PH-04-2014

Programming in a Distributed Data Network Environment: A Perspective from the Mini-Sentinel Pilot Project

Jennifer R. Popovic, Harvard Pilgrim Health Care Institute/Harvard Medical School

Boston, MA

ABSTRACT

Multi-site, healthcare-related distributed data networks are becoming increasingly popular, particularly at a time when 'big data' and privacy can have competing priorities. Distributed data networks allow individual-level data to remain behind the firewall of the data holder, permitting the secure execution of queries against those local data and the return of aggregated data produced from those queries to the requester. These networks allow the use of multiple, varied sources of data for study purposes ranging from public health surveillance to comparative effectiveness research, without compromising data holders' concerns surrounding data security, patient privacy or proprietary interests.

INTRODUCTION

This paper introduces the concept of a distributed data network, its purposes and benefits and, using the Mini-Sentinel pilot project as a case study, discusses using SAS to design and build infrastructure for a successful multi-site, collaborative distributed data network. Mini-Sentinel is a pilot project sponsored by the U.S. Food and Drug Administration (FDA) to create an active surveillance system - the Sentinel System - to monitor the safety of FDA-regulated medical products.

This paper focuses on the data and programming aspects of distributed data networks but also visits governance, administrative and systems-related issues as they relate to the maintenance of a technical network. This paper is focused primarily on the use of distributed data networks for healthcare-related surveillance and research, but distributed data networks can exist for any kind of purpose with any kind of data.

WHAT IS A DISTRIBUTED DATA NETWORK

A distributed data network is one in which no central repository of data exists. Rather, data are maintained by and reside behind the firewall of a data holder, which allows indirect access to their data through the use of a standard query approach. The data are therefore 'distributed' due to the lack of centrality.

Distributed data networks exist by a set of guiding principles:

- Data holder sites maintain control over their data,
- Data holder sites have standardized their data to a common data model,
- Data holder sites' ongoing involvement is needed in order to interpret data and findings; they know their data the best, so are true ***partners*** in the network (indeed, the terms 'data holder' and 'data partner' will be used interchangeably in this paper),
- Programming code gets securely distributed to data holders for them to execute locally and in a manner that makes it easy for them to execute,
- Following execution of programming code, data holders return aggregated results, rather than individual-level data, to the requestor.

Though a distributed data network is not required to have a central coordinating center, the existence of one is critical to building the necessary infrastructure and governance needed to maintain high-quality data and processes within a network.

PURPOSE OF A DISTRIBUTED DATA NETWORK

Distributed data networks often allow for access to more data than what a single or centralized site might be able to offer. By pooling resources (data) across several sites, with security in place such that each site maintains ownership over its own data, these networks provide several key benefits, including:

- Offering alternative ways to study occurrences of rare outcomes, uptake or usage of new drugs or therapies, and diverse populations of individuals,
- Achieving greater statistical power due to larger numbers of observations,
- Encouraging the development of novel analytic and statistical methods that do not rely on the use of patient-level data,
- Challenging programmers to approach projects with the intention of building reusable, flexible and scalable programs for infrastructure purposes, rather than a series of one-off programs,
- Addressing and alleviating data holders' concerns over data security, patient privacy and proprietary interests.

COMMON DATA MODEL AND STANDARD QUERY APPROACH

The purpose of any common data model is to standardize the format and content of data, such that standardized applications, tools and methods can be applied to them. Figure 1 is a schematic intended to represent, at a high-level, the process for populating a common data model using healthcare claims data from a data partner site. In this example, the data partner site is a health insurance company or an integrated managed care consortium.

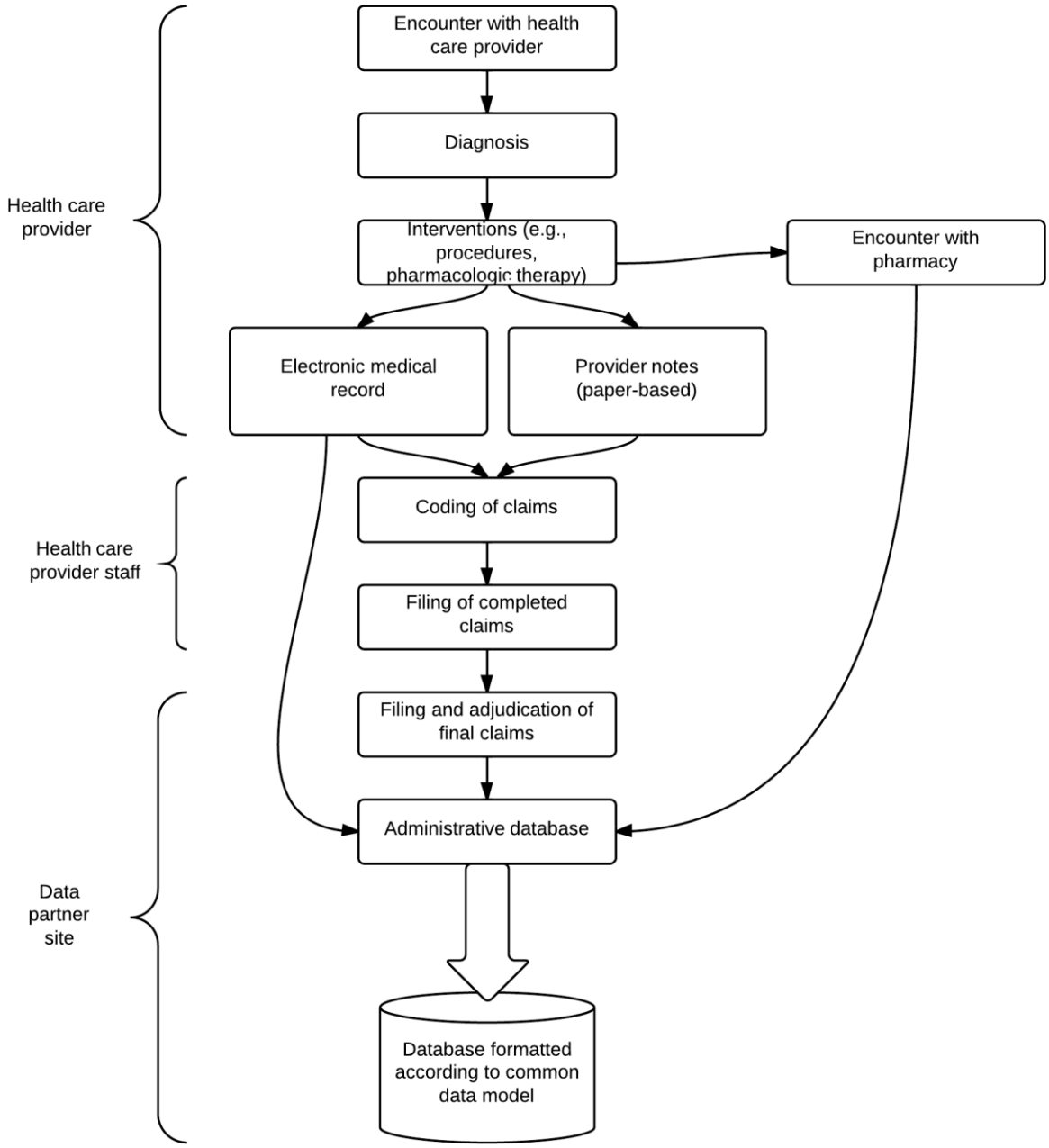

Figure 1. Process for populating a common data model using healthcare data from a data partner site

As patients interact with the healthcare delivery system, those interactions are captured in electronic medical record systems and/or administrative claims data systems. The data partner site in this example maintains its data in a central administrative database or warehouse and then converts those data into a common data model according to model specifications. All of these data reside behind the data partner's firewall, and that data partner maintains control over the usage and transfer of any of their data at all times.

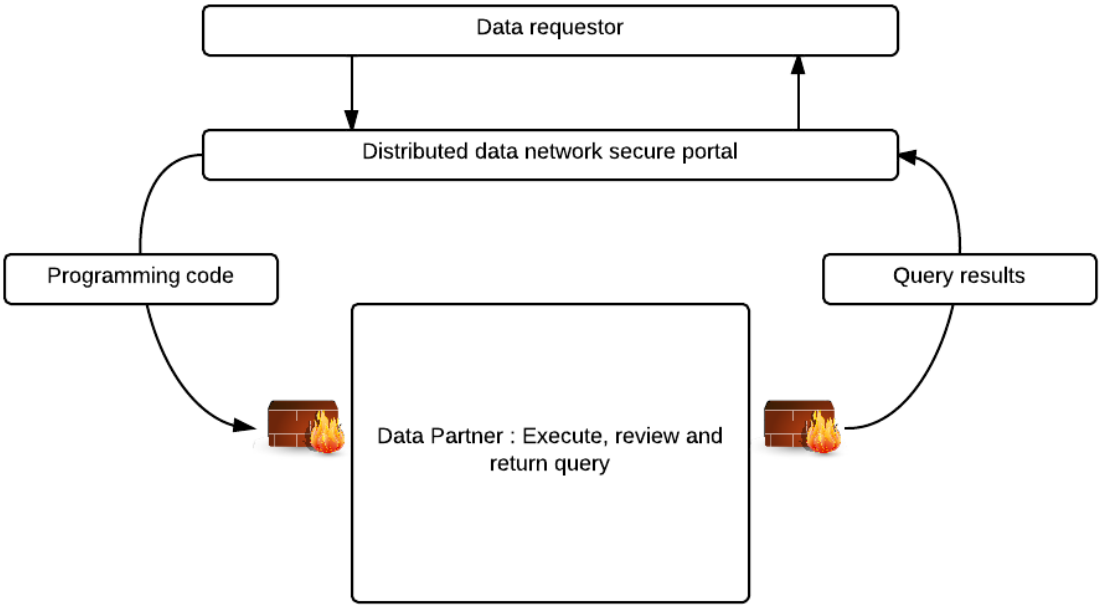

Figure 2. Query approach in a distributed data network

Figure 2 shows a schematic for the flow of data between a data requestor and a data partner site, depicting at a high-level a standard query approach in a distributed data network.

The existence of a secure file transfer protocol portal ensures that proper security is in place on both sides of the program and data transfer process. Data at the partner site reside behind the data partner's firewall.

DISTRIBUTED DATA NETWORKS IN EXISTANCE

The list below includes the names of healthcare-related distributed data networks in existence. This is by no means meant to be an exhaustive list but is rather intended to be used for illustrative purposes.

- FDA Mini-Sentinel
- PCORnet: The National Patient-Centered Clinical Research Network
- Innovation in Medical Evidence Development and Surveillance (IMEDS) Project.
- NIH Health Care Systems Research Collaboratory
- HMO Research Network
- Cancer and Cardiovascular Research Networks
- Vaccine Safety Datalink

Many of these networks have a particular focus. For example, the Mini-Sentinel project was funded to build an active, prospective surveillance system for health product safety; PCORNet focuses on conducting comparative effectiveness and patient-centered outcomes research; and the NIH Health Care Systems Research Collaboratory's focus is to improve the way clinical trials are conducted by creating infrastructure for collaborative research.

Several of these networks share the same data holders/partners. Some may also share the same common data model and, potentially, infrastructure as the backbone to support the manner in which their network operates and queries data.

INTRODUCTION TO OUR CASE STUDY: MINI-SENTINEL

Mini-Sentinel is a pilot project sponsored by FDA to create a distributed data network and supporting infrastructure in order to enable an active surveillance system for monitoring the safety of drugs, biologics and devices—effectively, any FDA-regulated product—in the United States.

Section 905 of the Food and Drug Administration Amendments Act (FDAAA) of 2007 mandated the FDA to develop an enhanced ability to monitor the safety of drugs after they reach the market. This system, named Mini-Sentinel in its pilot phase (and to be called Sentinel thereafter), is intended to augment FDA's existing post-market safety monitoring systems. Current systems rely on FDA gathering information about their regulated products through programs that rely on external sources (product manufacturers, consumers, patients, and healthcare professionals) to report suspected adverse reactions to FDA. This type of safety monitoring is known as "passive surveillance." In contrast, Sentinel is intended to be an "active surveillance" system, as it will enable FDA to initiate its own safety evaluations that use available electronic healthcare data to investigate the safety of medical products.

The Mini-Sentinel Distributed Database (MSDD) currently consists of quality-checked data held by 18 partner organizations (health insurers or managed care consortiums). As of July 2014, the MSDD contained data on 178 million individuals, nearly 400 million person-years of observation time, 4 billion outpatient dispensings and 4 billion unique medical encounters.

Data partners standardize their data from their source systems into the Mini-Sentinel Common Data Model (MSCDM) and store those datasets as SAS datasets. Each site maintains physical control and ownership of their data, controls all uses of their data and controls all transfer of their data.

Figure 3 depicts the various tables included in the MSCDM. The MSCDM consists of a suite of several tables; six of the tables are considered 'core' and are present across all data partner sites (enrollment, demographic, dispensing (outpatient only), encounter, diagnosis, procedure). There are additional tables that are considered ancillary, as they are not present at all sites. Those include death, cause of death, laboratory tests and vital signs.

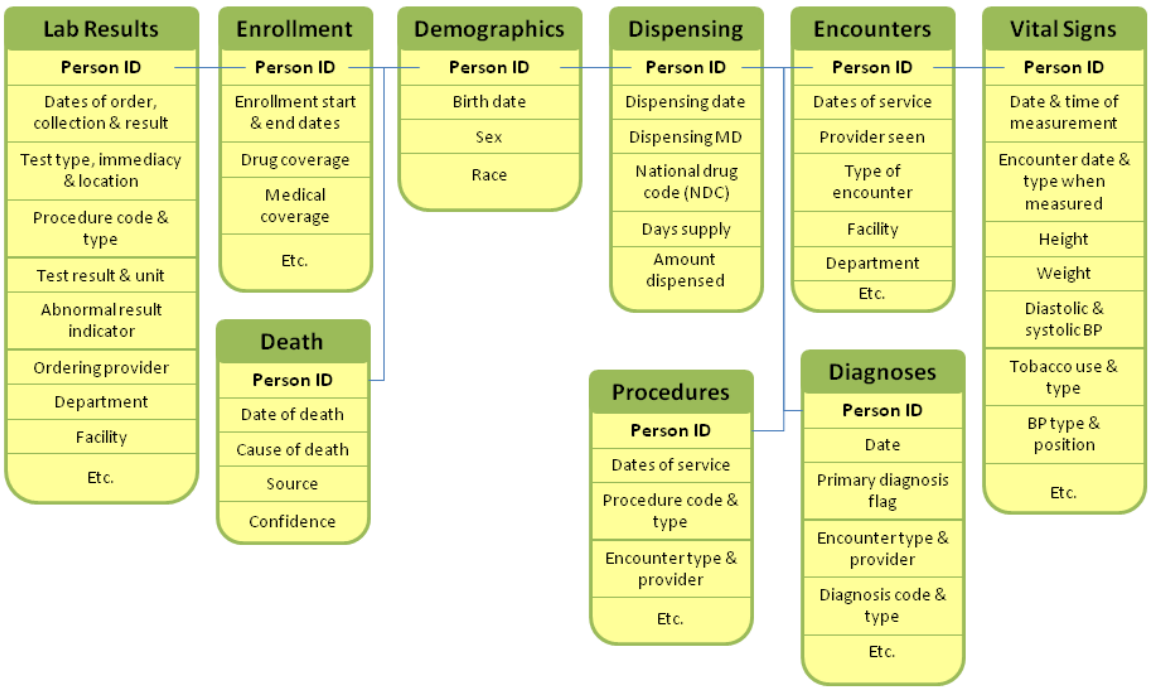

Figure 3. Mini-Sentinel Common Data Model

Data sites refresh their source data into MSCDM-formatted data quarterly to annually, depending on the site.

The Mini-Sentinel Operations Center (MSOC) is the centralized coordinating center for the entire project and plays the role of both a bridge and a gatekeeper between data partner sites and requesters. The dual-roles serve several important functions:

- To protect data holders from inappropriate use of their data,
- To protect requestors from asking the wrong kind of questions given available data.
- To protect requesters from asking for the wrong kind of data to answer their questions,

WHAT IS INFRASTRUCTURE?

Infrastructure is defined as the basic equipment and structures needed for a system to function properly.

Infrastructure-building within Mini-Sentinel revolves around the coordinating center developing and executing standard operating procedures encompassing the concepts below.

- Strategic approaches to developing SAS programs for reusability, flexibility and scalability,
- Auditing programming code to ensure adherence to overall goals of the analysis and to the concepts of reusability, flexibility and scalability,
- Quality and characterization checks of all data in the network
    - Making sure the format of each data partners' data adheres to the common data model,
    - Characterizing the nature and extent of data anomalies (e.g., overlapping enrollment periods, missing data where there should not be any, invalid data values),
- Reporting procedures to ensure appropriate presentation and interpretation of data,
- Procedures and systems to facilitate the distribution of SAS programming code to data partner sites for execution,
- Procedures and systems to facilitate the return of data from data partner sites following the execution of distributed SAS programming code: output files, including all SAS logs, should be routed to a single output folder, to be zipped and returned to the coordinating center.

Standard operating procedures drive the way new and existing programming tools and infrastructure are built and maintained, establish roles and responsibilities around processes and foster high-quality work products.

SAS PROGRAMMING INFRASTRUCTURE

The core concepts of building data- and programming-related infrastructure in a distributed data network are to recognize analytic- and programming-approach patterns where they exist, routinize programming tasks whenever possible, approach all programming tasks with reusability, flexibility and scalability in mind (rather than producing a series of one-offs), and to not reinvent the wheel.

We stress the concepts of reusability and flexibility because there is so much overlap in the analytic programming approaches used across the projects we support. One project may be interested in studying a cohort of patients exposed to drug A that experienced outcome event X, while another project may be interested in a different exposure and outcome pairing but an otherwise similar analysis. Building flexible programs with regard to study parameters saves programming and auditing time and effort, and ensures consistency in analytic approach across studies.

Program scalability is an important concept in a distributed data network because, for the sake of analytic consistency and administrative ease, it is imperative to approach programming with a "one program, multiple sites" frame-of-mind. This means, however, that programming code must be able to run in a variety of different computing environments, on data of various sizes.

The MSOC has written a set of flexible, reusable SAS programs that carry out several different types of common epidemiologic analyses; for example, identifying medical events of interest within a cohort of patients exposure to a drug or procedure. The codes and algorithms representing the exposure and event are flexibly defined within a set of input lookup tables (SAS datasets), as are other analysis-specific parameters such as study period dates, allowable enrollment gap, age groups to include, look-back period to define incident/new exposures, inclusion/exclusion criteria, and so on.

Using these standard-approach, parameterized programs provides for considerable flexibility while significantly reducing programming time and subsequent program audit/review efforts needed to respond to requestor questions.

All of the core infrastructure SAS programs have been audited, pre-tested and validated. Similarly, all output produced is consistent and predictable across all data sites, stream-lining data aggregation and analytic reporting activities. These programs are often used for feasibility or prep-to-research purposes prior to investing time and resources in larger, more in-depth and statistically complex surveillance or protocol-based assessment projects.

The in-depth, custom studies we support typically require custom programming, and the MSOC has built infrastructure to aid those efforts, as well. We maintain a library of programming “tools”, consisting of over a dozen utility and analytic SAS macro tools that each carry out a discreet task. These programs are flexibly written to allow users to specify certain analysis-specific parameters, with the intention of standardizing routine programming tasks, such as:

- Selecting a cohort of members exposed to specific medical products,
- Creating continuous treatment episodes,
- Identifying individuals with continuous enrollment surrounding an index date,
- Automating checking SAS logs for errors, warnings and notes of interest,
- “Deidentifying” a dataset by assigning a randomly generated 'caseid' to one or more user-specified variables in an existing dataset,
- Counting the number of distinct medical encounters from a reference date and within a user-specified look back period, to serve as a proxy for medical utilization,
- Computing a combined comorbidity score based on patients’ diagnosis and procedure claims from a reference date and within a user-specified look back period.

Any program written for distribution to one or more data partners must also allow for each data partner site to edit a portion of that program with their site-specific information. For example, nobody except an analyst at the site executing the program would know the physical drive location and names of their MSCDM-formatted datasets or the physical drive locations to which output datasets created by the program they are executing should be written. To make the program edit process easy for data partner sites—and standard across sites—the MSOC requires that an editable program header be included in the top-most portion of any distributed SAS program. Below is an example of a standard program header:

```
*-----------------------------------
* Assign STANDARD Parameters
*-----------------------------------;

/* Define Data Partner and Site IDs */

%let DPID=ms;
%let SITEID=oc;


/* Define Paths: End-of-path slashes are NOT needed (e.g., %let msoc=P:\project\msoc) */


/* Specify the full path location for your site's MSCDM-formatted data tables */
%let mscdm=Q:\mscdm;


/* Specify MSCDM table names */
%let enrtable=enrollment;
%let demtable=demographic;
%let distable=dispensing;
%let diatable=diagnosis;
%let proctable=procedure;
%let enctable=encounter;
%let deathtable=death;
%let codtable=cause_of_death;
%let labtable=labs;
%let vittable=vitals;


/* Specify the full path locations for this workplan's subdirectories */
%let msoc=P:\project\msoc;
%let dplocal=P:\project\dplocal;
%let infolder=P:\project\inputfiles;
```

```
*************************************************************************************************
*******                              END OF USER INPUT                                  *******
*******                        PLEASE DO NOT EDIT BELOW THIS LINE                       *******
*************************************************************************************************;


*-----------------------
* 0- Setup Environment
*-----------------------;

/* Direct log to file */
filename runlog "&msoc./&dpid.&siteid._program_name.log";
proc printto log=runlog new;
run;

/* Assign libnames */
libname mscdm "&mscdm." access=readonly;
libname msoc "&msoc.";
libname dplocal "&dplocal.";
libname infolder "&infolder." access=readonly;


*-----------------------
* 1- Programming step 1
*-----------------------;
```

PROGRAMMING CHALLENGES IN A DISTRIBUTED DATA NETWORK

There is a uniqueness to programming in a distributed data network because, essentially, you are writing a program for:

→Someone else to run,

→In myriad computing environments that vary across sites (operating system, SAS version, SAS modules licensed and installed),

→That may have site-specific default SAS options or other restrictions,

→On various sizes of data to which the requesting programmer does not have direct access.

Programming in this kind of setting presents a number of challenges, including:

- Data partner sites maintain control over their computing environment, including available hardware, software and data storage options, so all programs must be written to execute on one of any three major platforms (Windows, UNIX, Linux), across different SAS versions and cannot assume that all sites have SAS modules installed outside of Base SAS,
- The term “big data” is relative; large at one site may translate to 500,000 records, while at another it is used to characterize a table containing 2+ billion records, so a programming technique that works well for a small table may not work well (or at all) for a big table,
- Data across all partner sites are held in different source systems, with different structures and formats, making the conversion from each of these disparate source systems to one common data model potentially inconsistent across all sites,
- A programmers’ lack of direct access to any sites’ data means several things:
    - The programmer must have data stored locally that is formatted to the common data model, that can be used for program development and testing purposes,
    - These local data likely are not representative of all data nuances and anomalies that may exist across all data partner sites in the distributed data network, so just because the programmer does not “see” something happening in their test data, they may have to proactively and defensively code to account for it happening somewhere in the network.

SPECIFIC PROGRAMMING CHALLENGES

This section discusses some specific examples of programming challenges and considerations commonly encountered in a distributed data network environment.

Operating system (*i.e.,* Windows versus UNIX)

	Path/file name case sensitivity?	**Path separator recognition**
Unix	Yes	Only '/'
Windows	No	'/' OR '\'

SAS options

Be mindful of SAS options and their values. ***Most importantly, know YOUR program's requirements for options, in order to run as intended***.

- MERGENOBY= NOWARN |WARN | ERROR
    - Specifies the type of message that is issued when MERGE processing occurs without an associated BY statement (NOWARN is the default),
    - There are reasons someone might want to perform a MERGE without a BY statement,
    - If one or more sites at which your program will run have changed that default option to be set to MERGENOBY=ERROR and your program does not account for that, that step of your program will fail.
- EXTENDOBSCOUNTER=NO | YES
    - Specifies whether to extend the maximum observation count in a new output SAS data file,
    - Important to note: a SAS data file that is created with EXTENDOBSCOUNTER=YES is incompatible with releases before SAS 9.3.
- VALIDVARNAME=V7 | UPCASE | ANY
    - Specifies the rules for valid SAS variable names that can be created and processed during a SAS session (V7 is the default),
    - Unexpected results can occur if a site has a setting other than the default.

Keep your log informative yet parsimonious

Be mindful of the level of information that gets output to the log, such as by making judicious use of input function format modifiers

- Are you aware that invalid data exist when attempting a character→numeric conversion? Do you want the log to reflect all instances of these?
    - There are reasons you may not want to see all of those messages to the log (e.g., not a real problem or you are going to deal with those in another way, so you don't want to clutter the log with messages like this)
    - Use format modifiers (? Or ??) to suppress 'Invalid argument' NOTEs to the log and/or prevent the automatic variable _ERROR_ from being set to 1 when invalid data are read:

```
value_n=input (value_c,4.);
value_n=input (value_c,?4.);
value_n=input (value_c,??4.);
```

Data size (i.e., programming to the lowest (or highest, in this case) common denominator)

- Select the most appropriate merge/join method that will work on all sizes of data, given available (and variable) resources: Disk-based (DATA step merge, SQL join) versus memory-based (hash object) methods
- Be aware of common big data 'best practices' with regards to efficiencies
    - Only keep variables that you need,
    - Use appropriate length statements (and only what you need) when creating new variables,
    - Consider splitting a very large file into two or more smaller pieces before working with it,
    - Remember that sorting a file can require up to 4 times the resources of the file you are sorting (e.g., a 25MB file may require 100MB of space in the work library),
    - "Pack it in, Pack it out":

- Actively manage your work library (via proc datasets) either at key places in your program but definitely at the end
- Actively delete (%symdel) global macro parameters you create, to avoid potential collision later on

"Defensive" coding

- What does it mean? A prospective approach to anticipating (and handling) potential data anomalies or coding logic issues,
- Why is it important? In a distributed environment, we typically do not have the luxury of "seeing" all potential data anomalies that could cause a program to fail, or, worse, cause a program to run to completion but produce unanticipated or incorrect results. Anticipating (and coding for) these issues could save time and resources, and ensure that your program is remaining true to its original intended purpose
- Some specific examples:
    - Check for duplicate keys prior to a merge and issue a custom warning to the log,
    - Check for missing or out-of-range values and issue a custom warning to the log,
    - Check the lengths of variables on a by statement prior to a merge to avoid warnings such as, "WARNING: Multiple lengths were specified for the BY variable pain by input data sets. This may cause unexpected results."
    - Utilize a final ELSE statement to capture any observations that do not meet the conditions of IF-THEN clauses

CONCLUSIONS

In a distributed data network, it is ideal –and I would argue critical—to have a central operations center whose job is to understand the data within the network, perform quality/characterization checks, guide requesters through appropriate use/interpretation of data, develop infrastructure for standard data access and analysis, and ensure development of efficient and high-quality analytic programs.

I have discussed Mini-Sentinel as a case study for a discussion of programming-related principals that are critical to any distributed data network: flexible, scalable and reusable programs built to access data across multiple sites and platform quickly, predictably, and repeatably, as well as policies and supporting infrastructure in place to protect data privacy.

I have also discussed some unique programming challenges and solutions that are typically encountered in a distributed data network environment. These examples and concepts are transferable as "best-practices" to any programming environment.

REFERENCES

- Adibhatla R, VazquezBenitez G, Becker M, Butani A. 2013. "Creation and Implementation of an Analytic Dataset for a Multi-Site Surveillance Study using Electronic Health Records (EHR) and Medical Claims Data". Proceedings of the 2013 Mid-West SAS Users Group Annual Conference, Columbus, OH. http://www.mwsug.org/proceedings/2013/RX/MWSUG-2013-RX06.pdf . Accessed 20 October 2013.
- Bredfeldt CE, Butani A, Padmanabhan S, et al. 2013. "Managing protected health information in distributed research network environments: automated review to facilitate collaboration". BMC Med Inform Decis Mak. 13:39.
- Brown JB and Platt R. 2013. "Distributed Research Networks: Lessons from the Field". The Learning Health System Summit, Washington, DC. http://healthinformatics.umich.edu/sites/default/files/files/u24/7.Brown_.pdf . Accessed 20 October 2013.
- Curtis LH, Brown JB, Platt R. 2014. "Four Health Data Networks Illustrate The Potential For A Shared National Multipurpose Big-Data Network." Health Affairs; 33(7): 1178-86.
- Food and Drug Administration Amendments Act of 2007, Pub. L. no. 110-85, Page 121 Stat. 944 (2007). http://www.gpo.gov/fdsys/pkg/PLAW-110publ85/html/PLAW-110publ85.htm . Accessed 20 October 2013.
- Gagne JJ, Glynn RJ, Avorn J, Levin R, Schneeweiss S. 2011. A combined comorbidity score predicted mortality in elderly patients better than existing scores. J Clin Epidemiol. 64(7):749-59.
- Hamilton J. 2012. "What Do You Mean, Not Everyone Is Like Me: Writing Programs For Others To Run". Proceedings of the 2012 SAS Global Forum, Orlando, FL. http://support.sas.com/resources/papers/proceedings12/229-2012.pdf . Accessed 20 October 2013.

- Langston R. 2009. “Scalability of Table Lookup Techniques”. Proceedings of the 2009 SAS Global Forum, Washington, DC. http://support.sas.com/resources/papers/proceedings09/037-2009.pdf . Accessed 09 Jun 2014.
- Nelson GS and Zhou J. 2012. “Good Programming Practices in Healthcare: Creating Robust Programs”. Proceedings of the 2012 SAS Global Forum, Orlando, FL. http://support.sas.com/resources/papers/proceedings12/417-2012.pdf . Accessed 03 September 2014.
- Psaty BM and Breckenridge AM. 2014. “Mini-Sentinel and Regulatory Science — Big Data Rendered Fit and Functional”. N Engl J Med; 370:2165-7.
- Sherman T and Ringelberg A. 2013. “Defensive Programming and Error-handling: The Path Less Travelled”. Proceedings of the 2013 PharmaSUG Conference, Chicago, IL. http://www.pharmasug.org/proceedings/2013/TF/PharmaSUG-2013-TF24.pdf . Accessed 03 September 2014.
- Toh S, Reichman ME, Houstoun M, et al. 2013. “Multivariable confounding adjustment in distributed data networks without sharing of patient-level data”. Pharmacoepidemiology and Drug Safety; 22(11):1171-7.
- Toh S, Shetterly S, Powers JD, Arterburn D. 2014. “Privacy-preserving analytic methods for multisite comparative effectiveness and patient-centered outcomes research.” Med Care; 52(7):664-8.

ACKNOWLEDGMENTS

The author would like to extend a special thank you to all staff at the Mini-Sentinel Operations Center (MSOC). The expertise and ideas of my colleagues were invaluable to the development of this paper.

CONTACT INFORMATION

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

Jennifer R. Popovic, DVM, MA

Harvard Pilgrim Health Care institute/Harvard Medical School

133 Brookline Ave, 6th Floor

Boston, MA 02215

617.509.9811

jennifer_popovic@harvardpilgrim.org