ctools - Feature #3668

add minimal event counts to cssens

05/11/2021 05:30 PM - Sadeh Iftach

Status:	Closed	**Start date:**	05/11/2021
Priority:	Normal	**Due date:**	
Assigned To:	Sadeh Iftach	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2.0.0		

Description

For non-ctools sensitivity calculation, it is the standard to apply the following cuts on the test source, in addition to the 5sigma requirement:

1. demand a minimal event count for the test source of 10 events
2. demand that the test source counts are >= 5% of the background counts

This pull request adds these as optional settings, controlled via these new user parameters:

```
min_src_events,         i, h, -1,,, "minimal number of source counts"
non_source_uncert_frac, r, h, -1,,, "fraction of no-source counts serving as minimal threshold for source counts"
non_source_uncert_rad,  r, h, 0.33,,, "radius for circular region in which to integrate no-source counts for threshold"
```

where the default negative values mean that each cut is NOT applied.
For instance, the user will specify

```
min_src_events = 10
non_source_uncert_frac = 0.05
```

to apply the exact cuts defined above.

The parameter

```
non_source_uncert_rad
```

defines the radius of integration around the position of the test source in which to integrate counts not of source (eg background and maybe foreground sources), which is used for condition (2) above.
This is needed, in order to mimic the expected behavior for eg a Li&Ma calculation, where the corresponding region of interest for background integration is much smaller than the entire FoV.
The default value of 0.33 deg is chosen as the upper bound of the standard cut. In general an energy dependent value is needed, but this cut is only important for low energies (and long observation times), and so the chosen default is probably sensible.

The questions remains, of whether the two conditions should be applied by default or not.

The implementation includes a new function

```
_sim_evt_excess
```

which generates a collection of realizations of source and non-source counts, and takes the respective median values for the final cut.

The cuts are applied **after** the nominal test source flux is derived. In practice, the effect is to leave the final test flux as it is, or to incrementally increase it, until the two conditions are fulfilled.

This pull request also makes two small modifications for numerical safety:

1. make sure that the prefactor does not exceed the defined limit in for the test model:

```
        models        = test_model.copy()
        prefactor     = modutils.normalisation_parameter(models[self._srcname])
        crab_prefactor = prefactor.value() * crab_unit

        val_margin = 0.01
        min_pref = prefactor.min()
        max_pref = prefactor.max()

        val_now = crab_prefactor * test_crab_flux
        val_now = max(val_now, min_pref * (1 + val_margin))
        val_now = min(val_now, max_pref * (1 - val_margin))
        crab_prefactor = val_now / test_crab_flux

        prefactor.value(val_now)
```

- Open question : do we also want to add some warning here that the limit should be changed and the calculation redone? What is the standard for ctools on this?

2. prevent division by zero as part of

```
_predict_flux
```

by comparing the a threshold of 1e-10:

```
        rxy_norm = (mean_xx - mean_x * mean_x) * (mean_yy - mean_y * mean_y)
        if rxy_norm < 1e-10:
            rxy_norm = 1
        else:
            rxy_norm = math.sqrt(rxy_norm)

        rxy = (mean_xy - mean_x * mean_y) / rxy_norm

        regcoeff = rxy*rxy
```

Related issues:

Related to GammaLib - Action # 3693: Add statistics methods to GNdarray	**New**	**05/20/2021**
Duplicates ctools - Feature # 2130: Incorporate minimum counts to sensitivity...	**Closed**	

History

#1 - 05/11/2021 05:32 PM - Sadeh Iftach

Implemented here:
https://cta-gitlab.irap.omp.eu/sadeh/ctools/tree/cssens_source_min_event_cuts

#2 - 05/14/2021 05:47 PM - Sadeh Iftach

And yet another new feature in this pull request:
- in case edisp is set, only start using it in practice once the test flux is close to convergence.
This saves considerable CPU time.

Specifically, the modification follows (as part of the source flux calculation loop):

```
# if the flux has converged, check if the original value of
# edisp was set, and is not what has fo far been used
if edisp_orig and not self['edisp'].boolean():
```

```
    # set edisp on and continue the calculation
    self['edisp'] = True

    value = 'set edisp on after initial convergence withouth it'
    self._log_value(gammalib.TERSE, 'Converged result', value)
else:
    # finish the calculation after convergence
    break
```

This comes instead of directly breaking at this point of the calculation (after convergence).

#3 - 05/19/2021 05:36 PM - Knödlseder Jürgen

- Target version set to 2.0.0

Thanks for proposing these changes. I will look to them as soon as possible.

I justed wanted to make you aware that I worked the last days on a old features which is switching all ASCII file output to FITS files, see #1707. This will affect cssens, hence I will first finished this change and then merge in your changes.

#4 - 05/19/2021 05:37 PM - Knödlseder Jürgen

- Duplicates Feature #2130: Incorporate minimum counts to sensitivity calculation added

#5 - 05/19/2021 05:38 PM - Knödlseder Jürgen

- Status changed from In Progress to Closed

A duplicated feature was created with issue #3668. Since some code changes were done on this feature I close this issue now.

#6 - 05/19/2021 06:00 PM - Sadeh Iftach

- % Done changed from 70 to 100

Hi Jürgen,

I'm not sure I follow. Which duplicated issue are you referring to?
I'd like to keep track.

In any case, once you go through your other updates, please check out
https://cta-gitlab.irap.omp.eu/sadeh/ctools/tree/cssens_source_min_event_cuts

To summarize all changes:
1. Add optional absolute minimal event count on test source
2. Add optional minimal threshold of test source counts with respect to non-source counts
3. Numerical stability updates for source prefactor and for the _predict_flux regression function
4. Avoid extremely slow unnecessary energy dispersion calculations
5. Add optional random seed initialization for event generation

I know this is a lot for a single pull request. I hope you can integrate the content.

#7 - 05/19/2021 10:57 PM - Knödlseder Jürgen

user#374 wrote:

> Hi Jürgen,
>
> I'm not sure I follow. Which duplicated issue are you referring to?
> I'd like to keep track.

This was feature #2130 that was there since a while asking to incorporate minimum counts into the sensitivity calculation. Since no one was working on that I closed this feature now and keep this one alive.

> In any case, once you go through your other updates, please check out
> https://cta-gitlab.irap.omp.eu/sadeh/ctools/tree/cssens_source_min_event_cuts
>
> To summarize all changes:
> 1. Add optional absolute minimal event count on test source
> 2. Add optional minimal threshold of test source counts with respect to non-source counts
> 3. Numerical stability updates for source prefactor and for the _predict_flux regression function
> 4. Avoid extremely slow unnecessary energy dispersion calculations
> 5. Add optional random seed initialization for event generation
>
> I know this is a lot for a single pull request. I hope you can integrate the content.

No worries, I will go over the changes. I probably will propose shorter names for the new parameters as they are rather long compared to typical ctools names and a bit cryptic.

#8 - 05/20/2021 08:50 AM - Knödlseder Jürgen

- Status changed from Closed to Pull request

- % Done changed from 100 to 80

I had a first look over the code. I have seen that you use the statistics Python module that is only available from Python 3.4 on, which breaks the compatibility of ctools with older Python versions.

I see that you use the module only for the median() function, hence we should implement this function natively in the code so that we do not need the dependency.

#9 - 05/20/2021 08:51 AM - Knödlseder Jürgen

user#3 wrote:

> A duplicated feature was created with issue #3668. Since some code changes were done on this feature I close this issue now.

Closing was in fact not my intention here, sorry for the confusion. I closed the duplicated feature.

#10 - 05/20/2021 09:14 AM - Knödlseder Jürgen

I implemented the following code to replace the median function:

```
    def _median(self, array):
        """
        Compute median value for an array

        Parameters
        ----------
        array : list of floats
            Array

        Returns
        -------
        median : float
            Median value of array
        """
        # Initialise median
        median = 0.0

        # Get number of elements in array
        n = len(array)

        # Continue only if there are elements in the array
        if n > 0:

            # Sort the array
            wrk_array = sorted(array)

            # Get median
            if n % 2 != 0:
                median = wrk_array[int(n/2)]
            else:
                median = (wrk_array[int((n-1)/2)] + wrk_array[int(n/2)]) / 2.0

        # Return median
        return median
```

Testing for an empty array gives:

```
0.0
Traceback (most recent call last):
  File "./test.py", line 19, in <module>
    print(statistics.median(array))
  File "/Users/jurgen/anaconda3/envs/ctools-1.7.4/lib/python3.8/statistics.py", line 430, in median
    raise StatisticsError("no median for empty data")
statistics.StatisticsError: no median for empty data
```

where the first result is my code and the second the code from the statistics module.

Here a few tests with different arrays:

array	cssens._median	statistics.median
[1]	1	1
[2,1]	1.5	1.5
[2.0,1.0]	1.5	1.5
[2.0,3.0,1.0,5.0,1.0]	2.0	2.0
[1,2,3,4, 10,11,12,13,9,8,20,21]	9.5	9.5

The code seems to work as expected.

#11 - 05/20/2021 10:17 AM - Tibaldo Luigi

Median is a very low-level function. Shouldn't it be implemented somewhere else so that it can be used by other ctools/cscripts in the future? For example GMath or GNdarray?

#12 - 05/20/2021 10:52 AM - Knödlseder Jürgen

- Related to Action #3693: Add statistics methods to GNdarray added

#13 - 05/20/2021 10:53 AM - Knödlseder Jürgen

user#266 wrote:

> Median is a very low-level function. Shouldn't it be implemented somewhere else so that it can be used by other ctools/cscripts in the future? For example GMath or GNdarray?

Fully agree. The best place would probably be the GNdarray class, which should then also get mean() and std() methods. I created an issue for that: #3693.

#14 - 05/20/2021 12:38 PM - Sadeh Iftach

Thanks, Jürgen.

Of course, please feel free to make any changes to variable names etc. to fit the ctools conventions.
At the moment I understand that I have no action items for myself, so please let me know if you'd like me to modify any code.

An outstanding open question is the choice of default settings.
In my opinion, the minimal 10 evt source count can be the new default (rather than the current default of 0). This complies with what everyone expects with respect to the "official" cta results (produced with EventDisplay), and shown on the public website.

The second condition on the background counts is not relevant in most cases (except for very long exposures). In my opinion it is also less rigorously defined, especially considering the full-enclosure IRFs.
I'm agnostic about wether or not this one should be turned on by default.

#15 - 05/20/2021 01:12 PM - Knödlseder Jürgen

I agree that setting the default value to 10 is maybe what we want to do, although Gernot mentioned recently that these criteria may be revisited.

I'm basically done with the code integration, I did a bit of refactoring (introducing a method cssens._simulate_events that avoids having a loop over source and background in cssens._sim_evt_excess), modified slightly the logging (so that values all fit in the expected length and that detailed information is only written when chatter is in EXPLICIT) and changed the parameters to shorter names:

```
mincounts, i, h, 0,,, "Minimum number of source counts"
bkgexcess, r, h, 0.0,,, "Background uncertainty fraction"
bkgrad,    r, h, 0.33,,, "Background radius (deg)"
seed,      i, h, 1,,, "Initial random number generator seed"
```

I also updated the reference manual to add the new parameters. The documentation of all tools is in doc/source/users/reference_manual/ which is also used to build the help text that you get with cssens --help.

#16 - 05/20/2021 02:20 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

- % Done changed from 80 to 100

I merged the code into devel. Close the issue now.

#17 - 05/20/2021 04:18 PM - Sadeh Iftach

Great!

One final note, not really critical...
I tried to explicitly use "non-source" rather than "background" terminology for

```
bkgexcess, r, h, 0.0,,, "Background uncertainty fraction"
bkgrad,    r, h, 0.33,,, "Background radius (deg)"
```

This may be a bit misleading for the user, since the cut is done based on all background + [not test source]-sources in the RoI (not exclusively the IRF background).

#18 - 05/20/2021 04:23 PM - Knödlseder Jürgen

I understand, but I don't think that a user will understand what non-source means. We may clarify in the documentation that "background" means everything else than the source (a bit like for the DM community).

#19 - 05/20/2021 04:25 PM - Sadeh Iftach

Great.
Thanks again for the quick integration of this pull request.