ctools - Feature #1512

in ctbutterfly, methods to return energy/flux/errors in memory, without writing/reading butterfly.txt

07/15/2015 07:10 PM - Kelley-Hoskins Nathan

Status:	Closed	**Start date:**	07/15/2015
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.5.0		

Description

In ctbutterfly::save(), you can write the energies, fluxes, and errors of your spectral fit to a file. But, in a python pipeline with ctools.ctbutterfly, there's no other way to access the energies/fluxes/errors without saving the results to a file, then reading them back in and parsing them again.

So, I propose 4 new methods for ctbutterfly:

```
vector<float> ctbutterfly::energies() {
  return m_energies ;
}
vector<float> ctbutterfly::fluxes() {
  return m_fluxes ;
}
vector<float> ctbutterfly::errors() {
  return m_errors ;
}
??? ctbutterfly::covariance_matrix() {
  return m_covariance ; // ???
}
```

Each return a vector of floats for the (logarithmically center) energies, fluxes, and errors. Then, someone can do:

```
obs = GObservations()
# load observations into obs

emin = gammalib.GEnergy( 100, 'GeV' )
emax = gammalib.GEnergy( 100, 'TeV' )

spec = ctools.ctbutterfly( obs )
spec['srcname'].string( 'Crab' )
spec['emin'   ].real( emin.TeV() )
spec['emax'   ].real( emax.TeV() )
spec['enumbins'].integer(100)
spec.run()

# yay!
en   = spec.energies()
flux = spec.fluxes()
errs = spec.errors()
covar = spec.covariance_matrix()

# carry on with storage and plotting
```

In addition, letting the covariance matrix also be readable in python would be nice too. I'm not certain a GMatrixSparse gets converted to in python. Maybe a full matrix, like a list of lists?

History

#1 - 07/16/2015 01:33 PM - Kelley-Hoskins Nathan

Actually, for the energies() function, it might be nicer to return a list of GEnergy() objects, rather than a list of floats.

#2 - 07/17/2015 10:59 AM - Mayer Michael

Sounds like a good idea. I think the main obstacle is, in the Python interface, to convert the std::vector<float> to a python list. I don't know if this can be done easily. One could also think about moving away from std::vector to a GVector.
Regarding the covariance matrix: The GMatrixSparse object can be accessed via python, similar as every other gammalib class. The problem is that we don't have load() and save() methods for the matrix.

#3 - 07/17/2015 11:04 AM - Knödlseder Jürgen

Mayer Michael wrote:

> Sounds like a good idea. I think the main obstacle is, in the Python interface, to convert the std::vector<float> to a python list. I don't know if this can be done easily.

This can be easily done by adding

```
%include "std_vector.i"
namespace std {
   %template(FloatVector) vector<float>;
}
```

to the .i file (see GNodeArray for an example).

#4 - 07/17/2015 11:05 AM - Knödlseder Jürgen

Kelley-Hoskins Nathan wrote:

> Actually, for the energies() function, it might be nicer to return a list of GEnergy() objects, rather than a list of floats.

We actually have GEnergies which is a container of GEnergy objects. So I would just return that object.

#5 - 06/21/2016 09:50 PM - Knödlseder Jürgen

- Target version set to 1.2.0

#6 - 03/03/2017 10:34 AM - Knödlseder Jürgen

- Target version changed from 1.2.0 to 1.3.0

#7 - 06/07/2017 05:45 PM - Knödlseder Jürgen

- Target version changed from 1.3.0 to 1.4.0

#8 - 08/01/2017 09:49 AM - Knödlseder Jürgen

- Target version changed from 1.4.0 to 1.5.0

#9 - 01/23/2018 10:52 AM - Knödlseder Jürgen

- Status changed from New to In Progress

- Assigned To set to Knödlseder Jürgen

- % Done changed from 0 to 10

The covariance matrix can be accessed using

```
>>> butterfly=ctools.ctbutterfly()
>>> print(butterfly.obs().function().covariance())
=== GMatrixSparse ===
 Number of rows .....: 0
 Number of columns .....: 0
 Number of nonzero elements : 0
 Number of allocated cells .: 0
 Memory block size .....: 512
 Sparse matrix fill .....: 0
 Pending element .....: 0
 Fill stack size .....: 0 (none)
```

Since the file that is written is a CSV file I think that the best option is to add a method that returns a GCsv object, making the interface simpler. This means, however, that a user has to know the structure of the GCsv object.

#10 - 01/23/2018 11:32 AM - Knödlseder Jürgen

- Status changed from In Progress to Pull request

- % Done changed from 10 to 100

I added a ctbutterfly::butterfly() method that returns the computation results in form of a GCsv object.

#11 - 01/23/2018 12:24 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

Merged into devel.