GammaLib - Action #537

Feature # 536 (Closed): Add HESS instrument response functions

Convert HESS IRFs to FITS format and create a CALDB index

10/10/2012 11:36 AM - Deil Christoph

Status:	Closed	**Start date:**	10/10/2012
Priority:	Normal	**Due date:**	
Assigned To:	Deil Christoph	**% Done:**	20%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

The HESS IRFs are stored in 1- and 2-dimensional histograms in these ROOT files (the file sizes are for std cuts, for hard / loose cuts the file size is smaller/larger because there are more / less empty bins which are automatically compressed by ROOT):

- EffectiveAreas.root 100 MB
- EnergyInfo.root 500 MB
- EnergyReconstruction.root 400 MB
- EnergyResolution.root 300 MB
- EnergyResolution2.root 100 MB
- PSF.root 600 MB
- RadialAcceptance.root 2 MB
- RadialAcceptanceOff.root 2 MB
- ScaleInfo.root 300 MB

They are created for each set of cuts (e.g. std or hard or loose) basically by histogramming events from simulated point-source gamma-ray events and in the case of the radial acceptance from real background events in runs that contain no gamma-ray source by the fillLookups.pl script in the HESS software. In some cases after histogramming an extra smoothing or fitting step is applied to get good IRFs in parameter regions which contain only few events. The attached PDF document describes the procedure and IRFs in detail, note especially Table 1 which describes the parameters (like zenith angle, ...) and histogram axes.

For analysis with gammalib / ctools based on FITS event lists, we only need the effective area, energy resolution, PSF and radial acceptance. The others are used for gamma-hadron separation and energy reconstruction, which has already happened for the FITS event lists.

So please comment:

- What format do we want for the FITS files? Arrays or tables? Header keywords?
- In principle one can create few large arrays with many axes or many small arrays with few axes. In practice some ways are more efficient than others when using the IRFs in analysis, because more or less data has to be read from disk. This is certainly an important topic, but for now I would suggest we simply use one or two-dimensional arrays with the same axes as the HESS histograms.
- For now I would suggest we use Hillas hard / std / loose and leave e.g. TMVA and Model analysis for a later time.
- There's some details about the use of the effective area (with reco or true energy binning on the x-axis) and energy resolution (with energy_reco or energy_reco / energy_true - 1 on the y axis) we have to discuss.

History

#1 - 10/10/2012 11:42 AM - Deil Christoph

- File lookups.pdf added

#2 - 10/10/2012 08:48 PM - Knödlseder Jürgen

- Target version set to HESS sprint #1

#3 - 10/10/2012 10:36 PM - Knödlseder Jürgen

- File ea_test.fits added

- File edisp_test.fits added

- File psf_test.fits added

In the ctools software, script directory, there is a Python script called cta_root2irf.py. This script converts simulated CTA response files (in ROOT) to a CALDB like response format. I think this is a good starting point.

I would suggest to start from this script, creating a hess_root2irf.py script that puts the response functions in FITS format. I have an IRF format in mind that is largely inspired from Fermi/LAT and, I think, very convenient for CTA (and also HESS).

Here are 3 IRF FITS files that were created by cta_root2irf.py and that illustrate the format that I have in mind:

- attachment:ea_test.fits
- attachment:edisp_test.fits
- attachment:psf_test.fits

These file stores some parameters in a 2D array, as function of energy and offaxis angle. The dimensions can be easily extended, hence it should be possible to accommodate also the HESS IRFs.

#4 - 10/10/2012 10:43 PM - Knödlseder Jürgen

Christoph Deil wrote:

> So please comment:
>
> - What format do we want for the FITS files? Arrays or tables? Header keywords?

See the attached examples.

- In principle one can create few large arrays with many axes or many small arrays with few axes. In practice some ways are more efficient than others when using the IRFs in analysis, because more or less data has to be read from disk. This is certainly an important topic, but for now I would suggest we simply use one or two-dimensional arrays with the same axes as the HESS histograms.

Indeed, for now 2D responses would be best. The actual CTA scheme reads in a response file for each observation, and having a 2D response per run would fit this scheme best. Otherwise, we would need some interpolation / extraction procedure during loading, which could also be done, but requires more coding to get started.

The format proposed above is flexible, so we can start with 2D and change (or extend) it later if requested.

- For now I would suggest we use Hillas hard / std / loose and leave e.g. TMVA and Model analysis for a later time.

Agree.

- There's some details about the use of the effective area (with reco or true energy binning on the x-axis) and energy resolution (with energy_reco or energy_reco / energy_true - 1 on the y axis) we have to discuss.

For forwards folding, effective area should be as function of true energy, but maybe the HESS methods need reco energy? Maybe we should simply store both for the moment.

Another related question: do we want to implement energy dispersion in a first place. Nothing has been done for this in GammaLib so far, hence this would evolve quite some coding ... maybe we can defer this to later?

#5 - 10/15/2012 12:11 AM - Knödlseder Jürgen

Christoph Deil wrote:

> - In principle one can create few large arrays with many axes or many small arrays with few axes. In practice some ways are more efficient than others when using the IRFs in analysis, because more or less data has to be read from disk. This is certainly an important topic, but for now I would suggest we simply use one or two-dimensional arrays with the same axes as the HESS histograms.

I think this is one of the most important points to decide on. As I understand we have the following axes:

- energy
- offset angle
- zenith angle
- azimuth angle (is this used for HESS?)
- optical efficiency
- telescopes
- cuts
- (something else?)

We need all this information, somehow. I guess there are two options for storing the IRFs:

1. store a multi-dimensional cube
2. store a 2D IRF (energy, offset) for each run (or do we need a third dimension with trigger pattern?)

I know that Karl favors 1, for CTA I have a certain preference for 2 (mainly because this option allows probably a more precise response computation). If we use 1, we need to implement software in GammaLib that computes for each observation the relevant response (energy, offset, trigger?). If we use 2, this software lives somewhere outside GammaLib (HAP?), and we just compile the files as needed.

#6 - 10/15/2012 04:28 PM - Deil Christoph

Jürgen, Karl and I discussed this today in a teleconference and we decided to do this for now to create per-run IRFs in the format Jürgen proposed above (see example FITS files) by extending the hap-to-irf tool in the HESS software.

Here are the details:

- hap-to-irf will take the telescope pattern and muon efficiencies of that run (opt and telp in Table 1 in lookups.pdf) and average over the different azimuth and zenith angles that occur during the run and create these IRFs:
  - effective area for ranges of offsets and true energies (for use with energy resolution)
  - effective area for ranges of offsets and reco energies (for use without energy resolution, giving speed)
  - PSF for ranges of offsets, true energies and theta^2
  - energy resolution for ranges of offsets, true energies and (e_reco - e_true) / e_true
  - acceptance for ranges of offset^2 (these are simply energy-summed versions of the next IRF)
  - acceptance for ranges of offset^2 and reco energy
- These IRFs include the offset dependence and thus work for the whole field of view, allowing the analysis of extended sources.
- Because the ARF format doesn't allow the storage of effective areas, and there is no OGIP format for PSFs anyways, we'll just use the flexible format that Fermi uses and that Jürgen proposed above, which basically is a BINTABLE with one row storing an array :-)

```
$ ftlist ea_test.fits H
```

```
      Name              Type       Dimensions
      ----              ----       ----------
HDU 1   Primary Array      Null Array
HDU 2   EFFECTIVE AREA     BinTable     5 cols x 1 rows
$ ftlist ea_test.fits C
HDU 2

 Col  Name           Format[Units](Range)      Comment
   1 ENERG_LO         20E [TeV]           label for field   1
   2 ENERG_HI         20E [TeV]           label for field   2
   3 THETA_LO         16E [deg]           label for field   3
   4 THETA_HI         16E [deg]           label for field   4
   5 EFFAREA          320E [m2]           label for field   5
```

- For acceptance we decided to use actual FITS cubes (instead of array data in BINTABLEs) instead. (is this right? I don't care either way)
- Whether these IRFs are distributed as separate files or as FITS extensions in the same file as the event lists doesn't matter. We have to produce them separately because for event lists we use hap and for IRFs we use hap-to-irf, but we can easily combine them in one file afterwards to make things simpler for the user. X-ray instruments sometimes store the ARF and RMF filename in the event list FITS header and automatically connect the data and IRF that way, that works too.

Putting the whole HESS data in FITS CALDB and computing per-run IRFs from that will be left as a future task.

#7 - 10/15/2012 04:44 PM - Knödlseder Jürgen

Christoph Deil wrote:

> - For acceptance we decided to use actual FITS cubes (instead of array data in BINTABLEs) instead. (is this right? I don't care either way)

Thinking about it twice, I'm wondering what the best format is. Ultimately, we would feed a FITS cube into GammaLib, but this cube depends on the exact binning of the events (unless we implement internal rebinning). The event cube is defined using ctbin. I think the most logical would be to have a tool that converts the acceptance information in a map cube, using the same binning as was used for the events cube. So maybe we should store the acceptance also in a binary table ...

Karl, any thoughts about this?

> - Whether these IRFs are distributed as separate files or as FITS extensions in the same file as the event lists doesn't matter. We have to produce them separately because for event lists we use hap and for IRFs we use hap-to-irf, but we can easily combine them in one file afterwards to make things simpler for the user. X-ray instruments sometimes store the ARF and RMF filename in the event list FITS header and automatically connect the data and IRF that way, that works too.

Right, so we take provision for specifying file names for the events and the IRF files. In case that all extensions live in the same place, these names can be identical.

Putting the whole HESS data in FITS CALDB and computing per-run IRFs from that will be left as a future task.

Agree.

#8 - 10/15/2012 05:08 PM - Knödlseder Jürgen

We should define unique extension names for all the IRF informations. What about

- EFFECTIVE AREA effective area for ranges of offsets and true energies (for use with energy resolution)
- EFFECTIVE AREA RECO effective area for ranges of offsets and reco energies (for use without energy resolution, giving speed)
- POINT SPREAD FUNCTION PSF for ranges of offsets, true energies and theta^2
- ENERGY DISPERSION energy resolution for ranges of offsets, true energies and (e_reco - e_true) / e_true
- ACCEPTANCE acceptance for ranges of offset^2 (these are simply energy-summed versions of the next IRF)
- ACCEPTANCE RECO acceptance for ranges of offset^2 and reco energy

#9 - 10/15/2012 05:46 PM - Knödlseder Jürgen

- File CTA1DC-HESS-run00023523_std_psf.fits added

We even need more discussions and definitions. In the example file I sent we have the following:

- ea_test.fits[EFFECTIVE AREA]
  - EFFAREA (2D)
- psf_test.fits[POINT SPREAD FUNCTION]
  - R68 (2D)
  - R80 (2D)
- edisp_test.fits[ENERGY DISPERSION]
  - ERES68 (2D)

We have to decide upon what we want to store.

For the effective area, we probably want to store the effective area for true and reconstructed energy in the same extension. That's much better than inventing an own extension (unless we want to have different binning for those). So compare to the above I propose to drop EFFECTIVE AREA RECO.

For the point spread function, the file I uploaded has 68% and 80% error radii. We had something better for 1DC, which was a parametric prescription of the PSF. It is attached above. As you can see, there 6 parameters stored in the file, and I guess we want the same here.

For the energy dispersion, there is the 68% energy resolution so far. I leave it up to you to come up with some proposal.

So for the effective area and PSF, I propose to have the following in the files:

- ea_test.fits[EFFECTIVE AREA]
  - EFFAREA (2D)
  - EFFAREA_RECO (2D)
- psf_test.fits[POINT SPREAD FUNCTION]
  - SCALE (2D)
  - SIGMA_1 (2D)
  - AMPL_2 (2D)
  - SIGMA_2 (2D)
  - AMPL_3 (2D)
  - SIGMA_3 (2D)

#10 - 10/15/2012 05:50 PM - Deil Christoph

Jürgen Knödlseder wrote:

> We should define unique extension names for all the IRF informations. What about
>
> ...
>
> - ACCEPTANCE acceptance for ranges of offset^2 (these are simply energy-summed versions of the next IRF)
> - ACCEPTANCE RECO acceptance for ranges of offset^2 and reco energy

Both of these use reco energy, so I wouldn't put reco only in the name of the second.
I'm not sure what the best terms would be, maybe BACKGROUND and ENERGY-DEPENDENT BACKGROUND, because the French use acceptance to mean effective area, so acceptance is too unspecific?

We could also only distribute the energy-dependent version. We'll have to add code to smooth acceptance curves for low-stats regions sooner or later anyways to gammalib to allow the user to select the exact energy band.

#11 - 10/15/2012 05:57 PM - Deil Christoph

Jürgen Knödlseder wrote:

> We even need more discussions and definitions. In the example file I sent we have the following:
>
> - ea_test.fits[EFFECTIVE AREA]
>   - EFFAREA (2D)
> - psf_test.fits[POINT SPREAD FUNCTION]
>   - R68 (2D)
>   - R80 (2D)
> - edisp_test.fits[ENERGY DISPERSION]
>   - ERES68 (2D)
>
> We have to decide upon what we want to store.
>
> For the effective area, we probably want to store the effective area for true and reconstructed energy in the same extension. That's much better than inventing an own extension (unless we want to have different binning for those). So compare to the above I propose to drop EFFECTIVE AREA RECO.

Sounds good.

> For the point spread function, the file I uploaded has 68% and 80% error radii. We had something better for 1DC, which was a parametric prescription of the PSF. It is attached below.

I think we should store the actual theta^2 distributions, which is equivalent to having very many "error radii".
Parametric models are completely work in progress in HESS and having only two "error radii" will not be enough to get a good representation of the PSF shape.

And yes, the per-run IRF will be larger than the per-run event list, at least for HESS std and hard cuts, but I think that is acceptable.

#12 - 10/15/2012 05:58 PM - Deil Christoph

If you have per-run IRF FITS test files, please attach them here and I'll try to make hap-to-irf produce them for actual HESS runs in exactly that format.

#13 - 10/15/2012 06:32 PM - Knödlseder Jürgen

Christoph Deil wrote:

> Jürgen Knödlseder wrote:
>
> > We should define unique extension names for all the IRF informations. What about
>
> ...
>
> - ACCEPTANCE acceptance for ranges of offset^2 (these are simply energy-summed versions of the next IRF)
> - ACCEPTANCE RECO acceptance for ranges of offset^2 and reco energy
>
> Both of these use reco energy, so I wouldn't put reco only in the name of the second.
> I'm not sure what the best terms would be, maybe BACKGROUND and ENERGY-DEPENDENT BACKGROUND, because the French use acceptance to mean effective area, so acceptance is too unspecific?

I agree. When I read HESS PhD theses, I always feel that acceptance is used in various ways, and it's never really clear what exactly is ment. So let's go for BACKGROUND as extension name, and then use for example BGD and BGD_RECO or something similar for both quantities.

What are the axes here? A 3D data cube with energy, offset and zenith angles?

> We could also only distribute the energy-dependent version. We'll have to add code to smooth acceptance curves for low-stats regions sooner or later anyways to gammalib to allow the user to select the exact energy band.

I think that energy dependence would be useful.

#14 - 10/15/2012 06:34 PM - Knödlseder Jürgen

Christoph Deil wrote:

> If you have per-run IRF FITS test files, please attach them here and I'll try to make hap-to-irf produce them for actual HESS runs in exactly that format.

I will adapt the cta_root2irf.py script in the ctools/script folder to the format I have in mind. I'll attach the respective files later.

#15 - 10/15/2012 06:40 PM - Deil Christoph

Jürgen Knödlseder wrote:

> Christoph Deil wrote:
>
> > Jürgen Knödlseder wrote:
> >
> > > We should define unique extension names for all the IRF informations. What about
> >
> > ...
> >
> > - ACCEPTANCE acceptance for ranges of offset^2 (these are simply energy-summed versions of the next IRF)
> > - ACCEPTANCE RECO acceptance for ranges of offset^2 and reco energy
>
> Both of these use reco energy, so I wouldn't put reco only in the name of the second.
> I'm not sure what the best terms would be, maybe BACKGROUND and ENERGY-DEPENDENT BACKGROUND, because the French use acceptance to mean effective area, so acceptance is too unspecific?

I agree. When I read HESS PhD theses, I always feel that acceptance is used in various ways, and it's never really clear what exactly is ment.
So let's go for BACKGROUND as extension name, and then use for example BGD and BGD_RECO or something similar for both quantities.

What are the axes here? A 3D data cube with energy, offset and zenith angles?

It's a 2D data cube with offset^2, log(energy) on the axes and relative background rate as value.
There's separate cubes for a few zenith angle bands.
See RadialAcceptance.root for which binning we currently use.

This is something we really do poorly at the moment in HESS, and what we should improve if we want to use it for spectral analyses.
See https://dl.dropbox.com/u/4923986/Philippe_Peille_Report.pdf for some initial work.

#16 - 10/16/2012 05:57 PM - Knödlseder Jürgen

- Status changed from New to In Progress

#17 - 10/16/2012 05:59 PM - Knödlseder Jürgen

- File irf_test.fits added

- File cta_root2caldb.py added

Here a draft IRF per run file, as discussed yesterday. The keywords are normally CALDB compliant. I also attach the python script that builds this file (and the CALDB index), based on the CTA ROOT files from Barcelona.

Let me know what you think. For the energy dispersion, I just put the 68% resolution for the moment. What exactly should be put in there?

Also, what about the column names and extension names? Are they okay for you?

Once we agreed, I will make sure that GammaLib can digest this file.

#18 - 10/16/2012 06:00 PM - Knödlseder Jürgen

Jürgen Knödlseder wrote:

> Christoph Deil wrote:
>
> > If you have per-run IRF FITS test files, please attach them here and I'll try to make hap-to-irf produce them for actual HESS runs in exactly that format.
>
> I will adapt the cta_root2irf.py script in the ctools/script folder to the format I have in mind. I'll attach the respective files later.

The script is not cta_root2irf.py but cta_root2caldb.py. Sorry for that misleading comment.

#19 - 10/16/2012 10:44 PM - Knödlseder Jürgen

- File cta_root2caldb.py added

- File irf_test.fits added

Here an update of the IRF file and Python script.

I made some minor adjustments, related mainly to the content. As you see, the background extension is now filled.

I also changed a little bit the energy dispersion information, having now ERES (68% containment around Ereco/Etrue=1) and EBIAS (Ereco/Etrue). These parameters are in the latest 2D performance files for CTA, hence I could easily fill the histograms. This also looks like a reasonable

prescription of the energy dispersion to me (assuming that we could model it as a Gaussian). Does this appear reasonable to you, or do you think we need a more complex energy dispersion relation?

#20 - 10/18/2012 09:11 AM - Knödlseder Jürgen

On Wed, Oct 17, 2012 at 4:23 PM, Jürgen Knödlseder
<jurgen.knodlseder@irap.omp.eu> wrote:

> Dear Karl,
>
> > On Tue, Oct 16, 2012 at 5:57 PM, Jürgen Knödlseder
> <jurgen.knodlseder@irap.omp.eu> wrote:
> >
> > > Hi Christoph, Karl,
> > >
> > > Here a draft IRF per run file, as discussed yesterday. The keywords are
> > > normally CALDB compliant. I also attach the python script that builds this
> > > file (and the CALDB index), based on the CTA ROOT files from Barcelona.
> > >
> > > Let me know what you think.
> > > For the energy dispersion, I just put the 68%
> > > resolution for the moment. What exactly should be put in there?
> >
> > I think we really just need the RMF matrix, not just the gaussian models.
>
> Do you think we simply should store Ereco versus Etrue for different offsets? Or Ereco/Etrue versus Etrue for different offsets?

Probably it's easier to have Ereco vs Etrue, especially if we want to
be able to write an RMF file later (otherwise you need to do some
re-sampling of the data, and we've found that introduces some
artifacts)

> For the moment I just plugged in the information that is in the CTA root performance files, but this can be changed of course.
>
> The only reason while I like functional forms is that it's then easier to perform an unbinned analysis (as this requires some integrals to be computed).
>
> > Also, what about the column names and extension names? Are they okay for
> > you?
>
> It would be good to add units to the values, or a description of some
> kind of what is there. The BACKGROUND extension in particular is
> ambiguous: is it the predicted background or just the acceptance?
> Usually the background acceptance (which is basically like the A_eff
> for hadron events) is used to generate a predicted background data
> cube (X,Y,E), which finally has the units of counts. Therefore maybe
> calling it "BGACC" or "ACCEPTANCE" is a better name, since there will
> also be a "background" cube derived from it.

> Right, units need to be added. For the moment it's counts / square degrees (as extracted from the CTA performance files).

> Christoph argued against ACCEPTANCE as it's used for different things (see https://cta-redmine.irap.omp.eu/issues/537), that's why I used background. I have no strong opinion about this ... whatever you agree on is fine for me.

> > In the current CTA sims, they do provide a predicted cosmic ray rate per square degree, but that's not generally available with real data (it is too computationally intensive, and doesn't take into account atmospheric conditions, so it is not very useful for analysis, just for making sensitivity curves)

> Isn't this the same as the A_eff for hadron events you mentioned earlier?
>
> But maybe you point is that the A_eff for hadrons is derived from the data, while the predicted CR rate per square degrees is derived from simulations? In any case, we should be able to use the same format to store the data, right?

Yes, they are similar things, but one is in arbitrary units (what I call acceptance) since it is just a shape, and the other is in counts/deg^2. In X-rays they call the acceptance "vignetting", but I don't like that term either - the correct term is really "background acceptance", since it describes the efficiency of detecting a background event as a function of position in the detector. It is not really an effective detection area per se, since the units aren't cm^2 and it is a function of reconstructed energy. If is normalized to have a unitary integral, it is a probability of detecting a background event at a given position/reconstructed energy. There is no way of generating the same thing for true energy, unlike for Aeff, since the energies of background events are not known.

So I guess it's fine having background rate and background acceptance take the same format, but they should be differentiated in name, since the background rate is calculated differently and the energy scale is reco instead of true?

I'm just forseeing that we will also need to have a background map generator that will take effective areas, acceptances, and real data, and produce a "background" map. It would be therefore confusing to call background acceptance "background".

For the confusion with effective area, an effective area is also a type of acceptance for the signal, so we do need a good glossary to make sure we aren't confusing terms. We need to be clear about each of the following:

- signal effective area (m^2) vs true energy
- signal effective area (m^2) vs reconstructed energy
- background acceptance (a.u.) vs reconstructed energy
- background rate (Hz/deg^2) vs true energy
- signal level (counts/deg^2) vs reconstructed energy
- predicted background level (counts/deg^2) vs reconstructed energy (given real data to take into atmosphere conditions)

we also have to remember that what we call "Effective area" include effects of the signal/background selection (so they are really A_{eff}(E,...) \times \epsilon(E,...)), where epsilon is the signal/background separation efficiency, which is why it may be good to use the word "acceptance" to describe both.

#21 - 10/18/2012 09:27 AM - Knödlseder Jürgen

Karl Kosack wrote:

> Do you think we simply should store Ereco versus Etrue for different offsets? Or Ereco/Etrue versus Etrue for different offsets?

> Probably it's easier to have Ereco vs Etrue, especially if we want to be able to write an RMF file later (otherwise you need to do some re-sampling of the data, and we've found that introduces some artifacts)

I guess that re-sampling is always needed, as you don't know beforehand what the chosen binning would be. And you also need an interpolation scheme.

> Isn't this the same as the A_eff for hadron events you mentioned earlier?
>
> But maybe you point is that the A_eff for hadrons is derived from the data, while the predicted CR rate per square degrees is derived from simulations? In any case, we should be able to use the same format to store the data, right?

> Yes, they are similar things, but one is in arbitrary units (what I call acceptance) since it is just a shape, and the other is in counts/deg^2. In X-rays they call the acceptance "vignetting", but I don't like that term either - the correct term is really "background acceptance", since it describes the efficiency of detecting a background event as a function of position in the detector. It is not really an effective detection area per se, since the units aren't cm^2 and it is a function of reconstructed energy. If is normalized to have a unitary integral, it is a probability of detecting a background event at a given position/reconstructed energy. There is no way of generating the same thing for true energy, unlike for Aeff, since the energies of background events are not known.
>
> So I guess it's fine having background rate and background acceptance take the same format, but they should be differentiated in name, since the background rate is calculated differently and the energy scale is reco instead of true?
>
> I'm just forseeing that we will also need to have a background map generator that will take effective areas, acceptances, and real data, and produce a "background" map. It would be therefore confusing to call background acceptance "background".

Okay, so let's call them BACKGROUND ACCEPTANCE (background acceptance (a.u.) vs reconstructed energy) and BACKGROUND RATE (background rate (Hz/deg^2) vs true energy).

> For the confusion with effective area, an effective area is also a type of acceptance for the signal, so we do need a good glossary to make sure we aren't confusing terms. We need to be clear about each of the following:
>
> - signal effective area (m^2) vs true energy
> - signal effective area (m^2) vs reconstructed energy
> - background acceptance (a.u.) vs reconstructed energy
> - background rate (Hz/deg^2) vs true energy
> - signal level (counts/deg^2) vs reconstructed energy
> - predicted background level (counts/deg^2) vs reconstructed energy (given real data to take into atmosphere conditions)
>
> we also have to remember that what we call "Effective area" include effects of the signal/background selection (so they are really A_{eff}(E,...) \times \epsilon(E,...)), where epsilon is the signal/background separation efficiency, which is why it may be good to use the word "acceptance" to describe both.

I guess this is in the word "effective", so the epsilon should be absorbed in the A_eff.

#22 - 10/19/2012 09:38 AM - Deil Christoph

Jürgen Knödlseder wrote:

> I also changed a little bit the energy dispersion information, having now ERES (68% containment around Ereco/Etrue=1) and EBIAS (Ereco/Etrue). These parameters are in the latest 2D performance files for CTA, hence I could easily fill the histograms. This also looks like a reasonable prescription of the energy dispersion to me (assuming that we could model it as a Gaussian). Does this appear reasonable to you, or do you think we need a more complex energy dispersion relation?

I don't know if the approximation that energy resolution is Gaussian in Ereco/Etrue works well or not. Probably yes, but I haven't seen any study showing that this is the case, especially at the low- and high-energy end, i.e. at the safe energy cut, which we define as the energies where the bias is 10%.

In principle it is best to just take the actual x=Etrue vs. y=(Ereco/Ereco - 1) distribution (given in EnergyResolution.root for HESS) or x=Etrue vs. y=Ereco distribution (given in EnergyResolution2.root), but especially at the high-energy end MC statistics is low, so there will be large statistical fluctuations, which we don't smooth out at the moment, and for EnergyResolution2.root there might be binning errors in addition. So to avoid these problems it might even be better to take the Gaussian approximation, which is given by the EnergyBias_* TProfiles in EffectiveArea.root for HESS, but as I said, I've never seen a study of how well this Gaussian approximation holds, i.e. how large the resulting errors are compared to the true energy resolution.

I think we need some experimentation with IRFs for HESS and CTA (not only energy resolution, but all the other ones as well), gammalib / ctools could be a very nice tool to do such studies.
So for the energy resolution I have no advice, either pick any one of the methods to store it mentioned above (probably all work okish), or implement several if you are interested in this topic.

#23 - 12/01/2012 02:37 AM - Knödlseder Jürgen

- % Done changed from 0 to 20

The hess branch now includes code that allows the reading of the agreed effective area and PSF format. Scientific validation of the code is still needed.

#24 - 11/03/2013 02:09 AM - Knödlseder Jürgen

- Target version deleted (HESS sprint #1)

#25 - 06/21/2016 10:20 PM - Knödlseder Jürgen

- Status changed from In Progress to Closed

Files

lookups.pdf	1.16 MB	10/10/2012	Deil Christoph
ea_test.fits	11.3 KB	10/10/2012	Knödlseder Jürgen
edisp_test.fits	11.3 KB	10/10/2012	Knödlseder Jürgen
psf_test.fits	11.3 KB	10/10/2012	Knödlseder Jürgen
CTA1DC-HESS-run00023523_std_psf.fits	14.1 KB	10/15/2012	Knödlseder Jürgen
irf_test.fits	42.2 KB	10/16/2012	Knödlseder Jürgen
cta_root2caldb.py	24.6 KB	10/16/2012	Knödlseder Jürgen
cta_root2caldb.py	26.2 KB	10/16/2012	Knödlseder Jürgen
irf_test.fits	42.2 KB	10/16/2012	Knödlseder Jürgen