Paper TT03

An Experiment on the Use of Altair SLC for Clinical Study Analytics

Mark Holland, Veramed Ltd, London, England

ABSTRACT

Altair SLC (SAS Language Compiler) is a SAS language environment around which there has been much recent discussion and speculation. We hypothesised that SLC is a viable solution for clinical study analytics. In order to test our hypothesis, we designed an experiment which we conducted using the CDISC pilot study. This paper will describe the design, conduct, and results of the experiment, which we feel would be of great value to the statistical programming community.

INTRODUCTION

SLC – SAS Language Compiler, is a software application / platform from Altair, that can take SAS Language programs, compile and execute them against source data from many formats, to generate all the familiar outputs required in the statistical programming process. formally known as WPS (World Programming System) SLC has been in existence for over 20 years and used by over 600 organisations.

Veramed have been working with Altair to see how well SLC could be integrated into their own current reporting process and how well SLC is able to achieve the delivery of the expected data and results. In addition, Veramed were able to provide feedback to Altair to help the upward evolution of SLC.

OBJECTIVES

The objective of the project was to assess how SLC will work as the engine for the Veramed statistical programming environment “out of the box”, and what adjustments might need to be made to the current toolset to ensure data and outputs align with previously generated results. From this we’ll be able to conclude how we feel SLC can be used within our industry as a software platform to use from a point in time moving forward, and how well it would operate executing legacy SAS Language programs.

Our analysis consisted of two main parts.

1. Analysis of selected legacy programs to check compatibility of SAS language programs with SLC
2. Execution of pre-defined SAS language programs, using Veramed stat programming environment / macros / processes

WHAT IS SLC?

SLC is the Altair SAS language compiler, an engine for writing, debugging, compiling and running code written primarily in the language of SAS. From a user perspective this is primarily composed of two parts.

1. Programming Interface
2. SLC

PROGRAMMING INTERFACE.

There are two user interfaces available.

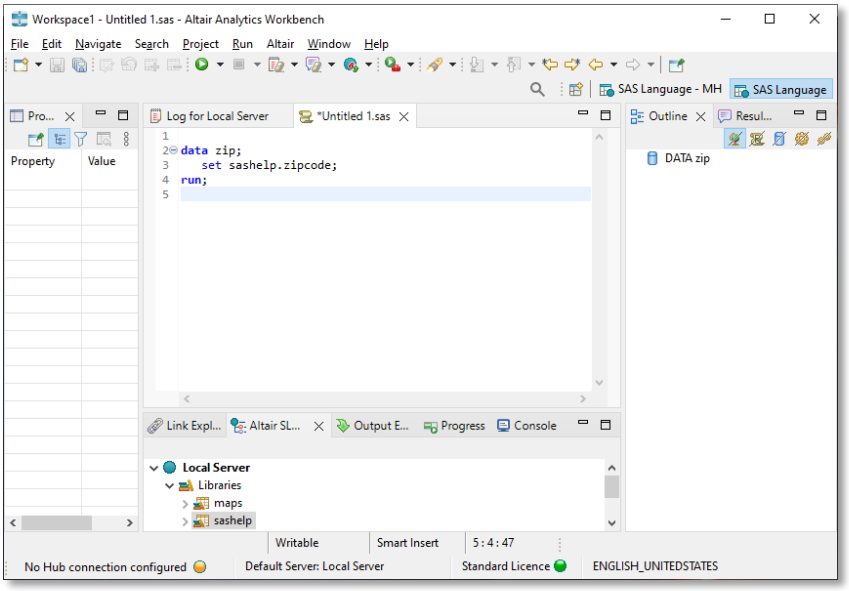
1. Locally installed IDE – “Altair Analytics Workbench”
2. Web based IDE running in VSCode

It is the former that was evaluated by Veramed, and described below.

The Altair Analytics Workbench is based on the Eclipse IDE (https://eclipseide.org/) and has been tailored by Altair to integrate directly with the SLC engine. The IDE provides the programming interface in a manner that will be familiar to many SAS language programmers. Primary features include (but are not limited to):

- Program Editor screen(s), with full colour coding and interactive syntax content.
- Log window for viewing streamed execution logs
- Direct access to all temporary and permanent data files generated during program execution
- Code submission to either local, or central SLC services, where the syntax compilation takes place
- Fully customisable views of all panel locations

In addition, the SLC engine integrates and interleaves R and Python code, and there is also a Python API where SLC SASs language functions can be call from a Python scrip (these were not evaluated by Veramed)



SLC

SLC is the engine for the whole system, and where the Analytics interface submits the SAS language code for compilation and execution.

As one would expect this engine can reside on your local machine, or can be remote across a number of different platforms. In our use we have the SLC engine installed locally on our laptops, where we use it both in interactive and batch modes.

PART 1: PROGRAM ANALYSIS

The initial part of the project was determining how our current programs would operate within SLC. Altair are very transparent that currently not all SAS language code is supported, and to help with the use of SLC, especially with legacy SAS language programs, the Code Analyser is provided within the Altair Analytics workbench. This allows the user to interrogate existing SAS language programs to understand more about how well they may operate within SLC. This focuses on two separate areas.

1. Program Compatibility
   This will give an indication of whether your existing SAS language programs contain any language elements that are not currently supported by SLC. This will deliver a spreadsheet with both a summary of all issues found, and also a full breakdown of all programs scanned, and which of those programs contain unsupported syntax.

Summary:

Program Compatibility Summary	
Spreadsheet produced by Altair Analytics Workbench™ program analyser version 5.24.5.39	
Please note that due to the nature of the SAS language it is not possible to guarantee these static analysis results are 100% accurate	
Distinct Problems	2
Individual Problems	8
Source files analysed	231
Ignored files	404
Analysed files with problems	6
Analysed files without problems	225
Unsupported procedure options	4
Unsupported procedures	4

Detail:

Category	Language Element	Status	Program Name	Line Number
Procedure option	B option on PROC COMPARE	Unknown	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_style\share\macros\s_compare.sas	790
Procedure option	B option on PROC COMPARE	Unknown	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_style\share\macros\s_compare.sas	802
Procedure	XSL	Unknown	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_style\define\adam\code\xml_2_html.sas	38
Procedure	XSL	Unknown	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\define\adam\code\xml_2_html.sas	38
Procedure	XSL	Unknown	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\define\SDTM\code\xml_2_html.sas	38
Procedure option	B option on PROC COMPARE	Unknown	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas	790
Procedure option	B option on PROC COMPARE	Unknown	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas	802
Procedure	XSL	Unknown	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_style\define\SDTM\code\xml_2_html.sas	38

2. Language Usage
   This looks at every language element in the selected programs, and indicates which elements are supported, and which are not.

Summary of Language Usage

Please note that due to the nature of the SAS language it is not possible to guarantee that these static analysis results are 100% accurate

Procedures and Data Steps	Reference Count	Statement Name	Reference Count	Option Name	Reference Cou	First Reference Location
PROC COMPARE	8					Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(637,15)
				B Option	*4*	*Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(790,22)*
				BASE Option	2	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(637,23)
				BRIEFSUMMARY Option	2	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_filecomp.sas(251,49)
				C Option	6	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(637,47)
				COMPARE Option	2	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_filecomp.sas(251,34)
				CRITERION Option	2	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(806,51)
				DATA Option	2	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_filecomp.sas(251,22)
				NOPRINT Option	4	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(637,59)
				OUT Option	4	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(637,67)
				OUTNOEQUAL Option	2	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(637,86)
		ID Statement	2			Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\share\macros\s_compare.sas(809,37)
PROC CONTENTS	18					Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\define\share\macros\d_varlength.sas(77,10)
				DATA Option	18	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\define\share\macros\d_varlength.sas(77,19)
				NOPRINT Option	18	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\define\share\macros\d_varlength.sas(77,71)
				OUT Option	18	Z:\Users\mark.holland\projects\Veramed\C999-123\re_example_FSP\define\share\macros\d_varlength.sas(77,37)

REPORT REVIEW

Both these reports can produce a lot of information to review, but one is able to work through these reports methodically by grouping like information together. It provides paths to all programs that have been analysed, so it is simple to review the original syntax for anything that is flagged, thus allowing the full context of and messages to be understood.

It is acknowledged that the code analyser will not be 100% accurate in finding all incompatible code. It is never going to be able to execute the code, especially to resolve macros variables, so it can also produce false positives, but in our experience it has always been clear when this is the case on review of the source programs.

Example of syntax not currently supported by SLC:

- Procedure PROC XSL

```
proc xsl
  in ="<path>\define.xml"
  xsl="<path>\define2-1-0.xsl"
  out="<path>\define.html";
run;
```

- DM statements (Considered erroneous syntax)

```
dm 'log;clear;';
```

  Similar functionality can be achieved through the IDE options
  e.g,. clearing logs on interactive submission

- Aliases within procedures

```
proc compare B=<data1> C=<data2;
```

  Alias C (COMPARE) is supported
  Alias B (BASE) is not supported

- Programmatically obtaining the currently executing program (in interactive mode)

```
%let path=%sysget(SAS_EXECFILEPATH);
```

  Currently there is no equivalent for SAS_EXECFILEPATH in interactive mode, but SYSIN is available during execution in batch mode.

As part of our investigations, we have been feeding all omissions back to Altair, and they have been open to all recommendations and are progressing to update SLC to enable further improvements in compiling all syntax.

PART 2: PROGRAM EXECUTION AND ADAPTATION

This is the main part of the project, where we took a suite of test programs and internal macros, and using CDISC dummy study data to assess how well SLC can support our reporting process, and which area may need adaptation. We then took our programs and executed them within SLC, in both interactive mode and batch mode, to see how well they ran, and to share feedback directly to Altair with our recommendations.

This part of the project was split into 3 specific subsections.

1. Data Processing
2. Output / Presentation
3. Statistical Analysis

DATA PROCESSING

Process Overview.

The Veramed data process follows a classic model of:

- Dataset specification to manage metadata
- SAS language programs and macros to extract and use the metadata
- Domain / dataset specific SAS language programs utilising the above metadata and user written code to generate CDISC datasets.

Across all our internal macros and example programs, there were only three areas where we identified adjustment was needed to ensure cross compatibility with SLC.

1. Syntax: Use of format E8601DT.:
   Source data, was ISO8601 variable (LB.LBDTC) without seconds (e.g. `2024-09-01T12:25`)

   Original code:

   ```
   ADTM=input(LBDTC,e8601dt.);
   ```

   Updated code:

   ```
   ADTM=input(substr(LBDTC,1,16)!!':00',e8601dt.);
   ```

   Configuration: data file type:
   By default SLC generates data in its own file format. *.WPD. This enables SLC to update records directly within a data file, and manage content as needed. WORK data is also of the format *.WPD.
   .SAS7BDAT files are supported, but can only be read, or written out in one hit. For the user to instigate the creation of .SAS7DBAT, LIBNAME statements need to be updated to include the SAS7BDAT engine.
   Veramed required .SAS7BDAT to be the default dataset type, and wanted to avoid having to add this engine into every LLIBNAME statement we had.
   SLC allows this default to be changed readily by adding the following option into the system configuration file.

   ```
   -ENGINE SAS7BDAT
   ```

   This negates the need for any programs to be adapted to achieve the desired result.

2. Configuration: XPT file type
   SLC has the concept of different engines for .XPT files (as created with PROC COPY and the XPORT engine). By default the engine is V8, whereas our expectation is that XPT 5 var based on a version 5 engine:
   This is another setting easily managed globally, this time in an autoexec file and the use of the following system option:

   ```
   OPTIONS XPORTVERSION=5;
   ```

   This ensures that all current creation of V5 transport / XPT files needs no change to meet submission standards.

SUMMARY

Once the above configuration was set within our SLC instance, we were readily able to execute our ADaM example dataset SAS language programs in SLC with no further adaptation needed, and the results we obtain through SLC matched directly with those we had on file.

OUTPUT / PRESENTATION

Process Overview.

The Veramed reporting process follows a model of:

- Titles and footnotes stored centrally for all programs
- SAS language programs and macros to extract and process titles for each individual output
- PROC TEMPLATE to manage the style of our outputs in RTF
- Output specific SAS language programs utilising the above titles and user written code to generate RTF outputs and datasets for QC purposes.

Generating outputs has been the area where the most time was spent, with the majority in relation to managing styles and templates. On the whole the issues identified can be considered as "global" issues that could be solved by adjusting selected central macros and programs.

Titles / Footnotes.

We found an unexpected result when managing titles and footnotes with our standard syntax. With the following SAS language code we were expecting the following output for RTF.

Syntax:

```
title1 "Veramed on SLC";
title3 "Protocol C999-123";
```

Expectation:

Veramed on SLC

Protocol C999-123

Actual result (no blank):

Veramed on SLC
Protocol C999-123

Simple Fix:

```
title1 "Veramed on SLC";
title2 " ";
title3 "Protocol C999-123";
```

RTF rendition.

Some items written out to RTF needed adjustment to give the desired outcome when running default programs and macros. One specific example of this was the page counting. Our process utilised the ODS escape character and RTF tags.

Syntax:

```
footnote1 j=r "Page |{thispage} of |{lastpage}";
```

Expectation:

Page 5 of 9

Actual results (space missing before "of"):

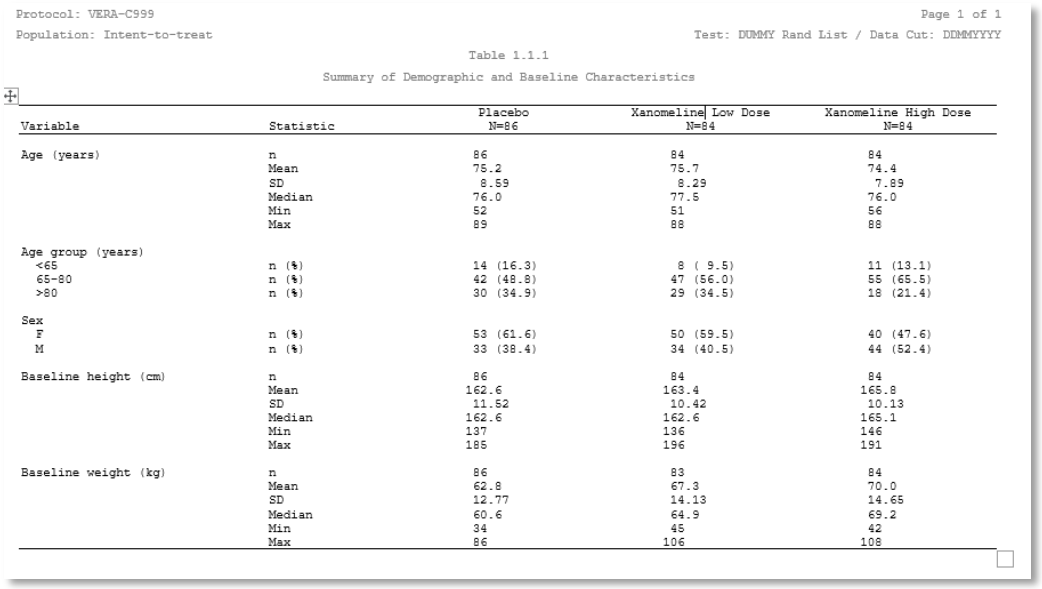
Page 5of 9

Simple Fix:

```
footnote1 j=r "Page |{thispage}  of |{lastpage}";
```

Style sheets (for RTF):

We needed to build a complete, all-encompassing style sheet for RTF destinations, as opposed to inheriting attributes and elements for other default styles and adjusting accordingly. Following this exercise, we were able to create output that we consider as perfectly suitable for the delivery of results for clinical trials. This same style sheet also gave us a suitable option for PDF destination.



Protocol: VERA-C999 Page 1 of 1

Population: Intent-to-treat Test: DUMMY Rand List / Data Cut: DDMMYYYY

Table 1.1.1

Summary of Demographic and Baseline Characteristics

Variable	Statistic	Placebo N=86	Xanomeline Low Dose N=84	Xanomeline High Dose N=84
Age (years)	n	86	84	84
	Mean	75.2	75.7	74.4
	SD	8.59	8.29	7.89
	Median	76.0	77.5	76.0
	Min	52	51	56
	Max	89	88	88
Age group (years)				
<65	n (%)	14 (16.3)	8 (9.5)	11 (13.1)
65-80	n (%)	42 (48.8)	47 (56.0)	55 (65.5)
>80	n (%)	30 (34.9)	29 (34.5)	18 (21.4)
Sex				
F	n (%)	53 (61.6)	50 (59.5)	40 (47.6)
M	n (%)	33 (38.4)	34 (40.5)	44 (52.4)
Baseline height (cm)	n	86	84	84
	Mean	162.6	163.4	165.8
	SD	11.52	10.42	10.13
	Median	162.6	162.6	165.1
	Min	137	136	146
	Max	185	196	191
Baseline weight (kg)	n	86	83	84
	Mean	62.8	67.3	70.0
	SD	12.77	14.13	14.65
	Median	60.6	64.9	69.2
	Min	34	45	42
	Max	86	106	108

Column headings in simple listings.

Whilst the PROC TEMPLATE enables output to be styled correctly, it was found that with simple listings, the column headings are only present on the first, as opposed to being repeated across every page.

The simple fix was to utilise our internal paging macros to create the necessary ID variables to control the paging, in which case the column headers were repeated as per expectation.

SUMMARY

Although it did take some effort to get the layout and presentation style correct, the numbers / results that were generated as part of this process match with those we had on file, and as such are confident in the results that SLC creates are accurate.

STATISTICAL ANALYSIS

The ability to run complex statistical analysis is a must have for SLC. Whilst unable to test every combination and permutation of every SAS Language statistical procedure across all study designs, working with experienced statisticians we built a comprehensive set of SAS language programs, utilising our CDISC ADaM datasets to give us evidence of how well SLC can handle statistical analysis. These programs were written without any steer to our Biostatisticians on what may, or may not work so we had a true and unbiases set of SAS language programs to test out.

The specific procedures we executed were:

- PROC MIXED
- PROC LOGISTIC
- PROC PHREG
- PROC LIFETEST
- PROC MI
- PROC GENMOD
- PROC TTEST
- PROC GLIMMIX
- PROC QUANTREG
- PROC MIANALYZE

From the Program analysis noted above, it was identified the following procedures are not currently supported

- PROC BGLIMM

To assess the results of the available procedures we focused on two specific areas.

1. Verbose listing results from procedures
   On review of the SLC listings we were able to gain the necessary confidence that the results created by SLC had parity with those previously generated. Whilst in some cases the number of significant digits used to present the result differs slightly, this was not at a level that gave us any concern on the accuracy of the results.

2. Output datasets from procedures
   Selected datasets were generated for each procedure focusing on those most readily used in clinical reporting to ensure these met our expectations, in accuracy of results and in structure and naming conventions. Other than some selected attribute differences in datasets from some procedures (see below), the results were as we were expecting / hoping for.

In the vast majority of cases, we were able to create the expected STAT procedure results from SLC that would be suitable for use in the reporting process. There are however a few specific items to note as part of this analysis

Odds Ratio's

Currently unsupported by SLC. Altair are looking at options to make this available within SLC

PROC TTEST

In a non-inferiority test, the delivered result was "1- expected answer" At time of writing the background of this result is in discussion with the Altair development team.

Output Datasets

We have identified some subtle differences to expectation in datasets output by STAT procedures. These are mostly labels, which we feel are of low criticality as there is no effect on the result itself. However there are cases where expected variable are not present either (e.g. PROC GENMOD, output dataset for "DIFFS" does not have expected variable STMTNO

SUMMARY

The output from these STAT procedures was reviewed against previously obtained results to ensure confidence in the numbers SLC is generating. Although in some cases different number of significant figures is being used which gave a technical difference, this is usually at the 6th or 7th decimal, and so we can take these results as matching.

CONCLUSION.

As an overall conclusion to our hypothesis of whether SLC is a viable solution for clinical study analytics, the overall response would be “Yes”. This does however come with select considerations / recommendations.

a. In what use case would SLC provide a viable option for successful deployment?
   As is discussed above, there are situations where SAS Language programs do need to be adapted to generate the expected results in SLC. This would mean that using SLC as a platform for running legacy SAS Language programs could well come with an overhead. It is expected that code would currently need updating to ensure clean execution, and as such programs would evolve away from those used for an actual delivery.
   However, if looking to use SLC in “new” studies only, following the right level of code review / execution analysis before deployment (like one would with any software / process deployment) we found we were able to readily execute a full suite of programs and generate the results we required, thus feel SLC is a viable option in this instance.

b. Which arm of the industry (Sponsor/Pharma or CRO) would gain most initial benefit from SLC?
   At this time we would suggest that Sponsor / Pharma companies would likely gain the most benefit from SLC within their own in house reporting environments, This enables them to be in control of the code used in their studies, and manage this across any interactions with a CRO.
   Within a CRO environment one of the challenging situations may be when working directly with Sponsors, and executing the sponsor SAS language programs in an independent platform. At this time there would need to be additional work to ensure the SAS language programs from the SAS application would execute as expected with SLC. The layout / presentation of the output will likely be the key aspect to consider here.

c. If considering SLC, then as well as being clear on what it can provide, it is important to fully understand what SLC is currently unable to support, and any adjustments that may need to be made to current processes and tools.
   Migration planning to assess adaptations needed to current SAS Language programs and macros will be required, and from our experience a high proportion of these can be achieved by altering current methodology to provide the same output. Time will be needed in fully defining your own output styles to meet your / your client requirements for tables / figures and listings, and there will be other factors in report delivery that will need to be understood.
   Similarly there are still some gaps in the statistical procedures that SLC can currently support.
   Finally Altair have been very open to suggestions / requests, and many of the items we have flagged during the project are already on their development roadmap, to fill gaps and further evolve SLC forward.

As already discussed, SLC certainly has a place within the industry, and as the software evolves and grows, gaps in procedures will be filled and required functionality added. SLC can provide a valid alternative programming platform to be incorporated in the clinical development process. More choice, and awareness of choice can only be a good thing.

CONTACT INFORMATION

(In case a reader wants to get in touch with you, please put your contact information at the end of the paper.)
Your comments and questions are valued and encouraged. Contact the author at:

Author Name:
Company:
Address:
Work Phone:
Email:
Website: