Paper SM07

# An Automated Approach to Generating Call Programs from an Analysis Request

David Maher-McWilliams, MSD
Michael Allie, MSD
Carl Herremans, MSD

## ABSTRACT

In clinical research statistical programming, it is common for an organisation to have a library of standard macros. This is an efficient way to re-produce analyses across a company's projects, with each project having call programs to the standard macros. Creating these call programs can be a manually intensive and time-consuming task, requiring a programmer to interpret the request, manually create the program and populate macro parameters. This is also a possible source of error from, but not limited to, misinterpretation and typos when populating parameters. Creating a structured analysis job request form can help to alleviate misinterpretation of requests, and this machine-readable document can be leveraged to programmatically generate and populate call programs. This paper presents a SAS® macro-based system that automates the generation of call programs, where many of the macro parameters are pre-populated based on a standardised analysis request document, increasing efficiency in clinical research programming.

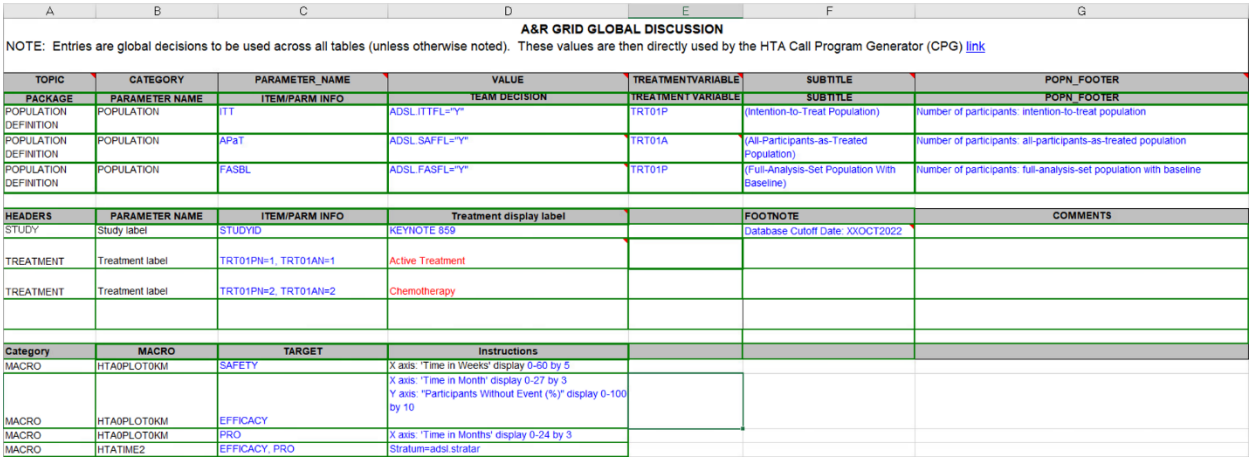
## INTRODUCTION

In clinical research within the pharmaceutical industry, data must be attributable, legible, contemporaneous, original, and accurate. In line with this data analysis must be traceable and provide reproducible results[1]. To ensure this and consistency between projects and statistical methods, maintaining a library of standard analysis and reporting (A&R) macros is very important. This also creates a consistent format of how the analysis is presented. Coupling this with standardising the format of the reports with clients allows us to implement an automated approach to the reporting process.

Clinical trial analysis and reporting is typically a collaboration between two functions, statisticians, and statistical programmers. Statisticians negotiate the analysis needs with a client (either internal or external), create a statistical analysis plan that scientifically address the business need and then create an analysis request document, which details this required analysis and in what format, tables, listings, or figures. In our company we call this analysis request document an A&R Grid. The A&R Grid differs from a project plan and a mock table package, in that it is essentially an inventory of the analysis required on a project, both on a project and output level. It creates a link between the outputs and the standardised analysis macros.

The A&R Grid guides statistical programmers as to which analysis and reporting programs to use. If a standard library of A&R macros is used, these programs can simply consist of calls to the standard A&R macros. These programs are script files that are referred to as 'call programs'. A call program defines the values for parameters of an already developed analysis macro. Creating a call program involves a lot of manual copying and pasting of values from an A&R Grid. This is time consuming and can be a source of human error.

[1] Food and Drug Administration, U.S. Department of Health and Human Services, 2017



| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| **A&R GRID GLOBAL DISCUSSION** | | | | | | |
| NOTE: Entries are global decisions to be used across all tables (unless otherwise noted). These values are then directly used by the HTA Call Program Generator (CPG) link | | | | | | |
| **TOPIC** | **CATEGORY** | **PARAMETER_NAME** | **VALUE** | **TREATMENTVARIABLE** | **SUBTITLE** | **POPN_FOOTER** |
| **PACKAGE** | **PARAMETER NAME** | **ITEM/PARM INFO** | **TEAM DECISION** | **TREATMENT VARIABLE** | **SUBTITLE** | **POPN_FOOTER** |
| POPULATION DEFINITION | POPULATION | ITT | ADSL.ITTFL="Y" | TRT01P | (Intention-to-Treat Population) | Number of participants: intention-to-treat population |
| POPULATION DEFINITION | POPULATION | APaT | ADSL.SAFFL="Y" | TRT01A | (All-Participants-as-Treated Population) | Number of participants: all-participants-as-treated population |
| POPULATION DEFINITION | POPULATION | FASBL | ADSL.FASFL="Y" | TRT01P | (Full-Analysis-Set Population With Baseline) | Number of participants: full-analysis-set population with baseline |
| | | | | | | |
| **HEADERS** | **PARAMETER NAME** | **ITEM/PARM INFO** | **Treatment display label** | | **FOOTNOTE** | **COMMENTS** |
| STUDY | Study label | STUDYID | KEYNOTE 859 | | Database Cutoff Date: XXOCT2022 | |
| TREATMENT | Treatment label | TRT01PN=1, TRT01AN=1 | Active Treatment | | | |
| TREATMENT | Treatment label | TRT01PN=2, TRT01AN=2 | Chemotherapy | | | |
| | | | | | | |
| **Category** | **MACRO** | **TARGET** | **Instructions** | | | |
| MACRO | HTA0PLOT0KM | SAFETY | X axis: 'Time in Weeks' display 0-60 by 5 | | | |
| MACRO | HTA0PLOT0KM | EFFICACY | X axis: 'Time in Month' display 0-27 by 3<br>Y axis: "Participants Without Event (%)" display 0-100 by 10 | | | |
| MACRO | HTA0PLOT0KM | PRO | X axis: 'Time in Months' display 0-24 by 3 | | | |
| MACRO | HTATIME2 | EFFICACY, PRO | Stratum=adsl.stratar | | | |

*Figure 1: Example of the Global tab from the A&R Grid.*

If reports, A&R macros, and A&R Grids are standardised, then it is possible to standardise and automate the creation of call programs. Figure 1 shows an example of the Global tab from the A&R Grid where project specific information is defined. This paper presents a system developed within the SAS programming language that automates the generation of call programs as well as the transferring of information from the A&R Grid to the call program.

## BACKGROUND

Within the MSD Health Technology Assessment (HTA) team prior to 2019 the statistical programming process for reporting HTA studies leveraged on a metadata file and call programs that called the A&R macros with a call identification number linking to the metadata file[2]. The metadata file was a list of every macro, macro parameter & parameter value and call identification number required to create the analysis for a project. If a table was requested, the metadata file would be updated to include the macro parameters for that table with an ID and a call program would be created to execute the analysis. Due to the size of the projects, this file was generally large and cumbersome. For example, for Patient Reported-Outcome (PRO) analysis where there can be a need to produce a table multiple times for different endpoints and populations, this could result in thousands of rows in the metadata file. Most of the information was repeated and unchanged for different outputs, with differences being isolated to population and observation selections, parameter codes, titles, and footnotes. Creating a metadata file for such analysis was a time consuming and manually intense process, where it was very easy to a make a mistake or overwrite previous entries.

Initially this problem was solved by creating metadata automatically through a process of using do loops and lists of required values for analysis outputting CSVs, that could then be copied into the metadata file. This worked and did save time; however, it was still quite cumbersome. Around this time, a decision was made within MSD to move away from the metadata file approach, and to fully populate macro parameters within call programs, rather than loading these from the separate file. At the same time within the department there was a drive to standardise the analysis with the internal clients. This included both standardising the format of the outputs as well as standardise analysis requests.

Although the initial problem and solution was redundant given the new A&R approach, the core of the problem still existed. Creating call programs was still a manually intensive and time-consuming task. However, with a drive to standardise in the HTA, it provides a space that would easily allow us to automate this step. Standardisation is essential to making automation possible. With this standardisation along with the initial solution came the idea to build a tool that would create call programs automatically with pre-populated parameters leveraging on the standardised A&R Grid. The tool is called the Call Program Generator (CPG).

## CALL PROGRAM GENERATOR OVERVIEW

CPG is a suite of macros that produce call programs based on a request detailed in the A&R Grid. The suite was developed in SAS. CPG can:

1. Create multiple call programs at once.
2. Archive existing programs, when creating a new program.
3. Convert special characters to rtf codes, ensuring no information is lost due to encoding issues.
4. Mask characters that SAS interprets in a special way, for example a comma.

[2] A Gillespie, M Varughese, 2007

A by-product of this suite is a process that creates a dynamic link between the A&R Grid and the call programs, meaning that if some information in the A&R Grid is updated, call programs only require a rerun rather than a manual update in the call program.

The two main sources of information the tool uses to create a call program are the A&R Grid and the reporting macros. The A&R Grid is a document populated by statisticians and programmers within the team. It details both project level information as well as output specific information, such as output titles, footnotes, macro required to create analysis, name of output created, population definitions, subpopulation definitions, and subgroup definitions. Population and subpopulation definitions define subtitles, footnotes, and population dataset subset. It is fully populated before analysis begins. The list of parameters required are derived from the macros used in the analysis.

## CALL PROGRAM GENERATOR PROCESS

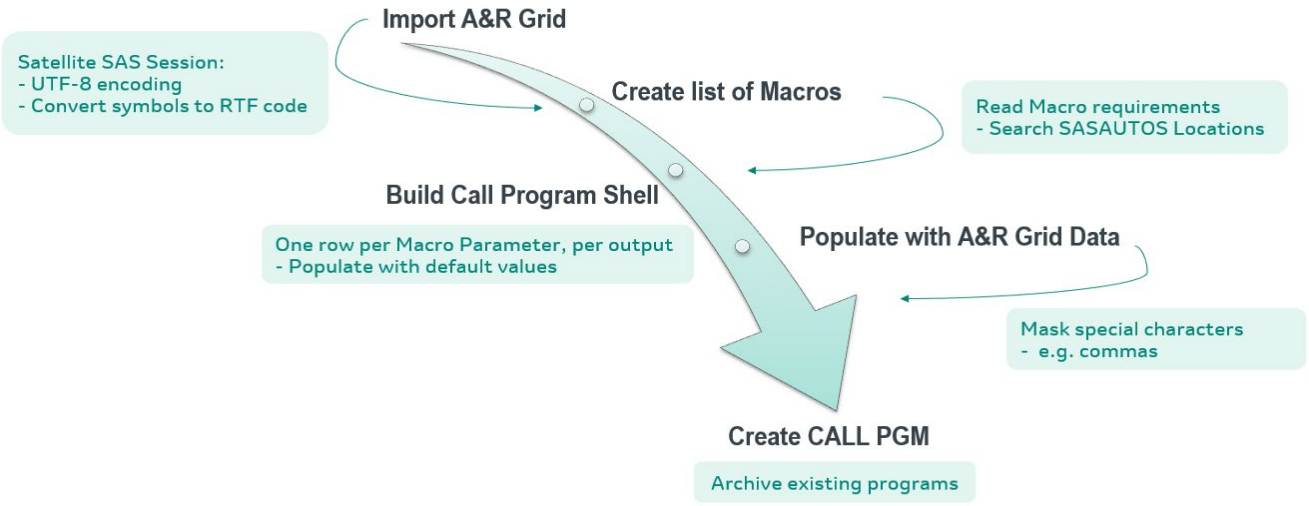



*Figure 2: Diagram of the CPG Process.*

The process of CPG is as follows. It is also illustrated in Figure 2:

- Import & convert the A&R Grid from Excel to SAS, and UTF-8 encoding to LATIN-1
- Create list of Macros & macro parameters required in call programs
- Building a SAS dataset template of call program (one row for every macro parameter and macro call)
- Populate dataset template with A&R grid data
- Final manipulation and outputs SAS call program

```
%hta0time2(population_from=
,population_where=%nrbquote(&gmvar_itt_whr)
,observation_from=
,observation_where=
,therapy_data_var=ADSL.TRT71P
,rename_output=sample_v17_0
,output_filename=fptotb
,title=Analysis of Progression-Free Survival in
months|%nrbquote(&gmvar_itt_subtitle)
,column_width=
,end_notes=a: %nrbquote(&gmvar_cutoff) |b:
%nrbquote(&gmvar_itt_footer) |c:
%nrbquote(&gmvar_endnotes_time2wald0test)
,page_orientation=p
,endpoint_label=Progression-Free Survival
,studyid=STUDYID
,time_unit=
,debug =A);
```

*Figure 3: Example of a macro call in a CPG generated call program.*

An example of a macro call in a CPG-generated call program is illustrated in Figure 3. Information which is common across multiple outputs such as patient population definitions, treatment variables, and database cut-off dates are inserted from the project-specific Global tab of the A&R Grid. More output-specific metadata such as the title, output name, and endpoint label come from the output tab. All macro parameters are present, but they are not all populated with values. Programmers are required to assign values for the macro-specific parameters based on the analysis requirements.

## HANDLING SYMBOLS WHICH CANNOT BE PROCESSED IN LATIN-1 ENCODING

One of the first challenges was dealing with symbols such as “≤” or “≥” which cannot be read using latin-1 encoding. The objective was to keep the A&R Grid as user friendly as possible so rather than prohibit the use of these symbols, the decision was made to find a way to replace these symbols with macro variables which other internal reporting macros assign to direct RTF codes. Figure 4 displays how CPG translates a symbol.

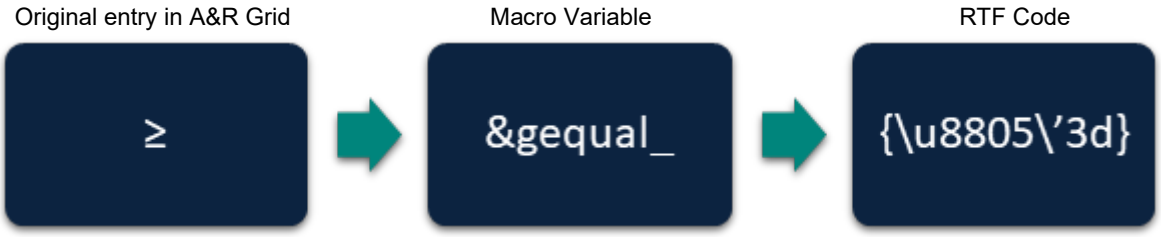



*Figure 4: Process of how CPG handles symbols.*

This was accomplished by launching an additional satellite SAS session with UTF-8 encoding (which is able to read these characters) and using the tranwrd() function to replace the symbols with the macro variable references. This satellite session is created by a supplementary macro which uses a pair of procedures: PROC OPTSAVE & PROC OPTLOAD to store information about the main session (the one where the macro is being called) and apply them to the satellite session. Additionally, all active library paths are saved so libname statements can be issued in the satellite session to make these libraries available.

```
data _null_;
    call symput('_dsnsuffix'
                ,compress(symget('sysuserid')||put(date(),date9.)||put(time(),time12.1)
                ,":. ")
                );
run;

proc optsave out=store.options&_dsnsuffix.(where =(not (OPTNAME='EVENTDS' and
OPTVALUE='(DEFAULTS)')));
run;
```

*Figure 5: Example of storing a SAS Sessions options.*

The date() and time() functions are used in addition to the automatic macro variable SYSUSERID to create a unique suffix for the output dataset containing the options from the main session. Figure 5 shows the code used to save a SAS Sessions option. As the dataset must be accessible to the satellite session, it must be saved in a permanent location during processing and deleted later. Creating a fully unique suffix allows multiple satellite sessions to run concurrently if needed, as an options dataset for each of the independent sessions can then be used.

## READING A MACRO’S PARAMETERS USING SASAUTOS

To correctly populate a call program based on the CPG query, a list of all required macros and parameters is required. Once this has been determined, the tool searches the SASAUTOS locations for the macros, creating an inventory of what parameters are required for each macro.

The getoption() function is used to retrieve the list of all locations available to the SAS session for compiling macros. Each of the items in this list is then analysed to see if it is a filepath or a fileref/libref, if the latter is true then the pathname function is used to retrieve the full directory location, providing us with a list of macro locations.

```
%* Get SASAUTOS locations;
  sasautos0loc = compbl(compress(getoption('sasautos'),'()'));
%* Loop locations and search for macro;
  do i = 1 to loc0cnt;
    %* Only search if it not found yet;
    if exist ^= 1 then do;
      %* Extract libref, resolve if needed;
     if index(scan(sasautos0loc,i,' '),'/') then do
        libref = dequote(scan(sasautos0loc,i,' '));
      end;
      else do;
        libref = pathname(scan(sasautos0loc,i,' '));
      end;
      %* Check if it exists;
      exist = fileexist(cats(libref,"/","&macroname..sas"));
      %* If so export path;
      if exist = 1 then do;
        call symputx('macrodir',libref);
      end;
    end;
  end;
```

*Figure 6: Example of code that verifies a macro exists in a SASAUTOS location.*

Figure 6 shows the code used to verify that the macro specified in the output tab exists in one of the SASAUTOS locations. The directories are searched in the same order of preference SAS uses to compile the macros. So, in the event of multiple versions of the macro across these locations, the macro analysed will be the same as the one compiled in the SAS session. Consequently, the parameters and default values written to the call program will be aligned with the intended version. Once the path is confirmed, the macro is read in via a data step and the parameters and default values are written out to a dataset.

## BUILDING A SAS DATASET TEMPLATE OF A CALL PROGRAM & POPULATING PARAMETERS

On completion of reading the parameters for all macros required for the current request, the tool then builds a shell of the required call programs. It creates a SAS dataset containing a row for each parameter in each macro required to produce all outputs. This is done by assessing the query and reading the A&R Grid to determine how many macro calls are required, what the name of the outputs will be, and which macro is required for each analysis. The dataset may have numerous rows for a single macro if that macro needs to be called numerous times to produce different analysis. The next step is to populate the macro parameters with data from the A&R Grid, this includes:

- Output location of analysis. i.e., libname of fileref location
- Titles
- Footnotes
- Population selection
- Treatment variable

The tool will then process titles and footnotes, to ensure characters that SAS will interpret in a special way are masked, for example:

- Single quotes
- Commas
- Percentages
- Semicolon
- Macro variables

Once all this information has been populated and masked, the tool then writes a SAS program. Within MSD there is a default header and format to the call programs, the tool creates a standardised compliant call program based on the MSD format.

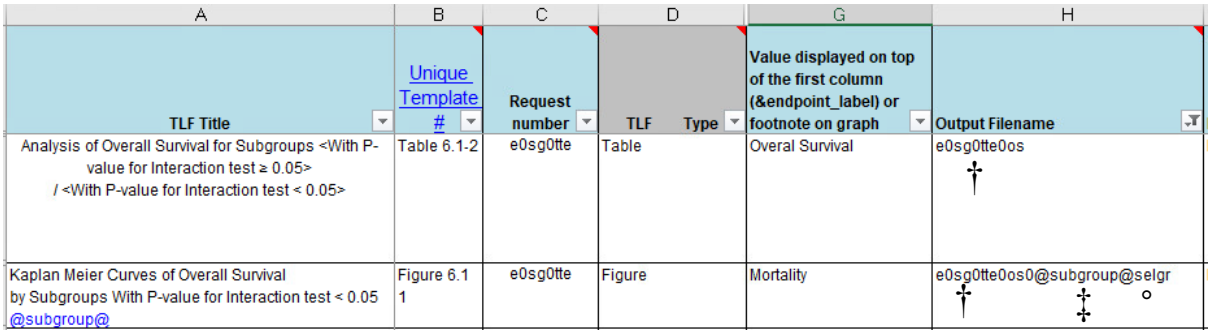
## PROBLEM SOLVING NON-STANDARD A&R GRID OUTPUT ENTRIES

A core concept of the A&R Grid is that it contains one row per output in the request. During the initial creation of CPG, all macros in the standards library created one output per call, so CPG was designed with the assumption that exactly one macro call was to be generated for each row in the A&R Grid. Since then, some macros have been developed which are capable of dynamically creating certain accompanying graphics, based on incidence and significance of data from the primary table information. So for certain requests there are now two rows in the A&R Grid for a primary table and dynamic supporting graphs that are served by a single macro call. This was outside of

the original scope of CPG, so some enhancements were needed. Ultimately, there were three challenges to overcome:

*A. Rows relating to the accompanying graphics had to be identified in the A&R Grid*

These outputs follow a standardised naming convention relating to the name of the table:



| A | B | C | D | G | H |
|---|---|---|---|---|---|
| TLF Title | Unique Template # | Request number | TLF Type | Value displayed on top of the first column (&endpoint_label) or footnote on graph | Output Filename |
| Analysis of Overall Survival for Subgroups <With P-value for Interaction test ≥ 0.05> / <With P-value for Interaction test < 0.05> | Table 6.1-2 | e0sg0tte | Table | Overal Survival | e0sg0tte0os † |
| Kaplan Meier Curves of Overall Survival by Subgroups With P-value for Interaction test < 0.05 @subgroup@ | Figure 6.1 1 | e0sg0tte | Figure | Mortality | e0sg0tte0os0@subgroup@selgr † ‡ ° |

*Figure 7: Example of standardised naming conventions for non-standard A&R Grid.*

It is always composed of Table Name†, Subgroup Name‡ and "SELGR°" appended to the end. This pattern is used to identify the rows related to these specific graphs. Figure 7 shows the rows identified by the pattern.

*B. The accompanying graphics needed to be excluded from the main call generation step*

When these rows are encountered, the output names (which are unique within the A&R Grid) are concatenated together into a macro variable which is used to exclude them from being written into the final call program

*C. Titles and footnotes for the accompanying graphics had to be included as additional parameters to the macro call which creates the primary table*

Inserting relevant information from the A&R Grid into parameter values in the shell call program is accomplished through a data step. A select statement is applied on the parameter name variable to control what value is written in for each parameter. Some macros produce additional graphics conditionally. Although this analysis requires only one macro call, it does require multiple entries in the A&R Grid for the multiple outputs. The macros have specific parameters for these graphical outputs, for example *title_graph* and *footnote_graph*. A unique hash object is dynamically created based on these specific parameters and output names to find related entries in the A&R Grid[3] (illustrated in Figure 8). The name of the TLF created from the macro call is used to build the hash object. This means that a different hash object is created and deleted for each relevant observation in the input dataset.

[3] Michael Allie, 2018

```
dcl hash sel(dataset:catt("__master5 (keep = reqnum outpfile titles collheader"
                          ," rename = (outpfile=sel_outpfile titles=g_titles
                                       collheader=g_footnotes)"
                         ," where = (sel_outpfile^=",quote(strip(outpfile)),
                           " and index(sel_outpfile,",quote(strip(outpfile)),") > 0"
                         ,"and substr(sel_outpfile,length(sel_outpfile)-4)='selgr'))"
                         )
                    );
      sel.defineKey("reqnum");
      sel.defineData(all:"yes");
    sel.defineDone();

    if parameter = 'title_graph' then do;
        order=13;
        rc = sel.find();
        if rc=0 then do;
            value=g_titles;
            exclusion_list = catx('|',exclusion_list,sel_outpfile);
        end;
    end;
    else do;
        order=14;
        rc = sel.find();
        if rc=0 then do;
            value=g_footnotes;
            exclusion_list = catx('|',exclusion_list,sel_outpfile);
        end;
    end;
```

*Figure 8: Hash object used to check for related entries in A&R Grid.*

The values of OUTPFILE (output file name) within the data step dynamically create a where clause, which is applied to a hash object to load the titles and footnotes for a table's corresponding graph.

| OUTPFILE | PARAMETER | VALUE |
|---|---|---|
| e0sg0tte0os | title_graph | Kaplan Meier Curves of Overall Survival\|by Subgroups With P-value for Interaction test < 0.05\|@subgroup@ |
| e0sg0tte0os | footnote_graph | Mortality |
| e0sg0tte0pfs | title_graph | Kaplan Meier Curves of Progression-Free Survival Based on BICR Assessment per RECIST 1.1\|by Subgroups With P-value for Interaction test < 0.05\|@subgroup@ |
| e0sg0tte0pfs | footnote_graph | Progression-Free Survival |

| OUTPFILE | TITLES | COL1HEADER |
|---|---|---|
| e0sg0tte0pfs0@subgroup@selgr | Kaplan Meier Curves of Progression-Free Survival Based on BICR Assessment per RECIST 1.1\|by Subgroups With P-value for Interaction test < 0.05\|@subgroup@ | Progression-Free Survival |
| e0sg0tte0os0@subgroup@selgr | Kaplan Meier Curves of Overall Survival\|by Subgroups With P-value for Interaction test < 0.05\|@subgroup@ | Mortality |

*Figure 9: Example of A&R Grid entries the hash object will find and utilise.*

This logic neatly integrates with the pre-existing code without the need for additional steps or modifying the existing data step into a match-merge.

## FINAL MANIPULATION AND OUTPUTS SAS CALL PROGRAM

One of the steps in the process is to mask special characters that SAS will interpret as syntax, e.g., commas or percentage sign. These are wrapped in a %str() or %nrbquote() to avoid potential issues when the call program is executed. After this the call program is written to a .sas file, archiving any existing program if needed.

## IMPACT & RESULT

Although it is hard to quantitively measure the improvement to the A&R process, qualitatively through colleague feedback, the following has been seen. An increased quality in the outputs produced meaning fewer updates are required to the outputs. CPG has addressed a key business need in delivering the correct outputs the first time they are delivered. It has allowed the programming team to spend less time on the manually intensive process of creating call programs, and more on the maintenance and development of the more complex standard macros within the standards library.

## DISCUSSION

CPG has automated the creation of call programs and reduced a significant amount of manual work within the A&R process; however, it doesn't create fully executable programs. Each call program that is created still needs input from the statistical programmer. This input includes populating macro specific parameters, such as column widths, which are unique for each table. The approach could be taken to populate these all within the A&R Grid, however this would then be similarly aligned with the metadata approach and it would also result in the A&R Grid becoming a large and cumbersome document. The team is currently exploring a way to automate the population of these parameters for future releases. Based on our previous analyses it should be possible to implement a best guess approach by analysing the parameter values of past macro calls. If for example, the width of the first column is often increased from the default value for a particular endpoint, this can be used to modify the value that is initially inserted into the generated macro call.

## CONCLUSION

Standardising and avoiding manual interventions in analysis and reporting within the pharmaceutical industry is important in creating analysis that is attributable, legible, contemporaneous, original, and accurate. Furthering this to standardising the input and output of the analysis allows us to automate the A&R approach. The development of CPG was made possible from standardising data entry at the earliest point in the A&R process. Combined with an extensive library of macros, CPG has enabled a more streamlined and automated approach to creating outputs within SAS. It has removed numerous manual tasks from the daily work and helped us ensure that the team creates high quality deliverables.

## TRADEMARK



## REFERENCES

1. Food and Drug Administration, U.S. Department of Health and Human Services, 2017 "Use of Electronic Records and Electronic Signatures in Clinical Investigations Under 21 CFR Part 11 – Questions and Answers Guidance for Industry" https://www.fda.gov/media/105557/download

2. A Gillespie, M Varughese, Developing Reporting & Analysis Methodologies Pharmasug, 2007 Paper TT14 https://www.lexjansen.com/pharmasug/2007/tt/TT14.pdf

3. Michael Allie, The Data Step Crystal Ball: Observing Future Rows, PhUSE EU Connect, 2018 Paper CT04 https://www.lexjansen.com/phuse/2018/ct/CT04.pdf

## ACKNOWLEDGMENTS

The authors would like to acknowledge the following individuals who made valuable contributions to the work described in this paper Harris Kampouris and Frederic Coppin.

## CONTACT INFORMATION

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

David Maher-McWilliams
Associate Principal Scientist
MSD
2 Pancras Square, London N1C 4AG, United Kingdom
Tel: +44 20 8154 8238
Email: david.maher-mcwilliams@msd.com

Michael Allie
Senior Scientist
MSD

The Circle 66 8058, Switzerland
Tel: +41 76 3994767
Email: michael.allie@msd.com

Carl Herremans
Principal Scientist
MSD
Clos du Lynx 5, B-1200 Brussels, Belgium
Tel: +32 2 776 6309
Email: carl_herremans@merck.com