PharmaSUG 2013 - Paper BB09

Processing MedDRA SMQs: Using Recursive Programming to Handle Hierarchical Data Structures

Paul Stutzman, Axio Research, Seattle, WA

ABSTRACT

Programmers sometimes need to process information that is organized hierarchically. Recursive programming is an effective tool for handling these tasks.

Recursive programs are programs or functions that call themselves. They are traditionally used in contexts where a basic task can be performed iteratively, as is the case when one processes an operating system's file system – moving through its directories and their subdirectories.

The MedDRA adverse event coding system provides a hierarchical way to gather related adverse event Preferred Terms (PTs) into hierarchical groups based on defined medical conditions or areas of interest. These groupings are called Standardized MedDRA Queries (SMQs). Recursive SAS macros are an efficient tool for processing SMQs.

INTRODUCTION

Programmers in a pharmaceutical setting are sometimes faced with data structures that are hierarchical. One common example of this is adverse events which are organized into specific areas of interest. The MedDRA coding dictionary provides an organizational structure called Standardized MedDRA Queries (SMQs) that enables programmers and biostatisticians to look at related events for defined medical conditions or specific areas of interest.

For example, events related to Ischaemic heart disease might be of interest. MedDRA provides an SMQ for events that are related to Myocardial infarction and it provides an SMQ for Other ischaemic heart disease. Both of these SMQs fall under the more general SMQ Ischaemic heart disease. See Figure 1.

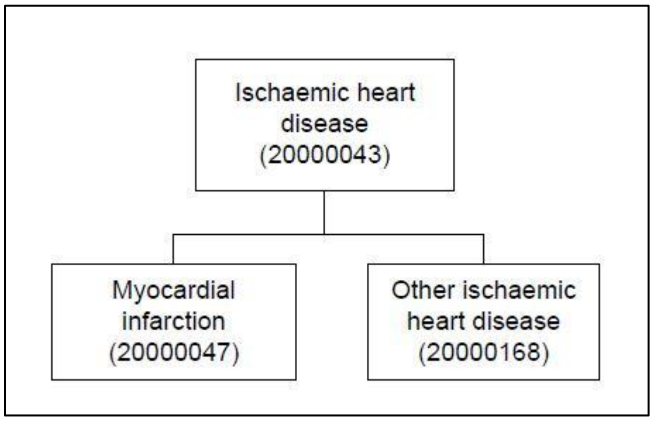

Ischaemic heart disease (20000043)
Myocardial infarction (20000047)
Other ischaemic heart disease (20000168)

Figure 1. Ischaemic heart disease SMQs

A programmer must be able to handle the hierarchical organization of this structure if they are to capture all the events that fall under the highest-level SMQ.

Recursive programming is a technique that is particularly well suited to handling hierarchical data structures. It involves solving complex processing problems by breaking them down into smaller tasks, and them performing these tasks iteratively. In this example, a program could process a specific SMQ. The program could then call itself in order to process any subordinate SMQs it finds, processing them in the same manner as it did when processing the original SMQ.

Recursive processing can be quite efficient and can it can simplify programming efforts. This paper will describe recursive programming in more detail, it will describe the MedDRA data structure – particularly as it relates to SMQs, and it will show how recursive programming can be employed to process MedDRA SMQs effectively.

RECURSIVE PROGRAMMING

Recursive programs are simply programs that call themselves. They seek to solve complex problems by breaking them up into simpler tasks that are performed iteratively. Recursive programs can be written is SAS by creating a SAS MACRO that calls itself.

A classic example of a problem that can be solved through recursion is calculating factorials. The factorial of an integer n (shown as n!) is the product of all integers up to and including the number of interest. For example, the factorial of 5 is 1 x 2 x 3 x 4 x 5 = 120.

A recursive program could be written to solve factorials by multiplying the starting value by the factorial of the next smaller integer, then repeating this process until the "next smaller integer" reaches 1. For example:

```
5! = 5 x 4!
   = 5 x 4 x 3!
   = 5 x 4 x 3 x 2!
   = 5 x 4 x 3 x 2 x 1!
   = 5 x 4 x 3 x 2 x 1
```

From as SAS coding perspective, this function could be performed by a macro similar to the following:

```
%MACRO Factorial(n);
       %if &n=1 %then 1;
       %else %eval(&n * %Factorial(%eval(&n - 1)));
%MEND Factorial;

%PUT The Factorial of 5 equals %Factorial(5).;
```

As it executes, the recursive nature of this program becomes evident as `%Factorial(%eval(&n - 1))` takes on the factorial of diminishing integers, as in:

```
%eval(5 * %Factorial(4));
%eval(5 * 4 * %Factorial(3));
%eval(5 * 4 * 3 * %Factorial(2));
%eval(5 * 4 * 3 * 2 * %Factorial(1));
%eval(5 * 4 * 3 * 2 * 1);
```

The resulting message in the SAS log from the %PUT statement is show in Output 1.

```
The Factorial of 5 equals 120.
```

Output 1. SAS log entry generated by: `%PUT The Factorial of 5 equals %Factorial(5).;`

Another class of problems that recursive programs can solve particularly well involves situations where it is necessary to process hierarchical data structures. Traversing an operating system's file structure falls into this category. The recursive program would start by processing an individual folder or directory in the structure, and then it would call itself recursively in order to process any subfolders or subdirectories it finds – handling them in an identical manner to the way the original folder or directory was processed. This general approach can be used to handle any hierarchal structure, including classification structures for adverse events, as will be shown later in this paper.

MedDRA AND STANDARDIZED MedDRA QUERIES (SMQs)

Processing adverse event (AE) information is critical for assessing safety within clinical trial settings. One aspect of the analysis of AEs is the coding of these events. Coding is the process of assigning consistent specific names to events based on a standardized medical dictionary. In addition to providing consistent terminology, the standardized hierarchal organization of coded events can provide ways of gathering and analyzing related events and events of particular interest in a consistent manner.

The Medical Dictionary for Regulatory Activities (MedDRA) is the most commonly use coding dictionary in North America, Europe and Japan. MedDRA was developed by International Conference on Harmonisation (ICH). Its goal is to provide common terminology that enables "health authorities and the biopharmaceutical industry to more readily exchange and analyze data related to the safe use of medical products." [2]

THE BASIC MedDRA DATA STRUCTURE

The hierarchy of MedDRA's terms is shown in Figure 2.

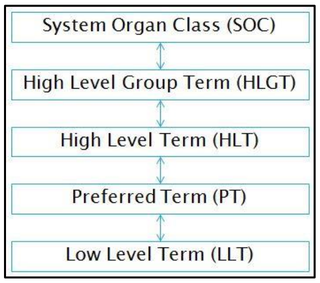


Figure 2. Structural Hierarchy of the MedDRA Terminology

Most of the analyses performed on coded AEs have to do with reporting on Preferred Terms (PTs). This is the lowest level of classification that specifies a "distinct descriptor (single medical concept) for a symptom, sign, disease, diagnosis, therapeutic indication, investigation, surgical, or medical procedure, and medical, social, or family history characteristic." [3] Low Level Terms (LLTs) are actually the lowest level terms in the hierarchy, but they are not often used in analyses, since different LLTs may be synonyms for the same PT (e.g. "Acquired immuno deficiency syndrome" and "AIDS" are two different LLTs that refer to the same PT "Acquired immunodeficiency syndrome"). There are approximately 20,000 unique PTs in the most current version of MedDRA.

Each PT is primarily associated with one of 26 System Organ Classes (SOCs). This is the highest grouping level and it relates to the etiology (e.g., SOC Infections and infestations), manifestation site (e.g., SOC Gastrointestinal disorders) or purpose (e.g., SOC Surgical and medical procedures) of the event. [3] PTs may be associated with multiple SOCs through differing High Level Terms (HLTs) and High Level Group Terms (HLGTs), but the reporting of PTs is most commonly done in terms of their single primary SOCs.

While this reporting of PTs within primary SOCs can be useful, a mechanism was needed in order to look at all events that may be related to specific defined medical conditions or areas of interest, without regard to where the PTs fall within the HLT/HLGT/SOC hierarchy. Hence, Standardised MedDRA Queries were born.

STANDARDISED MedDRA QUERIES (SMQs)

The creation of Standardised MedDRA Queries (SMQs) arose from a need for standardized tools for identifying and retrieving safety data. They are groupings of PTs that relate to defined medical conditions and areas of interest. Examples include Chronic kidney disease, Dementia, and Hepatic disorders. "The included terms may relate to signs, symptoms, diagnoses, syndromes, physical findings, laboratory and other physiologic test data, etc." [4]

SMQs may be hierarchal in nature. They may include a high level SMQ that contains various subordinate SMQs. Up to five levels may be represented in the SMQ structure. The following figure shows an example of SMQs of this type.

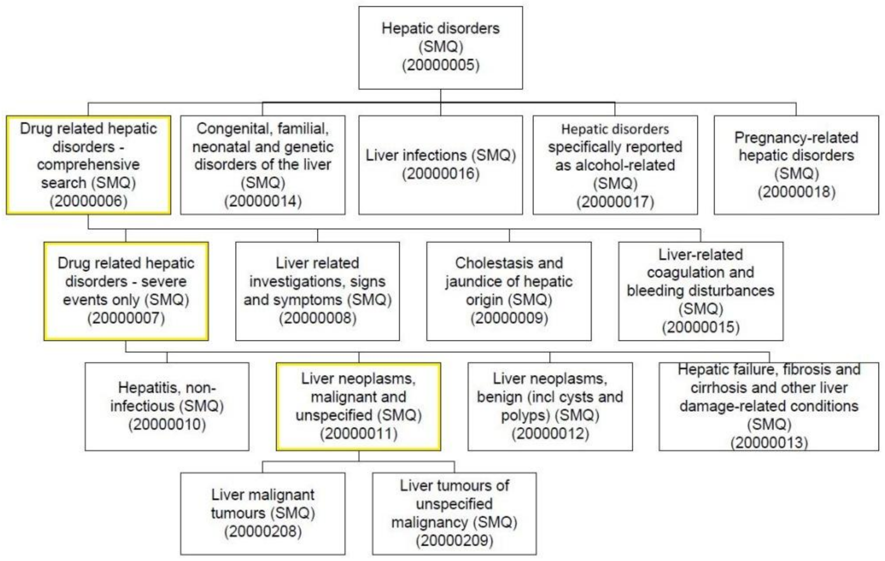

Hepatic disorders (SMQ) (20000005)
Drug related hepatic disorders - comprehensive search (SMQ) (20000006)
Congenital, familial, neonatal and genetic disorders of the liver (SMQ) (20000014)
Liver infections (SMQ) (20000016)
Hepatic disorders specifically reported as alcohol-related (SMQ) (20000017)
Pregnancy-related hepatic disorders (SMQ) (20000018)
Drug related hepatic disorders - severe events only (SMQ) (20000007)
Liver related investigations, signs and symptoms (SMQ) (20000008)
Cholestasis and jaundice of hepatic origin (SMQ) (20000009)
Liver-related coagulation and bleeding disturbances (SMQ) (20000015)
Hepatitis, non-infectious (SMQ) (20000010)
Liver neoplasms, malignant and unspecified (SMQ) (20000011)
Liver neoplasms, benign (incl cysts and polyps) (SMQ) (20000012)
Hepatic failure, fibrosis and cirrhosis and other liver damage-related conditions (SMQ) (20000013)
Liver malignant tumours (SMQ) (20000208)
Liver tumours of unspecified malignancy (SMQ) (20000209)

Figure 3. Hepatic disorders SMQ and its subordinate SMQs

The Hepatic disorders SMQ is made up of fifteen subordinate SMQs, three of which (highlighted in yellow) have subordinate SMQs of their own. In order to find all the PTs related to Hepatic Disorders, all PTs for each subordinate SMQ (and their subordinate SMQs) must be found.

PTs within an SMQ are defined as being of interest in a “narrow” scope or a “broad” scope. “Narrow” scope applies to events that are highly likely to represent the condition of interest, whereas “broad” scope events include conditions that may be “of little or no interest on closer inspection.” [4] For example, “narrow” terms included under the Dementia SMQ include “Dementia Alzheimer's type,” “Frontotemporal dementia,” and “Vascular dementia.” “Broad” terms included under the Dementia SMQ include “Apathy,” “Mood swings,” and “Negativism.” A “broad” search includes all “narrow” terms along with additional terms.

MedDRA AND SMQ METADATA AND CONTENT

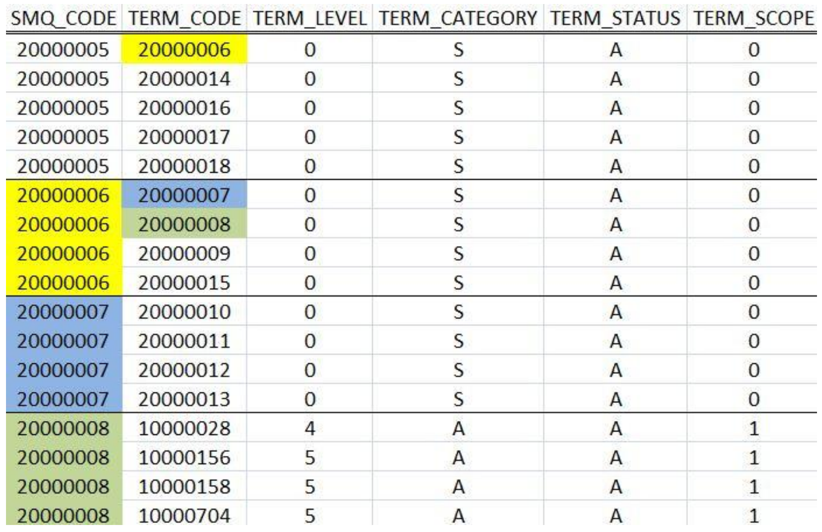
Subscribers to MedDRA my download a variety of files from the MedDRA Maintenance and Support Services Organization (MedDRA MSSO) Website. [2] These files include documentation regarding MedDRA, and they include the data files that makeup the MedDRA metadata and its content.

The documentation associated with each version of MedDRA includes an Introductory Guide for MedDRA itself and an Introductory Guide for SMQs. There is also a .PDF file that describes the layout and content for each of the ASCII text files that are the actual MedDRA content.

The MedDRA content itself is found in fourteen dollar-sign delimited ASCII text files which are housed in a folder called MedAscii. Three of these files are particularly useful when processing SMQs:

- **smq_list.asc**: contains one record for each of the high-level and subordinate SMQs. It includes the unique 8-digit code for each SMQ, the textual name of each SMQ and other descriptive information. This is a good place to find the unique code that is associated with a particular SMQ.
- **pt.asc**: contains one record for each Preferred Term (PT). It includes the unique 8-digit code for each PT, the textual name of each PT, the 8-digit code that identifies the PT’s primary SOC and other descriptive information. SMQ s are only associated with PT codes, so this is a good place to find textual names for associated PTs.
- **smq_content.asc**: contains the information for mapping PTs to SMQs. For each SMQ, there is one record for each SMQ or PT that is subordinate to it. These records contain the unique 8-digit code for the SMQ, the 8-digit code for the SMQ or PT that is subordinate to it, indicators for narrow or broad scope, and other relevant information. This is the main file that will be processed in order to determine the PTs that make up any SMQ.

  A portion of a smq_content file that relates to the Hepatic disorders SMQ from Figure 3 is shown in Figure 4. Records where TERM_CATEGORY = S contain a TERM_CODE that represents a subordinate SMQ; records where TERM_CATEGORY = A contain a TERM_CODE that represents an event.



SMQ_CODE	TERM_CODE	TERM_LEVEL	TERM_CATEGORY	TERM_STATUS	TERM_SCOPE
20000005	20000006	0	S	A	0
20000005	20000014	0	S	A	0
20000005	20000016	0	S	A	0
20000005	20000017	0	S	A	0
20000005	20000018	0	S	A	0
20000006	20000007	0	S	A	0
20000006	20000008	0	S	A	0
20000006	20000009	0	S	A	0
20000006	20000015	0	S	A	0
20000007	20000010	0	S	A	0
20000007	20000011	0	S	A	0
20000007	20000012	0	S	A	0
20000007	20000013	0	S	A	0
20000008	10000028	4	A	A	1
20000008	10000156	5	A	A	1
20000008	10000158	5	A	A	1
20000008	10000704	5	A	A	1

Figure 4.Some smq_content records for the Hepatic disorders SMQ

The first five records represent the five SMQs that are subordinate to the main Hepatic disorders SMQ (20000005). SMQ 20000006 (Drug related hepatic disorders - comprehensive search) has four subordinate

SMQs. It isn't until SMQ 20000008 (Liver related investigations, signs and symptoms) – which is subordinate to SMQ 20000006, which is subordinate to SMQ 20000005 – that individual events are shown.

It is this hierarchial nature of this structure that makes at an ideal candidate for recursive programming.

USING A RECURSIVE PROGRAMMING APPROCACH TO PROCESSING MedDRA SMQs.

Objective: Create a SAS data set that contains codes for all the PTs associated with an SMQ and its subordinate SMQs.

Strategy: The main data source for this system is the smq_content.asc file mentioned above. (For this example, it is assumed that this text file has already been converted into a SAS dataset.) Processing of this data source will be done by a recursive program structure based on two SAS macros.

A macro will process each SMQ individually as follows:

1. If the SMQ has subordinate SMQs, stick their IDs into a new macro variable. We'll call this macro variable &_SMQCDs.
2. If the SMQ has PTs, append them to the output data set.

A second macro will handle the hierarchal nature of the SMQ structure, ensuring all subordinate SMQs are processed. This is the recursive macro. This macro will:

1. Call the macro above, passing it the code for the SMQ that was passed to this macro. Initially, this is the main, highest-level SMQ of interest; subsequently it will be the code for a subordinate SMQ.
2. If &_SMQCDs is not empty (there are subordinate SMQs), create local macro variables (SMQ1, SMQ2… SMQn) for each subordinate SMQ code in &_SMQCDs.
3. For each SMQn variable created in step 2, this macro calls itself, passing itself the SMQ code in SMQn.

The method presented here was chosen for its simplicity. Steps 2 and 3 could be combined, for instance, by changing the first macro to append the new SMQ codes to an already-existing &_SMQCDs – instead of creating a new &_SMQCDs with each invocation.

Code for the first macro:

```
**************************************************************************
*  GetSMQ is called by SMQ_PT, which sets _SMQ_CODE to the code for the
*  current SMQ of interest.
*
*  GetSMQ puts the codes for all active SMQs that are subordinate to
*  _SMQ_CODE into a macro variable called _SMQCDs.  It also appends codes
*  for all active PTs that make up _SMQ_CODE to a dataset called SMQ.
**************************************************************************;
%MACRO GetSMQ;
   PROC SQL NOPRINT;
        *  Put codes for all active SMQs subordinate to _SMQ_CODE into _SMQCDs  *;
        select TERM_CODE into :_SMQCDs separated BY " "
        from InData.MedDRA_SMQ_Content
        where SMQ_CODE = &_SMQ_CODE and TERM_CATEGORY eq "S" and TERM_STATUS = "A";

        *  Codes for all active PTs that make up _SMQ_CODE  *;
        create table _smq as
        select SMQ_CODE, TERM_CODE, TERM_LEVEL, TERM_SCOPE
        from InData.MedDRA_SMQ_Content
        where SMQ_CODE = &_SMQ_CODE and TERM_LEVEL = 4 and TERM_STATUS = "A";
   QUIT;

   *  Append the PT codes that make up _SMQ_CODE to SMQ data set  *;
   PROC DATASETS LIBRARY=work NOLIST;
        append base=SMQ data=_smq;
        delete _smq;
   RUN;
%MEND GetSMQ;
```

Code for the second macro:

```
*************************************************************************
*  The macro SMQ_PT calls itself recursively to get all the PT codes
*  for the SMQ identified by _SMQ_CODE and all SMQs that are subordinate
*  to it.  It calls GetSMQ once per SMQ.
*************************************************************************;
%MACRO SMQ_PT(_SMQ_CODE);
   %local _i _n _SMQCDs;

   *  Call the first macro to process the current SMQ  *;
   %GetSMQ;

   *  Create a macro variable for each subordinate SMQ  *;
   %let _i = 1;
   %do %while (%scan(&_SMQCDs, &_i) ne);
       %local SMQ&_i;
       %let SMQ&_i = %scan(&_SMQCDs, &_i);
       %let _i = %eval(&_i + 1);
   %end;
   %let _n = %eval(&_i - 1);

   *  Call SMQ_PT recursively once for each subordinate SMQ  *;
   %do _i = 1 %to &_n;
       %SMQ_PT(&&SMQ&_i);
   %end;
%MEND SMQ_PT;
```

Note: One must always be aware of the scope of macro variables in any setting where macros call other macros. This is particularly true when macros call themselves recursively. GLOBAL and LOCAL statements should define macro variable scopes as narrowly as possible, in order to avoid potential pitfalls.

Invoking code:

```
*  20000005 is the code for 'Hepatic disorders (SMQ)'  *;
%SMQ_PT(20000005);
```

CONCLUSION

MedDRA SMQs provide a structure for looking at adverse events that are related to a defined medical condition or are of particular interest. This can be very important when assessing safety within a clinical trial setting. Recursive programming provides a tool that can effectively process the SMQ hierarchy in order to indentify and analyze events that are related to these specific SMQs.

REFERENCES

[1] Bartlett, Jonathan. “Mastering recursive programming.” IBM DeveloperWorks. Jun 16, 2005. Available at http://www.ibm.com/developerworks/linux/library/l-recurs/index.html.

[2] MedDRA Maintenance and Support Services Organization. “Welcome to MedDRA and the MSSO.” April 1, 2013. Available at http://www.meddramsso.com/index.asp.

[3] International Conference on Harmonisation (ICH). September, 2012. Introductory Guide MedDRA Version 15.1. MedDRA Maintenance and Support Services Organization

[4] International Conference on Harmonisation (ICH). September, 2012. Introductory Guide for Standardised MedDRA Queries (SMQs) Version 15.1. MedDRA Maintenance and Support Services Organization

CONTACT INFORMATION

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

Paul Stutzman
SAS Programmer Specialist
Axio Research, LLC
2601 4th Avenue, Suite 200

Seattle, WA 98121
pauls@axioresearch.com
www.axioresearch.com