Paper TT10

TFL Automation the CDISC 360 way

John McDade, PHASTAR, Glasgow, United Kingdom
Alison Macnab, PHASTAR, Glasgow, United Kingdom

ABSTRACT

Traditional programming practices lock vital metadata away within the code. Once this happens, even within macros, it is very difficult to recycle this efficiently and automation is lost. Keeping the metadata distinct from the codebase allows smart metadata management within and across studies.

This paper looks at a journey starting in an Excel driven process, similar to CDISC 360 TFL proof of concept, and how this evolved into a user interface to provide ultimate flexibility for building TFLs. By combining the modularised CDISC 360 approach with a modern web application, we can overcome some of the limitations with excel storage and manage the required metadata much more efficiently.

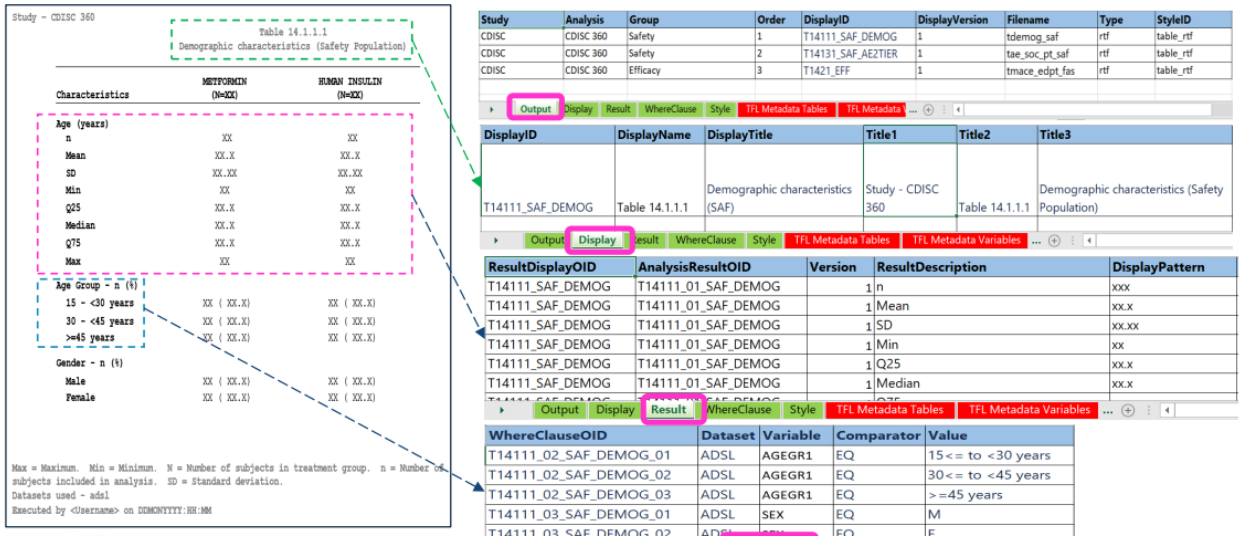
Using enriched TFL metadata, we can write open code TFL programs capable of generating most study CSR outputs.

INTRODUCTION

The CDISC 360 initiative [1] is aimed at "implementing standards as linked metadata with a conceptual foundation providing the additional semantics needed to support metadata driven automation across the end-to-end clinical research data lifecycle."

While CDISC 360 looks across SDTM, ADaM and TFL generation, this paper will concentrate on the TFL creation aspect.

To enable automation, we must understand the building blocks that make up each output; below is an example of breaking this metadata down. Once TFL components are modularised, a section can easily be added, removed, updated as needed based on the given study TFL mock shells.



The example above is part of the CDISC 360 proof of concept and shows the breakdown of metadata in Excel. This is also where we started the journey before moving into a web-based tool.

METADATA MANAGEMENT

PROOF OF CONCEPT

After walking through our own proof of concept with some hardcoded metadata to create some simple TFLs, it was clear that a process built on Excel tabs was not sustainable. Excel is an intuitive medium that everyone can easily use and understand but is limited in the world of metadata management.

For a pharmaceutical company with many ongoing studies in the drug development lifecycle, having standard TFL shells would be normal practice to try and streamline internal processes. This not only ensures consistency at the TFL review stage but also filters backwards through ADaM and SDTM standards and even the database design stage. Throw in different therapeutic areas and their particular needs, managing TFL standards is a real challenge. Now imagine a Clinical Research Organisation trying to manage not only their own internal standards but also using and consuming X number of sponsor standards.

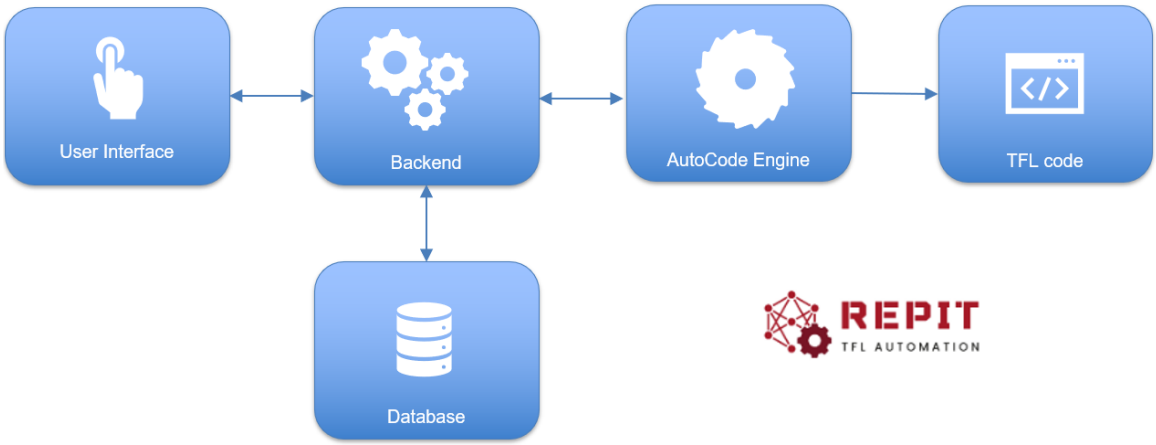
With a background against ever changing standards, it was clear that metadata management would need to be equally dynamic to meet requirements. A decision was made to develop a web-application to support the TFL automation process.

WEB APPLICATION

There are many good reasons to develop a web application for TFL automation but the primary one was around the user experience. Having the ability to build outputs in the system and customise as required meant working with an intuitive user interface. This would also allow logical separation within different screens of the various metadata components.

The application also provides security, scalability and robust metadata management. Users can be assigned credentials which can be associated at the study level, with administrative rights held for developers and super users. The database ensures metadata management at client and study level and templates we apply for each output.

Below is a high-level overview of the application components. Internally the tool is called REPIT and this is used throughout the rest of the paper:



PROCESS WORKFLOW

It is useful to look at the bigger picture of the CDISC 360 TFL Automation proof of concept workflow in Figure 1 and compare with our current workflow in Figure 2. The workflows are broadly similar where an application is processing TFL metadata in some format to produce SAS programs and an ADaM define.xml, both using the concept of templates in the process.

One of the questions we asked during development was, “What should be the starting point for the tool. Is it TFL shells? Is it Analysis Results Metadata (ARM)? Or does it matter?” The Study Shells annotated with required ADaM metadata and the ARM should reflect much of the same information and there should be a pathway between the two. The ARM can certainly be the driver but thought must be given to how this is generated to begin with and how fast

that can be achieved. Other factors to consider would be standardization of shells, however, this would look quite different between a pharmaceutical company and a CRO.

We have found that starting from the Study Shells and building the TFL in the application and storing the metadata against a given template, we can export that metadata in the form of ARM. With that metadata stored against the template ID, building the output becomes a one-time operation. The concept of templates is key in the whole process and covered more in the Output Templates section.

In our workflow in Figure 2, the dashed lines are current development points. There is some compliance checking against the ADaM data but more useful is checking against ADaM Define.xml or even feeding into the Define / Specification creation process. If you have a standard library of Shells with ADaM metadata associated, this can feed into the specification process and avoid ADaMs – TFLs becoming out of sync.

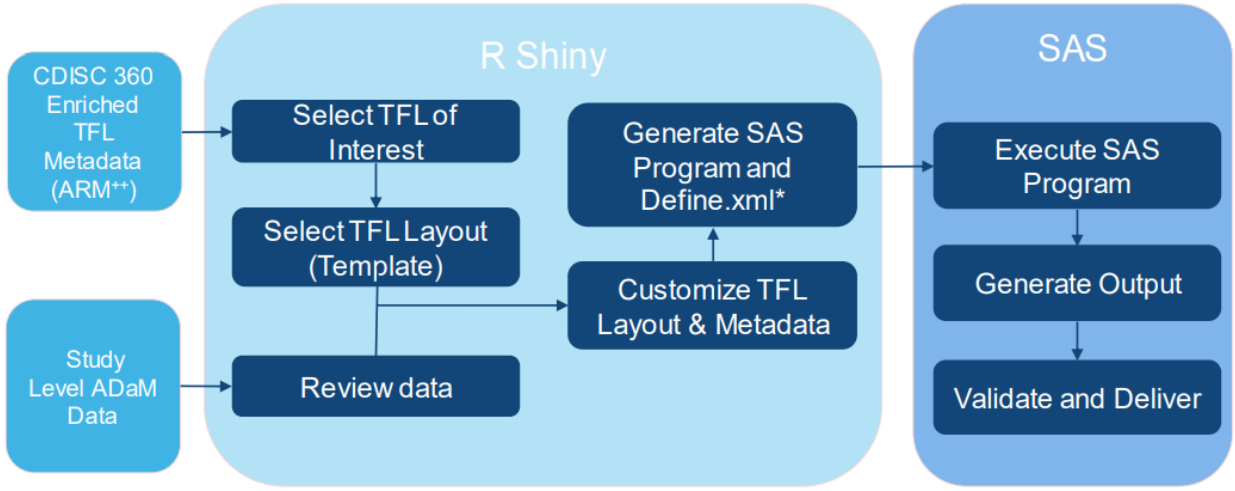


Figure 1: CDISC 360 TFL Automation proof of concept workflow.

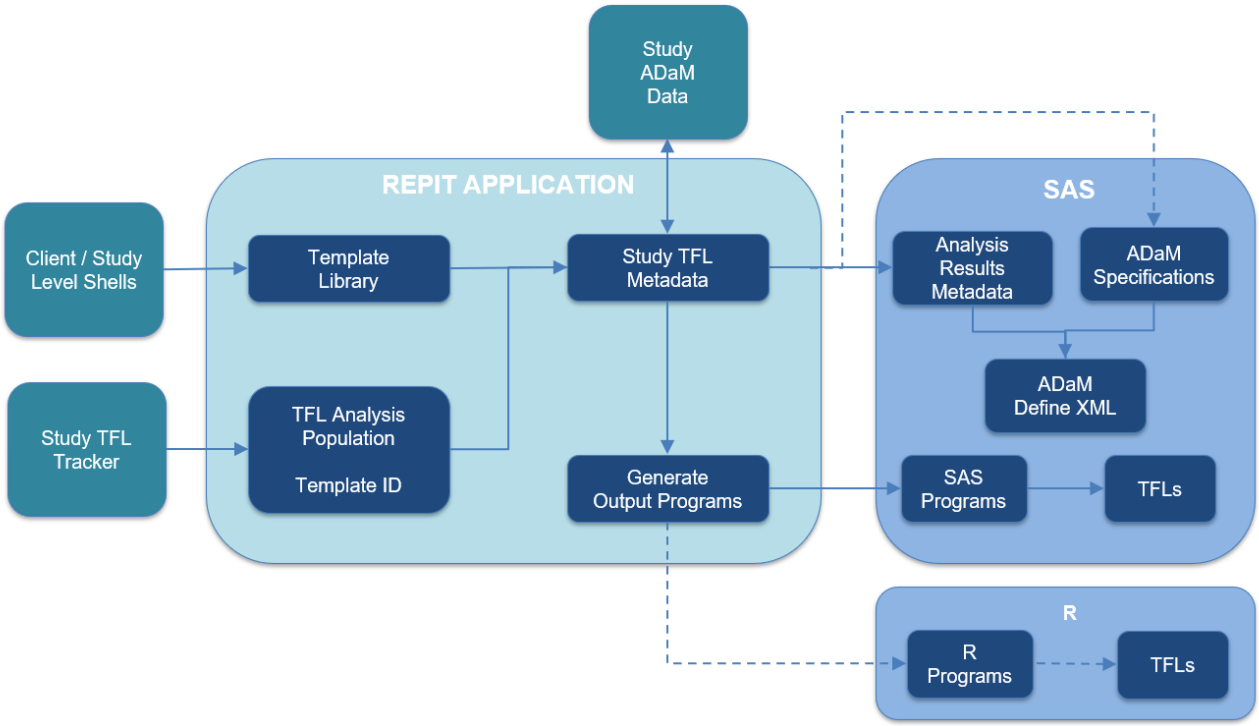
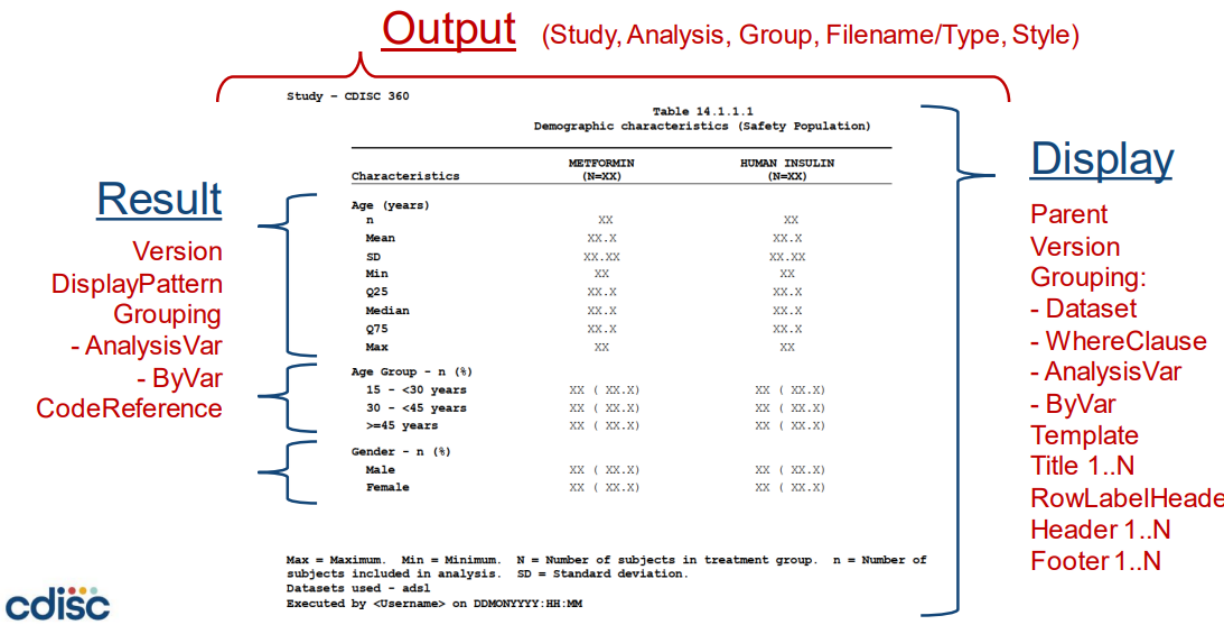


Figure 2 REPIT TFL Automation workflow.

USER INTERFACE

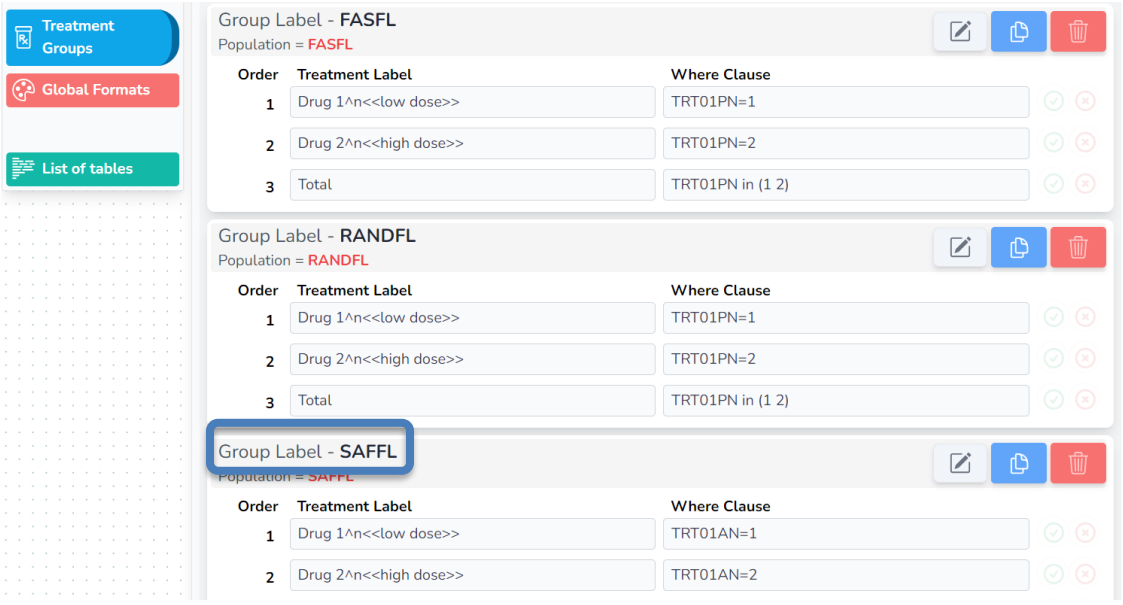
TFL COMPONENTS

To control the final output display, we need control over the individual components as given in the CDISC 360 breakdown below. This is the CDISC Analysis Results Metadata v1 with Extensions [1] required to create the final output. We will look in the next few sections how these are controlled from the User Interface.



TREATMENT GROUPS

The 'Treatment Groups' tab is a centralised location where we can store all the treatment group information required for a study. These can be defined for a study population and are easy to edit on a per study basis for relevant ordering of treatment, the treatment label and the ADSL where clause applicable. The number of groups applicable to a population can be altered and the outputs will be auto-sized to produce a constant width and central alignment of statistics.

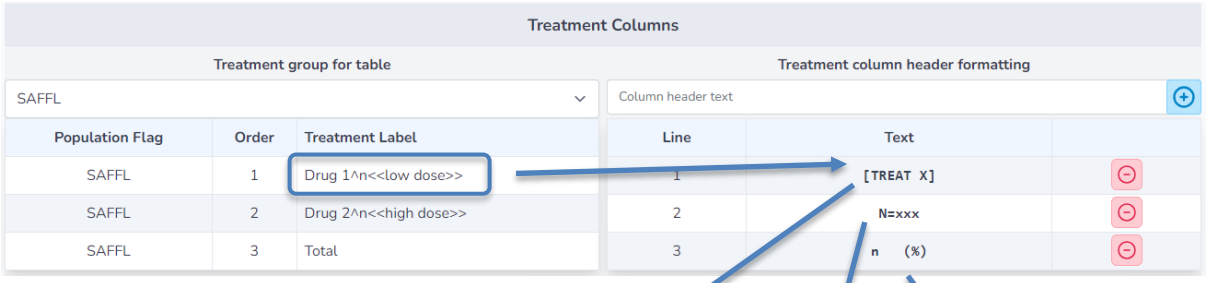


Our outputs are stored in a central online tracking document which is linked to the REPIT tool. Each output has an associated population (containing the ADSL population flag) which is matched with the Group Label to automatically assign the relevant population metadata per output.

Output Type	Output Number	Title	Population	REPIT Template Category	REPIT Unique Template ID
All Adverse Events					
Table	14.3.2.1.1	Overall summary of adverse events - subject count <<-Period>> (Safety set)	SAFFL	EVENTSUM	PHAE01
Table	14.3.2.1.2	Overall summary of adverse events <<optional text>> - subject count ([[Safety set]])	SAFFL	EVENTSUM	PHAE01A
Table	14.3.2.1.3	Overall summary of adverse events <<optional text>> - event count ([[Safety set]])	SAFFL	EVENTSUM	PHAE01B
Table	14.3.2.2	Adverse events by system organ class and prefered term <<- Period>> (Safety set)	SAFFL	EVENTS2	PHAE02A
Table	14.3.2.3.1	Adverse events of special interest by system organ class and preferred term <<- Period>> (Safet set)	SAFFL	EVENTS2	PHAE02B
Table	14.3.2.4	Adverse events sorted by decreasing frequency on preferred term level <<- Period>> (Safety se	SAFFL	EVENTS1	PHAE03
Table	14.3.2.5.1	Adverse events by maximum intensity on preferred term level <<- Period>> (Safety set)	SAFFL	EVENTS3	PHAE04

A huge benefit to storing the treatment group information in a centralised location like this, is that any updates required for the treatment groups can be made in this one location, and then automatically applied to all the outputs.

When creating/editing the metadata for a specific output within the user interface, the user is able to select a treatment group and then choose how they want the treatment columns to look. Here's an example (using dummy data) showing what an output produced by REPIT looks like that uses the SAFFL treatment group, with the following treatment column header formatting:



System organ class Preferred term (MedDRA version 25.0)	Drug 1 <<low dose>> N=8 n (%)	Drug 2 <<high dose>> N=9 n (%)	Total N=17 n (%)
Any Adverse Events	8 (100)	9 (100)	17 (100)
INFECTIONS AND INFESTATIONS	5 (62.5)	5 (55.6)	10 (58.8)
COVID-19	3 (37.5)	3 (33.3)	6 (35.3)
Fungal skin infection	1 (12.5)	0	1 (5.9)
Herpes virus infection	1 (12.5)	0	1 (5.9)
Herpes zoster	1 (12.5)	0	1 (5.9)
Latent tuberculosis	0	1 (11.1)	1 (5.9)

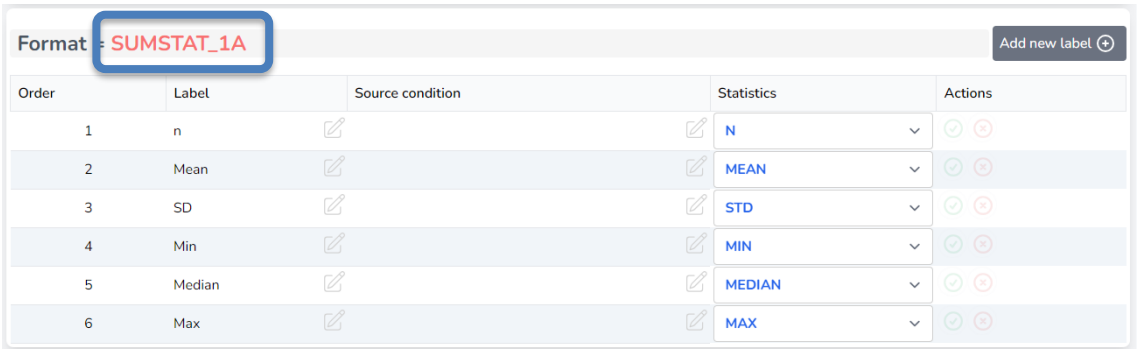
This process allows us to dynamically build the column headers as required. The '[TREAT X]' automatically brings in each treatment label, 'N=xxx' brings in the Big N counts for each column and we have some additional formatting on level 3 of the header. These can be edited / added / removed as required but will generally be standard within a given set of shells.

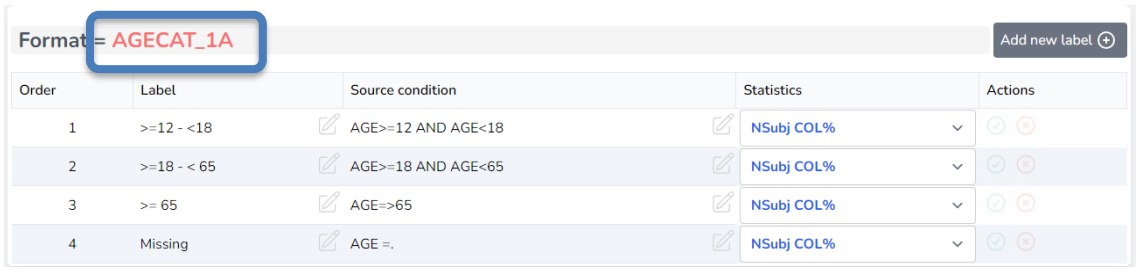
OUTPUT FORMATS AND STATISTICAL METADATA

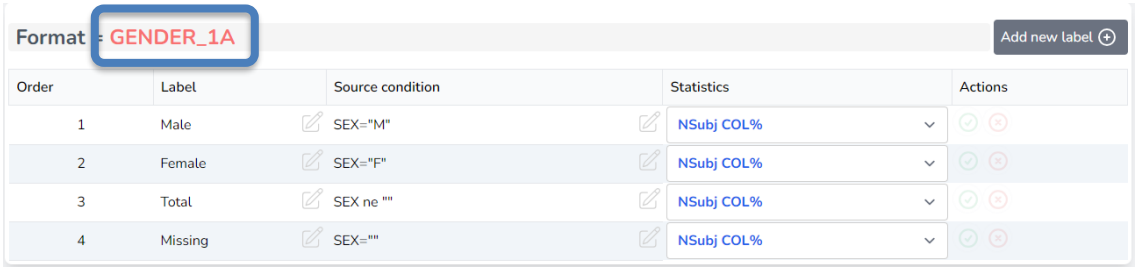
The 'Output Formats' tab in the UI is where we can store formats that we need to use in our output metadata. For example, to create this Demography table, we need to use 3 different formats.

		Statistic	Drug 1 <<low dose>> N=8	Drug 2 <<high dose>> N=9
Age (years)		n	8	9
		Mean	68.9	67.9
	1	SD	7.57	5.51
		Min	52	62
		Median	71.5	66.0
		Max	77	75
Age group (years)				
>=12 - <18		n (%)	0	0
>=18 - < 65	2	n (%)	1 (12.5)	4 (44.4)
>= 65		n (%)	7 (87.5)	5 (55.6)
Missing		n (%)	0	0
Sex				
Male		n (%)	6 (75.0)	7 (77.8)
Female	3	n (%)	2 (25.0)	2 (22.2)
Total		n (%)	8 (100)	9 (100)
Missing		n (%)	0	0

The first of these is a summary statistics format called SUMSTAT_1A followed by categorical formats, AGECAT_1A and GENDER_1A. These formats hold a finite collection of metadata that can be recycled across outputs and studies.

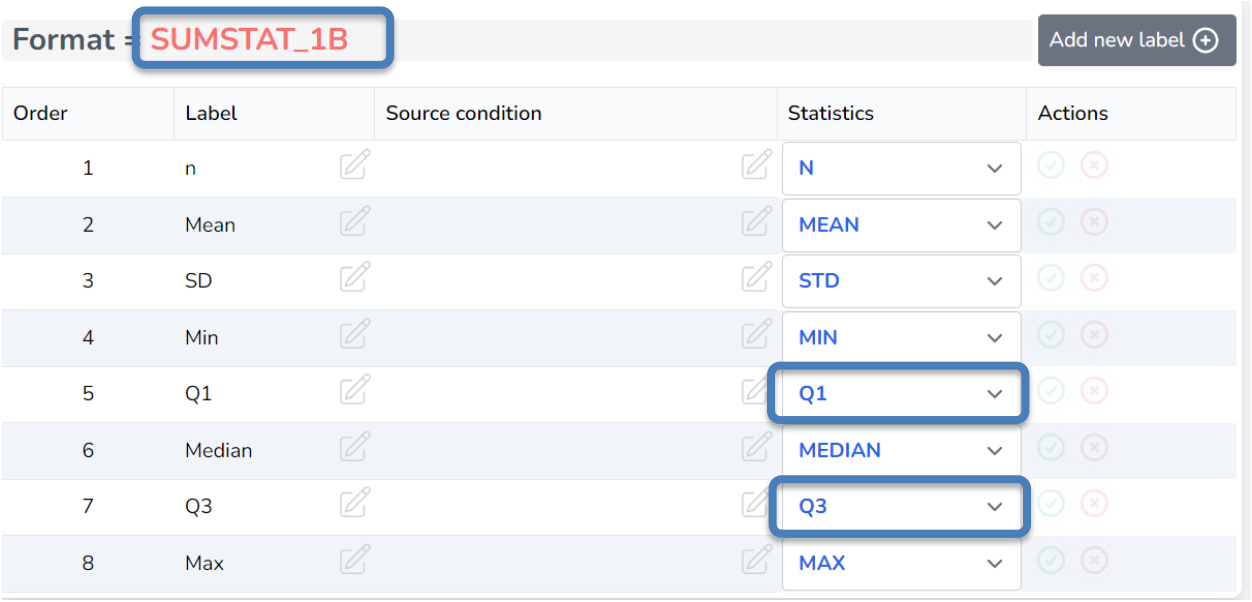






The emphasis here is on the flexibility of approach, our outputs by default will start with a generic format but there are several options to update this table section. We have additional formats which will contain logical groupings of summary statistics or categories, and these can be switched in for the default value. If a global change is required, the default format can be edited and this will update through to all outputs using that format.

Looking at the example below, SUMSTAT_1B is another stored format which adds the first and third quartiles. This can be switched in for SUMSTAT_1A without worrying about several code updates – it will just flow through to the code and output.



The first section in the output above now has the first and third quartiles included with a simple switch of format used:

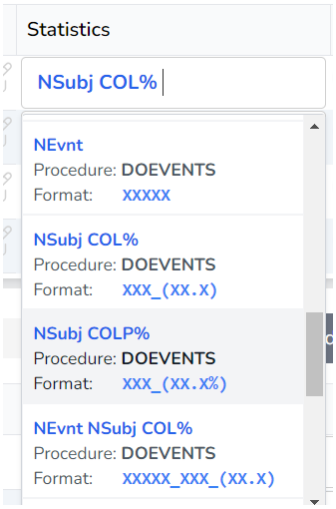
	Statistic	Drug 1 <<low dose>> N=8	Drug 2 <<high dose>> N=9
Age (years)	n	8	9
	Mean	68.9	67.9
	SD	7.57	5.51
	Min	52	62
1	Q1	67.0	63.0
	Median	71.5	66.0
	Q3	72.5	73.0
	Max	77	75

The formats themselves are linked to row block metadata which associates the relevant ADaM datasets and variables required along with some labels and formatting options.

Block Level	Block Order	Indent	Block title	Source Table	Source Column	Block Filter	Format	Decimal	Sort By
1	1	0	Age (years)	ADSL	AGE		SUMSTAT_1A	0	
2	0	0	Age group (years)						
2	1	1	n (%)	ADSL	AGE		AGECAT_1A		
3	0	0	Sex						
3	1	1	n (%)	ADSL	SEX		GENDER_1A		

For each of the formats used, there is a drop-down list of statistics which can be updated as required. These will either be summary statistics (N, MEAN, MIN, MAX etc) or display formats available for categorical counts.

For something like Adverse Events tables, this gives the option to quickly move between events counts, subject counts and percentages. The order and formatting of each of these can be controlled by a specific selection including dropping of any count not required or display of things like % sign as per client preference.



OUTPUT TEMPLATES

The concept of output templates is very important within the REPIT framework. There is a group of core templates available which are useful for building outputs within the tool. The templates provide default metadata for creating a particular output and this value also triggers some options in the SAS code generation.

Template ID	Template Category	Template Description
EVENTS1	EVENTS	Events table presenting 1 dictionary variable
EVENTS2	EVENTS	Events table presenting 2 dictionary variables, nested
EVENTS3	EVENTS	Events table presenting 2 dictionary variables, nested, plus additional category variable, e.g., AE severity
EVENTSUM	EVENTS	Summary table presenting counts of subjects per event category
FINDINGA	FINDINGS	Findings table presenting the summary statistics as columns, with column headers containing treatment group labels
FINDINGD	FINDINGS	Findings table presenting the summary statistics as rows, with ID column containing statistics labels
SUMMARY	GENERAL	Summary table presenting frequency counts and/or summary statistics

While the default metadata supplied by the core templates creates a useful starting point for building outputs, this leads on to the second concept of specific templates which is used to load outputs directly. If you have a standard set of shells, the ADaM metadata and display format required can be pre-populated within a template.

The templates are associated with each output in the tracking document, as these are loaded to the REPIT tool, the metadata associated with each output is loaded from the template library.

Output Type	Output Number	Title	Output Name	REPIT Template Category	REPIT Unique Template ID
All Adverse Events					
Table	14.3.2.1.1	Overall summary of adverse events - subject count <<-Period>> (Safety set)	ae200	EVENTSUM	PHAE01
Table	14.3.2.1.2	Overall summary of adverse events <<optional text>> - subject count ([[Safety set]])	ae200a	EVENTSUM	PHAE01A
Table	14.3.2.1.3	Overall summary of adverse events <<optional text>> - event count ([[Safety set]])	ae200b	EVENTSUM	PHAE01B
Table	14.3.2.2	Adverse events by system organ class and prefered term <<- Period>> (Safety set)	ae201	EVENTS2	PHAE02A
Table	14.3.2.3.1	Adverse events of special interest by system organ class and preferred term <<- Period>> (Safety set)	ae202	EVENTS2	PHAE02B
Table	14.3.2.4	Adverse events sorted by decreasing frequency on preferred term level <<- Period>> (Safety set)	ae203	EVENTS1	PHAE03
Table	14.3.2.5.1	Adverse events by maximum intensity on preferred term level <<- Period>> (Safety set)	ae204	EVENTS3	PHAE04
Table	14.3.2.5.2	Adverse events by maximum CTCAE grade on preferred term level ([[Safety set]])	ae204a	EVENTS3	PHAE04A

SAME METADATA DIFFERENT TEMPLATE

Another useful approach using templates can be switching the output format with ease without changing the input metadata. The three tables below are essentially different views on the same metadata. Having a designated template for each view means that moving between them is a simple change in template name applied and covers different client preferences efficiently.

Table 14.3.6.1 Haematology and clinical chemistry laboratory variables over time (Safety analysis set)

			Result						
Lab Parameter	Treatment	Timepoint	n	Mean	SD	Min	Median	Max	Q1	Q3
B-Hemoglobin (g/L)	Drug 1 (N=8)	Baseline	8	114.0	14.25	92	113.5	136	104.0	124.5
		Cycle 02	6	123.2	14.44	104	124.5	141	110.0	135.0
		Cycle 03	8	124.9	16.95	106	125.0	150	107.5	139.0

Figure 3: First variation of lab table, three individual columns.

Table 14.3.6.12 Haematology and clinical chemistry laboratory variables over time (Safety analysis set)

			Result						
Parameter	Treatment Timepoint	n	Mean	SD	Min	Median	Max	Q1	Q3
B-Hemoglobin (g/L)	Drug 1 (N=8) Baseline	8	114.0	14.25	92	113.5	136	104.0	124.5
	Cycle 02	6	123.2	14.44	104	124.5	141	110.0	135.0
	Cycle 03	8	124.9	16.95	106	125.0	150	107.5	139.0

Figure 4: Second variation of lab table, two columns with Treatment and Timepoint information combined.

Table 14.3.6.11 Haematology and clinical chemistry laboratory variables over time (Safety analysis set)

B-Hemoglobin (g/L)

	Result						
Treatment Timepoint	n	Mean	SD	Min	Median	Max	Q1	Q3
Drug 1 (N=8) Baseline	8	114.0	14.25	92	113.5	136	104.0	124.5
Cycle 02	6	123.2	14.44	104	124.5	141	110.0	135.0
Cycle 03	8	124.9	16.95	106	125.0	150	107.5	139.0

Figure 5: Third variation of lab table with Treatment and Timepoint combined column only, lab parameter now by value in header.

TFL CODE

Ultimately the end goal of the process is to generate some TFL code which can be executed to produce our outputs. Achieving an output which looked the part with accurate numbers was the number one priority but how the TFL code looked and was constructed was also extremely important.

Code that was open and transparent and met programming best practice was really the starting point. Code that relies on a chain of black box macros becomes quickly detached from users and almost impossible to update or debug without developer intervention. The idea is that the code is completely untouched once generated but the option to tweak for an impending deadline would be possible too. To reach a standard of full automation, the TFL code must perform the below by default. These could be handled differently depending on the programmer so having a level of consistency over this helps cover the full study and for tackling future updates.

- **No Data Reporting** – often when doing subgroup analysis, more likely a certain group might contain no data
- **Zero filling** – zero counts required for treatment columns where no data exists. A few different scenarios where this can happen.
- **Pagination** – must cover all types of outputs with different requirements.
- **Decimal Precision** – number of different scenarios from baseline characteristic value to varying by parameter (e.g., lab data)
- **Auto-column width and padding** – changes in number of treatment groups must reflect different column widths, also impacts required left alignment to keep centrally under header.
- **Indentation** – common to see with AE tables, nesting of SOC and PT. Also possible to see multiple levels of nesting within some disposition outputs.

The code generated is SAS code and works within the same framework as 'manual' SAS programs. As R packages mature and client requirements evolve, R code will also be automated. This will be within a very similar framework to SAS code generation as the process inputs will be identical.

```
*************************************************************************************************
*
* Program Name      : ae201.sas
* Protocol / Study  : TESTAREA\REPIT\Development\REPIT Master
* Type              : TFL Generation
* Description       : table 14.3.2.2 adverse events by system organ class and prefered term (safety se
*
* Author            : John McDade
*
* Date Created      : 10SEP2023
* Input Dataset     : ADAM.ADAE, ADAM.ADSL
* Macros Used       : REPIT_DOEVENTS CHECKLOG RSTART RSTOP
* Files Used        :
* Status:           : in progress
*************************************************************************************************
* Change History
* Changed By        :
* Reason for Change :
* Date Changed      :
*************************************************************************************************;
```

Figure 6: Automated header block.

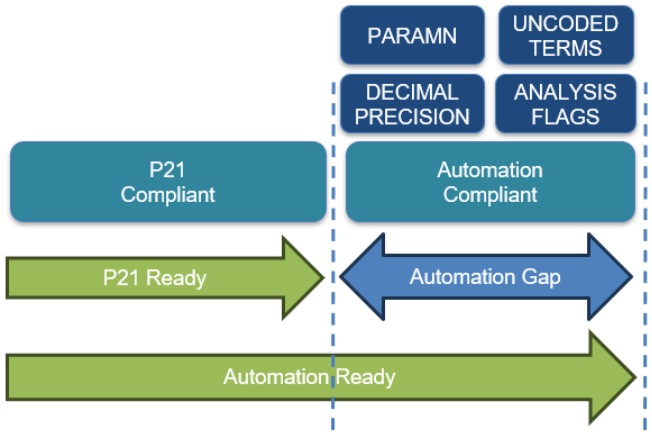
```
** Proc report to generate output **;

%rstart(pdfoptions=uniform);
ods escapechar='^';

%if &_repit_nobs > 0 %then %do;
  proc report data = tab_14_3_6_1 missing spanrows split='~'
              style(report)=[asis=on protectspecialchars=off cellspacing=0]
              style(column)=[asis=on protectspecialchars=off]
              style(header column)=[asis=off protectspecialchars=off rules=none fontsize=9pt];

    column rowblk rowid1 rowtitle1 col01 col02 col03;
    define rowblk / order order=internal noprint;
    define rowid1  / order order=internal noprint;
    define rowtitle1 / group id "Specific adverse event of interest" style(header column)=[just=l] style(column)=[just=l
    define col01 / display style(column)=[just=l asis=on cellwidth=20% pretext="      "]  flow style(header)=[asis=on];
    define col02 / display style(column)=[just=l asis=on cellwidth=20% pretext="      "]  flow style(header)=[asis=on];
    define col03 / display style(column)=[just=l asis=on cellwidth=20% pretext="      "]  flow style(header)=[asis=on];
    compute after rowblk;
      line ' ';
    endcomp;
  run;
%end;
```

Figure 7: Automated proc report code.

ADAM QUALITY

While not our primary focus at the start of the project, the positive impact of the process on ADaM was quick to see. The templates we use for standard shells contain the target ADaM metadata required and early compliance checking against the ADaM data is possible. This is moving towards automation of the ADaM specifications themselves from the TFL metadata and ADaM code where this makes sense.

As the target is output code which is untouched for true automation, this reinforces the concept of '**one-proc away**' or '**analysis ready**' to facilitate the automation. Pinnacle 21 compliance of the ADaM datasets alone is not sufficient to ensure data is ready for automation purpose – this is the **automation gap** and some form of compliance check with your data or better, ADaM define.xml will ensure ADaMs are fit for purpose earlier in the study lifecycle.



CONCLUSION

Following a similar framework to the CDISC 360 TFL Automation proof of concept, some of the benefits we have seen are:

- **TFL production time** – most outputs are ready with none or minimal customisation following template approach.
- **ADaM quality / consistency** – ADaM need to follow analysis ready concept to facilitate automated code. Early compliance checking ensures ADaM are fit for purpose.
- **Global updates** – centralised storage of treatments groups, output formats and statistical metadata all mean global or individual TFL updates can be applied as required.
- **QC Pathway** – with templates on production side, equivalent qc template programs can be used.
- **Robustness** – library of TFL metadata representing standard shells already exist meaning code effectively exists.
- **Analysis Results Metadata** – ARM was previously an entirely manual process, this path expedites ARM creation without delaying ADaM or TFL activities.

REFERENCES AND FURTHER READING

[1] Automation of TFL Generation using CDISC 360 Enriched Metadata

https://www.cdisc.org/sites/default/files/2021-08/Automation_of_TFL_Generation_using_CDISC_360_Enriched_Metadata.pdf

[2] Stuart Malcolm, Large-scale TFL Automation for regulated Pharmaceutical trials using CDISC Analysis Results Metatadata (ARM) https://www.lexjansen.com/pharmasug/2019/AD/PharmaSUG-2019-AD-203.pdf

[3] CDISC 360 White Paper: https://www.cdisc.org/sites/default/files/2021-06/CDISC_360_Project_White_Paper.pdf

ACKNOWLEDGEMENTS

We would like to thank the fantastic team working on this large-scale project. Lead developers Rajwanur Rahman on web-application and Spencer Renyard on SAS code automation. Nuala Gallagher and Scott Kosten for invaluable testing, training and support throughout the project.

CONTACT INFORMATION

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

John McDade
PHASTAR
2D Bollo Ln,
Chiswick,
London, W4 5LE,
United Kingdom

Email: john.mcdade@phastar.com

Alison Macnab
PHASTAR
2D Bollo Ln,
Chiswick,
London, W4 5LE,
United Kingdom

Email: alison.macnab@phastar.com

Web: phastar.com