

ctools - Feature #1330

ctmodel does not support multiple observations

10/08/2014 04:43 PM - Mayer Michael

|   |                  |                        |                   |
|---|------------------|------------------------|-------------------|
| <b>Status:</b>  | Closed           | <b>Start date:</b>     | 10/08/2014        |
| <b>Priority:</b>  | High             | <b>Due date:</b>       |                   |
| <b>Assigned To:</b>   | Knödseder Jürgen | <b>% Done:</b>         | 100%              |
| <b>Category:</b>  |                  | <b>Estimated time:</b> | 0.00 hour         |
| <b>Target version:</b>  | 00-08-00         |                        |                   |
| <b>Description</b><br>When computing a model map, ctmodel requires exactly <b>one</b> binned observation in the container. This is problematic when the user wants to compute a model map based on multiple runs (each with its own response).<br>ctmodel should therefore be able use an Observation container as input. The problem occurs in the function ctmodel::setup_obs() (line 605). Where a GCTAEventCube is required to continue.<br>The cube however, could also be built by user parameters (such as 'xref', 'yref', 'nxpix', ...). A loop over the observations in ctmodel:model_map() would be necessary to fill the sky map in the end.<br>If you agree, I would try to implement this functionality that in ctmodel. |                  |                        |                   |
| <b>Related issues:</b><br>Related to ctools - Action # 1287: Modify ctmodel to always give one cube   |                  |                        |                   |
|   |                  | <b>Closed</b>          | <b>07/20/2014</b> |

History

#1 - 10/08/2014 05:13 PM - Knödseder Jürgen

The logic of ctbin and ctmodel was changed with the introduction of the stacked cube analysis (i.e. putting events from different runs in a single event cube). Now, ctbin and ctmodel only operate on single runs, and if one really wants to keep event cubes per run, one has to add an explicit loop on the run outside the tools. The according scripts need still to be written.

The rational behind this is that a binned analysis in the "old fashion" with one event cube per run is not something very efficient, be it in memory requirements or CPU requirements. If data should be analysed run-wise, unbinned analysis should be the default option. In contrast, if data are binned, one should use the stacked cube analysis that computes the average exposure and Psf for the stacked cube.

Still, the old style run-wise binned analysis is still possible by looping over ctbin or ctmodel.

#2 - 10/08/2014 05:32 PM - Mayer Michael

I fully agree that using one event cube for several runs makes sense for the stacked cube analysis. It is great that this way is supported in ctools. I just stumbled upon this issue when I wanted to make a residual map. I use unbinned analyses with several runs.

For the residual map, I first run ctbin to get a count map containing all my events, which worked fine. But ctbin does not change or averages my corresponding response, does it? In my opinion, it should be possible to compute a model map in the same way as ctbin. Since every run has a different response, I would first have to compute an average response which is not desirable if the user wants to use his/her observation container further on.

So I would propose to allow passing any kind of observation container to ctmodel, which is used to fill the map based on the user parameters. Via the command line it also should be possible to pass any kind of observation xml-file.

Of course, one could loop over ctbin and ctmodel. This is however not desirable for any user I guess.

#3 - 10/08/2014 05:51 PM - Knödseder Jürgen


I see your point. You want to get a model for an unbinned analysis. So we have to add a loop over all observations in the container. I guess the respective code in ctbin can be basically copied/pasted for that purpose.

**#4 - 10/08/2014 05:58 PM - Mayer Michael**

- Assigned To set to Mayer Michael

Exactly. I guess, generally in all ctools, we should support both, binned and unbinned analyses. So you agree if we implement that functionality in ctmodel?

I guess the respective code in ctbin can be basically copied/pasted for that purpose.

Most of it, yes. But I guess we might need a lot of if-statements in the functions `get_parameters()` and `setup_obs()`. I hope, I won't break the current functionality when implementing this 

**#5 - 10/08/2014 09:17 PM - Knödseder Jürgen**

I was looking into the ctmodel code and recognized that it still has the same logic that was formerly in ctbin, which is that one model cube is created per observation found in an XML file. This logic has been removed from ctbin when we implemented the stacked cube analysis, which in fact simplified pretty much the code of ctbin. ctmodel needs now a similar modification. If you feel easy with doing that you may just go ahead, otherwise I can also find a little time to implement that change.

**#6 - 10/09/2014 03:26 PM - Mayer Michael**

Yes. Currently, ctbin and ctmodel are not fitting together.

I looked into the code and I am not sure how to implement the same functionality as in ctbin. Do we want ctmodel to support only one binned observation per container.

As far as I can judge, ctmodel must be mostly rewritten to support both binned and unbinned observations.

I could give it a try tomorrow.

**#7 - 10/09/2014 04:52 PM - Knödseder Jürgen**

I'll see whether I can find some time this evening to look into that. I keep you informed.

**#8 - 10/09/2014 11:49 PM - Knödseder Jürgen**

- Status changed from New to Feedback

- Assigned To changed from Mayer Michael to Knödseder Jürgen

- Target version set to 00-08-00

- % Done changed from 0 to 80

I changed the code to make ctmodel equivalent to ctbin. You may check the branch 1330-ctmodel-multiple-handling. Please see the NEWS file for some information. A new parameter `obsfile` has been introduced that allows specifying an observation definition XML file and ctmodel will loop over all observations and build a combined model (I have actually not tested this specific functionality, but checked that the old stuff and basically the unit tests all pass).

So please go ahead and do some tests with that new version and provide me with some feedback.

**#9 - 10/10/2014 10:29 AM - Mayer Michael**

Excellent, I checkout the new branch and tested it with the 4 1DC runs from HESS. The logic is very straight forward and the results look fine.

I encountered just a problem which occurs when using runs with different energy thresholds.  
An example:

1. Loading 4 runs with different energy threshold
2. cutting each run with ctselect according to its respective energy threshold
3. create a count map using ctbin between 0.1-100 TeV.
4. create a model map with ctmodel using the count map as infile.

ctmodel now computes expected counts between 0.1 and 100 TeV and does not take into account the energy cuts of the respective observation.  
Subtracting the model map from the count map accordingly results many negative bins.  
In my case, adding the following line in ctmodel.cpp.763:

```
m_cube.ebounds(obs->events()->ebounds());
```

fixed this issue. However, I don't know if adding this line breaks the other functionality using a binned observation as input.  
Any concerns?

**#10 - 10/10/2014 10:52 AM - Knödlseider Jürgen**

I think this is not the right solution. The ebounds structure defines the number of energy bins and the boundaries of the cube itself, if you have an event list, you will only get the minimum and maximum energy and a single energy bin, while physically the cube still has several bins. I'm in fact surprised that this works for you, but maybe your cube had just a single energy bin anyways?

I have added a check to fill\_cube so that only bins are considered that fall within the energy boundaries for a single observation. This needed also some tweaking of get\_parameters for the case that no observation is defined (since then ctmodel sets up its own observation from the task parameters).


Can you give the new version a try?

**#11 - 10/10/2014 10:54 AM - Knödlseider Jürgen**

Btw: ctmodel does not integrate in energy, it just evaluates the model at the bin centre. You thus have to make sure that the cube is sufficiently well binned in energy to get the correct solution. We should add some function to obsutils that computes then the residual by summing e.g. over the energy axis.

**#12 - 10/10/2014 11:28 AM - Mayer Michael**

I'm in fact surprised that this works for you, but maybe your cube had just a single energy bin anyways?

Yes, I just had one energy bin, since I was interested in a residual map. I guess this was the **only** reason that it worked out 

Can you give the new version a try?

Yes, the new version takes the different energy thresholds perfectly into account.

Btw: ctmodel does not integrate in energy, it just evaluates the model at the bin centre. You thus have to make sure that the cube is sufficiently well binned in energy to get the correct solution. We should add some function to obsutils that computes then the residual by summing e.g. over the energy axis.

Ok, I guess for some reason, I was missing this information completely. Of course, then it makes sense to produce a fine-binned count and model map and sum the content over energy axis. Instead of using obsutils, we could also just add a method to GSkymap? Something like GSkymap::project2D(), or GSkymap::project\_z().

For the creation of residual maps which does the job in the end, would it make sense to have a simple csscript for that purpose?

**#13 - 10/10/2014 12:11 PM - Knödlseider Jürgen**

- % Done changed from 80 to 90

Mayer Michael wrote:

Btw: ctmodel does not integrate in energy, it just evaluates the model at the bin centre. You thus have to make sure that the cube is sufficiently well binned in energy to get the correct solution. We should add some function to obsutils that computes then the residual by summing e.g. over the energy axis.

Ok, I guess for some reason, I was missing this information completely. Of course, then it makes sense to produce a fine-binned count and model map and sum the content over energy axis. Instead of using obsutils, we could also just add a method to GSkymap? Something like GSkymap::project2D(), or GSkymap::project\_z().

We can add such a method. I would propose however not to use project\_ as for a skymap this refers more to the projection of the pixels on the sky, and we have already a projection method. How about stack, stack\_maps, collapse, or collapse\_maps. Maybe this method is sufficiently specialized

that a very explicit method name such as `stack_maps` is justified?

For the creation of residual maps which does the job in the end, would it make sense to have a simple `csscript` for that purpose?

Indeed, a `csscript` would be probably appropriate.

**#14 - 10/10/2014 11:16 PM - Knödlseider Jürgen**

- *Status changed from Feedback to Closed*

- *% Done changed from 90 to 100*

Merged into devel branch.