ctools - Action #1828

Enhance the code quality

07/29/2016 01:40 PM - Knödlseder Jürgen

Status:	Closed	**Start date:**	07/29/2016
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.2.0		

Description

Today there are a number of major issues in the code quality analysis of ctools, which can be seen at https://cta-sonar.irap.omp.eu/component_issues?id=ctools#resolved=false|severities=MAJOR.

The ultimate goal is to reduce the number of major issues to zero.

This action concerns **everybody**. Just create a feature branch with the number of this issue and some text indicating what feature you are working on, and drop a message if you have some improvements to merge in.

Improvement concern mainly increasing of code coverage (and specifically the branch coverage), but improvements are certainly also possible in other areas.

There are a number of issues saying that there are not enough comments in the code. You should however not add comments to make these issues go away. Only add comments when they make sense. The Sonar Qube rules will eventually be adjusted to require less comments in the Python scripts, as the code is most pretty self explanatory.

Related issues:

Related to ctools - Feature # 1741: Refactor the cscripts to reduce the Sonar...	**In Progress**

History

#1 - 07/29/2016 03:45 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 10

- Add unit test for csinfo script, comment csinfo script and add test of cscripts module
- Revise cslightcrv script
- Add model definition file for all spectral and spatial models to increase the csmodelinfo test coverage
- Add a pointing definition file with maximum information to increase the csobsdef test coverage
- Test all algorithms in csresmap, refactor code to avoid many member variables

These changes led to the following improvements:

cscript	before	after
csfindobs	5 branches	-
csinfo	38 comments, 71 lines, 12 branches	20 comments
cslightcrv	21 comments	15 comments
csmodelinfo	9 comments, 5 branches	9 comments
csobsdef	16 comments, 5 branches	16 comments
csresmap	7 branches	-
cssens	51 comments, 1 duplicate, 26 branches	49 comments, 1 duplicate
cstsdist	19 comments, 2 duplicate, 18 branches	2 comments, 2 duplicate
cspull	32 comments, 1 duplicate	16 comments, 2 duplicate

The number of major issues went down from 106 to 97.

#2 - 08/01/2016 11:54 PM - Knödlseder Jürgen

- % Done changed from 10 to 20

I continued the work on the improvement of the code quality. The branch coverage of cscripts and example scripts have been improved. There are still some major issues remaining, mostly related to insufficient comments. I still need to go over the scripts and the example and to check whether commenting needs to be improved. Besides the "comments" issues there are remaining issues with the following scripts and examples:

cscript	**Issues**
csiactcopy.py	58 lines, 67 branches
csiactdata.py	1 branch
csiactobs.py	38 branches
csroot2caldb.py	26 branches

examples	**Issues**
show_pha.py	2 branches
show_pull_histogram.py	1 branch
show_sensitivity.py	3 branches

There are now 63 major issues remaining.

#3 - 08/02/2016 04:46 PM - Mayer Michael

- % Done changed from 20 to 30

I have been working on the IACT cscripts and tried to reduce the issues. At least I think I did something helpful - I have no measure if some of the issues really went away (or is there a way to run sonar on my local code?).

I have added some more tests and comments for csiactdata, csiactcopy and csiactobs (I didnt find issues for csfindobs). I also fixed a bug in csiactcopy (#1831). This tool however, cannot be very nicely covered by unit tests. For more extensive tests, we would need to add a larger database and test various scenarios (runlist, bad files, etc) which would take a lot of time.

I also realised that the unit tests fail for all IACT cscripts if the environment variable $VHEFITS is set. I am not sure yet how to circumvent this. For now, $VHEFITS overwrites the data path, leading to a wrong data path in the unit tests. Should we change the logic that if a datapath is provided (which is the case in our unit tests), the environment variable gets ignored?

My changes and modifications are available on branch *1828-quality-of-iact-cscripts*.

#4 - 08/02/2016 11:04 PM - Knödlseder Jürgen

user#77 wrote:

> I have been working on the IACT cscripts and tried to reduce the issues. At least I think I did something helpful - I have no measure if some of the issues really went away (or is there a way to run sonar on my local code?).

Unfortunately not. I will look into a possibility to set this up, but for the moment I can't see an obvious solution.

> I have added some more tests and comments for csiactdata, csiactcopy and csiactobs (I didnt find issues for csfindobs). I also fixed a bug in csiactcopy (#1831). This tool however, cannot be very nicely covered by unit tests. For more extensive tests, we would need to add a larger database and test various scenarios (runlist, bad files, etc) which would take a lot of time.

The results now are:

cscript	**Issues before**	**Issues now**
csiactcopy.py	58 lines, 67 branches	48 lines, 62 branches
csiactdata.py	1 branch	-
csiactobs.py	38 branches	12 branches

Concerning csiactcopy.py, I could improve the branch coverage by checking also with a higher chatter level of 4. I recognised that _merge() is never called. Also a test with a runlist file should increase the branch coverage. A test with a non-existing prodname would also be good.

For csiactobs.py a test case without IRF files should trigger some branches. Also a test case with the "irf" hierarchy would be good.

> I also realised that the unit tests fail for all IACT cscripts if the environment variable $VHEFITS is set. I am not sure yet how to circumvent this. For now, $VHEFITS overwrites the data path, leading to a wrong data path in the unit tests. Should we change the logic that if a datapath is provided (which is the case in our unit tests), the environment variable gets ignored?

Can't you simply set $VHEFITS to the correct path?

The self._datadir member in the test scripts point always to the root of the data directory. This is required since the tests are executed in two variants: one for the source package, and one once ctools are installed. For the later some test data files are copied in the distribution, but the relative file path will be different. By having all data files relative to self._datadir there should be no problem.

> My changes and modifications are available on branch *1828-quality-of-iact-cscripts*.

Merged these in.

#5 - 08/02/2016 11:38 PM - Knödlseder Jürgen

Latest results in devel:

cscript	Issues
csiactcopy.py	43 lines, 62 branches
csiactobs.py	11 branches

The total number of remaining major issues is 54 (and minor issues is 227).

P.S. Have you logged in to SonarQube? Would be good to check if this automatically creates a user that I can assign issues to.

#6 - 08/03/2016 01:05 AM - Knödlseder Jürgen

- Related to Feature #1741: Refactor the cscripts to reduce the Sonar issues added

#7 - 08/03/2016 10:13 AM - Mayer Michael

> Concerning csiactcopy.py, I could improve the branch coverage by checking also with a higher chatter level of 4. I recognised that _merge() is never called. Also a test with a runlist file should increase the branch coverage. A test with a non-existing prodname would also be good.

Ok I will think about a test case with a runlist and try to cover more branches.

> For csiactobs.py a test case without IRF files should trigger some branches. Also a test case with the “irf” hierarchy would be good.

Indeed, I will create a new branch from latest devel and add more tests.

> Can’t you simply set $VHEFITS to the correct path?

Well on default $VHEFITS points to my HESS FITS data (sometimes i forget to do an unset VHEFITS before running the tests, and then they fail). I am fine leaving as it is - however, other developers who have a similar setup might wonder why test cases fail.

> P.S. Have you logged in to SonarQube? Would be good to check if this automatically creates a user that I can assign issues to.

Yes, I am logged in.

#8 - 08/03/2016 11:52 AM - Knödlseder Jürgen

To simplify ctools and cscripts I introduced some additional GApplication methods that take a GChatter level as argument:

```
void log_string(const GChatter& chatter, const std::string& string);
void log_value(const GChatter& chatter, const std::string& name, const std::string& value);
void log_header1(const GChatter& chatter, const std::string& header);
void log_header2(const GChatter& chatter, const std::string& header);
void log_header3(const GChatter& chatter, const std::string& header);
void log_parameters(const GChatter& chatter);
```

This avoids the if statements related to the log levels in the client code, considerably simplifying the code. Here an example of how the new methods are used:

```
log_string(TERSE, "This is a message");
log_value(TERSE, "Counts cube file", m_outcube.url());
log_header1(TERSE, gammalib::number("Bin observation", m_obs.size()));
log_parameters(TERSE);
```

Note that the interface of the log_parameters() method has been changed, implying changes in all ctools and cscripts. This is to enforce usage of the new logging logic.

Code is now merged into devel.

#9 - 08/03/2016 01:35 PM - Mayer Michael

Ok sounds good, in oder to avoid conflicts: should I take care of adapting the ctools and cscripts to the new logger interface? Or are you already on it?

#10 - 08/03/2016 04:49 PM - Mayer Michael

> Ok sounds good, in oder to avoid conflicts: should I take care of adapting the ctools and cscripts to the new logger interface? Or are you already on it?

Ah I was too quick - just saw the updates in the repository. Thanks!

#11 - 08/03/2016 06:23 PM - Mayer Michael

- % Done changed from 30 to 40

I have been further working on enhancing the TestSuite for the IACT scripts. The changes I made are the following:

- Added a more realistic IACT database (consisting of 9 observations on two different sky positions). I scattered missing information here and there to cover as many branches in the IACT scripts as possible.
- Added csobsdef::obs() - function (I needed it to create the simulated database). Btw. I now have a script that creates such a database. Would this be useful in ctools? I wouldnt know a place where to put it.
- Restructured csiactobs how files are handled in order to avoid code duplication
- Added two runlists to test more functionality in csiactcopy.

The changes are available on branch *1828-enhance-csiactdata-tests*.

#12 - 08/05/2016 12:45 PM - Knödlseder Jürgen

Congratulations, there are no longer Sonar issues on the IACT scripts! Thanks for having increases the test coverage.

All cscripts and example now only have major issues related to the comment density. I still want to see whether the scripts are sufficiently well documented, without however adding comments to simply increase the density. One solution may then be to simply accept the comment densities as is. Another would be to change the density limit. I think I prefer the former.

I will now work on improving the coverage of the ctools.

#13 - 08/05/2016 02:26 PM - Mayer Michael

Thanks for merging the code! I noticed that you modified csiactcopy a bit and removed the str conversion of json dictionary entries. This conversion is in fact necessary since json dictionaries don't return string but unicode and GammaLib cannot deal with this. Therefore, the conversion is necessary. I tried to quickly copy an additional FITS production and got the following error (which apparently is not covered by a unit test):

```
Traceback (most recent call last):
  File "/Users/mimayer/Software/ctools/bin/csiactcopy", line 754, in <module>
    app.execute()
  File "/Users/mimayer/Software/ctools/bin/csiactcopy", line 739, in execute
    self.run()
  File "/Users/mimayer/Software/ctools/bin/csiactcopy", line 552, in run
    self._log_header3(gammalib.EXPLICIT, msg)
  File "/Users/mimayer/Software/gammalib/lib/python2.7/site-packages/gammalib/app.py", line 345, in _log_header3
    return _app.GApplication__log_header3(self, chatter, header)
TypeError: in method 'GApplication__log_header3', argument 3 of type 'std::string const &'
2016-08-05T12:24:01:
2016-08-05T12:24:01: Application "csiactcopy" terminated after 1 wall clock seconds, consuming 0.011701 seconds of CPU time.
```

I therefore suggest to keep the str conversions from json throughout this tool.

#14 - 08/05/2016 04:41 PM - Knödlseder Jürgen

Have you checked the latest version? It should already be corrected ...

#15 - 08/08/2016 10:32 AM - Mayer Michael

> Have you checked the latest version? It should already be corrected ...

Yes, I have used the latest devel version. I still get the same error. The problem is when I add the following statement in the code:

```
print(type(msg))
```

it returns <type 'unicode'> and not <type 'str'>. So either we add support for unicode strings in GammaLib Python extension, or we use the conversion str(msg). By the way: the same also goes for filenames that were read from the json file.

#16 - 08/08/2016 01:38 PM - Mayer Michael

By the way: On branch *1828-enhance-csiactdata-tests*, added the following code to test_python_cscripts.py

```
    # Check for VHEFITS environment variable
    if 'VHEFITS' in os.environ:

        # If existent unset for the test cases
        # Since Python is executed in a subprocess
        # this will not impact the environment variable in
        # the parent shell
        del os.environ['VHEFITS']
```

This removes the VHEFITS environment variable from the current Python process. It however leaves the the variable set in the parent shell (since Python is always called in a subprocess).

Could you merge that in?

#17 - 08/09/2016 01:10 AM - Knödlseder Jürgen

- % Done changed from 40 to 50

I merged the change and also put back the former version of the csiactcopy script, hope everything is fine.

I also finished to upgrade all ctools to the new logging interface. I also enhanced some unit tests.

Now need to work on the remaining branch coverage issues.

#18 - 08/09/2016 10:10 AM - Knödlseder Jürgen

The actual csiactcopy.py script fails on Python 3. This was in fact the reason in the first place that I did some modifications. The problem is an integer division, which in Python 3 should be done using // instead of / (the later will give a float, not an integer).

I corrected the bug and I'm currently testing the corrected version.

#19 - 08/09/2016 10:23 AM - Mayer Michael

Thanks, it seems to work from my side now.

#20 - 08/09/2016 11:15 AM - Knödlseder Jürgen

Code checked successfully, now in devel.

#21 - 08/10/2016 10:45 PM - Knödlseder Jürgen

Added GSkyMap::is_empty() method to simply testing of empty maps and cubes in ctools.

#22 - 08/11/2016 10:00 AM - Knödlseder Jürgen

There is still a mix of the observation setup methods used in the ctools.

ctools should use if possible ctool::setup_observations(), which requires a valid inobs parameter and handles observation definition XML file, event lists and event cubes. For example

```
// If there are no observations in container then load them via user
// parameters
if (m_obs.size() == 0) {

    // Throw exception if no input observation file is given
    require_inobs(G_GET_PARAMETERS);

    // Build observation container
    m_obs = get_observations();

} // endif: there was no observation in the container

// ... otherwise add response information and energy boundaries in case
// that they are missing
else {
    setup_observations(m_obs);
}
```

can be replaced by

```
// Setup observation container from "inobs" parameter
setup_observations(m_obs);
```

Eventually, some of the former ctools setup methods may become obsolete.

#23 - 08/11/2016 12:27 PM - Knödlseder Jürgen

- % Done changed from 50 to 60

Merged the change of all ctools into devel. This should reduce code duplication. Have not yet checked for unused ctool methods.

#24 - 08/11/2016 01:59 PM - Knödlseder Jürgen

I added the ctool::is_valid_filename() method that checks whether a filename is either empty or "NONE". This simplifies a lot the code, reducing the branches.

#25 - 08/12/2016 12:35 PM - Knödlseder Jürgen

I introduced a base class for likelihood tools names ctlikelihood. The main purpose (for now) of this base class is to avoid code duplication (specifically the evaluate() method that was identical in ctulimit and cterror).

So far the ctlike, ctulimit, cterror and cttsmap tools derive from that base class.

#26 - 08/12/2016 02:15 PM - Knödlseder Jürgen

Another reason for code duplication is iterations over an observation container. The typical example that is seen in many ctools is

```
   // Loop over all observation in the container
   for (int i = 0; i < m_obs.size(); ++i) {

      // Write header for the current observation
      log_header3(TERSE, get_obs_header(m_obs[i]));

      // Get CTA observation
      GCTAObservation* obs = dynamic_cast<GCTAObservation*>(m_obs[i]);

      // Skip observation if it's not CTA
      if (obs == NULL) {
         std::string msg = " Skipping "+m_obs[i]->instrument()+
                           " observation";
         log_string(NORMAL, msg);
         continue;
      }

      // Skip observation if we have a binned observation
      if (obs->eventtype() == "CountsCube") {
         std::string msg = " Skipping binned "+obs->instrument()+
                           " observation";
         log_string(NORMAL, msg);
         continue;
      }

      // Map events into sky map
      map_events(obs);
```

This code could be simplified in C++ to

```
for (GCTAObservation* obs = first_unbinned_observation(); obs != NULL; obs = next_unbinned_observation()) {
   do_something(obs);
}
```

and in Python to

```
for obs in unbinned_observations():
   do_something(obs)
```

#27 - 08/12/2016 05:43 PM - Knödlseder Jürgen

The following code now also works in Python:

```
# Test of iterator
for obs in bin._unbinned_observations():
    print(obs)
```

Here the relevant part of the Python code in ctobservation.i:

```
%pythoncode %{
def _unbinned_observations(self):
    obs = self._first_unbinned_observation()
    while obs != None:
        yield obs
        obs = self._next_unbinned_observation()
ctool._unbinned_observations   = _unbinned_observations
cscript._unbinned_observations = _unbinned_observations
%}
```

#28 - 08/14/2016 03:23 AM - Knödlseder Jürgen

- % Done changed from 60 to 90

Coming close to the end ...

#29 - 08/14/2016 07:52 AM - Knödlseder Jürgen

- Status changed from In Progress to Closed

- % Done changed from 90 to 100

- Remaining (hours) set to 0.0

Quality Gate passed!

#30 - 08/15/2016 01:15 PM - Mayer Michael

- Estimated time set to 0.00

Great job and worth the effort! I believe we also fixed a lot of bugs on the way.

#31 - 08/15/2016 03:41 PM - Mayer Michael

Btw: I checked the latest code found that ctselect still uses a 'classical' observation loop. Should this tool also inherit from ctobservation and use the new observation loop?

#32 - 08/22/2016 07:58 AM - Knödlseder Jürgen

Definitely. All tools that hold observation containers should indeed derive from ctobservation. I have not yet had the time to make this change. Created issue #1846 for this.