PharmaSUG 2015 - Paper AD01

The Implementation of Display Auto-Generation with Analysis Results Metadata Driven Method

Chengxin Li, Boehringer Ingelheim Pharmaceuticals Inc., Ridgefield, CT, USA

ABSTRACT

Standard data structures by CDISC should lead to standard programs and result in auto-generation of SDTM and ADAM data sets and tables, listings and figures. This paper provides a display auto-generation solution implemented with SAS® macros. The solution is driven by analysis results metadata and illustrated with a survival analyses display. Together with display auto-generation, this paper also describes the methods handling dynamic footnoting and dynamic reporting.

INTRODUCTION

The Clinical Data Interchange Standards Consortium (CDISC) [1] has defined a series of data models, e.g., Clinical Data Acquisition Standards Harmonization (CDASH), Study Data Tabulation Model (SDTM), Analysis Data Model (ADaM), and also ADaM Data Structure for Adverse Event Analysis (ADAE) and ADaM Basic Data Structure for Time-to-Event Analyses (ADTTE).

CDASH is harmonized with SDTM. SDTM should fully reflect the collected data (e.g., mapping for any collected and a limited number of derived variables, but no imputation for any missing data). Furthermore, ADaM should only be derived from SDTM. The key endpoint analyses, inferential analyses, and complicated analyses should be designed with ADaM datasets. However, not every analysis needs to have a corresponding ADaM dataset. Some simple displays can be directly coded from SDTM domains such as data summary displays that do not need derivations for timing widows, nor imputations for missing data, nor derivations with complicated algorithms. The traceability should be facilitated wherever possible from analysis result (e.g., p value) back to ADaM, to SDTM, and further back to CDASH. The CDISC data processing models are illustrated in Figure 1.

Standard data structures by CDISC should lead to standard programs and further result in auto-generation of SDTM, ADAM and displays. From Figure 1, there are three programming sections in CDISC data models: SDTM transformation, ADaM generation, and display generation, corresponding to SDTM auto-generation, ADaM auto-generation, and display auto-generation, respectively.

The pharmaceutical and biotech industries have not fully taken advantage of the ability to auto-generate these data structures and displays. Traditionally, programming has still quite a bit of study specific information and is re-purposed to other studies in varying ways. Although SAS macros are widely used, the data selection criteria and the statistics models used for analyses are coded inside of the program as well. The data selections and models are dynamic, which means different from analysis to analysis and different from project to project, causing limitations in generic applications, e.g., not able to cross therapeutic areas.

1 http://www.cdisc.org

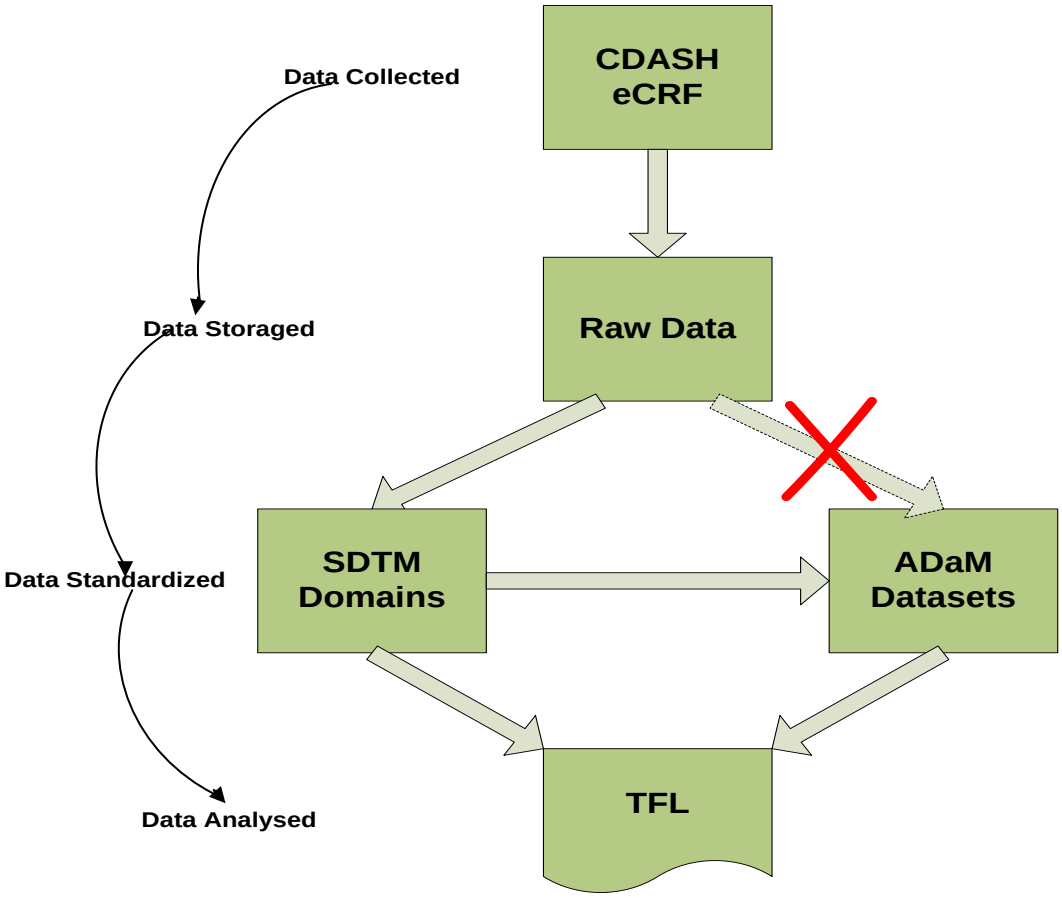

Figure 1 CDISC Data Processing Model

This paper will only specify the design and implementation for the display auto-generation. SDTM dataset auto-generation and ADaM dataset auto-generation are beyond the scope of this paper.

DATASET DESIGN PROCESS

Figure 2 illustrates the general design diagram for an ADaM dataset. The SDTM domains and an ADaM dataset such as the Subject-Level Analysis Dataset (ADSL), are the input data to further derive ADaM datasets. The ADaM standard and agency requirements (e.g., FDA Study Data Technical Conformance Guide[2]) are the documents with which the ADaM dataset should comply. Statistical Analysis Plan (SAP), Table of Contents (TOC) and Display Templates are key inputs to determine what ADaM datasets are required, and what variables are needed in an ADaM dataset.

Besides traceability, "analysis-ready" is another principle for ADaM design. The "analysis-ready" concept means to apply data selection criteria and then invoke a statistical procedure. Typically, should not be any further data processing after applying the data selection criteria on an ADaM dataset. To implement display auto-generation, it is assumed that the "analysis-ready" concept has already been applied to an ADaM dataset.

2 http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM384744.pdf

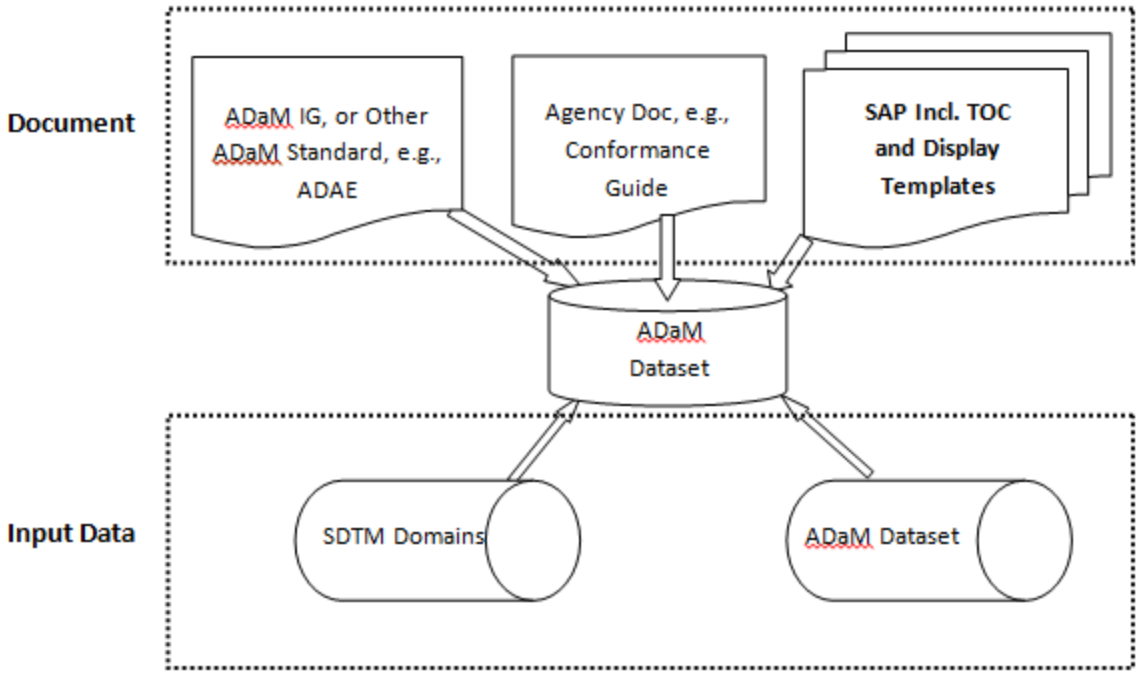

Figure 2 Design Diagram of Analysis Dataset

ANALYSIS RESULTS METADATA

CDISC Analysis Data Model (ADaM) v2.1 defines three types of metadata: Analysis Dataset Metadata, Analysis Variable Metadata, and Analysis Results Metadata.

Analysis results metadata describe the main attributes of the analysis results and provides the link between analysis results and analysis dataset to facilitate reviews. Within the sponsor company, the metadata provides the link among the biostatistician, the primary programmer, and the QC programmer.

ADaM v2.1 describes the analysis results metadata fields (p22, table 5.3.1) and provides several examples to illustrate analysis results metadata (p23-27). The implementation of display auto-generation will utilize the metadata fields of the analysis results metadata, thus the solution is named as display auto-generation driven by analysis results metadata.

STATISTICAL PROGRAMMING PROCESS

Figure 3 illustrates the general programming flow for a display. The input ADaM dataset, where conditions, analysis variables or statistical models, display template, titles, and footnotes are the parameters which should be incorporated in the programming process. As the analysis-ready feature is facilitated in an ADaM dataset, there should not have to be any data processing between subsetting data (applying data selection criteria) and invoking the statistical procedure for descriptive or inferential statistics. It is the display template-oriented for formatting statistical data and outputting the display.

Associations exist between statistical programming parameters and metadata fields of analysis results metadata. Figure 4 illustrates the mapping key fields of analysis results metadata to SAS macro parameters. Further, a BY-Group (BYGRP) variable for subgroup analyses should be added if applied.

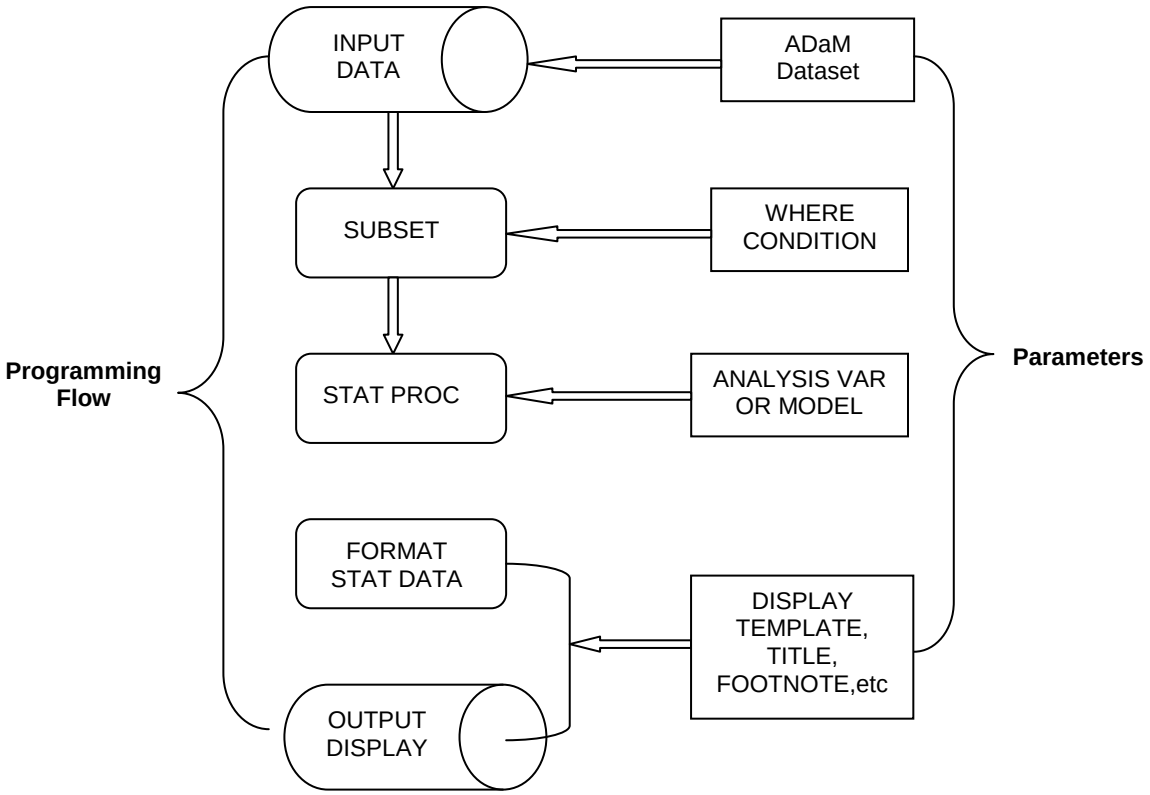

Figure 3 General Process for Statistical Programming

Analysis Results Metadata Field	Macro Parameters
DISPLAY IDENTIFIER	UNIQUE TABLE OR FIGURE NUMBER
DISPLAY NAME	DISPTAB (DISPLAY TEMPLATE NAME)
RESULT IDENTIFIER	UNIQUE ID FOR AN ANALYSIS RESULT (POSSIBLY SKIPPED)
PARAM/PARAMCD	PARAM, PARAMCD WITH ADaM BDS
ANALYSIS VARIABLE	STATVAR (e.g., AVAL, AVALC, or CHG)
DATASET	INDATA (ADaM INPUT DATASET)
SELECTION CRITERIA	WHERE (DATA SELECTION CRITERIA)
PROGRAMMING STATEMENTS	MODEL (INFERENTIAL STATISTICS MODEL WITH SAS CODE)
REASON/DOCUMENTATION	POSSIBLY SKIPPED
NOTE: BY GROUP (BYGRP), TITLE, FOOTNOTES MAY BE ADDED AS NEEDED.	

Figure 4 Mapping of Fields of Analysis Results Metadata to the Macro Parameters

The programming parameters mapped to metadata fields of analysis results metadata enable the display generation to be driven by analysis results metadata, which results in a standard program. Further, since the structure of analysis results metadata can be standardized by the metadata fields being mapped to generic parameters (e.g., with SAS generic macros) or to input factors of a clinical trial report-generating system, the display auto-generation can be implemented.

A display template for a table/figure/listing (TFL) is another key factor for the auto-generation. A display template contains the "what-and-how" of the analysis results. It also provides the target of formatting statistical data from a statistical procedure. Display auto-generation requires the display template to be stable. Display templates, especially for common analyses, can be designed generically across studies and therapeutic areas.

The following sections describe a generic SAS macro as an implementation method of display auto-generation.

MACRO DESIGN

Design Goals: display auto-generation implemented with SAS macros.

Design Principles:

- Study independent
- Structured design
- Macro parameters standardized with ADaM analysis results metadata
- Display template oriented
- Dynamic footnoting and dynamic reporting supported
- Readability (less complexity)

To decrease the complexity, multiple macros should be designed and implemented. The macros can be categorized based on ADaM classes (ADSL, BDS, and ADAE), statistical analysis types (descriptive or inferential), and display template. For instance, based on ADSL, the subject evaluations such as demographic analyses, population set analyses, and disposition analyses can be implemented with macro POPU, DEMO, and DISP, respectively; based on ADAE, one or multiple macros designed for AE analyses; based on ADTTE, macro TTEKM and TTEPH for time to event analyses. The programming model for display auto-generation should make all the dynamic factors as macro parameters. Therefore the macro can be generic. Dynamic footnotes will be supported. The programmer can put any footnotes at any display positions or adjust the footnotes according to the display results. Further, to accommodate possible display template changes, dynamic reporting will also be facilitated.

The parameters in each macro call are a subset of the analysis results metadata. This makes all the project participants (biostatistician, primary programmer, QC programmer) or even the regulatory reviewer, focus on the parameters being defined, and testing the program.

Here are simplified macro definitions and the corresponding macro calls:

```
%POPU(DISPTAB= TPOPU1#,
      INDATA = ADaM.ADSL,
      WHERE  = %str(STUDYID in ("'xxxx_0001")),
      STATVAR= %str(ENRLFL# RANDFL# SAFFL# FASFL# COMPLFL#),
      TITLE1 =%str("Table 15.1.1 The ......")
             );
%DEMO(DISPTAB= TDEMO1#,
      INDATA = ADaM.ADSL,
      WHERE  = %str(STUDYID in ("xxxx_0001")),
      STATVAR= %str(SEX# RACE# AGE AGEGR1# HEIGHT WEIGHT BMI BMIGR1#),
      TITLE1 =%str("Table 15.1.2 The ......")
     ); /* On STATVAR, '#' for frequency statistic (e.g., SEX), otherwise for
           numeric descriptive statistic (e.g., AGE) with display template */

%TTEPH(DISPTAB = TTE6#,
      INDATA = ADaM.ADTTE,
      PARAM  =%str("Time to Rebound"),
      PARAMCD=%str("TTRBD"),
      WHERE  =%str(STUDYID in ("xxxx_0001") and FASFL='Y' and PARAMCD="TTRBD"),
      MODEL  =%str(proc phreg data = ADTTE;
                     model aval*cnsr(1) = trtpn age /risklimits=pl;
                   run;),
      TITLE1 =%str("Table 15.2.2 The ......"),
```

```
FOOT1  = %str("The model includes treatment and age as covariates"));
```

In the above macros, the parameter DISPTAB is defined for display template, INDATA for input dataset, WHERE for data selection criteria, STATVAR for analysis variable(s), MODEL for statistical model. The parameters should be SAS executable codes, which can be embedded into macro. In this way, combining dynamic coding on ODS OUTPUTs, the macro can accommodate model variations such as different ALPHA, different CLASS or BY variables.

A WHERE statement can be dynamically constructed depending on the ADaM dataset and analysis. Figure 5 lists one example of a WHERE statement construction:

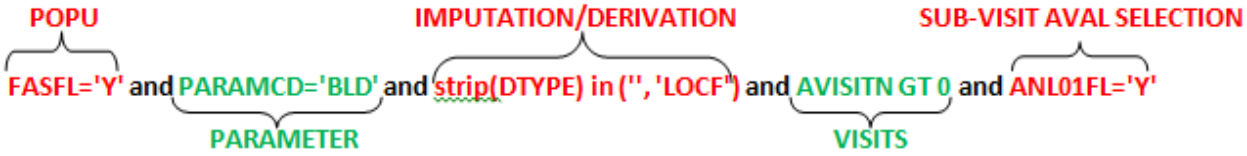

Figure 5 WHERE Statement Construction

The display auto-generation assumes an input dataset with basic ADaM compliance. The REQ (required) variables by ADaM standards (e.g., ADaM IG, ADAE, ADTTE) need not to be checked for availability in the dataset and can be directly used in the program. The display auto-generation also assumes the input dataset with 'analysis-ready', i.e., the variables should be populated in the dataset as analyses need. However, in some cases, the availability of COND, PERM, user-added variables may be pre-checked by the macro.

In next section, Time to Event (TTE) analyses will be used as an example for only the purpose of illustration of display auto-generation implementation.

IMPLEMENTATION

Survival analysis typically applies to time to event data. The non-parametric Kaplan-Meier (KM) estimator and parametric Cox Proportional Hazards (PH) model are the two most common approaches used for survival analyses. With SAS, PROC LIFETEST and PROC PHREG are used correspondingly. In CDISC, the ADaM Basic Data Structure for Time-to-Event Analyses (ADTTE) was defined for survival analysis. The ADTTE can be treated as BDS plus some TTE variables.

PROGRAMMING FLOW

Figure 6 illustrates the programming flow with a structured design. The Initialization step is for setting up the environment. The Selecting Data step applies WHERE data selection criteria.

In the Analyzing Data step, the corresponding ODS OUTPUT options are enabled according to SAS PROC in the MODEL, e.g., for PROC LIFETEST, the ODS OUTPUT ProductLimitEstimates, Quartiles, SurvDiff etc should be enabled; for PROC PHREG, the ODS OUTPUT ParameterEstimates should be enabled. To execute the model codes specified in the MODEL parameter, define and run the following sub-macro:

%macro model; &model; %mend; %model;

It should be noted that if a BY statement is provided in the MODEL, the BY variable(s) should be dynamically extracted. The BY variable(s) need to be available in the dataset, and the dataset sorted accordingly (analysis-ready) except in the case of the keyword NOTSORTED or DECENDING being further specified in the BY statement.

In the MODEL, by fault, take ALPHA=0.05. However, ALPHA may be different depending on trial design, e.g., ALPHA set as 0.1 in a phase II study. To accommodate dynamic ALPHA and thus getting the corresponding confidence interval (CI), ALPHA can be scanned from the MODEL or post-processed from label of CI in ODS OUTPUT dataset.

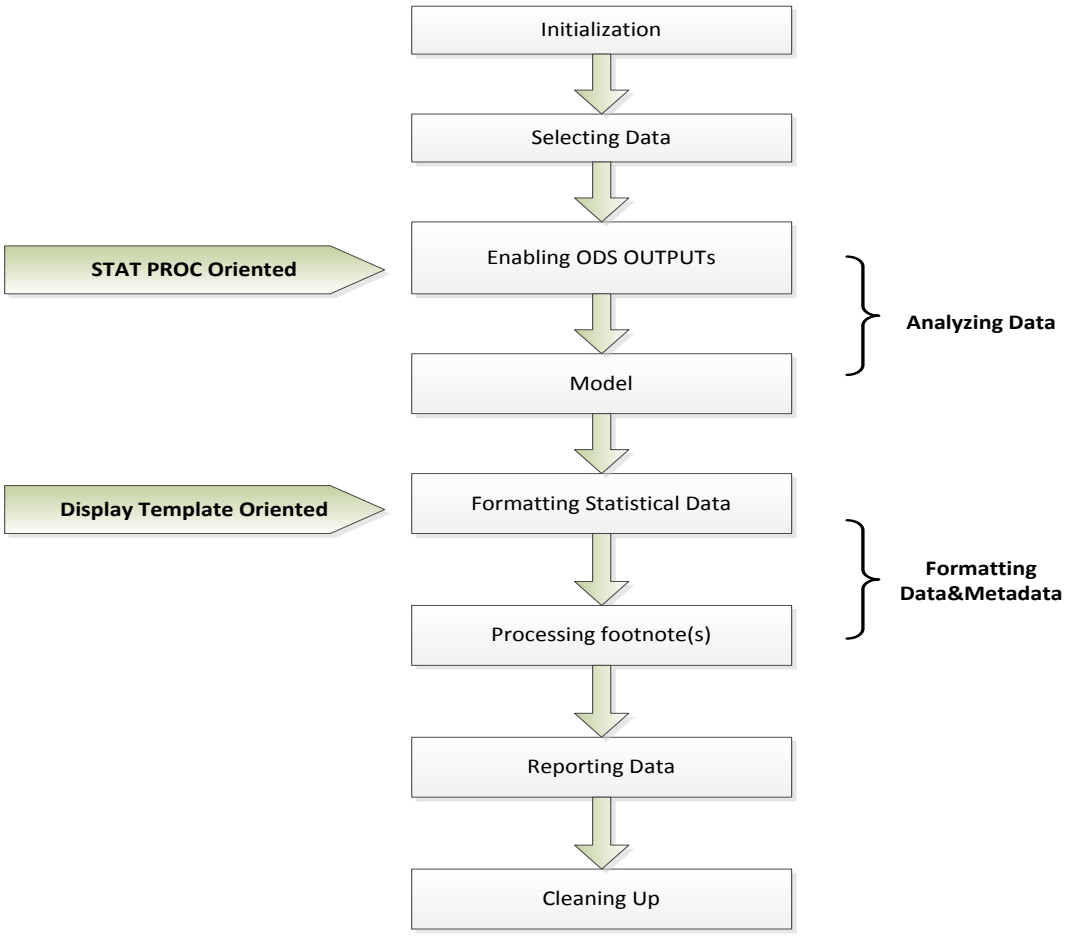

Figure 6 Programming Flow with Structured Design for Survival Analyses

Figure 7 gives one table template for proportional hazard analysis. To work out this table, SAS PROC PHREG is needed and ODS OUTPUT-ParameterEstimates enabled. PARAM is used as the Endpoint name. The unit❶ should be contained in PARAM as ADaM required. However, if unit not included in PARAM, the program may take DAYS as default unit and send WARNING message to the log:

%put %sysfunc(compress(WARNING)): UNIT does NOT exist in PARAM, assumed [days] as default unit.;

The xx❷ can be calculated with (1-alpha)*100. The alpha can be dynamically scanned from the MODEL parameter. If alpha is not specified in the MODEL, take alpha=0.05 as default.

Table TTTE - Parameter estimates of proportional hazard analysis

<Endpoint name (unit❶)>	Estimate	<Hazard ratio>	<xx❷% CI>	p-value
<parameter 1>	xx.xxxx	x.xx	(xx.xx , xx.xx)	x.xxxx
<parameter 2>	xx.xxxx	x.xx	(xx.xx , xx.xx)	x.xxxx
.....				

Figure 7 Display Template for Proportional Hazard Analysis

There may be other dynamic scanning to code other templates, e.g., scanning STRATA variable in the MODEL of PROC LIFETEST, or optionally get from column of ODS OUTPUT, which are not further specified here.

In a display auto-generation programming model, it is the display template-oriented and targeted for formatting statistical data.

To facilitate dynamic reporting, the formatted output (REPORTDATA) should be as close to the final display as possible such that PROC REPORT only functions similar to PROC PRINT: no further computing, labeling, and formatting needed in PROC REPORT. The SAS dataset REPORTDATA can be saved separately to facilitate QC process as well.

Since the footnotes become parameters, it is necessary to provide dynamic footnoting. In the next two subsections, the implementations of dynamic footnoting and reporting will be described separately.

For the last step--Cleaning Up, it is preferable to restore the SAS working environment to the status of prior to running the macro, e.g., delete all generated temporary datasets, clear any created temporary library name, etc.

DYNAMIC FOOTNOTING

To provide full flexibility of footnoting, the following cell notations for the REPORTDATA are defined (see figure 8).

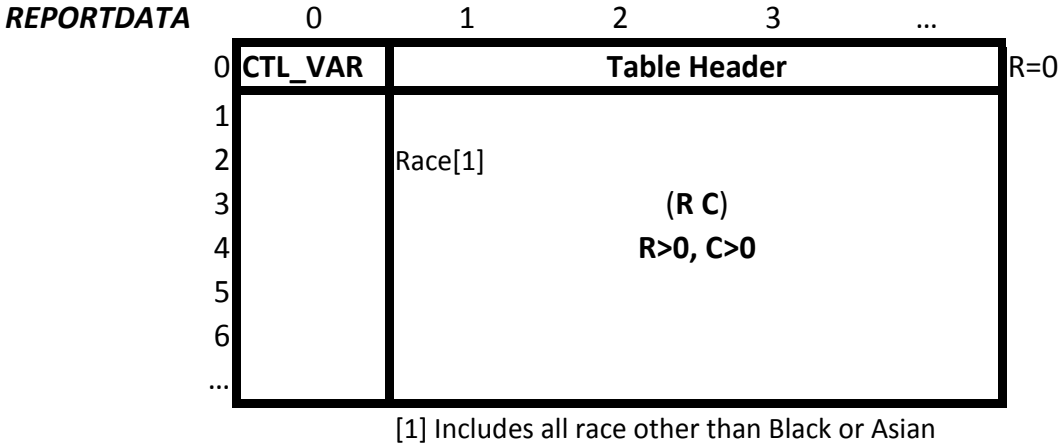

Figure 8 REPORTDATA Positioning for Dynamic Footnotes

Any report data location is treated as a cell with row(R)>0 and column(C)>0. When assigning a footnote icon to a location (R C) in the final report, C should be >=1. However, R can be any value. R<=0 is for positioning table headers. R=0 may be commonly used for positioning a footnote icon on one layer table header.

Here gives one example of footnote usage. A footnote parameter of *FOOT1=%str("[1] Includes all race other than Black or Asian"\{[1](2 1)});* instructs the macro to put icon [1] to the locations (2 1) in the REPORTDATA for the footnote of "*[1] Includes all race other than Black or Asian"*. See the Race[1] illustrated in figure 8.

To implement this dynamic footnote design with one layer table header, follow below algorithm:

```
For each pair {[n] (R C)} do;

  if R=0 then CAT[n] to the label of column C of REPORTDATA;

  if R>0 CAT[n] to the data of cell (R C);

  if R or C outside of table cell, put WARNINGs;

end;
```

In this way, the full flexibility of footnoting is provided to users.

DYNAMIC REPORTING

To implement dynamic output with PROC REPORT, three types of variables are defined: control variables (row control and page control), ID variables for those columns appearing at the left of every page of a report, and FLOW variables for possibly wrapping the values of character variables in the columns. The ID variables are also defined as FLOW, and all numeric variables are formatted as character variables. The column lengths and display labels in the report take default lengths and labels of variables. Figure 9 illustrates the defined three types of variables and the simplified implementations with PROC REPORT. The number of FLOW variables or ID variables, and the FLOW variable list or ID variable list can be easily retrieved by using PROC SQL.

VIEWTABLE: Work.Vars_typ

	COL_TYP	Variable Name	Variable Number	Variable Label
1	ID	DISP	1	Time to XXXXXX (days)
2	FLOW	EST	2	Estimate
3	FLOW	HR	3	Hazard ratio
4	FLOW	CI	4	95% CI
5	FLOW	P_VALUE	5	p-value
6	CTL	ROW_ORDER	6	

```
proc report data=reportdata nowd headline headskip spacing=1;
  columns ('--' ROW_ORDER
                %if &ID_varn>0   %then &ID_varlist;
                %if &FLOW_varn>0 %then &FLOW_varlist;
           );

  define ROW_ORDER /order noprint;
  %if &ID_varn>0 %then %do;
    %do i=1 %to &ID_varn.;
      %let ID_var=%scan(&ID_varlist.,&i.,' ');
      define &ID_var./display left ID FLOW;
    %end;
  %end;
  %if &FLOW_varn>0 %then %do;
    %do i=1 %to &FLOW_varn.;
      %let FLOW_var=%scan(&FLOW_varlist.,&i.,' ');
      define &FLOW_var./display left FLOW;
    %end;
  %end;

  break after ROW_ORDER /skip;
run;
```

Figure 9 Implementation of Dynamic Reporting

BENEFITS OF THE METHOD

In clinical statistical analyses, safety analyses are more straightforward, then subject evaluation analyses. A lot of MACROs have been developed for those analyses in the industry. As efficacy analyses are more flexible, thus much more challenges, especially for inferential analyses. The display auto-generation driven by analysis results metadata makes the codes maximally reusable. The method is generic, not only applying to inferential analyses but possibly to any kind of analyses.

The intention of analysis results metadata is to facilitate reviews. The method combines macro parameters and analysis results metadata together, thus facilitating coding and reviewing. From the macro interface, it's clear how the

display implemented, especially MODEL transparently as one macro parameter rather than coded inside of program. With finalized macro parameters, the validation can also be conducted smoothly and efficiently. In summary, the macro parameters aligned with ADaM analysis results metadata keep the primary programmer, the validation programmer, and the reviewer in the same page.

Currently the CDISC Analysis Results Metadata Specification for Define-XML supporting submission was published[3] although FDA has not listed it in the submission requirements. With the method discussed in this paper, along with display generation, the fields of analysis results metadata can be automatically extracted in the same program to facilitate the generation of metadata submission document.

DISCUSSION

Standard data structures by CDISC should lead to standard programs and result in auto-generation of displays. The implementation of auto-generations should be fully characterized as flexible and highly usable. Ease of use is the key for a solution to be widely adopted.

The method depicted in this paper is focused on display auto-generation with analysis results metadata and structured display templates as prerequisites. Up to now, the ADaM TFL standards are not published. Therefore, the company internal TFL display shells should be used.

Footnote is an important component in the display. Currently the footnote implementation introduced in this paper is only partially "dynamic", not fully auto-generated yet. To achieve the footnote to be completely auto-generated both for the footnote itself and positioning, it would be dynamically coded based on applied model and targeted display template.

With current analysis results metadata driven method, it still presents challenges how to automatically output for one-sided p-value or two-sided p-value, one-sided CI or two-sided CI, and inferential statistic type, etc. Those statistics requirements are normally specified in Statistical Analysis Plan (SAP). To facilitate auto-generations, more macro parameters would be needed and more fields in the analysis results metadata would be further defined.

Due to the ADaM analysis-ready, one WHERE is sufficient to select data in most cases. If two or more blocks in the table need to be with different WHERE on the same dataset, multiple WHEREs would be needed. However, for ADaM Occurrence Data Structure (ODS), e.g., ADAE, the programming model is designed as ADAE+ADSL for a display. For this case, from current practices, it's recommended still only one WHERE on ODS dataset for the main body of the table, and take WHERE on ADSL for denominator as default coding such as ADSL.SAFFL='Y' for safety analyses.

Compared with current programming methods in the pharmaceutical or biotech industries, the display auto-generation method introduced here presents only a small change, but the gains in productivity and efficiency are obviously significant. The user interface may be further improved by the macro parameters incorporated with a web-based user interface or integrated into company available reporting system.

ACKNOWLEDGMENTS

The author would like to thank Michael Pannucci and Nancy Bauer for their paper reviews and supports on the auto-generation work. The author would also like to thank other paper reviewers, whose names are not specifically listed here.

CONTACT INFORMATION

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

Name: Chengxin Li
Enterprise: Boehringer Ingelheim
Address: 900 Ridgebury Rd./P.O.Box 368
City, State ZIP: Ridgefield, CT 06877
Work Phone: 203 791-6835
Fax: 203 837-5552
E-mail: chengxin.li@boehringer-ingelheim.com

[3] http://cdisc.org/adam