

Convert / store / query HESS data for ctools usage

10/14/2012 10:50 PM - Deil Christoph

Status:	Closed	Start date:	10/14/2012
Priority:	Normal	Due date:	
Assigned To:	Deil Christoph	% Done:	0%
Category:		Estimated time:	0.00 hour
Target version:			

Description

We want to make it possible to run analyses of HESS data with ctools.

Here we give an overview of how HESS data is currently handled in the HESS software, what we want for ctools and finally the concrete tasks required to make it happen.

Once the HESS data (this issue) and IRFs (issue #536) are available from gammalib / ctools, we can discuss tools and toolchains for actual HESS data analysis.

The HESS data is private to the HESS collaboration, still it makes sense to develop the formats / tools to make this possible partly in the gammalib / ctools repository. A very small subset of the HESS data has already been released as part of the first CTA data challenge (1DC), and it is planned to release further larger subsets for future CTA data challenges. An alternative to using real data to test the data handling / tool chains would be to simulate 1000s of HESS or CTA runs for a future data challenge.

HESS data processing with findruns / HAP

Let me start by briefly describing how an end-user runs a HESS analysis in Heidelberg. I believe this info will be useful when discussing the formats and tools we want for gammalib / ctools.

- First a runlist is created for the target of interest, e.g.

```
findruns.pl --name Crab --radius 3 --mintels 3 --selection spectral > crab_runs.lis
(actualy there's a dozen options)
```

This will query a MySQL database table with run quality information (and a table of common sources as a convenient alternative to giving sky position coordinates directly) and write an ascii file like

```
# RunNumber TelescopePattern
23037 30
24099 14
```

which specifies for each run which telescopes have good-quality data that should be used in analysis.

- Then the user runs an analysis, e.g.

```
hap --runlist crab_runs.lis --config std --Background/Method ReflectedBg --Spectrum/Method Standard
(actualy there's about 100 options)
```

which for each run in the runlist reads one ROOT data file in \$HESSDST (a folder specified via an environment variable), where runs are stored according to a fixed scheme, namely in sub-folders each covering 200 runs, e.g. run030000-030199 with filenames such as run_030011_DST_001.root .

Note that the DST input files for HAP contain shower images (after tail cuts) and Hillas parameters for all events so are quite large (a few 100 MB per run). HAP runs direction and energy reconstruction based on the telescope pattern from the runlist and then performs gamma-hadron separation based on the "config" chosen. Only then the high-level spectral or morphological analysis starts using the post-select event list and pre-computed lookups appropriate for this telescope pattern and config. See issue #536 for a discussion on how to integrate the HESS lookups in gammalib / ctools. HAP then creates an output ROOT file with a Stats, Spectrum or Map object from which end results can easily be obtained e.g.

by fitting a model.

The database tables and files in \$HESSDST are created and maintained by experts, the end-user simply copies and uses them.

HESS data processing with gammalib / ctools

We want to define formats and implement tools to make it possible to run similar analyses of HESS data with ctools, with one very important difference: for now we want to create post-select FITS event lists for a given "telescope pattern" and "config" for each run with HAP and only run the high-level analysis with ctools (i.e. no shower image analysis, Hillas parameters, direction and energy reconstruction, gamma-hadron separation). Using the HD cluster I could within a day create six event list versions for each run for the common data quality choices ("spectral" and "detection") and selection cuts ("hard", "std" and "loose"). By only supporting these most common data quality and selection cuts, we also get huge savings for the end user in disk space and processing speed (because much fewer events have to be processed using fewer lookups) by sacrificing the flexibility to use your own data quality and gamma-hadron separation cuts (which is rarely ever done anyways because you also have to create custom lookups, processing about a terabyte of gamma simulations).

Tasks

So actually converting the HESS data for ctools is relatively straight-forward, I think the work can be split in these subtasks:

- A) Create FITS event lists for all good-quality HESS data for the common configs and distribute / store them in some well-defined form.
- B) Create a run info table summarising what runs are available (locally and / or for download)
- C) Create a ctfndruns tool (findruns equivalent) that queries the run info table for a given target, data quality selection and gamma-hadron selection cuts, creating a run or file list.

I will create sub-tasks to discuss the details (file format, location, content) of these three tasks.

History

#1 - 10/14/2012 11:08 PM - Deil Christoph

- Description updated

Actually, before we write down concrete subtasks, Jürgen, can you please comment on what has already been discussed / decided and what you think?

Should this issue of how to distribute / store / query the available HESS (and possibly other experiments) data on the user system be dealt with by gammalib / ctools? Or do you think that (like for Fermi LAT e.g.) this should not be part of the tools, but there is e.g. a data server and it is up to the user to get the data and organise it however she likes on her system?

I think we should make it as simple as possible for HAP users to start switching to ctools, but managing all data and run lists could also be done by a few small python scripts in the HESS CVS instead, either way is fine with me.

#2 - 10/14/2012 11:50 PM - Knödlseider Jürgen

Christoph Deil wrote:

Actually, before we write down concrete subtasks, Jürgen, can you please comment on what has already been discussed / decided and what you think?

You mean discussed within CTA, or in the various feature issues?

Should this issue of how to distribute / store / query the available HESS (and possibly other experiments) data on the user system be dealt with by gammalib / ctools? Or do you think that (like for Fermi LAT e.g.) this should not be part of the tools, but there is e.g. a data server and it is up to the user to get the data and organise it however she likes on her system?

I have so far not a clear picture how this should be done, so I guess for the moment I would opt for a user organization. I guess the most critical thing is to get the data and response functions in the right format, and then implement the format readers. For the event list this is basically done. For the response, I would propose some format, but I need then help from a HESS expert to make the conversion (as I don't have the IRF data).

I think we should make it as simple as possible for HAP users to start switching to ctools, but managing all data and run lists could also be done by a few small python scripts in the HESS CVS instead, either way is fine with me.

I agree. The essential thing is a low entry threshold.

#3 - 10/15/2012 12:03 AM - Knödlseider Jürgen

Christoph Deil wrote:

The HESS data is private to the HESS collaboration, still it makes sense to develop the formats / tools to make this possible partly in the gammalib / ctools repository. A very small subset of the HESS data has already been released as part of the first CTA data challenge (1DC), and it is planned to release further larger subsets for future CTA data challenges. An alternative to using real data to test the data handling / tool chains would be to simulate 1000s of HESS or CTA runs for a future data challenge.

Are the response functions also private, or can they be distributed?

HESS data processing with findruns / HAP

Let me start by briefly describing how an end-user runs a HESS analysis in Heidelberg. I believe this info will be useful when discussing the formats and tools we want for gammalib / ctools.

- First a runlist is created for the target of interest, e.g.
[...]
This will query a MySQL database table with run quality information (and a table of common sources as a convenient alternative to giving sky position coordinates directly) and write an ascii file like
[...]
which specifies for each run which telescopes have good-quality data that should be used in analysis.
- Then the user runs an analysis, e.g.
[...]
which for each run in the runlist reads one ROOT data file in \$HESSDST (a folder specified via an environment variable), where runs are stored according to a fixed scheme, namely in sub-folders each covering 200 runs, e.g. run030000-030199 with filenames such as run_030011_DST_001.root .

I guess the HESS data will be one FITS file per run with increasing sequence number? Could the essential quality information also be packed into a FITS file? One could then use standard tools for Good Time Interval (GTI) selection.

Note that the DST input files for HAP contain shower images (after tail cuts) and Hillas parameters for all events so are quite large (a few 100 MB per run). HAP runs direction and energy reconstruction based on the telescope pattern from the runlist and then performs gamma-hadron separation based on the "config" chosen. Only then the high-level spectral or morphological analysis starts using the post-select event list and pre-computed lookups appropriate for this telescope pattern and config.

And all this is done in one shot?

HESS data processing with gammalib / ctools

We want to define formats and implement tools to make it possible to run similar analyses of HESS data with ctools, with one very important difference: for now we want to create post-select FITS event lists for a given "telescope pattern" and "config" for each run with HAP and only run

the high-level analysis with ctools (i.e. no shower image analysis, Hillas parameters, direction and energy reconstruction, gamma-hadron separation). Using the HD cluster I could within a day create six event list versions for each run for the common data quality choices ("spectral" and "detection") and selection cuts ("hard", "std" and "loose"). By only supporting these most common data quality and selection cuts, we also get huge savings for the end user in disk space and processing speed (because much fewer events have to be processed using fewer lookups) by sacrificing the flexibility to use your own data quality and gamma-hadron separation cuts (which is rarely ever done anyways because you also have to create custom lookups, processing about a terabyte of gamma simulations).

Are these 6 event lists inclusive (i.e. events of one list are also included in another list), or are the events different? If they were inclusive, one could have a bit mask to flag to which selection cut they belong.

In any case, we also need the IRF for these 6 event list versions. The CALDB scheme I have in mind for the IRFs can nicely handle this.

Tasks

So actually converting the HESS data for ctools is relatively straight-forward, I think the work can be split in these subtasks:

- A) Create FITS event lists for all good-quality HESS data for the common configs and distribute / store them in some well-defined form.
- B) Create a run info table summarising what runs are available (locally and / or for download)
- C) Create a ctfindruns tool (findruns equivalent) that queries the run info table for a given target, data quality selection and gamma-hadron selection cuts, creating a run or file list.

For the moment I use XML files to combine runs. The format is pretty simple and straight forward, and adapted to 1DC. It probably will (should?) evolve. I still have not checked what a VO compliant format for observations (i.e. runs) would be, but ideally I would like to implement a VO compliant format (if it fits and if it exists).

I will create sub-tasks to discuss the details (file format, location, content) of these three tasks.

#4 - 10/15/2012 09:41 AM - Deil Christoph

Jürgen Knödlseder wrote:

Christoph Deil wrote:

The HESS data is private to the HESS collaboration, still it makes sense to develop the formats / tools to make this possible partly in the

gammalib / ctools repository. A very small subset of the HESS data has already been released as part of the first CTA data challenge (1DC), and it is planned to release further larger subsets for future CTA data challenges. An alternative to using real data to test the data handling / tool chains would be to simulate 1000s of HESS or CTA runs for a future data challenge.

Are the response functions also private, or can they be distributed?

I just talked to Werner Hofmann. The HESS response functions can be distributed, but we should add a license.txt similar to what we added for the HESS 1DC data, stating that the purpose of the release is to help CTA software / data flow development, but the HESS IRFs may not be used for publications.

which for each run in the runlist reads one ROOT data file in \$HESSDST (a folder specified via an environment variable), where runs are stored according to a fixed scheme, namely in sub-folders each covering 200 runs, e.g. run030000-030199 with filenames such as run_030011_DST_001.root .

I guess the HESS data will be one FITS file per run with increasing sequence number?

Yes, there's about 20,000 runs, each with a unique number, chronological in time, but not consecutive, because there are calibration runs.

Could the essential quality information also be packed into a FITS file? One could then use standard tools for Good Time Interval (GTI) selection.

I think the essential quality information is only which telescopes were used for direction / energy reconstruction for that run. This is given by the telescope pattern and that info should be in the TELARRAY FITS extension, but I didn't check if it is filled correctly for 3-tel runs yet. The way HESS calibration and DST production works there will never be sub time intervals within a run where the telescope pattern e.g. is different, so I don't think we should worry about that now.

The only information missing from the current FITS event lists is the muon efficiency for each telescope. This is needed for computing the effective area and PSF of that run and the FITS event list file is most likely the right place to add it. Let's discuss with Karl which extension is most appropriate (EVENTS, TELARRAY, GTI), if we ever plan to combine several runs in one FITS file (e.g. to avoid having to distribute several 10,000 files) we have to think carefully, if not it doesn't matter much.

Note that the DST input files for HAP contain shower images (after tail cuts) and Hillas parameters for all events so are quite large (a few 100 MB per run). HAP runs direction and energy reconstruction based on the telescope pattern from the runlist and then performs gamma-hadron separation based on the "config" chosen. Only then the high-level spectral or morphological analysis starts using the post-select event list and pre-computed lookups appropriate for this telescope pattern and config.

And all this is done in one shot?

In HD, yes. In France they have two steps, where the first is basically creating on and off event list, and all the high-level analysis (including lookup access) is done in a second step.

HESS data processing with gammalib / ctools

We want to define formats and implement tools to make it possible to run similar analyses of HESS data with ctools, with one very important difference: for now we want to create post-select FITS event lists for a given "telescope pattern" and "config" for each run with HAP and only run the high-level analysis with ctools (i.e. no shower image analysis, Hillas parameters, direction and energy reconstruction, gamma-hadron separation). Using the HD cluster I could within a day create six event list versions for each run for the common data quality choices ("spectral" and "detection") and selection cuts ("hard", "std" and "loose"). By only supporting these most common data quality and selection cuts, we also get huge savings for the end user in disk space and processing speed (because much fewer events have to be processed using fewer lookups) by sacrificing the flexibility to use your own data quality and gamma-hadron separation cuts (which is rarely ever done anyways because you also have to create custom lookups, processing about a terabyte of gamma simulations).

Are these 6 event lists inclusive (i.e. events of one list are also included in another list), or are the events different? If they were inclusive, one could have a bit mask to flag to which selection cut they belong.

Well, for different telpatterns the event lists will not be inclusive.

I think we have hard < std < loose, so in principle we could just distribute the loose cut event list and have flags for each event if it also passes std or hard cuts.

Thinking about it again, I think to make quick progress the easiest solution is to only distribute "spectral" quality data (this is what we use for all publications anyways, detection is mostly used to find hotspots) and make separate files for hard, std, loose, unless combining them into one would work easily with the FITS event list format as it is now already.

In any case, we also need the IRF for these 6 event list versions. The CALDB scheme I have in mind for the IRFs can nicely handle this.

Great!

Tasks

So actually converting the HESS data for ctools is relatively straight-forward, I think the work can be split in these subtasks:

- A) Create FITS event lists for all good-quality HESS data for the common configs and distribute / store them in some well-defined form.
- B) Create a run info table summarising what runs are available (locally and / or for download)
- C) Create a ctfindruns tool (findruns equivalent) that queries the run info table for a given target, data quality selection and gamma-hadron selection cuts, creating a run or file list.

For the moment I use XML files to combine runs. The format is pretty simple and straight forward, and adapted to 1DC. It probably will (should?) evolve. I still have not checked what a VO compliant format for observations (i.e. runs) would be, but ideally I would like to implement a VO compliant format (if it fits and if it exists).

OK, for now I'll then do this:

- A) Store one FITS file per config / run in a folder, user has to set \$HESSDATA to point there. Data can be distributed as a tarball of that folder.
- B) I'll write a python script that walks \$HESSDATA and creates a CSV table summarising the run number, telpattern, target, position, time, absolute file location ... of each run from the FITS header. The reason for this run info table is to make C) quick.
- C) Write a python script to create file lists in your format. Let me know what you decide.

I think the most important point is that we will be working with "file lists" not "run lists", i.e. the actual file locations will be included in the list, so that the ctools don't need to know / care about \$HESSDATA and the file structure there any more. I like it.

Christoph Deil wrote:

I just talked to Werner Hofmann. The HESS response functions can be distributed, but we should add a license.txt similar to what we added for the HESS 1DC data, stating that the purpose of the release is to help CTA software / data flow development, but the HESS IRFs may not be used for publications.

That's great. I just started the download. Takes some time :-)

The only information missing from the current FITS event lists is the muon efficiency for each telescope. This is needed for computing the effective area and PSF of that run and the FITS event list file is most likely the right place to add it. Let's discuss with Karl which extension is most appropriate (EVENTS, TELARRAY, GTI), if we ever plan to combine several runs in one FITS file (e.g. to avoid having to distribute several 10,000 files) we have to think carefully, if not it doesn't matter much.

So this efficiency is not in the root IRF files but has to be added on later? If true, this would speak somehow for a scheme where we have the IRF per run as we then could fold-in the efficiencies in the file.

Well, for different telpatterns the event lists will not be inclusive.

I think we have $\text{hard} < \text{std} < \text{loose}$, so in principle we could just distribute the loose cut event list and have flags for each event if it also passes std or hard cuts.

Thinking about it again, I think to make quick progress the easiest solution is to only distribute "spectral" quality data (this is what we use for all publications anyways, detection is mostly used to find hotspots) and make separate files for hard, std, loose, unless combining them into one would work easily with the FITS event list format as it is now already.

Okay. There is no problem with having separate FITS files. They are anyways not very large.

OK, for now I'll then do this:

A) Store one FITS file per config / run in a folder, user has to set \$HESSDATA to point there. Data can be distributed as a tarball of that folder.

B) I'll write a python script that walks \$HESSDATA and creates a CSV table summarising the run number, telpattern, target, position, time, absolute file location ... of each run from the FITS header. The reason for this run info table is to make C) quick.

C) Write a python script to create file lists in your format. Let me know what you decide.

I think for the moment we keep the actual format. We can always change this later.

Note that the format I introduced for 1DC as fields for ARF, RMF, PSF. I propose to leave them just blank, and handle the response access by GCaldb.

What is the reason for having step B) and C) not combined in a single script, i.e. write the result directly in XML?

I think the most important point is that we will be working with "file lists" not "run lists", i.e. the actual file locations will be included in the list, so that the ctools don't need to know / care about \$HESSDATA and the file structure there any more. I like it.

Good.

Jürgen Knödlseider wrote:

Christoph Deil wrote:

I just talked to Werner Hofmann. The HESS response functions can be distributed, but we should add a license.txt similar to what we added for the HESS 1DC data, stating that the purpose of the release is to help CTA software / data flow development, but the HESS IRFs may not be used for publications.

That's great. I just started the download. Takes some time :-)

The only information missing from the current FITS event lists is the muon efficiency for each telescope. This is needed for computing the effective area and PSF of that run and the FITS event list file is most likely the right place to add it. Let's discuss with Karl which extension is most appropriate (EVENTS, TELARRAY, GTI), if we ever plan to combine several runs in one FITS file (e.g. to avoid having to distribute several 10,000 files) we have to think carefully, if not it doesn't matter much.

So this efficiency is not in the root IRF files but has to be added on later?

Each IRF is for a set of efficiencies for the four telescopes used in the gamma-ray simulations that were used to fill the IRFs, e.g. if CT4 was built first and then all tels degraded you could have

CT1, CT2, CT3, CT4
0.1, 0.1, 0.1, 0.08
0.09, 0.09, 0.09, 0.07

Then each run has a set of efficiencies, e.g. run 12345 could have

CT1, CT2, CT3, CT4
0.1, 0.095, 0.96, 0.086

and then you'd interpolate the two most closely matching simulations to compute an effective area and PSF for that run.

If true, this would speak somehow for a scheme where we have the IRF per run as we then could fold-in the efficiencies in the file.

This is what MAGIC does (as I understand). Each scheme has advantages and disadvantages, we have to investigate both for CTA. For HESS I propose we simply convert the same (per-muon-efficiency-phase) IRFs to FITS format and think about additionally implementing per-run IRFs (a big project) later.

I think for the moment we keep the actual format. We can always change this later.

Can you link to some code /script / description of the XML format you use?

What is the reason for having step B) and C) not combined in a single script, i.e. write the result directly in XML?

To limit disk I/O and be fast. If you don't do the indexing in step B), the tool that creates the runlist would have to open 20,000 files and read FITS header keywords each time it runs.

#7 - 10/15/2012 03:01 PM - Knödseder Jürgen

- File *obs_binned.xml* added
- File *obs_unbinned.xml* added

Here the XML files for binned and unbinned analysis. For event lists you start with unbinned ...

#8 - 10/15/2012 05:37 PM - Knödseder Jürgen

- File *irf_binned.xml* added
- File *irf_unbinned.xml* added

Coming back to the XML file definition, this is probably close to what we want to implement. The ARF, RMF, and PSF was really shaped towards 1DC, but it is preferable to have human readable parameter names.

I updated GammaLib to support this new format (in hess branch).

We furthermore need some more definitions (see action #537)

#9 - 11/03/2013 02:09 AM - Knödseder Jürgen

- Target version deleted (*HESS sprint #1*)

#10 - 06/21/2016 10:21 PM - Knödseder Jürgen

- Status changed from *New* to *Closed*

Files			
obs_binned.xml	329 Bytes	10/15/2012	Knödseder Jürgen
obs_unbinned.xml	329 Bytes	10/15/2012	Knödseder Jürgen
irf_binned.xml	547 Bytes	10/15/2012	Knödseder Jürgen
irf_unbinned.xml	547 Bytes	10/15/2012	Knödseder Jürgen