PharmaSUG 2013 - Paper AD18

A Simple Interface for Metadata

Magnus Mengelbier, Limelogic AB, Malmö, Sweden

ABSTRACT

The use of metadata in Life Sciences, and specifically clinical trials, has been around from the very beginning. The most common form is to embed information about the trial or analysis in evolving standards, documentation and different forms of specifications. Very seldom are these readable from SAS and other analytics environments. We consider a simple interface that will allow the same details to be captured and managed consistently and provide a few powerful utilities to use it in programming.

INTRODUCTION

Metadata is a constant throughout the analysis and report programming for clinical data. Captured through protocols, analysis plans and specifications, it is very often inaccessible from SAS® and other analysis programs.

The effort to extend common document formats to incorporate additional metadata can be a simple or daunting exercise, depending on the acceptable degree of flexibility. As flexibility is introduced, complexity increases, which may render current document formats less usable.

The balance of desired flexibility, tools and metadata drives the development of both a metadata approach and the ensuing selection of an interface, both the technology and overall conventions and design. The end result provides two subtle interfaces that prove that it is possible to define rich metadata as well as provide simple access from within an analysis and report program.

METADATA

There are different views on the common definition of metadata and we shall adopt a simple, and possibly naïve view, that metadata is “data about data” or “information about data”. A common ground within Life Science is for metadata to combine both structural and descriptive or guide metadata, such that Systolic Blood Pressure at Visit 16 or the White Blood Cell (WBC) count laboratory result at baseline can be observed and included in analysis.

Structural metadata is used to describe the structure of variables, data sets and outputs, e.g. tables, listings, figures and graphs. Guide metadata would be used to identify the context of the observed data, such as a specific test, visit or time point.

The interface approach we shall consider employs both and intentionally obscures the distinction between the types of metadata used, whether it is a data set specification or a code list used by a specific variable. As such, metadata that is managed by our interfaces is simply information or data about data with focus on the end result.

COMMON TOOLS

The metadata used for analysis and reporting of clinical data is most often captured across several different documents. A few examples, and the ones we will focus on, are the well-known study protocols, analysis plans, data set specifications and output shells/mock-ups, although some may not be independent or standalone documents.

An important consideration is that the mentioned documents primarily follow under the direction, standards and conventions of the trial or analysis sponsor with any influence by industry standards as a secondary constraint. Another important consideration is that the documents are also business documents, commonly Portable Document Format (PDF) and Microsoft Word or Excel based on company templates, which can be difficult or near impossible to extract content from for use in programming.

The ultimate use of metadata has not changed as we are still expected to create source, intermediary, and analysis data sets and subsequently tables, listings and figures using either SAS or other similar analytic

environments. The continued adoption and maturing of industry standards beyond the data set level should facilitate convergence towards a common transparent industry best practice.

Due to the varying inflexibility of the document formats, e.g. PDF or Microsoft Word and Excel, they are common tools of the trade and references used in programming rather than a resource or source for analysis metadata. For the rest of the paper, we shall also assume that the analysis is *program generated*, which is slightly different than programmed as we shall discuss.

COMMON STRUCTURES

The metadata approach should ultimately be able to define or describe all the inputs and outputs that are used or produced in the analysis of clinical trial data, which includes interim analysis, study reports to pooled analysis and submissions. This implies that metadata can represent a data set as well as a table, listing, or figure/graph (TLF or TLG), which each look and displays quite differently. For now, we assume that a figure and graph are synonymous as is a common perspective.

The different deliverables do share some common traits, which can be very apparent when comparing two tables or a table and a figure. Some organisations go as far as to use listings to perform quality checks/controls (QC) on figures. It is also not uncommon that project planning considers both the total and unique number of data sets, tables, listings, and figures, in essence the concept of repeated structures within a narrow predefined and acceptable level of variation. We shall use both as a foundation to develop an approach in order for metadata to describe data sets and TLFs.

The first step is to identify common and differentiating characteristics for each. A data set or data table, as a simple example, is more or less, defined by a data set name, label, a set of keys, a sort order and an ordered list of variables. A variable can be subsequently defined by a variable name, label, type, length, format, code list or encoding. With little effort, we have described the general structure of a SAS data set that is applicable to both SDTM or ADaM data sets.

The data set example highlights the beginnings of a common structure to metadata. We aim to define metadata through a set of attributes. One of these attributes can be an ordered list or link to other metadata definitions, e.g the ordered list of variables. As we shall see, this approach can define anything from a data set to TLFs, although with varying degree of complexity and thus becomes the underlying concept for our new interface to metadata.

This also highlights a convention used for the remainder of this paper. We use the term define to mean describing or defining a data set or TLFs using metadata, which includes both structural and descriptive or guide metadata . Similarly, the term structure is used explicitly to describe how the metadata definitions are stored and referenced.

STANDARDS AND SPECIFICATIONS

Our primary challenge and eventual effort behind defining a metadata standard and subsequent interface is to reproduce the current and common standards and specification documents with one distinction. The goal is to find or design an interface that allows SAS and other analysis programs to read and consume any required metadata definitions.

The result shall assume that any non-essential and process specific definitions are contained within custom attributes. Before we decide on the interface, briefly consider some common metadata definitions required for our principal data set and TLF deliverables.

The paper considers variables, data sets and summary tables to construct conventions and standards for metadata definitions and assumes for brevity that the remaining standard deliverables, e.g. listings and figures, are variants or follow the same approach.

DATA SETS AND VARIABLES

The primary definition for a data set or data table, as we briefly discussed previously, is defined by a data set name, label, a set of keys, a sort order and an ordered list of variables. We also include the data set type, e.g. raw versus analysis or SDTM versus ADaM, as it can be an integral part of identifying and using a data set appropriately in the analysis and submission process.

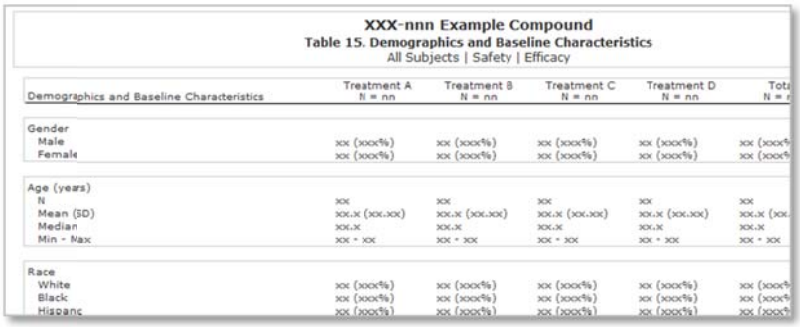
The variable in the above variable list is defined by a variable name, label, type, length, format, code list or encoding. A common approach to code lists and encoding is to use a value and a label or description, which can be as simple as *F* and *Female*, respectively. With SAS data sets, code lists are sometimes used interchangeably with user defined SAS formats, which we will avoid. In our context, user defined SAS formats are one of many SAS representations of code lists.

Another important element of the variable definition is the variable source, whether it is mapped or derived. The source can be a CRF page, a transfer data set or derived using simple or complex rules. Regardless, the approach should accommodate cases of varying complexity. A sufficient approach can be to identify source, e.g. mapped or derived, and the source or derivation rule reference (see section on rules and automation for further discussion).

If a variable is an independent metadata item or always defined within a data set is a matter of convention and preference. For example, do you create one definition for the Study ID variable in each data set or once and reuse. We shall for this paper employ the latter convention and treat variable as a separate definition that is linked to a data set through the ordered variable list.

TABLES

Summary tables (Figure 1) are another essential component for presenting analysis results. Very apparent is that tables look and are used differently than data sets and require its own metadata definition. As we shall discover, we can use the same concept of attributes and nested metadata items as for data sets, just that the resulting metadata definition is more verbose and slightly more complex.



XXX-nnn Example Compound
Table 15. Demographics and Baseline Characteristics
All Subjects | Safety | Efficacy

Demographics and Baseline Characteristics	Treatment A N = nn	Treatment B N = nn	Treatment C N = nn	Treatment D N = nn	Tota N = n
Gender					
Male	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%
Female	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%
Age (years)					
N	xx	xx	xx	xx	xx
Mean (SD)	xx.x (xx.xx)	xx.x (xx.xx)	xx.x (xx.xx)	xx.x (xx.xx)	xx.x (xx.
Median	xx.x	xx.x	xx.x	xx.x	xx.x
Min - Max	xx - xx	xx - xx	xx - xx	xx - xx	xx - xx
Race					
White	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%
Black	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%
Hispanic	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%)	xx (xxx%

Figure 1. Demographic and Baseline Characteristics summary table

A summary table most often consists of a table reference, one or more titles and footnotes, ordered set of columns and summary information, i.e. an ordered list of summary sections or blocks, which follows the general approach for data sets previously discussed. In addition, a summary table may summarise a subset of the data, either for a specific population and/or study phase, epoch or event, which we should also capture in the metadata definition.

The summary information, e.g. the summary table body, varies considerably if we consider the different summaries that are incorporated into analysis reporting. For example, the summary information format for *Demographics and Baseline Characteristics* and *Haematology Laboratory Results by Visit* illustrate just two cases that the metadata should be able to define. As a convention, summary information is composed of one or more distinct or repeated (visits and time points) summary sections or blocks.

The definition of the statistical summary block (Figure 2) is equally elementary, regardless if the summary format is for a variable at a single time point or across multiple visits and/or time points. The summary block consists of a label and an ordered list of statistics or categorical values, which in turn is defined by a label and format specification. As in the example above, a statistic can be a simple statistic or the more complex composite that concatenates two statistical estimates, such as Mean and SD to display as Mean (SD) or Minimum and Maximum values to form Min - Max.

Age (years)	
N	xx
Mean (SD)	xx.x (xx.xx)
Median	xx.x
Min - Max	xx - xx

Figure 2. Table summary section or block

There is also no barrier to extend the metadata for a statistical summary block to refer or link to a data set domain and variable. The summary block label can represent a custom label or simply refer to information in the variable metadata. Categorical value summaries, such as Race or Gender, will then be able to reference the values within a code list associated with a variable.

There will undoubtedly be summary tables that cannot easily be defined using metadata attributes, but that is more often a restriction on the interface rather than the metadata structure developed as part of this paper. As long as a summary table can be defined as a set or nested collection of attributes, then it is solely to provide a suitable interface, which is discussed in more detail below.

A (NEW) INTERFACE

The requirement for an efficient interface is apparent considering the time invested into maintaining all documentation associated with clinical trials and subsequent analysis and reports. The initial scope was restricted to the data sets specifications and the output specifications, e.g. shells and mock-ups, in order to keep the scope manageable.

The interface, singular, should be able to define the metadata for the following deliverables.

- Source or SDTM data sets
- Analysis or ADaM data sets
- Transport, transfer or special use data sets
- Tables
- Listings
- Figures

In addition, the ability to generate define files used for submissions was considered an essential feature.

MICROSOFT EXCEL APPROACH

The initial attempt was to use Microsoft Excel since it was already in use for data set specifications with existing utilities to incorporate the metadata definitions in SAS programs. As most organisations, the format of the Microsoft Excel workbook was one data set specification per Worksheet with a top section for defining the data set name, label, keys and sort order and a lower section that contained the ordered list of variables and associated definitions.

The variable definitions were also defined as one variable per row with the columns representing the variable attributes, e.g. one column each for name, label, type, length, source, reference, etc.

A mature set of utilities were used to validate and ensure consistency of variable definitions across data set specifications, such that common variables were defined consistently.

The Excel-based specification slowly transformed as the metadata structure evolved. All variables were defined under a *Variables* tab using the one line per variable convention. Each data set in turn would reference the variable by name and define any specific source reference and derivation rules, if other than what was defined in the variable definition under the *Variables* tab.

Using this same convention, a prototype specification of summary tables was created using a Microsoft Excel Workbook. One summary table specification was defined per Worksheet with the top section to define the section reference, titles, footnotes, columns and any data subsets, e.g. populations. Each summary block of the table was defined in the lower section, with one line per statistic. In an attempt to create more consistency, the summary blocks were moved to a *Summary* tab, similar to the data set specification *Variables* tab.

The inefficiency of the approach became very apparent as different summary table types were added. Essentially, the choice was to simplify, i.e. restrict flexibility, in order to retain Microsoft Excel or to investigate alternative interfaces.

LEVEL OF DETAIL

The Microsoft Excel experiment did clearly highlight the intricate balance of flexibility, the interface, metadata structure, and the subsequent usability. As we introduced greater flexibility, the metadata became increasingly verbose and complex, even for the most simple TLF. If end users are expected to edit raw metadata values, flexibility will induce complexity. Most often flexibility is sacrificed and configurability restricted to a few options associated with standard templates. The concept of drawing a table, listing, figure or graph as in Microsoft Word would not be possible.

The metadata structure selected is entirely based on attributes, which provides considerable flexibility and can be extended as other types of metadata, such as elements of the Statistical Analysis Plan, is added. The concept is to allow an item, e.g. data set or TLF, to define its own collection of metadata attributes and relationships.

The actual structure is simple; an item reference, type, name and collections for associated attributes and links to other metadata items. As an example, the complete summary table definition for the Demography and Baseline Characteristics summary table, excerpt provided as Figure 3, is beyond 250 attributes that encompasses everything from the table reference number, titles, footnotes, treatment group references (treatment group labels is by design defined in another metadata structure) down to the pattern and decimal precision for presenting each statistic. Given that the metadata attributes are defined specifically for each type, the effort to create, manage and subsequently use metadata definitions is at most a logistical exercise rather than coercing or bending a rigid standard that was initially designed for another type of deliverable or output.

H42

	E	F	G
2	t_dm	study-compound	XXX-nnn Example Compound
3	t_dm	table-number	15
4	t_dm	table-title	Demographics and baseline characteristics
5	t_dm	population	All Subjects
6	t_dm	population	Safety
7	t_dm	population	Efficacy
8	t_dm	treatment-group	trt_a
9	t_dm	treatment-group-label	Treatment A
10	t_dm	treatment-group	trt_b
11	t_dm	treatment-group-label	Treatment B
12	t_dm	treatment-group	trt_c
13	t_dm	treatment-group-label	Treatment C
14	t_dm	treatment-group	trt_d
15	t_dm	treatment-group-label	Treatment D
16	t_dm	treatment-group-total	right
17	t_dm	treatment-group-total-label	Total
18	t_dm	table-body-title	Demographics and baseline characteristics
19	t_dm	section-label	Age (years)
20	t_dm	section-variable	ADDM.AGE
21	t_dm	section-variable-unit	ADDM.AGEU
22	t_dm	section-statistics	N
23	t_dm	section-statistics-label	N
24	t_dm	section-statistics-format	8
25	t_dm	section-statistics-justify	decimal
26	t_dm	section-statistics-select	_ALL_
27	t_dm	section-statistics-exclude	_NONE_
28	t_dm	section-statistics	MEAN_SD
29	t_dm	section-statistics-label	Mean (SD)
30	t_dm	section-statistics-format	null
31	t_dm	section-statistics-justify	space
32	t_dm	section-statistics-select	_ALL_
33	t_dm	section-statistics-exclude	_NONE_
34	t_dm	section-statistics	MEAN
35	t_dm	section-statistics-label	_COMPOSITE_
36	t_dm	section-statistics-format	8.1
37	t_dm	section-statistics-justify	null
38	t_dm	section-statistics-select	null
39	t_dm	section-statistics-exclude	null
40	t_dm	section-statistics	SD
41	t_dm	section-statistics-label	_COMPOSITE_
42	t_dm	section-statistics-format	8.2

Figure 3. Excerpt of metadata definition for Demography and Baseline Characteristics summary table

The metadata structure finally selected was also not driven by technology as it is solely based on evolving business requirements, essentially the opportunity to provide access to existing metadata from within a SAS or other analysis program. This is the same metadata that is represented in PDF and Microsoft Excel and Word documents. In addition, it was recognised early that standards management was pivotal and a central design consideration. The difficulty would be to marry flexibility with a strict standards management philosophy.

THE INTERFACE

The current, and this may change with future technology, is a simple Web interface based on HTML and Javascript using common techniques and standard libraries and components. The interface approach is to have the user interact with metadata entirely through graphical representations and elements (Figure 4). As such, adding a new variable to a data set or table summary section block is through an HTML form, rather than manipulate the raw metadata structure directly.

The principal design goal was, and remains, to provide a natural interface for each of the different metadata types, e.g. your data set, table, listing, graph and figure. The data set is represented as a grid much like the Microsoft Excel specifications that is common throughout the industry. However, the interface for drawing a summary table is inherently more complex and introduces the principle that tables are constructed one summary block at a time rather than through one all-inclusive edit. More interestingly, the final interface closely resembles the interaction that users are accustomed to through Microsoft Work when drawing a table shell or mock-up.

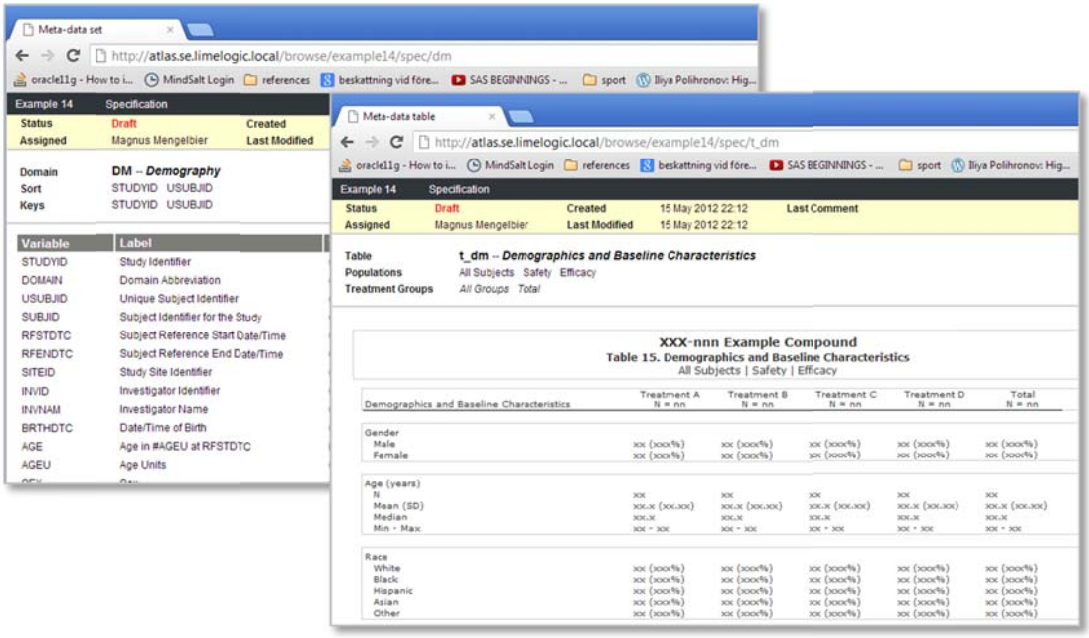


Figure 4. The prototype Web interface for editing a data set (back) and a summary table (front)

THE OTHER INTERFACE

The Web interface is practical to create, manage and view metadata definitions, but the interface simply replaces PDFs and Microsoft Excel and Word. A few simple SAS macros, some discussed below, ensure that the metadata edited through the Web interface is efficiently and effortlessly incorporated into regular programming.

ACCESS TO METADATA

The SAS macro %rules_metadata() provides direct access to the raw metadata definition for a single item. Using a data set as an example, this implies the data set type, name, label, keys, sort order and the ordered list of variables, but not the definition for each variable included in the data set. The design is intentional given the arbitrary levels of nesting allowed by the metadata structure.

```
%rules_metadata( meta = dm, output = work.meta_dm );

%rules_metadata( meta = t_dm, output = work.meta_t_dm );
```

The *meta* parameter is a reserved name that references the metadata for a single item and is consistently used through all the macros. The *output* parameter is the output data set that will contain the retrieved metadata,

The %rules_metadata() macro includes the experimental parameter *levels*, which would be used to retrieve nested metadata items. In our example, we could specify *levels = 2* to also retrieve the variable item metadata linked to the data set.

At this point, we have access to all the detailed metadata that is defined through the user interface. However, we are still required to construct the specified deliverables through regular programming as the metadata is solely a specification.

BUILDING DATA SETS AND OUTPUTS

Two experimental macros have been developed to automatically build data sets, %rules_make_dataset(), and summary tables, %rules_make_table(), using the metadata definitions for each.

```
/*  generate data sets  */

%rules_make_dataset( meta = dm, output = sdtms.dm );
%rules_make_dataset( meta = addm, output = adams.addm );
```

```
/*  generate a summary table */

filename tables "X:\/my\tables\folder";
%rules_make_table( meta = t_dm, output = tables );
```

The %rules_make_dataset() macro is designed to assemble the final output data set from a collection of temporary data sets located in the SAS WORK library. The data set make macro is able to identify the temporary data set using a standard naming convention based on the meta parameter value and principal variable name.

The %rules_make_table() macro is similarly designed to assemble the final output table from a temporary or permanent reporting data set that is either located in the SAS libraries WORK or a specified library, respectively. For simplicity, the reporting data set and subsequent output has the same name as the meta parameter.

As these two macros evolve, additional parameters may be introduced to provide additional features or further flexibility.

RULES AND AUTOMATION

An interesting development is to employ metadata to drive automation, i.e. to create all the programs necessary to create analysis data sets and ensuing TLFs from source data. It is very obvious that automation is not a trivial exercise, which is evident in the comprehensive business processes and specifications, both explicit and implicit, which revolve around Life Science analysis.

A practical approach is to implement a rule based programming convention, whereby derivations can be either be standard or non-standard. Standard derivations are performed using a set of standard rules. Non-standard derivations are simply performed by new rules, which in practice contribute to a growing set of standard rules.

A rule consists of two components, the descriptive information that includes the derivation specification and the code. The descriptive information is of course rule metadata that can be used in our interface. The code is essentially a small macro or other code template that can be referenced or included in analysis programs [1].

Our metadata is extended incorporate a rule book, wherein the rules are managed. It is more practical to separate derivation rules associated with data sets and those that derive statistics. The rule book is therefore associated with data set specifications rather than a singular output.

This also emphasises a convention adopted early in the development process. It is more efficient for the analysis data sets to follow the convention that only generating statistics and rendering the output is required to produce an output from analysis data sets, sometimes loosely referred to as one PROC away. The process can be further evolved by separating the effort to generate statistics and the final rendering of the output [2][3].

```
/*  generate tables */

filename p_adams "my folder for ADaM data set programs";
%rules_make_programs( type = adam, output = p_adams );
```

A rule book and the analysis ready data sets promises to allow the majority of programs to be generated directly from metadata. The experimental %rules_make_program() macro above is designed to generate SAS program code from the rules associated with a data set specification. In actuality, it generates two programs. The first is a standard program that is entirely constructed from metadata. The second program is an empty SAS program that can be used to add custom code, given that there will most often be specific cases not implemented through rules. The second program is actually included, through a simple %include, in the first program just prior to the final rendering of the data step.

```
/*  generate an entire analysis */

filename a_prg " my analysis programs folder";
%rules_make_analysis( type = safety, output = a_prg );
```

If we can generate one program, why not all programs for a specific or an entire analysis? The experimental macro %rules_make_analysis() fails more often than it succeeds in generating all the required programs and

executing them in the correct order, which highlights the complex logistics and intricate interdependencies in generating a complete analysis.

CONCLUSION

The effort and subsequent usability when extending common formats, e.g. Microsoft Excel, to incorporate metadata associated with summary tables, listings and figures is possible, but there can be a significant constraint on flexibility if the goal is to retain a practical format.

It is possible to provide very flexible and extendible metadata, if the view of the metadata definitions are managed closely and appropriate interfaces, note plural, are provided. A single interface would be difficult to capture all the complexities that are inherited from the analysis and report tasks associated with generating data sets and outputs.

The graphical interface discussed is solely a replacement of the current tools and document formats, however the new interface does enable simple access to metadata from within a program. The mentioned SAS macros provide a fair indication of how easy it can be to access and use metadata to drive analysis and report programming.

REFERENCES

[1] Mengelbier, Magnus, 2012, “Programming By Rules”, *Proceedings of the PhUSE 2012 Conference*
[2] Mengelbier, Magnus, 2007, “Reporting by Elements – A New Way of Looking at Clinical Reports”, *Proceedings of the PhUSE 2007 Conference*
[3] Mengelbier, Magnus, 2013, “Clinical Reporting by Elements”, *Proceedings of the PharmaSUG 2013 Conference*

CONTACT INFORMATION

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

Magnus Mengelbier
Limelogic AB
Jungmansgatan 12
211 19 Malmö
Sweden

E-mail: mmr@limelogic.com
Web: www.limelogic.com