by a factor of 16/12 = 1.33.

2. Before making the exposure correction itself, first use the ftool **fkeyprint** (or fdump) to determine the current value of the exposure in the off-source spectrum:

```
rufus [80] [day] redux: fkeyprint herx1_off.pha EXPOSURE

# FILE: herx1_off.pha
# KEYNAME: EXPOSURE

# EXTENSION:    0

# EXTENSION:    1
EXPOSURE=  9.28000000000000E+02 / Exposure time
```

3. Next, use the ftool **fmodhead** to change the value of the keyword from its current value of 928 seconds to 928 x 12/16 = 696 seconds:

```
fmodhead herx1_off.pha modhead.txt
```

where modhead.txt is an ASCII file containing the line:

```
EXPOSURE 696.0
```

Extracting Science Event Data

As described in the *Data Files* chapter, science event files contain irregularly spaced time-stamped event words which, in the case of the PCA, are interspersed at regular time intervals with various EDS flags (also known as clock events). This means that extracting light curves and spectra from an event file involves removing the clock events, and, if desired, picking a subset of the valid events, e.g. data from PCU0 only. In practice, this entails defining a **bitmask**: a binary string (or set of strings) which the event words in the file must match in order pass through extraction.

Seextrct applies but does not define bitmask selections, so another program, sefilter, must be run before the extraction. This caveat applies to:

- All PCA and HEXTE data if you want to apply a particular bitmask, e.g. to pick out events from a single PCU or HEXTE detector.

- All PCA event mode configurations except Good Xenon regardless of whether you want to apply a particular bitmask (to remove the clock events - Good Xenon data files don't have clock events).

This section is split into two. The first, immediately following, covers running sefilter. The second covers running seextrct. As for the corresponding section on saextrct, a fully worked example is given along with some data reduction scenarios.

Defining Bitmasks with Sefilter

Before running sefilter, it's a very good idea to review the sections in the *PCA Issues* chapter that describe the various event-mode configurations, namely, Good Xenon and Generic event. Revise, too, the footnote on DDL. And it almost goes without saying that you should type **fhelp sefilter**.

Sefilter is a Perl script that helps you create an ASCII bitmask filter. The script then uses the bitmask to run either of the following:

- **fselect to create a single bitmask-filtered event file.** This is the only valid option if the bitmask includes alternatives, e.g. if you want to select events from *either* PCU0 anode X1L *or* PCU0 anode X1R (i.e. the top layer of PCU0). It may also be used if you want to create an intermediate filtered event file.

- **seextrct to create bitmask-filtered light curves and spectra.** Seextrct, unlike fselect, cannot implement a bitmask that contains alternatives. However, for simple bitmasks, e.g. events from PCU0 *and* PCU1, sefilter offers the choice of creating the filtered event file with fselect or of going straight to seextrct to create light curves and spectra.

An example of using sefilter follows, but please note the following:

1. Bitmask files are simple, containing only the coded selection criteria and no record of their origin. Moreover, neither fselect nor seextrct can check whether a given bitmask is sensible. Fortunately, sefilter contains built-in safeguards: after creating a valid bitmask, sefilter initiates either fselect or seextrct, whichever is appropriate, to apply the filter.

2. Bitmasks contain a prefix (b for binary, h for hex, o for octal - all RXTE bitmasks are binary) followed by a string of zeroes, ones or x's. The x is a wild card: for binary bitmasks it means "0 or 1". When you run sefilter, not selecting a particular token is equivalent to leaving all its values as x's. Practically, this means that if, say, you want events from all five PCU, you should not make a selection based on the D-token which would otherwise enable you to single out a particular PCU.

3. By default, and if no other selections are specified, sefilter creates a bitmask for PCA data which will select only good science events, i.e. those with the `M[1]{1}` marker or M-token. It is an ASCII file containing a line like

   ```
   Event == b1xxxxxxxxxxxxxx.
   ```

4. Although, for some configurations, channel information is contained in the bitmask, selection by channel is *not* performed via sefilter. Seextrct is used instead.

Example of using sefilter

Here, we'll use sefilter to generate a bitmask corresponding to the top layers of all the PCU. The input file, `E_1ms_128M_0_8s.list` is a list of data files in the `E_1ms_128M_0_8s` configuration, hence the @ sign. In this example, user input is in boldface.

```
rufus [41] [day] test: sefilter
Enter SE FITS file for filtering:[]@E_1ms_128M_0_8s.list
Enter the M-token to be processed (M[1]{1}):[M[1]{1}]
Enter the column name to be operated upon (Event):[Event]
Enter the output bitfile name to contain processed boolean expression:[]top.bit
```

Defaults are entered for the M-token and column name. We've named the filter file itself top.bit.

```
Beginning generation of filter description.
###########################################

This file contains a Z-token, specifically:
Z[D[0]&E[X1L],D[0]&E[X1R],D[0]&E[X2L],D[0]&E[X2R],D[0]&E[X3L],D[0]&E[X3R],D[1]&E
[X1L],D[1]&E[X1R],D[1]&E[X2L],D[1]&E[X2R],D[1]&E[X3L],D[1]&E[X3R],D[2]&E[X1L],D[
2]&E[X1R],D[2]&E[X2L],D[2]&E[X2R],D[2]&E[X3L],D[2]&E[X3R],D[3]&E[X1L],D[3]&E[X1R
],D[3]&E[X2L],D[3]&E[X2R],D[3]&E[X3L],D[3]&E[X3R],D[4]&E[X1L],D[4]&E[X1R],D[4]&E
[X2L],D[4]&E[X2R],D[4]&E[X3L],D[4]&E[X3R]].
We have 5 bits for this token.
So this token can take on values from 0 --> 31

The Z-token is SPECIAL. So we will deal with it in detail.
Number of Z-tokens is 30
D[0]&E[X1L] ==  0          D[2]&E[X2R] == 15
D[0]&E[X1R] ==  1          D[2]&E[X3L] == 16
D[0]&E[X2L] ==  2          D[2]&E[X3R] == 17
D[0]&E[X2R] ==  3          D[3]&E[X1L] == 18
D[0]&E[X3L] ==  4          D[3]&E[X1R] == 19
D[0]&E[X3R] ==  5          D[3]&E[X2L] == 20
D[1]&E[X1L] ==  6          D[3]&E[X2R] == 21
D[1]&E[X1R] ==  7          D[3]&E[X3L] == 22
D[1]&E[X2L] ==  8          D[3]&E[X3R] == 23
D[1]&E[X2R] ==  9          D[4]&E[X1L] == 24
D[1]&E[X3L] == 10          D[4]&E[X1R] == 25
D[1]&E[X3R] == 11          D[4]&E[X2L] == 26
D[2]&E[X1L] == 12          D[4]&E[X2R] == 27
D[2]&E[X1R] == 13          D[4]&E[X3L] == 28
D[2]&E[X2L] == 14          D[4]&E[X3R] == 29
```

After reading the datafile headers, sefilter prints on the screen the part of the `TEVTB2` keyword describing the selectable parts of the template. For this particular configuration, Z-tokens enumerate the various combinations of PCU and anode. We'll be selecting the ones corresponding to the X1L and X1R anodes of PCU0-5.

```
Do you want to select any of these? [Yes, No]: yes
Enter the numbers associated with the items you want to select: [0-5,15,24-30]
The above syntax must be followed (no negative values!): 0-1,6-7,12-13,18-19,24-25
You entered 0-1,6-7,12-13,18-19,24-25.
```

```
Is this correct? [Yes, No]: yes
The Z-token expression built up was:  Z[] <= 1 | Z[] >= 6 & Z[] <= 7 | Z[] >= 12
 & Z[] <= 13 | Z[] >= 18 & Z[] <= 19 | Z[] >= 24 & Z[] <= 25
Z-Filter is  Z[] <= 1 | Z[] >= 6 & Z[] <= 7 | Z[] >= 12 & Z[] <= 13 | Z[] >= 18
& Z[] <= 19 | Z[] >= 24 & Z[] <= 25


The constructed filter is:
(  Z[] <= 1 | Z[] >= 6 & Z[] <= 7 | Z[] >= 12 & Z[] <= 13 | Z[] >= 18 & Z[] <= 1
9 | Z[] >= 24 & Z[] <= 25  )

Running sebitmask to create the bitmask pattern:
Completed running sebitmask:

Expression generated by SEFILTER stored in outfile_368
Do you want to save this file? [Yes,No]no
Deleting outfile_368
```

The file we've elected to delete, outfile_368, contains the lines `( Z[] <- 1...` etc. It is not essential. The more important file, top.bit, contains the bitmask itself, as shown below (sebitmask is the actual ftool that creates it).

```
Updating parfile for sebitmask:

Final bitmask generated by SEBITMASK is:
Event == b1xxxxxxxxxxxxxxx &&
(Event <= bxxxxxxxx00001xxx ||
Event >= bxxxxxxxx00110xxx &&
Event <= bxxxxxxxx00111xxx ||
Event >= bxxxxxxxx01100xxx &&
Event <= bxxxxxxxx01101xxx ||
Event >= bxxxxxxxx10010xxx &&
Event <= bxxxxxxxx10011xxx ||
Event >= bxxxxxxxx11000xxx &&
Event <= bxxxxxxxx11001xxx )
```

After creating the bitmask, sefilter then prompts us for an output file name for the filtered events (top.fits). Since our bitmask contains alternatives, fselect, not seextrct, will be run.

```
Enter the output filename to contain the processed data:[]top.fits
Using output of sebitmask as input and running Fselect:
Completed running fselect and created top.fits
```

We can now input top.fits into seextrct to extract light curves and spectra.

Example: Creating a Light Curve from PCA E_1ms_128M_0_8s Data

Here, we'll use seextrct to create a light curve with 10-ms bins from data in the PCA configuration E_1ms_128M_0_8s. This is a continuation of the above example where we used sefilter to create a bitmask corresponding to the top layer of the PCA which was then used by fselect to create the file top.fits. Apart from excluding events from the middle and bottom layers, top.fits also excludes clock events. Please assume we have already run: a) maketime to create a GTI file excluding elevations greater than 10 degrees (elv_gt_10.gti) and b) timetrans to create an ASCII time-interval file from a

Standard-2 lighcurve (flare.timeint). No other selections are applied. In addition:

- User inputs are shown in boldface.
- Default inputs, where appropriate, are included in square brackets and are entered with a carriage return (not shown explicitly).
- To avoid repetition, some links will take you to the saextrct section in cases where seextrct inputs that have identical explanations.
- For full documentation about seextrct, type **fhelp seextrct**.

The corresponding screen dialogue is as follows:

```
rufus [93] [day] test: seextrct

Running SEEXTRCT version 3.6
==============================================
Input file name or @file-of-filenames:[] top.fits
Input GTI files to be OR'd with INFILE (APPLY):[APPLY]
Input GTI file to be AND'd with INFILE:[-] elv_gt_10.gti
Root name for output file:[sac] flare_10ms
Name of TIME column:[TIME]
Name of COLUMN to be accumulated:[Event]
Input the binsize in seconds, use 0.1 etc. if nec (INDEF):[INDEF] 0.01
Minimum acceptable fractional exposure (INDEF):[INDEF]
Chose print option, LIGHTCURVE, SPECTRUM, or BOTH:[BOTH] lightcurve
Type of binning for LIGHTCURVE: (SUM, RATE, MEAN):[RATE]
Type of binning for SPECTRUM (SUM, RATE, MEAN):[SUM]
Maximum acceptable intensity for Light Curve (INDEF):[INDEF]
Maximum acceptable intensity for Spectrum (INDEF):[INDEF]
Column titles for SUM written as header Keywords:[-]
Column titles for MEAN written as header Keywords:[-]
Starting time for summation (INDEF):[INDEF]
Ending time for summation (INDEF):[INDEF]
Input time intervals t1-t2,t3-t4 in seconds (INDEF):[INDEF] @flare.timeint
Minimum energy bin to include in Spectra (INDEF):[INDEF]
Maximum energy bin to include in Spectra (INDEF):[INDEF]
Input channels to be retained 1-2,3-4 (INDEF):[INDEF]
Input channels for each bin 1-5,6-256 (INDEF):[INDEF]
```

Explanation of the Various Inputs in the Example above

- **Name of COLUMN to be accumulated:** Unlike science array files, science event files have just one column for the science data. It's called `Event` and contains the binary event words.

- Introduction
- Time Selections
- Extracting Science Array Data
- Extracting Science Event Data

- Appendix: HEXTE Cluster Position

Appendix: HEXTE Cluster Position

HEXTE data files contain data from only one of the two clusters but are not split further into on-source and off-source files. To create separate data files for further processing, you can use the ftool **fselect**. How exactly this is done depends on whether your HEXTE data are in science array or science event format. In both cases, fselect operates on the `ClstrPosition` column, but the selection criteria differ.

For HEXTE science array data, the values of `ClstrPosition` corresponding to the on-source position are 0 and 3 for 1.5-degree and 3-degree rocking, respectively. This means that to create a file containing only on-source data, the following fselect criterion should be used:

```
ClstrPosition.eq.0.or.ClstrPosition.eq.3
```

And to create the off-source file, use:

```
ClstrPosition.ne.0.and.ClstrPosition.ne.3
```

For HEXTE science event data, the values of `ClstrPosition` corresponding to the on-source position are 1 (or 65) and 8 (or 72) for 1.5-degree and 3-degree rocking, respectively. This means that to create a file containing only on-source data, the following fselect criterion should be used:

```
ClstrPosition.eq.1.or.ClstrPosition.eq.65.or.ClstrPosition.eq.65.or.
  ClstrPosition.eq.72
```

And to create the off-source file, use:

```
ClstrPosition.ne.1.and.ClstrPosition.ne.65.and.ClstrPosition.ne.65.and.
  ClstrPosition.ne.72
```

For example, to create the file event_off.fits containing only off-source data from the science event file `FS50_3eb70b0-3eb8950`:

```
kaah[16]% fselect
Name of FITS file and [ext#][] FS50_3eb70b0-3eb8950
Name of output FITS file[] event_0_off.fits
Selection Expression[] ClstrPosition.eq.1.or.ClstrPosition.eq.65.or.ClstrPosition.eq
  ClstrPosition.eq.72
```

HEASARC Director: Dr. Nicholas E. White, white@adhoc.gsfc.nasa.gov, 301-286-8443
Technical Rep: Sherri Calvo, sherri.calvo@gsfc.nasa.gov, 301-286-5668
Questions/Comments/Feedback to: feedback@athena.gsfc.nasa.gov
Tell me about black holes, astronomy, and more!