

SAS UNIT VALIDATION OF YOUR MACROS TO UPGRADE CALIBRE!!

Roshan Stanly
Manager - Clinical Data Analytics, Genpro Research
roshan.stanly@genproresearch.com

Siji Elsa Paul
Statistical Programmer II, Genpro Research
siji.paul@genproresearch.com

ADVANTAGES OF MACROS

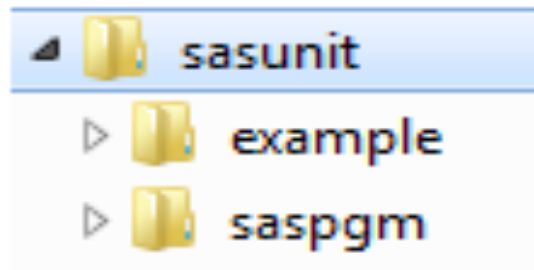
- It saves time and results in an error-free code.
- Make our job easier by re-using similar code multiple times after defining it once.
- Make changes to a variable in a single place and reflect them at multiple locations.
- A program can be driven based on data values.

SASUNIT

- It is a unit testing framework for SAS programs, which is available for free.
- It is used for the development, execution and documentation of tests for SAS programs.
- It is completely written in SAS base code with a few OS commands where necessary.

INSTALLATION AND SET UP

- SAS Unit can be installed from <https://sourceforge.net/p/sasunit/wiki/Getting%20started%20with%20unit%20testing%20of%20SAS%20programs/>.
- Folder structure will look like



Name	Date modified
bin	10/6/2019 1:20 PM
dat	10/6/2019 1:20 PM
doc	10/6/2019 1:20 PM
saspgm	3/21/2020 7:13 PM

SAS UNIT MACROS

```
%initScenario(i_desc=Tests for descriptive.sas);  
/*first test case*/  
%initTestcase(i_object=descriptive.sas, i_desc=Macro parameter indsn cannot be missing)  
%descriptive(indsn      = , outdsn      = out,  
              statlist  = n mean std stderr median ql q3 min max,  
              var        = weight,  
              byvars     = age sex,  
              byvarfmts  = $age. $sex.,  
              compvar    = height);  
  
%endTestcall()  
%assertLogMsg(i_logMsg=%str(USER WARNING: descriptive Macro Aborted. At least one required parameter was not specified.))  
%endTestcase();  
  
/*second test case*/  
%initTestcase(i_object=descriptive.sas, i_desc=Macro parameter outdsn cannot be missing)  
%descriptive(indsn      = sashelp.class , outdsn      = ,  
              statlist  = n mean std stderr median ql q3 min max,  
              var        = weight,  
              byvars     = age sex,  
              byvarfmts  = $age. $sex.,  
              compvar    = height);  
  
%endTestcall()  
%assertLogMsg(i_logMsg=%str(USER WARNING: descriptive Macro Aborted. At least one required parameter was not specified.))  
%endTestcase();  
  
%endScenario;
```

ASSERTS IN SAS UNIT

assertColumns:

It compares the two datasets passed into `i_expected` and `i_actual` using PROC COMPARE. Check fails when there is a mismatch between both datasets.

Example:

```
%assertColumns (i_expected=work.manual,  
                i_actual=work.macro,  
                i_desc=%str(Compare the  
estimated values),  
                i_allow=LENGTH)
```

assertLogMsg:

Verifies that the expected messages, warnings, and errors are sent to the log file correctly. Check fails if an expected log message is not found.

Example:

```
%assertLogMsg (i_logMsg=%str(USER  
WARNING: Macro Aborted, At least one required  
parameter was not specified.))
```

ASSERTS IN SAS UNIT

assertEquals:

Checks the difference between the value of a macro variable to the expected value. This assertion fails when both i_actual and i_expected values differ.

Example:

```
%assertEquals (i_expected = %union(set1=one  
two three four,set2=one two three,unique=T),  
                i_actual   =%str(one two three  
four),  
                i_desc     = Compare Values);
```

assertLog:

Checks the log for errors and warnings. The check fails when the expected number of errors/warnings are not found.

Example:

```
%assertLog ( i_errors  = 1,  
              i_warnings = 0,  
              i_desc= Scan log for errors)
```

ASSERTS IN SAS UNIT

assertRecordExists:

Checks the existence of records satisfying the specified conditions, or else the assertion fails.

Example:

```
%assertRecordExists (i_dataset  
=work.Contracts  
  
,i_whereExpr=%str(ContractType=1)  
                  ,i_desc    =Do we have at least  
one record per contract type 1?);
```

assertTableExists:

Checks the existence of specified datasets, views, or catalogues.

Example:

```
%assertTableExists(i_libref = OUT,  
                  i_memname = LB,  
                  i_target  = DATA,  
                  i_desc    = Dataset with ZERO  
observations Should not exist - LB,  
                  i_not     = 1);
```

ASSERTS IN SAS UNIT

assertReport:

Checks the existence of an expected report file. The assertion fails if the report is not created in the current SAS session. For the comparison of both reports, we require a manual check.

Example:

```
%assertReport (i_actual= &path  
  \manual.xlsx,  
               i_expected=&path \macro.xlsx,  
               i_desc=%str(Compare both XLSX  
  outputs ))
```

assertRecordCount:

Checks whether the expected number of records exist in the specified dataset.

Example:

```
%assertRecordCount (i_libref      = outmacro,  
                    i_memname     = _list,  
                    i_recordsExp  = 0,  
                    i_desc        = Check for a specific  
  number of records);
```

DOCUMENTATION – MAIN PAGE

SASUnit Examples SASUnit ☒ Test Documentation		
Main Page Test Scenarios Test Cases Units under Test		
Properties of this test suite		
Properties of project		
Name of project	&g_project	SASUnit Examples
Root directory	&g_root	G:\Working Folder\Roshan\SAS Unit\sasunit\example
Path to test repository	&g_target	doc\sasunit\en
Program libraries (macro autocall paths)	&g_sasautos	saspgm
Folder for test data	&g_testdata	dat
Folder for reference data	&g_refdata	dat
Folder for specification documents	&g_doc	doc\spec
Configuration of SASUnit		
Path to SASUnit macros	&g_sasunit	G:\Working Folder\Roshan\SAS Unit\sasunit\saspgm\sasunit
Path to SASUnit macros	&g_sasunit_os	G:\Working Folder\Roshan\SAS Unit\sasunit\saspgm\sasunit\windows
SASUnit version	&g_version	2.0.2
SASUnit data base version	&g_db_version	2.0
SAS log of reporting job		doc\sasunit\en\run_all.log
SASUnit language	SASUNIT_LANGUAGE	en
Encoding of HTML reports	&g_rep_encoding	UTF8
SASUnit started with test coverage	SASUNIT_COVERAGEASSESSMENT	Yes
	&g_testcoverage	Yes
SASUnit started in overwrite mode	SASUNIT_OVERWRITE	Yes
Document call hierarchy	&g_crossref	Yes
Call hierarchy includes SASUnit macros	&g_crossrefsasunit	Yes
Test results		
Number of test scenarios (failed)		14 (8)
Number of test cases (failed)		61 (7)
Number of assertions (failed)		155 (12)
Runtime Scenarios		0:00:19
Runtime SASUnit (Scenarios + Assertions)		0:02:02 (14)

DOCUMENTATION – TEST SCENARIOS

SASUnit Examples

- ▶ Scenarios
- ▶ Units under Test
- ▶ Program documentation

Test Scenarios | SASUnit Examples - SASUnit Test Documentation

[Main Page](#) [Test Scenarios](#) [Test Cases](#) [Units under Test](#)

No. ▲	Test Scenario	↕ Program	↕ Last Run	↕ Duration	↕ Result
001	Test examples for assertExternal.sas	saspgm/assertexternal_example_test.sas	21MAR2020:20:13:28	0.7 s	✓
002	Test examples for assertImage.sas	saspgm/assertimage_example_test.sas	21MAR2020:20:13:29	1.8 s	✗
003	Test examples for assertText.sas	saspgm/asserttext_example_test.sas	21MAR2020:20:13:31	0.9 s	✓
004	Tests for boxplot.sas	saspgm/boxplot_test.sas	21MAR2020:20:13:33	3.2 s	□
005	Tests for comparison.sas	saspgm/comparison_test.sas	21MAR2020:20:13:36	2.4 s	✗

DOCUMENTATION – TEST CASES

SASUnit Examples

- ▶ Scenarios
- ▶ Units under Test
- ▶ Program documentation

Test Cases by Scenario | SASUnit Examples - SASUnit Test Documentation

[Main Page](#) [Test Scenarios](#) [Test Cases](#) [Units under Test](#)

No. 001
Scenario Test examples for assertExternal.sas
Program saspgm/assertexternal_example_test.sas
Last Run 21MAR2020:20:13:28
Duration 0.7 s

Test Cases

No.	Test Case	Unit under Test	Last Run	Duration	Result
001	Successful test with one path and one variable	assertexternal.sas	21MAR2020:20:13:28	0.0 s	✓
002	Tests with with two paths as params und file compare command	assertexternal.sas	21MAR2020:20:13:28	0.0 s	✓

DOCUMENTATION – UNITS UNDER TEST

SASUnit Examples

- Scenarios
- Units under Test
- Program documentation

Units under Test | SASUnit Examples - SASUnit Test Documentation

[Main Page](#) [Test Scenarios](#) [Test Cases](#) [Units under Test](#)

Program Library: G:\Working Folder\Roshan\SAS Unit\sasunit\saspgm\sasunit

Unit under Test	Test Scenario	Test Cases	Assertions	Test Coverage [%]	Calling Sequence	Result
assertexternal.sas	001	2	7	78	Calling Called by	✓
assertImage.sas	002	1	8	85	Calling Called by	✗
asserttext.sas	003	1	8	85	Calling Called by	✓

MACRO VALIDATION CHECKLIST

S. No	Testcase	Macro Check	Expected Result
1	* Macro parameter indsn cannot be missing	%count(indsn =, outdsn = out1, var = age);	Log message - "USER WARNING: MACRO ABORTED!!! THE MACRO PARAMETER INDSN IS NOT SPECIFIED!!!".
2	* Macro parameter outdsn cannot be missing	%count(indsn = sashelp.cars, outdsn =, var = age);	Log message - "USER WARNING: MACRO ABORTED!!! THE MACRO PARAMETER OUTDSN IS NOT SPECIFIED!!!".
3	* One existing dataset is passed as macro parameter indsn * Input dataset contains 0 record.	%count(indsn = adae0_count, outdsn = out, var = age);	Log message - "USER WARNING: COUNT Macro Aborted, The dataset (adae0_count) has zero observations.".
4	* One existing dataset is passed as macro parameter indsn.	%count(indsn = adae_count, outdsn = out, var = age);	Derive the dataset in a separate program and compare both the datasets

CONCLUSION

- Macros can help reduce programming effort and improve efficiency and quality.
- Validation of these macros are important as we are dealing with clinical trial data.
- SAS Unit is a great platform to validate these macros properly and for documentation.

THANK YOU

Any Questions ?