ctools - Feature #1663

ctool to simulate events for a given obsdef and model?

02/11/2016 01:37 PM - Deil Christoph

Status:	In Progress	**Start date:**	
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	30%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

We've been using this script to call ctobssim for a given list of HESS runs / IRFs and model:
https://github.com/gammapy/gammapy-extra/blob/master/datasets/hess-crab4/simulate_eventlists.py
https://gist.github.com/cdeil/54be2a0d9005b7123693

Now that ctools is flexible enough to work with our HDU index names, the copying to temp IRF files with renamed HDUs can be removed.

Also, I'd like to have this as an easy-to use command line tool:
https://github.com/gammapy/gammapy/issues/437

I just realised that this doesn't use Gammapy in any way, so I think it makes more sense as a cscript.

Do you think it's useful / general enough to add?

--

As a side note, possibly a separate issue, we noticed that the simulated times aren't time-ordered:
https://github.com/gammapy/gammapy/pull/415#issuecomment-180766342

Here's the output example file from that script where event times aren't in order:
https://github.com/gammapy/gammapy-extra/blob/master/datasets/hess-crab4/hess_events_simulated_023523.fits

I'd like to propose a requirement that event lists should be ordered in EVENT_ID and TIME soon, so that tools that care about time don't have to check or sort all the time.

Assuming you agree that time-ordered ctobssim output would be a good thing:
can you reproduce the issue?

History

#1 - 02/11/2016 02:06 PM - Knödlseder Jürgen

user#66 wrote:

> We've been using this script to call ctobssim for a given list of HESS runs / IRFs and model:
> https://github.com/gammapy/gammapy-extra/blob/master/datasets/hess-crab4/simulate_eventlists.py
> https://gist.github.com/cdeil/54be2a0d9005b7123693
>
> Now that ctools is flexible enough to work with our HDU index names, the copying to temp IRF files with renamed HDUs can be removed.
>
> Also, I'd like to have this as an easy-to use command line tool:
> https://github.com/gammapy/gammapy/issues/437
>
> I just realised that this doesn't use Gammapy in any way, so I think it makes more sense as a cscript.
>
> Do you think it's useful / general enough to add?

I'm not sure that I understand what exactly the tool should do besides running ctobssim? Can you clarify what exactly is needed?

> As a side note, possibly a separate issue, we noticed that the simulated times aren't time-ordered:
> https://github.com/gammapy/gammapy/pull/415#issuecomment-180766342

> Here's the output example file from that script where event times aren't in order:
> https://github.com/gammapy/gammapy-extra/blob/master/datasets/hess-crab4/hess_events_simulated_023523.fits
>
> I'd like to propose a requirement that event lists should be ordered in EVENT_ID and TIME soon, so that tools that care about time don't have to check or sort all the time.
>
> Assuming you agree that time-ordered ctobssim output would be a good thing:
> can you reproduce the issue?

The events are not time ordered because ctobssim simulates one model component after the other. One would need to add a sort method to arrange the events at the end (I was just lazy on doing this).

Is there a requirement for time ordering of events (I'm not sure whether this exists so far for pipelines and MC, probably need to check with Karl and Gernot). I know that is can become very tricky for real data to get the right time order as sometimes the event data are just written as they come, and if some event needs more time for processing than other events, it could well be that the events are not time ordered.

Do you have a Use Case where you need the event data to be time ordered?

#2 - 02/11/2016 06:37 PM - Deil Christoph

I've started to implement a script to do what I want here:
https://github.com/gammapy/gammapy/pull/437/files#diff-442dc758d43bfb5c43614ce475e40608R78

When I run it on a HESS input event list with 7613 rows I get this error!?

RuntimeError: *** ERROR in GFitsTable::insert(int, GFitsTableCol&): Column length (7613) is not compatible with the number of rows (7813) in the table

My goal is this:
Input: HESS event list (for run time, position, etc.) and IRFs, as well as a model xml file
Output: simulated event list for this model, IRFs and with the same parameters as the real run.

The point that's most unclear to me is whether I should give the real event list via the XML file and inobs,
or if this can't work properly for some reason (see error above) and I should not use such an XML file at all, but pass all parameters by hand (by extracting them from the events header and putting them in via the Python ctobssim interface)

!???

Jürgen or Michael, if you could have a look and help me with this?

#3 - 02/11/2016 06:47 PM - Mayer Michael

My goal is this:
Input: HESS event list (for run time, position, etc.) and IRFs, as well as a model xml file
Output: simulated event list for this model, IRFs and with the same parameters as the real run.

There shouldn't be a problem in using an input observation definition file with ctobssim. If you use a proper obs.xml and model.xml file, the output should be usable obs XML file in the end.
Could you provide the input XML file that I can check what could possibly go wrong?

Apart from that, I want to write documentations and howto about simulating IACT data very soon (#1647).

#4 - 02/11/2016 06:58 PM - Deil Christoph

I'm using this with the new HD-HAP example files that Alexander posted yesterday:

```
<observation_list title="observation library">
<observation name="Crab" id="23523" instrument="HESS">
<parameter name="EventList" file="run023400-023599/run023523/hess_events_023523.fits.gz"/>
<parameter name="EffectiveArea" file="run023400-023599/run023523/hess_aeff_2d_023523.fits.gz[AEFF_2D]"/>
<!--<parameter name="EnergyDispersion" file="run023400-023599/run023523/hess_edisp_2d_023523.fits.gz[EDISP_2D]"/>-->
<parameter name="PointSpreadFunction" file="run023400-023599/run023523/hess_psf_3gauss_023523.fits.gz[PSF_2D_GAUSS]"/>
<!--<parameter name="Background" file="hess_bkg_offruns_023523.fits.gz"/>-->
</observation>
</observation_list>

<source_library title="source library">
<source name="Crab" type="PointSource">
<spectrum type="PowerLaw">
<parameter name="Prefactor" scale="1e-16" value="5.7"  min="1e-07" max="1000.0" free="1"/>
<parameter name="Index"     scale="-1"    value="2.48" min="0.0"   max="+5.0"   free="1"/>
<parameter name="Scale"     scale="1e6"   value="0.3"  min="0.01"  max="1000.0" free="0"/>
</spectrum>
<spatialModel type="SkyDirFunction">
<parameter name="RA"  scale="1.0" value="83.6331" min="-360" max="360" free="0"/>
<parameter name="DEC" scale="1.0" value="22.0145" min="-90"  max="90"  free="0"/>
</spatialModel>
</source>
<!--<source name="CTABackgroundModel" type="CTAIrfBackground" instrument="HESS">-->
<!--<spectrum type="PowerLaw">    -->
<!--<parameter name="Prefactor" scale="1.0"  value="1.0"  min="1e-3" max="1e+3"   free="1"/>    -->
<!--<parameter name="Index"     scale="1.0"  value="0.0"  min="-5.0" max="+5.0"   free="1"/>    -->
<!--<parameter name="Scale"     scale="1e6"  value="1.0"  min="0.01" max="1000.0" free="0"/>    -->
<!--</spectrum>-->
<!--</source>    -->
</source_library>
```

If you have time to try and make a working example that uses HESS event list / IRFs as input to create a simulated event list, that would be very helpful.

#5 - 02/11/2016 08:12 PM - Knödlseder Jürgen

user#66 wrote:

> I've started to implement a script to do what I want here:
> https://github.com/gammapy/gammapy/pull/437/files#diff-442dc758d43bfb5c43614ce475e40608R78
>
> When I run it on a HESS input event list with 7613 rows I get this error!?
>
> RuntimeError: *** ERROR in GFitsTable::insert(int, GFitsTableCol&): Column length (7613) is not compatible with the number of rows (7813) in the table
>
> My goal is this:
> Input: HESS event list (for run time, position, etc.) and IRFs, as well as a model xml file
> Output: simulated event list for this model, IRFs and with the same parameters as the real run.
>
> The point that's most unclear to me is whether I should give the real event list via the XML file and inobs,
> or if this can't work properly for some reason (see error above) and I should not use such an XML file at all, but pass all parameters by hand (by extracting them from the events header and putting them in via the Python ctobssim interface)
>
> !???
>
> Jürgen or Michael, if you could have a look and help me with this?

Interesting, I never tried this. The ctobssim tools does not expect an event file on input, that's probably the reason why you get the error message. The tool can certainly be modified to drop any existing event list.

For the moment, ctobssim expects that there is no event file in the observation definition XML file and hence needs information on pointing, energy boundaries, GTI, RoI, and deadtime correction factor that normally is extracted from the event file. The format for this is (autogenerated by csobsdef):

```
<observation name="Crab" id="01" instrument="CTA">
  <parameter name="Pointing" ra="83.63" dec="22.01" />
  <parameter name="EnergyBoundaries" emin="100000" emax="1e+08" />
  <parameter name="GoodTimeIntervals" tmin="0" tmax="1800" />
  <parameter name="TimeReference" mjdrefi="51544" mjdreff="0.5" timeunit="s" timesys="TT" timeref="LOCAL" />
  <parameter name="RegionOfInterest" ra="83.63" dec="22.01" rad="5" />
  <parameter name="Deadtime" deadc="0.95" />
  <parameter name="Calibration" database="prod2" response="South_0.5h" />
</observation>
```

Now in your case I guess this information is in fact extracted from the event file, but at the step of writing the tool tries somehow to append a new event table to the old table. Could you just produce a simple test case so that I can reproduce that and fix the issue (by dropping the initial event information)?

#6 - 02/11/2016 08:41 PM - Knödlseder Jürgen

I tried to emulate the problem on my side but did not succeed, everything works. So I really would need your input files to make progress on that.

#7 - 02/11/2016 08:47 PM - Deil Christoph

I just sent you example files from HESS via email 10 seconds ago.

Thank you very much for helping me get this working ... ctobssim-simulated event lists are the next-best thing we can produce for tests / examples until a HESS data release, which will take some time.

#8 - 02/11/2016 10:59 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 20

I understand the problem. Your HESS event file has non-standard columns (e.g. BUNCH_ID) and ctobssim does not properly drop these columns from the event list but tries appending these columns upon writing.

The thing is that ctobssim does in fact not delete the old event list, but just appends additional events to an existing event list. But it only appends data to the standard columns, and leaves the additional FITS table columns untouched. This leads at the end to a mismatch between the columns of the FITS file.

We introduced this problem when we re-organised the GCTAEventList class (see #1600).

We should change ctobssim so that it never tries appending events to an existing event list, but always creates a new one.

#9 - 02/12/2016 12:52 AM - Knödlseder Jürgen

- % Done changed from 20 to 30

The problem should now be fixed. You need to get the devel branch from GammaLib and ctools (I made changes in both packages).

#10 - 02/12/2016 08:31 AM - Deil Christoph

Thanks!

I still could use some help to get the event simulation working ...

How can I set an IRF file like e.g. run023400-023599/run023523/hess_aeff_2d_023523.fits.gz[AEFF_2D] for a given observation?
Related, but different: https://github.com/ctools/ctools/blob/devel/examples/simulate_events.py#L90

So should passing the input event list directly to ctobssim work now?
Or should I write a helper function which creates an observation object and copy all relevant parameters over from the input observation list or event list?
https://github.com/ctools/ctools/blob/devel/examples/simulate_events.py#L39

#11 - 02/12/2016 08:35 AM - Deil Christoph

Also, I don't have cube background models yet for this data set.
Michael, I think you mentioned it's possible to use effective area for background modeling?
Can you please point me to an example how to declare this in the model XML file?

#12 - 02/12/2016 09:08 AM - Deil Christoph

PS: the obs index, HDU index and IRF files I'm using to try to implement this and test it is here:
https://github.com/gammapy/gammapy-extra/tree/master/datasets/hess-crab4-hd-hap-prod2

One thing you might notice is that we're exporting PSF in different formats, one of which is this one, because the analytical PSF models have given us grief:
http://gamma-astro-data-formats.readthedocs.org/en/latest/irfs/psf/psf_table/index.html

Is this format already supported by Gammalib or is someone working on this?
Or should I file a feature request?

#13 - 02/12/2016 09:44 AM - Mayer Michael

> Also, I don’t have cube background models yet for this data set.
> Michael, I think you mentioned it’s possible to use effective area for background modeling?
> Can you please point me to an example how to declare this in the model XML file?

Have a look at $GAMMALIB/inst/cta/test/cta_model_aeff_bgd.xml
I would recommend to create the XML files (model and obs) via csiactobs. The tool will automatically fall back to aeff-background in case no irf background is present.

> How can I set an IRF file like e.g. run023400-023599/run023523/hess_aeff_2d_023523.fits.gz[AEFF_2D] for a given observation?

I am not sure if I understand your question. But if you want to simulate e.g. 50h using a particular IRF, you should use csobs2caldb to create a database entry before running ctobssim. As I said, I will work on docs to run data analysis and simulations with IACT data very soon (#1646, #1647).

#14 - 02/12/2016 10:24 AM - Deil Christoph

I want to simulate data for a given HESS run (28 min) and IRFs.

So my question is: what's the equivalent Python script to the following XML obsdef dile to set up a GObservation pointing to the IRF files, (because I don't want to write temp XML files)?

<observation_list title=“observation library">
<observation name=“Crab” id=“23523” instrument=“HESS">
<parameter name=“EventList” file=“run023400-023599/run023523/hess_events_023523.fits.gz"/>
<parameter name=“EffectiveArea” file=“run023400-023599/run023523/hess_aeff_2d_023523.fits.gz[AEFF_2D]"/>
<!–<parameter name=“EnergyDispersion” file=“run023400-023599/run023523/hess_edisp_2d_023523.fits.gz[EDISP_2D]"/>–>
<parameter name=“PointSpreadFunction” file=“run023400-023599/run023523/hess_psf_3gauss_023523.fits.gz[PSF_2D_GAUSS]"/>
<!–<parameter name=“Background” file=“hess_bkg_offruns_023523.fits.gz"/>–>
</observation>
</observation_list>

#15 - 02/12/2016 10:43 AM - Mayer Michael

> So my question is: what’s the equivalent Python script to the following XML obsdef dile to set up a GObservation pointing to the IRF files, (because I don’t want to write temp XML files)?

You need an input runlist and an input model. To simulate the data, the following code snippet should help.

```
# Run csiactobs
iactobs = cscripts.csiactobs()
iactobs["datapath"] ="/to/your/data/" #where master.json is located
iactobs["prod name"] = "my-fits-production-name"
iactobs["infile"] = "myrunlist.lis" # ascii file of observation IDs
iactobs["inmodel"] = "mymodel.xml" # Just the sky models (background models get added automatically)
iactobs["bkgpars"] = 1
iactobs["outobs"] = "NONE"
iactobs["outmodel"] = "NONE"
iactobs["debug"] = True
iactobs.run()

# Run ctselect (apply energy thresholds, set RoI etc)
select = ctools.ctselect( iactobs.obs())
select["usepnt"] = True
select["usethres"] = "DEFAULT"
select["rad"] = 2.5
select["tmin"] = 0.0
select["tmax"] = 0.0
select["emin"] = 0.1
select["emax"] = 100.0
select["debug"] = True
select.run()

# Run ctobssim
sim = ctools.ctobssim(select.obs())
sim["debug"] = True
sim.run()

# Here is now a simulated observation container
sim_obs = sim.obs()
```

#16 - 02/12/2016 11:07 AM - Deil Christoph

Thanks, this helps.

I think this is is close to what I was looking for:
https://github.com/ctools/ctools/blob/devel/cscripts/csiactobs.py#L598

1. assign events and IRFs to observation
   obs.append(ev)
   obs.append(aeff)
   obs.append(psf)
   obs.append(edisp)
   obs.append(bck)

Do I have to wrap the IRF filenames in GXmlElement objects or can I set them more directly?

Is it necessary / recommended to call ctselect before ctobssim?

You do it in the example you posted above, but it's not done here:
https://github.com/ctools/ctools/blob/devel/examples/simulate_events.py#L153

I'll probably only be able to get back to this next week, so if some documentation on obs simulation becomes available, please let me know.

#17 - 02/12/2016 11:21 AM - Mayer Michael

> Do I have to wrap the IRF filenames in GXmlElement objects or can I set them more directly?

Don't know if there is a better way.

> Is it necessary / recommended to call ctselect before ctobssim?

I would say it is highly recommended for simulating IACT data. Otherwise you end up simulating events between at 100 GeV for a zenith angle of e.g. 45 degrees. Just simulating in the safe energy range saves time and we don't get into ranges where the IRFs might lack of MC statistics.

In simulate_events.py it is not necessary since we use a fixed energy range of all runs. If ctselect usethres=DEFAULT was applied we can have different energy thresholds per observation.

> I'll probably only be able to get back to this next week, so if some documentation on obs simulation becomes available, please let me know.

Sure.

#18 - 02/12/2016 03:50 PM - Knödlseder Jürgen

user#66 wrote:

> I want to simulate data for a given HESS run (28 min) and IRFs.
>
> So my question is: what's the equivalent Python script to the following XML obsdef dile to set up a GObservation pointing to the IRF files, (because I don't want to write temp XML files)?
>
> <observation_list title=“observation library">
> <observation name=“Crab” id=“23523” instrument=“HESS">
> <parameter name=“EventList” file=“run023400-023599/run023523/hess_events_023523.fits.gz"/>
> <parameter name=“EffectiveArea” file=“run023400-023599/run023523/hess_aeff_2d_023523.fits.gz[AEFF_2D]"/>
> <!–<parameter name=“EnergyDispersion” file=“run023400-023599/run023523/hess_edisp_2d_023523.fits.gz[EDISP_2D]"/>–>
> <parameter name=“PointSpreadFunction” file=“run023400-023599/run023523/hess_psf_3gauss_023523.fits.gz[PSF_2D_GAUSS]"/>
> <!–<parameter name=“Background” file=“hess_bkg_offruns_023523.fits.gz"/>–>
> </observation>
> </observation_list>

Something like this should work (have not actually tried out):

```
irf = GCTAResponseIrf()
irf.load_aeff("run023400-023599/run023523/hess_aeff_2d_023523.fits.gz[AEFF_2D]")
irf.load_psf("run023400-023599/run023523/hess_psf_3gauss_023523.fits.gz[PSF_2D_GAUSS]")
irf.load_edisp("run023400-023599/run023523/hess_edisp_2d_023523.fits.gz[EDISP_2D]")
irf.load_background("hess_bkg_offruns_023523.fits.gz")
run = GCTAObservation()
run.load("run023400-023599/run023523/hess_events_023523.fits.gz")
run.response(irf)
obs = GObservations()
obs.append(run)
sim = ctools.ctobssim(obs)
```