ctools - Support #1925

ctools flux points and upper limits

02/09/2017 04:40 PM - Kelley-Hoskins Nathan

Status:	New	**Start date:**	02/09/2017
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

I'm trying to calculate flux and upper limit points for a point source in a gobservation. I can get some flux points, but I have two problems.

a) Several flux points have errors larger than the flux value, meaning the lower error bound is unphysically negative. However, the bins' test statistics are quite high.

flux erg/cm2/s	flux_err erg/cm2/s	NEvents	TS
1.49e-10	2.26e-10	2605	193.9
2.88e-10	4.75e-10	892	298.9

(values are from the flux and flux_err variables in the snippet of code below)

Does this mean there was a problem with the likelihood setup or calculation?

Theres a general consensus that Li&Ma flux points are only statistically valid/useful if their bin significance is above a certain threshold (~3-5 sigma), but I'm not sure what the ctools/likelihood protocol is. I would guess switch to an upper limit when TS < 9, but that doesn't apply for these points.

Due to their large errors, should I replace the above flux points with upper limits?

b) on others flux calculations, ctulimit throws an error:

```
ValueError: *** ERROR in ctlikelihood::evaluate(GModelPar&, double&): Invalid value.
Value 3.45239631834042e-21 of parameter "Prefactor" is above its maximum boundary 0.
To omit this error please raise the maximum parameter boundary.
```

The code that produces that error is this (adapted from csspec.py):

```
    # earlier stuff to assemble gobservation 'obs' and to select the events that are only in our energy bin

    # run test likelihood computation
    print('running ctlike...')
    like = ctools.ctlike( obs )
    like['debug'  ] = True
    like['chatter'] = 4
    like['clobber'] = True
    print('warning, disabling energy dispersion in likelihood calculation...')
    like['edisp'  ] = False
    like.run()

    # run upper limit calculation tool
    confidence_interval = 0.95
    elogmean  = gammalib.GEnergy( pow( 10, ( math.log10( self['energy_min_TeV'] ) + math.log10( self['energy_max_TeV'] ) ) / 2 ) ,
'TeV' )
    elogmean2 = elogmean.MeV() * elogmean.MeV()
    if like.obs().models()[source_name].spectral()['Prefactor'].value() == 0.0 :
```

```
    print('source has a flux prefactor of 0.0, unable to run ctulimit...')
    ulimit_value = -1.0
  else :
    print('running ctools.ulimit...')
    ulimit = ctools.ctulimit( like.obs () )
    ulimit['debug'     ] = True
    ulimit['chatter'   ] = 4
    ulimit['srcname'   ] = source_name
    ulimit['eref'      ] = elogmean.TeV()
    ulimit['sigma_min' ] =  0.1
    ulimit['sigma_max' ] = 10.0
    ulimit['confidence'] = confidence_interval

    # error occurs during this call to run()
    ulimit.run()
    ulimit_value = ulimit.diff_ulimit()

   # calculate the wanted output data
  print('calculating output data...')
  source      = like.obs().models()[source_name]
  if like.obs().logL() == 0.0 :
    flux       = 0.0
    flux_err   = 0.0
    ts_value   = 0.0
    ulim_value = 0.0
    npred      = 0.0
  else :
    # calculations transcribed from $CTOOLS/cscripts/csspec.py
    fitted_flux = source.spectral().eval(elogmean) # 1/s/cm2/MeV
    parvalue    = source.spectral()[0].value()
    rel_error   = source.spectral()[0].error() / parvalue if parvalue != 0 else 0.0
    e_flux      = fitted_flux * rel_error

    flux       = fitted_flux * elogmean2 * gammalib.MeV2erg
    flux_err   = e_flux      * elogmean2 * gammalib.MeV2erg
    ts_value   = source.ts()
    ulim_value = ulimit_value * elogmean2 * gammalibMeV2erg if ulimit_value > 0.0 else ulimit_value
    npred      = like.obs().npred()
```

I dug around in the code to figure out that the error occurs because in the ctulimit::run() function, the m_model_par->factor_max() is zero,
which is limiting the Initial Parameter Range, but I don't know how that parameter got to be zero, or where to set it to be larger.

Does anyone know whats causing this error?

History

#1 - 04/05/2017 10:23 AM - Knödlseder Jürgen

user#111 wrote:

> I'm trying to calculate flux and upper limit points for a point source in a gobservation. I can get some flux points, but I have two problems.
>
> a) Several flux points have errors larger than the flux value, meaning the lower error bound is unphysically negative. However, the bins' test statistics are quite high.
>
> [...]
> (values are from the flux and flux_err variables in the snippet of code below)
>
> Does this mean there was a problem with the likelihood setup or calculation?
>
> Theres a general consensus that Li&Ma flux points are only statistically valid/useful if their bin significance is above a certain threshold (~3-5 sigma), but I'm not sure what the ctools/likelihood protocol is. I would guess switch to an upper limit when TS < 9, but that doesn't apply for these points.
>
> Due to their large errors, should I replace the above flux points with upper limits?

This may depend a bit on how the model fit was setup, and I would need more information to judge this. If you have for example a power law with a free spectral index, the flux error contains the combined uncertainty in the flux and the spectral index. If the spectral index is poorly constrained this may lead to a relatively large flux error, although the source is significantly detected. If you want to compare flux errors to the significance you have to fix all other spectral parameters.

> b) on others flux calculations, ctulimit throws an error:
>
> [...]
>
> The code that produces that error is this (adapted from csspec.py):
> [...]
>
> I dug around in the code to figure out that the error occurs because in the ctulimit::run() function, the m_model_par->factor_max() is zero, which is limiting the Initial Parameter Range, but I don't know how that parameter got to be zero, or where to set it to be larger.
>
> Does anyone know whats causing this error?

There was indeed a bug in ctulimit for parameters that had no limits. This was corrected. The code is now:

```
// Compute parameter bracketing
double parmin = value - m_sigma_min * error;
double parmax = value + m_sigma_max * error;
if (m_model_par->has_min() && m_model_par->factor_min() > parmin) {
    parmin = m_model_par->factor_min();
}
if (m_model_par->has_max() && m_model_par->factor_max() < parmax) {
    parmax = m_model_par->factor_max();
}
```

Files

test.log	75.9 KB	02/09/2017	Kelley-Hoskins Nathan