

Review programs used for submission and their input to the define-XML 2.0

Ari Knoph, Bo S. Andersen, Morten H. Jensen, Novo Nordisk A/S

ABSTRACT

In recent years the regulatory authorities have been requiring programs for generation of ADaM data and programs for main endpoint analysis as part of the submission package. Generally in the pharmaceutical industry, analysis programs used for clinical trial reports make use of internal sponsor-defined metadata repositories and highly complex macros to facilitate e.g. the generation of several outputs. Therefore, it has become gradually clear, that the authorities demands high readability, more transparency and more traceability in analysis programs for submission as well as in the related analysis results metadata (ARM).

This paper presents a novel approach where 1) main analysis programs are created using the metadata and programming tools available in order to have a faster deliverable for internal stakeholder interaction and the clinical trial report, and 2) where review programs are made submission-ready by reducing the use of metadata, enhancing readability and keeping the programs macro-free. In order to keep an end-to-end traceability from the analysis program to the analysis results metadata included in the define-XML, in-line tags are utilized as well as an ARM library.

INTRODUCTION

With the regulatory adoption of CDISC end-to-end standards the scope of submission deliverables has been re-defined by the concept of 'e-data' e.g. ADaM datasets, programs, analysis data reviewer's guide and the define-XML. In the Technical Conformance Guides¹ and from regulatory feedback, it is stressed that an important component of the regulatory review is to understand the relationship and thereby trace between the analysis results and the analysis datasets as well as the rationale for the analysis. This is facilitated by the submission of 'transparent' analysis programs and the use of the 'Analysis Results Metadata v1.0 for Define-XML v2' specification for define-XML 2.0.

During the trial reporting phase, the generation of analysis results is often supported by complex macros that can create multiple analysis outputs in the same program. In such cases, regulatory authorities might allow that the macros are submitted, however those programs do not facilitate above-mentioned traceability necessary for the reviewer.

In the following sections, the paper describes a proof-of-concept process on how to prepare analysis programs for submission as well as a structure for the programs and a method to enforce the end-to-end traceability from analysis program to the define-XML 2.0 and the analysis results.

THE SETUP

It should be noted that the paper assumes a structured output programming process where a programmer creates the program and a reviewer reviews the program according to a specified level of review. If this is not the case of the reader, the following could be read as a guidance on how to structure and setup submission programs as well as implement and enforce the end-to-end traceability requested by regulatory authorities. The paper also assumes that the reader has knowledge of CDISC standards such as ADaM, the define-XML 2.0 and Analysis Results Metadata (ARM).

THE PROCESS FOR SUBMISSION PROGRAMS

Before starting the programming and review of primary and secondary analyses, it is to be decided which analyses should be submitted, with the intent to use the review program as a submission program. We

¹ FDA and PMDA

will call this level of review ‘full parallel programming for submission’ or simply ‘FPPSUB’. Looking into the technical conformance guides from FDA and PMDA, FDA specifies that:

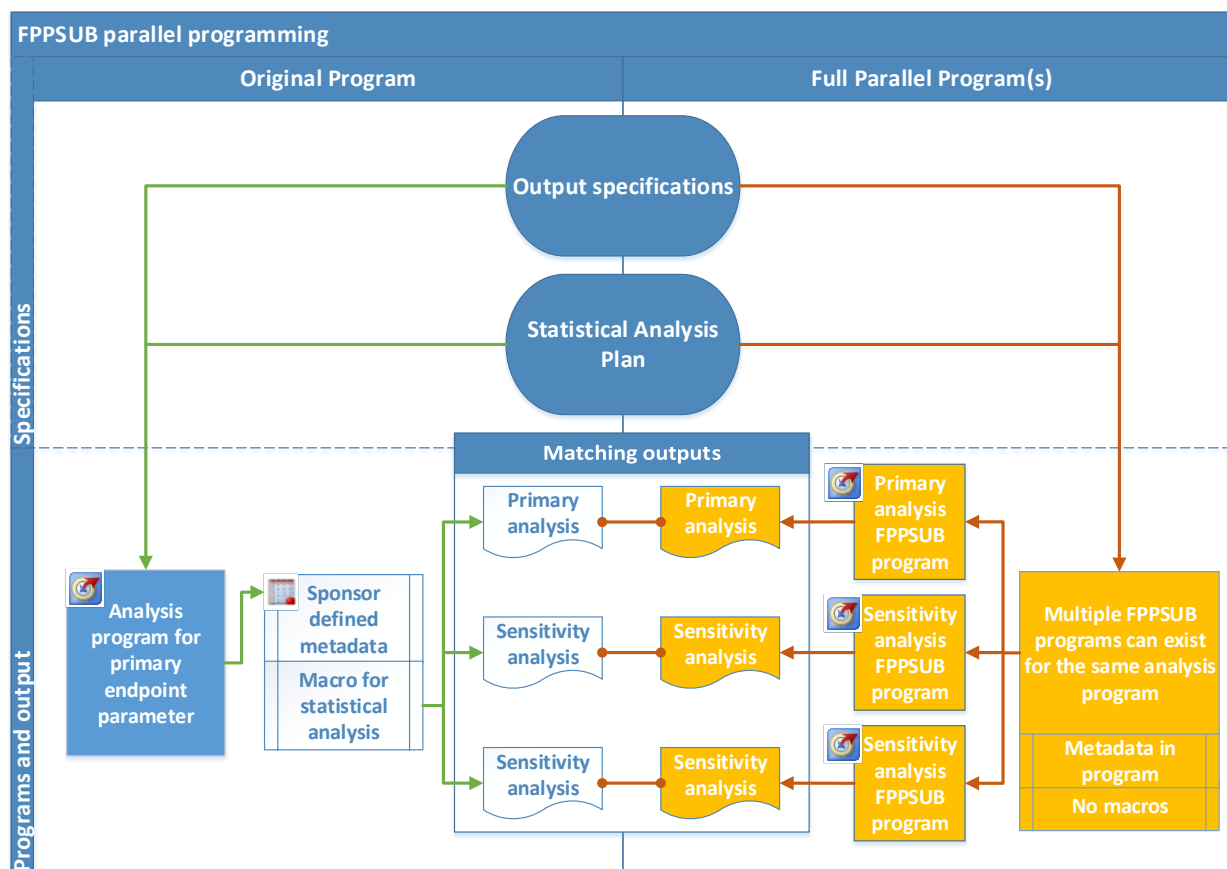
‘Sponsors should provide the software programs used to create all ADaM datasets and generate tables and figures associated with primary and secondary efficacy analyses.’²

While PMDA specifies that:

‘...the programs used to create the ADaM datasets and programs used for analyses must be submitted.’³

Depending on the trial and/or the requirements of a potential submission package, a good approach would be to choose the analyses related to the main outcome of the trial i.e. the primary analyses as well as the related sensitivity analyses. If a hierarchical testing procedure, or a similar procedure to adjust for Type I errors, is used, the analyses included in the hierarchy should also be reviewed with FPPSUB as the level of review. The analyses chosen here will most likely also be the ones where analysis results metadata are submitted for.

Once the scope of the submission programs has been decided, the remaining prerequisites for starting the programming, are the output specifications and the statistical analysis plan. The main analysis program will produce the outputs included in the clinical trial report using the sponsor-defined metadata available such as titles, labels, footnotes, font size etc. as well as metadata driven tools for output generation, and the statistical macros available to create such outputs, as displayed in the left-hand side of Figure 1. The generation of these outputs by the original program will generally be faster than the outputs generated by the review programs and therefore earlier available for internal/external stakeholder exchange.



² FDA, "Study Data Technical Conformance Guide", p. 15, section 4.1.2.10.

³ PMDA, "Technical Conformance Guide on Electronic Study Data Submissions", p. 14, section 4.1.6.1.

Figure 1 - The FPPSUB process. Left-hand side displays the process of programming one main program which creates several output using sponsor-defined metadata and macros. Right-hand side displays the review programming process where one program is created for each analysis and doesn't contain macros or retrieve metadata.

The review program will reproduce the same analysis output but be self-contained in the way that, it will not retrieve any metadata during the generation of output nor will it make use of any statistical macros. Any metadata e.g. title, labels etc. will be available in the program itself. The structure of the program will be described in further details in the next section.


When this parallel approach is chosen, one review program is created for each analysis output as opposed to the main program that might create several as seen in Figure 1. In this way it should be easier to locate and confirm the analysis algorithm as desired.

THE SUBMISSION PROGRAM STRUCTURE

The purpose of defining a general structure for the submission programs is to facilitate the regulatory review of not only one program but also across multiple programs. The aim is, that the reviewer, once familiar with the structure, will be able to locate and understand the analysis algorithm across multiple programs. Besides the structure, focus should also be given to the following:

- Adequate header information
- Well written code comments
- Consistent naming convention of datasets and variables that are self-explanatory (to the extent possible) e.g. not 'ds1' but 'adae_fas' as below:

```
data ds1;
    set adam.adae;
    where FASFL = "Y";
run;
```



```
data adae_fas;
    set adam.adae;
    where FASFL = "Y";
run;
```

As we will see later on, the information specified in the header and the code in the program will be utilised in an automatic retrieval of input for the analysis results metadata in the define-XML. This is further explained in the Enforcing traceability from define-XML 2.0 section.

THE HEADER INFORMATION

The header included in the submission program will be more extensive than the header in the main program. The purpose of the submission header is to introduce the program and the statistical analysis to the regulatory reviewer. The introduction should, much like the analysis results metadata in the define.xml, give an overview of the parameter(s) and endpoint(s) handled in the program and the statistical method(s) applied. Additionally, the header should aid the regulatory reviewer in reading the program by giving an overview of the ADaM datasets and variables used, as well as any variables derived in the program as seen in Header Sample 1.

The rationale behind including a table of variables used and derived, is also to aid the understanding of the analysis block of the program (further described below) when looking at the code isolated. In this way the regulatory reviewer doesn't have to follow the entire logic of the preceding code in order to see what a derived variable expresses.

When the statistical analysis plan has been created for the trial, it is recommended to include and align any text written about the specific analysis, in the program header. This will enforce the traceability from the statistical analysis plan to the submission program and the actual analysis.

Note that creating the submission program header will be an iterative process as it assumes knowledge of the variables used and created in the program.

```
/******  
Description: Submission program for "Adverse events - treatment  
            emergent - statistical analysis - full analysis set"  
            Produces output <output_name>.
```

```
Input      : adam.adae
```

```
Parameters : AEDECOD
```

```
Endpoints  : Number of treatment emergent adverse events
```

```
Trial      : XXXX-YYYY
```

```
Statistical Analysis :
```

```
Number of adverse events is analysed using a negative  
binomial regression model with a log-link function and the  
logarithm of the time period in which an adverse event  
is considered treatment emergent as offset.
```

```
Usage notes:
```

```
VARIABLES USED IN THIS PROGRAM
```

```
FASFL.....: Full Analysis Set Population Flag
```

```
TRTEMFL.....: Treatment emergent flag
```

```
TRTA.....: Actual treatment
```

```
TRTAN.....: Actual Treatment (N)
```

```
AEDECOD.....: Dictionary-Derived term
```

```
TRTDURY.....: Total Treatment Duration (Yrs)
```

```
NEW VARIABLES CREATED IN THIS PROGRAM:
```

```
AE_COUNT.....: Number of adverse events per dictionary-derived  
                term and per treatment
```

```
LOG_TRTDURY...: Logarithm of total treatment duration in years
```

```
EXPESTIMATE...: Exponential of estimate
```

```
LOWEREXP.....: Exponential of lower confidence limit of estimate
```

```
UPPEREXP.....: Exponential of upper confidence limit of estimate
```

```
CI.....: Confidence interval of estimate
```

```
probz.....: 2-sided p-value
```

Header Sample 1 – An excerpt of the submission program header. Additional information regarding the statistical analysis, variables used and variables derived is included in the header to aid the regulatory reviewer.

THE STRUCTURE

For the structure of the submission program, five blocks besides the header are defined. These blocks will appear with their own header in the program to clearly distinguish between sections of the program. Code examples will be included below to display block headers and samples of the programming statements in that block. Please note that any examples included below is only to display the elements of each block and not necessarily a recommendation for the look of block headers or code.

The first one will be the 'DATA SELECTION' block, where a selection of data is made from one or more main ADaM datasets with the appropriate 'where clause' applied:

```
/******/  
/** DATA SELECTION **/  
/******/
```

```

data adae_fas;
  set adam.adae;
  where FASFL = "Y" and TRTEMFL = "Y";
run;

```

Code Sample 1

The 'where clause' reference used in this DATA step should match the one displayed in the analysis results metadata for this analysis. When doing the data selection, one should consider using the KEEP statement to keep only the variables relevant to the analysis and variables that aids the understanding of the selected data.

The second block is the 'DATA PROCESSING' block. This is where additional variables or analysis datasets are derived:

```

/*****/
/** DATA PROCESSING **/
/*****/

proc sql;
  create table adae_fas_count as
  select TRTA, TRTAN
         , AEDECOD
         , count(AEDECOD)           as AE_COUNT
  from adae_fas
  group by TRTA, AEDECOD;
quit;

```

Code Sample 2

Depending on the derivation of the variables included, a programmer would traditionally do the derivation of the variables, if applicable, in the DATA step in the previous block, or do the data selection in the sample SQL procedure above, as this would programmatically be most time- and space-saving. However, more complex derivations or data selections would be harder to locate when browsing through the program, and therefore, this split between selection and derivation is recommended.

Block number three is the 'DATA ANALYSIS' block. This is where the analysis algorithm will be located for the reviewer to look at:

```

/*****/
/** DATA ANALYSIS **/
/*****/

proc genmod data=adae_fas_count;
  class TRTAN;
  model AE_COUNT = TRTAN / link=log dist=negbin
                        offset=LOG_TRTDURY alpha=0.05;

  lsmestimate TRTAN
    '<Treatment 1>' [1, 1],
    '<Treatment 2>' [1, 2],
    '<Treatment 1 / Treatment 2>' [1, 1] [-1, 2]
    / cl exp alpha=0.05 om;
  ods output LSMestimates = estimates;
run;

```

Code Sample 3

In this block it is vital for the understanding of the model, that dataset and variable names are kept clear and self-explanatory to the extent possible. Depending on the derivation in the 'DATA PROCESSING' block, it is worth considering whether any additional variables derived there, actually should be included in the main ADaM dataset if possible. This would reduce the complexity of the submission program and

header. Also any variable included directly in the 'DATA ANALYSIS' block from the ADaM dataset could be identified in the define.xml as well.

Fourth block is the 'DATA COLLECTION AND ALIGNMENT' block. This is where results created in the 'DATA ANALYSIS' block is collected into a common structure and the variables arranged. Any arrangement to fit the layout specification is also done here, such as applying formats and collapsing variables:

```

/*****/
/** DATA COLLECTION AND ALIGNMENT **/
/*****/

    data final_stats;
        set estimates;

    /*Apply formats and create a common variable (CI) to hold*/
    /*the confidence interval.*/
        CI    = "[ " ||    put(LowerExp,5.2) || "; " ||
                put(UpperExp,5.2) || "]" ;

run;

```

Code Sample 4

The fifth and last block is the 'DATA OUTPUT' block. This is where any output specific metadata that would be retrieved by the main program is setup e.g. output path, output title and footnotes. Please note that the following code sample contains pseudo-code(<pseudo-code>) to ease the readability:

```

/*****/
/** DATA OUTPUT **/
/*****/

options orientation=portrait ls=96 ps=89;
ods listing file= "/example_drive/trial/output/output_name.txt";
title "Adverse events - treatment emergent - statistical
      analysis - full analysis set";

proc report data=final_stats nowd split='|' spacing=0;
    column (string Expeestimate CI probz);
    define string          / display ""           <options>;
    define Expeestimate    / display "Estimate"    <options>;
    define CI              / display "95% CI"      <options>;
    define probz           / display "p-value*"    <options>;

    compute before;
        <output specific layout>;
    endcomp;

    compute after;
        line @1 "";
        line @1 "p-value is from the 2-sided test for
                treatment difference evaluated at the 5% level.";
    endcomp;

run;

ods listing;

```

Code Sample 5

When creating the output from the submission program, the output specifications for the clinical study report output from the main program should be followed. As the submission program is created as part of

the formal internal sponsor review, it is recommended that any output created are compared through a text compare or a utility to compare figures.

CODE COMMENTS

When writing comments to the code in the submission program, the author should have the regulatory reviewer in mind. Any comment given should be there to aid the understanding of the proceeding code. The comments should preferably be written in plain text, without sponsor-specific lingo and abbreviations and be coherent as the reviewer might not be a programmer.

The suggested guidelines for writing code comments are:

- Describe data extractions in plain text and what that data should be used for.
- Describe variable derivations in plain text and what their purposes are. A suggestion here is to include the derived variable name in the comment to link the plain text explanation with the variable.
- If any derivation is done that directly links to the output, include in the comment that e.g. 'the values in the "Estimate" column in the output is derived here'.
- Depending on the complexity of the statistical method used, a minimal explanation of the method should be included in the program with the role of the variables in the model explained if possible. If the method is very complex a reference could be given to the statistical analysis plan and/or the analysis results metadata.

Looking at Code Sample 2 again a comment for the derivation of 'AE_COUNT' could be:

```
/******  
/** DATA PROCESSING **/  
/******  
  
/*In the following we create the counts of adverse events (AE_COUNT)*/  
/*grouped by the actual treatment (TRTA) and the dictionary-derived*/  
/*term (AEDECOD).*/
```

Code Sample 6

Looking at Code Sample 3 again a comment for the statistical method could be:

```
/******  
/** DATA ANALYSIS **/  
/******  
  
/*The data is analysed using a negative binomial regression model*/  
/*with the adverse event count (AE_COUNT) as the response and the*/  
/*actual treatment as the effect (TRTAN). The logarithm of the */  
/*treatment duration in years (LOG_TRTDURY) is used as offset.*/  
/*"Estimate", "95% CI" and "p-value" displayed in the*/  
/*output is calculated here.*/
```

Code Sample 7

ENFORCING TRACEABILITY FROM DEFINE-XML 2.0

With the ARM extension to the define.xml 2.0, sponsors are now able to add and link additional information for statistical analyses to further add traceability from analysis dataset to analysis result. This includes:

- The Analysis Parameter(s) and Variable(s)
- Analysis Reason
- Data References (incl. Selection Criteria)

- Documentation
- Programming Statements

In the following, an approach is shown, where the analysis results metadata are partly kept in an ARM repository and partly auto-populated from the submission program facilitated by the header and structure defined above. In this case, an Excel sheet forms the basis of the ARM repository and the later SAS dataset to be used as input for the generation of the define.xml through the Clinical Standards Toolkit, as seen in Figure 2 below. The sections of the ARM auto-populated are the two latter in the bullet list above: 1) the Documentation section and 2) the Programming statements section.

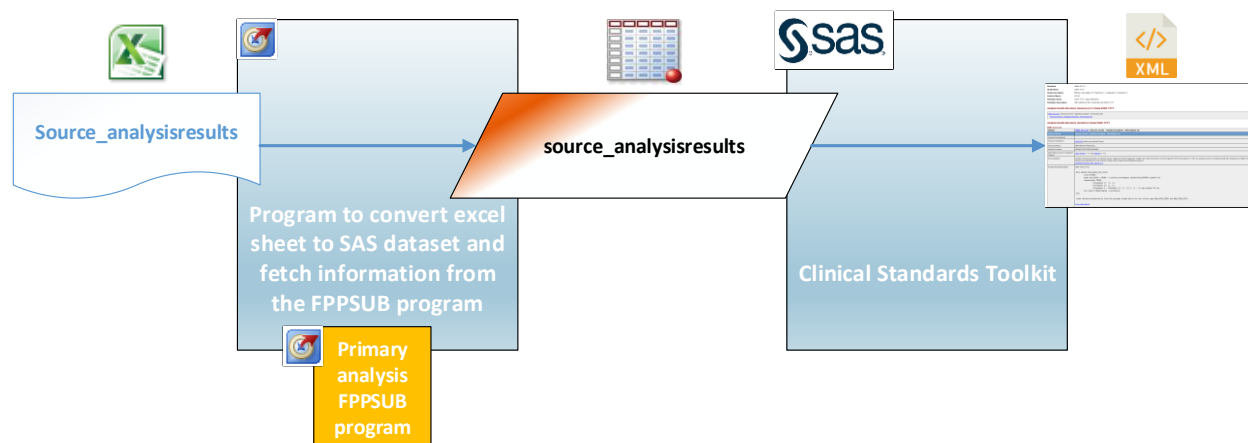


Figure 2 – The ‘source_analysisresults’ SAS dataset is created via input from the excel ARM repository and the information automatically retrieved from the submission program. SAS’ Clinical standards toolkit creates the define.xml.

The ARM parameters needed to generate the define.xml are defined in the Excel sheet through written text, references to controlled terminology or through a keyword (shown in red boxes), as seen in Table 1 below. The auto-population is activated through the keywords once the step to create the SAS dataset is initiated.

Title/Label	Analysis Dataset Define	Analysis Variables Define	Where Clause Define
Adverse events - treatment emergent - statistical analysis full analysis set	ADAE	AEDECOD	FASFL='Y' and TRTEMFL='Y'
Title/Label	Analysis Reason	Analysis Purpose	
Adverse events - treatment emergent - statistical analysis full analysis set	C117752	C98772	
Title/Label	Documentation	Document Ref	PDF Page Ref
Adverse events - treatment emergent - statistical analysis full analysis set	Statistical analysis plan section 2.3	xxxx-yyyy-statistical-analysis-plan.pdf	10
Title/Label	Description	Code	
Adverse events - treatment emergent - statistical analysis full analysis set	FETCH_DESCRIPTION_FROM_PARPROG	FETCH_CODE_FROM_PARPROG	

Table 1 – Excerpt of the analysis results metadata working repository. The ‘Description’ cell and the ‘Code’ cell in the red boxes uses a keyword to specify that this will be automatically fetched from the submission program when creating the input datasets for the Clinical Standards Toolkit.

In the Header Sample 1 we included information about the statistical method in the ‘Statistical Analysis:’ section of the header. When the ‘FETCH_DESCRIPTION_FROM_PARPROG’ keyword is activated in the ‘Description’ cell, a program will read this section of the header and populate it into the SAS dataset for define.xml generation. An automated comment is added to the documentation text to clarify that this section is auto-populated as seen in Figure 3.

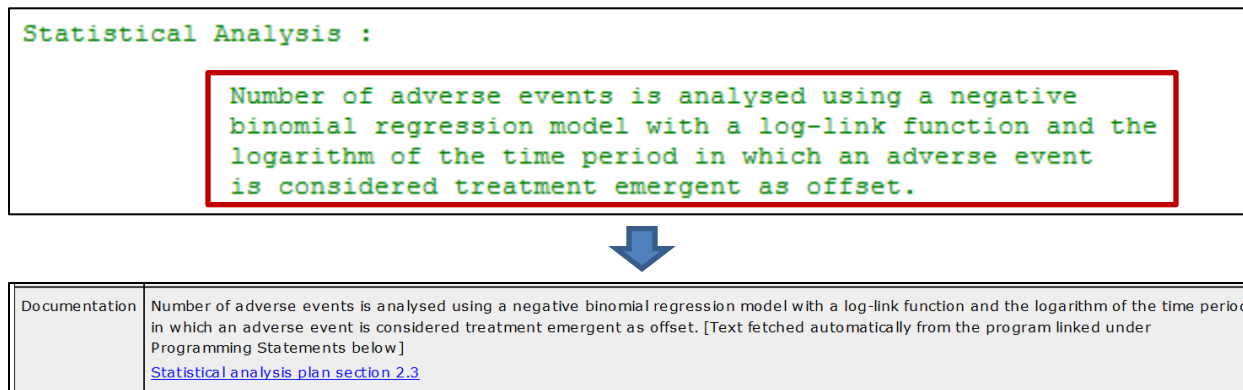


Figure 3 – The header information from the submission program populates the ‘Documentation’ section of the Analysis Results Metadata in the define.xml.

Together with the statistical documentation reference defined in Table 1 this creates the ‘Documentation’ section of the ARM.

In order to retrieve the programming statement for the define.xml we will need to look at Code Sample 3 again and add the in-line tags ‘ARM_CODE_START’, to represent the beginning of the programming statement, and ‘ARM_CODE_STOP’, to represent the end of a programming statement as seen below:

```

/*****/
/** DATA ANALYSIS **/
/*****/

/*ARM_CODE_START: Procedure is added to Analysis Results Metadata*/
proc genmod data=adae_fas_count;
  class TRTAN;
  model AE_COUNT = TRTAN / link=log dist=negbin
                        offset=LOG_TRTDURY alpha=0.05;

  lsmestimate TRTAN
    '<Treatment 1>' [1, 1],
    '<Treatment 2>' [1, 2],
    '<Treatment 1 / Treatment 2>' [1, 1] [-1, 2]
  / cl exp alpha=0.05 om;
ods output LSMestimates = estimates;

run;
/*ARM_CODE_STOP*/

```

Code Sample 8

When the ‘FETCH_CODE_FROM_PARPROG’ keyword is activated in the ‘Code’ cell in Table 1 the same program will read the submission program top down to fetch the position of the in-line tags and the code in between. The in-line tags can in principle be added multiple times if e.g. the sorting order from the ‘DATA PROCESSING’ block is desired in the programming statement. In that case the program should append the code statements in the same order as they appear in the program. The resulting programming statement can be seen in Figure 4.

Programming Statements	<pre>[SAS version 9.4] proc genmod data=adae_fas_count; class TRTAN; model AE_COUNT = TRTAN / link=log dist=negbin offset=LOG_TRTDURY alpha=0.05; lsmestimate TRTAN '<Treatment 1>' [1, 1], '<Treatment 2>' [1, 2], '<Treatment 1 / Treatment 2>' [1, 1] [-1, 2] / cl exp alpha=0.05 om; ods output LSMestimates = estimates; run; [Code fetched automatically from the program linked below via the inline tags ARM_CODE_START and ARM_CODE_STOP] a_ae_stat_fas.txt</pre>
------------------------	---

Figure 4 – The results of the automatically retrieved code from the submission program using the in-line tags ‘ARM_CODE_START’ and ‘ARM_CODE_STOP’.

As with the information retrieved from the program header, an automated comment is added, as well as a link to the submission program. With this, the traceability from the analysis program to the analysis results metadata in the define.xml is enforced by creating a direct link from program to display.

CONCLUSION

Although software programs are required in the ‘e-data’ submission package, no official regulatory guidance has been given on the content and structure of the programs other than it should be possible to locate and validate the analysis algorithm. Furthermore, the programs submitted are not yet required to be executable.

The approach shown in this paper provides a common structure for the submission program that facilitates the review of the analysis algorithm. Also it provides the sponsor with a guarantee that the analysis algorithm shown in the ARM of the define-XML, is the actual analysis algorithm used to produce the output in the CSR. This makes the approach robust against any late changes in the program. Making the programs self-contained i.e. macro- and metadata-free also gives a reviewer the possibility to execute the program(s) directly on submitted analysis datasets to re-create CSR outputs or simply re-run each block of the program.

As with all processes, there are some limitations to the setup described. Creating multiple submission programs for each main program can be time-consuming. When creating the program, the programmer should return multiple times to the header, making sure that the variables specified in the header are up-to-date with the actual program. Also, a submission program of high complexity entails a lesson in moderation, balancing, on one hand, the overview and overall readability of the program, and on the other hand, not compromising on any information that aids the reviewer in understanding the analysis algorithm.

Working with the above presented approach, one could expand the auto-population to include the analysis parameters, variables, datasets as well as the ‘where clause’ and the table join comment, making the ARM section of the define-XML almost fully automated. In a broader perspective, one could imagine that the variable hyperlinks in the define-XML could be developed to be shown in the ‘Programming Statement’ of the ARM. Utilising the defined table of variables from the expanded header in Header Sample 1, one could also create a mouse-over view on each of the variables in the ‘Programming Statement’, also for the variables derived in the program, to aid the understanding of the analysis algorithm, when looking only at the define-XML.

REFERENCES

CDISC, “Analysis Results Metadata v1.0 for Define-XML v2”. Jan 2015.

<https://www.cdisc.org/standards/foundational/analysis-data-model-adam/analysis-results-metadata-arm-v10-define-xml-v20>

FDA, “Study Data Technical Conformance Guide”. Oct. 2017.

<https://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM384744.pdf>

PMDA, "Technical Conformance Guide on Electronic Study Data Submissions". April 2015.

<https://www.pmda.go.jp/files/000206449.pdf>

DISCLAIMER

The views and opinions presented here represent those of the authors and should not be considered to represent the views of Novo Nordisk A/S.

ACKNOWLEDGEMENTS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

CONTACT INFORMATION

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

Ari Knoph

Novo Nordisk A/S

aikp@novonordisk.com

Bo S. Andersen

Novo Nordisk A/S

bsaa@novonordisk.com

Morten H. Jensen

Novo Nordisk A/S

mohj@novonordisk.com