Paper TS08

The Automated Metadata-driven Table Generation Process (TFLGen) at Amgen Inc.

Neil Lyon, Amgen Inc., Uxbridge, UK

ABSTRACT

Every day near-identical programs are written by countless programmers to create similar tables. They have been created in this manner for the last 20+ years, sometimes with macros, sometimes re-using code, but always by manually editing SAS® programs. Amgen recently developed a suite of single purpose modular SAS macros and wondered if there was a better way to use them rather than having users learn what they do and then manually inserting them into code.

The solution is the metadata-driven automated table generation process (TFLGen). By breaking tables down into their components, areas of repetition were identified and metadata then used to describe each component in a single place, simplifying the process and optimizing re-use. Using familiar, intuitive user interfaces to supply this metadata, the time to create and update table producing SAS programs was reduced.

INTRODUCTION

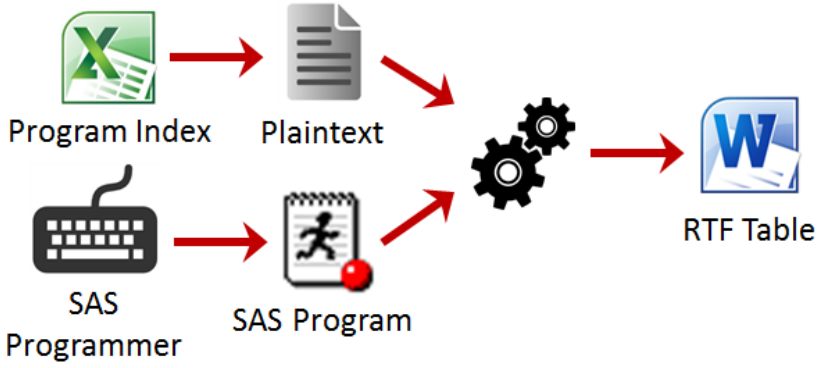
Amgen has a suite of standard end-to-end macros to produce our most common outputs. They were written over a decade ago and have been modified in the intervening years to add more functionality. These macros have grown rather large and have become hard to maintain. The macro that produces tiered summary tables has 9554 lines of code, a 60 page user manual, requires a user to specify up to 76 input parameters and produces a log that can grow to over 100,000 lines!

In response to this problem, all processes required to produce a table were identified and refactored until the lowest common actions were found. SAS macros were written to perform these actions. The macros are then combined and built upon to form a suite of modular macros that can be used to build any table. This generated many macros that would require users to understand and use correctly; not the easiest of tasks. To simplify the use of these macros a WYSIWYG interface was conceived that would allow users to describe the statistics they required using metadata and SAS terminology.

This paper will describe how Amgen approached the problem and how the solution was designed and implemented, together with examples of some commonly produced tables.

EXISTING TABLE GENERATION PROCESS

Every analysis at Amgen has a master list of all TFLs that are required, called the Program Index. The Program Index is an Excel spreadsheet that has one row for every output, storing information such as the output number, titles, footnotes, output and program names alongside details of the programmer, tester and the testing status. When saved the Program Index makes a plaintext version of itself.



The SAS programmer writes a SAS program to generate, format and report the necessary statistics (utilizing macros where applicable) to generate the output. When the SAS program is run it reads in the plaintext version of the Program Index and extracts the information from the row that has been identified within the program. This information is added prior to the reporting stage.

The screenshot below shows the typical output specific information that is currently stored within the Program Index.

Output Number	Titles	Footnotes	Output Name	Program Name
14-2.1.1	Baseline Demographics{\line}(Primary Analysis Set)	N = number of subjects randomized and dosed in the safety analysis set.{\line}[1] Percentages based on number of subjects of each sex.	t14-02-001-001-dm-sum-pas.rtf	t-dm-sum-pas.sas
14-6.2.1	Treatment Emergent Adverse Events by System Organ Class and Preferred Term{\line}(Safety Analysis Set)	N = number of subjects randomized and dosed in the safety analysis set.{\line}Coded using MedDRa version &meddrav..	t14-06-002-001-ae-soc-pref-saf.rtf	t-ae-soc-pref-saf.sas

REQUIREMENTS

Prior to starting work on a solution the high level requirements that would drive the design of the solution were identified:

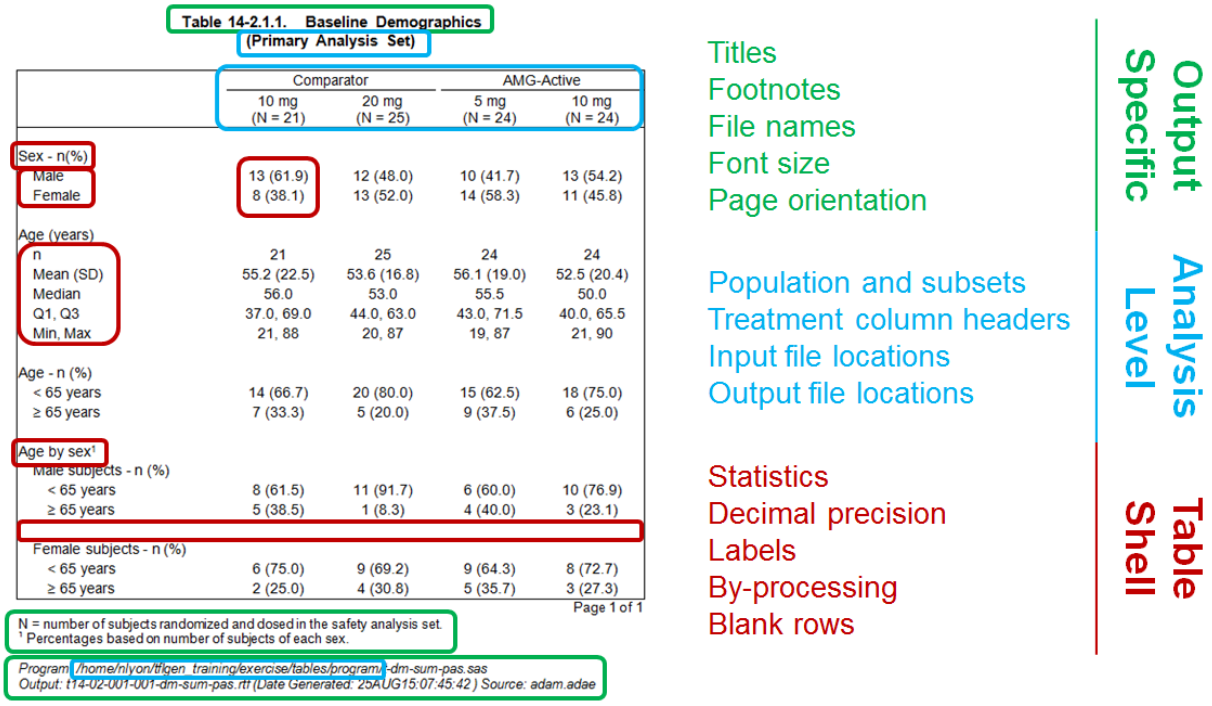
1) Leverage the new modular macro library that was created to replace the ageing standard macros.
2) Maximize the re-use of information, reducing duplication of effort.
3) Minimize the information that is supplied.
4) Create a tool that is flexible enough to produce as many commonly produced tables as possible.
5) Have a simple to use, intuitive interface.

DESIGN

The end product of TFLGen is a tabular RTF report. Before developing a product that creates these reports, we required a detailed understanding of the structure of various tables. The first step was to break down a selection of tables into their components.

THE ANATOMY OF A TABLE

Each table component was analyzed and categorized into distinct types that could be described using metadata. The figure below highlights some commonly occurring components and the categories they were assigned to.



Three distinct component types and their metadata were identified:

- **Output specific** components, many of which are already stored within the Program Index.
- **Analysis level** components that can be repeatedly used by many tables within an analysis.
- **Table shell** components that can be used by several outputs in an analysis (or even across analyses), but are combined with different analysis level and output specific metadata (e.g. a demography table repeated for multiple populations). Two distinct table types were identified, tiered tables such as adverse events and concomitant medications and what Amgen call base tables containing non-tiered categorical and univariate type descriptive statistics, such as baseline demography and safety summaries by visit. These table types will be referred to as tiered and base tables respectively.

METADATA STORAGE SOLUTIONS

Once the types of components were identified, the metadata that would describe each component was designed and the optimal location for storing each metadata item decided.

Output specific

It was decided to leverage the output specific information that already exists in the Program Index. It is already used by all programmers at Amgen to keep the output specific information separate from the SAS program, so is a widely understood tool.

Analysis level

As each analysis already has its own Program Index it seemed logical to extend the workbook to include the analysis level information.

Table shell

The remaining table shell information comes from components that describe the statistics and their presentation. This information is supplied from the statistician to the programmer by a TFL shell Word document. The inset below shows a typical demography table shell, it uses a grid structure to hold all the information. To simplify the transcription of table shell information into metadata we wanted to mimic the shell structure in the metadata storage solution. The requirements for the storing and editing of the table shell information are:

- Grid structure
- Edit cells
- Move, copy, delete, insert rows
- Provide pre-filled drop-down lists
- Apply conditional statements to cells
- Export the contents to XML

Baseline Demographics

	[Headers] (N = xx)
Sex - n (%)	
Male	x (x.x)
Female	x (x.x)
Age (years)	
n	x
Mean (SD)	x.x (x.x)
Median	x.x
Q1, Q3	x.x, x.x
Min, Max	x, x
Age group - n (%)	
< 65 years	x (x.x)
≥ 65 years	x (x.x)
≥ 75 years	x (x.x)

Excel provides the vast majority of these requirements, so it was used to develop the working prototype. Paired with Visual Basic, it has the capability to cover all the requirements.

It was decided to use XML as the transport format for the metadata. It is a much more flexible format than the plaintext that was originally used by the Program Index. By decoupling the user interface from the underlying TFLGen application changes can be made to either part much more easily.

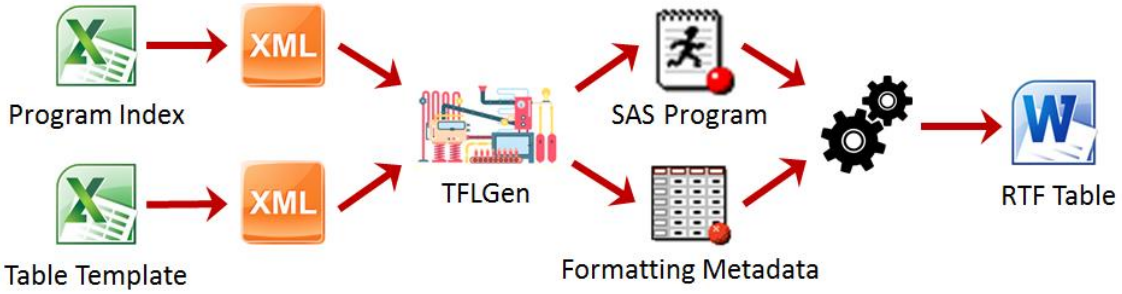
USING THE METADATA

All the metadata are compiled from each source into a single SAS dataset that reflects the structure and content of the table being produced and describes how the output will be displayed. The metadata dataset is then passed through a series of macros that read the contents and write SAS code to generate the statistics, apply the formatting and then produce the report.

This metadata dataset has several advantages:

1) The metadata dataset can be reviewed to validate the table.
2) The metadata dataset can be merged onto the statistics that get derived in the generated program to access the formatting information. This saves hardcoding the information into the generated SAS program.
3) By separating the formatting and labelling elements from the SAS program, changes can be made just to the SAS dataset that will not require a change to the SAS program. However, fundamental changes to the output such as new statistics will require both the SAS dataset and SAS program to be updated.
4) The metadata dataset can be fed through a macro that will validate the contents for appropriateness (e.g. a mean can only be calculated on a numeric variable).

The process flow now looks like that in the diagram below.



IMPLEMENTATION

EXTENDING THE PROGRAM INDEX

To store the analysis level metadata, the following new worksheets were added alongside the main worksheet that lists all outputs. These new worksheets are described below.

Populations

Contains metadata that describes all populations used in the analysis.

Population name.

Where Clause describing the population with a SAS expression.

Population	Where Clause
(Safety Analysis Set)	saffl = 'Y'
(Primary Analysis Set)	pasfl = 'Y'
(All Randomized Subjects)	n(randdt)

Subsets

Contains metadata that describes all subsets of populations used in the analysis.

Subset name.

Where Clause describing the subset with a SAS expression.

Subset	Where Clause
Subjects with Anemia at Baseline	.< blhgb < 10
Male subjects Only	sexn = 1
Female Subjects Only	sexn = 2

Treatment Column Headers

Contains metadata that describes the different sets of treatment colunms used within the tables in an analysis.

Header name attributed to a row to describe which treatment header set it belongs to.

Span Labels are labels that span multiple treatment columns.

Treatment order that columns are to appear in.

Column Label to apply above the statistics in the treatment columns.

Data Set that can be used to describe the treatment columns.

Variable present on the dataset whose values are used to describe the treatment columns.

Quantifying value of the variable that describes the treatment columns.

Where Clause to describe treatment groups using more complex SAS expressions.

Column Width in inches.

Column Alignment is left, center or right.

Header	Span Labels	Treatment	Column Label	Data Set	Variable	Quant	Where Clause	Column Width	Column Alignment
safety		1	Comparator	adam.adsl			trt01an in(1,2)	1	C
safety		2	AMG-Active	adam.adsl			trt01an in(3,4)	1	C
safety		3	Total	adam.adsl			trt01an in(1,2,3,4)	1	C
primary	Comparator	1	10 mg	adam.adsl	trt01an	1		1	C
primary	Comparator	2	20 mg	adam.adsl	trt01an	2		1	C
primary	AMG-Active	3	5 mg	adam.adsl	trt01an	3		1	C
primary	AMG-Active	4	10 mg	adam.adsl	trt01an	4		1	C

Template Locations

Contains metadata that describes where the sources of the table shell specific information (table templates) are located.

Template Location name.

UNIX Directory for each of template location.

Template Location	UNIX Directory
therapeutic_area	/home/therapeutic_area/docs/templates
product	/home/therapeutic_area/product/docs/templates
protocol	/home/therapeutic_area/product/protocol/docs/templates

General Information

Contains the remaining metadata that has not been captured in any other location. It describes the file locations needed by the generated SAS program, the subject variable used in the analysis, locations for any macros that may be requried by the programmer and file locations for any dataset libraries that are needed.

Item is a keyword that describes the metadata stored in the row.

Description of the metadata item.

Value to be assigned to the metadata item.

Item	Description	Value
table_program_dir	File location for table programs	/home/therapeutic_area/product/protocol/analysis/tables/program
table_metadata_dir	File location for table metadata sets	/home/therapeutic_area/product/protocol/analysis/tables/metadata
table_bpdf_dir	File location for table BPDFs	/home/therapeutic_area/product/protocol/analysis/tables/validation
table_output_dir	File location for table outputs	/home/therapeutic_area/product/protocol/analysis/tables/output
subject_var	Name of the subject/patient variable	usubjid
custom_init_macro_dir1	First location searched for custom macros	/home/therapeutic_area/product/protocol/analysis/utilities
custom_init_macro_dir2	Second location searched for custom macros	/home/therapeutic_area/product/utilities
adam_dir	Fiel location for ADaM data sets	/home/therapeutic_area/product/protocol/analysis/statdata/adam

Linking the Metadata to an Output

The metadata described in the extended Program Index needs to be linked to each output. The primary page that lists all outputs and contains the titles, footnotes, etc. was extended so the new metadata can be selected. Where possible the fields use drop-down lists that are dynamically generated from metadata names in the new worksheets.

Population has a drop-down list of all populations described in the Populations worksheet.

Subset has a drop-down list of all subsets described in the Subsets worksheet.

Numerator Where SAS expressions limit all data reported in the table, does not affect big N.

Header has a drop-down list of all headers described in the Headers worksheet.

Template Location has a drop-down list of all table template locations in the Template Locations worksheet.

Template names the table template providing the statistics metadata.

Sort Section lists the sections within the table template that require sorting based on counts.

Sort Source Treatment lists the treatment columns to supply counts to sort sections within the table template.

Body Font Height specifies the font size to use.

Orientation of the page for the output.

Allow Code Generation prevents SAS program being inadvertently overwritten.

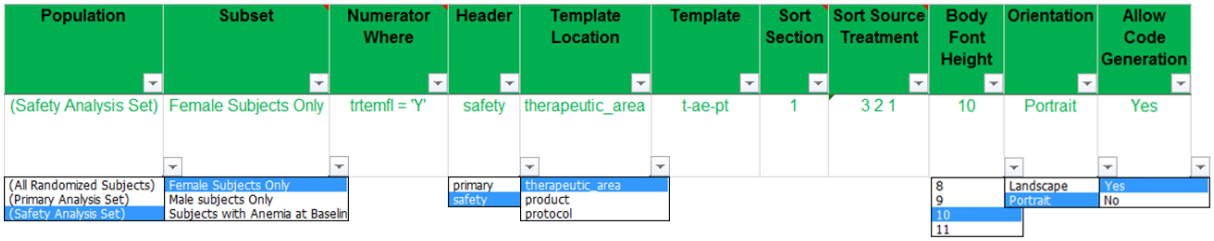


TABLE TEMPLATE

The table template is the name given to the new Excel workbook containing the metadata that describes the statistics in the main body of the table. The metadata is split into 3 sections:

1) General metadata that is applicable to the whole table described by the table template.
2) Column label metadata that describes what information will be displayed in leftmost and treatment columns.
3) Main body metadata that describes the statistics and their formatting.

As mentioned earlier there are 2 types of table: tiered and base. Below are examples of both, they have the same:

General metadata items:

sps_generate_module contains the keyword for this type of table template.

section_wrap_flag describes whether sections will be allowed to wrap over multiple pages.

Column label metadata items:

col_width specifies the width in inches of the leftmost column in the table.

col_align specifies the alignment (left/center/right) of the leftmost column in the table.

col_big_n_label specifies whether or not the big N are to be displayed in the treatment columns.

col_n_pct_label specifies the label underneath the big N that shows what statistics are being presented.

Tiered Table Template

METADATA	T1	TREATMENTS	STATISTIC	INDENT	DATA_SET	VARIABLE	SORT_SECTION	SORT_DIRECTION
sps_generate_module	table_tiered_001							
section_wrap_flag	Y							
col_width	AUTO							
col_align	L							
col_label	System Organ Class{\line} Preferred Term	[Headers]						
col_big_n_label		(N = xx)						
col_n_pct_label		n (%)						
blank_row								
level_0	Subjects experiencing &level_0_text	x (x.x)	n percent	0				
blank_row								
level_1	System Organ Class	x (x.x)	n percent	0	adam.adae	aebodsys	1	descending
level_2	Preferred Term	x (x.x)	n percent	1	adam.adae	aedecod	2	descending

Main body metadata:

METADATA uses keywords from a drop-down menu to describe the metadata present within the row. In this example the *level_n* specifies the tier level for the hierarchical statistics; *blank_row* shows where a blank row will be inserted in the table.

T1 is the text that is to be displayed in the leftmost column in the table. In tiered tables the values for rows where METADATA is *level_1* or greater will be replaced with the actual values from the variables specified in VARIABLE.

TREATMENTS describes how statistics are displayed, the trailing "x" after a "." showing the decimal precision.
STATISTIC uses keywords from a drop-down list to specify the statistics that are to be displayed.
INDENT shows how many times the T1 field is to be indented.
DATA_SET specifies the dataset that will provide the variable from which the statistics will be derived.
VARIABLE specifies the variable that statistics will be derived from.
SORT_SECTION identifies parts of the table that can be sorted based on counts in the treatment columns.
SORT_DIRECTION specifies the direction the sort should use.

Base Table Template

METADATA	T1	TREATMENTS	STATISTIC	INDENT	DATA_SET	VARIABLE	QUANT	WHERE_CLAUSE	DENOMINATOR	BY_DISPLAY_VARIABLE
sps_generate_module	table_base_001									
section_wrap_flag	N									
col_width	AUTO									
col_align	L									
col_label		[Headers]								
col_big_n_label		(N = xx)								
col_n_pct_label										
blank_row										
label	Sex - n(%)			0						
statistic	Male	x (x.x)	n percent	1	adam.adsl	sexn	1			
statistic	Female	x (x.x)	n percent	1	adam.adsl	sexn	2			
blank_row										
label	Age (years)			0						
statistic	n	x	n	1	adam.adsl	age				
statistic	Mean (SD)	x.x (x.x)	mean stdev	1	adam.adsl	age				
statistic	Median	x.x	median	1	adam.adsl	age				
statistic	Q1, Q3	x.x, x.x	q1 q3	1	adam.adsl	age				
statistic	Min, Max	x, x	min max	1	adam.adsl	age				
blank_row										
label	Age - n (%)			0						
statistic	< 65 years	x (x.x)	n percent	1	adam.adsl			.<age<65		
statistic	≥ 65 years	x (x.x)	n percent	1	adam.adsl			age>=65		
blank_row										
by_start	Age by sex[1]			0	adam.adsl	sexn				sex
label	!BY_DISPLAY_VARIABLE! subjects - n (%)			1						
statistic	< 65 years	x (x.x)	n percent	2	adam.adsl			.<age<65	sexn>.	
statistic	≥ 65 years	x (x.x)	n percent	2	adam.adsl			age>=65	sexn>.	
by_end										

Main body metadata:

METADATA uses keywords from a drop-down menu to describe the metadata present within the row. In this example the *label* specifies the row will have a text label, but no statistics; *blank_row* shows where a blank row will be inserted in the table; *statistic* shows that the row will display statistics; *by_start* and *by_end* identify the start and end point for a section that contains by-processing.

T1 is the text that is to be displayed in the leftmost column in the table. The keyword *!BY_DISPLAY_VARIABLE!* will be replaced at run-time by the values present in the variable specified in the BY_DISPLAY_VARIABLE, allowing dynamic text generation.

TREATMENTS describes how statistics are displayed, the trailing "x" after a "." showing the decimal precision.

STATISTIC uses keywords from a drop-down list to specify the statistics that are to be displayed.

INDENT shows how many times the T1 field is to be indented.

DATA_SET specifies the dataset that will provide the variable from which the statistics will be derived.

VARIABLE specifies the variable from which statistics will be derived or if SPS_META = *by_start* then it provides the variable that the by-processing will use for sorting.

QUANT specifies the particular value of VARIABLE for which to calculate the categorical statistics.

WHERE_CLAUSE takes a SAS expression that is used for complex descriptions of data to present categorical statistics for or to limit the data being reported in univariate type statistics.

DENOMINATOR takes a SAS expression describing alternative denominators to the big N for percentages.

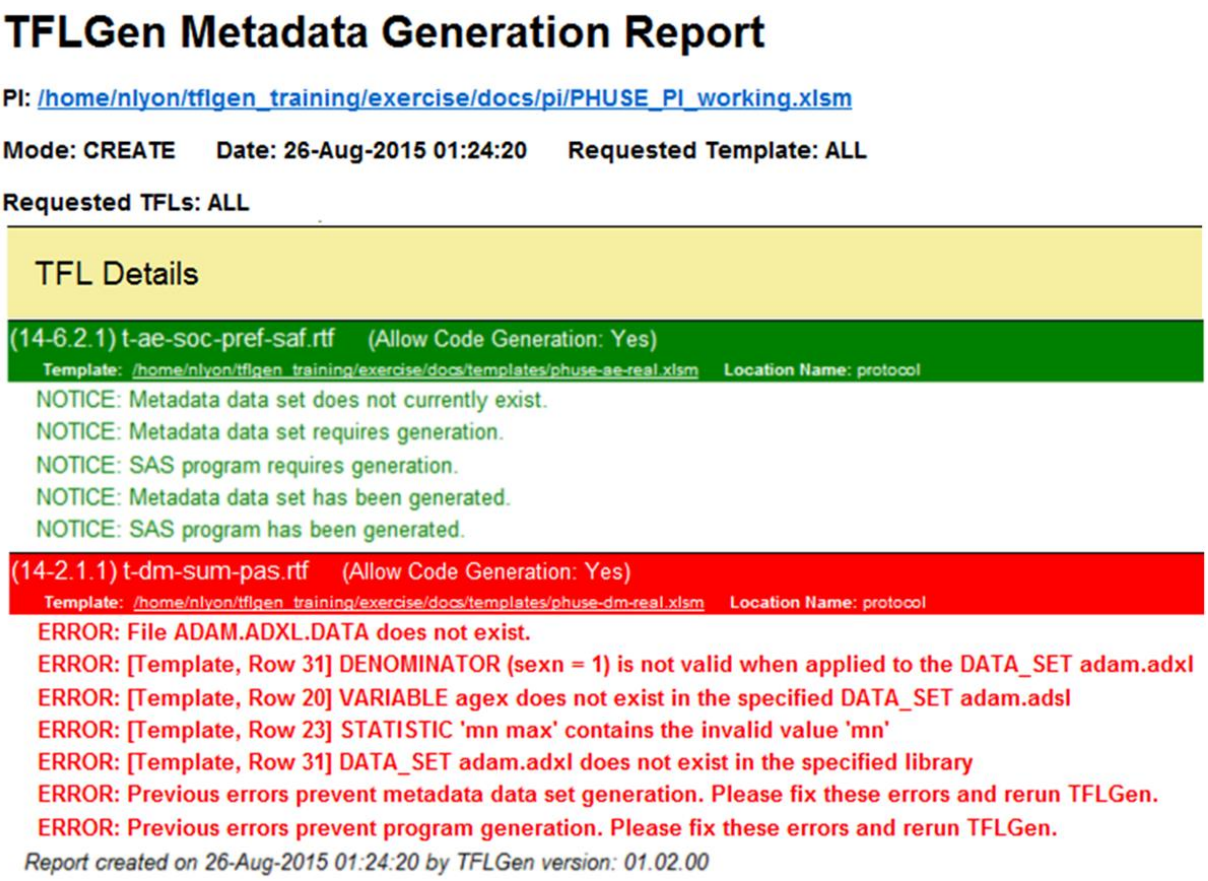
BY_DISPLAY_VARIABLE when SPS_META = *by_start* it provides a variable that the by-processing can use for displaying in the label on the leftmost column of the table.

RUNNING TFLGEN

Amgen uses SAS Grid installed on a UNIX server. To take advantage of its concise and robust XML processing capabilities, a Groovy® application was written that reads the Program Index metadata file, creating a short program for each output that TFLGen is required to run for. These programs run concurrently and for each output will generate a dataset of the compiled metadata. The metadata dataset is then validated for appropriateness and if passed is used to generate the SAS program that, once run, will produce the required output.

The application has several options so that users can target specific outputs, all outputs created using a single table template or all outputs within the Program Index. When TFLGen has finished running it provides the user with a message in the UNIX console that informs the user whether it was successful or if there were errors. Error messages

are designed to pinpoint the exact location of any problems helping the user to correct any mistakes. The screenshot below shows a typical report with a successful generation followed and another that has issues with some metadata.



TFLGen Metadata Generation Report

PI: /home/nlyon/tflgen_training/exercise/docs/pi/PHUSE_PI_working.xlsm

Mode: CREATE **Date: 26-Aug-2015 01:24:20** **Requested Template: ALL**

Requested TFLs: ALL

TFL Details

(14-6.2.1) t-ae-soc-pref-saf.rtf (Allow Code Generation: Yes)

Template: /home/nlyon/tflgen_training/exercise/docs/templates/phuse-ae-real.xlsm **Location Name:** protocol

NOTICE: Metadata data set does not currently exist.
NOTICE: Metadata data set requires generation.
NOTICE: SAS program requires generation.
NOTICE: Metadata data set has been generated.
NOTICE: SAS program has been generated.

(14-2.1.1) t-dm-sum-pas.rtf (Allow Code Generation: Yes)

Template: /home/nlyon/tflgen_training/exercise/docs/templates/phuse-dm-real.xlsm **Location Name:** protocol

ERROR: File ADAM.ADXL.DATA does not exist.
ERROR: [Template, Row 31] DENOMINATOR (sexn = 1) is not valid when applied to the DATA_SET adam.adxl
ERROR: [Template, Row 20] VARIABLE agex does not exist in the specified DATA_SET adam.adsl
ERROR: [Template, Row 23] STATISTIC 'mn max' contains the invalid value 'mn'
ERROR: [Template, Row 31] DATA_SET adam.adxl does not exist in the specified library
ERROR: Previous errors prevent metadata data set generation. Please fix these errors and rerun TFLGen.
ERROR: Previous errors prevent program generation. Please fix these errors and rerun TFLGen.

Report created on 26-Aug-2015 01:24:20 by TFLGen version: 01.02.00

DESIGN PROBLEMS AND SOLUTIONS

TOO MANY NEAR-IDENTICAL TABLE TEMPLATES BEING GENERATED

It was found that many table templates were very similar to each other, only differing by a couple of metadata items. In a normal SAS program this would warrant the development of a macro to produce all similar tables with macro parameters allowing changeable values.

In order to reduce the number of table templates being developed the idea of changeable parameters was extended to the table templates. If an item could be replaced across versions of a table then the changeable item would be replaced with a keyword prefixed with an ampersand (&), similar to SAS macro variables. A new worksheet was added to the Program Index that would allow the values to be specified. When TFLGen runs it replaces all values that it finds in the metadata dataset that match the parameters specified in the Program Index.

Base Output Name on main Program Index worksheet that changeable parameter is for.
Parameter name specified in the table template (e.g. &level_0_text).
Value to replace the parameter with.
Description of the parameter's purpose.

Base Output Name	Parameter	Value	Description
t-ae-soc-pref-fas.rtf	level_0_text	an adverse event	Text for top row of AE table
t-ae-soc-pref-ser-saf.rtf	level_0_text	a serious adverse event	Text for top row of AE table
t-ae-soc-pref-disc-saf.rtf	level_0_text	an adverse event leading to discontinuation	Text for top row of AE table

USERS LIMITED TO USING DATASETS DIRECTLY FROM A LIBRARY

TFLGen was originally conceived to use ADaM datasets that are normally considered as being one procedure away from the necessary statistics (i.e. no processing or manipulation of data prior to deriving statistics). In reality, several ADaM datasets fall short of this goal and require processing prior to use or the creation of data-driven macro variables.

To solve this problem another worksheet was added to the Program Index that would allow user-specified code to be run at the start of the generated SAS program. This allows the creation of work datasets that would have the data needed for reporting.

Base Output Name on main Program Index worksheet that custom code is to be used with (allows wildcards).
Custom Init Code to Include at Start of Program is field where SAS code can be entered (which can include calls to external user-developed macros).

Base Output Name	Custom Code to Include at Start of Program

t-ae*	*--- Get MedDRA version number ---;
	data _null_;
	set adam.adae(keep = aemeddra obs = 1);
	call symputx('meddrav',aemeddra,'G');
	run;

HOW TO QC PROGRAMS GENERATED BY TFLGEN

TFLGen is a fully validated software suite whose results can be trusted to reliably generate the numbers as per the metadata compiled in the metadata dataset. Initially we allowed programmers to review the metadata dataset and the RTF output. This was a cumbersome task as the SAS datasets are not always the easiest files to review.

To simplify this process, TFLGen now creates a pseudo-table in the same folder that contains an output's metadata data set. Below is a screenshot of a typical file. TFLGen adds the red text that shows what metadata is used and the blue highlighting that shows what metadata has changed since the metadata dataset was last created.

Table 14-2.1.1. Baseline Demographics
(Primary Analysis Set)
Population:pasfl = 'Y'

	Comparator		AMG-Active	
	10 mg adam.adsl. (trt01an=1) (N = xx)	20 mg adam.adsl. (trt01an=2) (N = xx)	5 mg adam.adsl. (trt01an=3) (N = xx)	10 mg adam.adsl. (trt01an=4) (N = xx)
drop_zero_row_flag:N **subject_var:**usubjid				
Sex - n(%)				
Male adam.adsl.(sexn=1)	x (x.x) n percent	x (x.x)	x (x.x)	x (x.x)
Female adam.adsl.(sexn=2)	x (x.x) n percent	x (x.x)	x (x.x)	x (x.x)
Age (years)				
n adam.adsl.age	x n	x	x	x
Mean (SD) adam.adsl.age	x.x (x.x) mean stdev	x.x (x.x)	x.x (x.x)	x.x (x.x)
Median adam.adsl.age	x.x median	x.x	x.x	x.x
Q1, Q3 adam.adsl.age	x.x, x.x q1 q3	x.x, x.x	x.x, x.x	x.x, x.x
Min, Max adam.adsl.age	x, x min max	x, x	x, x	x, x
Age - n (%)				
< 65 years adam.adsl.(.<age<65)	x (x.x) n percent	x (x.x)	x (x.x)	x (x.x)
≥ 65 years adam.adsl.(age>=65)	x (x.x) n percent	x (x.x)	x (x.x)	x (x.x)
Age by sex[1]				
!adam.adsl.sexn! subjects - n (%)				
< 65 years adam.adsl.(.<age<65)	x (x.x) n percent(sexn>.)	x (x.x)	x (x.x)	x (x.x)
≥ 65 years adam.adsl.(age>=65)	x (x.x) n percent(sexn>.)	x (x.x)	x (x.x)	x (x.x)

Page 1 of 1

N = number of subjects randomized and dosed in the safety analysis set.
[1] Percentages based on number of subjects of each sex.

Program: ./t-dm-sum-pas.sas
Output: t_dm_sum_pas.rtf (Date Generated: 25AUG15:07:44:30) Source: adam.adae

DEVELOPING MANY INTERCONNECTED MACROS

Due to the interconnectedness of the underlying macros used by TFLGen it was hard for multiple developers to work on modules and make decisions as they could easily have ramifications for parts that other developers were working on. To get a handle on all the moving pieces, one programmer oversaw the primary development, putting initial versions of all the macros in place with all basic functionality present. The connections where data is passed between macros and the XML files that come from the Program Index and the table template were identified and areas of commonality were determined. To avoid tramping macro parameters throughout the application to where they are utilized, data sets were used to pass information across macros.

The different areas were distributed amongst the team to finish:

- Changes to the Program Index and the new table template
- Groovy application that sets up the area, performs preliminary validation of the Program Index, runs concurrent SAS jobs and generates report of successes and error messages
- Metadata dataset generation
- Metadata dataset validation
- SAS program generation
    - Data preparation
    - Statistic derivation
    - Applying formatting metadata to derived statistics
    - Reporting

Close contact was required amongst all developers to ensure that any changes that could affect another area were disclosed in a timely and efficient manner.

TESTING TFGLEN FOR ROBUSTNESS, RELIABILITY AND VALIDITY

When creating an application such as TFLGen it is essential that it runs without problems and generates the expected numbers every time. To ensure that the software suite performs as expected every module undergoes an extensive unit test to ensure that its behavior follows the individual requirements and returns the expected results. Once the whole software package was complete, integration testing was performed to prove that the individual elements worked as a whole. This involved creating Program Index and table templates that would test all the functions that are available within TFLGen. The results were dual-programmed for validity.

Prior to full production roll-out a limited production roll-out tested TFLGen with a few handpicked studies to iron out any issues. The outputs were dual-programmed to further prove that the generated numbers were correct.

TFLGEN COVERAGE

Currently TFLGen is expected to produce at least 65% of the tables in an analysis.

TFLGen Can Produce	TFLGen Cannot Produce

• Demography
• Patient Disposition
• Analysis Set Inclusion and Exclusion
• Baseline Characteristics
• Medical History
• Protocol Deviations
• Safety Summaries by Visit (Labs/Vitals/ECG)
• Adverse Events (n %)
• Concomitant Medications
• ...most tables with n, % or univariate statistics
	• Complex statistics
	  ○ Analysis of variance/covariance
	  ○ Regression
	  ○ Binomial statistics
	  ○ Kaplan-Meier
	• Treatment differences
	• Rates
	  ○ Exposure adjusted event rates[1]
	  ○ Exposure adjusted subject rates[1]

[1] The next release of TFLGen will include these additional analysis types.

TFLGen is not expected to be able to produce all tables. SAS is a highly flexible tool and as soon as any simplification of its use is attempted this flexibility is naturally reduced. If TFLGen were designed to produce complex tables such as analysis of covariance, it would require a much more complex and crowded table template, which defeats the requirement of having a simple and intuitive interface. To keep the application easy to use and maintain, TFLGen has been deliberately limited to only attempt production of simpler descriptive statistics.

NEXT STEPS

TFLGen is not yet an optimized solution for Amgen. There are some outputs that cannot yet be created with the application, but should be possible with some minor changes. We are looking to increase coverage and simplify the process with new features within the Program Index and table templates.

- There are multiple copies of the Excel table template file across the company. We intend to simplify this by creating a single template editor that will import and export the XML version of the table template to and from itself. This will enable users to always use the latest version of table template without needing to upgrade the table template and will make rolling out of new table templates simpler and more controlled.
- The functions of the base and tiered templates can be combined into a single table template, enabling more complex tables to be created.
- Update table template to allow users to declare a variable that will contain the decimal precision such as can exist on the ADaM LB domain.
- Update table template to generate exposure adjusted event rates and exposure adjusted subject rates.
- Update table template to allow formats to be defined, removing the need to specify the values to be displayed in the output.

EXAMPLES

ADVERSE EVENTS BY SYSTEM ORGAN CLASS AND PREFERRED TERM

Analysis level metadata (header)

Header	Span Labels	Treatment	Column Label	Data Set	Variable	Quant	Where Clause	Column Width	Column Alignment
safety		1	Comparator	adam.adsl			trt01an in(1,2)	1	C
safety		2	AMG-Active	adam.adsl			trt01an in(3,4)	1	C
safety		3	Total	adam.adsl			trt01an in(1,2,3,4)	1	C

Table shell metadata

METADATA	T1	TREATMENTS	STATISTIC	INDENT	DATA_SET	VARIABLE	SORT_SECTION	SORT_DIRECTION
sps_generate_module	table_tiered_001							
section_wrap_flag	Y							
col_width	AUTO							
col_align	L							
col_label	System Organ Class{\line} Preferred Term	[Headers]						
col_big_n_label		(N = xx)						
col_n_pct_label		n (%)						
blank_row								
level_0	Subjects experiencing &level_0_text	x (x.x)	n percent	0				
blank_row								
level_1	System Organ Class	x (x.x)	n percent	0	adam.adae	aebodsys	1	descending
level_2	Preferred Term	x (x.x)	n percent	1	adam.adae	aedecod	2	descending

Table 14-6.2.1. Treatment Emergent Adverse Events by System Organ Class and Preferred Term (Safety Analysis Set)

System Organ Class Preferred Term	Comparator (N = 50) n (%)	AMG-Active (N = 49) n (%)	Total (N = 99) n (%)
Subjects experiencing an adverse event	41 (82.0)	43 (87.8)	84 (84.8)
Gastrointestinal disorders	17 (34.0)	19 (38.8)	36 (36.4)
Dyspepsia	4 (8.0)	3 (6.1)	7 (7.1)
Abdominal pain	0 (0.0)	5 (10.2)	5 (5.1)
Abdominal discomfort	4 (8.0)	1 (2.0)	5 (5.1)
Flatulence	1 (2.0)	3 (6.1)	4 (4.0)
Infections and infestations	19 (38.0)	14 (28.6)	33 (33.3)
Pharyngitis	3 (6.0)	0 (0.0)	3 (3.0)
Musculoskeletal and connective tissue disorders	15 (30.0)	15 (30.6)	30 (30.3)
Arthralgia	6 (12.0)	3 (6.1)	9 (9.1)
Myalgia	1 (2.0)	3 (6.1)	4 (4.0)
Muscle spasms	3 (6.0)	1 (2.0)	4 (4.0)
Rotator cuff syndrome	3 (6.0)	0 (0.0)	3 (3.0)
Cardiac disorders	16 (32.0)	13 (26.5)	29 (29.3)
Angina pectoris	5 (10.0)	3 (6.1)	8 (8.1)

Page 1 of 4

N = number of subjects randomized and dosed in the safety analysis set.
Coded using MedDRa version 16.1.

Program: /home/nlyon/tflgen_training/exercise/tables/program/t-ae-soc-pref-saf.sas
Output: t14-06-002-001-ae-soc-pref-saf.rtf (Date Generated: 25AUG15:08:55:24) Source: adam.adae

BASELINE DEMOGRAPHICS

Analysis level metadata (header)

Header	Span Labels	Treatment	Column Label	Data Set	Variable	Quant	Where Clause	Column Width	Column Alignment
primary	Comparator	1	10 mg	adam.adsl	trt01an	1		1	C
primary	Comparator	2	20 mg	adam.adsl	trt01an	2		1	C
primary	AMG-Active	3	5 mg	adam.adsl	trt01an	3		1	C
primary	AMG-Active	4	10 mg	adam.adsl	trt01an	4		1	C

Table shell metadata

METADATA	T1	TREATMENTS	STATISTIC	INDENT	DATA_SET	VARIABLE	QUANT	WHERE_CLAUSE	DENOMINATOR	BY_DISPLAY_VARIABLE
sps_generate_module	table_base_001									
section_wrap_flag	N									
col_width	AUTO									
col_align	L									
col_label		[Headers]								
col_big_n_label		(N = xx)								
col_n_pct_label										
blank_row										
label	Sex - n(%)			0						
statistic	Male	x (x.x)	n percent	1	adam.adsl	sexn	1			
statistic	Female	x (x.x)	n percent	1	adam.adsl	sexn	2			
blank_row										
label	Age (years)			0						
statistic	n	x	n	1	adam.adsl	age				
statistic	Mean (SD)	x.x (x.x)	mean stdev	1	adam.adsl	age				
statistic	Median	x.x	median	1	adam.adsl	age				
statistic	Q1, Q3	x.x, x.x	q1 q3	1	adam.adsl	age				
statistic	Min, Max	x, x	min max	1	adam.adsl	age				
blank_row										
label	Age - n (%)			0						
statistic	< 65 years	x (x.x)	n percent	1	adam.adsl			.<age<65		
statistic	≥ 65 years	x (x.x)	n percent	1	adam.adsl			age>=65		
blank_row										
by_start	Age by sex[1]			0	adam.adsl	sexn				sex
label	!BY_DISPLAY_VARIABLE! subjects - n (%)			1						
statistic	< 65 years	x (x.x)	n percent	2	adam.adsl			.<age<65	sexn>.	
statistic	≥ 65 years	x (x.x)	n percent	2	adam.adsl			age>=65	sexn>.	
by_end										

Table 14-2.1.1. Baseline Demographics
(Primary Analysis Set)

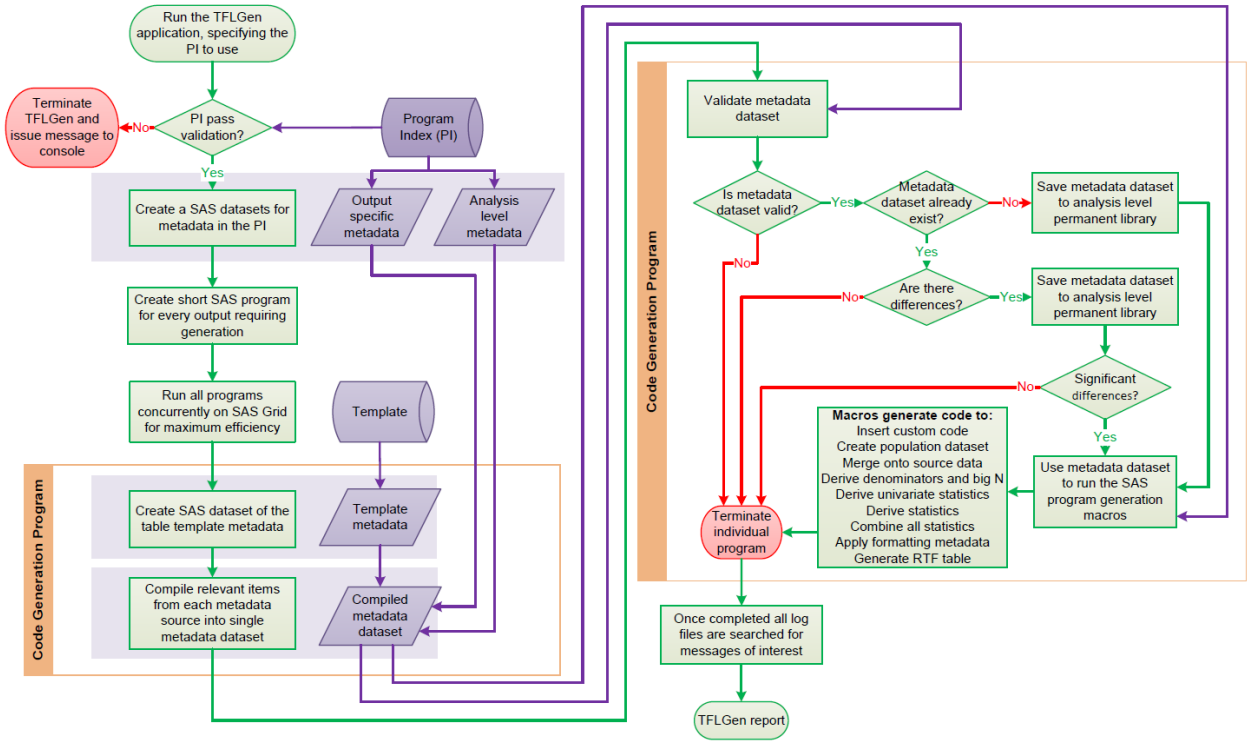
	Comparator		AMG-Active	
	10 mg (N = 21)	20 mg (N = 25)	5 mg (N = 24)	10 mg (N = 24)
Sex - n(%)				
Male	13 (61.9)	12 (48.0)	10 (41.7)	13 (54.2)
Female	8 (38.1)	13 (52.0)	14 (58.3)	11 (45.8)
Age (years)				
n	21	25	24	24
Mean (SD)	55.2 (22.5)	53.6 (16.8)	56.1 (19.0)	52.5 (20.4)
Median	56.0	53.0	55.5	50.0
Q1, Q3	37.0, 69.0	44.0, 63.0	43.0, 71.5	40.0, 65.5
Min, Max	21, 88	20, 87	19, 87	21, 90
Age - n (%)				
< 65 years	14 (66.7)	20 (80.0)	15 (62.5)	18 (75.0)
≥ 65 years	7 (33.3)	5 (20.0)	9 (37.5)	6 (25.0)
Age by sex[1]				
Male subjects - n (%)				
< 65 years	8 (61.5)	11 (91.7)	6 (60.0)	10 (76.9)
≥ 65 years	5 (38.5)	1 (8.3)	4 (40.0)	3 (23.1)
Female subjects - n (%)				
< 65 years	6 (75.0)	9 (69.2)	9 (64.3)	8 (72.7)
≥ 65 years	2 (25.0)	4 (30.8)	5 (35.7)	3 (27.3)

Page 1 of 1

N = number of subjects randomized and dosed in the safety analysis set.
[1] Percentages based on number of subjects of each sex.

Program: /home/nlyon/tflgen_training/exercise/tables/program/t-dm-sum-pas.sas
Output: t14-02-001-001-dm-sum-pas.rtf (Date Generated: 25AUG15:07:45:42) Source: adam.adae

TFLGEN PROCESS FLOW



CONCLUSION

TFLGen increases standard macro code consistency, maintainability, reusability, and extensibility through modularization. It eliminates redundancy (and its inherent risk) in specifying input parameters by using centralized metadata. Table templates can be re-used both within and across analyses/studies/TAs. QC requirements have been minimized (metadata QC vs. dual programming). It increases resource portability between teams by providing a consistent framework for TFL production.

It is expected that TFLGen can be used to produce >65% of tables for a typical analysis. In a test, a single TFLGen - user managed to generate 462 out 697 (66%) tables on a very large study in only 3 days. These tables were produced using 37 table templates. The tables that could not be produced were efficacy tables that TFLGen was not designed to generate. This was the first time these table templates were created. For further analyses on the same product it is anticipated that many of them can be re-used with no or little amendment, further reducing the time to create the tables.

Feedback for TFLGen has been overwhelmingly positive with many teams reporting quicker table production and QC, with further gains expected on future analyses as teams have more experience and a larger pool of pre-constructed table templates.

ACKNOWLEDGMENTS

I would like to thank my colleagues Neville Cope, Jack Fuller, Benno Kurch and Chris St Peter for their invaluable help in designing and coding TFLGen. I would also like to thank David Edwards and Mark Stetz for their patience and support in bringing TFLGen to fruition.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Neil Lyon
Amgen Inc., Uxbridge Business Park
Uxbridge,UB8 1DH, UK
Work Phone: +44 (0)1895 525424
Email: nlyon@amgen.com