



Submission Guide

TeamCBTek

May 8, 2017

Table of Contents

1. General Submission Overview.....	2
1.1 The rstats_binary Folder.....	2
1.2 The rstats_source Folder.....	3
1.3 Source Code Documentation and Comments.....	3
2. Features and Differences.....	4
2.1 General features overview.....	4
2.2 508 Compliance.....	4
2.3 Error Handling.....	5
2.4 Input / Output Differences.....	6
2.5 Development Differences.....	6
3. Contributed Software.....	6
3.1 The catch library.....	7
3.2 The common_utils library.....	7
3.3 The basic_excel library.....	7
3.4 The mini_excel library.....	7
3.5 The zLib library.....	7
3.6 The tinyxml2 library.....	8
3.7 The minizip library.....	8
3.8 The SourceGen utility.....	8
3.9 The ProjectGen utility.....	8
4. Development Guide.....	8
4.1 Setting up an environment on Windows.....	8
4.2 Setting up an environment on Linux.....	25
4.3 Build artifacts and deployment (Windows).....	26
5. Extending RAT-STATS with additional modules.....	27
5.1 Internal Modules.....	28
5.2 External Modules.....	28
5.3 Adding additional modules to RAT-STATS (Internal or External).....	29
5.4 Creating new internal module projects.....	31
6. Contact and Support.....	36

1. General Submission Overview

The **TeamCBTek_Second_Submission.zip** archive contains the following items:

- 1) `rstats_binary` – The RAT-STATS application folder
- 2) `rstats_source` – The Source Code folder
- 3) `rstats_overview.pdf` – This document
- 5) `rstats_submission.pdf` – Single page submission document

1.1 The `rstats_binary` Folder

RAT-STATS 2017 implements a “micro-service” approach to developing the four target RAT-STATS functions. As a result the binary folder contains multiple executables, each representing a distinct isolated piece of functionality. This differs from RAT-STATS 2007/2010 “monolithic” approach which puts all functionalities into a single executable. Please see the “SDK Development” section for more details.

The `rstats_binary` folder contains the following executables which should be of interest:

- 1) `rstats.exe` – The main menu for RAT-STATS
- 2) `module_rn_single_stage_random_numbers.exe` – Single Stage Random Numbers function
- 3) `module_aa_unrestricted` – Unrestricted Attribute Appraisal function
- 4) `module_va_stratified.exe` – Stratified Variable Appraisal function
- 5) `module_va_unrestricted.exe` – Unrestricted Variable Appraisal function

1.2 The `rstats_source` Folder

RAT-STATS 2017 was created with C++ utilizing the QtSDK for its user interface and CMake as the build system. All primary dependencies are embedded in the source code and are provided by TeamCBTek. Please see the “SDK Development” section for a step by step guide on building RAT-STATS.

The `rstats_source` folder contains the following:

- 1) `CMakeLists.txt` – top-level CMake setup file that describes what projects need to be built
- 2) `common` – This folder contains source code and projects for commonly used support libraries
- 3) `products` – This folder contains all projects related to building the RAT-STATS tool
- 4) `cmake` – This folder contains commonly used CMake scripts that are used by other projects

1.3 Source Code Documentation and Comments

1.3.1 Source Code Documentation

I have included two PDFs with this release that document all the source code created by TeamCBTek.

- 1) rstats_common_sdk_documentation.pdf
- 2) rstats_products_sdk_documentation.pdf

1.3.2 Source Code Comments

I have increased comments in the code for both the common_utils project and all RAT-STATS projects. The majority of the code was written to be self documenting. That is, it should be easy to follow along without comments. Nevertheless comments are included in the header (.h) files for nearly every class. At the very least each public function should have a Doxygen comment associated with it.

1.3.3 Line count for common utility

Below I ran a line count program for the common_utility project: It primarily shows the line of comments vs the lines of code vs blank lines. Overall there are over 7700 lines of pure code in the common_utils library.

```
cbtek@zeroefusion-server:~/dev/RStats2017/common/utility$ ohcount
Examining 27 file(s)

Ohloh Line Count Summary

Language      Files      Code      Comment      Comment %      Blank      Total
-----
cpp            18        7706        2212         22.3%         1043       10961
cmake           2          37          21          36.2%          10         68
-----
Total          20        7743        2233         22.4%         1053       11029
cbtek@zeroefusion-server:~/dev/RStats2017/common/utility$
```

1.3.4 Line count for RAT-STATS projects

Below I ran a line count program for all the RAT-STATS projects. The XML refers to the UI files that represent Qt's UI forms. Overall there are over 17000 lines of pure code.

```
cbtek@zeroefusion-server:~/dev/RStats2017/products$ ohcount
Examining 341 file(s)

Ohloh Line Count Summary

Language      Files      Code      Comment      Comment %      Blank      Total
-----
cpp            81       12505        3690         22.8%         2725       18920
xml            14        4493          0          0.0%           5         4498
cmake           9         292         121         29.3%          95         508
-----
Total          104       17290        3811         18.1%         2825       23926
cbtek@zeroefusion-server:~/dev/RStats2017/products$
```

2. Features and Differences

2.1 General features overview

- 1) Supports both Windows and Linux (RHEL, Ubuntu, Fedora etc)
- 2) Does not require Microsoft Excel installation for importing / exporting Excel formats (XLSX / XLS)
- 3) Supports adding additional modules as compiled executables or as embedded scripts
- 4) Each module supports “Recently Used” history for up to ten previous runs
- 5) New windows can be created from module window without going back to main menu
- 6) All functions in RAT-STATS 2017 have an associated shortcut key

2.2 508 Compliance

2.2.1 Screen Readers

We noticed some minor issues with Microsoft Narrator while testing RAT-STATS 2017. The Narrator application would only work correctly around 70-80% of the time. Sometimes it took a few seconds to read an item and at other times it would require refocusing on an item before it would read it. Upon researching these problems we discovered that these are some known issues when using Narrator with the keyboard. For that reason we recommend using the free NVAccess screen reader tool from <https://www.nvaccess.org>. It has a lot more features and in our testing it has worked 100% of the time.

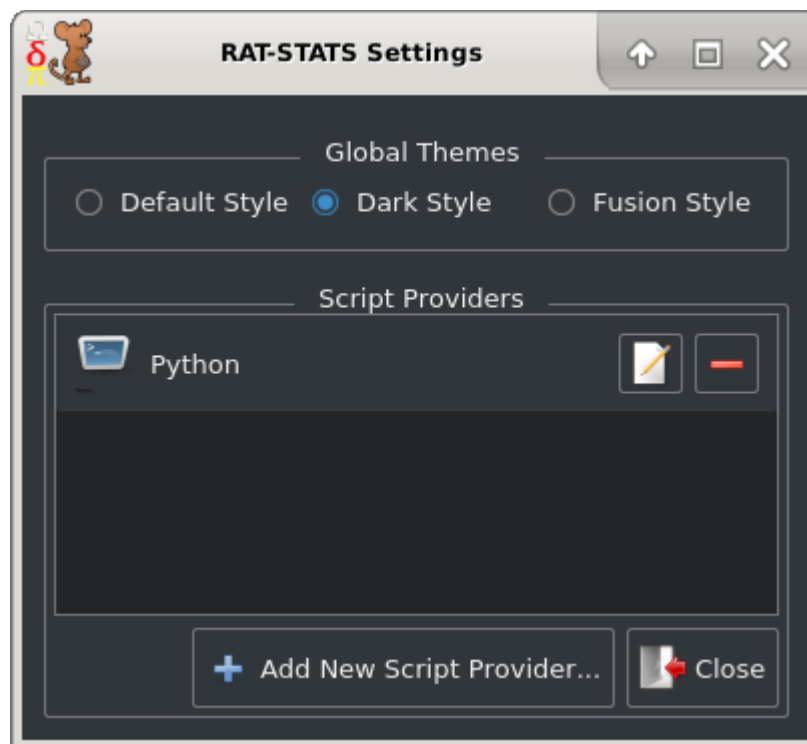
2.2.2 Keyboard

RAT-STATS 2017 can be controlled entirely by the keyboard.

Every item on the RAT-STATS 2017 main menu is automatically assigned a shortcut key to launch, edit and delete modules. This eliminates the need for configuration / maintenance of shortcuts definitions.

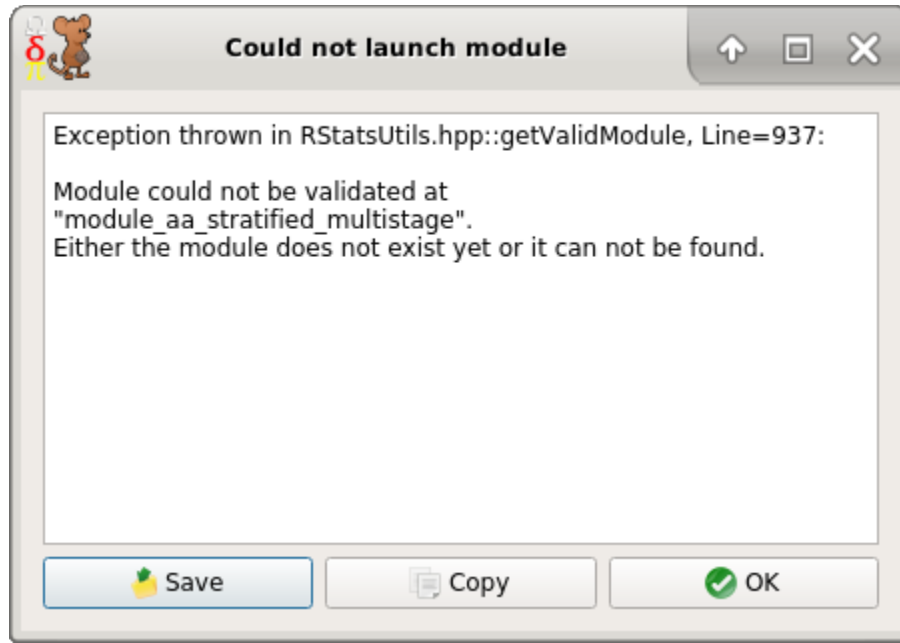
2.2.3 Display

RAT-STATS 2017 also supports using the operating system theme (by default) to style its controls/forms. There are also a couple of application centric themes available within the settings dialog.



2.3 Error Handling

RAT-STATS 2017 uses a common exception dialog for displaying exceptions. This allows users to easily save or copy error messages so they may be investigated. Most exceptions will provide a filename and line number of where the exception was thrown which will help developers find and solve issues faster.



2.4 Input / Output Differences

RAT-STATS 2017 supports saving output results to the following formats:

- 1) XLS – Microsoft Excel 97 and higher format.
- 2) CSV – Comma separated values filename
- 3) TXT – Evenly spaced human readable file
- 4) Web Browser (HTML) – Opens output in default web browser

For the stratified and unrestricted variable appraisal functions the following inputs are supported:

- 1) XLS – Microsoft Excel 97 and higher format.
- 2) XLSX – Excel 2007 and higher format.
- 3) CSV – Comma separated values filename
- 4) TXT, DAT, SSV – Tab and/or space delimited files

2.5 Development Differences

The previous versions of RAT-STATS looks like they were developed in Visual Basic 6 or lower. RAT-STATS 2017 is developed in C++11 using the Qt SDK as its GUI framework. External modules can be added to RAT-STATS 2017 using any language desired. Internal modules can be added using the C++.

3. Contributed Software

The number of external libraries / dependencies used by RAT-STATS 2017 was intentionally kept low. There are currently **six** source code libraries and **two** external executables used by RAT-STATS 2017. The source code libraries are compiled in with the project which negates the need to include binary DLLs with the source code.

3.1 The catch library

Name: catch
URL: <https://github.com/philsquared/Catch>
License (Boost): <https://github.com/philsquared/Catch/blob/master/LICENSE.txt>
Author: Phil Nash (philsquared)
Usage: Used for unit testing various components in RAT-STATS
Location: rstats_source/common/contrib/catch

3.2 The common_utils library

Name: common_utils
URL: <https://github.com/CBTek/utility>
License (MIT): <https://github.com/cbtek/CommonUtils/blob/master/LICENSE>
Author: Corey Berry (CBTek, TeamCBTek)
Usage: Provides hundreds of generic utility functions to ease coding in C++
Location: rstats_source/common/utility

3.3 The basic_excel library

Name: common_utils
URL: <https://www.codeproject.com/Articles/13852/BasicExcel-A-Class-to-Read-and-Write-to-Microsoft>
License: Public Domain, Unlicensed
Author: Yap Chun Wei
Usage: Provides low-level XLS read and write support
Location: rstats_source/common/contrib/basic_excel

3.4 The mini_excel library

Name: common_utils
URL: <https://github.com/qcdong2016/MiniExcel>
License: Public Domain, Unlicensed
Author: qcdong
Usage: Provides low-level XLSX read support
Location: rstats_source/common/contrib/mini_excel

3.5 The zlib library

Name: zlib
URL: <http://zlib.net>
License (zLib): http://zlib.net/zlib_license.html
Author: Jean-loup Gailly and Mark Adler
Usage: Popular compression library that is embedded within the mini_excel library
Location: `rstats_source/common/contrib/mini_excel/zlib`

3.6 The tinyxml2 library

Name: TinyXML2
URL: <https://github.com/leethomason/tinyxml2>
License (zLib): http://zlib.net/zlib_license.html
Author: Lee Thomason
Usage: Provides support for reading XML documents in the mini_excel library
Location: `rstats_source/common/contrib/mini_excel/tinyxml2`

3.7 The minizip library

Name: minizip
URL: <https://github.com/madler/zlib/tree/master/contrib/minizip>
License (zLib): http://zlib.net/zlib_license.html
Author: Mark Adler
Usage: Provides support for extracting ZIP files in the mini_excel library
Location: `rstats_source/common/contrib/mini_excel/minizip`

3.8 The SourceGen utility

Name(s): `sgen.exe`, `sgen_qt.exe`
URL: <https://github.com/cbtek/SourceGen>
License (MIT): <https://github.com/cbtek/SourceGen/blob/master/LICENSE>
Author: Corey Berry (CBTek, TeamCBTek)
Usage: Generates Qt and C++ classes that conform to coding standard
Location: `TeamCBTek_Tools.zip`

3.9 The ProjectGen utility

Name(s): `pgen.exe`, `pgen_qt.exe`
URL: <https://github.com/cbtek/ProjectGen>
License (MIT): <https://github.com/cbtek/ProjectGen/blob/master/LICENSE>
Author: Corey Berry (CBTek, TeamCBTek)
Usage: Generates CMake sub-projects that conform to coding standard. (This is one way to create internal module projects)
Location: `TeamCBTek_Tools.zip`

4. Development Guide

This guide will help you setup a development environment for RAT-STATS 2017.

First you need to decide which platform you are targeting:

4.1 Setting up an environment on Windows

4.1.1 Download and install CMake

CMake is a cross-platform build tool used to generate makefiles.

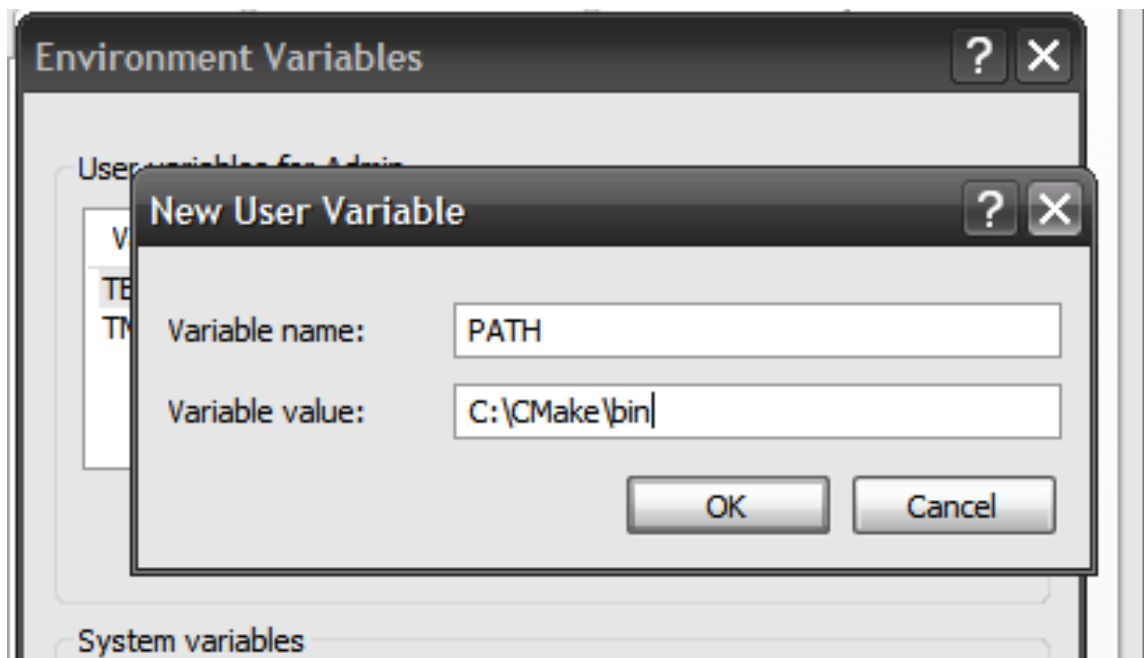
As of this writing the latest version is 3.8.0 and can be found here:

<https://cmake.org/files/v3.8/cmake-3.8.0-win32-x86.zip>

If the URL above is no longer valid then you can go to <https://cmake.org> and download the appropriate version. RAT-STATS 2017 requires at least version 2.8.0 of CMake.

Extract the CMake ZIP file to any path somewhere on your local machine. (e.g C:\CMake)

Close all command windows and add the URL to the “bin” folder located in the CMake installation to your PATH variable:



Click OK and close the dialogs.

Open a command prompt and enter `cmake --help`. Press Enter.
You should see some output that looks similar to below:

```
Usage

  cmake [options] <path-to-source>
  cmake [options] <path-to-existing-build>

Specify a source directory to (re-)generate a build system for the
current working directory. Specify an existing build directory to
re-generate its build system.

Options
  -C <initial-cache>           = Pre-load a script to preconfigure
  -D <var>[:<type>]=<value>    = Create a cmake cache variable <var>
                                with type <type> and value <value>
```

4.1.2 Install QtSDK

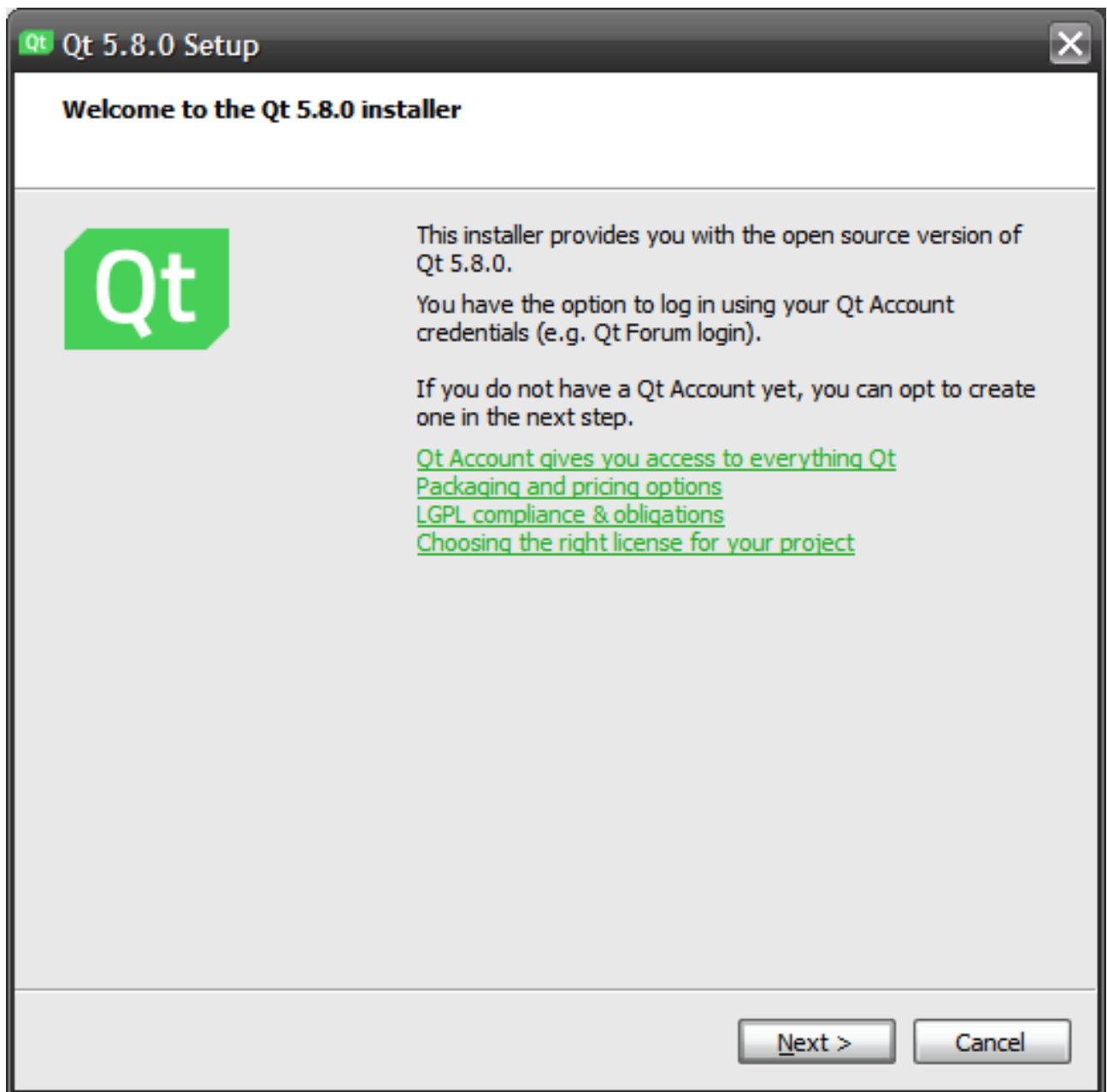
QtSDK is an open source software library designed to provide professional user interfaces for C++. It comes with a mature integrated development environment called Qt Creator which can be used to develop both Qt and Non-Qt applications.

As of this writing the latest version is 5.8.0 and can be found here:

http://download.qt.io/official_releases/qt/5.8/5.8.0/qt-opensource-windows-x86-mingw530-5.8.0.exe

If the URL above is no longer valid then you can go to <http://download.qt.io> to find the appropriate version.

After the download has completed double click the executable to begin the installation:



Press the **Next** button to continue

Qt 5.8.0 Setup

Qt Account – Your unified login to everything Qt



Please log in to Qt Account

Login

Email

Password

[Forgot password?](#)

Need a Qt Account?

Sign-up

Valid email address

Password

Confirm Password

☐ I accept the [service terms](#).

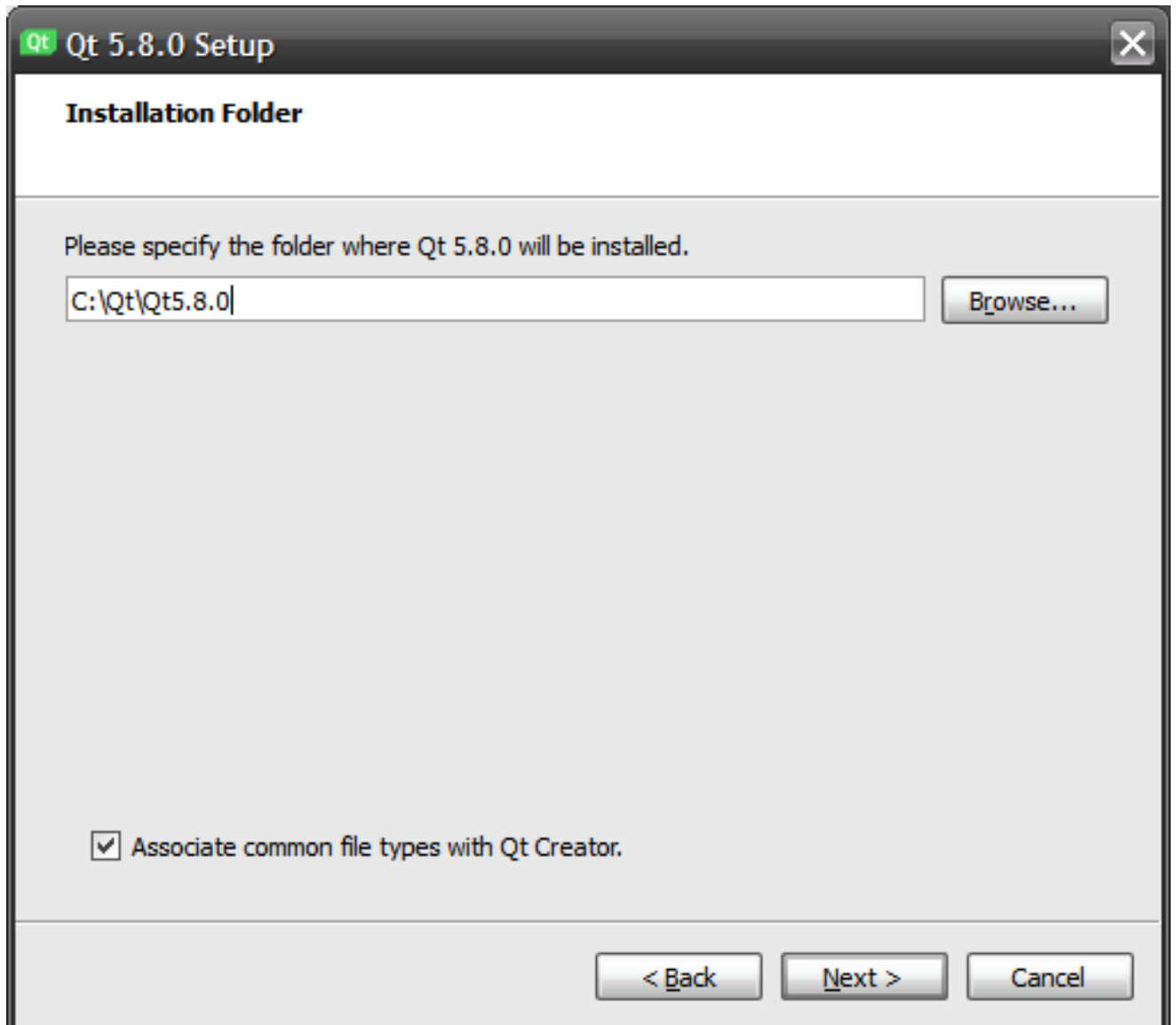
Settings

< Back

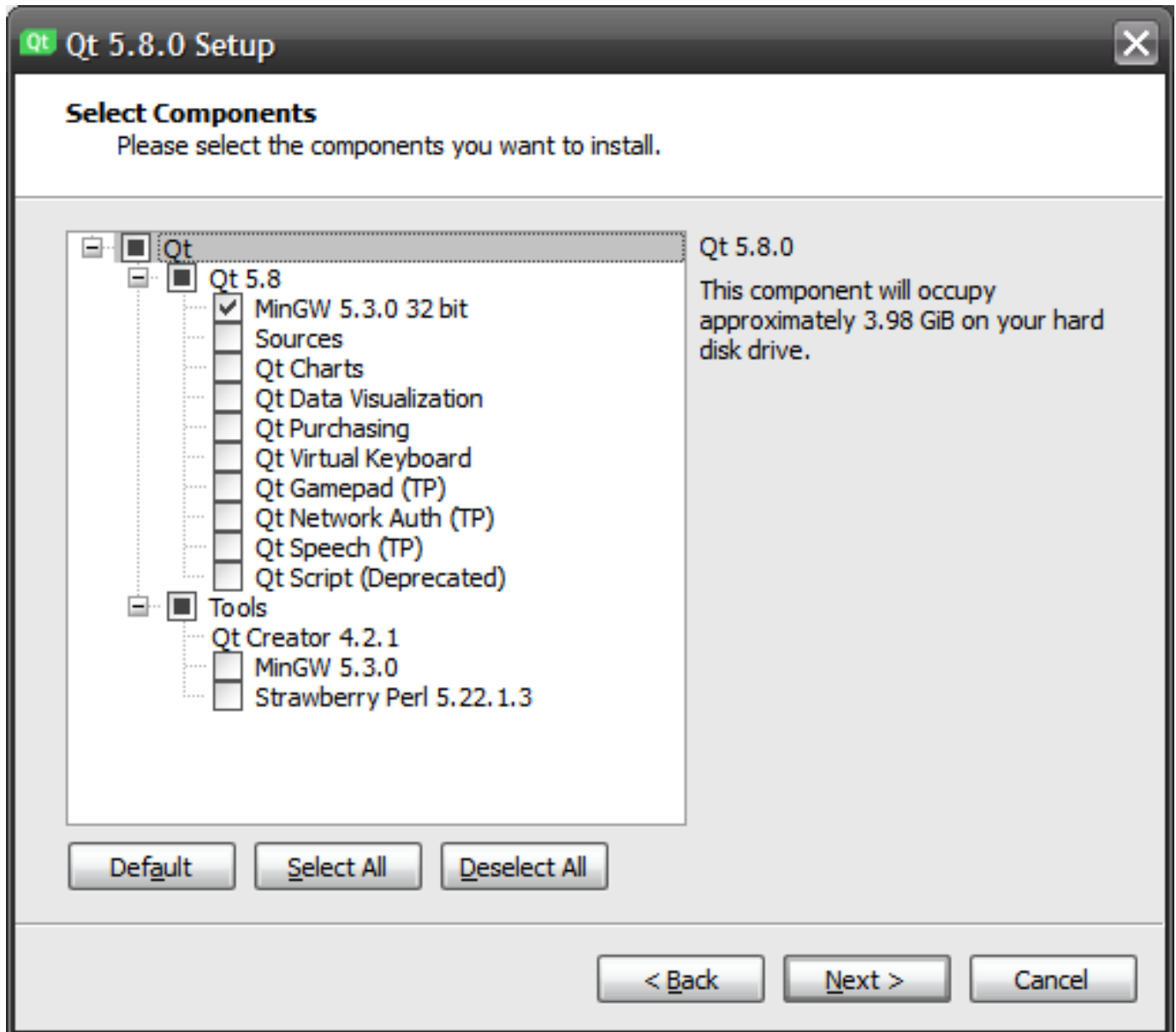
Skip

Cancel

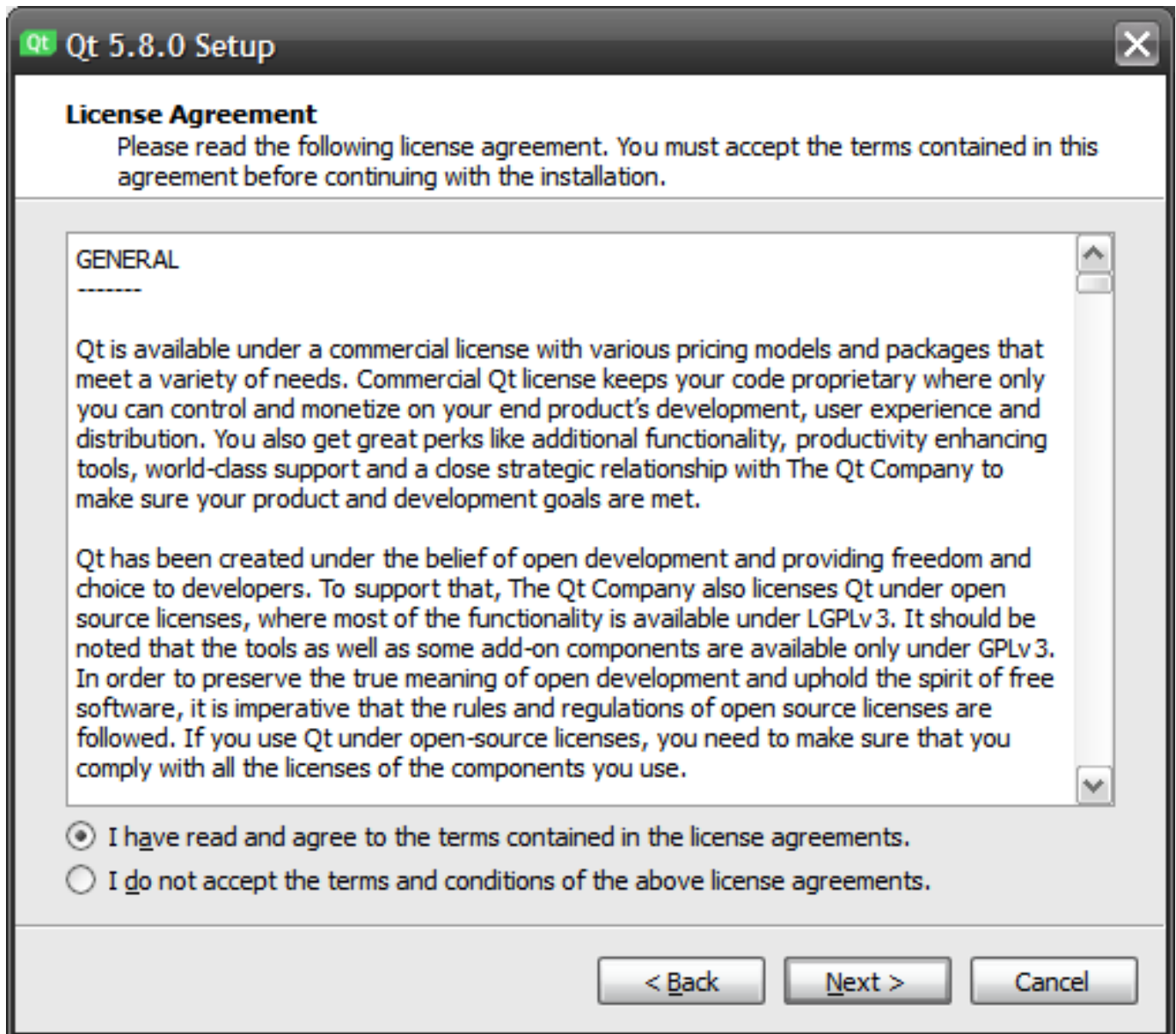
Press the **Skip** button to continue



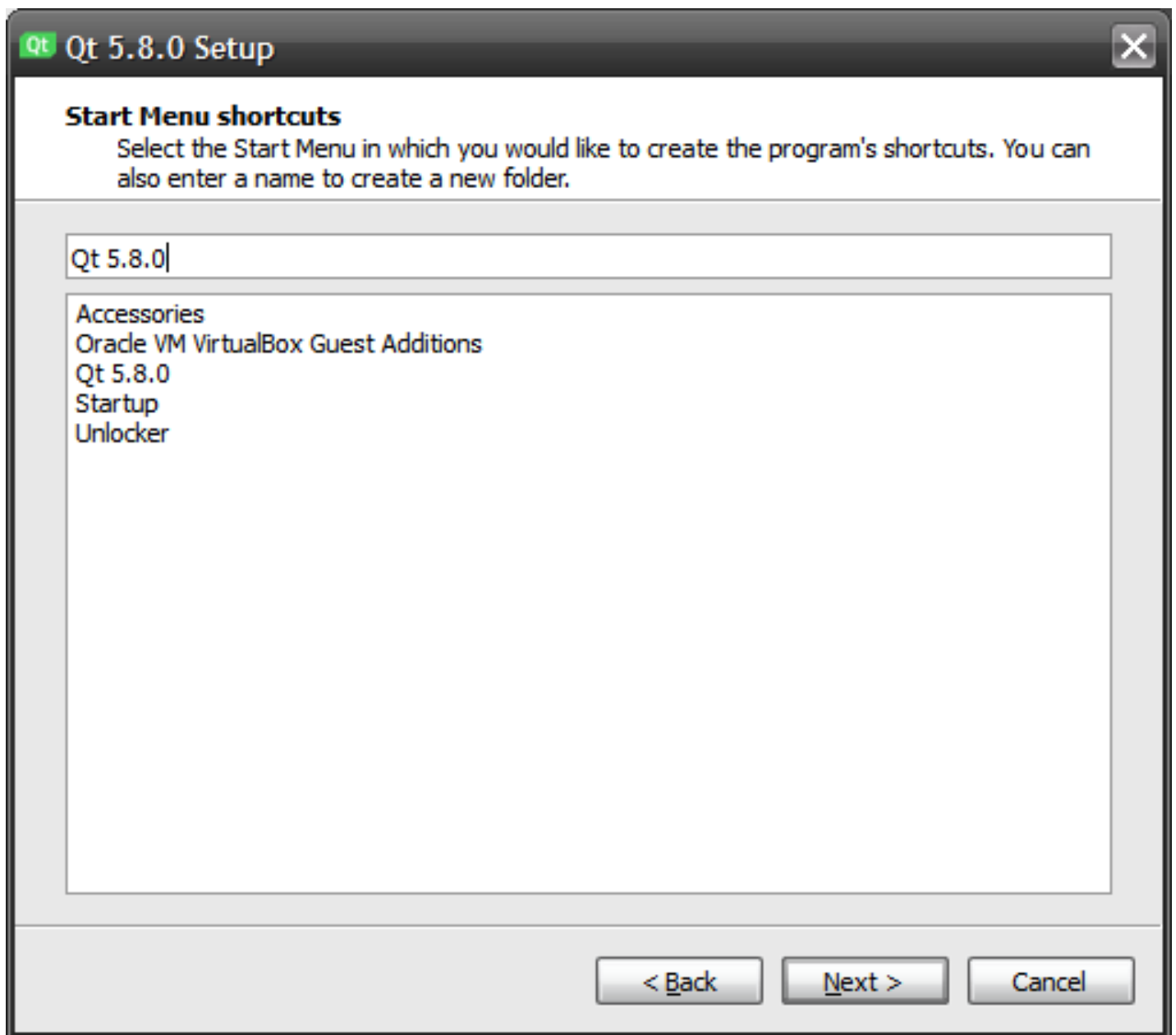
Set the path where you wish to install QtSDK
Press the **N**ext button to continue



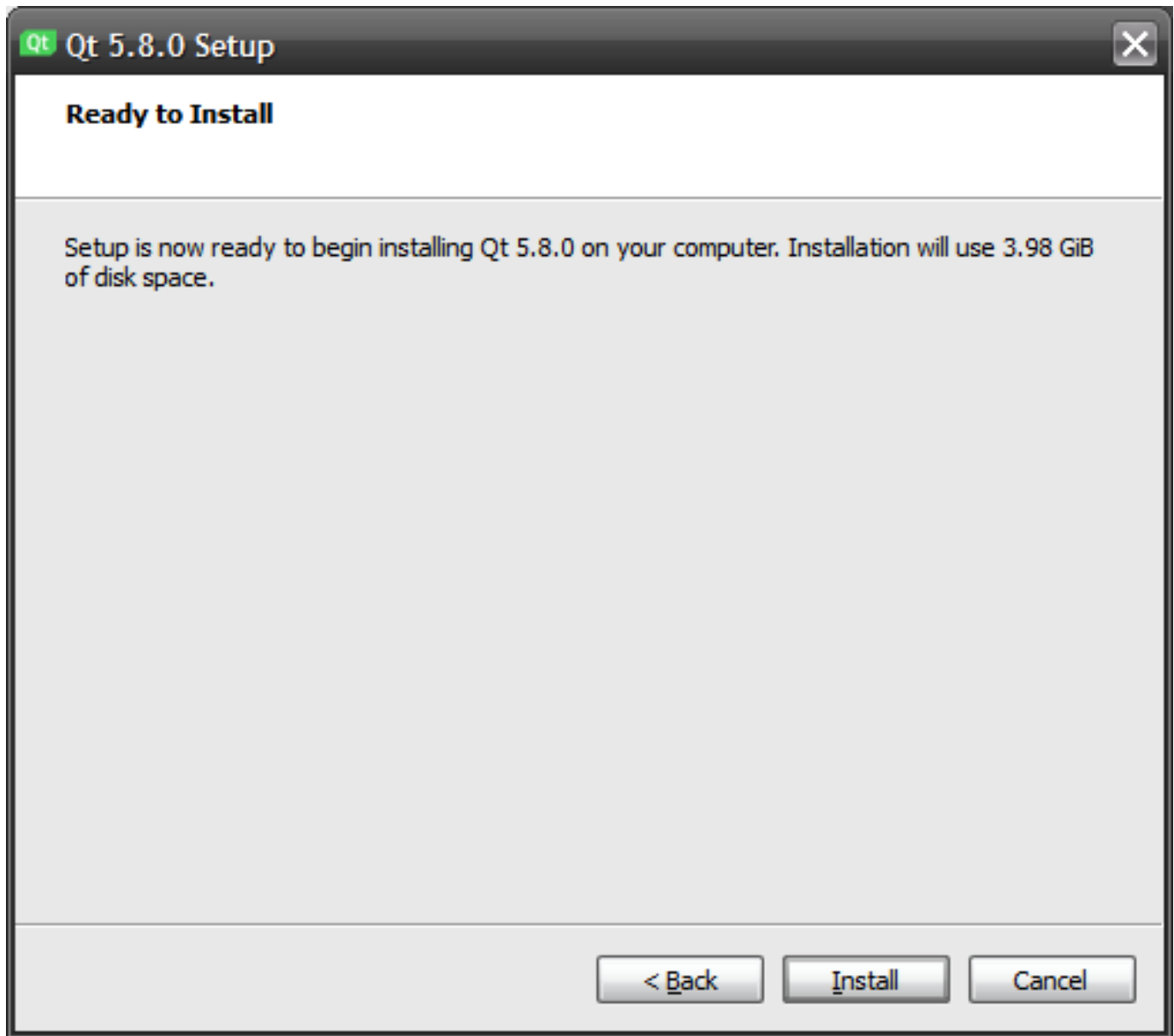
Expand the “Qt 5.8” tree and un-check all the extra Qt components since they won't be used. Ensure that your screen looks like the one above and then press the **Next** button



Click the radio button to accept the license agreement for QtSDK.
Press the **Next** button to continue



Select the "Start Menu" shortcut name
Press the **N**ext button to continue

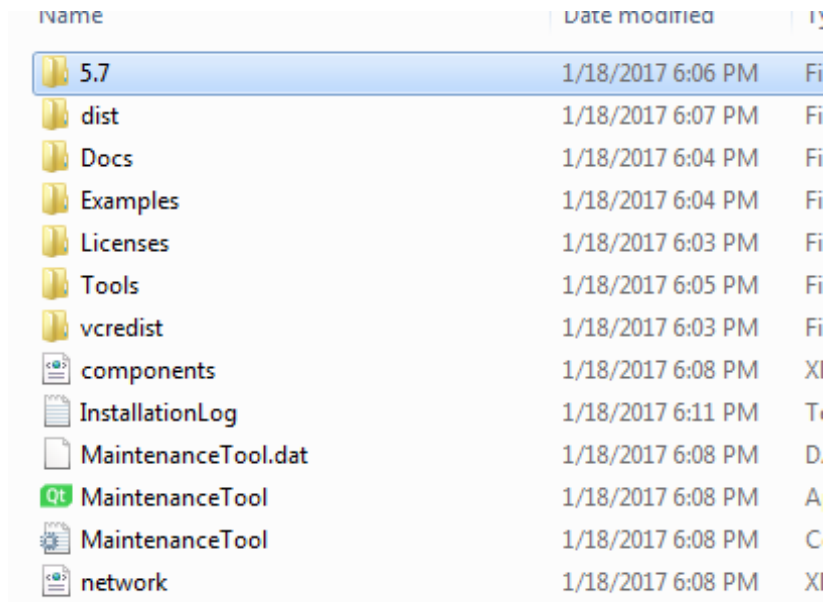


Click the **Install** button to begin the installation.
Installation should take between 5 and 10 minutes depending on machine speed.

NOTE: If you chose to install QtSDK to your user folder and don't have administrative privileges, you may be presented with dialog asking for administrative credentials. You should also be able to safely continue the install by clicking the "Ignore" button.

4.1.3 Configuring QtSDK

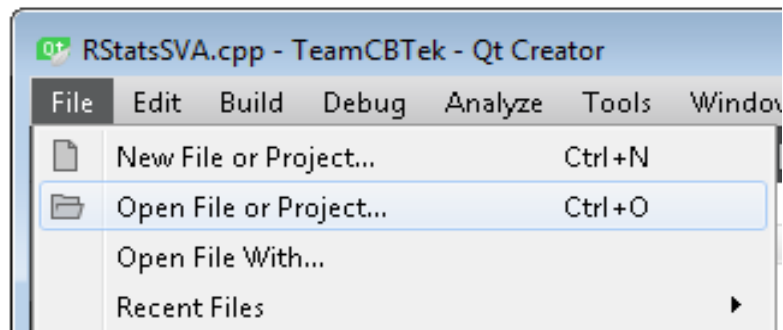
When installation completes locate your installation directory (See the image below. Note you may have a “5.8” folder instead of “5.7”)



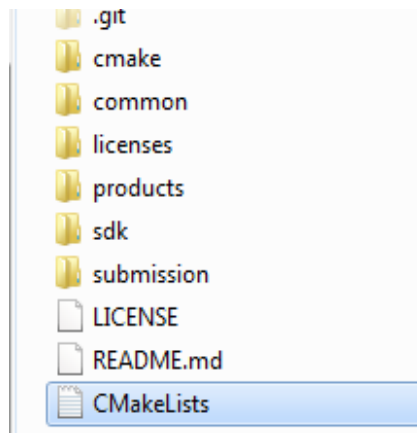
name	Date modified	Type
5.7	1/18/2017 6:06 PM	Folder
dist	1/18/2017 6:07 PM	Folder
Docs	1/18/2017 6:04 PM	Folder
Examples	1/18/2017 6:04 PM	Folder
Licenses	1/18/2017 6:03 PM	Folder
Tools	1/18/2017 6:05 PM	Folder
vcredist	1/18/2017 6:03 PM	Folder
components	1/18/2017 6:08 PM	Folder
InstallationLog	1/18/2017 6:11 PM	Text File
MaintenanceTool.dat	1/18/2017 6:08 PM	Data File
MaintenanceTool	1/18/2017 6:08 PM	Application
MaintenanceTool	1/18/2017 6:08 PM	Configuration File
network	1/18/2017 6:08 PM	Folder

There are three paths that need to be added to the PATH environment variable. This will allow the RAT-STATS project to detect your newly installed Qt version.

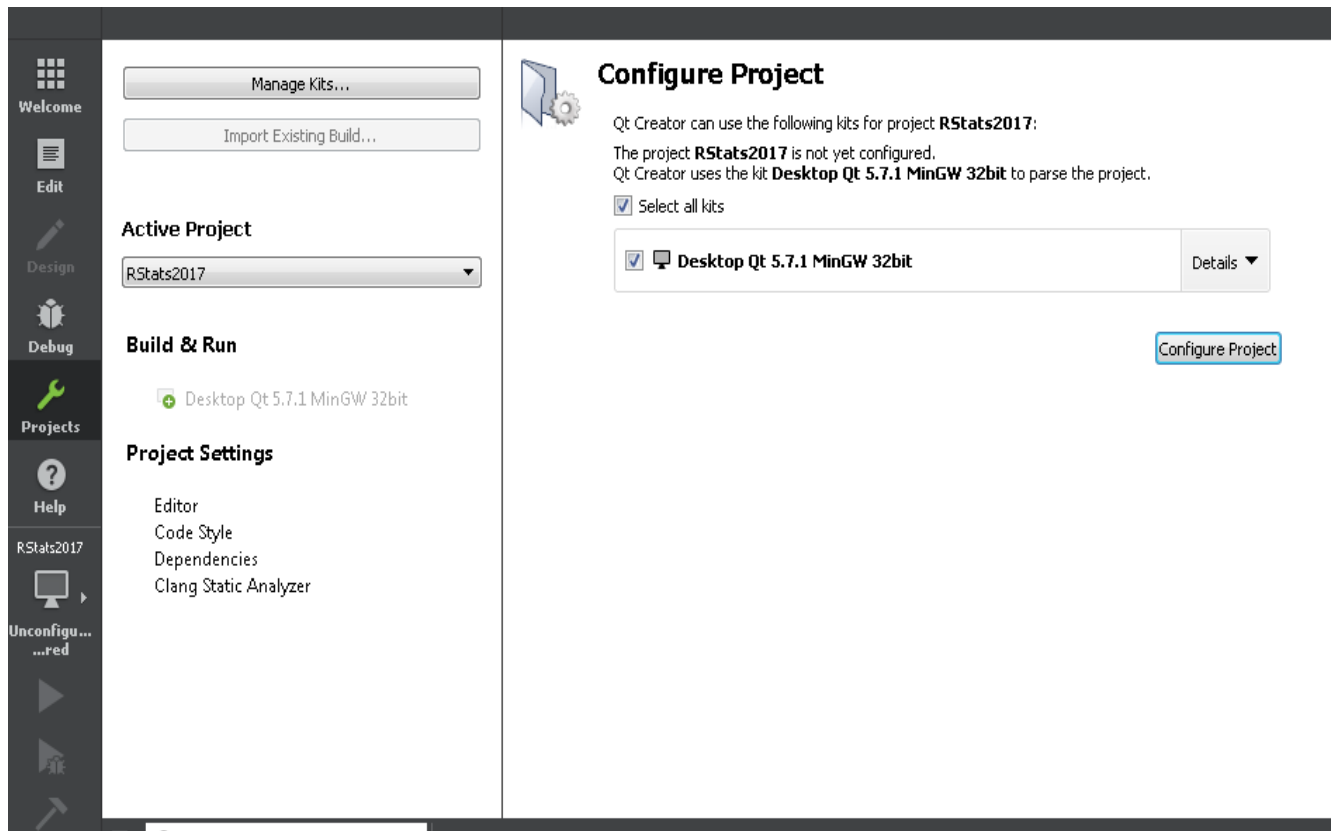
- (1) Path to QtSDK bin folder (e.g C:/Qt/QtSDK5.8.0/5.8/mingw_53_32/bin)
- (2) Path to MinGW bin folder (e.g C:/Qt/QtSDK5.8.0/tools/mingw_530_32/bin)
- (3) Path to QtCreator bin folder (e.g C:/Qt/QtSDK5.8.0/tools/QtCreator/bin)
- (4) Add your paths to PATH, close any open command prompts and (re)start QtCreator.



Once QtCreator restarts click on “File” in the menu bar and select “Open File or Project”



Navigate to the “rstats_source” folder and select “CMakeLists.txt” to open.



Once it opens in QtCreator you will see the CMake configure project screen. Click the “Details” drop down on the far right:



Configure Project

Qt Creator can use the following kits for project **RStats2017**:

The project **RStats2017** is not yet configured.

Qt Creator uses the kit **Desktop Qt 5.7.1 MinGW 32bit** to parse the project.

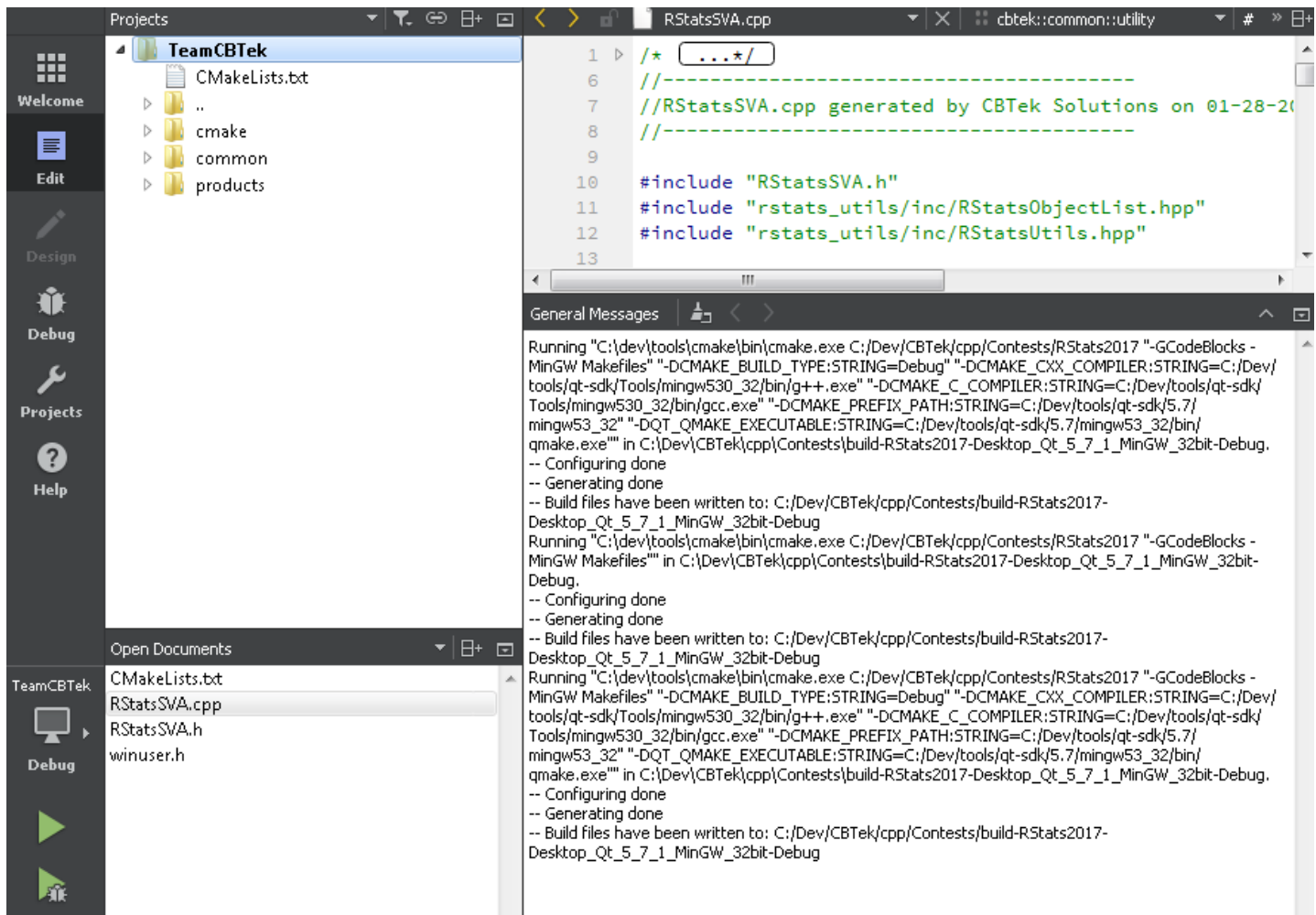
☒ Select all kits

<input checked="" type="checkbox"/>  Desktop Qt 5.7.1 MinGW 32bit	Manage...	Details ▲
<input type="checkbox"/> Default	Desktop Qt 5.7.1 MinGW 32bit-Default	Browse...
<input checked="" type="checkbox"/> Debug	Desktop Qt 5.7.1 MinGW 32bit-Debug	Browse...
<input checked="" type="checkbox"/> Release	Desktop Qt 5.7.1 MinGW 32bit-Release	Browse...
<input type="checkbox"/> Release with Debug Information	32bit-Release with Debug Information	Browse...
<input type="checkbox"/> Minimum Size Release	Desktop Qt 5.7.1 MinGW 32bit-Minimum Size Release	Browse...

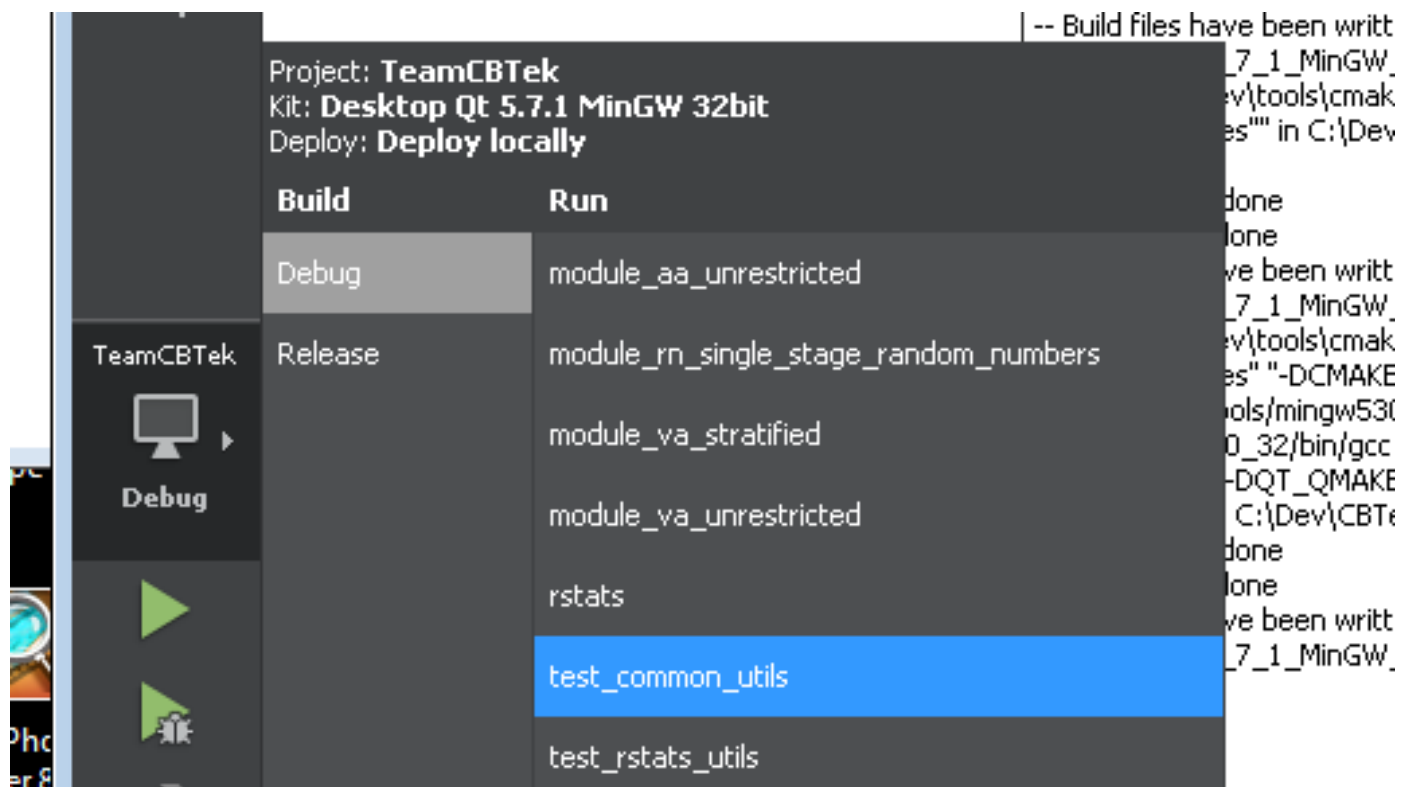
Configure Project

Make sure you only check the “Debug” and “Release” boxes. You can also optionally change the build output folder for the debug/release configurations if desired.

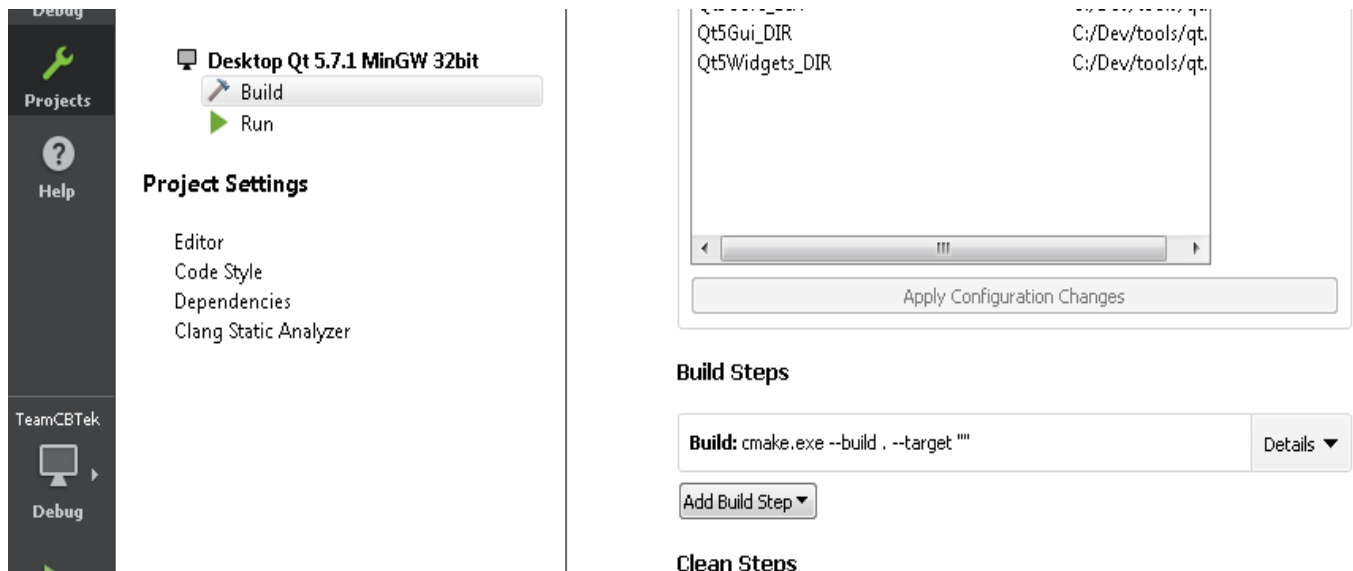
Press the **Configure Project** button to continue.



If everything has been configured correctly you should see the build output below complete without any errors. If you see “Configuring Done” followed by “Generating Done” then you are ready to build RAT-STATS 2017.



You can change which executable you build/launch by clicking the computer icon under “TeamCBTek” toward the bottom left of the screen.



Clicking the “Projects” icon on the far left, takes you to advanced configuration settings for the project. Note the “Build Steps” section on the right. Click the “Details” drop down.

Build Steps

Build: cmake.exe --build . --target all -- -j8

Details ▲

Tool arguments:

Targets:

☐ Current executable

☒ all

☐ clean

☐ common_utils

☐ edit_cache

☐ install

☐ install/local

☐ install/strip

☐ list_install_components

☐ module_aa_unrestricted

☐ module_rn_single_stage_random_numbers

☐ module_va_stratified

▲
≡
▼

Add Build Step ▼

The “Targets” list allows you to select an individual module / executable to build.

Also notice the “Tool Arguments” text box (yours will probably be empty initially).

The “-j8” tells the compiler to use 8 parallel cores to build the project. This results in a much faster build depending on how many cores your CPU has. I highly recommend using this if you can.

4.2 Setting up an environment on Linux

If you wish to setup a Linux development environment for RAT-STATS then the following will be necessary.

4.2.1 Download and install CMake

Regardless of which package manager you use, ensure that it has at least CMake 2.8 available. Please install this using your desired package manager.

4.2.2 Download and install QtSDK

I recommend using the 64-bit QtSDK installer for Linux which can be found at:

https://download.qt.io/official_releases/qt/5.8/5.8.0/qt-opensource-linux-x64-5.8.0.run

32-bit Linux installers are no longer available for download for the latest version of QtSDK. if you must have a 32-bit SDK then the QtSDK 5.5.1 is the last version to support 32-bit. This version (HAS NOT BEEN TESTED) but is available below:

<https://download.qt.io/archive/qt/5.5/5.5.1/qt-opensource-linux-x86-5.5.1.run>

The installation will be very similar to the windows version

Setting up the build environment only has to happen once and does not need much maintenance. If you have any issues setting up your environment we would be glad to help. We can be reached at TeamCBTek@gmail.com.

4.3 Build artifacts and deployment (Windows)

All build are automatically copied to the bin folder located at the same level as the “rstats_source” folder. Each build is separated by configuration (Debug/Release). The following files are required for shipping/deploying modules and executables:

- (1) Qt5Widgets.dll
- (2) Qt5Core.dll
- (3) Qt5Gui.dll
- (4) libgc_s_dw2-1.dll
- (5) libstdc++-6.dll
- (6) libwinpthread-1.dll
- (7) platforms/qminimal.dll
- (8) platforms/qoffscreen.dll
- (9) platforms/qwindows.dll

This are all provided for you in the rstats_binary folder

5. Extending RAT-STATS with additional modules

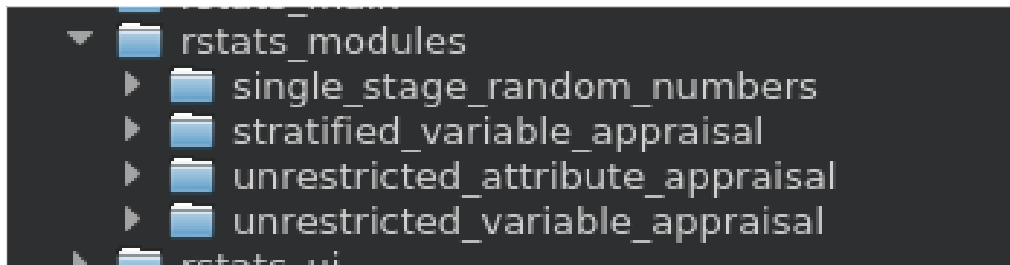
RAT-STATS 2017 was designed from the start to allow for easy, no none sense, modular extending. A **module** is a separate, self-contained executable or script that can be launched from the main menu.

RAT-STATS 2017 separates modules into two distinct categories:

- 1) External Modules
- 2) Internal Modules

5.1 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. They will appear as follows in QtCreator IDE:



5.1.1 Internal Module PROS:

- 1) Can share and utilize existing source code at runtime from other internal modules.
- 2) Can utilize the speed and performance of C++ to implement more advanced time consuming algorithms

5.1.2 Internal Module CONS:

- 1) At the very least internal modules require installation of a C++ compiler as well as CMake. Additional if Qt UI will be used then the QtSDK needs to be installed.
- 2) Requires good knowledge of C++

5.2 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.

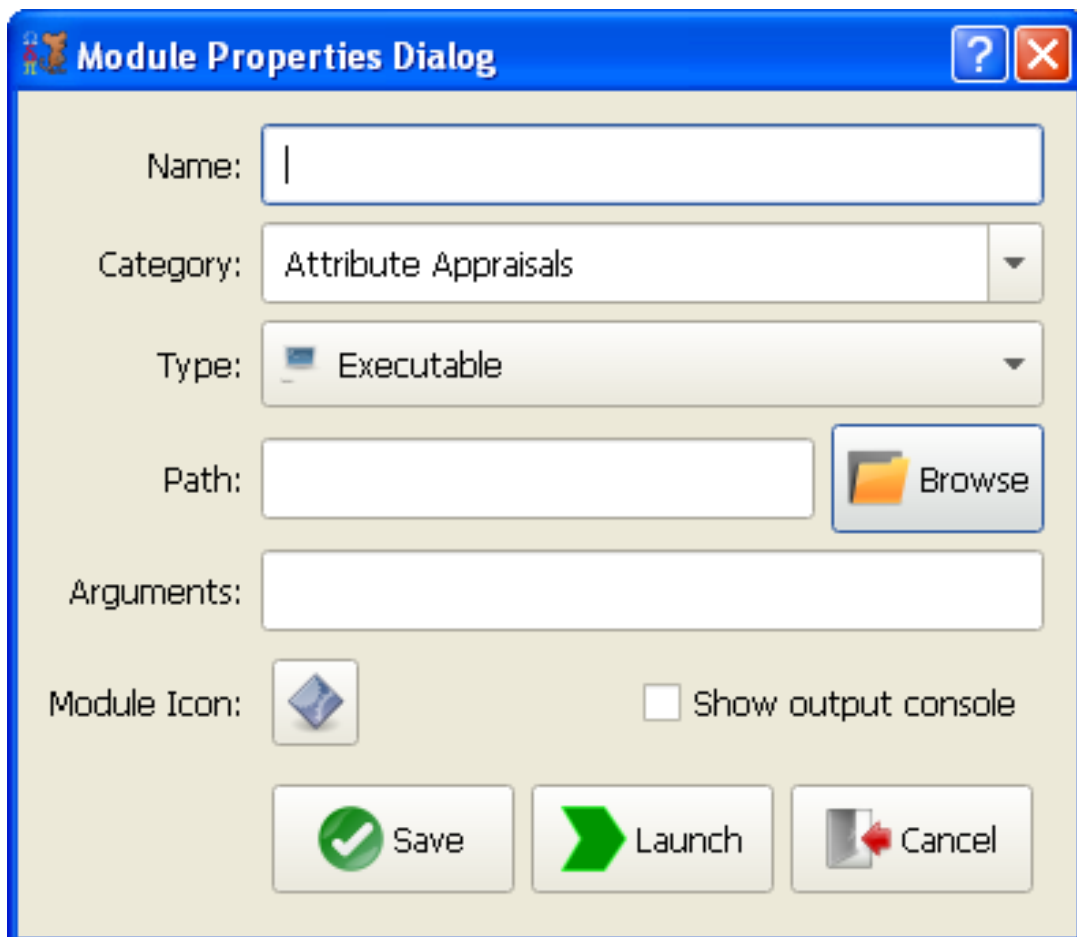
5.2.1 External Module PROS:

- 1) Programmers can utilize any language they desire to extend RAT-STATS
- 2) End users can create custom scripts to extend RAT-STATS without needing access to the code base

5.2.2 External Module CONS:

- 1) Sharing data between different external modules may only occur through command line arguments
- 2) May require additional files to support external scripts / executables

5.3 Adding additional modules to RAT-STATS (Internal or External)

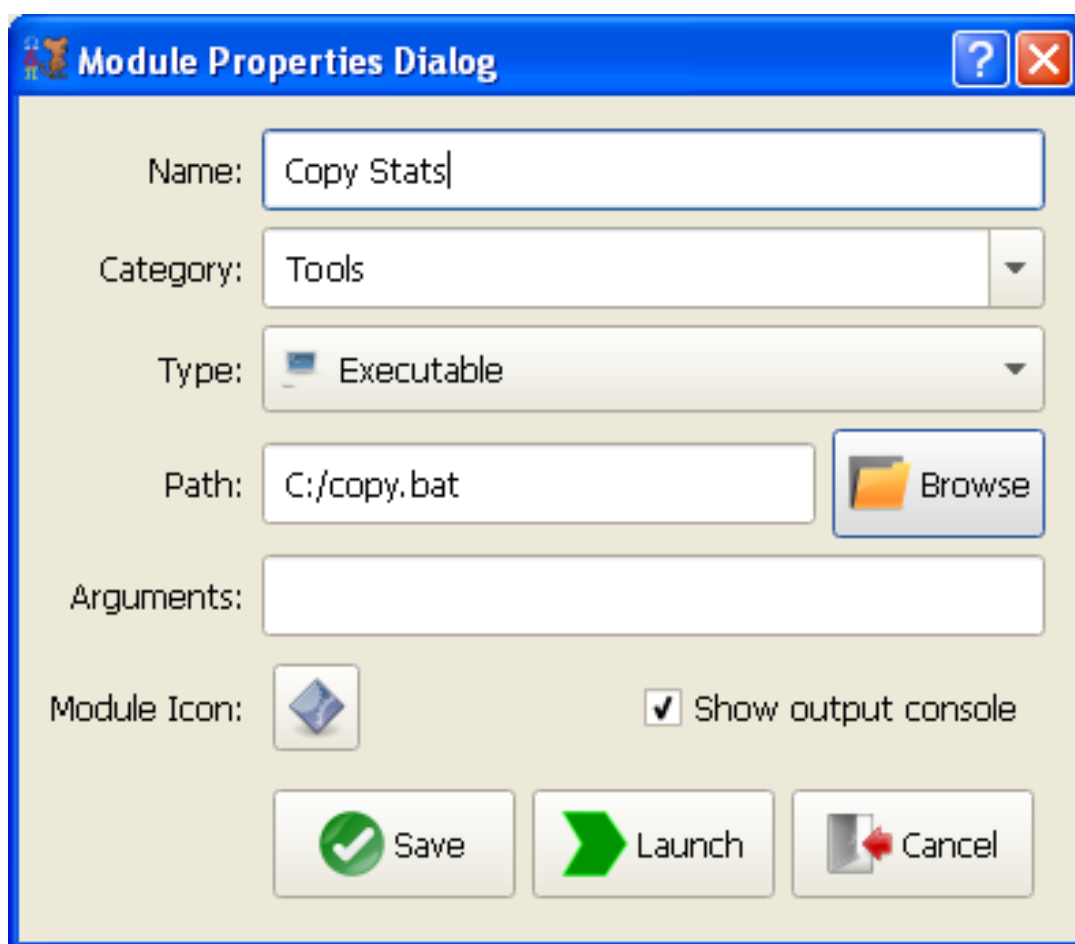


The screenshot shows the 'Module Properties Dialog' window. It has a blue title bar with a question mark and a close button. The dialog contains several fields and controls:

- Name:** A text input field.
- Category:** A dropdown menu currently showing 'Attribute Appraisals'.
- Type:** A dropdown menu currently showing 'Executable'.
- Path:** A text input field next to a 'Browse' button with a folder icon.
- Arguments:** A text input field.
- Module Icon:** A button with a blue diamond icon.
- Show output console:** An unchecked checkbox.
- Buttons:** At the bottom are three buttons: 'Save' (with a green checkmark icon), 'Launch' (with a green arrow icon), and 'Cancel' (with a red X icon).

Adding a module to RAT-STATS is a simple process.

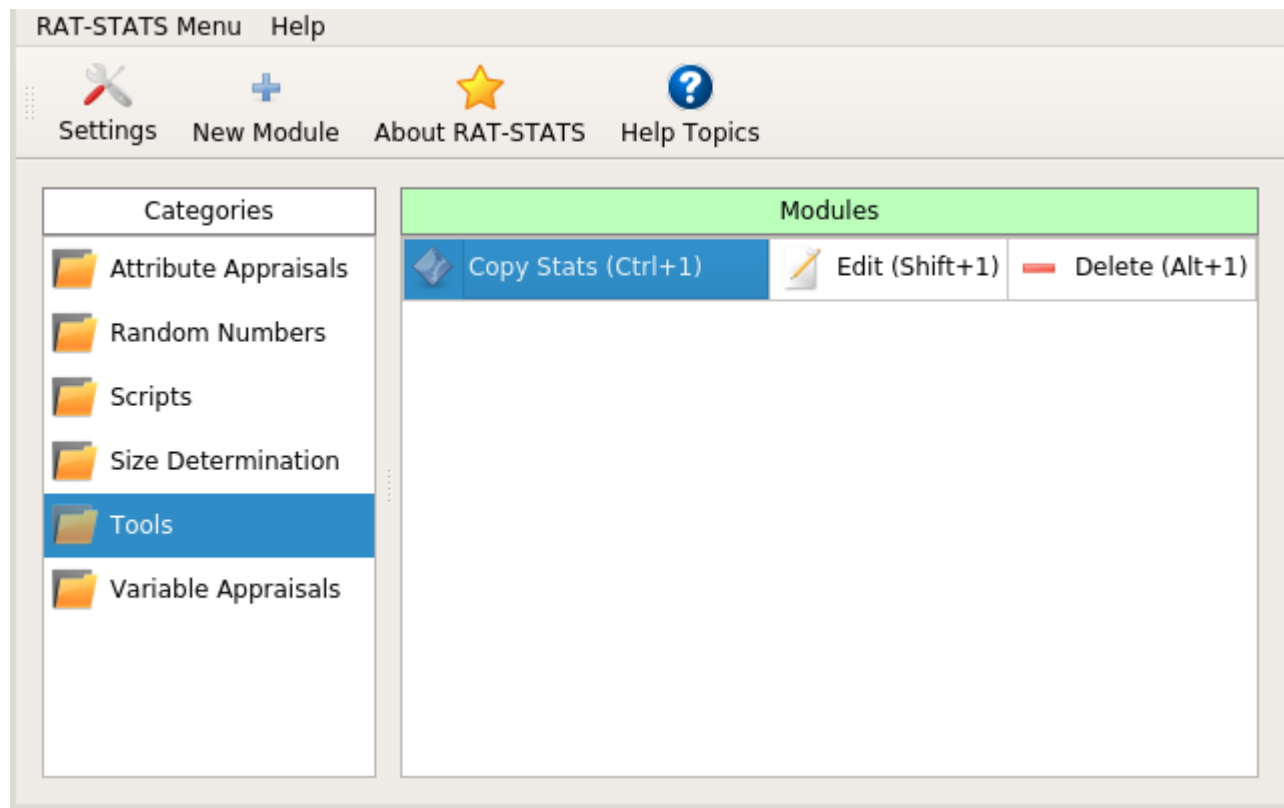
Both internal and external modules are added to RAT-STATS using this method. The dialog above appears after clicking the “**Add module**” button or using the Alt+N shortcut key.



The image shows a Windows-style dialog box titled "Module Properties Dialog". It has a blue title bar with a question mark icon and a close button (X). The dialog contains several input fields and controls:

- Name:** A text box containing "Copy Stats".
- Category:** A dropdown menu showing "Tools".
- Type:** A dropdown menu showing "Executable".
- Path:** A text box containing "C:/copy.bat". To its right is a "Browse" button with a folder icon.
- Arguments:** An empty text box.
- Module Icon:** A button with a blue diamond icon.
- Show output console:** A checked checkbox.
- Buttons:** At the bottom are three buttons: "Save" (with a green checkmark icon), "Launch" (with a green arrow icon), and "Cancel" (with a red X icon).

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.



After clicking the **Save** button a new category called “Tools” is added with the single module I created. 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 are provided three methods to launch, delete and edit modules:

- 1) Use the mouse and double click on the desired cell action
- 2) Use the arrow keys to navigate to the desired cell and press <ENTER>
- 3) Use the automatically assigned shortcut keys to execute an action.

5.4 Creating new internal module projects

It is highly recommended that developers use the ProjectGen (pgen) / SourceGen (sgen) tools for adding new module projects and skeleton source code. These tools are located in the TeamCBTek_Tools.zip archive.

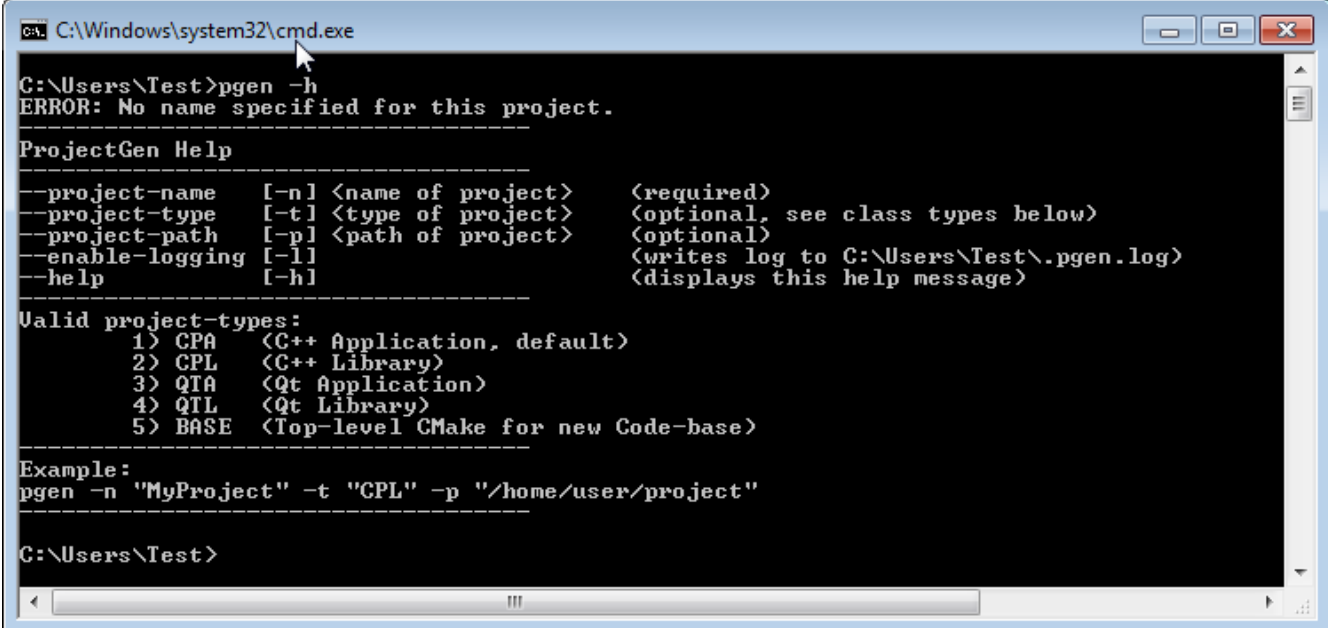
It is recommended that the location of these tools be added to the PATH environment variable so that they may easily be ran from the command line. Note that there are also graphical versions of these tools (**sgen_qt.exe** and **pgen_qt.exe**) that do not require command line interaction.

There are four steps involved in adding an internal module project.

- 1) Generate a new module project using pgen
- 2) Update the generated CMakeLists.txt
- 3) Generate additional classes/forms with sgen
- 4) Update main.cpp to include newly generated classes

5.4.1 Generating a module using pgen

- 1) Open a command window and
- 2) Navigate to rstats_source/products/RAT-STATS/rstats_modules
- 3) Type **pgen -h** (You should see the following help screen)



```
C:\Windows\system32\cmd.exe

C:\Users\Test>pgen -h
ERROR: No name specified for this project.

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 C:\Users\Test\.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"

C:\Users\Test>
```

You have the option of creating either a simple command-line only project (CPA) or a Qt UI based project (QTA)

5.4.2 Generating new command-line module

Command line modules are useful for creating modules that don't need to use any Qt user interface class or libraries.

At the command prompt enter the following to generate your project:

```
pgen -n "MyNewModule" -t "CPA"
```

Remember to replace MyNewModule with your own module name

If the command is successful you should see your new module project folder with four items:

- 1) inc – Will hold all .h/.hpp header and interface definition files
- 2) src – Will hold all .cpp implementation files
- 3) src/main.cpp – Main entry point for application
- 4) CMakeLists.txt – CMake project configuration file

Open the CMakeLists.txt file in an editor:

On lines 35 and 40 the `add_dependencies` and `target_link_libraries` are commented out (using pound symbol). These commands tell CMake which libraries to build and link with your application.

If you wish to use the `common_utils` library then just uncomment those two lines and save the file.

5.4.3 Generating new Qt UI based module

Qt UI based modules are useful if you wish to create a user interface front-end.

At the command prompt enter the following to generate your project:

```
pgen -n "MyNewQtModule" -t "QPA"
```

Remember to replace MyNewQtModule with your own module name

If the command is successful you should see your new module project folder with four items:

- 1) inc – Will hold all .h/.hpp header and interface definition files
- 2) src – Will hold all .cpp implementation files
- 3) src/main.cpp – Main entry point for application
- 4) CMakeLists.txt – CMake project configuration file

Open the CMakeLists.txt file in an editor:

On lines 50 and 55 the `add_dependencies` and `target_link_libraries` commands are commented out (using pound symbol). These commands tell CMake which libraries to build and link with your application. For the purposes of using Qt you must uncomment line 55 (`target_link_libraries`).

If you wish to use the `common_utils` library then just uncomment `add_dependencies` call and add "common_utils" to the end of both commands.

You may also mimic the functionality provided in CMakeLists for the other internal modules that are included with this submission.

5.4.4 Adding additional classes/forms with SourceGen

In the tools section open the sgen_qt.exe program. The form below gives an example of how to generate a simple class skeleton. I used this project to generate the skeleton code for all of RAT-STATS in just a matter of seconds.

The screenshot shows the 'MyModuleFunction - SourceGen Alpha (12/12/2016)' dialog box. It has a dark theme with yellow text and buttons. The 'Class Type' section on the left has a list box with 'Normal' selected, and 'Static', 'Singleton', 'Virtual', and 'QWidget' are also visible. The 'Class Name' field contains 'MyModuleFunction'. The 'Includes' field contains '<vector>, <string>'. The 'Base Classes' field is empty. The 'Namespace' field contains 'oig.ratstats.modules.my_module_function'. The 'Output Dir' field contains 'C:/Users/Test/MyNewModule' with a 'Browse' button next to it. The 'Copyright File' field is empty with a 'Browse' button next to it. The 'Gets/Sets' section has a text area containing 'm_id : std::string' and 'm_value : double'. On the right side, there are buttons for 'About', 'Help', 'Clone', 'Clear', 'View Output Log', 'Toggle Graphics' (checked), 'Overwrite' (unchecked), 'Save to Subfolder' (checked), 'Header' (dropdown set to 'inc'), 'Source' (dropdown set to 'src'), 'UI' (dropdown set to 'src'), 'Save', and 'Exit'.

There are a few important things to consider about the dialog above:

- 1) First set the output directory to the location of your newly created module.
- 2) Class name refers to both the actual class name and the file(s) that will be generated.
- 3) Each item in the "Includes:" section will become in the generated MyModuleFunction.h file:
#include <vector>
#include <string>

4) The namespace defines the package/group a class belongs to. The above namespace will expand to:

```
namespace oig{
namespace ratstats{
namespace modules{
namespace my_module_function{
    //Class will go here
}}}} // end of namespace
```

5) The “Gets/Sets:” text box allows you to define getters/setters and member variables to be generated in the class output. For the example above the following functions and members will be created:

Based on the dialog above the following would be generated in the MyModuleFunction.h file:

```
class MyModuleFunction
{
public:
    const std::string& getId() const;
    double getValue() const;
    void setId(const std::string& value);
    void setValue(double value);
private:
    std::string m_id;
    double m_value;
};
```

The following would be generated in the MyModuleFunction.cpp file:

```
const std::string& MyModuleFunction::getId() const
{
}

double MyModuleFunction::getValue() const
{
}

void MyModuleFunction::setId(const std::string& value)
{
}

void MyModuleFunction::setValue(double value)
{
}
```

6) The “Save to Subfolder” checkbox will save the generated **.h** and **.cpp** files into the **inc** and **src** folders respectively. RAT-STATS 2017 uses this organization style for all its code (minus the contrib libraries).

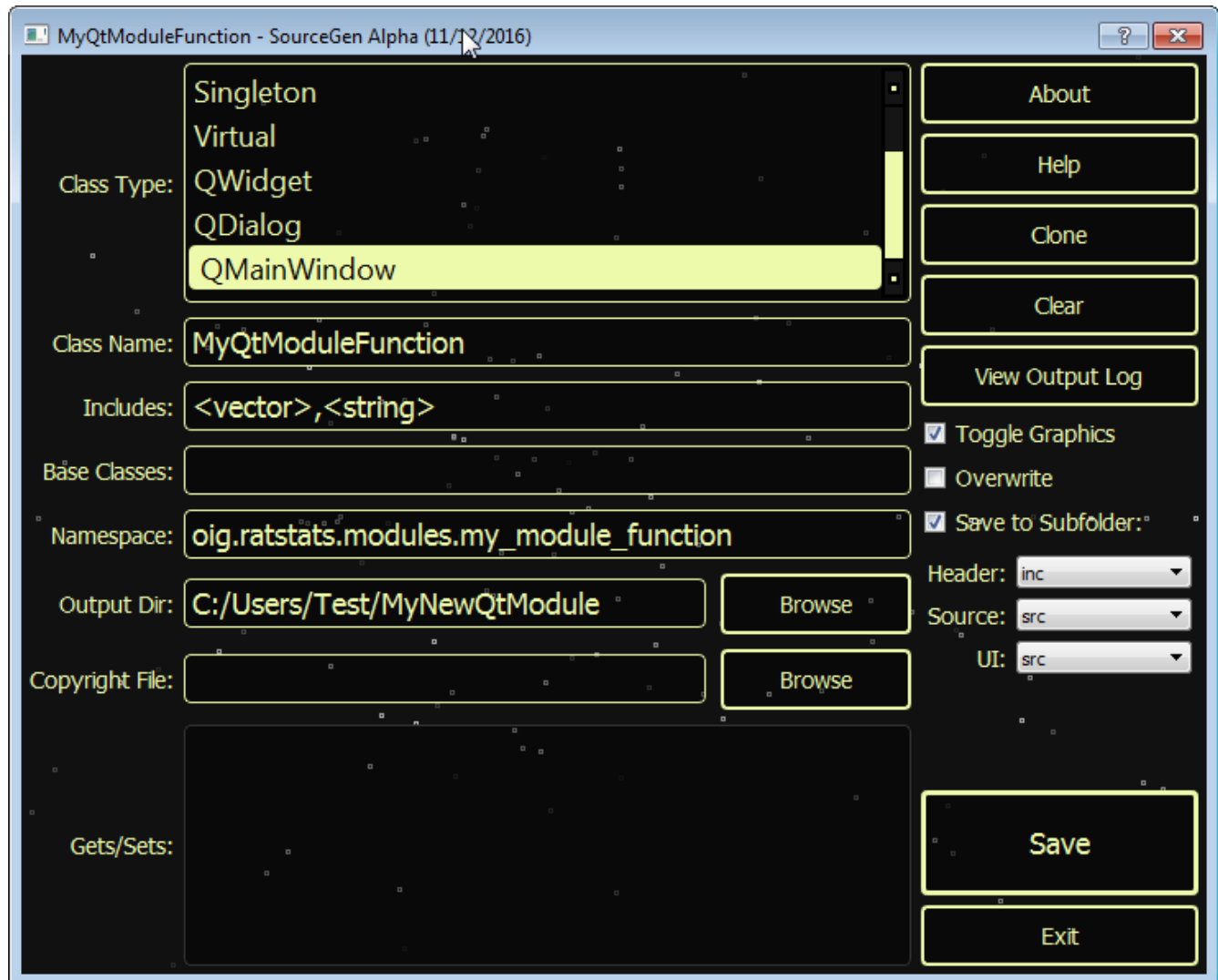
This following example shows how to generate a form class for developing a Qt module.

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. The “Add New Module” dialog uses this.

QWidget – This form type is meant to be embedded within other QDialogs or QMainWindows

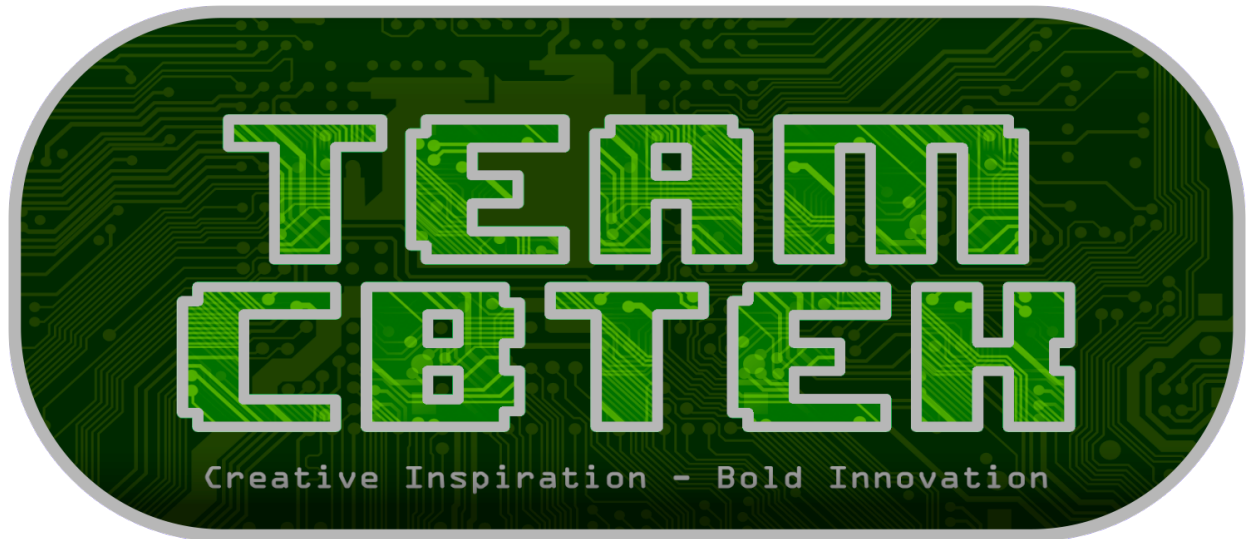
For the example below I am going to go ahead and use QMainWindow.



Things to note about this dialog:

- 1) This dialog will generate a Qt user interface file in addition to C++ class files (.h and .cpp)
- 2) When using the Save to Subfolder feature, be sure to put the UI(.ui) and Source (.cpp) files in the same **src** folder. There is currently a bug in CMake that requires that these two files be side by side.

6. Contact and Support



We take pride in the software that we create and want to continue helping others enjoy using it. For any questions or comments about our software please email us at:

TeamCBTek@gmail.com