PhUSE 2017

Paper AD05

The TFL Workbench: From Pilot to Production

Iain Humphreys, PRA Health Sciences, Reading, UK
Hansjörg Frenzel, PRA Health Sciences, Mannheim, Germany

ABSTRACT

In 2016 PRA presented the TFL Workbench, a tool to standardise and accelerate table, figure, and listing (TFL) programming. Comprising a user front-end for configuring output metadata; the ability to auto-generate reporting code; and a suite of validated SAS® macros, statistical programmers can quickly adapt pre-configured table templates for commonly occurring outputs and automatically generate code to produce the required output.

This paper builds on that previous presentation by describing the tasks that were necessary in taking the tool from early proof-of-concept pilots to a production release. These include: facilitating study team working by developing a better multi-user solution around a central database; formal validation of SAS macros; introducing standards compliance reporting and metrics to gauge the extent to which standards are adopted at the study level; adapting and developing standard operating processes to accommodate the new approach to TFL programming and quality control; and establishing training and support structures.

INTRODUCTION

The TFL Workbench was developed to bring about efficiency gains and quality improvements in the production of tables, figures and listings (TFLs), aiming to overcome some of the drawbacks associated with the traditional programming process. The adoption of TFL standards was seen as the key to this, facilitating automation of those parts of the process lending themselves to automation.

Described more fully elsewhere [1], the initial version of the TFL Workbench that was piloted consisted of:

- Extended TFL metadata, describing tables in a machine-readable and configurable format
- Suite of SAS 'building block' macros, each fulfilling a major function relating to table programming
- Code generation facility, capable of interpreting the metadata and writing a TFL program consisting of calls to the relevant 'building block' macros

Early pilots were successful in proving the concept and confirmed that the tool merited further development. Often in this industry, these developments are led by the biometrics departments who will also be the users of the solution, and utilising tools with which the department is familiar, predominantly SAS and Microsoft Excel®. Yet such initiatives are solutions to business problems which require a broader view on their implementation. As a business investment, there are additional stakeholders and the need to engage with other interested parties (e.g. IT, Quality Assurance, Business Operations). It is likely that sooner or later other languages and technologies may be required, and system design is clearly important in developing a sufficiently robust and scalable solution. Consideration of how the new solution will fit with existing platforms and processes is needed. Attention needs to be paid to operational issues: how the user base will be trained and supported; also to how software will be versioned, deployed, tracked and maintained. Some measures of the impact of the new solution need to be put in place.

These various aspects demand additional time and resource, and in many ways extend the way solutions are traditionally implemented in many biometrics departments. This paper takes five such aspects that needed to be considered in moving the TFL Workbench from its initial proof-of-concept to the first production release, and examines them in more detail: system architecture; software validation; measurement of compliance; development of standard operating processes; and user training and support.

EVOLUTION OF TFL WORKBENCH APPLICATION ARCHITECTURE

Early on during the design of the TFL Workbench, it became clear that in order to reduce complexity in any of its components, we would need to split up the tool into two major parts:

- A set of validated SAS macros that can produce tables, figures, and listings from analysis datasets.
- Any software capable of storing configurations and writing simple text files.

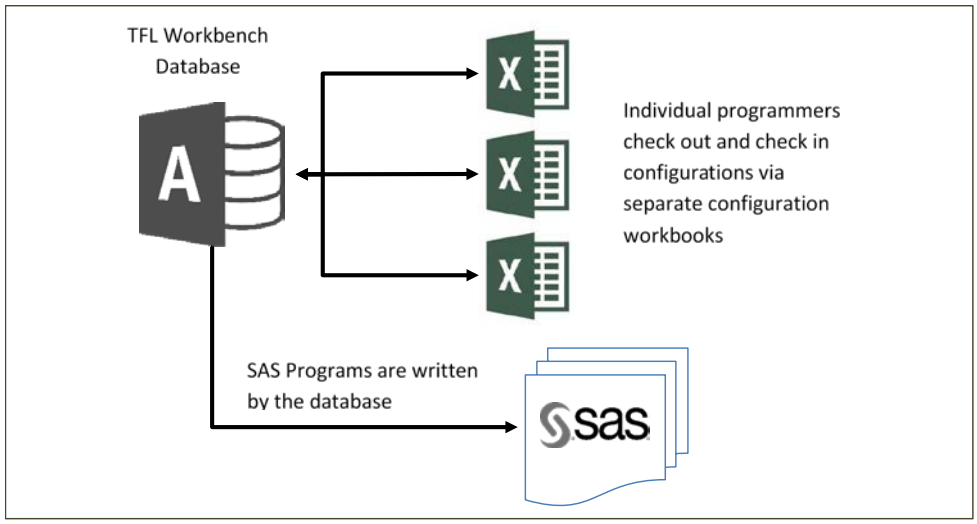
By splitting up the TFL Workbench functionality into these two parts, we allowed each part to do what it is most capable of: SAS Macros are good at generating SAS code flexibly, while a database is good at storing configuration

data, and providing a structured and simple interface.
We started with the SAS part, by discussing how to split the functionality needed to create an output into the units we called 'building block' macros, and decided which metadata needed to be passed to each of those macros, and how we would structure them. It was important to discuss and agree these topics first, as they determined what the database side would need to accomplish in terms of writing SAS code.

For the data storage side Excel was our choice for the first proof-of-concept application, since its tabular nature gave us a very good interface to systematically define metadata for TFL output. After developing the proof-of-concept application, we developed two more major iterations until we abandoned Excel as the sole tool for storing and maintaining output configurations.

Although Excel is a good tool for organizing and manipulating configuration data, it is lacking in multi-user functionality. Two main problems came up when moving from a single-user workbook to a shared, multi-user workbook: in order to format the different spreadsheets efficiently, and to select and sort data, we relied heavily on the tables feature within Excel. When adding new rows to a table, formatting and in-cell validation is propagated automatically. Also, specific features of tables facilitate the selection and sorting of data, without the need to write search or sort algorithms. Unfortunately, in a shared workbook, tables are not supported. Although we were able to develop alternative solutions, none was very satisfactory. The breaking point however was reached when we detected that we had to terminate shared mode in order to create custom templates (in particular, adding or deleting worksheets). The problem was that we could not eliminate the risk of programmers losing work because they were disconnected from the workbook while somebody else created a new template.

For a production version we decided to replace the multi-user workbook with a combination of an Access® database, containing all configurations for a study and the ability to generate SAS code, and an Excel configuration workbook template, which allowed programmers to 'check out' (download) configurations from the database, modify them to adapt them for their study, and 'check in' (upload) the configurations again. Users are encouraged to check in and check out configuration workbooks often, in order to stay up-to-date with metadata that is shared between multiple outputs. Using Excel in effect as the front end onto the database kept interface development to a minimum. However, care was needed to minimise the risk that users could accidently overwrite other users' or shared configurations.



To mitigate the risk of programmers overwriting each other's configurations, we developed

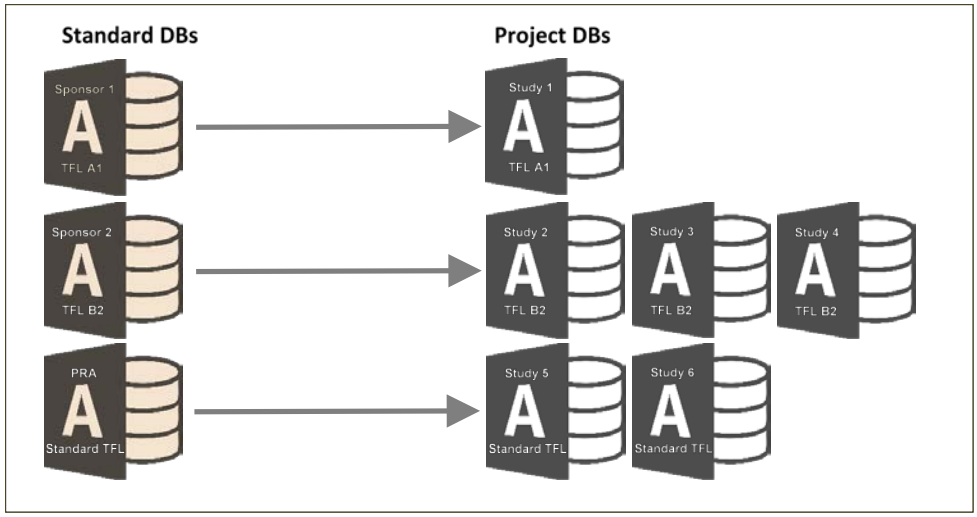
- Incremental check-out/check-in
  - to ensure that programmers only can work on outputs and templates they are assigned to work on. The configuration workbook keeps track of the worksheets that are updated, and the TFL Workbench only updates its tables with contents of updated worksheets.
  - to limit the ability to delete configurations from the database. This makes it possible to update configurations which are shared amongst multiple outputs, like population configurations or treatment group configurations. If programmers need to change some of those configurations, e.g. only the safety population configuration, they can delete all the other configurations in the respective worksheet prior to check-in. The upload process then simply ignores them. Although this does not completely reduce the risk of overwriting each other's configurations, it reduces it substantially. Good communication within the team and well thought through work assignments by the lead are the keys of preventing damage.

- Complete check-out/check-in
    - for the purpose of cleaning up the database or restoring a previous version of the configurations

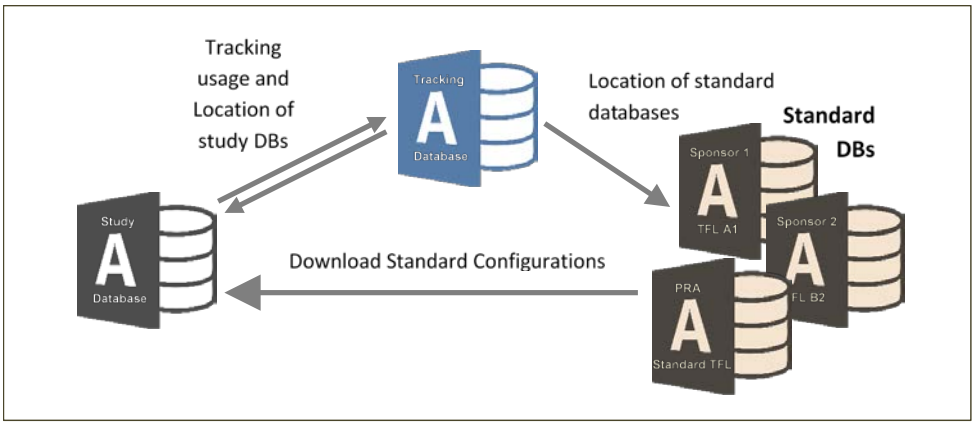
There remained one problem: if programmers use complete check-in with a workbook created via incremental check-out, as a minimum this would delete all templates of outputs that are not assigned to them. We developed a tracking feature for check-outs in order to prevent this from happening.

Retaining parts of the Excel based solution in the form of the configuration workbook template allowed us to re-use some of its functionality and some of its VBA code. In addition, we migrated the code to write SAS programs from the workbook to the database and adapted it to work with its table structure. This meant that we only needed to develop code for importing or exporting data (i.e. checking in and out), plus some minor interface development.

Previously, when using Excel alone, we also could not come up with a good solution for distributing TFL standards to our programming teams. Our goal was to have TFL standard configurations for PRA and sponsor TFL standards, which study teams could download and use in their studies. An obvious solution would have been to keep a separate copy of the Excel workbook for every standard. But this would require maintaining the VBA code and structure of multiple standard data sources, implementing changes to the structure of the TFL Workbench or its code base in every one of those.



In order to distribute sponsor standards, we set up an access-controlled server location, where we stored copies of the current version of the TFL Workbench Access databases, each containing its own PRA or sponsor standard, and entered the locations of these databases and the standards they contain into a separate tracking database. A downloadable copy of the TFL Workbench database contains two tables which are directly linked to this database, but omits all of the tables in which TFL configurations are stored. After a team downloads this copy of the TFL workbench database and open it for the first time, they see a welcome screen requiring them to select a sponsor standard. This selection is based on the contents of the linked tables. Once they click the ‘Retrieve Standard’ button, all tables containing standard configurations are copied from the selected standard database. Once this step is completed, programmers can use all the functionality of the TFL Workbench and start configuring outputs for their study.



In parallel to downloading standard configurations, the study instance of the TFL Workbench also adds an entry into a tracking table containing each database downloaded by a team, providing us with information about its name and location, the standard it contains, and its version. This allows us to easily find databases, and run metrics reports across all studies using the TFL Workbench. It also allows us to introduce measures to prevent cross-copying of configurations from one study to another, which helps to enforce the rule that every project should start from a fresh copy of the standard configurations.

These linked tables allow us to keep the list of available standards up-to-date, without constantly updating the downloadable TFL Workbench. Furthermore, we can now track usage of standards across all distributed databases, and monitor which version of the TFL Workbench is used in different projects.
Most importantly, it allows us to develop code for only one Access database and one Excel based configuration workbook. If changes are made to the configuration workbook, teams can simply download a new version and save it in the same location as the study TFL Workbench, replacing the older version. If the Access database is updated, the study teams can download a newer version, extract all configurations from their old database, and continue using the newer version. These newer versions will still prevent teams from cross-copying configurations from other studies. Finally, all standard databases are maintained by the Global Data Standards TFL team.

FORMAL VALIDATION OF TFL WORKBENCH SAS MACRO SUITE

With the decision to separate SAS macros from the database part of the TFL Workbench, we also separated the validation of the SAS code creating tables, figures and listings from the TFL Workbench database.

User requirements for the TFL Workbench SAS macros were developed soon after we introduced its concept during a meeting with programmers, programming managers, statisticians, and the Global Data Standards team in 2015. Based on this input, we also developed the main functions of the tool alongside the requirements the tool has to meet in order to be accepted by Statistical Programmers, the future users of the application:

- Initialise the programming environment
- Specify treatment columns, stratifications, and population counts
- Retrieve the data to be analysed
- Produce statistical summaries
- Save report-ready data summaries and global macro variables values as permanent SAS datasets
- Create RTF output

For most of these functions one ‘building block’ macro was developed. In the case of generating summary statistics, this main function is implemented in the form of two building block macros: one summarizing continuous variables, and one for categorical variables. The call to each of the building block macros is written by the TFL Workbench database. Building block macros are modular, i.e. each building block macro is comprised of a set of sub-macros. To date the TFL Workbench SAS macro catalog contains 32 macros.

PRA’s Quality Assurance work instruction and reference guide for SAS macro validation only asks for user acceptance testing of any SAS macro. However during design and development of the building block macros and their relationship with each other, it became obvious that there would be too many possible configurations, and too many separate macros to validate them by simply following test cases derived from user acceptance testing, as they were often too unspecific. One example is the user requirement to configure summary statistics for continuous variables by usage of statistical keywords for PROC SUMMARY:

"As a statistical programmer, I want to be able to configure the summary statistics presented in a table, by using statistical keywords available from PROC SUMMARY. Summary statistics shall be presented with two or more results concatenated, or separately one parameter in each row. "

With more than thirty statistical keywords available in PROC SUMMARY it is difficult to test all possible table configurations for summarizing continuous variables. It would simply take too long, and the risks of missing combinations of statistical keywords is too high. Thus, instead of creating all possible tables, we performed unit testing of each macro in the TFL Workbench SAS macro suite to ensure that each performed according to its specifications. After validating the individual macros, we performed user acceptance testing to test whether the user requirements were fulfilled.

While writing the code for the TFL Workbench macro suite the author also produced documentation detailing the macro functionality, what inputs are expected (either as parameters or data, or both), how parameter values are validated, what the output is and how it is structured, and how global macro variables would control the process flow. This documentation was sufficiently detailed to be used as input for developing validation steps. Following the macro documentation, test data and expected outcomes (often also data) were developed, with the focus on making it easy to visually inspect the results. We tailored the test data to the functionality of the tested macro, in order to make it easy for the tester as well as for auditors to decide whether the macro was working correctly. This also allowed us to test the units (macros) independently, ensuring that interactions between macros were tested explicitly. During the planning of the validation we carefully ensured that macros were validated in the appropriate order, i.e. all macros which are called by other macros are validated before the calling macros. All details were documented in a validation plan for unit testing. After the documentation was completed it was reviewed by the macro author to ensure that the test developer had understood the function of the macros correctly.

Although all macros can be tested by a tester visually reviewing the results, we decided to automate the testing, to avoid human error and increase efficiency. Following the validation plan, SAS programs were written which generate the input data as well as the expected output data, contain the planned macro calls, and compare the macro output with the expected output using PROC COMPARE. In addition, input validation of the macro parameters was also tested, and the results were read from the log files produced by the testing programs.

Usually the number of expected output states of the macros were countable and small, thus it was easy to manually configure the different alternative macro calls to produce all expected outputs. The macro producing summary statistics for continuous variables and returning them in the order in which statistical keywords were configured was a clear exception. The documentation for PROC MEANS and PROC SUMMARY alone list 34 different keywords (not counting synonyms for statistics, or all of the possible percentiles). In addition to these 34 keywords we introduced custom keywords in the TFL Workbench to allow calculation of statistics based on the geometric mean, a request made by our statisticians.

The aim of the validation of this macro was not to validate PROC SUMMARY, but rather to test whether the macro would return the results reliably, independent of the sequence in which the statistical keywords were assigned to its parameters. It is not possible to efficiently test all possible combinations for over thirty statistical parameters – each would need to run at least one PROC SUMMARY, which would require a tremendous amount of time and resources on most SAS infrastructure platforms; also it seems to be impossible in SAS to calculate all possible permutations of 30+ values (>2.65 * 10^{32} permutations) efficiently. We decided to limit testing to a minimum of 4 (24 permutations) and a maximum of 9 (362,880 permutations) different, randomly selected statistical keywords. These numbers were in the range of statistical parameters we usually configure for a summary table. We used CALL ALLPERM [2] to create all permutations (CALL ALLPERM can handle up to 15 values (i.e. 1.31 * 10^{12} permutations)), in separate batches depending on the number of statistical key words. To keep the run time short, we randomly selected 50 permutations from each batch, if the number of permutation exceeded 50, and used them to run our summary statistics macro. In each batch the output of the first call of the macro was electronically compared to the output of every other call to the macro, with the expectation that every call consistently returns the same results for each statistical keyword, independent of the sequence of keywords. We used the return code of PROC COMPARE (stored in automatic macro variable &SYSINFO) to determine whether the testing was successful or not, and printed a success message, or the PROC COMPARE output in the case of a comparison failure. This kind of 'brute force' testing revealed problems in the code which had not been detected via code review or the manual configurations used for earlier test steps.

As the individual macros were developed in a modular design, integration testing up to the level of the building block macros was already done during the validation. With the proven confidence that each individual macro works to its specifications, and within its building block macro, we continued with user acceptance testing.

The basis for developing test cases for user acceptance testing were the documented user requirements. To

illustrate the process, one request was to initialize the programming environment to a known state before each output is produced. This is important, as it prevents application settings or results from previously created outputs being carried over to the next output, in the event where a program fails to execute correctly. The request translated into two requirements:

4.1 Clean up the SAS Work Environment

As a Statistical Programmer, I want the SAS environment/work library to be devoid of any result datasets from previous tables or column header variables/macro variables from the production of a previous table. I want everything deleted that could lead to reporting incorrect results.

4.2 Reset Environment to a Well-Known State

All global macro variables should be reset to a known state, such that the macros of the TFL Workbench execute as defined.

These are requirements that need to be tested with every test case, as they apply to each output created by the TFL Workbench SAS Macros. Other requests can be implemented as unique test cases, as they refer to different output types. Six test cases were developed, corresponding to the different types of tables we wanted to create:

- Tables presenting subject counts
- Tables for occurrence data
- Tables for shift data from BDS datasets
- Tables for categorical data from BDS datasets
- Tables for continuous data from BDS datasets
- Tables presenting statistics for both categorical and continuous data

To ensure that the macros performed as intended, we used real world clinical data for the testing. For the macro catalog to pass for each of the six test cases, the user acceptance tester had to electronically compare the outcome of programs written by the TFL Workbench (using building block macros) with the output of manually written programs. With minor amendments to code, all TFL Workbench macros successfully passed validation.

COMPLIANCE REPORTING AND ASSOCIATED METRICS

MEASURING COMPLIANCE TO A STANDARD

One of the overarching aims of the TFL Workbench is to facilitate the adoption of standards for TFLs. As with any standard there is the need to measure compliance: the extent to which the standard is being followed. This is necessary for two principal purposes:

- It aids the TFL QC process, by informing the study teams of the changes made at study level, indicating where special attention should be paid during the review and verification of TFL configurations.
- It permits the owners of the relevant standard to gauge its uptake and make decisions based on this. For example, consistent deviation from a standard might highlight a lack of awareness and therefore a training need; alternatively it might indicate that the standard itself requires amendment.

To meet these needs, a compliance report has been developed and implemented. The starting point for the deployment of the TFL Workbench at individual study level is a copy of the Access database, populated with the relevant standard configurations (sponsor or PRA). Once study-specific configuration changes have been made to generate the outputs required for a study, the compliance report is run to compare the study instance of the database with the standard database on which it was originally based.

The report was developed as a SAS macro, with sub-macro utilities to:

- read the Access databases and retrieve configuration details;
- make the comparisons and flag changes;
- report the configurations used on a study, and highlight the changes detected.

Since the compliance report plays an important role in the QC process (covered in more detail later), these macros were subjected to formal validation in the form of user acceptance testing.

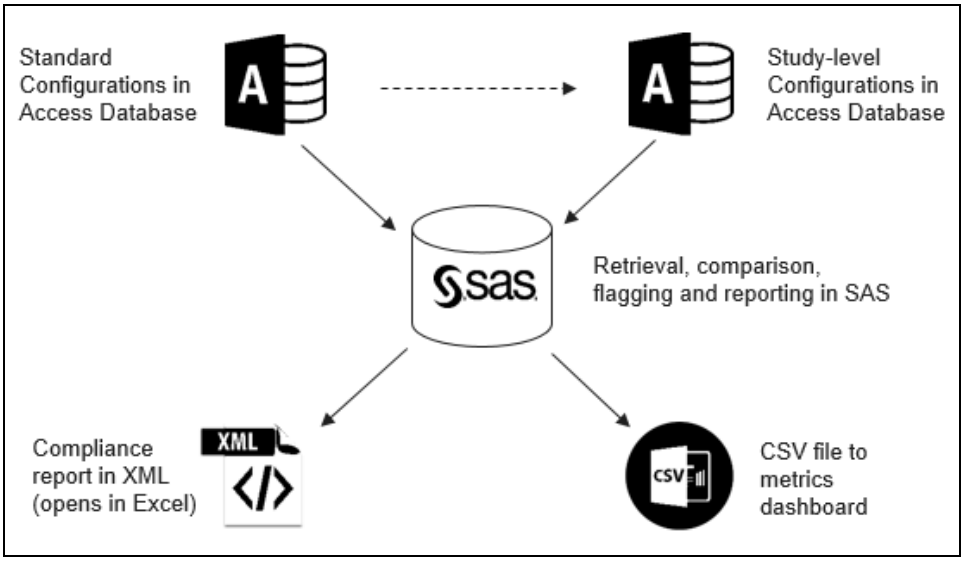
The actions of the compliance report are as follows: the name and location of the study-level TFL Workbench database are passed to the primary macro. This database is queried to determine which standard is being followed, and which template configurations have been used in the study reporting. Using this information, the macro locates and retrieves the relevant ‘master’ configurations from the standard database, and performs a comparison of each

individual item of metadata. In addition to comparing the individual table templates (each analogous to a unique table shell), other TFL Workbench entities are compared, including treatment sets, population sets, pre- and post-processing code snippets, and other settings. One of the benefits of using a database such as Access to house the TFL metadata configurations is that this can be more simply and reliably read in SAS, compared with Excel (another factor in TFL Workbench architecture decisions).

Changes that are detected between the standard and the study instance are classified as being merely **cosmetic** (e.g. a change to a row label, or column header), or **data-related** (i.e. a change which would impact the numbers being presented). These changes are stored as a series of flags and used in the subsequent reporting.

The entire TFL metadata for a study is written to the compliance report, which is in the form of an XML file, produced via the EXCELXP ODS tagset.

An overview of the compliance reporting process is shown in the diagram below.



When the report is opened in Excel, the user can see a separate worksheet for each template used in the study, depicting original standard configuration on the top half, followed by the study configuration on the bottom half, and colour-coded differently. Any changes are highlighted in different colours in the study-level settings. The example below shows the configuration for a concomitant medication summary which has been altered. The standard version of the table specifies both preferred term and therapeutic class. The study summary presents only preferred term, so the TFL Workbench class variables have been updated, and also the corresponding row labels, their header, and level of indentation. A change to a class variable is categorised as data-related, and therefore colour-coded differently to the other changes which are classified as cosmetic.

Source	Section	Row	Metadata	RowLabel1	Indent1	Stati-stic	Class Var	Analysis Vars	Where	Denomi-nator
1	-1	.	row label hdr	Therapeutic Class!Preferred Term	0!2					
1	10	1	statistic	_Row_Label_From_TOO_	0	n_pct		UNIQUE(_USUBJID_)		_BigN_
1	20	1	statistic	_CLASS1_	0	n_pct	CMCLAS	UNIQUE(_USUBJID_)		_BigN_
1	20	5	statistic	_CLASS2_	2	n_pct	CMCLAS#CMDECOD	UNIQUE(_USUBJID_)		_BigN_
2	-1	.	row label hdr	Preferred Term	0					
2	10	1	statistic	_Row_Label_From_TOO_	0	n_pct		UNIQUE(_USUBJID_)		_BigN_
2	20	5	statistic	_CLASS1_	0	n_pct	CMDECOD	UNIQUE(_USUBJID_)		_BigN_

Colour-coding is achieved during the reporting step, using CALL DEFINE statements in PROC REPORT compute blocks to set cell style attributes conditionally, according to the change flags previously derived. The standard process for using the TFL Workbench requires a review of configurations (discussed more fully in a later section); by highlighting changes the compliance report assists study teams, directing them to configurations that need additional special attention.

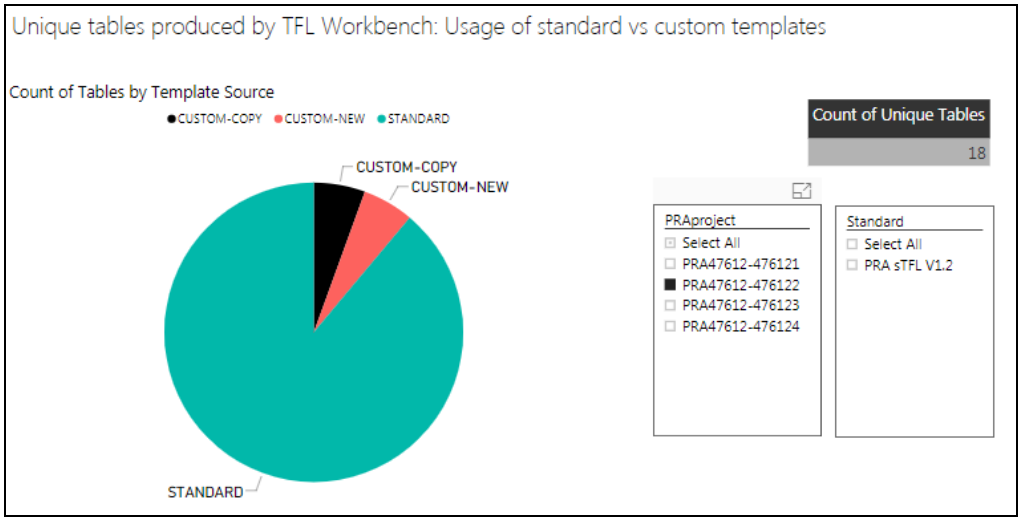
DERIVATION AND REPORTING OF METRICS

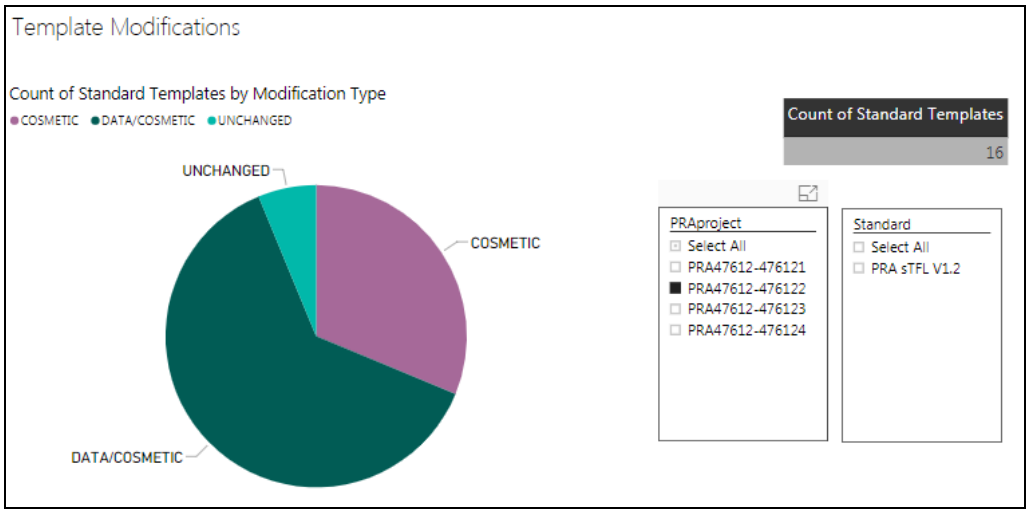
Whereas the study team will run the compliance report for their individual study, the Global Data Standards team has the facility to run the same report over multiple studies' TFL Workbench databases. When run in this mode, the retrieved metadata and related change flags are stored permanently for each study. The main purpose of this is to drive metrics reporting. The compliance data from individual studies are pooled, integrated with information from other sources and converted to a comma-separated variable (CSV) file; this is periodically uploaded to a repository in order to refresh a metrics dashboard.

The metrics dashboard is published to the PRA intranet and includes the following metrics in the displays:

Measure	Significance
Number and percent of tables produced by the TFL Workbench, versus those traditionally programmed	A measure of coverage: the extent to which TFLs can be produced by the TFL Workbench
Number and percent of TFL Workbench-produced tables based on standard templates, versus those based on custom templates	A measure of the degree to which the standard templates met the study reporting requirements
Number and percent of TFL Workbench-produced tables based on standard templates that are unmodified, versus those with cosmetic changes, versus those with data-related changes	A measure of the degree to which standard templates needed study-specific amendment
Number of tables requiring pre-processing and post-processing code	A measure of the need to overcome a constraint of either the TFL Workbench or the source data

These metrics can be filtered by the study ID and by the standard being followed, to permit groupings to be examined. Some examples from the metrics dashboard are shown in the panels that follow:





Over time such metrics will play a role in the governance of TFL standards, alerting their custodians to any persistent deviation from the standard, and facilitating a root cause analysis. Judgments can then be made about possible corrective actions needed to ensure higher levels of future compliance.

PROCESS DEVELOPMENT

The TFL Workbench clearly represents a departure from traditional TFL programming practice, and standard operating procedures (SOP) have had to be adapted accordingly. Since not all TFL outputs for a clinical study will be produced by the TFL Workbench, existing processes were reviewed and expanded to incorporate the new practices. This exercise was carried out in conjunction with the process owners, specialists and PRA's Quality Assurance (QA) department, who scrutinised every step suggested by the development team. The involvement of QA was essential to ensure that adjacent processes were not adversely impacted, that standard industry practices were not violated, that SOP standards were met, that an appropriate amount of documentation was available, and that the process would withstand auditory inspection. Finally, in order to permit traditional TFL programming in conjunction with the TFL Workbench, a new work instruction was drawn up to complement traditional SAS TFL programming.

Firstly a decision is taken by the Lead Statistical Programmer, based on comments from the Lead Biostatistician and the annotated output shells, whether to produce a summary table with the TFL Workbench or not. If not, the traditional TFL programming process is followed, and the rationale for this decision is documented.

For a table to be generated using the TFL Workbench, certain tasks are carried out by the primary programmer allocated to produce the output. These tasks are analogous to the generic steps of:

- choosing the correct population
- specifying the correct set of treatment columns
- selecting the correct records for analysis from the relevant dataset
- performing frequency counts and percentage calculations on categorical data
- producing summary statistics on continuous data
- assembling a report-ready table dataset
- reporting the output, with associated paging and titling considerations.

With the exception of the last of these steps, traditional table programming has been replaced by configuring the TFL Workbench and using its code generation facility to produce a program. The starting point is always the standard configurations which are then adapted to meet study-specific needs if necessary. Custom configurations can be created in the absence of a standard template being available, for tables lending themselves to automation. Changes are made according to the Statistical Analysis Plan (SAP), annotated TFL shells, and source data (e.g. (ADaM) specifications. Although manual program coding tasks have largely been replaced by checking and amendment of table configurations, a sound knowledge of the study and thorough understanding of the analysis datasets is still required on the part of the primary programmer, as is the normal diligence in ensuring the table meets the study requirements prior to releasing to QC.

Major changes are also apparent in the QC process: much of the independent double-programming that is carried out when outputs are traditionally programmed has been replaced by review of TFL Workbench configurations by a QC reviewer (a programmer other than the primary programmer). Using the compliance report, the settings are

checked to verify that they are appropriate for each output. If cosmetic changes have been made, the formatting is checked; if data-related amendments were made, the relevant variables, statistic names, where clauses are all inspected for correctness. A few results are also spot checked, and the usual visual checks are still carried out to verify that each output matches its shell.

Additionally, and regardless of method of production, all outputs undergo review by the lead programmer, lead statistician and a senior statistical reviewer to ensure consistency in numbers between outputs, that all intended outputs are present, and that the results make sense.

The ability to replace much of the need for independent double programming is a direct consequence of having formally validated macros in the TFL Workbench, as described earlier.

As with changes to any standard processes, the documentation must be subjected to the formal QA approval process, and implementation must be planned in alignment with the release cycles for such processes. The communication of the process changes to the programming user base, and the availability of relevant training also need to be planned accordingly.

TRAINING AND SUPPORT FOR THE USER BASE

Reaping the potential benefits of the TFL Workbench relies on the statistical programming community firstly adopting it and then using it optimally. To help meet this aim, a number of materials have been prepared, using a variety of media.

INITIAL TRAINING AND EDUCATION

The implementation of the TFL Workbench is ongoing on a standard-by-standard basis, as and when libraries of different sponsor standards are configured and made available. When study teams are ready to start using the tool for the first time, instructor-led training is carried out. This covers the following topics:

- introduction and rationale for the TFL Workbench
- initial set-up of a TFL Workbench database and the programming environment
- how to auto-generate and run a TFL program with the TFL Workbench
- how to check out configurations, make study-specific updates, and check them back in
- how to run a compliance report
- programming process changes

Delegates carry out hands-on exercises in a training area to give them direct experience. Training is timed to coincide with the start of TFL programming. However, for teams using the TFL Workbench for the first time, Global Data Standards will review the SAP and TFL shells and advise on any special considerations, including how data might be arranged in ADaM for subsequent optimal use of the tool. This review would take place ahead of training, to coincide with ADaM programming. We will also work with Lead Statistical Programmers who are first-time users, in the initial set up of their study. In the future we plan to develop self-training modules, so that programmers can access on-line training on demand.

SUPPORT STRUCTURES

Once programmers are trained and using the TFL Workbench, a number of resources are available to them:

- A User Guide has been developed. This contains all instructions needed for set-up; it also provides information on the different TFL metadata entities: Table of Outputs (TOO), population sets, treatment sets, study-level metadata, pre- and post-processing. Additional notes explain the standard template configurations for commonly occurring output types. Finally, hints and tips for how to handle special situations such as cross-over or other multiphase studies, or tables with adverse event rates.
- A TFL Workbench 'Wiki' is being developed with detailed information about each TFL standard that is configured and available. It is intended to show example outputs resulting from standard templates.
- An intranet-based Issues and Enhancements tracker is available, in which problems can be reported, questions asked, or ideas submitted. Automatic alerts to the Global Data Standards team ensure a timely response to issues.
- A monthly User Forum is held, to which all TFL Workbench users are invited. Global Data Standards will typically present a TFL Workbench-related topic, often to refresh or reinforce an aspect of the training; this is followed by an open session in which attendees are encouraged to ask questions, exchange ideas and give feedback.

As the user base grows we anticipate increased demand for such resources. To meet that demand early adopters of each standard available in the TFL Workbench will become subject matter experts (SMEs) and assist in providing support to their peers.

CONCLUSION

The five topics discussed above were all, amongst others, pre-requisites to the successful deployment of the TFL Workbench solution to study teams:

- Providing a platform sufficiently robust to support a multi-user study team required an architectural change: the migration from Excel to a database. This also facilitates the access to TFL metadata for other purposes such as compliance reporting.
- To achieve efficiency gains by reducing independent double programming required software to be formally validated; in turn this permitted QC to be streamlined to a review of configurations and outputs.
- The solution relies on the adoption of standards, since it is this that permits automation and its associated benefits of deceased programming time and increased consistency. In order to monitor the uptake of standards it was necessary to develop the compliance report and its related metrics.
- The compliance report also plays a part in the QC process, which along with other standard operating procedures needed to evolve to incorporate the usage of a new tool to produce TFLs.
- Appropriate materials and support structures were needed, to help ensure optimal use of the TFL Workbench and thereby achieve the maximum efficiency gains.

All of these topics form part of a broader applications development landscape, of which programming the solution is just one part. As such they extend the horizons of the biostatistical programming function, sometimes requiring additional resources, new approaches and possibly new skills also.

REFERENCES

[1] Humphreys, Iain and Frenzel, Hansjörg. 2016. “The TFL Workbench: Tools to Standardise and Accelerate Table, Figure, and Listing (TFL) Programming”, PhUSE 2016, AD01

[2] Wicklin, Rick, "Generate all permutations in SAS", in: The Do Loop, Oct. 31, 2010: http://blogs.sas.com/content/iml/2010/10/13/generate-all-permutations-in-sas.html

RECOMMENDED READING

A good description of how to use the value stored in &SYSINFO for efficient display of PROC COMPARE results is provided in: Hinson, Joseph; Margaret Coughlin: "Deciphering PROC COMPARE Codes: The Use of the bAND Function". SAS Global Forum 2012.

CONTACT INFORMATION

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

Iain Humphreys
PRA Health Sciences (UK)
500 South Oak Way
Reading, RG2 6AD
Email: HumphreysIain@prahs.com
Web: www.prahs.com

Hansjörg Frenzel
PRA Health Sciences (Germany)
Gottlieb-Daimler-Strasse 10
Mannheim, D-68165
Email: FrenzelHansjoerg@prahs.com