ctools - Bug #1422

Parameter checking conflicts with ctselect

02/13/2015 06:47 PM - Mayer Michael

Status:	Closed	**Start date:**	02/13/2015
Priority:	Urgent	**Due date:**	
Assigned To:	Mayer Michael	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0		

Description

The new functionality to check each GApplicationPar for sanity results in a conflict in ctselect:
In case the user wants to omit the RoI selection, a simple "-1" could be given for "ra", "dec" and "rad" parameters. Now, this throws an exception since, e.g. the radius is not in the requested range (0-90).
We should maybe think about changing the values for "no selection" to 0.0 for all these parameters?

History

#1 - 02/13/2015 06:52 PM - Knödlseder Jürgen

There is the possibility to pass INDEF, NONE, UNDEF or UNDEFINED as a real parameter and gammalib will set the status of these parameters to ST_UNDEFINED (can be checked using GApplicationPar::is_undefined()). We should use this functionality instead of setting RA and DEC to -1, which formally is not really an invalid coordinate.

#2 - 02/13/2015 06:53 PM - Knödlseder Jürgen

- Target version set to 1.0.0

#3 - 02/13/2015 06:59 PM - Mayer Michael

I agree, it would be better to pass UNDEFINED to the parameter values. How would that look work with the python interface, where ST_UNDEFINED is not existent?
What does the the following line pass to the double in case the parameter is undefined?

```
double xref = (*this)["ra"].real()
```

#4 - 02/13/2015 08:41 PM - Knödlseder Jürgen

You don't have to worry about ST_UNDEFINED, just use the GApplicationPar::is_undefined() method, e.g.

```
if sim["ra"].is_undefined():
    print("This parameter is undefined.")
```

Try for example

```
$ ctobssim ra=UNDEFINED dec=UNDEFINED
*** ERROR encounterted in the execution of ctobssim. Run aborted ...
*** ERROR in GApplicationPar::real(): Invalid value.
Parameter "ra" is undefined. Please specify a valid parameter value.
```

so if you use sim["ra"].real() without that check you will get an exception, hence nothing gets passed to the double.

For parameters that may be undefined you should use GApplicationPar::is_undefined() (or better GApplicationPar::is_valid()) before reading a parameter:

```
if sim["ra"].is_valid():
    ra = sim["ra"].real()
else:
    ra = -1 # Although I won't use -1 to signal a bad value
```

#5 - 02/15/2015 06:20 PM - Mayer Michael

I see. My basic idea was to omit the RoI and time selection in csspec (#1364)

So the required changes to ctselect would be to add three protected booleans: m_select_energy, m_select_roi, m_select_time.
Then, in ctselect::get_parameters(), we incorporate the following checks:

```
if ((*this)["ra"].is_valid()) {
    m_ra = (*this)["ra"].real();
}
else {
    m_select_roi = false;
}
```

The same has then to be done for "dec", "rad", "emin", "emax", "tmin", and "tmax". The new protected booleans can then be used to define the selections in ctselect::select_events().
The drawback would be that this option could not be used from the command line, because the parameter will be checked directly after query. Accordingly, an exception would be thrown if the user wants to pass an undefined value.

An alternative would be to just add three hidden boolean parameters to ctselect that steer if a selection of RoI, Energy or time is performed respectively.
What do you think?

#6 - 02/15/2015 08:42 PM - Knödlseder Jürgen

Mayer Michael wrote:

> I see. My basic idea was to omit the RoI and time selection in csspec (#1364)
>
> So the required changes to ctselect would be to add three protected booleans: m_select_energy, m_select_roi, m_select_time.
> Then, in ctselect::get_parameters(), we incorporate the following checks:
> [...]
>
> The same has then to be done for "dec", "rad", "emin", "emax", "tmin", and "tmax". The new protected booleans can then be used to define the selections in ctselect::select_events().
> The drawback would be that this option could not be used from the command line, because the parameter will be checked directly after query.
> Accordingly, an exception would be thrown if the user wants to pass an undefined value.

No, the exception is only thrown if you try to access the value of a parameter with the real() method. Hence if a value is undefined, but you check whether the parameter is_valid() before reading, there should be no exception thrown as you never would call the real() method.

> An alternative would be to just add three hidden boolean parameters to ctselect that steer if a selection of RoI, Energy or time is performed respectively.
> What do you think?

This is not needed (see above)

#7 - 02/16/2015 09:37 AM - Mayer Michael

- Status changed from New to Pull request

- % Done changed from 0 to 100

Ok thanks for the clarification. I adapted ctselect to use the GApplicationPar::is_valid() function to decide if a selection should be performed. The change is available on *1422-adapt-ctselect-for-parameter-validity-checks*.
Quick tests and make check did work well.

#8 - 02/16/2015 10:33 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

- Assigned To set to Mayer Michael

Merged into devel.