ctools - Feature #541

Tools for reflected - background method spectra

10/10/2012 03:37 PM - Deil Christoph

Status:	New	**Start date:**	10/10/2012
Priority:	Normal	**Due date:**	
Assigned To:	Deil Christoph	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Introduction

We should implement tools to fill and fit spectra, i.e. counts and exposure in energy bins, using the reflected-background (and other) methods.

This requires some gammalib additions first (see issues #535, #536, #540).

Heidelberg HESS methods /tools

In the Heidelberg HESS software there are two tools:

1. The hap tool fills the Spectrum, running an in-memory chain of a BgMaker and a SpectrumMaker (and others that are not so important for the discussion here).
2. The FitSpectrum tool fits models to the Spectrum

Spectrum basically contains a few 1-dimensional ROOT histograms for bins in log(energy) with 24 bins/decade: n_on, n_off, acceptance_on, acceptance_off, gamma_exposure_on (see #540)

The hap step typically runs for an hour on one CPU for about 100 runs.
One reason it's slow is that it processes all data, typically many GB (because we currently don't have post-cut event lists for all our data), whereas the output Spectrum only contains a MB of data.
The other reason it's slow is that computing the off region positions can be slow for complicated shapes / exclusion masks (see #535) and because computing gamma_exposure_on from the effective area lookups for the many time steps (and in some cases may points in the on region) can be slow.

The FitSpectrum step only runs for a few seconds (i.e. can be used interactively), because we currently don't take energy resolution into account and only fit the TotalSpectrum, which is the sum of the runwise Spectrum objects.

French HESS methods / tools

If I understand correctly, in French HAP and Model++ spectral analysis also use a two-step process, but what is done in each step is very different from Heidelberg:

1. The first step generates on and off event lists, storing the energy (and maybe zenith angle, ...) for each event.
2. The second step bins the events in energy, accesses the effective area and energy resolution lookups to compute the exposure (effective area x livetime) and effective energy resolution (averaging over the energy resolution every few minutes) and then does the fit on the run wise spectra.

The first step takes about an hour for 100 runs because like Heidelberg they don't maintain post-select event lists and so have to read and process a few GB / MEvents.
The second step also takes about an hour for 100 runs, because they do take the energy resolution into account and fit runwise data (i.e. have 100 x as many bins for 100 runs).

Vincent Marandon tells me that reading the event lists from disk and binning them in energy, and reading the effective area and energy resolution lookups from disk and computing the exposure and effective energy resolution for each run is fast compared to the fitting, so they never bothered storing these intermediate results which are model-independent and could be reused when fitting e.g. different spectral shapes.

Analysis steps

I think computing a spectrum can be split in these steps:

1. Compute the off regions
2. Make an on and off event list
3. Bin the events in energy
4. Compute exposure (=effective area x livetime) and energy resolution
5. Computing acceptance_on and acceptance_off is trivial for the reflected background method (acceptance_on=1, acceptance_off=number of off regions), but would be a complicated computation for other background methods.
6. Sum up runwise spectra into a total spectrum (optional, introduces small errors, looses some sensitivity, but large gains in speed for the fitting)
7. Run the OnOffFitter for a user-selected spectral model

What tools / file formats do we want?

As shown by the fact how different the HD and French HESS tools are, there are several ways to combine analysis steps, so we have to discuss what ctools we want.

The steps 2, 3, 6 and 7 could be done by existing ftools (fools accoding to my spellchecker) or xspec / sherpa.
On the other had 2, 3 and 6 are trivial one-page programs in case we don't want to depend on the ftools.

creflectedbg_regions

I think we should have a tool and output data format for step 1, e.g. Karl wrote something similar as a python script and used ds9 region files (ascii format):
https://bitbucket.org/kosack/act-analysis/src/8e2642c8e710/regionbg.py
This file would be an input for steps 2, 4 and 5.

cexposure

This tool would take the on region info, run list info and compute the exposure and energy resolution from the corresponding lookups.
Output in ARF and RMF format?

creflectedbg_background

Needed?
Output format?

csumruns

Sum up n_on, n_off, acceptance_on, acceptance_off, gamma_exposure_on and perform some per-run weighting with background rate for acceptance_on, acceptance_off.
Needed?
Output format?

Discussion

At some point we want to produce spectra / maps / cubes with the reflected-, ring-, template-, on-off background methods.

How do we structure our tools / toolchains so that it is still simple to understand for the end-user and we keep code duplication and file formats for intermediate files to an acceptable level?

Maybe have many small tools and then the end-user only deals with toolchains, like what HAP does (only with well-defined interfaces, not a global HESSArray)?

History

#1 - 10/10/2012 08:54 PM - Knödlseder Jürgen

- Target version set to HESS sprint 1

#2 - 10/10/2012 09:10 PM - Knödlseder Jürgen

I think that ideally we would like to have a toolchain that is XSPEC compatible, i.e. the end product of the spectral binning operation should be able to be fed into XSPEC.

I would start by putting this toolchain together using cscripts, i.e. Python scripts that emulate ftools. In that way the code can be put quickly together, and me may get a better feeling for what part of the code warrants extension of GammaLib and what part of the code should be simple implemented in a ctools.

#3 - 01/19/2014 02:05 AM - Knödlseder Jürgen

- Target version deleted (HESS sprint 1)

#4 - 01/19/2014 02:07 AM - Knödlseder Jürgen

- Target version set to 2nd coding sprint

#5 - 07/19/2014 02:15 AM - Knödlseder Jürgen

- Target version deleted (2nd coding sprint)