GammaLib - Change request #832

Remove _swigregister entries in gammalib Python package

04/11/2013 09:48 PM - Deil Christoph

Status:	Closed	**Start date:**	04/11/2013
Priority:	Normal	**Due date:**	
Assigned To:	Deil Christoph	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	00-08-00		

Description

As already noted in issue #484, the gammalib namespace is polluted with a duplicate gammalib.G<Class>_swigregister for every gammalib.G<Class>.

Is it possible to get rid of this so that interactive work (tab completion at the IPython prompt) becomes more pleasant?
(I couldn't find a solution on the web ... if you don't either we can ask on the Swig mailing list.)

Other minor things I noticed:

- After import gammalib there is a self-reference, i.e. gammalib and gammalib.gammalib and gammalib.gammalib.gammalib are all references to the gammalib module.
  It doesn't seem to cause any problems, but if it can be cleaned up that would be nice.
- There's gammalib.SwigPyIterator, gammalib.VecDouble and gammalib.vectori. Do they have any purpose? Can they be removed?

from gammalib import * is considered bad practice ... I plan to write the tutorial examples using import gammalib instead.

History

#1 - 04/11/2013 10:25 PM - Knödlseder Jürgen

I googled around some time ago, and it does not seem to be possible to get rid of it, that's a swig feature.

Also the self referecing is a swig side effect ...

But you don't see this if you use

```
from gammalib import *
```

instead of

```
import gammalib
```

I know this is bad practice (for some reason), but I hate namespaces anyways as they add an unnecessary burden to find out in which namespace a class exists. That's why I write all my example using the from ... syntax.

So the real question is: should we just follow a rule because people say it's bad practice? What are the real arguments behind not using from gammalib import *?

#2 - 04/11/2013 11:02 PM - Deil Christoph

OK, I"ll give it a shot convincing you.

The main reason is that this is hard to read because you don't know if foo is from package_a or package_b:

```
from package_a import *
from package_b import *

foo()
```

In the worst case you wanted to call package_a.foo() but from package_b import * re-defined the foo name in your local namespace.
So the order of the import statements matters and by inserting such an import somewhere you can break code in very hard to debug ways.

In contrast here it is always clear what is going on:

```
import package_a
import package_b

package_a.foo()
```

and this is also fine:

```
from package_a import foo, bar, baz
import package_b

foo()
bar()
```

Now I agree this problem doesn't really exist for gammalib, because all class names start with a G and it is easy to see in a script which names are from gammalib and the chance for a name collision with other things in the local namespace is small.

But as I've already mentioned in issue #558, my suggestion would be to properly use the C++ and Python namespace feature and in C++ write

```
#include <gamma.h>
gamma::Models model()
```

and in Python

```
import gamma
model = gamma.Models()
```

Note that this question of whether to use namespaces at all is completely independent of the question whether the library should be flat (like numpy) or structured into sub-modules (like scipy).

The issue of name collisions will become more apparent when you add functions to the gammalib C++ and Python API.
Actually, can you add one function to the C++ and Python API, so that I can see how it appears in the Doxygen docs
and the Swig Python wrapper?
How about the Li & Ma formula to compute the significance of an on / off observation?
Would that be g_li_ma_significance or GLiMaSignificance?

```
import numpy as np
def significance_lima(on, off, alpha):
    """Compute significance according to the Li & Ma formula.
    1983ApJ...272..317L, equation (17)"""

    # Precompute some quantities that occur more than once
    # in the Li & Ma formula
    sum = on + off
    temp = (alpha + 1) / sum
    l = on * np.log(on * temp / alpha)
```

```
    m = off * np.log(off * temp)
    sign = np.where(on - alpha * off > 0, 1, -1)

    # Compute significance
    return sign * np.sqrt(np.abs(2 * (l + m)))
```

#3 - 04/12/2013 07:39 AM - Knödlseder Jürgen

Christoph Deil wrote:

> OK, I"ll give it a shot convincing you.
>
> The main reason is that this is hard to read because you don't know if foo is from package_a or package_b:
> [...]
>
> In the worst case you wanted to call package_a.foo() but from package_b import * re-defined the foo name in your local namespace.
> So the order of the import statements matters and by inserting such an import somewhere you can break code in very hard to debug ways.
>
> In contrast here it is always clear what is going on:
> [...]
> and this is also fine:
> [...]
>
> Now I agree this problem doesn't really exist for gammalib, because all class names start with a G and it is easy to see in a script which names are from gammalib and the chance for a name collision with other things in the local namespace is small.

Exactly, that was the whole point of having all GammaLib classes defined as GFoo. Namespaces are helpful in case someone choses common names, if not they are useless and even error prone.

> But as I've already mentioned in issue #558, my suggestion would be to properly use the C++ and Python namespace feature and in C++ write
> #include <gamma.h>
> gamma::Models model()
> and in Python
> import gamma
> model = gamma.Models()

This makes the code just longer, there is no gain at all.

> Note that this question of whether to use namespaces at all is completely independent of the question whether the library should be flat (like numpy) or structured into sub-modules (like scipy).
>
> The issue of name collisions will become more apparent when you add functions to the gammalib C++ and Python API.
> Actually, can you add one function to the C++ and Python API, so that I can see how it appears in the Doxygen docs
> and the Swig Python wrapper?
> How about the Li & Ma formula to compute the significance of an on / off observation?
> Would that be g_li_ma_significance or GLiMaSignificance?
> [...]

There are already a functions in GammaLib. See GTools.cpp and GNumerics. For the moment they just have common names.

#4 - 04/12/2013 08:12 AM - Knödlseder Jürgen

Okay, I see my last point strengthens your argument. So we need a namespace for the functions, and I guess once you have it for the function you also want to have it for the classes.

But even if we introduce namespace, we should aim to find names that do not conflict with (at least) any standard names, so that 99% of code written with

using gamma;

will work (I really hat the long names that will make it impossible to have a 80 character length page size code).

#5 - 04/12/2013 08:16 AM - Knödlseder Jürgen

Thinking once more about it, do we really need the namespace for the classes, as the GFoo names make them probably very unique? Limiting the namespace to the functions would allow to keep the code short and readable, avoiding at the same time the need for having

using gamma;

which then would mean that functions are always access using the namespace.

#6 - 04/12/2013 09:56 AM - Deil Christoph

I think gamma as a namespace name is acceptably short and sweet (5 characters plus the . in Python and the :: in C++):

```
#include <gamma.h>
double significance = gamma::significance(42, 43, 0.2);
gamma::Models models();
```

```
import gamma
significance = gamma.significance(42, 43, 0.2)
models = gamma.Models()
```

Import conventions to shorten namespace names and line length could also work, e.g. numpy uses

```
import numpy as no
result = np.sin(42)
```

and gammalib could use

```
import gamma as G
significance = G.significance(42, 43, 0.2)
models = G.Models()
```

and

```
#include <gamma.h>
using G = gamma;
double significance = G::significance(42, 43, 0.2);
G::Models models();
```

But as you say, using a G prefix for everything (or only for classes and put functions and constants in a namespace) will probably work as well, because there will be no users that use GammaLib together with other libraries that have conflicting names or write code where they choose a variable, function or class name that conflicts with a class name in GammaLib.

Btw.: sorry for starting such a big fundamental change in this technical thread on how to clean up the contents of the Python gammalib package. I think the things described in the original description (user shouldn't see _swigwrapper, gammalib.gammalib, gammalib.SwigPyIterator, gammalib.VecDouble and gammalib.vectori) is still a valid feature request for improvement ... unfortunately I don't know how to do it.

#7 - 04/12/2013 11:26 AM - Knödlseder Jürgen

Well, the only real reason for using namespaces is to avoid naming conflicts, and even namespaces do not achieve to meet this goal 100% (see for example http://www.jelovic.com/articles/using_namespaces.htm).

GammaLib tries in a first place to avoid naming conflicts using the GFoo logic. I think this is quite safe. In particular because GammaLib is self contained, so for a large fraction of applications, GammaLib is the only thing you need (that was one of the main goals behind setting up the library).

The only place where namespace conflicts can occur are functions, and I think it's in fact a good choice to use the gamma namespace for the functions.

In fact, one of the dangers of namespaces is that people will just prepend using namespace and then code as if there was no namespace. You find many references to this when you google around. So I consider that it's at the end safer to not enforce namespace using but to find unique names.

#8 - 01/13/2014 04:15 PM - Deil Christoph

- Status changed from New to Pull request

- Assigned To set to Deil Christoph

This issue got completely sidetracked by the discussion if `from gammalib import *` or `import gammalib` should be taught in the examples / docs.

The original issue was to remove the _swigregister entries from the Python wrapper.

Simply deleting those symbols from the gammalib namespace on import seems to work:
https://github.com/cdeil/gammalib/compare/issue_0832

I only tested this with Python 2.7 on Mac ... it should be tested with Python 3 and on Linux also.

#9 - 01/13/2014 05:40 PM - Knödlseder Jürgen

- % Done changed from 0 to 90

Thanks for this. I added also a

del gammalib.gammalib

to remove the self inclusion of gammalib. Pushed into integration to see whether there is any problem on a platform.

#10 - 01/13/2014 06:56 PM - Knödlseder Jürgen

- Status changed from Pull request to Feedback

- Target version set to 00-08-00

- % Done changed from 90 to 100

Pushed into devel, works on all continuous integration platforms.

#11 - 01/13/2014 08:55 PM - Deil Christoph

Hey ... I think you stole my commit :-)
https://github.com/cdeil/gammalib/commit/6c9529b91b1dc90b4365ce5bb42f48ff2596412a
https://github.com/gammalib/gammalib/commit/ba4dfdf7c684d81c100e3f8ddfb62c48206d02d8

It doesn't matter for this small contribution, but in general integration managers should only git merge and not git rebase contributions.

#12 - 01/13/2014 09:54 PM - Knödlseder Jürgen

Not stolen the way you though smile.png As I could not find the relevant branch I just typed in the couple of lines myself ... now I can see the branch, it was not there when I looked earlier ...

#13 - 02/17/2014 10:19 PM - Knödlseder Jürgen

- Status changed from Feedback to Closed