lccorr_pcms

January 12, 2017

Abstract

This task calculates and optionally applies an exposure correction for timeseries.

1 Instruments/Modes

Instrument	Mode
EPIC MOS:	IMAGING
EPIC PN:	IMAGING

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

The task **lccorr** takes, as its input, source and (optionally) local-background timeseries (TS) created by **evselect**, in which the time variation of the flux is represented by counts within time bins of equal duration. The main task of **lccorr** is to correct these time series for losses in flux due to a number of causes (exposure correction); these corrections are described in subsections 3.3 and 3.4. In addition, **lccorr** performs several other operations on these TS, which are described in subsection 3.5.

3.1 Caveats

The attention of the user is directed in particular to the following caveats:

- The attitude is assumed to be stable throughout the duration of the time series. If this is not the case, the correction may be suspect.
- Loss of events due to pileup is not corrected. This affects worst bright sources with sudden spikes in emission.

- Timing and burst mode are not yet supported.
- For the present, spectral information cannot be supplied to **lccorr**. Internally, the task assumes a flat spectrum.

3.2 Quick start.

What you need to get started:

- A Calibration Constituents File or CCF (pointed to by SAS_CCF). This is needed because the CCF contains all the details about the telescope needed to correct for events lost (exposure).
- An EPIC event list. This is mainly needed for the timing and bad pixel information contained in the `EXPOSUnn`, `STDGTInn`, `OFFSETS` and `BADPIXnn` extensions.
- A time series or light curve compiled from that event list. In addition to the data itself, this should contain a Data SubSpace (DSS). The DSS contains information about the selection of events used to make up the time series.
- The task needs to know the position of the source in order to account correctly for any part of the point spread function (PSF) chopped off by the edge of a spatial extraction region (however see also section 3.6). If you set `srcposstyle`='dss' (the default value for this parameter), **lccorr** will try to guess the source location from the information stored in the DSS. But the algorithm which has this job is not very clever. It is best if possible to inform the task about the source location by setting `srcposstyle`='user' and then supplying the source coordinates directly via parameters `srcra` and `srcdec`. The latter are angle-style params: their syntax can be found in the documentation of the `param` interface.

About the simplest possible call to **lccorr** is therefore as follows:

```
setenv SAS_CCF <name of CCF file>

lccorr eventset=<evlist name> srctsset=<src ts name> srcra=<ra of src>
  srcdec=<dec of src>
```

3.3 Exposure corrections - introduction:

A timeseries is created by **evselect** from an event list which includes events from the exposure/instrument/ccds of interest. It is customary to use the calibrated, whole-camera event lists which are made as one of the standard set of SAS products: **lccorr** has been designed under this assumption. The event-list parent of the timeseries must be supplied to **lccorr** as one of its inputs, and an event list that does not adhere to the product standard may not contain all the information required by **lccorr**.

The events that end up in the time series represent only a fraction of the photons from that source which impinge on the aperture of the mirror. There are two ways in which events may be lost:

- Events lost involuntarily, through inefficiencies of various sorts in the mirror-detector system. Examples:
  - The spatial variation in mirror efficiency across the field of view (vignetting);

- Imperfect quantum efficiency of the detectors;
- Loss of events outside the field of view, on bad pixels or in chip gaps;
- Loss of events in passing through an x-ray filter on the camera;
- Loss of events due to pileup (occurs when the event rate is so high that the probability of more than one event arriving per CCD per frame becomes significant).

- Events selectively discarded by the user. This is done usually in an attempt to discard excess events caused by factors unconnected with the source. Some examples:
  - Events outside of good time intervals (GTIs);
  - Spurious events from warm pixels;
  - Spurious events from xray fluorescence of the substrate;
  - Spurious events from electronic noise in the detector;
  - Background counts.

The real world is of course never that simple, and some losses are a mixture of both types - eg events that are discarded by the on-board electronics, or by the processing chain that constructs the event list. Some things that belong to the 'involuntary inefficiencies' list also behave like selections (eg the lack of events on a bad pixel), because of their all-or-nothing character.

The primary task of **lccorr** is to attempt to reconstruct the pre-loss flux values. Methods of doing this are discussed in [2]; the algorithms used in **lccorr** are derived from the equations within this reference. Essentially, the algorithm consists of modelling the original distribution of events over all the columns in the event list, subjecting this distribution (as far as is possible, and subject also to some user specification) to the same inefficiencies and selections as the real events; the original and the attenuated distributions are then both integrated over all degrees of freedom except time; the ratio between the resulting functions of time is the desired result, the exposure function fracexp.

There are 12 columns in the MOS event list: this suggests that an integration over 11 degrees of freedom (12 - TIME) are needed. Actually it is not that bad:

- The `RAWX`, `RAWY`, `CCDNR`, `DETX`, `DETY`, `X` and `Y` columns can be treated as just 2 spatial dimensions, since there is in practice little time-varying slippage between these coordinate systems.
- It is assumed that any selection on column `PHA` can be just as well done on `PI`; hence selections on `PHA` are not corrected. You'll be warned by the task if the DSS of the time series contains a selection on `PHA`.
- Many flag values are such that selection of events by these values is equivalent to a spatial selection. The flags that are recognized by **lccorr** as such are
  - EVATT_CLOSE_TO_CCD_BORDER
  - EVATT_CLOSE_TO_CCD_WINDOW
  - EVATT_CLOSE_TO_NODE_BOUNDARY
  - EVATT_CLOSE_TO_ONBOARD_BADPIX
  - EVATT_CLOSE_TO_BRIGHTPIX
  - EVATT_CLOSE_TO_DEADPIX
  - EVATT_OUT_OF_FOV
  - EVATT_ON_OFFSET_COLUMN
  - EVATT_NEXT_TO_OFFSET_COLUMN

- EVATT_CLOSE_TO_BADCOL
- EVATT_CLOSE_TO_BADROW

No other flags are recognized at present. You'll be warned by **lccorr** if it detects selection on non-recognized flags.

That brings the number to a much more manageable 5 dimensions: time, x, y, energy and pattern. The pristine event distribution is however only a function of the first four. **lccorr** makes the simplifying assumption that the shape (as opposed to the amplitude) of neither the spatial distribution nor the spectrum of the events is time-variable. The first is a very reasonable assumption but the latter is more or less forced upon us due to the cumbersome nature - a set of spectra sampled at various times - of the input information necessary to correct the time series in such circumstances. If the object of interest has such behaviour (which may be by no means a rare occurrence), a better alternative is to chop the spectrum into pieces for which the assumption of spectral stability is more reasonable, and perform light-curve correction on each separately.

The attenuation in the camera is a function of all 5 dimensions. It is assumed that, within a CCD, the time-varying part may be separated from the space-, pattern- and energy-varying part. The latter is assembled in practice from a vignetting function (depends on space and energy), a filter transmission (energy only) and a quantum efficiency (energy and pattern). Note that QE is in principle a function also of spatial coordinates; **lccorr** assumes that it is spatially uniform.

The selection, which in effect cuts up the 5-dimensional space into chunks, within which the event distribution multiplied by the attenuation function is to be integrated, permits considerable freedom. Any combination of filters on energy, on any of the spatial coordinates or flags, and on pattern will in principle be teased out correctly by the task, although a too-finely divided 5-dimensional space will not be conducive to numerical accuracy. Time filters however are only permitted through good time intervals (GTIs), and may not depend on selection on any of the other dimensions or columns except `CCDNR`. It is not anticipated that this will in practice greatly restrict the user.

lccorr offers command-line parameters to enable or disable calculation of many of the factors causing loss of events. These are tabulated immediately below and described in greater detail in the next subsection.

Factor:	Corrected?	Parameter:
Vignetting	optional, default=yes	`treatvignet`
Filter losses	optional, default=yes	`treatfiltertrans`
Quantum efficiency	optional, default=yes	`treatquantumeff`
Cosmic ray masking	optional, default=yes	`treatcosmicrays`
Dead time between frames	optional, default=yes	`treatdeadtime`
GTI selection	optional, default=yes	`treatgti`

The exposure function, after it is calculated, is stored in a column called `FRACEXP` in the output TS. Although this column is always present in the output, the values may or may not have been used to correct the rate values in that file. This choice is at the discretion of the user via the parameter `applycorrections`, and is recorded in the keywords `DEADAPP` and `VIGNAPP`. Note that although the OGIP standard for TS requires the presence of both these two keywords, representing respectively the time- and space-dependent parts of the exposure correction, it was found to be undesirable to separate the exposure contributions in this fashion within **lccorr**: the keywords are therefore 'ganged', ie given the same value of T or F by **lccorr**.

If `applycorrections` is set, the exposure correction is carried out by simply dividing the rates and errors by the relevant value of `FRACEXP`. For those time bins in which the value of `FRACEXP` is zero, the rates and errors are set to null (IEEE NaN).

3.4 Exposure corrections in detail:

These are described below under headings which take the name of the relevant command-line enabling parameter.

3.4.1 treatvignet

This enables evaluation of the vignetting function, which is a measure of the decrease in efficiency of the XMM telescope mirror with increase in the off-axis angle θ of the source.

3.4.2 treatfiltertrans

Enables calculation of the attenuation of xray flux due to the filter.

3.4.3 treatquantumeff

This parameter enables calculation of the detected fraction of events impinging on the ccd. This fraction, called the quantum efficiency (QE), is recorded in the CCF as a function of position and energy and is, in addition, divided into components, each of which is equal to the net QE multiplied by the fraction of events having a given pattern. Those values corresponding to selected event patterns must therefore be summed to obtain the net QE.

3.4.4 treatcosmicrays

A proportion of detected events are 'minimum ionising particles' (MIPs). During each ccd frame, pixels occupied by a MIP are not free to detect an xray; hence MIPs lead to xray flux loss. The fraction of affected pixels is stored in the `FRACEXP` column of the `EXPOSUnn` extensions (the 'nn' indicating the ccd/node) of the event list. Note that, although this column has the same name as that used to store the total exposure in the output TS, their values are not closely related.

3.4.5 treatdeadtime

Part of the ccd frame duration (T_f) is taken up by two sorts of dead time. The first is the time taken to read the ccd row by row (T_r). Events occurring during T_r are called out-of-time events (OOTEs) and are spread over the whole ccd column (see reference [2], for more details). If the source collection region has a small extent in the column direction, most of the OOTEs from sources nominally within the collection region will not fall within it and will therefore not find their way into the TS. An approximate correction can be calculated by evaluating $T_f/(T_f - T_r)$ (the smaller the extent of the region in the column direction, the better the approximation will be).

In some instrument modes there is an additional period of 'true' dead time, during which events are not recorded at all. For these modes the additional dead time is added to T_r in the expression above and the calculation carried out as before.

The quantity T_f - T_r is obtainable from the `TIMEDEL` column or keyword of the `EXPOSUnn` extensions of the event list, whereas the readout and other times are contained in the CCF.

3.4.6 treatgti

Events occurring outside of good time intervals (GTIs) will usually be discarded before the TS is made. Where the collection region overlaps just one ccd/node, flux discarded in this way is lost 100% and cannot be reconstructed. In these cases the value of the exposure is set to zero.

Reconstruction is possible where the collection region overlaps more than 1 ccd/node, not all of which are outside GTIs at the same time, or within time bins which contain a GTI boundary.

GTIs are extracted from the data subspace (DSS) of the relevant TS.

3.5 Additional processing

The following additional operations are carried out by **lccorr**:

3.5.1 Mandatory:

- The input timeseries may have values in either counts per bin (of integer data type int16 or int32) or counts s^{-1} (real32). If **lccorr** finds counts per bin in the input TS it converts them to (real32) counts s^{-1} in the output; the same goes for errors, if present. If an error column is not present, **lccorr** calculates Poissonian values and stores them in the output.
- If a background TS is supplied via the `withbkgtsset` and `bkgtsset` parameters, **lccorr** converts these values similarly to the above and stores them in columns `BACKV` and `BACKE` in the output file. If the source TS already contains columns of these names, the values these contain are copied to the output only if no separate background file is supplied to **lccorr**. If a separate background file is supplied, then the contents of `BACKV` and `BACKE` columns in the source TS are discarded. Columns of these names in the background TS are always discarded.
- If a background TS is supplied, but this has a different binning scheme to the source TS, **lccorr** rebins the background TS to the same binning scheme as the source TS. Bins which are outside the duration of the original background time series are filled with nulls; sections of the original background time series which extend beyond the duration of the source time series are discarded.
- The background TS values are multiplied by the ratio of the source to the background collection area (see subsubsection 5.5.3 for details on how these areas are calculated).
- Each event is initially assigned the central time of the readout frame in which it occurs. But the bin sizes of time series are in general not integer multiples of the CCD readout period. This has potential to cause aliasing or Moiré noise, illustrated as follows. Say we have the following (graphically illustrated) situation:

```
Frames boundaries along the time axis:

  |---------|---------|---------|---------|---------|---------|---------|

Number of events in each frame, aligned with the time of frame centre:
      6         6         6         6         6         6         6

Numbered bins:

      |--------- 1 ---------|--------- 2 ---------|--------- 3 ---------| etc

Events in bins: 18                   12                   12

but should be:  13.2                 13.2                 13.2
```

I have neglected the usual natural noise in the signal in order to illustrate the problem better. **lccorr** corrects for this problem. In the case that the collection region lies entirely on one ccd/node, the correction is done by applying the following formula:

```
                                                     bin_duration
true_rate(bin) = measured_rate(bin) x ------------------------------------ .
                                        (no. frames in bin) x frameDuration
```

The correction is more complicated when multiple ccds are involved: in that case the total frame time in the bin for each ccd must be weighted by the fraction of events recorded from that ccd.

The MOS event-processing chain contains a facility to dither the event times, which prevents the occurrence of binning noise. This is recorded in the keyword `RAND_TIM` in the `EXPOSUnn` extensions of the event list. If **lccorr** finds that this keyword is set, that ccd/node is not included in the binning noise correction. Such dithering is not yet done in PN.

3.5.2 Optional:

- Parameter 'subtractbkg': if this is set, the background TS is subtracted from the source TS. This affects just the `RATE` and `ERROR` columns of the output TS.

3.6 Correction of non-point-source time series:

So far the discussion has assumed that the 'source' time series is a record of events from a point-like source of x-rays. However it may be convenient also to be able to correct time series which arise from phenomena that are not spatially localized. Background events, whether arising from true x-rays or from energetic particles which have collided with the detector, are one example of such. For such a time series, it is clearly inappropriate to use the point-source PSF to calculate the fraction of events lost to non-live areas of the detector. Ideally **lccorr** would allow the user to provide an image of the source. Such a facility is planned, but at present all that can be done is to allow the user to provide a list of weights which gives the fraction of the events received onto each of the CCDs of the detector. The mode is accessed via the parameters `srcweightstyle` and `srcweightsnode0`. A typical call of **lccorr** in this non-point-source mode might be as follows:

```
lccorr eventset=<MOS evlist name> srctsset=<src ts name> withbkgtsset=no
  srcweightstyle=user srcweights='1.2 1.0 1.0 1.0 1.0 1.0 1.0'
```

To avoid breaking users' scripts, the default behaviour of **lccorr** if these new parameters `srcweightstyle` and `srcweightsnode0` are omitted has been left exactly the same as before.

Note that events lost to vignetting, to absorption by the filter or to quantum inefficiency cannot be estimated in this mode: only time-dependent corrections can be made. Hence the parameters `treatvignet`, `treatfiltertrans` and `treatquantumeff` have no effect.

4 Parameters

This section documents the parameters recognized by this task (if any).

Parameter	Mand	Type	Default	Constraints

srctsset	yes	dataset		

Name of the input timeseries.

eventset	yes	dataset		

Name of the whole-camera event list (for either MOS1, 2 or PN, as appropriate).

withbkgtsset	no	boolean	no	

Look for a bkg timeseries name on the command line.

bkgtsset	no	dataset	bkg_ts.ds	

Name of the input local background timeseries.

bkgweightstyle	no	string	flat	none—user—flat

In order to correct properly for GTIs, **lccorr** needs to know what fraction of events fall on which CCD. (It is assumed that this fraction does not vary over time.) If the spatial distribution of the events is known, these weights can be computed by projecting this distribution against the detector, masked by the selection region; alternatively it may on occasion be simplest just input the CCD weights directly. **lccorr** now allows the user several options (with possibly more to come in future). If `bkgweightstyle`=‘none’, the task assumes that all the weights equal 1. If `bkgweightstyle`=‘user’, the weights are read from parameter `bkgweightsnode0` (and, optionally, `bkgweightsnode1`). Finally, for `bkgweightstyle`=‘flat’, the task assumes a uniform spatial distribution of events and projects this distribution against the masked detector CCDs to calculate the weights.

bkgweightsnode0	yes	real list		$0 \leq$ `bkgweightsnode0`

If `bkgweightstyle`=‘user’, this list of values is read by the task. Each value, as a fraction of their sum, should be the fraction of events that are expected to fall on the respective CCD. (If both primary and redundant nodes were used to readout any CCD, the respective weight in the `bkgweightsnode0` list should instead be that fraction of bkg events expected to fall on the primary-node half of the CCD. The user should also in this case set `withbkgnode1`=‘yes’ and supply the redundant-node weights via `bkgweightsnode1`.) The user must supply as many values as there are CCDs in the detector: ie 12 for PN and 7 for MOS. The weights should be listed in order of increasing CCD number.

withbkgnode1	no	boolean	no	

The task reads this parameter if `bkgweightstyle`=‘user’. `withbkgnode1` instructs the task to look for a list of redundant-node weights in parameter `bkgweightsnode1`.

bkgweightsnode1	yes	real list		$0 \leq$ `bkgweightsnode1`

If `bkgweightstyle`=‘user’ and `withbkgnode1`=‘yes’, this list of values is read by the task. Each value, as a fraction of their sum, should be the fraction of events that are expected to fall on the redundant-node half of the respective CCD. The user must supply as many values as there are CCDs in the detector: ie 12 for PN and 7 for MOS. The weights should be listed in order of increasing CCD number.

bkgarea	no	real	1.0	$0 <$ `bkgarea`

If `bkgweightstyle`=‘user’, this value is read by the task. The user should provide the area of the extraction region, in units of CCD pixels, not counting dead pixels, chip gaps or other non-live areas of the detector. This parameter is necessary if the background is to be subtracted from the source, in which case the ratio between the areas of the source and background extraction regions must be known.

subtractbkg	no	boolean	no	

Whether to subtract the background from the source count rate.

outset	no	dataset	output_ts.ds	

Name of the output dataset containing the corrected source and background time series.

tempset	no	dataset	temp_output_ts.ds	

Temporary name of output dataset.

srcweightstyle	no	string	psf	none—user—psf

In order to correct properly for GTIs, **lccorr** needs to know what fraction of events fall on which CCD. (It is assumed that this fraction does not vary over time.) If the spatial distribution of the events is known, these weights can be computed by projecting this distribution against the detector, masked by the selection region; alternatively it may on occasion be simplest just input the CCD weights directly. **lccorr** now allows the user several options (with possibly more to come in future). If `srcweightstyle`='none', the task assumes that all the weights equal 1. If `srcweightstyle`='user', the weights are read from parameter `srcweightsnode0` (and, optionally, `srcweightsnode1`). Finally, for `srcweightstyle`='psf', the task assumes that the source is point-like and projects the appropriate Point Spread Function (PSF) against the masked detector CCDs in order to calculate the weights.

srcweightsnode0	yes	real list		0 ≤ `srcweightsnode0`

If `srcweightstyle`='user', this list of values is read by the task. Each value, as a fraction of their sum, should be the fraction of events that are expected to fall on the respective CCD. (If both primary and redundant nodes were used to readout any CCD, the respective weight in the `srcweightsnode0` list should instead be that fraction of src events expected to fall on the primary-node half of the CCD. The user should also in this case set `withsrcnode1`='yes' and supply the redundant-node weights via `srcweightsnode1`.) The user must supply as many values as there are CCDs in the detector: ie 12 for PN and 7 for MOS. The weights should be listed in order of increasing CCD number.

withsrcnode1	no	boolean	no	

The task reads this parameter if `srcweightstyle`='user'. `withsrcnode1` instructs the task to look for a list of redundant-node weights in parameter `srcweightsnode1`.

srcweightsnode1	yes	real list		0 ≤ `srcweightsnode1`

If `srcweightstyle`='user' and `withsrcnode1`='yes', this list of values is read by the task. Each value, as a fraction of their sum, should be the fraction of events that are expected to fall on the redundant-node half of the respective CCD. The user must supply as many values as there are CCDs in the detector: ie 12 for PN and 7 for MOS. The weights should be listed in order of increasing CCD number.

srcarea	no	real	1.0	0 < `srcarea`

If `srcweightstyle`='user', this value is read by the task. The user should provide the area of the extraction region, in units of CCD pixels, not counting dead pixels, chip gaps or other non-live areas of the detector. This parameter is necessary if the background is to be subtracted from the source, in which case the ratio between the areas of the source and background extraction regions must be known.

srcposstyle	no	string	user	user—keyword—dss

If `srcweightstyle`='psf', the task attempts to calculate the fraction of events falling on each CCD by projecting the PSF onto the detector. To do this correctly, the task needs to know the RA and dec of the source. The task provides the user, via the present parameter, with a choice of ways to specify this position. If `srcposstyle`='user', the task reads the source position from parameters `srcra` and `srcdec`; if 'keyword', the task looks for keywords `SRC_RA` and `SRC_DEC` (in decimal degrees) in the dataset given in parameter `srcposset`; if 'dss', the task attempts to deduce the centre of the spatial selection regions stored in the Data Sub Space (DSS) of the input source time series.

srcra	yes	angle	0.0	0 ≤ `srcra` < 360

RA of the source. Read by the task if `srcposstyle`='user'.

srcdec	yes	angle	0.0	-90 $\leq$ `srcdec` $\leq$ 90

Dec of the source. Read by the task if `srcposstyle`=‘user’.

srcposset	no	dataset	srcposset.ds	

If `srcposstyle`=‘keyword’, the present parameter is read by the task. It contains the name of the dataset containing source position keywords `SRC_RA` and `SRC_DEC`. Their values should be in decimal degrees.

efrac	yes	real	0.95	0 < `efrac` < 1

The PSF model is truncated to a square area which contains (approximately) this fraction of the total flux. Read by the task if `srcweightstyle`=‘psf’.

treatvignet	no	boolean	yes	

Calculate corrections for the mirror vignetting?

treatfiltertrans	no	boolean	yes	

Calculate corrections for the filter attenuation?

treatquantumeff	no	boolean	yes	

Calculate corrections for quantum efficiency of the ccds?

treatcosmicrays	no	boolean	yes	

Calculate corrections for cosmic ray masking?

treatdeadtime	no	boolean	yes	

Calculate corrections for frame readout dead time?

treatgti	no	boolean	yes	

Calculate corrections for events lost outside GTIs?

treatalias	no	boolean	no	

Where the event times are not dithered within the frame duration (the default for PN event lists at time of writing, on 23 Sept 2004), one can generally expect that the binned events will exhibit some form of periodic noise at the difference frequency (plus harmonics) between the time series binning frequency and the CCD frame readout frequency. If `treatalias` is set ‘yes’, **lccorr** will attempt to correct this effect. At present it is recommended to leave this parameter at default.

applycorrections	no	boolean	yes	

Apply the exposure corrections to source and background count rates?

5 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

noSrcArea *(error)*

The task calculated that no events from the source data set passed the selection expression.

badInstrument *(error)*
Instrument not supported.

tsNoData *(error)*
There are no rows in the time series `RATE` table.

tsTooManyEbands *(error)*
The time series contains data in more than the 1 allowed energy band.

tsBadTimeUnits *(error)*
The time series has inconsistent time units.

tsNoTimes *(error)*
The timeseries has neither a `TIME` column nor a `TIMEDEL` keyword.

tsDifferentInstrumeKwds *(error)*
When a background series is supplied: source and background timeseries `INSTRUME` keywords don't match.

tsDifferentTimeunitKwds *(error)*
When a background series is supplied: source and background timeseries `TIMEUNIT` keywords don't match.

tsDifferentMJDrefKwds *(error)*
When a background series is supplied: source and background timeseries `MJDREF` keywords don't match.

noTimeDel *(error)*
No TIMEDEL information (either column or keyword) was found in the file.

bkgTimeBinsLeZero *(error)*
When a background series is supplied: an error during the rebinning.

srcTimeBinsLeZero *(error)*
When a background series is supplied: an error during the rebinning.

bkgErrorsNegative *(error)*
When a background series is supplied: one or more of the background `ERROR` values was fond to be negative.

badTimeDel *(error)*
The value of the `TIMEDEL` keyword is significantly different from the approximate value calculated from the time series itself.

tsBadUncertaintyType *(error)*
The `POISSERR` keyword is set to F, but there is no `ERROR` column.

tsCountsLeZero *(error)*
`COUNT` values ¡ 0 detected.

evlistWrongInstrument *(error)*
The `INSTRUME` keyword in the event list doesn't match that of the time series.

noDss *(error)*
The time series has no data subspace. The task cannot determine which filtering criteria were used without this.

noDssComponents *(error)*
The data subspace of the time series has 0 components. The task cannot determine which filtering criteria were used in this case.

noCcdsPassDss *(error)*
The data subspace of the time series will not pass any values of `CCDNR`.

evlistWrongTimeunit *(error)*
The `TIMEUNIT` keyword in the event list doesn't match that of the time series.

evlistWrongMJDref *(error)*
The `MJDREF` keyword in the event list doesn't match that of the time series.

noExposuExtensions *(error)*
Events were selected for a CCD for which there is no `EXPOSUnn` extension in the event list.

badInstrumeMode *(error)*
The instrument mode is unrecognized or unsupported by **lccorr**.

unableGetSrcPosFromDss *(error)*
Source position extraction from the DSS was requested, but this was not possible.

badSrcPosStyle *(error)*
Unrecognized value of `srcposstyle`.

noCcdFiltersInDssComponent *(error)*
Bad DSS.

noFilterThisCcdInDssComponent *(error)*
Bad DSS.

badChanType *(error)*
Unrecognized value of the `CHANTYPE` keyword. Currently the only value recognized by **lccorr** is 'PI'. The value is set, at the time the time series is made, from the value of the –energycolumn parameter of **evselect**. Unfortunately the default for this at time of writing is 'PHA' so to make time series which **lccorr** will accept you will need to set the –energycolumn value to 'PI' explicitly.

noERangeOverlap *(error)*
The energy ranges of the time series and the DSS don't overlap.

badRegridStatus *(error)*
Something went wrong with the rebinning of the PSF samples cube. You should contact the code developer.

badBadPixType *(error)*
The value of the `TYPE` column of the `BADPIXnn` table was not recognized.

badFlagFilter *(error)*
The bit-mask filter on column FLAG obtained from the data subspace was not well-formed.

noBkgArea *(warning)*
The task calculated that no events from the background data set passed the selection expression.
corrective action: The background time series in the output will be zero-valued.

deprecatedUnit *(warning)*
The unit of the time column was read as 'SECONDS', but the OGIP standard specifies 's'.
corrective action: 's' is assumed.

tsHasBackv *(warning)*
The time series has a background column.
corrective action: This column will be discarded.

noOverlap *(warning)*
When a background series is supplied: the background timeseries does not overlap in time at all with the source timeseries.
corrective action: The background columns in the output will be filled with nulls.

onlyPartialOverlap *(warning)*
When a background series is supplied: the background timeseries does not entirely overlap in time with the source timeseries.
corrective action: The non-overlapping rows of the background columns in the output will be filled with nulls.

tsNoPoisserrKwd *(warning)*
There is neither a `POISSERR` keyword nor an `ERROR` column in the time series.
corrective action: Poissonian errors assumed.

illegalTwoAxisDssFilter *(warning)*
The data subspace contains a two-axis filter on a pair of event-list columns outside the set `X/Y`. `RAWX/RAWY`, `DETX/DETY`.
corrective action: The task ignores this filter.

illegalDssFilterAxis *(warning)*
The data subspace contains a filter on column `PHA` of the event list: but **lccorr** cannot deal with this.
corrective action: The task ignores this filter.

timeSelectionTooComplicated *(warning)*
The task can handle only 1 type of time filter per CCD. However, a more complicated time-filtering scheme was found in the data subspace.
corrective action: The filter is ignored.

allSpatialMapsEmpty *(warning)*
None of the maps of spatial selections contained non-zero pixels for any CCD.
corrective action: None

dubiousDssEnergyRange *(warning)*
The energy selection recorded in the DSS appears to extend beyond the energy limits given by the E_MIN and E_MAX keywords of the time series.
corrective action: The DSS limits will be used. However the resulting exposure values might be inaccurate.

dubiousCalEnergyRange *(warning)*
The energy limits given by the `E_MIN` and `E_MAX` keywords of the time series appear to extend beyond the maximum extent of the energy limits defined in the **cal**.
corrective action: The smaller (ie, cal) limits will have to be used.

badBadPixCoords *(warning)*
`RAWX` < 0, `RAWX` $>$ 1+chipXsize, `RAWY` < 1 or `RAWY`+`YEXTENT` $>$ 1+chipYsize.
corrective action: Task ignores this row of the `BADPIXnn` table.

inconsistentNodeInfo *(warning)*
The CLOSE_TO_NODE_BOUNDARY event attribute was set, but **lccorr** can't find any bits of the flag mask with this value.
corrective action: No action.

pointlessWindowSelection *(warning)*
The CLOSE_TO_CCD_WINDOW event attribute was set, but no events get flagged with this value by the MOS software when the window occupies the whole ccd.
corrective action: No action.

6 Input Files

1. (Mandatory) A source timeseries dataset created by **evselect**. The header must contain the following keywords:

   - `INSTRUME` - either 'EMOS1', 'EMOS2' or 'EPN'

   The dataset must contain a binary table extension named `RATE` which must contain the following keywords:

   - `E_MIN`, `E_MAX` - energy interval of the TS.
   - `E_UNITS` - units of the previous, either 'eV' or 'keV'.
   - `TIMEUNIT` - eg 's'.
   - `MJDREF`

   The `RATE` extension must contain either a column `COUNTS`, recording the integer number of events occurring in the bin, or a column `RATE` recording the number of events in the bin divided by the bin duration in seconds. Both these columns may be of any numeric data type. If the `RATE` extension contains no `TIME` column then it must contain a either a `TIMEDEL` keyword or column, also an intelligible subset of `TIMEZERI`, `TIMEZERF` and `TIMEZERO` to denote the time of the centre of the first bin. Selection-region and GTI information must be stored in a data subspace (DSS) of this `RATE` extension.

2. (Optional) A background timeseries dataset, also created by **evselect**. If it is supplied it must have the same specifications as the source TS, and the values of the mandatory keywords must be the same.

3. (Mandatory) A calibrated event list for the relevant EPIC camera, created by either **emchain** or **epchain**. The header must contain the following keywords:

   - `INSTRUME` - must be the same value as the TS one.
   - `FILTER`

   The event list must contain the binary table extension `EVENTS`, which must contain the following keywords:

   - `TIMEUNIT` - must be the same value as the TS one.
   - `MJDREF` - must be the same value as the TS one.

   The event list must contain at least one binary table extension `EXPOSUnn`, where `nn` indicates the ccd/node. Each of these extensions must contain the following keywords:

   - `WINDOWX0`
   - `WINDOWY0`
   - `WINDOWDX`
   - `WINDOWDY`
   - Any other keywords needed to set the state of the cal via a call to CAL_setState.
   - an intelligible subset of `TIMEZERI`, `TIMEZERF` and `TIMEZERO`

   The `EXPOSUnn` extensions need not contain any columns, but correct exposure calculations cannot be performed unless each relevant extension (ie, each one corresponding to a ccd within the collection region) contains the following columns:

   - `FRACEXP` - real32.
   - `TIMEDEL` - real32.

- TIME - real64.

If the event list contains BADPIXnn extensions (not mandatory), these must contain the following columns:

- RAWX - int16.
- RAWY - int16.
- YEXTENT - int16.

7 Output Files

1. An OGIP-standard [1] timeseries. All the keywords of the header of the source time series are copied over. This output dataset contains the following binary-table extensions:

   - RATE. All the keywords in the RATE extension of the input source time series are copied over. The extension contains additional or altered keywords as follows:
     - DEADAPP, VIGNAPP - Both T if the output timeseries is corrected for exposure, F otherwise.
     - BACKAPP - T if the background has been subtracted from the output timeseries, F otherwise.
     - BKGRATIO - a real-valued keyword which records the ratio between the background collection area and the source collection area.

     The HDUCLAS3 keyword of the RATE extension is set to 'RATE'. If a background TS was supplied and has been subtracted, The HDUCLAS2 keyword of the RATE extension is set to 'NET'; otherwise it is set to 'TOTAL'.

     The column names of the RATE extension are as follows:
     - RATE - the source TS in counts s^{-1}.
     - ERROR - uncertainties in the source TS values.
     - BACKV - the local background TS, normalised to the source collection area.
     - BACKE - uncertainties in the background TS values, similarly normalised.
     - FRACEXP - the fractional exposure. This is a dimensionless number which can have any value between 0 to 1. (Values > 1 can occur if the events have not been randomized in time: see section 3.5.)

     All these columns have real32 data type.
   - SRC_GTIS. This extension contains GTIs which cover all the non-null RATE values.
   - BKG_GTIS. This extension (only present if withbkgtsset='yes') contains GTIs which cover all the non-null BACKV values.

   Note that the output dataset contains no DSS

8 Algorithm

NOTE!! This is now out of date - I haven't got around to writing a new one.

```
subroutine lccorr

foreach (srcTS, bkgTS) {
```

```
# Do basic existence and sense checks on the datasets.

RATE = COUNTS / bin duration
ERROR = ERROR / bin duration

# Retrieve region from DSS and make list of all pixels within it.

# Calculate the total area of the region.

if (use measured spectrum) {
  # Rebin spectrum if specified by user.
} else {
  # Make a 'natural' or model spectrum.
}

# Obtain the effective area as a function of energy.

# Calculate the correction for GTIs as a function of time.
tdx = calcGtiCorrection()

# Calculate the other time-dependent exposure factors (cosmic rays and
# dead time).
tdx = tdx * calcOtherTimeDepCorrns()

# Now calculate spatial contributions.
if (numCcdsInRegion == 1) {
  if (withspecset) {
    # Routine 00: 1 ccd/node, spectrum supplied.
    (numerator, denominator) = calcSpatial00(ccd, node, imageModel, etc)

  } else {
    # Routine 01: 1 ccd/node, model spectrum used.
    (numerator, denominator) = calcSpatial01(ccd, node, imageModel, etc)
  }
  FRACEXP = tdx * numerator / denominator

} else {
  if (withspecset) {
    # Routine 10: many ccds/nodes, spectrum supplied.
    (numerator, denominator) = calcSpatial10(imageModel, etc)
    FRACEXP = tdx * numerator / denominator

  } else {
    # Routine 11: many ccds/nodes, model spectrum used.
    FRACEXP = calcFracexp11(tdx, imageModel, etc)
  }
}

# Correct for external-source OOTEs.

# Correct for binning noise.

if (applyCorrections) {
  foreach (timebin) {
        if (FRACEXP(timebin)) > 0) {
          RATE[timebin]  =  RATE[timebin] / FRACEXP(t)
          ERROR[timebin] = ERROR[timebin] / FRACEXP(t)
        } else {
          RATE[timebin]  = setToNull
          ERROR[timebin] = setToNull
        }
      }
    }
  } # end foreach (srcTS, bkgTS)

  # Normalise bkgTS to src collection area.

  if (subtractBkg) {
    foreach (timebin) {
      RATE[timebin] = RATE[timebin] - BACKV[timebin]

      ERROR[timebin] = sqrt(ERROR[timebin] * ERROR[timebin]
      + bkgERROR[timebin] * bkgERROR[timebin])
    }
  }

  # Delete bkg FRACEXP column from output file.

  # Release output file.

  end subroutine lccorr
```

The workings of the four ‘spatial’ routines are very similar to each other. To serve as an example, the first is laid out below.

```
  subroutine calcSpatial00(ccd, node, imageModel, etc)

  # startEbin = first Ebin with centre energy greater than TS minimum E.
  # stopEbin = last Ebin with centre energy less than TS maximum E.

  numerator = 0.0
  denominator = 0.0
  if (imageModel eq ’point’) { # eqn 33, 34 of SSC-LUX-TN-0053
    foreach Ebin (startEbin .. stopEbin) {
      call CAL_getPSFmap(PSFmap)
      expr = 0.0
      foreach pixel (listOfPixels) {
        call CAL_getFilterTransmission(filterTrans)

        totalQe = 0.0
        foreach patternID (0 .. 31) {
          qeOfPattern = CAL_getQuantumEfficiency(patternID etc)
          totalQe = totalQe + qeOfPattern
        } # next patternID

        expr = expr + filterTrans * totalQe * psf(srcX, srcY, pixelX, pixelY, PSFmap)
      } # next pixel
      call CAL_getVignettingFactor(vigFactor)
      expr = expr * pixelArea * vigFactor

      denominator = denominator + spectrum(Ebin) * dE / expr
      numerator   = numerator   + spectrum(Ebin) * dE
    } # next Ebin

  } elsif (imageModel eq 'uniform') { # eqn 35, 36 of SSC-LUX-TN-0053
    foreach Ebin (startEbin..stopEbin) {
      expr = 0.0
      foreach pixel (listOfPixels) {
        call CAL_getFilterTransmission(filterTrans)

        totalQe = 0.0
        foreach patternID (0 .. 31) {
          qeOfPattern = CAL_getQuantumEfficiency(patternID etc)
          totalQe = totalQe + qeOfPattern
        } # next patternID

        call CAL_getVignettingFactor(vigFactor)
        expr = expr + filterTrans * totalQe  * vigFactor
      } # next pixel

      denominator = denominator + spectrum(Ebin) * dE / expr
      numerator   = numerator   + spectrum(Ebin) * dE
    } # next Ebin
    denominator = denominator * regionArea
  }

  end subroutine calcSpatial00
```

9 Comments

1. The correction for pixels masked by cosmic rays can only be approximate because some events are rejected as cosmic rays onboard the spacecraft. Full spatial information is not returned, but the number of pixels involved per frame is calculated and stored in the EXPOSUnn extensions of the calibrated event list product. At present **lccorr** bin-averages these numbers to the same bin times and size as the TS and includes them on this basis in the calculation of FRACEXP. Several alternative schemes are possible:

Scheme	Advantages	Disadvantages
Ignore	No added noise Simplest scheme	Wrong total flux No provision for slow drift in CR intensity
Bin to same fineness as TS	Correct total flux Correct for slow drift in CR intensity	Added noise
Average over whole TS	Correct total flux No added noise	No provision for slow drift in CR intensity
Bin somewhat coarser than TS	Optimized correction	Most complex scheme

The last scheme may be adopted in a future version of **lccorr**, or the choice of scheme may be made selectable by parameter.

2. No account is taken as yet of significant variations in spacecraft pointing or attitude. It is assumed that events occurring during unacceptably large attitude excursions have been rejected via GTIs set further 'upstream'.

References

[1] W. Pence L. Angelini and A.F. Tennant. The Proposed Timing FITS File Format for High Energy Astrophysics Data. Technical Report OGIP/93-003, NASA/GSFC, Nov 1993. Found at the URL: `http://legacy.gsfc.nasa.gov/docs/heasarc/ofwg/docs/summary/ogip_93_003_summary.html`.

[2] I.M. Stewart. Exposure Correction of XMM MOS and PN SAS Products. Technical Report SSC-LUX-TN-0053 Issue 1.0, SSC, February 2000.