RAT-STATS 2017 Module Extending
April 20, 2017

Table of Contents

Description

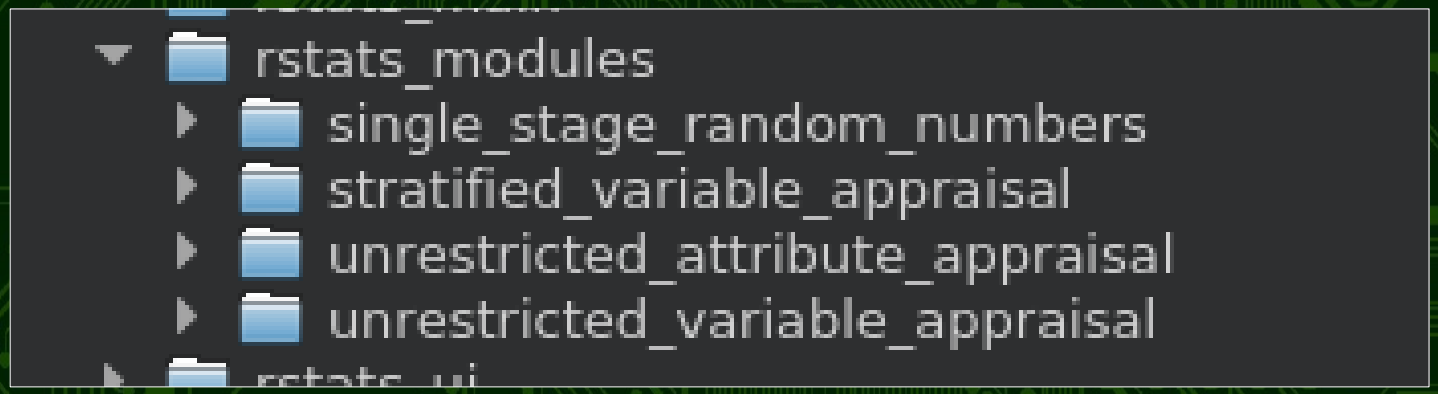
RAT-STATS 2017 was designed from the start to allow for easy, no none sense, modular extending. A **module** is a separate, self-contained executable or script that can be launched from the main menu.

RAT-STATS 2017 separates modules into two distinct categories:

1) External Modules
2) Internal Modules

Internal Modules

- Internal modules represent independent executables that is part of the C++ code base
- All four modules provided for this submission are internal modules. These are located in the rstats_modules folder in the code base:



Internal Modules PROS and CONS

PROS	CONS
Can share / utilize existing source code to / from other internal modules	Requires installation of QtSDK and CMake for building additional internal modules
Can utilize speed / performance of C++ to implement advanced statistical algorithms	Requires good knowledge of C++

External Modules

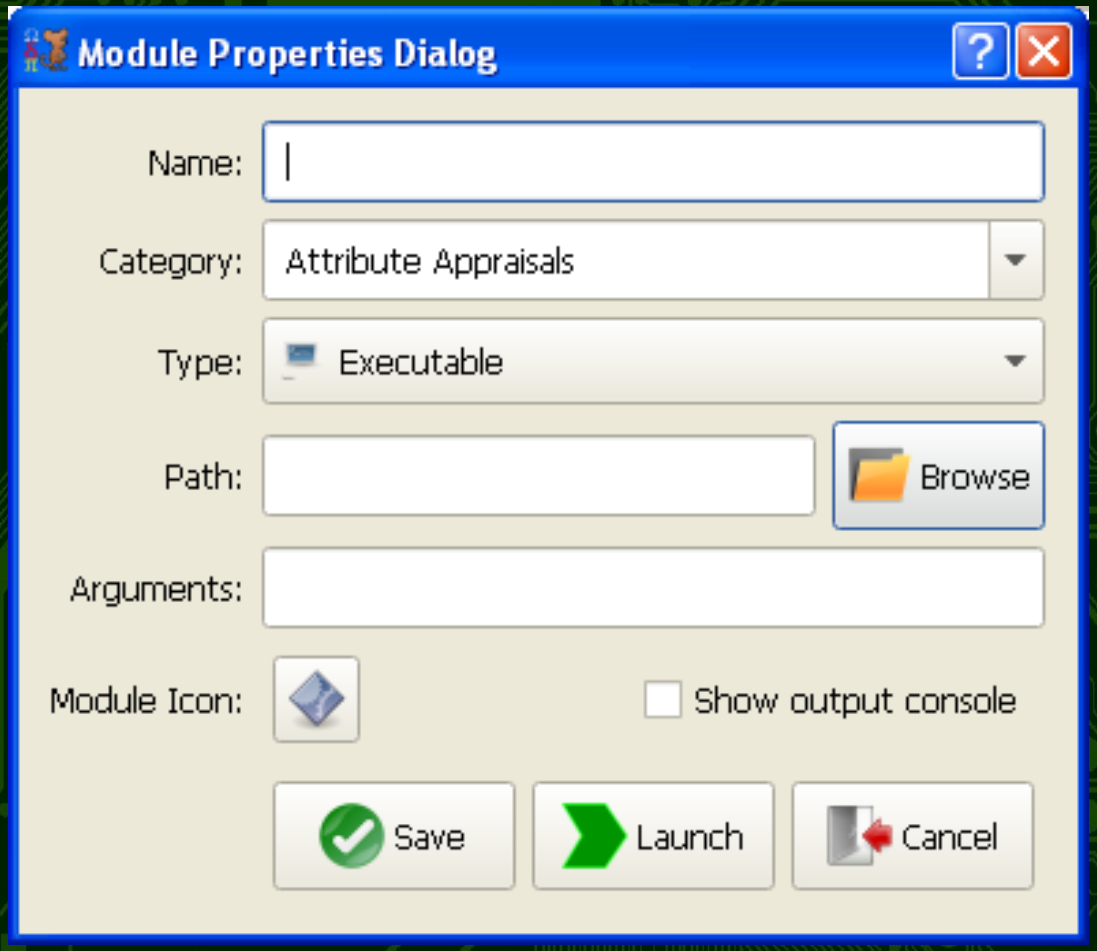
- External modules represent any executable program or script that provides value to the end user. They can be added with optional command line arguments to the RAT-STATS 2017 launch menu.

External Modules PROS / CONS

PROS	CONS
Programmers can utilize any language they desire to extend RAT-STATS	Sharing data between different external modules may only occur through command line arguments
End users can create custom scripts to extend RAT-STATS without needing access to the code base	May require additional files to support external scripts / executables

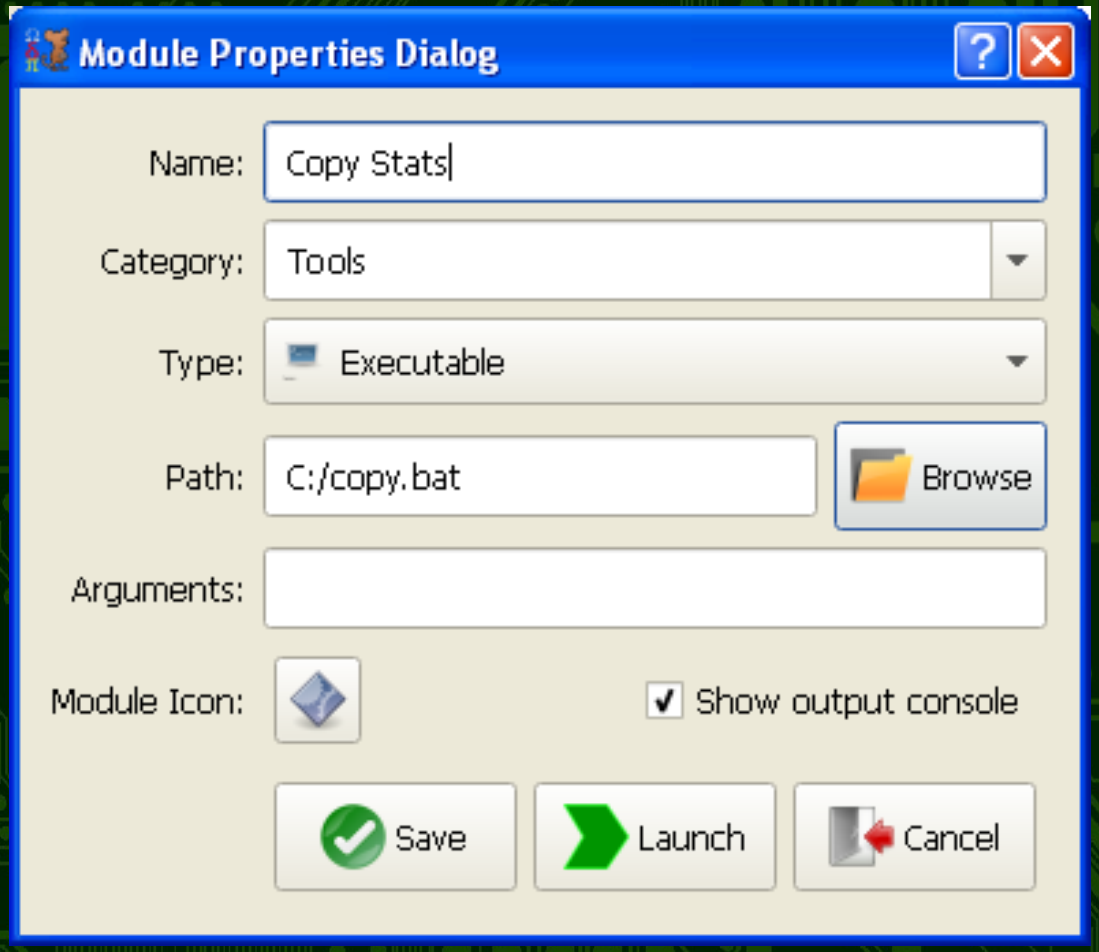
Adding any module to RAT-STATS

- Adding a module to RAT-STATS is a simple 1-step process.
- Both internal and external modules are added to RAT-STATS using this method.
- Click the “Add module” button or use the Alt+N shortcut key from the main menu.



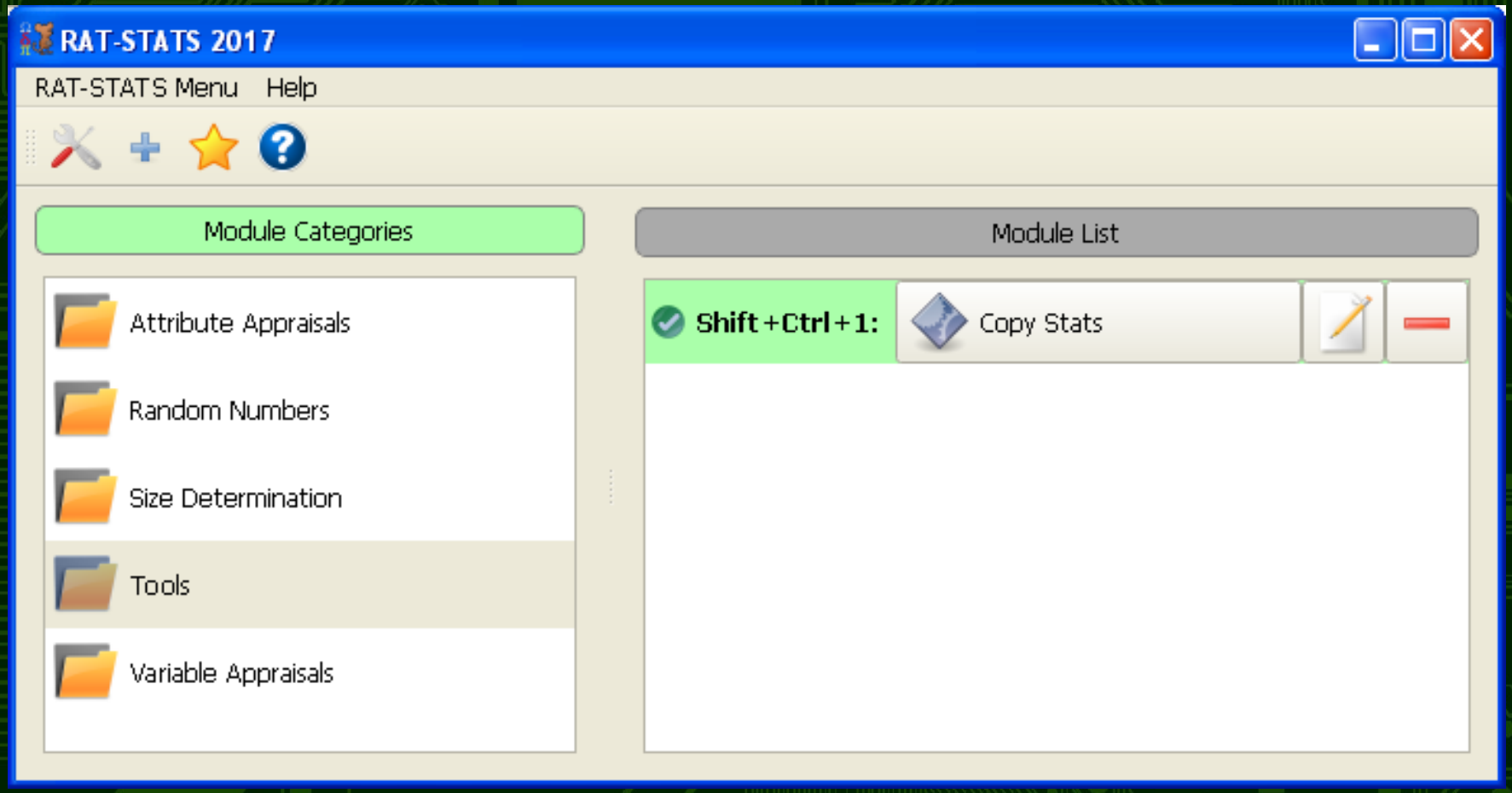
Adding any module to RAT-STATS

- As an example I will be adding a simple Windows copy script as a module.
- Notice that I am also creating a new category named “Tools”. You also have the option to select an existing category to add your script to. Keep in mind that all modules are sorted and that the shortcut key for a module may change if that sorting order is changed.



Adding any module to RAT-STATS

After clicking “Save” I get a new category “Tools” with a single module. Notice that every module that gets added to a category gets sorted and gets assigned a automatic shortcut key. Thats pretty much all there is to adding a module. You can delete and edit modules using the <Delete> key and <F1> key respectively. Or you could just use the mouse.



Adding an external module projects

- External module projects exist outside of the code base for RAT-STATS 2017. They can be in whatever language desired as long as the final product is an executable or a script that is supported by RAT-STAT 2017’s list of script providers.

Adding an internal module project

- It is highly recommended that developers use the ProjectGen (pgen) / SourceGen (sgen) tools for adding new module projects and source code. These tools are located at:
- **rstats_source/products/RAT-STATS/rstats_contrib/pgen**
- **rstats_source/products/RAT-STATS/rstats_contrib/sgen**
- It is recommended that the location of these tools be added to the PATH environment variable so that they may easily be ran from the command line. Note that there are also graphical versions of these tools (**sgen_qt.exe** and **pgen_qt.exe**) that do not require changes to the environment variables.

Adding an internal module project 4-step process overview

- Generate new module project using pgen or pgen_qt
- Add rstats_ui, rstats_utils and common_utils to the CMakeLists.txt file as dependencies for module project
- Generate new Qt-based GUI form using sgen or sgen_qt
- Add two lines of code to load the new GUI form in main.cpp of module project.

Adding an internal module project using ProjectGen (command-line) Step 1

- Open a command line window and navigate to the **rstats_source/products/RAT-STATS/rstats_modules**
- At the prompt, type **`pgen -help`** and press <ENTER>
- You should see the following help information:

```
-------------------------------------
ProjectGen Help
-------------------------------------
--project-name   [-n] <name of project>    (required)
--project-type   [-t] <type of project>    (optional, see class types below)
--project-path   [-p] <path of project>    (optional)
--enable-logging [-l]                      (writes log to /home/cbtek/.pgen.log)
--help           [-h]                      (displays this help message)
-------------------------------------
Valid project-types:
        1) CPA   (C++ Application, default)
        2) CPL   (C++ Library)
        3) QTA   (Qt Application)
        4) QTL   (Qt Library)
        5) BASE  (Top-level CMake for new Code-base)
-------------------------------------
Example:
pgen -n "MyProject" -t "CPL" -p "/home/user/project"
-------------------------------------
```

Adding an internal module project using ProjectGen (command line) Step 1 (cont.)

In the command prompt do the following:

Type `pgen -n “ModuleName” -t “QTA”` to create a Qt GUI based module

Or type `pgen -n “ModuleName” -t “CPA”` to create a command line based module

Example:

```
C:\dev\RStats2017\products\RAT-STATS\rstats_modules>pgen -n "MyModule" -t "QTA"

C:\dev\RStats2017\products\RAT-STATS\rstats_modules>dir
 Volume in drive C has no label.
 Volume Serial Number is 041E-DD0B

 Directory of C:\dev\RStats2017\products\RAT-STATS\rstats_modules

04/19/2017  11:32 PM    <DIR>          .
04/19/2017  11:32 PM    <DIR>          ..
04/19/2017  11:32 PM    <DIR>          MyModule
04/19/2017  02:13 PM    <DIR>          single_stage_random_numbers
04/19/2017  02:13 PM    <DIR>          stratified_variable_appraisal
04/19/2017  02:13 PM    <DIR>          unrestricted_attribute_appraisal
04/19/2017  02:13 PM    <DIR>          unrestricted_variable_appraisal
               0 File(s)              0 bytes
               7 Dir(s)   4,979,023,872 bytes free

C:\dev\RStats2017\products\RAT-STATS\rstats_modules>
```

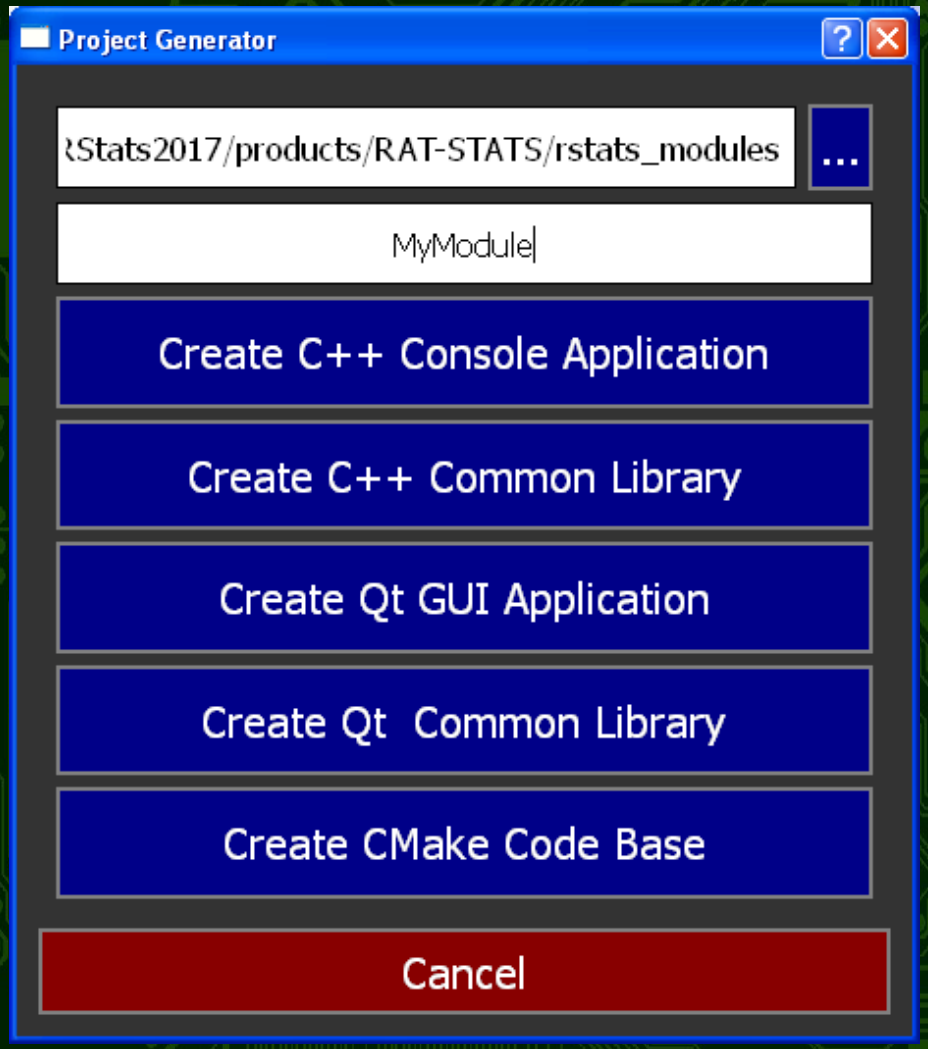
Adding an internal module project using ProjectGen (GUI) Step 1 (cont.)

1) Select the location of the rstats_modules folder.

2) Enter the name of the project

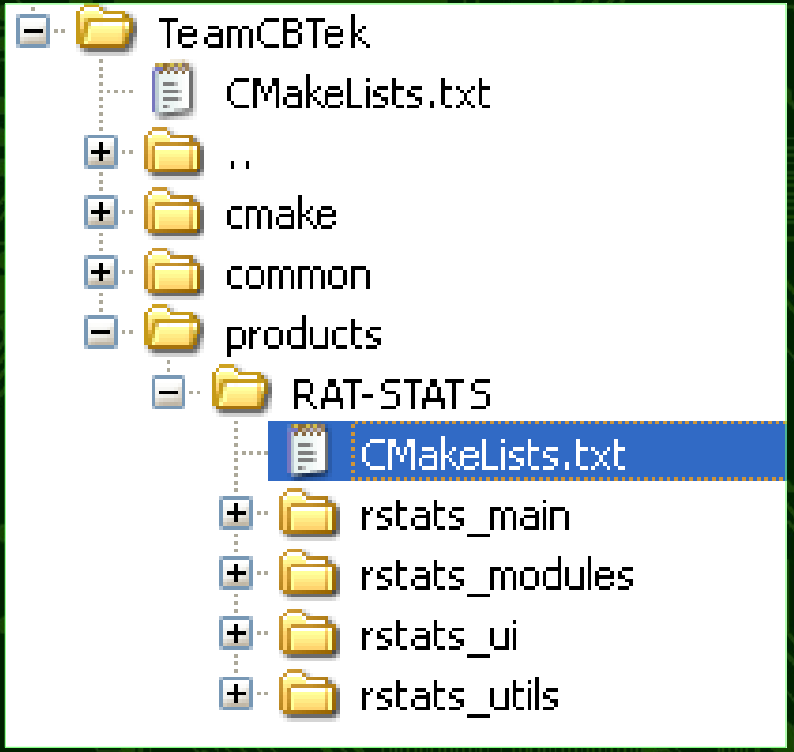
3) Click “Create C++ Qt GUI Application” or “Create C++ Console Application”

4) Click Cancel to close



Adding internal module to CMake Step 1 (cont.)

- Go back to QtSDK and locate the CMakeLists.txt file under the “RAT-STATS” folder:



This file contains configuration for all RAT-STATS projects

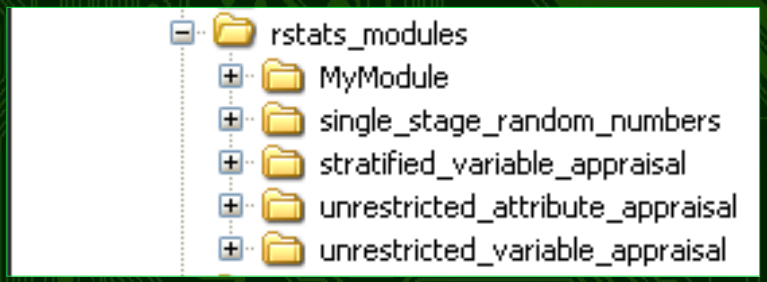
Go ahead and open this file in the editor. (Double click it)

Adding internal module to CMake Step 2

Locate the “Internal Modules” section and add a new line “add_subdirectory” that points to the location of your newly created module. In the example below I added a new line that tells CMake to load the “MyModule” project:

```
1  ##################################################
2  #!This file is autogenerated by the CBTek ProjectGen
3  ##################################################
4  cmake_minimum_required(VERSION 3.0)
5  project(RAT-STATS)
6  add_subdirectory("${PROJECT_SOURCE_DIR}/rstats_utils")
7  add_subdirectory("${PROJECT_SOURCE_DIR}/rstats_ui")
8  add_subdirectory("${PROJECT_SOURCE_DIR}/rstats_main")
9
10 #Internal Modules
11 add_subdirectory("${PROJECT_SOURCE_DIR}/rstats_modules/MyModule")
12 add_subdirectory("${PROJECT_SOURCE_DIR}/rstats_modules/single_stage_random_numbers")
13 add_subdirectory("${PROJECT_SOURCE_DIR}/rstats_modules/unrestricted_attribute_appraisal")
14 add_subdirectory("${PROJECT_SOURCE_DIR}/rstats_modules/unrestricted_variable_appraisal")
15 add_subdirectory("${PROJECT_SOURCE_DIR}/rstats_modules/stratified_variable_appraisal")
16
```

1) Save the file.
2) Right click the “TeamCBTek” folder in the project tree and click “Run CMake”
3) After CMake finishes you should see your module pop up in the project tree. (See image to right for example)



Updating internal module configuration Step 2 (cont.)

- Now that the internal module has been created and added to the project, we need to update its configuration and (if applicable) add a GUI form.
- In QtCreator go ahead and expand the project tree for the newly created module. You will notice that there are only two files in the project.
- Go ahead and open the **CMakeLists.txt** file for your module in the editor. (Double-click it)

Updating internal module configuration Step 2 (cont.)

- At the bottom of the CMakeLists.txt file you will notice a couple of lines commented out:

```
#==============================================================
# Add Dependencies
#==============================================================
qt5_use_modules(${CURR_TARGET} Widgets)
#add_dependencies(${CURR_TARGET} )

#==============================================================
# Target Link Libraries
#==============================================================
#target_link_libraries(${CURR_TARGET} Qt5::Widgets)
```

Updating internal module configuration Step 2 (cont.)

1) Uncomment the lines for “add_dependencies” and “target_link_libraries”

2) Make your CMakeLists.txt look like the image to the right.

3) This ensures that your module has access to the rstats_ui, rstats_utils and common_utils library.

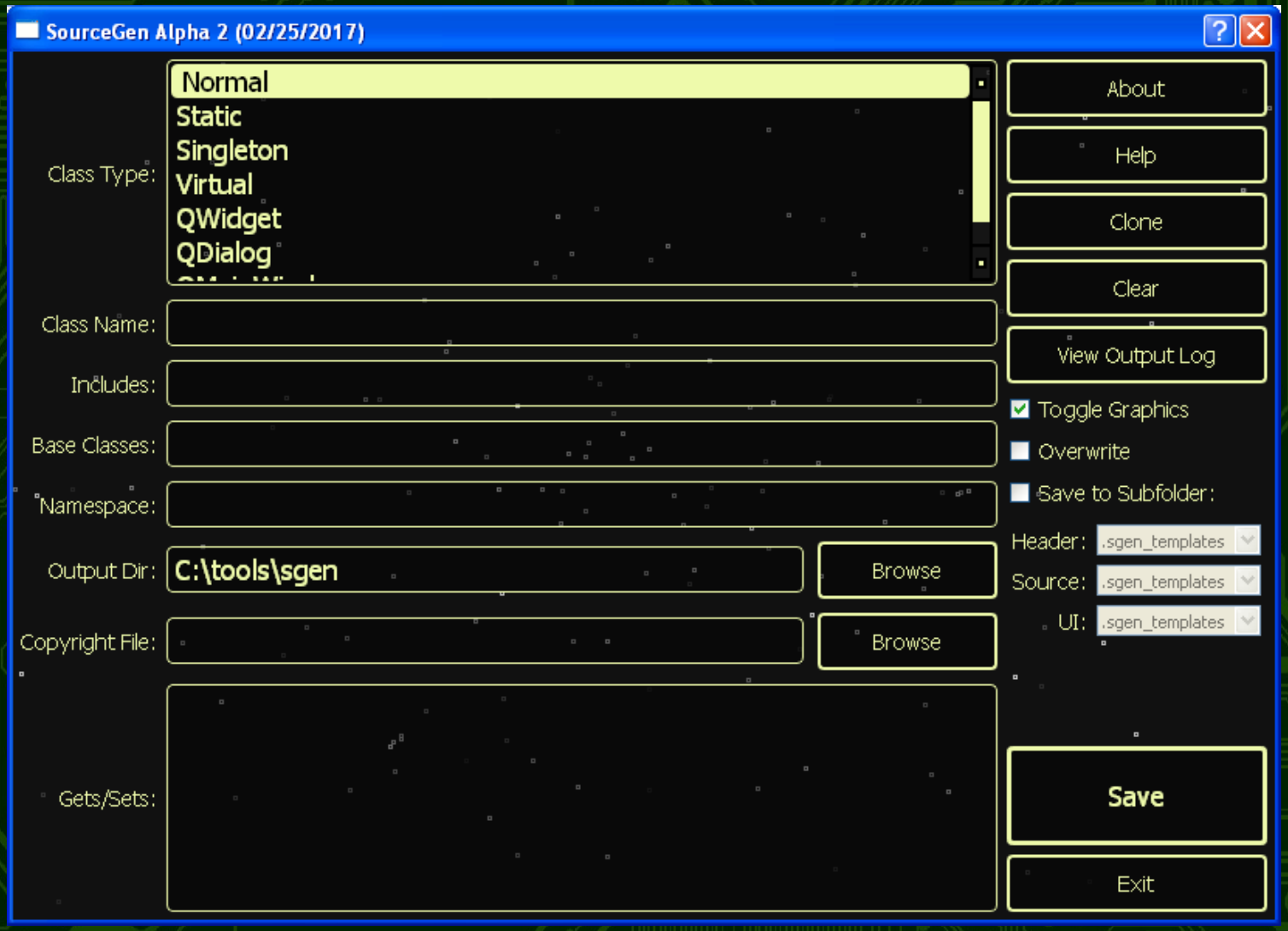
4) When you are done save and re-run CMake.

```
45
46  #=========================================
47  # Add Dependencies
48  #=========================================
49  qt5_use_modules(${CURR_TARGET} Widgets)
50  add_dependencies(${CURR_TARGET}
51                   rstats_ui
52                   rstats_utils
53                   common_utils)
54
55  #=========================================
56  # Target Link Libraries
57  #=========================================
58  target_link_libraries(${CURR_TARGET}
59                        rstats_ui
60                        rstats_utils
61                        common_utils
62                        Qt5::Widgets)
```

Adding a GUI class form using SourceGen (GUI) Step 3

- If you created a C++ application (CPA) with ProjectGen or will not need a GUI for your module then you can skip this part. This tutorial can be still be used however to add additional classes to your module.
- I suggest using the SourceGen GUI for adding for this part of the tutorial.
- In your sgen folder double click to open **sgen_qt.exe**

Adding a GUI class form using SourceGen (GUI) Step 3 (cont.)



Adding a GUI class form using SourceGen (GUI) Step 3 (cont.)

- First you need to decide what kind of GUI form you wish to create. SourceGen supports creating:
- **QMainWindow** – This is a full window form for simple to advanced applications. The other four internal modules use this type.
- **QDialog** – This is a form that meant to be used for quickly showing very small amounts of data / controls.
- **QWidget** – This form type is meant to be embedded within other QDialogs or QMainWindows
- I am going to go ahead and use QMainWindow for the example.

Adding a GUI class form using SourceGen (GUI) Step 3 (cont.)

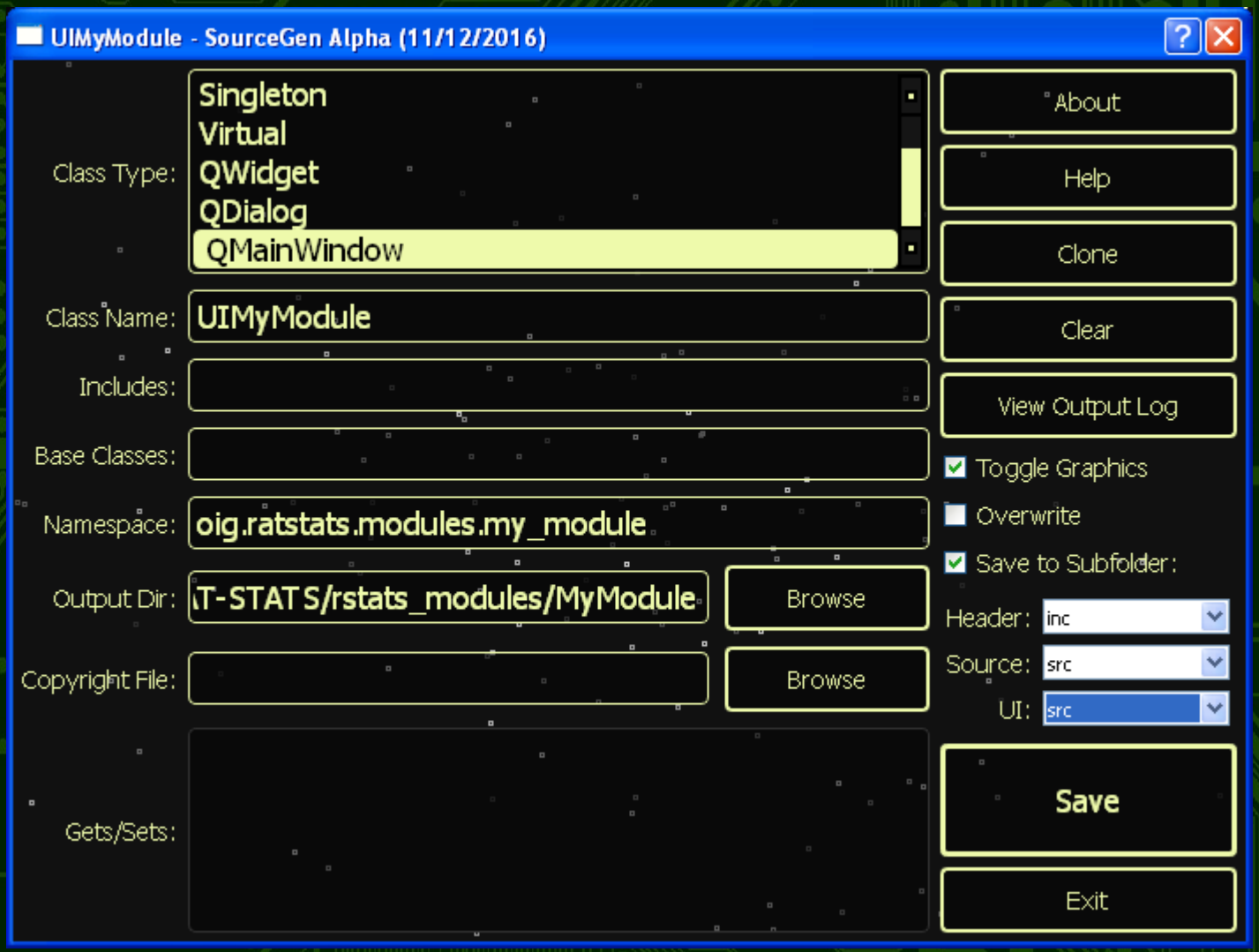
1) Select QMainWindow from the “Class Type” list

2) Enter the name of the class. I prefix all forms with “UI” in RAT-STATS.

3) Perhaps the most time saving feature of this application is the “Namespace” field. Here I can add a “dot-seperated” namespace that will be added to both my header and source files.

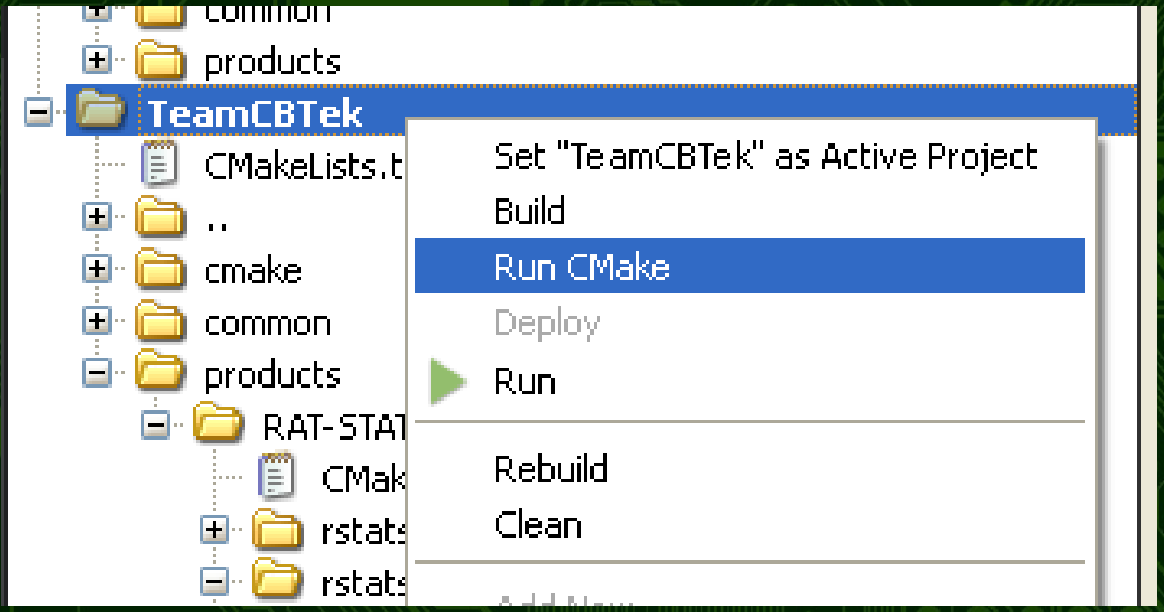
4) Another important thing to note is the “Save to Subfolder” checkbox. RAT-STATS puts all header files (.h, .hpp) into a sub-folder called “inc” and all implementation / user interface files (.cpp, .ui) into a sub-folder called “src”. This helps keep the code base clean and organized.

If everything looks right, click “Save” to continue.



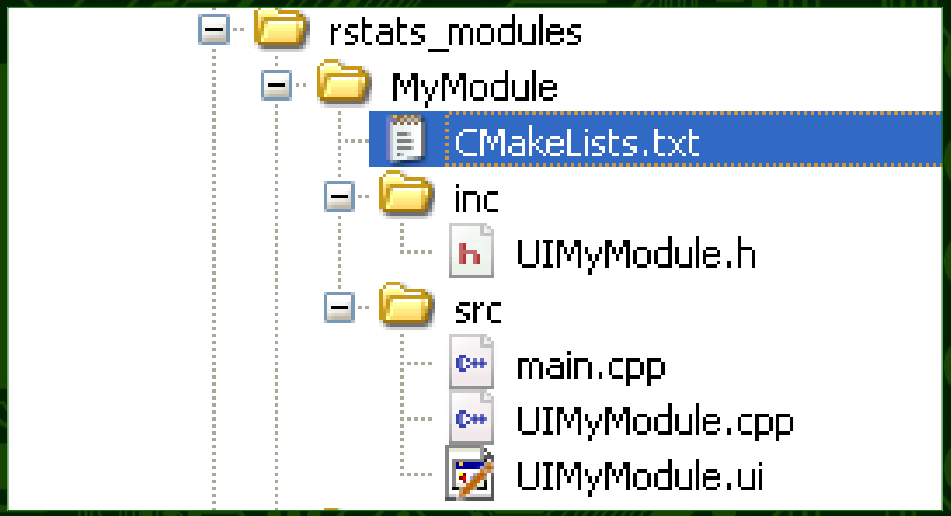
Adding a GUI class form using SourceGen (GUI) Step 3 (cont.)

- Go back to QtCreator and re-run CMake(Right-click on “TeamCBTek” in the project tree and click “Run CMake”



Adding a GUI class form using SourceGen (GUI) Step 4

- After CMake finishes you will notice that your module project has three new files.
- The C++ class (.h and .cpp)
- The Qt UI Form (.ui)
- Go back to your main.cpp file and add code to load in your new form.




```cpp
#include <QApplication>
#include "UIMyModule.h"

using namespace oig::ratstats::modules::my_module;

int main(int argc, char ** argv)
{
    QApplication a(argc,argv);
    UIMyModule myModuleForm;
    myModuleForm.resize(1024,768);
    myModuleForm.show();
    return a.exec();
}
```