SASUnit – Confidence in Coverage

Handling Edge Cases and Ensuring Code Quality & Adaptability Through Unit Tests

Chuck Castillo

Introduction

- What is SAS Unit?
- Adaptability (changing your code)
- Extensibility (adding to your code)
- Versatility (handling dirty data)
- SASUnit – The Report
- Questions? Answers!

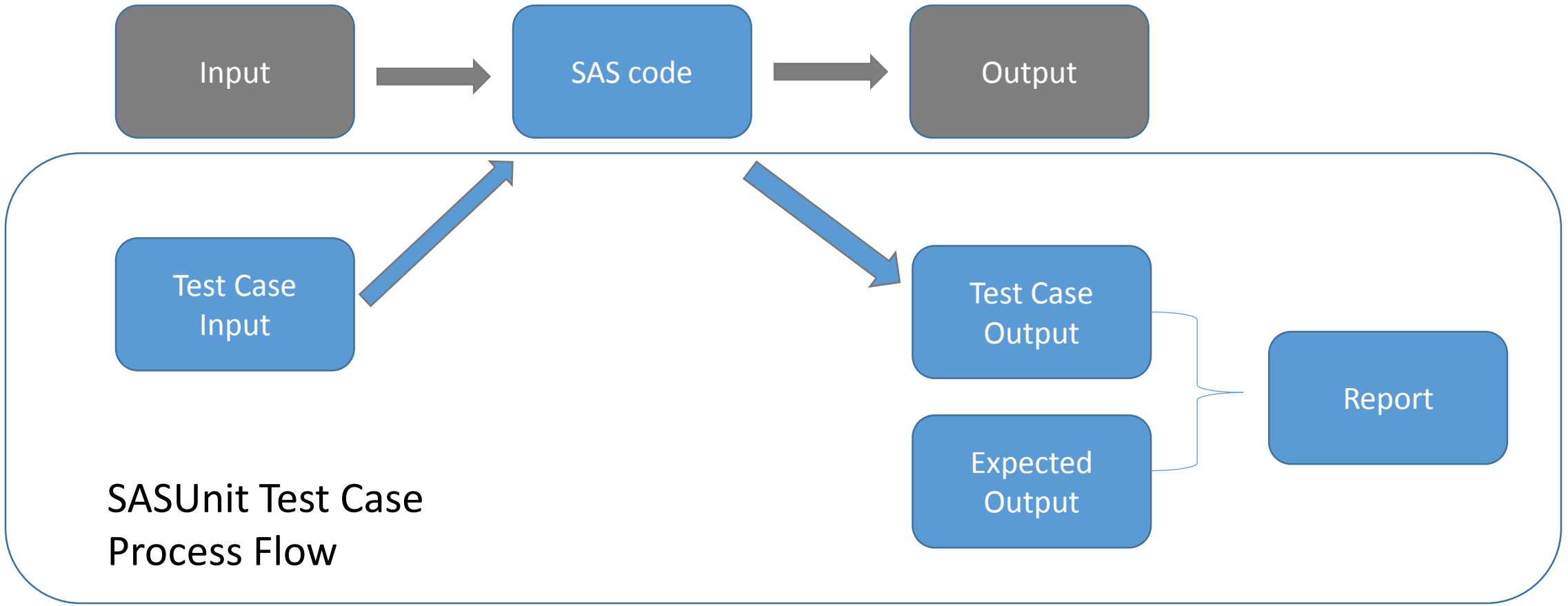
What is SASUnit?

- SASUnit is a collection of programs and macros that are used to execute your code and verify the results
- Verifies the results by inspecting datasets, macrovariables, the log file…
  - Compares “actual” results to the “expected” results
- Generates a report with results (in .html format, .xml format)

What is SASUnit?

Normal
Process Flow

What is SASUnit?



Adaptability

	class_id	student_id	grade
1	1	1	86
2	1	2	65
3	1	3	49
4	1	4	91
5	1	5	77
6	1	6	70
7	2	2	80
8	2	4	76
9	2	6	80
10	2	7	72
11	3	1	99
12	3	3	95
13	3	5	81
14	4	5	92
15	4	6	27

Grades

	student_id	student_last_name	student_first_name
1	1	Anderson	Sean
2	2	Beharry	Jon
3	3	Fraser	Russ
4	4	Gorman	Ian
5	5	Sepers	Gerit
6	6	Vaughan	Roger
7	7	Virag	Isaac

Students

	class_id	class_name
1	1	Biology
2	2	English
3	3	History
4	4	Physics

Classes

Adaptability

- SASUnit allows you to modify your code while ensuring the output doesn’t change.
  - Dealing with new requirements
  - Need to speed up performance

Adaptability

```
/* Sort grades and students by student_id to prepar
PROC SORT data=DAT.grades
          out=WORK.grades_by_student;
    BY student_id;
RUN;
PROC SORT data=DAT.students
          out=WORK.sorted_students;
    BY student_id;
RUN;

/* Merge grades and students */
DATA WORK.grades_and_students;
    MERGE WORK.grades_by_student (in=G)
          WORK.sorted_students (in=S);
    BY student_id;
    IF G and S;
RUN;

/* Sort grades/students by class_id to prepare for
PROC SORT data=WORK.grades_and_students
          out=WORK.grades_and_students_by_class;
    BY class_id;
RUN;

/* Sort classes by class_id to prepare for the merg
PROC SORT data=DAT.classes
          out=WORK.sorted_classes;
    BY class_id;
RUN;

/* Merge grades/students and classes */
DATA WORK.student_grades_data;
    MERGE WORK.grades_and_students_by_class (in=GS)
          WORK.sorted_classes (in=C);
    BY class_id;
    IF GS and C;

    DROP class_id;
RUN;
```

Adaptability

```
/* Sort grades and students by student_id to prepar
PROC SORT data=DAT.grades
          out=WORK.grades_by_student;
    BY student_id;
RUN;
PROC SORT data=DAT.students
          out=WORK.sorted_students;
    BY student_id;
RUN;

/* Merge grades and students */
DATA WORK.grades_and_students;
    MERGE WORK.grades_by_student (in=G)
          WORK.sorted_students (in=S);
    BY student_id;
    IF G and S;
RUN;

/* Sort grades/students by class_id to prepare for
PROC SORT data=WORK.grades_and_students
          out=WORK.grades_and_students_by_class;
    BY class_id;
RUN;

/* Sort classes by class_id to prepare for the merg
PROC SORT data=DAT.classes
          out=WORK.sorted_classes;
    BY class_id;
RUN;

/* Merge grades/students and classes */
DATA WORK.student_grades_data;
    MERGE WORK.grades_and_students_by_class (in=GS)
          WORK.sorted_classes (in=C);
    BY class_id;
    IF GS and C;

    DROP class_id;
RUN;
```

No.	Test Case	Unit under Test	Last Run	Duration	Result
001	Data testing of Data based steps	merge_data_step.sas	01SEP2016:11:45:05	0.1 s	☑

No.	Assertion	Description	Expected	Actual	Result
001	assertLog	Scan log for errors	Errors: 0, Warnings: 0	Errors: 0, Warnings: 0	☑
002	assertColumns	Testing the data is as expected	Table listing DSLABEL LABEL COMPVAR	Table listing Comparison	☑

Adaptability

```
/* Sort grades and students by student_id to prepar
PROC SORT data=DAT.grades
          out=WORK.grades_by_student;
    BY student_id;
RUN;
PROC SORT data=DAT.students
          out=WORK.sorted_students;
    BY student_id;
RUN;

/* Merge grades and students */
DATA WORK.grades_and_students;
    MERGE WORK.grades_by_student (in=G)
          WORK.sorted_students (in=S);
    BY student_id;
    IF G and S;
RUN;

/* Sort grades/students by class_id to prepare for
PROC SORT data=WORK.grades_and_students
          out=WORK.grades_and_students_by_class;
    BY class_id;
RUN;

/* Sort classes by class_id to prepare for the merg
PROC SORT data=DAT.classes
          out=WORK.sorted_classes;
    BY class_id;
RUN;

/* Merge grades/students and classes */
DATA WORK.student_grades_data;
    MERGE WORK.grades_and_students_by_class (in=GS)
          WORK.sorted_classes (in=C);
    BY class_id;
    IF GS and C;

    DROP class_id;
RUN;
```

```
PROC SQL noprint;
    CREATE TABLE WORK.student_grades_sql AS
    SELECT S.student_id, S.student_last_name
            ,S.student_first_name
            ,C.class_name, G.grade
    /* Merge grades and students */
    FROM DAT.grades AS G
    LEFT JOIN DAT.students AS S
        ON G.student_id = S.student_id
    /* Merge grades/students and classes */
    LEFT JOIN DAT.classes as C
        ON G.class_id = C.class_id
    /* Sort by last, first name and class name */
    ORDER BY S.student_last_name
            ,S.student_first_name
            ,C.class_name;
QUIT;
```

No.	Test Case	Unit under Test	Last Run	Duration	Result
001	Data testing of Data based steps	merge_data_step.sas	01SEP2016:11:45:05	0.1 s	☑
002	Data testing of Proc Sql	merge_proc_sql.sas	01SEP2016:11:45:05	0.1 s	☑

Adaptability

```
/* Sort grades and students by student_id to prepar
PROC SORT data=DAT.grades
          out=WORK.grades_by_student;
    BY student_id;
RUN;
PROC SORT data=DAT.students
          out=WORK.sorted_students;
    BY student_id;
RUN;

/* Merge grades and students */
DATA WORK.grades_and_students;
    MERGE WORK.grades_by_student (in=G)
          WORK.sorted_students (in=S);
    BY student_id;
    IF G and S;
RUN;

/* Sort grades/students by class_id to prepare for
PROC SORT data=WORK.grades_and_students
          out=WORK.grades_and_students_by_class;
    BY class_id;
RUN;

/* Sort classes by class_id to prepare for the merg
PROC SORT data=DAT.classes
          out=WORK.sorted_classes;
    BY class_id;
RUN;

/* Merge grades/students and classes */
DATA WORK.student_grades_data;
    MERGE WORK.grades_and_students_by_class (in=GS)
          WORK.sorted_classes (in=C);
    BY class_id;
    IF GS and C;

    DROP class_id;
RUN;
```

```
PROC SQL noprint;
    CREATE TABLE WORK.student_grades_sql AS
    SELECT S.student_id, S.student_last_name
           ,S.student_first_name
           ,C.class_name, G.grade
    /* Merge grades and students */
    FROM DAT.grades AS G
    LEFT JOIN DAT.students AS S
        ON G.student_id = S.student_id
    /* Merge grades/students and classes */
    LEFT JOIN DAT.classes as C
        ON G.class_id = C.class_id
    /* Sort by last, first name and class name */
    ORDER BY S.student_last_name
           ,S.student_first_name
           ,C.class_name;
QUIT;
```

```
DATA WORK.student_grades_hash;
    ATTRIB student_last_name    length=$32;
    ATTRIB student_first_name   length=$32;
    ATTRIB class_name   length=$32;

    SET DAT.grades;

    /* Declare the hashes */
    IF (_N_ = 1) THEN DO;
        /* Students hash */
        DECLARE hash students(dataset:"DAT.students");
        rc = students.defineKey("student_id");
        rc = students.defineData("student_last_name","student_first_name");
        rc = students.defineDone();
        /* Class hash */
        DECLARE hash classes(dataset:"DAT.classes");
        rc = classes.defineKey("class_id");
        rc = classes.defineData("class_name");
        rc = classes.defineDone();
    END;

    rc=students.find();
    rc=classes.find();

    DROP class_id rc;
RUN;

/* Sort by last, first name and class name */
PROC SORT data=WORK.student_grades_hash;
    BY student_last_name student_first_name class_name;
RUN;
```

No.	Test Case	Unit under Test	Last Run	Duration	Result
001	Data testing of Data based steps	merge_data_step.sas	01SEP2016:11:45:05	0.1 s	✓
002	Data testing of Proc Sql	merge_proc_sql.sas	01SEP2016:11:45:05	0.1 s	✓
003	Data testing of Hash	merge_hash.sas	01SEP2016:11:45:06	0.1 s	✓

Extensibility

- SASUnit allows for testing inputs and outputs of one-or more steps in a process.
- Test as much or as little as you want: macrovariables, table attributes, table values…

Extensibility

Scenario No.	003
Scenario	Tests for reporting on students and grades
Program	testscenario/report_test.sas
Last Run	01SEP2016:11:45:08
Duration	5.1 s
Test Case No.	005
Test Case	Test the student summary has the correct attributes
Unit under Test	/mnt/hgfs/shared_sas/sas/sasunit/sasunit_confidenceincoverage/src/report_hodge_schooling.sas
Last Run	01SEP2016:11:45:11

Assertions

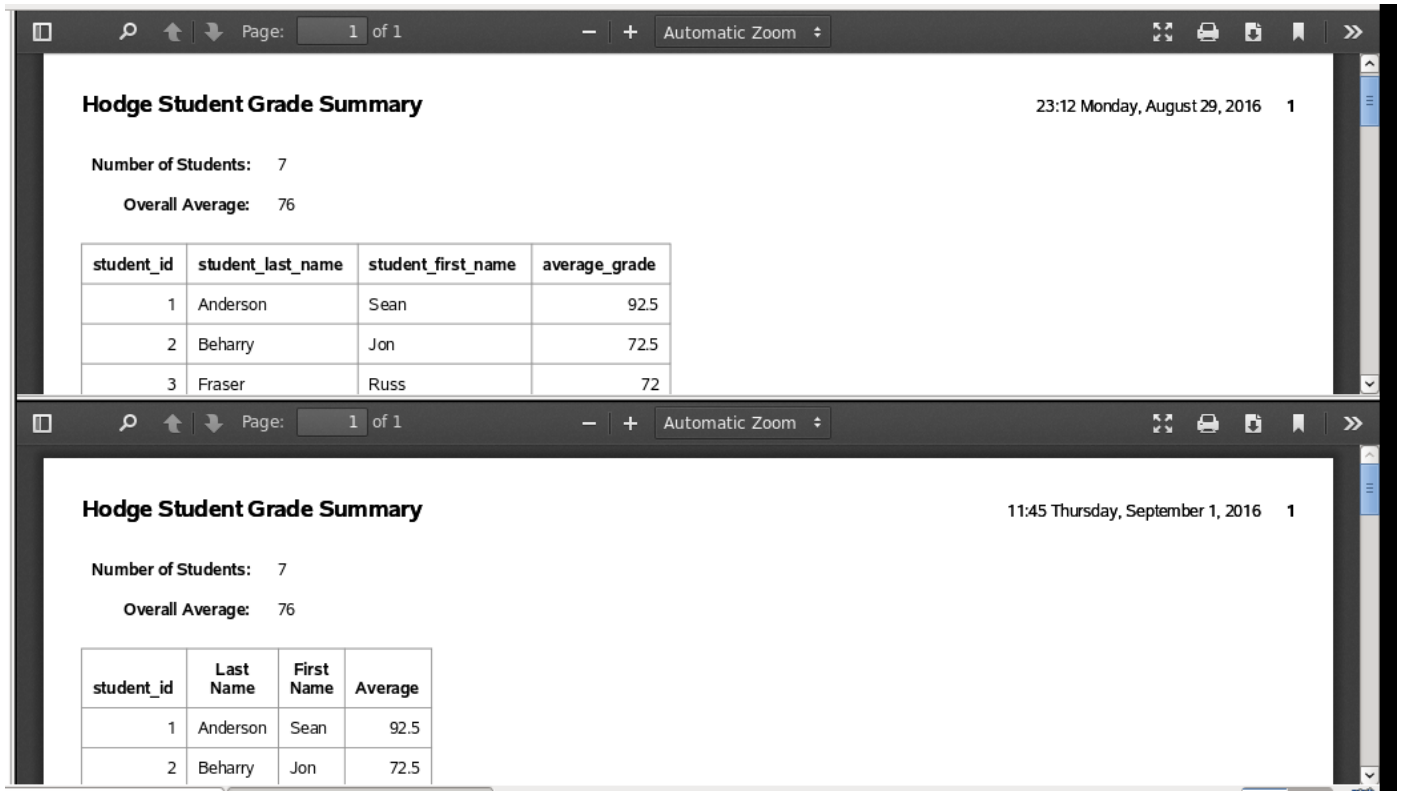
No.	Assertion	Description	Expected	Actual	Result
001	assertColumns	Testing the length of the variables	Table listing INFORMAT FORMAT LABEL VALUE	Table listing Comparison VALUE	
002	assertColumns	Testing the format of the variables	Table listing INFORMAT LENGTH LABEL VALUE	Table listing Comparison VALUE	
003	assertColumns	Testing the label of the variables	Table listing INFORMAT FORMAT LENGTH VALUE	Table listing Comparison VALUE	

Extensibility

No.	Assertion	Description	Expected	Actual	Result
001	assertReport	Testing the pdf report is generated and looks correct	.pdf	.pdf	

Extensibility

No.	Assertion	Description	Expected	Actual	Result
001	assertReport	Testing the pdf report is generated and looks correct	.pdf	.pdf	



Versatility

- Easy to manage expected data. Need to know how to handle outliers.
- Writing unit tests forces you to consider edge cases
  - Missing values, unexpected formats, case sensitivity
- Allows you to be proactive in providing troubleshooting tools (e.g. log messages, error tables)

Versatility

	id_no	stdt_last	stdt_first	district
1	NSA535	Anderson	Sean	North
2	CJB943	Beharry	Jon	Central

Student_Districts

Versatility

No.	Test Case	Unit under Test	Last Run	Duration	Result
001	Test that information is merged by name when data matches	student_regional_information.sas	01SEP2016:11:45:13	0.1 s	
002	Test data is merged by name case insensitive	student_regional_information.sas	01SEP2016:11:45:14	0.1 s	
003	Test that unmatched students are handled as expected	student_regional_information.sas	01SEP2016:11:45:14	0.1 s	

Versatility

003	Test that unmatched students are handled as expected	student_regional_information.sas	01SEP2016:11:45:14	0.1 s	

No.	Assertion	Description	Expected	Actual	Result
001	assertLog	Test that a warning is displayed indicating a student was not found in the district data	Errors: 0, Warnings: 1	Errors: 0, Warnings: 1	
002	assertLogMsg	Test that the expected warning message appears in the log	Message WARNING: One or more students do not have a record in the regional data present	Message found	
003	assertColumns	Testing unmatched students are marked with a district of NO DATA FOUND	Table listing DSLABEL LABEL COMPVAR	Table listing Comparison COMPVAR	

SASUnit – The Report

- Viewing Results
  - Navigate to test logs, test code, unit under test
  - Manual (report) assertion
  - Compare tables – lots of options
  - More…

Resources

- SASUnit - https://sourceforge.net/projects/sasunit/
- Unit Testing - http://martinfowler.com/bliki/UnitTest.html
- Test Driven Development - https://en.wikipedia.org/wiki/Test-driven_development

Questions? Answers!

- (Insert amusing clipart here)

- chuck.castillo@sas.com