binned_att

February 1, 2016

Abstract

binned_att is a library which contains the essential routines for constructing, writing and reading a binned-attitude time series.

version (from `VERSION`): 1.3.3

1 Description

X-ray or optical photons are detected as 'events' on the CCD cameras of XMM-Newton. Before one can do some science with these events, it is necessary to determine where in the sky each event comes from. This task is complicated by the fact the the attitude (ie, pointing direction and roll angle) of the XMM-Newton spacecraft does not stay exactly constant during the course of an exposure. Given a sufficiently detailed record of this wandering of the attitude it is of course possible to compute the sky origin of each individual event, but it is much more efficient in computational terms to approximate this notional continuous record by a discrete series of N attitude samples. Each sample is separated from its neighbours by time boundaries: it is sensible to choose the boundaries such that the attitude wander within a given pair of boundaries is restricted to within a preset acceptable range. Events falling between boundaries i and $i+1$ are therefore assumed to be associated with attitude $i+1$, and their sky position determined accordingly.

Matters become a little more complicated if it is desired to create maps in sky coordinates which derive from things other than the simple occurrence of events. For example, a map of the relative exposure of the camera at each point in the sky. For this one needs additionally to associate a weight with each attitude sample. One can therefore perform N transforms, one per attitude sample, of the exposure map from CCD coordinates to sky coordinates, weight them accordingly and then simply sum them together.

The present library therefore defines the concept of a series of N attitude samples, with optional weights associated, separated by $N-1$ time boundaries. The library provides routines for calculating the series of attitude samples from a supplied fine-detail record of the attitude wander, for calculating weights from GTIs, and also for reading and writing such an attitude series to or from a FITS table.

The attitude series is implemented in f90 as a variable type with the following structure:

```
type :: attBinType
  type(SpacecraftAttitudeType) ::& ! type defined in module caloaldefs
    att
  real(double) ::&
    binStartTime,&
    gtiDuration
  logical ::&
    isGood
end type attBinType
```

Some points:

- A series of attitude samples is represented within this library as a vector of type 'attBinType'. In this situation the value of 'binStartTime' of the first element of the vector is ignored by convention.
- The name 'gtiDuration' for the weight is just because of historical inertia and may be changed to the more general term at some later time.
- The field 'isGood' is included so as to be able to label bad attitudes independently of a GTI schema.

The FITS implementation of the series of N attitude samples is a little different, in that only elements 2 to N are stored in a table, the attitude and weight (not the start time, which is undefined) of the first element being stored in keywords of the table header. The details of the convention are as follows. The table should contain the following header keywords:

- (mandatory)(double) `RA_FIRST`, `DECFIRST` and `PA_FIRST`: respectively RA, dec and position angle (all in decimal degrees) of the first attitude in the series.
- (mandatory)(boolean) `FIRST_OK`: indicates whether the first attitude is 'good' or not.
- (optional)(double) `WGTFIRST`: the weight of the first attitude sample. If this keyword is present, the corresponding column in the table must also be present; if the keyword is not present, any `WEIGHTS` column is ignored and all weights are assumed to be 1.
- (mandatory)(double) `RA_PNT`, `DEC_PNT` and `PA_PNT`: average or nominal attitude, in decimal degrees.
- (mandatory)(string) `AVRG_PNT`: meaning of the '_PNT' values, eg mean or median.
- (mandatory)(double) `RA_DELTA`, `DEC_DELTA`, `PA_DELTA`: the attitude-wander limits discussed in section **??** of the documentation for task **attbin**. These are in decimal degrees.

The required structure of the table itself is best described by a chunk of code from the library:

```
time    => real64Data(addColumn(binnedAttTab, 'TSTART',  REAL64, 's'))
raDeg   => real64Data(addColumn(binnedAttTab, 'RA',      REAL64, 'deg'))
decDeg  => real64Data(addColumn(binnedAttTab, 'DEC',     REAL64, 'deg'))
paDeg   => real64Data(addColumn(binnedAttTab, 'PA',      REAL64, 'deg'))
isGood  => boolData(  addColumn(binnedAttTab, 'IS_GOOD', BOOLEAN))
weights => real64Data(addColumn(binnedAttTab, 'WEIGHTS', REAL64))
```

Of these, only the column WEIGHTS is optional (if the keyword WGTFIRST is not present in the table header). If it is not present, all weights are assumed to equal 1.

It should be noted that this binned-attitude 'time series' does NOT conform to OGIP recommendations. However in my view this is a deficiency in OGIP, which should be expanded to allow the edges of time bins to be recorded, not just the bin centre times.

1.1 binUpAttitudeFromOdf

The algorithm for constructing the series of attitude samples is to sample the attitude at a number of relatively finely-spaced times, and keep only those samples for which the difference from the last 'kept' sample exceeds preset limits. The present subroutine accepts a time sample and the prescribed limits from the caller, obtains (and stores) the spacecraft attitude at that time value, and returns this to the caller if its displacement from the last 'kept' sample exceeds the limits. The latter circumstance signals the start of a new attitude 'bin'. A new bin should also start if the sampled attitude is not 'good', or is again 'good' after a period of 'bad' attitudes.

```
subroutine binUpAttitudeFromOdf(time, maxDelta, attInBin, attIsGood&
, attIsNew)
  real(double),                 intent(in)  :: time
  type(SpacecraftAttitudeType), intent(in)  :: maxDelta
  type(SpacecraftAttitudeType), intent(out) :: attInBin
  logical,                      intent(out) :: attIsGood, attIsNew
end subroutine
```

1.2 sampleBinnedAttitude

```
subroutine sampleBinnedAttitude(binnedAtt, time, outAtt, attIsGood&
, attIsNew)
  type(attBinType),             intent(in)  :: binnedAtt(:)
  real(double),                 intent(in)  :: time
  type(SpacecraftAttitudeType), intent(out) :: outAtt
  logical,                      intent(out) :: attIsGood, attIsNew
end subroutine
```

1.3 writeBinnedAttitude

```
subroutine writeBinnedAttitude(binnedAttSetName, binnedAttTabName&
, binnedAtt, pntDeg, maxDelta, pntStyle, writeWeightColumn)
  character(*),     intent(in)           :: binnedAttSetName,&
                                            binnedAttTabName
  type(attBinType), intent(in)           :: binnedAtt(:)
  type(SpacecraftAttitudeType),&
                    intent(in)           :: pntDeg, maxDelta
  character(*),     intent(in)           :: pntStyle
  logical,          intent(in), optional :: writeWeightColumn
end subroutine
```

1.4 readBinnedAttitude

```
subroutine readBinnedAttitude(binnedAttSetName, binnedAttTabName&
, binnedAtt, noWeightColumn)
  character(*),     intent(in)            :: binnedAttSetName,&
                                             binnedAttTabName
  type(attBinType), pointer               :: binnedAtt(:)
  logical,          intent(out), optional :: noWeightColumn
end subroutine
```

1.5 gtiWeightBinnedAttitude

This calculates weights from the amount of GTI within each attitude bin.

```
subroutine gtiWeightBinnedAttitude(gtis, binnedAtt)
  type(IntervalT),  intent(in)    :: gtis(:)
  type(attBinType), intent(inout) :: binnedAtt(:)
end subroutine
```

References