PharmaSUG 2021 – Paper AD-002

Macro Development from Design to Implementation Applied to Occurrence Data Frequency Analysis

Chengxin Li, Medical Analytics, Regeneron Pharmaceuticals Inc.
Michael Pannucci, Medical Analytics, Regeneron Pharmaceuticals Inc.
Toshio Kimura, Medical Analytics, Regeneron Pharmaceuticals Inc.

Abstract

This paper demonstrates macro development from design to implementation detailing the macro design concept including dataset requirements, parameterization, statistical analysis, reporting and utility functions. While this paper uses occurrence data analyses as the motivating example, the concepts are generally applicable to macro development for any statistical analysis.

The paper contains two main sections: 1) General macro design concepts and 2) organization and flow of macro implementation modules with specific considerations for occurrence dataset structure (OCCDS) analysis components. The macro design concept includes key decisions regarding the scope of the macro. One design concept is to leverage the analysis-ready principle thereby requiring analysis-ready datasets such as analysis data model (ADaM) OCCDS. As such, the macro will not include data building steps. Another macro design decision is parameterization. This macro design aligns input parameters with analysis results metadata (ARM); therefore, not only will the macro output ARM as a description of the analysis, it will use ARM as the specification and input that drives the analysis. The organization and flow of the macro implementation modules will then be discussed along with explanation of core methods utilized in the macro including initiation, checks, analysis, formatting, reporting and clean up. OCCDS frequency analysis specific considerations will be explained as it applies to the various design considerations and implementation modules.

By following the depicted methods, similar applications for different statistical analyses can be developed.

Introduction

Reusable, organized and robust macros are at the core of efficient statistical programming. In order to develop such macros, thorough planning is required taking into account the overall macro design and organization. Key decisions regarding the macro scope and parameters must be made in order to simplify the user experience while allowing for a feature-rich macro that fulfills analysis needs. The macro should be organized into sequential modules where each module performs a specific objective. This paper will outline the macro development process from design to implementation using occurrence data analysis as an example.

Standardization of the input dataset greatly simplifies macro development since it eliminates the need for the macro to either build or modify the dataset into a common structure before further processing. This macro will leverage existing Clinical Data Interchange Standards Consortium (CDISC) standards. Since the motivating example is the analysis of occurrence data, the analysis data model (ADaM) occurrence dataset structure (OCCDS) will be used. Another key element of macro design is parameterization. Parameterization should be user-friendly, meet the demands of various analysis needs, and serve as a specification for the output being generated. The general approach taken in this macro implementation is to leverage the ADaM analysis results metadata (ARM) framework. The use of ARM has generally been limited metadata that describe outputs that are already generated. However, this macro utilizes ARM as a way of parametrizing and specifying the analysis to be performed.

The macro should also be organized into sequential, logical modules. This will allow for grouping SAS® code that performs a certain set of tasks. For example, all checking whether they are parameter checks or dataset checks are performed in the same module. Grouping code in this way facilitates module reuse and allows for easier maintenance.

Motivating Example: Frequency analysis for occurrence data

The analysis of occurrence data by frequency of subjects with events are critical statistical analyses in clinical studies and serves as the motivating example for this macro. This commonly includes summary tables for adverse events (AE), medical history (MH), and concomitant medications (CM). The data are frequently derived from medical coding dictionaries with built-in hierarchy (e.g., MedDRA or WHO) and are further described by levels of an event attribute such as AE severity (mild, moderate, severe). CDISC defined the OCCDS structure specifically to meet these needs. OCCDS is further explained as part of the data considerations in the section on macro design concepts.

The reporting of occurrence analysis typically includes count and percentage of subjects at the event level as well as at different attribute levels. There are 3 particular challenges with the analysis of occurrence data. First, not all subjects will have an event (e.g., some subjects may not have an AE). Since not all subjects will have an event, the denominator used in calculating the percentage is derived from the analysis population (e.g., safety set). Second, not all levels of an event attribute may be observed (e.g., none of the subjects have a severe AE). Even though a specific event attribute level may not be observed in the data, the table should display the complete set of the event attribute levels (such as mild, moderate, severe). Third, the sort order of the event and event attribute levels must be flexibly defined. Event and event attribute levels generally have a preferred display sort order which may be the frequency of the event displaying the most frequent AEs in descending order or by a sort order of the attribute levels such as mild, moderate and severe depending on the situation. The solutions to these challenges along with other analysis options are presented in the context of the overall macro design and implementation modules.

Figure 1 presents an AE analysis by severity display template for a common parallel group clinical study design.

Figure 1 Display Template for Hierarchical Occurrence Analysis (AE Summary Table)

Table 14.3.2: Treatment-Emergent Adverse Events by System Organ Class, Preferred Term and Severity
Safety Analysis Set

Primary System Organ Class Preferred Term Severity	Treatment Group A (N=XX)		Treatment Group B (N=XX)		Total (N=XX)	
	n	%	n	%	n	%
Subjects with at least one TEAE	xx	xx.x	xx	xx.x	xx	xx.x
Infections and infestations	xx	xx.x	xx	xx.x	xx	xx.x
Nasopharyngitis	xx	xx.x	xx	xx.x	xx	xx.x
Mild	xx	xx.x	xx	xx.x	xx	xx.x
Moderate	xx	xx.x	xx	xx.x	xx	xx.x
Severe	xx	xx.x	xx	xx.x	xx	xx.x
Sinusitis	xx	xx.x	xx	xx.x	xx	xx.x
Mild	xx	xx.x	xx	xx.x	xx	xx.x
Moderate	xx	xx.x	xx	xx.x	xx	xx.x
Severe	xx	xx.x	xx	xx.x	xx	xx.x

MedDRA Version: XX.X
TEAE: Treatment Emergent Adverse Events
A subject with multiple TEAEs is counted once for the same preferred term or system organ class with the worst severity.
The table is sorted by descending frequency order of the treatment total group for system organ class and preferred term.

Macro Design Concepts

1. Data

Standard input datasets greatly aides the development of reusable code. Fortunately, CDISC dataset implementation has become standard practice within the pharmaceutical industry; therefore, study data tabulation model (SDTM) and ADaM datasets can serve as those standard dataset inputs. To support statistical analyses, there are three common data structures defined in ADaM: 1) horizontal structure with a subject-level analysis dataset (ADSL), 2) vertical basic data structure (BDS), and 3) vertical OCCDS.

OCCDS is specifically designed to support occurrence data analyses. Occurrence data are categorized into intervention or event class domains (e.g., CM or AE) with the required topic variable --TRT or --TERM (e.g., CMTRT or AETERM). When developing ADaM, the topic variables are carried over with their dictionary-derived variables into the corresponding ADaM OCCDS dataset. Only the actual observed events or interventions are included in the SDTM or ADaM datasets.

To better understanding the macro design and its implementation within OCCDS, Table 1 summarizes key differences between two common ADaM data structures, OCCDS and BDS.

Key differences between OCCDS and BDS

Topic	OCCDS	BDS
Data Source	SDTM event or intervention domain	SDTM finding domain
Applications	Safety evaluations	Efficacy or safety evaluations
Subjects	Only includes subjects with actual observed events or interventions	Includes all subjects under specific populations, e.g., ITTFL or SAFFL
Analysis Variables	Usually structured hierarchical variables	Response variable (AVAL, CHG, PCHG) with associated PARAM/PARAMCD
Denominator	Summarized from ADSL	Summarized from BDS

Being analysis-ready is a core principle in all ADaM datasets including OCCDS. The analysis-ready principle implies that there should not be a need for any data processing between sub-setting the data and invoking the primary statistical procedure. For example, there is no need to merge data from additional dataset in order to perform the analysis. Leveraging the analysis-ready principle greatly simplifies the macro design and implementation as it negates the need to merge or build data within the macro, thereby focusing the scope of the macro to the statistical analysis procedures and output generation.

Technically, not all the occurrence analyses have to be with ADaM OCCDS. Some summary tables may use SDTM and ADSL (e.g., medical history summary table with MH and ADSL). SDTM as the input dataset is generally limited to situations when SDTM is immutable to the analysis (e.g., SDTM is ready for the analysis without any further derivations).

2. Macro Design Factors

Macros need to be designed with careful consideration of a multitude of factors. Table 2 summarizes the macro design consideration factors organized into the following categories: objective, engineering, CDISC principles, statistical analysis and reporting functionality, utility functions and documentation.

Table 2. Macro Concept Design Consideration Factors

Category	Factor	Detail
Objective	**Goal**	SAS macro to autogenerate displays in order to achieve high efficiency, quality, and reusability
Engineering	**Structured Design**	Structured design with sequential programming processes using module-based coding for readability, testing/debugging, and maintenance
	Simplicity	One macro for one type of analysis
	User Experience	User friendly parameterization and execution, e.g., easily understandable parameters and minimal execution processing time
CDISC Principles	**Data Structure**	Dataset requirements for the macro should be clearly defined
	ADaM Analysis-ready	Analysis-ready input dataset is required; after applying WHERE criteria, dataset is ready for statistical analysis
	ADaM ARM Driven	Macro parameters are harmonized with ARM; therefore, this macro is ARM driven; ARM define-xml can also be optionally stored as an output
Statistical Analysis and Reporting	**Analysis**	Analysis parameterization should allow for flexibility in how the statistical analyses are performed in order to cover the common variations for the analysis.
	Report Layout	Table layout must meet the analysis need. There should be considerations to cover the common variations in the table layout.
Utility	**Platform**	Both Windows SAS and Linux SAS are supported
	Debug	Keep intermediate datasets and print macro resolution for troubleshooting
	Results Dataset (RDS)	Dataset version of the display being generated should be stored. RDS facilitates quality control (QC) and can be the basis for alterative report layouts
	SAS LOG Window	Utilized as the communication channel to the user for any issues encountered during processing. All log messages including SAS logs and macro produced logs are saved to an external file and replayed back to the SAS LOG Window
Documentation	**Good Programming Practice (GPP)**	Adhere to GPP principles
	User Manual	User manual including macro requirements (such as input dataset requirements), macro scope, limitations, macro parameter dictionary and macro call examples with associated display types

3. Macro Parameters

Macro parameterization is another key element of macro design. The parameterization must meet several needs: 1) address analysis need, 2) easy to use, and 3) minimize redundancy or extraneous parameters. These parameters can be grouped into two categories which are owned by different roles. For example, a set of parameters may be owned by the statistician and thus documented in the statistical analysis plan. Examples of this type would include parameters that define the statistical analysis and are thus reflected in the title, population, and footnote parameters which are noted as one(❶) in the example macro call below. Another set of parameters may be owned by the statistical programmer as it relates specifically to implementation which are noted as two (❷) in the example macro call below.

In order to accommodate multiple values for a given field, for example allowing for Title Line 1 and Title Line 2 or allowing multiple analysis variable levels in the hierarchy, the '#' sign is used to separate multiple values. For example, multiple titles are separated by '#' as follows: Title=%str(Title 1 # Title 2). Sub-parameters may be needed to provide further parameterization within a parameter. Sub-parameters are denoted by using the '@' sign. For example, the sort order for an analysis variable can be further parameterized as follows: ASEV@preloadfmt=$AESEV. Sort order options will be further discussed in the Macro Implementation section. Lastly, paired parameters are separated using the '/' sign. For example, treatment character and treatment sort variables are entered as a pair and would be parameterized as follows: TRT01A/TRT01AN.

While flexibility allows for alternative options to be entered, this can be a double-edged sword as it then places burden on the user to enter a value for those parameters. One way to reduce this burden is through setting default values. Default values may be pre-defined within the macro to generally reflect basic statistical principles, e.g., safety analyses shall be with safety population set and actual treatment group. Several of the parameters define default values as shown in Table 3.

An example macro call is shown below, and Table 3 summarizes the macro parameters.

```
%macro FREQSUBJ1(
   ❶ TableID      =  %str(Table 14.3.1),
     Title        =  %str(Summary of Adverse Events by SOC, PT and Severity in DB Period
                  =        # Safety Population Set),
     Popu         =  %str(SAFFL),
     Footnote     =  %str(SOC and PT are coded using MedDRA version 13.1),

     Indata       =  %str(ADaM.ADAE),
     Where        =  %str(APERIODC='Blinded Treatment'),
     SubGroup     =  /*No subgroups are defined for this call*/,
   ❷ Analvar      =  %str(AESOC # AEDECOD # ASEV@preloadfmt=$AESEV.),
     ADSL_TRT     =  %str(TRT01A/TRT01AN),
     Total        =  %str(Study Drug 10mg # Study Drug 20mg),
     FreqThreshold =  0.05
);
```

Table 3. Macro Parameters

Category	Parameter	Description
❶ SAP	TableID	Table number; the number is also used to define the name of table output file
	Title	Table titles; separated with '#' for multiple title lines
	Popu	Analysis population set; default=SAFFL
	Footnote	User defined footnotes; separated with # for multiple lines of footnote. Macro generated footnotes are also created and explained in the Macro Implementation section.
❷ Programming	Indata	Input analysis dataset
	Where	Data selection criteria using SAS executable statements; default=1
	SubGroup	Subgroup analysis variable
	Analvar	Analysis variables; separated with '#' if multiple, and hierarchical relationship usually required between the variables; four sort options defined for an analysis variable and started with '@'
	ADSL_TRT	The treatment period variables from ADSL will be used as the treatment variables for analysis; treatment period variables will be entered as a pair of values with the first part being the treatment character variable followed by the treatment sort variable separated by '/'; default=TRT01A/TRT01AN
	Total	Total or subtotal for the treatment group; to display total which will include all treatment groups, set Total=Y or Yes; to display subtotal for selected treatment groups, include the treatment character values separated by '#' for the treatment groups to be included in the subtotal: %str(Study Drug 10mg # Study Drug 20mg); by default, treatment total or subtotal will not be displayed
	FreqThreshold	Frequency threshold minimum cut-off value for display; between 0 and 1, e.g., 0.05; the events for which the frequency is higher than the minimum frequency threshold will be displayed in the table; the threshold is applied to the frequency within any of the treatment groups

While it is important to detail the parameters included in the macro, it is also noteworthy to mention what is not included as macro parameters. Limiting parameters that users specify will make the macro more user-friendly. However, functionality without user input requires standardization or parameterization that are defined elsewhere. An example of functionality that depends on standardization is the use of ADSL. The location of ADSL does not have to be specifically defined as an input parameter. The macro will specifically search for "ADSL" (dataset must be named as such) in SASHELP.VTABLE view and select the one with the latest creation date. Another method is to define parameters outside of the macro. The utility functions built into the macro described in Table 2 uses this approach. For example, the value for DEBUG is defined outside of the macro (%let DEBUG=Y).

The macro parameterization also follows the CDISC ARM structure. CDISC describes ARM as metadata that describes the major attributes of specified analysis results. ARM provides a link between the results and the data used to generate the results. The implementation of ARM is meant to explain and provide more information about the analysis results; hence its name. However, the ARM concept and fields can also be used as a specification concept that prospectively drives analysis. ADaM 2.1 (Dec. 17, 2009) provides details with examples of ARM fields in inferential analyses from a BDS dataset. This concept has been covered in a 2015 conference paper[1] where the ARM fields were mapped to BDS analysis macro parameters. Following the same concept and method, Table 4 provides the mapping of ARM fields to OCCDS analysis macro parameters.

Table 4. Mapping ARM fields to Macro Parameters for OCCDS frequency analysis

ARM Field	Macro Parameter	Note
Display identifier	TableID	Table number
Display name	Title	Table title
Dataset	Indata	The name of OCCDS dataset used to generate the analysis result. ADSL is also used as a supportive dataset with OCCDS to perform the analysis such as summarizing denominator by treatment group under population set.
Selection criteria	Where	Data selection criteria from OCCDS, analysis-ready to perform the analysis
	Popu	Analysis population set; subjects selected from ADSL to summarize denominator; Popu may also be part of above WHERE statement.
Analysis variable	Analvar	One or more analysis variables, with hierarchical or parallel relationship
Documentation	Subgroup	Subgroup, frequency cut-off, treatment total/subtotal, or footnote may be specified in display template document or listed in the statistical analysis plan.
	FreqThreshold	
	Total	
	Footnote	
PARAM	N/A	As per ADaM 2.1 documentation, does not apply since the result is not based on a BDS analysis dataset.
PARAMCD	N/A	As per ADaM 2.1 documentation, does not apply since the result is not based on a BDS analysis dataset.
Reason	N/A	This is not a specification that is used to run the macro.
Programming statements	N/A	The analysis does not come from a complex statistical model.
Results identifier	N/A	The analysis does not come from multiple statistical procedures.

Macro Implementation

One basic software engineering principles is to organize code into sequential modules. This allows for organizing code into sections that achieves a specific objective. Modular code is easier to maintain, since any issues that may appear can be isolated to a particular module and displayed in the SAS log window. It is also easier to reuse, since the modules can be swapped in and out depending on the macro being developed. This macro design implements the sequential module concept. Table 5 summarizes the six sequential programming modules used in the macro.

Table 5. Sequential Programming Modules used in the Macro

Module	Objective	Description
1	**Initiation**	1) Set system options, 2) initiate sub-macros, 3) define output RTF template, and 4) record the initiation status (used later during clean-up)
2	**Parameter and Dataset Checking**	Check conformance with defined rules and requirements such as LIBREF, FILE PATH, and validity of macro parameters and write the outcomes of the checks if any to SAS LOG window
3	**Analysis**	Perform analysis
4	**Formatting**	Format the analysis results in preparation for reporting
5	**Reporting**	Generate the output and store in the designated folder
6	**Clean-up**	Reset settings and environment to the status prior to the macro call (such as deleting intermediate datasets)

[1] https://www.lexjansen.com/pharmasug/2015/AD/PharmaSUG-2015-AD01.pdf

Most of the modules such as Initiation, Parameter and Dataset Checking and Clean-up are common across analysis macros. The analysis specific differences will generally occur in the Analysis, Formatting and Reporting modules; therefore, the OCCDS analysis specific challenges will be discussed in some detail in those sections.

1. Initiation

The initiation module will define some of the set-up that is needed to run the macro, produce the output and aide in clean-up. This includes defining system options such as papersize which will be used in producing output, defining variables that are used by the macro but not included as a macro parameter (as previously discussed under Macro Parameters), defining the RTF template to be used and storing the initiation status which will be used during clean-up.

2. Parameter and Dataset Checking

Checking parameters and input datasets is another important practice in macro development. Parameter level checks include checking for required parameters and that the defined parameters are valid. Dataset checks can include valid libname and file paths, existence of required or defined datasets and variables, and format and/or informat availability.

3. Analysis

The Analysis Module will greatly vary as this is specific to the analysis need that is being addressed by the macro. As described in the section on Motivating Example, this macro deals with OCCDS frequency analysis. There are 3 particular challenges with the analysis of occurrence data.

First, not all subjects will have an event (e.g., some subjects may not have an AE). Since not all subjects will have an event, the denominator used in calculating the percentage is derived from the analysis population (e.g., safety set).

Second, not all levels of an event attribute may be observed (e.g., none of the subjects have a severe AE). Even though a specific event attribute level may not be observed in the data, the table should display the complete set of the event attribute levels (such as mild, moderate, severe).

Third, the sort order of the event and event attribute levels must be flexibly defined. Event and event attribute levels generally have a preferred display sort order which may be the frequency of the event displaying the most frequent AEs in descending order or by a sort order of the attribute levels such as mild, moderate and severe depending on the analysis need.

Since the first 2 challenges are with respect to Analysis, the solutions to those challenges will be presented below. The solution to the third challenge will be presented in Module 4 on Formatting.

The first challenge is that unlike BDS not all subjects appear in OCCDS. For example, some subjects may not have an AE; therefore, not all subjects will be observed in the OCCDS dataset. Since not all subjects will be included in the OCCDS dataset, the denominator of the analysis must be derived from ADSL instead of OCCDS alone.

The second challenge is that not all levels of an event attribute will be observed. For example, there may not be any subjects who had a severe AE. Even though not all levels of AE severity were observed in the data, the display should report all levels of the AE attribute: Mild, Moderate and Severe. Therefore, the macro must have a method to know all the levels of a category and perform a zero-fill.

The typical solution to display the complete set of categories is to create a table shell containing all possible categories and then merge the summary count results to the table shell; however, this method is inefficient. Alternatively, SAS provides a more powerful and efficient solution using the COMPLETETYPES and PRELOADFMT options. The COMPLETETYPES option is available in four procedures: PROC MEANS, PROC TABULATE, PROC SUMMARY and PROC REPORT. The format used in PRELOADFMT defines all possible levels, and the COMPLETETYPES option will generate and zero-fill the non-observed categories.

4. Formatting

After performing the analysis, the dataset must be further massaged before it is ready for reporting. The Formatting Module will perform these modifications. As previously mentioned, the third challenge to the occurrence analysis is to

define the sort order of attribute categories. The Formatting Module would be the ideal code block to ensure that the proper sort order is defined.

In developing a solution for this challenge, four methods to define the sort order are provided in the macro which are summarized in Table 6. The PRELOADFMT option also provides the complete set of categories as described in the section above. Additionally, the sequential order of format value can be used as the sort order. The ORDER option can be used if a numeric sort variable is available in the dataset. The INFORMAT option can also be used if an informat is available within which the numeric value will be used as the sort order. Lastly, the FREQ option will sort by the descending observed frequency. The default will evaluate this for the total of all treatment groups; however, the user can define this to only include a specific treatment group. If the sort order is not defined by the user, the macro will attempt to search for the necessary information in the priority order as shown in Table 6.

Table 6. Sort Order Implementation

Priority	Option	Description	Example
1	preloadfmt	The preloadfmt option uses a defined and compiled SAS format which includes both a complete list of available categories utilized by the COMPLETETYPES option while simultaneously used as the display sort order.	ASEV@preloadfmt=$AESEV. where $AESEV. format includes: 'MILD'='Mild', 'MODERATE'='Moderate', 'SEVERE'='Severe', and the display sort order will be the sequential order of Mild, Moderate, Severe
2	order	The order option uses a variable in the dataset as the sort order. The order variable to be used can be defined by the user. If the user does not define an order variable, the macro searches for a variable with a suffix of 'N' after the analysis variable name by following the *N naming conventions defined in ADaM Implementation Guide (ADaMIG).	ASEV@order=ASEVN where ASEVN can be used as a numeric sort order
3	informat	The informat option uses a defined SAS informat. The informat to be used can be defined by the user. The values of the informat are defined and compiled outside of the macro. If the user does not define an informat, the macro searches for an informat with a suffix 'N' after the analysis variable name.	ASEV@informat=AESEVN. where AESEVN. informat includes: 'MILD' =1, 'MODEREATE' =2, 'SEVERE' =3
4	freq	The freq option will sort the categories based on the observed frequency. By default, the frequency observed in the treatment total will be used, but the user can define the specific treatment group to be used.	AEDECOD@freq=%str(Study Drug 20mg) or AEDECOD@freq=%str(Total)

5. Reporting

The Reporting Module will contain the PROC REPORT that will output the final display. Most of the information required for this module are already defined elsewhere. The Reporting Module will pull those information together with the report-ready dataset in developing the PROC REPORT such as applying the predefined RTF template, displaying the proper indentation for the analysis variables in a hierarchical relationship, and printing titles and footnotes.

6. Clean-up

As a final step, the macro should clean-up and restore the settings and environment to the status prior to running the macro call. This includes removing intermediate datasets, formats and informats. One method to facilitate this process is to store the state of this information during the Initiation Module so that the Clean-up Module will know which datasets, formats, informats were created as part of the macro call.

Discussion

Developing a reusable, organized and robust macro is a complex undertaking. Thoughtful consideration is required from both a macro design and macro implementation perspective. Macro developers can use the macro design framework as outlined in this paper to critically evaluate and plan their own macro development journey. The macro implementation modules as presented in this paper can guide macro developers in defining the steps necessary in organizing blocks of code that will serve as the foundation to a more complex macro application.

One concept that is particularly noteworthy is the alignment of the macro parameters with ARM. The role of ARM has previously been limited to describing the analysis after the analysis has been performed. However, the use of ARM in prospectively specifying the analysis is the ultimate measure of linking analysis results with the data used to generate the analysis.

Further development of the implementation modules could lead to interchangeable parts. While some modules will differ by analysis need, some of the common core modules can be a sub-macro onto its own. Parameterization of these common modules can occur within the sub-macro or perhaps the parameterization can be driven by metadata stored outside of the macro.

The concepts presented in this paper provides a general framework for macro design and implementation which can be readily applied to a multitude of analysis macros.

Acknowledgement

The authors would like to thank Weiming Du for invaluable input and paper review.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:
Name: Chengxin Li
Enterprise: Regeneron Pharmaceutical Inc.
Address: 777 Old Saw Mill River Road
City, State ZIP: Tarrytown, NY 10591
Work Phone: (914) 847-3687
E-mail: chengxin.li@regeneron.com