ctools - Bug #1791

ctbin results for in-memory / cmd-line workflow

06/16/2016 09:27 AM - Ziegler Alexander

Status:	Closed	**Start date:**	
Priority:	High	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.1.0		

Description

I have written an in-memory pipeline (following the given examples in the ctools example folder) and observe the following feature:
Making observation selection with ctselect and passing them in-memory directly to ctbin gives slightly different results for the counts-cube than writing the selected observations out (which I am doing simultaniously when running the script) and running ctbin with the same parameters afterwards 'by hand' (via the cmd-line).
The differences are in my example only in the lowest energy bin.
To test this further, I changed the value of const GEnergy g_energy_margin in $CTOOLS/src/ctbin/ctbin.cpp (line 44) to 1.0e-6 with the result that the resulting counts cube is now exactly the same for in memory creation and running ctbin via cmd line afterwards on the out-written selected observations.
However the resulting cube is now not compatible with the ones before (neither in memory nor cmd-line based output for 1.0e-12), the differences again only in a single bin (e.g. compared to previous results of cmd-line based approach not the lowest one, but still a low energy range bin, meaning possible threshold range of the observations).
I think with the changed (relatively large) value 1.0e-6 the difference to the result before might be due to the fact that the bounds are only tested in one direction (not +/-, either + or - in $CTOOLS/src/ctbin/ctbin.cpp lines 539/540)?
I think the overall behavior should be changed in some way to get consistent results from in-memory based analysis and cmd-line based workflow - if not the analysis will give slightly different results at the end with the exactly same input and its not obvious for the user what causes the differences.

Related issues:

Related to GammaLib - Change request # 1789: Add weighting array to CTA count...	**Closed**	**06/11/2016**

History

#1 - 06/17/2016 12:29 PM - Knödlseder Jürgen

- Tracker changed from Feature to Bug

- Subject changed from ctbin results for in-memory/ cmd-line workflow to ctbin results for in-memory / cmd-line workflow

- Assigned To set to Knödlseder Jürgen

- Priority changed from Normal to High

- Target version set to 1.1.0

Thanks for reporting. Could you most some script or the input parameters you use so that I can reproduce the problem and look into that?

#2 - 06/17/2016 03:59 PM - Ziegler Alexander

I just tried to create a short example script which could be used by everyone (without the need for specific data) - I observed this using HESS data. However, I was (within my first try) not able to reproduce this feature with data simulated from scratch, using caldb = "prod2", irf= "South_0.5h" for the simulation settings. I realized that there is no save energy range defined for the simulated observations, which I think makes it impossible to observe this feature on the data generated with the simulation? However, the most important settings running ctbin I use and can observe the described feature are:

```
bin["ebinalg"]  = "LOG"
bin["emin"]     = 0.501187233
bin["emax"]     = 79.43282347
bin["enumbins"] = 22
bin["nxpix"]    = 100
bin["nypix"]    = 100
bin["binsz"]    = 0.02
bin["coordsys"] = "CEL"
bin["proj"]     = "CAR"
```

bin is an instance of ctbin of course. Here, the energy bounds are chosen in such a way to get 10 bins/decade, in an arbitrary range. Of course, ctselect which was run before covered a broader range in energy.

#3 - 06/17/2016 08:25 PM - Knödlseder Jürgen

Do you know what the save energy threshold is handled in HESS data? Are you using header keywords in the IRFs?

There could be an issue with the precision of energy values written into the FITS file, which may be smaller than 1e-12. When energy values are read back from FITS their value differs slightly from what the values were before in memory.

It would be helpful if you could post a full example that produces the problem, and also the exact manifestation of the problem (e.g. what exactly you observe). In that way I can reproduce it here and try to better understand what's going on.

#4 - 06/18/2016 09:42 AM - Mayer Michael

> Do you know what the save energy threshold is handled in HESS data? Are you using header keywords in the IRFs?

Yes we are using the header keywords HI_TRHES and LO_THRES which are stored in the effective area files. This energy range is applied specifically to each run using the hidden parameter usethres=DEFAULT in ctselect.
I think ctselect writes the DSS keywords into the events FITS header with a lower precision and thus we run into trouble when trying to directly compare these values to the original ones.

Alexander, can you e.g. reproduce this problem also with one example Crab run (e.g. 23523)? With this run there shouldn't be a problem sharing it.

#5 - 06/18/2016 09:45 AM - Knödlseder Jürgen

Thanks for clarifying. If the H.E.S.S. run can not be shared we could also use an IRF where we put the header keywords in.

#6 - 06/18/2016 10:08 AM - Mayer Michael

Yes I think a unit test with a replica of an IACT event list in the current format would be great.

#7 - 06/20/2016 01:05 PM - Ziegler Alexander

> I think ctselect writes the DSS keywords into the events FITS header with a lower precision and thus we run into trouble when trying to directly compare these values to the original ones

I think thats it- I just used only one run- the 23523 which Michael suggested- and looked at the precision. The original values are stored with a precision of 1e-15, whereas the values refilled in the fits-header from ctselect is only written out with 1e-6/1e-4. It seems like both values are stored in only one variable which is called DSVAL2 (the values for LO_THRES and HI_THRES), separated by semi-colon, if I get everything correctly (somehow, the LO_THRES value has a higher precision).

What I did then was setting the lower energy border of ctbin to the low-precision value written out by ctselect (1e-6) and the result is now that the in memory pipeline did not fill the lowest energy bin, as the exact low-threshold energy is slightly larger than that value (just because the precision is higher, when reading the original value in, 1e-15), whereas via cmd-line, the low energy bin is filled as it uses the value which was written out by ctselect with a precision of 1e-6.

So when having energy bounds for ctbin in the threshold range of the runs, this can create the observed problems. I think when having a large runlist,

its not too unlikely that some runs hit the defined values. I think now it should be easy to reproduce the feature, one just needs an irf file providing HI/LO_THRES values with a high precision.

And, I mean this all is clearly connected to the variable GEnergy g_energy_margin as described above, which might change the behavior again, depending on its value, testing only in +/- direction and so on...

Maybe Michael can shortly cross-check by doing the same for one run- setting the low energy bound in ctbin to the value written out by ctselect for LO_THRES.

#8 - 06/20/2016 02:34 PM - Mayer Michael

I quickly had a look at the FITS files from PA and HAP. The difference is that HAP stores the LO/HI_THRES values with extreme precision. The PA values are between 4 and 6 digits after the comma. Therefore I might not see this issue in PA data. Since the energy threshold is defined arbitrary anyway, I would recommend we limit ourselves to maximum two or three digits after the comma.

Nevertheless, ctools should be able to cope with different precisions. The question is how can we achieve that?
As Alexander already mentioned, I guess we could increase the g_energy_margin to 1e-6 or even bigger and allow a positive and negative tolerance. Are there any pitfalls with that approach?

#9 - 06/21/2016 11:09 AM - Knödlseder Jürgen

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

#10 - 06/21/2016 02:06 PM - Knödlseder Jürgen

I have not yet looked into the problem, but I have implemented a rather fundamental change in the handling of counts cubes (see #1789) that may also have solved that problem. At least, there are no longer energy margins in ctbin. Have a look at #1789 and try the latest devel version of the code, and tell me if there is still a problem.

#11 - 06/23/2016 09:42 AM - Ziegler Alexander

I re-did the checks for the original analysis where the problem was observed first and for the particular case of the single run as described in #7. With the latest devel version, there are no differences in the count cubes anymore - now I get exactly the same result for in-memory and cmd-line based approach (which is slightly different to results obtained before updating the software, ~5% total event number difference in the cube, but I think thats also expected after that fundamental changes).

#12 - 06/23/2016 09:49 AM - Knödlseder Jürgen

- Status changed from New to Closed

- % Done changed from 0 to 100

Good, thanks for having done the checks.

The differences in event numbers may be explained by the new way of handling the energy threshold. Before, entire energy bins have been excluded, while now the overlap between valid energy range and cube bins is computed. Since there are many events at low energies, a 5% difference seems plausible.

I close the issue now.