

GammaLib - Action #1362

Implement run-wise energy thresholds for stacked analysis

11/11/2014 11:19 AM - Lu Chia-Chun

Status:	Closed	Start date:	11/11/2014
Priority:	Normal	Due date:	
Assigned To:		% Done:	100%
Category:		Estimated time:	0.00 hour
Target version:	1.1.0		
Description While energy thresholds of each run are specified and used in ctselect. ctexpcube should skip bins outside thresholds when adding the exposure. This functionality should go to fill() in GCTAExposure. Nevertheless, I am not sure what the best approach is to do this. One has to make the emin of each run consistent with what is used in ctselect (either given by users or by headers). I would suggest option1: ctselect adds emin and emax used for selection in the output xml file. option2: ctselect adds a new hdu in the output fits file to store the energies used for selection. (EOI = Energy of Interest?) I personally prefer option 1. What do you think?			
Related issues: Related to GammaLib - Change request # 1789: Add weighting array to CTA count... Closed 06/11/2016			

History

#1 - 11/11/2014 12:00 PM - Mayer Michael

- Description updated

I also agree that this should be handled in GCTAExposure. The RoI is already checked for but not the energy boundaries. A mixture of Option1 and Option2 you describe is already there. Currently, the data selections are handled via the FITS header keyword DSVAL. ctselect was recently tested to support different energy thresholds and multiple energy selections (see #1039, will be in devel soon).

To make use of this feature, we should add the following to line 353 of GCTAExposure.cpp:

```
GEbounds obs_ebounds = cta->ebounds();
```

And then, later on in line 376, an if-statement should be added to check if the cube energy is contained in the given GEbounds.

We probably need to think about how to handle a bin in which the threshold energy is contained. Drop it, take it, or add a weight to the effective area, depending on the threshold energy with respect to the bin edges.

#2 - 11/11/2014 01:14 PM - Lu Chia-Chun

Yes, I noticed issue 1039. (This is why I invited you to watch this issue.)
About the bins which contain the energy thresholds, I don't have a really good approach.

My suggestion is

- In the exporter, use the irf energy boundaries which are closest to the energy threshold to represent the energy thresholds.
- Force to generate exposure cube with the same energy boundaries as those of the original irfs.
- Exposure cube files are larger than irf files, but usually we don't generate many cubes as the purpose of ctexpcube is to merge runs. So it shouldn't be a big problem to have fine bins as the original irfs.

#3 - 11/11/2014 01:38 PM - Lu Chia-Chun

Mayer Michael wrote:

I also agree that this should be handled in GCTAExposure. The Rol is already checked for but not the energy boundaries. A mixture of Option1 and Option2 you describe is already there. Currently, the data selections are handled via the FITS header keyword DSVAL. ctselect was recently tested to support different energy thresholds and multiple energy selections (see #1039, will be in devel soon).

I didn't realize that the code on the ctool side has been committed to 1039-allow-energy-thresholds-in-ctselect. Now I'll have a look

#4 - 11/11/2014 01:39 PM - Mayer Michael

Lu Chia-Chun wrote:

Yes, I noticed issue 1039. (This is why I invited you to watch this issue.)
About the bins which contain the energy thresholds, I don't have a really good approach.
My suggestion is

- In the exporter, use the irf energy boundaries which are closest to the energy threshold to represent the energy thresholds.

In my view, the exposure cube should not depend on the binning of the IRF. Anyway, gammalib is interpolating between the bins. In addition, a user is able to provide own thresholds, (e.g. increasing every energy threshold of a run by say 10%). This should be recognised by gammalib and be independent from the IRF

- Force to generate exposure cube with the same energy boundaries as those of the original irfs.

For a binned analysis, you generally define a binning which is the same for events (ctbin), background (ctbkgcube), psf (ctpsfcube) and exposure (ctexpcube). The binning definition should, in my view, be up to user and not be predefined by the IRF.

- Exposure cube files are larger than irf files, but usually we don't generate many cubes as the purpose of ctexpcube is to merge runs. So it shouldn't be a big problem to have fine bins as the original irfs.

I would definitely vote for a common way to deal with bins which do not contain the full energy range. I just noticed that ctbkgcube and ctpsfcube are also missing this functionality. To do it properly, we would for sure need some kind of weighting to fill the respective cubes. The easiest way I can think of is to not fill these bins at all (we could only fill if a bin is fully contained in the provided energy range of the run). This would imply quite some loss in statistics, since ctbin would have to drop these events, too. Therefore, the exposure, psf and background should be adjusted accordingly to match the event cube produced with ctbin.

#5 - 11/11/2014 02:46 PM - Lu Chia-Chun

You reminded me of one thing: Now `ctexpcube` doesn't do "integration over energy", which gets important when we drop events according to different `Eths`. Now `ctexpcube` only does sampling -- it works exactly as `GCTAAeff2D` at the moment. The only difference is that `livetime` is used to weight the run-wise `Aeff`.

I think there are two steps to get things right:

1. Use fine bins to sample the `irfs`.
2. Merge fine bins into rough bins to match the binning of `ctbin`. The new content would be the weighted (in `livetime`) averaged (in energy) exposure.

The binning of step 1 should be internally decided using the binning of the original `irfs`, and the binning of step 2, which gives the final product, should match `ctbin`

#6 - 11/11/2014 02:59 PM - Mayer Michael

I think there are two steps to get things right:

1. Use fine bins to sample the `irfs`.
2. Merge fine bins into rough bins to match the binning of `ctbin`. The new content would be the weighted (in `livetime`) averaged (in energy) exposure.

Now I see the point of the fine binning. You mean the fine binning for the integration, right? We could use a simple `GFunction` for that. Since `gammalib` interpolates between the `IRF` bins, I guess we could use the available `GIntegral`-class to do that.

#7 - 11/11/2014 03:27 PM - Lu Chia-Chun

Mayer Michael wrote:

Now I see the point of the fine binning. You mean the fine binning for the integration, right? We could use a simple `GFunction` for that. Since `gammalib` interpolates between the `IRF` bins, I guess we could use the available `GIntegral`-class to do that.

I am not familiar with these two classes. Could you offer an example? I believe it's more efficient to use them to calculate the averaged exposure than filling a fine-bin cube and using it as a function.

#8 - 11/11/2014 03:39 PM - Mayer Michael

You can check out the `GCTAResponse_helper.cpp` files. Lots of functions get integrated there.

In general (as far I understand it), it works the following:

```
//Define GMyFunction class
class GMyFunction: public GFunction {
public:
    GMyFunction();
    double eval(double x);
protected:
    // protected members
    ...
};
```

And then in the source code, one could integrate the function by:

```
GMyFunction integrand;
GIntegral integral(&integrand);
double integral_value = integral.romberg(xmin,xmax);
```

#9 - 11/12/2014 12:12 AM - Knödlseider Jürgen

Mayer Michael wrote:

I would definitely vote for a common way to deal with bins which do not contain the full energy range. I just noticed that `ctbkgcube` and `ctpsfcube` are also missing this functionality. To do it properly, we would for sure need some kind of weighting to fill the respective cubes. The easiest way I can think of is to not fill these bins at all (we could only fill if a bin is fully contained in the provided energy range of the run). This would imply quite some loss in statistics, since `ctbin` would have to drop these events, too. Therefore, the exposure, `psf` and background should be adjusted accordingly to match the event cube produced with `ctbin`.

I fully support this idea, just drop the entire bin as you do not know what happens inside the bin.

Indeed, we need a coherent support of the energy selection in:

- `ctbin`
- `ctexpcube`
- `ctpsfcube`
- `ctbkgcube`
- `ctmodel`

#10 - 11/12/2014 12:16 AM - Knödlseider Jürgen

Lu Chia-Chun wrote:

You reminded me of one thing: Now `ctexpcube` doesn't do "integration over energy", which gets important when we drop events according to different `Eths`. Now `ctexpcube` only does sampling -- it works exactly as `GCTAAeff2D` at the moment. The only difference is that `livetime` is used to weight the run-wise `Aeff`.

I think there are two steps to get things right:

1. Use fine bins to sample the `irfs`.
2. Merge fine bins into rough bins to match the binning of `ctbin`. The new content would be the weighted (in `livetime`) averaged (in energy) exposure.

The binning of step 1 should be internally decided using the binning of the original `irfs`, and the binning of step 2, which gives the final product, should match `ctbin`.

I'm not sure that I understand why you need different binnings. I think an event cube should be always sufficiently fine binned so that the IRF gets sampled correctly. This is under user control. What would be a use case where you have a coarse energy binning for the cube but a fine binning for the IRF? Would you then also have the same fine binning for PSF or background cube?

#11 - 11/12/2014 12:16 AM - Knödlseider Jürgen

- Target version set to 1.0.0

#12 - 11/12/2014 12:17 AM - Knödlseider Jürgen

- Subject changed from *Adjust GCTAExposure* to *accomodate usage of run-wise energy thresholds* to *Implement run-wise energy thresholds for stacked analysis*

#13 - 11/12/2014 04:49 PM - Mayer Michael

I fully support this idea, just drop the entire bin as you do not know what happens inside the bin.

I agree that this is easiest way. However, we might still think of a way to save these events.

Just as an example:

If the user chooses a rather coarse binning with one energy bin between 500-800 GeV. By accident, many runs may have a threshold at 550 GeV (and the user does not care/has no idea about energy thresholds). Accordingly, we would throw away lots of events. Maybe we find a way to somehow weight the corresponding exposure depending on the location of the energy threshold with respect to the bin boundaries.

This weighting could then be implemented in the other tools as well.

#14 - 11/13/2014 10:50 AM - Lu Chia-Chun

Let me take an example:

Run Eth (TeV)

```
1 0.2
2 0.21
3 0.25
4 0.3
```

Binned analyses usually have 0.1-0.2 TeV bin size. When we combine these four runs, we have two approaches:

1. Drop all events below 0.3 TeV. Make the first energy bin have a lower boundary of 0.3 TeV.
2. Drop events according to the run-wise Eth. Make the first energy bin have a lower boundary of 0.2 TeV and a upper boundary of 0.4 TeV.

If we take option 2, it's a bit tricky to get the irf for the bin of 0.2-0.4 right. We have to calculate the run-wise 'average exposure' in this energy band. An easy way to calculate it is probably

```
for ( Ei = 0.2 ; Ei < 0.4 ; Ei = Ei + dE){
sumAeff = sumAeff + Aeff(Ei)*dE;
}
```

$\text{avgAeff} = \text{sumAeff} / (0.4 - 0.2)$

#15 - 11/13/2014 11:01 AM - Mayer Michael

That sounds like a reasonable approach to me. However, I would not change the binning of the cube according to the thresholds. It may confuse the user if the binning is different than she/he introduced to begin with. If the user has a binning of 0.1-0.4 TeV, it should be filled with the correct weighted exposure afterwards. In an extreme case, if the user creates the first bin to be from 0.1-0.2TeV, the map should be empty.

#16 - 11/13/2014 11:37 AM - Lu Chia-Chun

Mayer Michael wrote:

That sounds like a reasonable approach to me. However, I would not change the binning of the cube according to the thresholds. It may confuse the user if the binning is different than she/he introduced to begin with. If the user has a binning of 0.1-0.4 TeV, it should be filled with the correct weighted exposure afterwards. In an extreme case, if the user creates the first bin to be from 0.1-0.2TeV, the map should be empty.

Sure. What I took is just an example. (I didn't intend to fix the first bin to be 0.2-0.4.)

#17 - 11/13/2014 01:45 PM - Mayer Michael

I see, sorry for the misunderstanding.
Of course, a similar logic has to apply then for ctbkgcube, ctpsfcube and ctmodel.
I just checked the source code:

- ctbkgcube: we do sth like:

```
sum += model * bin->size();
```

We could add an additional scaling factor, e.g. containment fraction of how much the model contributes to that bin. Maybe we just add such a function to GEbounds:

```
double overlap(int& index, GEbounds& eventlist_ebounds);
```

This function should return values between 0 and 1. Do I think too simple here?

- ctpsfcube: Could probably stay as it is? Or do we need to add weights here, too?
- ctmodel: Currently bins which are not fully contained in the energy range are thrown away. We might have to change that, such that these bins are considered and their contents are weighted as for the exposure or bgcube tools.

What do you think?

#18 - 11/19/2014 11:40 AM - Knödlseider Jürgen

I think that all tools would need the weighting so that they are compatible. That's why I had initially proposed the "throw away" option as it is simpler and avoids any boundary problems. When you perform a binned analysis, you need a reasonable number of energy bins per decade (say 10) to follow the response evolution. This is particularly true at the lowest energies, where A_{eff} changes rapidly with energy.

We can of course think about integration schemes, but the problem there is that you always have to make an assumption on how things evolve with energy throughout the bin.

Note that we can specify energy bins in a file in all tools, and one could think about a little script that scans the XML file to provide a reasonably well adapted binning for the actual threshold encountered (i.e. sufficient fine sampling at lower energies, normal logarithmic sampling above the highest threshold).

#19 - 11/27/2014 02:12 PM - Mayer Michael

I also think it would be nice to have such a script which provides the 'recommended' energy binning. However, since the energy binning should be up to the user in the end, I would still vote for a proper consideration of the bins which are only partly filled by certain observations.
(Or do we want to take this user freedom away? Just add a ctool called ctmkbins to provide a proper binning definition, using accuracy parameters provided by the user. This output file would have to be used throughout the binned analysis then).

If we want to go for the proper consideration, we might need to figure a good and efficient way to finally deal with this. But I think, we could start with

just integrating the required value (e.g. exposure) and subsequent division by the bin width. I hope I don't think too simplistic here.

#20 - 12/08/2014 12:20 PM - Knödlseider Jürgen

Ok, if you'd prefer the proper handling then we should implement that.

Before going to the implementation we should make sure that we agree on the mathematics of the implementation so that we are all aware of the underlying assumptions. And then we should also make sure that the implementation is extensively tested (using e.g. pull distributions).

#21 - 03/12/2015 01:49 PM - Mayer Michael

Coming back to this:

A correct implementation to determine e.g. the exposure in an energy bin would be to use the integral divided by the binwidth:

$$\{\{ \text{latex}(\text{Aeff}(\text{bin}_i) = \frac{\int_{E_{\min}}^{E_{\max}} dE \cdot \text{Aeff}(E)}{\Delta E_i} \}\},$$

while $\{\{ \text{latex}(\Delta E_i) \}\}$ depicts the widths of the energy bin_i. The integration boundaries Emin and Emax are determined from the observation and the respective energy bin.

We could implement this with a rather coarse integration precision to not sacrifice too much computational speed.

An alternative would be to use the logarithmic mean of Emin and Emax and just evaluate Aeff there.

What do you think?

#22 - 06/30/2015 10:39 AM - Knödlseider Jürgen

For release 1.0, keep as is and make sure that the current behaviour is well documented, and provide information on how one could use user defined binning.

One could create a cscript that provides an appropriate binning for a given set of observations.

#23 - 07/01/2015 02:59 PM - Knödlseider Jürgen

- Status changed from New to Rejected

- Remaining (hours) set to 0.0

#24 - 06/21/2016 03:30 PM - Knödlseider Jürgen

- Related to Change request #1789: Add weighting array to CTA counts cube added

#25 - 06/21/2016 03:31 PM - Knödlseider Jürgen

- Status changed from Rejected to Closed

- Target version changed from 1.0.0 to 1.1.0

- % Done changed from 0 to 100

This issue has been fixed by #1789.