GammaLib - Change request #1371

GCTAResponseCube should hold GCTAModelCubeBackground member

11/27/2014 01:40 PM - Mayer Michael

Status:	Closed	**Start date:**	11/27/2014
Priority:	Normal	**Due date:**	
Assigned To:	Mayer Michael	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0		

Description

Currently, in stacked binned analysis, we use the GCTAReponseCube class to hold the detector response in the user defined binning. This response currently uses the GCTAExposure and GCTAMeanPsf classes. In contrast, for unbinned analysis where we use GCTAResponseIrf, the background is an additional part of the response.
I therefore suggest, to make GCTAModelCubeBackground a protected member of GCTAResponseCube. This would make the usage of binned and unbinned analysis more homogenous.
In the XML model file, the user would just have to specify a spectral model (analogous to GCTAResponseIrf):

```
<source name="CTABackgroundModel" type="CTACubeBackground" instrument="CTA">
  <spectrum type="ConstantValue">
    <parameter name="Value" scale="1.0" min="0.01" max="1000.0" value="1.0" free="1"/>
  </spectrum>
</source>
```

History

#1 - 02/08/2015 08:28 PM - Knödlseder Jürgen

I won't make GCTAModelCubeBackground a member of GCTAReponseCube, as GCTAModelCubeBackground is really living in the model space. But I see your point of adding the stacked background model directly to GCTAReponseCube, similar to GCTABackground which is part of GCTAReponseIrf.

I would then suggest to create a new class GCTACubeBackground that would be analogous to GCTABackground. GCTAModelCubeBackground should then be modified similar to GCTAModelIrfBackground so that the background stored in GCTACubeBackground could be used.

This would imply, however, that one cannot use a map cube any longer in the model XML file to provide the background. The background would then be supplied as a response component (which I admit is more logical).

If we agree that this should be done then I would target version 1.0 for that to get this straight before the first release.

#2 - 02/15/2015 06:35 PM - Mayer Michael

That sounds like a good plan to me. I think having the background stored in the response seems more logical and also more aligned with the unbinned analysis.

#3 - 03/03/2015 10:31 AM - Mayer Michael

Just to make sure I understand correctly before I start with this, here are the action items:
On gammalib level:

- GCTAResponseCube needs to have a protected member GCTAEventCube m_bkgcube

- GCTAModelCubeBackground needs modification to be analogous to GCTAModelIrfBackground

On ctools level:

- ctbkgcube needs some adjustments to store the proper model (and assign the correct identifier to it)

Do I miss anything?

#4 - 03/09/2015 01:25 PM - Mayer Michael

Do you agree with the above items? I would then start to make the modifications.

#5 - 03/09/2015 01:49 PM - Knödlseder Jürgen

Mayer Michael wrote:

> Just to make sure I understand correctly before I start with this, here are the action items:
> On gammalib level:
>
> - GCTAResponseCube needs to have a protected member GCTAEventCube m_bkgcube

No, you need to create a new class GCTACubeBackground with an interface similar to GCTABackground. GCTAResponseCube would then have a protected member GCTACubeBackground m_background.

We then also need to add a constructor

GCTAResponseCube(const GCTACubeExposure& exposure, const GCTACubePsf& psf, const GCTACubeBackground& background);

and methods

const GCTACubeBackground& background(void) const;
void background(const GCTACubeBackground& background);

> - GCTAModelCubeBackground needs modification to be analogous to GCTAModelIrfBackground

Yes.

> On ctools level:
>
> - ctbkgcube needs some adjustments to store the proper model (and assign the correct identifier to it)
>
> Do I miss anything?

#6 - 03/09/2015 02:45 PM - Mayer Michael

Ok I see, this makes sense. To avoid confusion, we might think of renaming GCTABackground to GCTABackgroundIrf or GCTAIrfBackground.

#7 - 03/09/2015 02:55 PM - Knödlseder Jürgen

Mayer Michael wrote:

> Ok I see, this makes sense. To avoid confusion, we might think of renaming GCTABackground to GCTABackgroundIrf or GCTAIrfBackground.

I was thinking in fact about renaming all classical IRF classes to GCTAIrf... (#1269) but at the end I decided of not doing that as the class names would become pretty cumbersome (e.g. GCTAIrfPsfPerfTable). Instead I renamed all cube related stuff to GCTACube... to tie all cube-related classes together.

#8 - 03/09/2015 03:19 PM - Mayer Michael

Ok great. I wasn't aware of that issue. Fine with me, let's do it that way.

#9 - 03/10/2015 11:18 AM - Mayer Michael

- % Done changed from 0 to 30

I started to modify the code and got a bit stuck while implementing the method GCTAModelCubeBackground::mc():
Is it actually foreseen to conduct simulations from a binned observation?
Of course, we can simulate a GCTAEventList from the background cube, however, the GObservation we pass to this method would have to contain an unbinned event list (to read GEbounds and GCTARoI) and, at the same time, a GCTAResponseCube to retrieve the (new) background model. I actually cannot think of a scenario where this is given at the same time. I therefore propose to throw a "feature not implemented" error in GCTAModelCubeBackground::mc.

By curiosity, I tried ctobssim using a binned observation as input, which threw the following error:

```
*** ERROR encounterted in the execution of ctobssim. Run aborted ...
*** ERROR in GCTAEventCube::naxis(int): CTA event cube axis 33835690 is outside the valid range [0,2].
```

We might think about throwing a clearer exception explaining that binned observations should not be used as input for ctobssim?

#10 - 03/10/2015 01:52 PM - Knödlseder Jürgen

Mayer Michael wrote:

> I started to modify the code and got a bit stuck while implementing the method GCTAModelCubeBackground::mc():
> Is it actually foreseen to conduct simulations from a binned observation?

Not through the GCTAModelCubeBackground::mc() method which only works for event lists. The idea would be to have a ctools that creates a MC samples from a model (see #544, although we may use a different name, e.g. ctcubesim).

> Of course, we can simulate a GCTAEventList from the background cube, however, the GObservation we pass to this method would have to contain an unbinned event list (to read GEbounds and GCTARoI) and, at the same time, a GCTAResponseCube to retrieve the (new) background model. I actually cannot think of a scenario where this is given at the same time. I therefore propose to throw a "feature not implemented" error in GCTAModelCubeBackground::mc.

It actually throws an exception if no event list is found in the observations. Is this not working or do you need more?

> By curiosity, I tried ctobssim using a binned observation as input, which threw the following error:
> [...]
> We might think about throwing a clearer exception explaining that binned observations should not be used as input for ctobssim?

This should indeed not happen. Can you create a bug issue so that we can fix that?

#11 - 03/10/2015 01:52 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- Assigned To set to Mayer Michael

#12 - 03/11/2015 11:06 AM - Mayer Michael

- Status changed from In Progress to Pull request

- Target version set to 1.0.0

- % Done changed from 30 to 100

> Not through the GCTAModelCubeBackground::mc() method which only works for event lists. The idea would be to have a ctools that creates a MC samples from a model (see #544, although we may use a different name, e.g. ctcubesim).

Ok I see. GCTAModelCubeBackground::mc() now exists and throws an exception in case no GCTAEventList is provided. ctcubesim or ctbinsim would certainly be a useful tool.

> It actually throws an exception if no event list is found in the observations. Is this not working or do you need more?

Sure it works like that for me. I was just wondering what kind of observation could be passed which contains a GCTAEventList (unbinned) and a GCTAResponseCube (binned) at the same time. But this would than be handled by ctcubesim, I agree.

> This should indeed not happen. Can you create a bug issue so that we can fix that?

see #1439

The code for this change request is now available on branch *1371-GCTAResponseCube-holds-GCTACubeBackground*.
I did the following changes:

- Created GCTACubeBackground.
- The GCTACubeBackground::mc() method was basically copied over from GModelSpatialDiffuseCube.
- The interface and interpolation of GCTACubeBackground was inspired by GCTACubeExposure (since these classes now basically live on the same response level).
- I've added GCTACubeBackground::integral(const double& logE), returning the content of the cube in units of ph/s/MeV/sr. It is called by the GCTAModelCubeBackground::npred()-method.
- Modified GCTAModelCubeBackground analogous to GCTAModelIrfBackground.
- Added test cases for the GCTACubeBackground/GCTAModelCubeBackground constructors and XML I/O.
- Modified GCTAResponseCube to hold a protected member m_background.

On the ctools level some modifications where necessary, too (available on branch *1371-adapt-ctools-for-new-binned-interface*).

- changed several .par-files to now query for a background cube file in case of binned analysis
- slightly modified ctbkgcube to make the interface more homogenous to ctexpcube and ctpsfcube.
- Now the outmodel written by ctbkgcube also contains the previously given sky models. Accordingly, the output file can directly be used as input for ctlike in binned mode.
- Restructured the script test_ctools.sh to be conform with the new code.

#13 - 03/12/2015 11:42 PM - Knödlseder Jürgen

- Status changed from Pull request to Feedback

I looked into the GammaLib code, overall is fine.

Shouldn't GCTACubeBackground::operator() take a GInstDir argument, not a GSkyDir? We may also think about passing simply GCTAEventBin to that operator so that in principle (if the dimensions are identical) one could directly use the bin index (avoiding any interpolations). The tricky question however remains on how to tell GCTACubeBackground that the dimensions are identical. That could be maybe handled at the GCTAModelCubeBackground::eval level as here we have a GObservation object, hence we can access the dimensions of the event cube and compare to the dimensions of the background cube. One would not like to do this for every event, though.

Do we really need the GCTACubeBackground::mc method? I now understand in fact better your earlier question, and I would suggest to put simply a feature not implemented exception for this method. We then do not have to deal with MC cache and could remove the

GCTACubeBackground::set_mc_cone method.

Are you sure that GCTACubeBackground::integral should divide by the solid angle at the end? The method is used by GCTAModelCubeBackground::npred which wants a spatial integral, not something divided by sr.

Finally, I renamed BgCube to BkgCube to comply with naming conventions we have already in place.

I pushed the code back into the 1371-GCTAResponseCube-holds-GCTACubeBackground branch.

I put this issue back on feedback. Once the questions are settled I'll look also in the ctools code.

#14 - 03/13/2015 10:37 AM - Mayer Michael

Thanks for the feedback.

> I looked into the GammaLib code, overall is fine.
>
> Shouldn't GCTACubeBackground::operator() take a GInstDir argument, not a GSkyDir? We may also think about passing simply GCTAEventBin to that operator so that in principle (if the dimensions are identical) one could directly use the bin index (avoiding any interpolations). The tricky question however remains on how to tell GCTACubeBackground that the dimensions are identical. That could be maybe handled at the GCTAModelCubeBackground::eval level as here we have a GObservation object, hence we can access the dimensions of the event cube and compare to the dimensions of the background cube. One would not like to do this for every event, though.

I am actually not sure about this. I was trying to be aligned with GCTACubeExposure. GCTACubeBackground is also somewhat living in the sky reference frame. It gives the background rate at a certain sky position and energy (like the exposure). Using the current implementation, the user would (in principle) be able to use different binnings throughout the analysis. Since exposure, psf and background are IRFs, I would support interpolation to avoid jumps in their functions. If we want to change the operator() to take GCTAInstDir, we should probably do this homogenous throughout GCTACubeExposure and GCTACubePsf.

> Do we really need the GCTACubeBackground::mc method? I now understand in fact better your earlier question, and I would suggest to put simply a feature not implemented exception for this method. We then do not have to deal with MC cache and could remove the GCTACubeBackground::set_mc_cone method.

Yes, I fully agree. Let's put in feature_not_implemented and remove the MC caches. This simplifies the class significantly.

> Are you sure that GCTACubeBackground::integral should divide by the solid angle at the end? The method is used by GCTAModelCubeBackground::npred which wants a spatial integral, not something divided by sr.

Oh I see, you are probably right. I think I was expecting to to have 1/sr, which is clearly wrong. So the division should be removed I guess. Probably, we need to check for sanity of results again.

> Finally, I renamed BgCube to BkgCube to comply with naming conventions we have already in place.

Agreed.

> I pushed the code back into the 1371-GCTAResponseCube-holds-GCTACubeBackground branch.
>
> I put this issue back on feedback. Once the questions are settled I'll look also in the ctools code.

Thanks, the code looks good.
Concerning the ctools-code: it came to my mind that could add GCTACubeBackground::fill(GObservations& obs), which is then called from

ctbkgcube. This would align the code more with GCTACubeExposure and ctexpcube.

#15 - 03/24/2015 03:28 PM - Mayer Michael

Did you have time to have a look at the ctools-code yet?

#16 - 03/25/2015 07:12 PM - Knödlseder Jürgen

Mayer Michael wrote:

> Thanks for the feedback.
>
> > I looked into the GammaLib code, overall is fine.
> >
> > Shouldn't GCTACubeBackground::operator() take a GInstDir argument, not a GSkyDir? We may also think about passing simply GCTAEventBin to that operator so that in principle (if the dimensions are identical) one could directly use the bin index (avoiding any interpolations). The tricky question however remains on how to tell GCTACubeBackground that the dimensions are identical. That could be maybe handled at the GCTAModelCubeBackground::eval level as here we have a GObservation object, hence we can access the dimensions of the event cube and compare to the dimensions of the background cube. One would not like to do this for every event, though.
>
> I am actually not sure about this. I was trying to be aligned with GCTACubeExposure. GCTACubeBackground is also somewhat living in the sky reference frame. It gives the background rate at a certain sky position and energy (like the exposure). Using the current implementation, the user would (in principle) be able to use different binnings throughout the analysis. Since exposure, psf and background are IRFs, I would support interpolation to avoid jumps in their functions. If we want to change the operator() to take GCTAInstDir, we should probably do this homogenous throughout GCTACubeExposure and GCTACubePsf.

The difference is that exposure and PSF are given in true sky direction while background is given in reconstructed sky direction. True sky directions are implemented by GSkyDir, while reconstructued sky directions are implemented by GCTAInstDir. For CTA, both have attributes RA,DEC, but for a general instrument the nature of the coordinates can be very different.

So for having a clean interface we should use GCTAInstDir for the background model.

I agree that interpolations should also be done for the background.

> Concerning the ctools-code: it came to my mind that could add GCTACubeBackground::fill(GObservations& obs), which is then called from ctbkgcube. This would align the code more with GCTACubeExposure and ctexpcube.

This would certainly be useful.

#17 - 03/26/2015 05:03 PM - Mayer Michael

> The difference is that exposure and PSF are given in true sky direction while background is given in reconstructed sky direction. True sky directions are implemented by GSkyDir, while reconstructued sky directions are implemented by GCTAInstDir. For CTA, both have attributes RA,DEC, but for a general instrument the nature of the coordinates can be very different.

Now I fully see your point. I totally agree that hence we should use GCTAInstDir instead of GSkyDir. I had the impression that people often confuse the two (including me). Would it therefore be useful to rename GCTAInstDir into something like GCTARecoDir?

> This would certainly be useful.

I changed the code in ctbkgcube and implemented GCTACubeBackground::fill(GObservations& obs).

I also removed GCTACubeBackground::set_mc_cone() and added feature_not_implemented to GCTACubeBackground::mc and GCTAModelCubeBackground::mc.
The solid angle division is now also removed. I quickly checked with some Crab data. The results look sane.
I pushed the changes to the both branches, i.e. gammalib and ctools respectively. All unit tests run smoothly. We might be ready to merge?

#18 - 03/26/2015 05:09 PM - Knödlseder Jürgen

Mayer Michael wrote:

> > The difference is that exposure and PSF are given in true sky direction while background is given in reconstructed sky direction. True sky directions are implemented by GSkyDir, while reconstructued sky directions are implemented by GCTAInstDir. For CTA, both have attributes RA,DEC, but for a general instrument the nature of the coordinates can be very different.
>
> Now I fully see your point. I totally agree that hence we should use GCTAInstDir instead of GSkyDir. I had the impression that people often confuse the two (including me). Would it therefore be useful to rename GCTAInstDir into something like GCTARecoDir?

The base class is GInstDir, hence the convention is to add just the instrument specific abbreviation XXX using the naming scheme GXXXInstDir. For CTA I agree that reconstructed direction makes sense, but for a general instrument this is not necessarily the case, hence I would like to keep the GInstDir class name.

> > This would certainly be useful.
>
> I changed the code in ctbkgcube and implemented GCTACubeBackground::fill(GObservations& obs).

> I also removed GCTACubeBackground::set_mc_cone() and added feature_not_implemented to GCTACubeBackground::mc and GCTAModelCubeBackground::mc.
> The solid angle division is now also removed. I quickly checked with some Crab data. The results look sane.
> I pushed the changes to the both branches, i.e. gammalib and ctools respectively. All unit tests run smoothly. We might be ready to merge?

Thanks, I'll check the code.

#19 - 03/26/2015 05:12 PM - Mayer Michael

> The base class is GInstDir, hence the convention is to add just the instrument specific abbreviation XXX using the naming scheme GXXXInstDir. For CTA I agree that reconstructed direction makes sense, but for a general instrument this is not necessarily the case, hence I would like to keep the GInstDir class name.

I understand and agree.

> Thanks, I'll check the code.

Thanks.

#20 - 03/26/2015 05:23 PM - Knödlseder Jürgen

I'm about to check the code. As the base class does not require to have an mc method I would in fact simply propose to remove the method. Having a method that is not required and that we are not planning to implement is in fact rather disturbing.

About the integral method, I'm still not sure it's doing what is expected. I think we need to multiply by the solid angle, right?

#21 - 03/26/2015 05:48 PM - Mayer Michael

> I'm about to check the code. As the base class does not require to have an mc method I would in fact simply propose to remove the method. Having a method that is not required and that we are not planning to implement is in fact rather disturbing.

Sounds good to remove it.

> About the integral method, I'm still not sure it's doing what is expected. I think we need to multiply by the solid angle, right?

The result should be in units of 1/MeV/s/sr, right? So if we multiply by the solid angle, we get 1/MeV/s. But I am not sure I might be missing something here.

#22 - 03/28/2015 06:47 PM - Knödlseder Jürgen

- Status changed from Feedback to Closed

Finally I found the time to merge everything into devel.

I just did a minor change in ctbkgcube: I removed the section where the model ID is set in the output observation container as this would prevent the model from being used (ctbin sets the obs_id, which is an integer in the header file and has nothing to do with the id in the XML file).

#23 - 03/30/2015 11:04 AM - Mayer Michael

Excellent, thanks for merging.
I checkout out the new devel branch and performed some tests.
I discovered two minor issues (The first has been there all along):

1. When using ctbkgcube with an input count map, the map is subsequently set to 0.0. This function calls GSkymap::operator=(const double& value) which reads the following:

   ```
   GSkymap& GSkymap::operator=(const double& value)
   {
       // Loop over all pixels
       for (int i = 0; i < m_num_pixels; ++i) {
           m_pixels[i] = value;
       }

       // Return this object
       return *this;
   }
   ```

   This, however, only concerns the first map of a cube. To set all values of the sky map, we should loop over m_num_pixels * m_num_maps instead of m_num_pixels only. Otherwise, maps in other energy bands are not set to 0.0 and we still have unwanted content. So I would say this is a bug in this function.
2. In ctbkgcube::get_parameters, I made a mistake when using the incube parameter: I used m_background = GCTACubeBackground(incube), which also does not set all cube values to zero. Instead, we should use the following two lines (also analogous to ctexpcube or ctspfcube):

   ```
   // Load event cube from filename
   GCTAEventCube cube(incube);

   // Define background cube
   m_background = GCTACubeBackground(cube);
   ```

Could you make these two minor (but important) adjustments?

Another thing I noticed is that you changed the default spectral model from a GModelSpectralConst to a Power law. Wouldn't it be better to provide the most simple model and let the user add complexity if required?

#24 - 03/30/2015 11:15 AM - Knödlseder Jürgen

Mayer Michael wrote:

> Excellent, thanks for merging.
> I checkout out the new devel branch and performed some tests.
> I discovered two minor issues (The first has been there all along):
>
> 1. When using ctbkgcube with an input count map, the map is subsequently set to 0.0. This function calls GSkymap::operator=(const double& value) which reads the following: [...]
>   This, however, only concerns the first map of a cube. To set all values of the sky map, we should loop over m_num_pixels * m_num_maps instead of m_num_pixels only. Otherwise, maps in other energy bands are not set to 0.0 and we still have unwanted content. So I would say this is a bug in this function.
> 2. In ctbkgcube::get_parameters, I made a mistake when using the incube parameter: I used m_background = GCTACubeBackground(incube), which also does not set all cube values to zero. Instead, we should use the following two lines (also analogous to ctexpcube or ctspfcube): [...]
>
> Could you make these two minor (but important) adjustments?

I'm about doing that.

> Another thing I noticed is that you changed the default spectral model from a GModelSpectralConst to a Power law. Wouldn't it be better to provide the most simple model and let the user add complexity if required?

Indeed, I forgot to mention that. From Fermi-LAT the experience is that users typically want to use a power law for the modulation of the cube model. My philosophy would be to provide what typically would be needed, which avoids that users need to hand modify the XML file. Do you have any feeling from your HESS analysis what would be better?

#25 - 03/30/2015 11:22 AM - Mayer Michael

> Indeed, I forgot to mention that. From Fermi-LAT the experience is that users typically want to use a power law for the modulation of the cube model. My philosophy would be to provide what typically would be needed, which avoids that users need to hand modify the XML file. Do you have any feeling from your HESS analysis what would be better?

I agree for Fermi-LAT, power law is the most common ansatz. So far, in unbinned HESS analyses, I used a GModelSpectralConst (with GCTABackground3D), where I did not encounter problems. This is also implemented in cshessobs. Of course, for unbinned analysis we have many runs and thus many free parameters. For stacked binned analysis, I also tried the constant function, which seemed to give reasonable results. But since there are much less free parameters in stacked binned analysis, we could have the additional power law index to better model the background.

#26 - 03/30/2015 12:01 PM - Knödlseder Jürgen

Changes are merged into devel.