You've Got Data! : Using SAS® from Data Receipt to Reporting

Phil Vecchione, Cognigen Corporation, Buffalo NY

ABSTRACT

In a fast-paced, pharmaceutical data analysis environment, the transfer of data needs to be quick and accurate. A number of steps are involved in coordinating data transfers. Data must be received, processed, and verified; proof of receipt needs to be communicated to the sender; and the internal team needs to be updated about the newly arrived datasets. These steps can be time consuming, prolonging the time to analyze and submit data to the FDA.

SAS® can be used to automate each of these steps, decreasing the time for data transfers. Macros can be written to access the Data Dictionary Tables to assemble and report on metadata. Additionally, macros utilizing the Proc Contents OUT option can be used to look at libraries of data, as well as individual tables, and identify missing variables or datasets. ODS HTML output can be used to create web-based reports to keep track of received data. Furthermore, SAS generated email can be used to update internal team members. Utilizing a combination of the above programming strategies will make the data receipt process more efficient, allowing for data analysis to start sooner, and contributing to reduced costs in the development of new drugs.

INTRODUCTION

Data transfer should be a transparent process. That is to say, in the larger scheme of planning an analysis, one should not concern oneself with the steps taken in the receipt of new data. Data should be sent from a site to your company and should be available to use when it arrives. Senders should know that their data has been received correctly, and internal users should be informed as soon as the new data arrives. In a good system, data transfers should be taken for granted.

A number of steps occur during data transfer, and critical information needs to be communicated to the sender and to the internal team. A tight communication loop between sender and internal team ensures that problems can be identified quickly and resolved shortly after they are detected. Often problems that occur are not in the data transfer process, but rather within the data itself. Incomplete data can include missing variables, missing datasets, or even missing observations. Problems that can occur within the transfer are often communication issues where internal members are unaware of new data or unable to locate archived data. A tight communication loop allowing for faster and less problematic data transfers is crucial. At Cognigen we strive to establish this communication loop with each of our data transfers.

Cognigen provides data analysis and consulting services for the pharmaceutical and biotechnology industries during clinical development of new medicines. The population-based statistical and pharmacokinetic/pharmacodynamic modeling performed at Cognigen requires targeted data assembly of multiple datasets within and across clinical trials. As a data analysis firm, we receive data from a number of different sponsors for various projects. Each of these sponsors has their own data management systems and database structures. Many sponsors also contract with laboratories or CROs that generate data and transmit it directly to Cognigen periodically during the course of the study.

Typically we receive clinical data as SAS transport files containing 20 or more datasets, comprising the various database tables used by the sponsor. CROs will often send Excel® files instead of SAS datasets. During the course of a given project we may receive updates of data several times. At each transfer we need to make sure that any data issues are identified and communicated to our sender as quickly as possible and that the internal team is aware of new data and knows how to locate that data within our data storage model.

To facilitate these types of checks and communications, we have developed a number of SAS programs that utilize various features of the SAS language. These programs have helped to reduce the time it takes to perform the data transfer process and have increased our ability to identify potential problems as new data arrives rather then during the data assembly or analysis phase.

This paper addresses the overall data receipt process, some SAS features that can be utilized to automate the collection of metadata, and then demonstrates how they are put together to create a faster and more information rich process.

DATA TRANSFER PROCESS

To fully understand how SAS can facilitate the data transfer process, it is best to start with the definition of the process. In the simplest sense, data transfer is the movement of data from one source to another. The sender is therefore the person or entity that sends the data, and the recipient is the person or entity that receives the data.

Typical data transfers should include the following steps: Receipt of Data, Processing of Data, External Confirmation, and Internal Confirmation.

RECEIPT OF DATA

The receipt of data is simply the arrival of data to the recipient. This data can come by conventional mail, email, or FTP. It may be compressed, encrypted or both. Compressed data can be in the form of data compression programs such as ZIP or GZIP or it can be in the form of SAS Transport Files. Received data has to be moved to some file system where it can be processed.

The only types of errors that occur at this stage have little to do with the actual data, but rather in how it was sent, for example, files may have been corrupted.

PROCESSING OF DATA

At this point the newly arrived data is processed. Part of this process is the SOPs that are used to handle new data and the use of programs to transform the data into a format that can be used for data assembly and analysis. Often part of this phase is to load the data into a database system (Oracle®, Ingres®, etc.), or to transform this data into SAS datasets. The overall goal at this phase is to transform the data into a format that is useable by others within the company.

This is the stage where the data content errors are detected. The sender may not have sent all the datasets required, datasets may be missing crucial variables, or there may be missing observations. The absence of any of the types of data described above may impede data analysis or may stall it completely, depending on the severity of the discrepancy.

EXTERNAL CONFIRMATION

Once data has been received and transformed, it is common to inform the sender that the data has been received. In the external confirmation it is good practice for the recipient to give control totals for the data that has been received. This would include the number and names of the datasets received, as well as number of observations and variables for each dataset. This allows the sender to verify that the data was received intact.

Often errors in sent data can be identified at this phase, when the sender reviews the external receipt.

INTERNAL CONFIRMATION

Internal confirmation is the process by which the other members of the recipient's company are informed of the receipt of new data. This can take the form of updating a database, sending out an email, publishing an HTML report, or some combination of these and other techniques. The goal at this phase is to make the other members of the recipient's company aware of new data, so that they can take appropriate action.

Documentation is very important at this point. A good data transfer system employs a data receipt log. This log can be a flat file, a SAS dataset, even a database. The log contains metadata about the transmission of data, including what files were sent, their contents, etc. This log is a valuable resource for determining what data has been received and when it was received.

Communication errors could occur at this stage. If the team is not informed about the newly arrived data, the analysis could be delayed until it is located, or the data is skipped and not used in the analysis.

SAS TECHNIQUES

SAS has a number of metadata and reporting tools available that can be utilized to view information about datasets and libraries. Most are easy to program and provide useful information during the Data Receipt process. Each technique is described below, including example programs to highlight each technique.

DATA DICTIONARY TABLES

These tables are a collection of SAS views that contain information about SAS libraries and datasets. They are the best source of metadata. They are located in the SASHELP library, and can be accessed with PROC SQL or by using a DATA step.

The VMEMBER table contains some basic information about each dataset in a library. Its two most useful data elements are the Engine Name (engine) and the Path Name (path). The Engine Name will tell you the dataset version, and the Path will give you the physical location of the dataset on your file system.

The VTABLE table contains more information about individual datasets, including Number of Observations (nobs), and Number of Variables (nvar). This is great information that can be summarized about each dataset in a library.

At Cognigen, we employ a macro called datasum to create a custom report on a library of data. Datasum exclusively uses the data dictionary tables to accomplish this.

Datasum

MACRO CALL: %datasum(whatlib=libref, mode=[C,S])

FUNCTION: Displays information about the specified library, creates a list of all the datasets in a given library, and lists the number of observations and variables for each dataset.

MECHANISM: The macro variable whatlib refers to the library to be analyzed. The macro variable mode has two settings, C for creation and S for summary. In creation mode, the creation and modification date for each dataset are suppressed. This mode is used when datasets are first created and the creation date and modification date are the same. In summary mode the dataset creation date and modification date are shown, and if the two dates differ, a text flag is displayed in the output to indicate that the dataset has been altered.

The VMEMBER view is used to provide information about the library, and the VTABLE view is used for creating a summary for each dataset within the library.

The Library Summary is created by using PROC SQL to retrieve the information from VMEMBER. This PROC SQL uses the flow option so that the directory path of the library wraps rather then skews the output. Grouping by libname and adding the count statement reduces the info in VMEMBER from individual tables to information about the library.

The Dataset Summary is created by accessing VTABLE using a data step with a where clause for the library of interest. The Dataset Summary table is then printed using a PROC PRINT. Using both PROC SQL and a DATA step, the program is able to summarize information about the library and the individual datasets within the library. Excerpts of the code are shown below:

```
/* Create Library Summary from DDTs */
title "Library Summary from library: &whatlib";
proc sql flow;
    select  libname, engine, path,
            count(1) as Datasets
    from sashelp.vmember
    where libname=upcase("&whatlib")
    group by libname, engine, path;

/* Create Data Set Summary Table from DDTs */
   data work.dsinfo;
        set sashelp.vtable
        (keep= libname memname memtype nobs
                nvar memlabel crdate modate);
        where libname=upcase("&whatlib");
   run;

 /* Print out Data Set Summary */
proc print data=work.dsinfo label noobs;
     var memname memtype nobs nvar memlabel;
     title 'Data Set Summary';
run;
```

EXAMPLE OUTPUT: The output that datasum creates for a fictitious library named GENLIB, looks like this:

Library Summary from library: genlib

Library Name	Engine Name	Path Name	DATASETS
GENLIB	V612	/doc/temp/data/	3

Data Set Summary

Member Name	Member Type	Number of Observations	Number of Variables	Dataset Label
CONC_01	DATA	201	15	PK Data
SERA_01	DATA	202	12	Site 01
SERA_1_1	DATA	201	10	Site 09

PROC CONTENTS OUT OPTION

PROC CONTENTS is a very useful tool for describing individual datasets. The information contained in PROC CONTENTS is the same information that is contained in the Data Dictionary Tables. In the case of PROC CONTENTS this data is pulled together by SAS for a specific purpose, such as summarizing the metadata about a given dataset. If you are not familiar with the Data Dictionary Tables, PROC CONTENTS is a great source of metadata.

PROC CONTENTS has an OUT option that will create a dataset that contains all the information from the PROC CONTENTS procedure. This dataset can then be manipulated to obtain information about the variables for each dataset.

We have created a suite of macros designed to rapidly summarize the structure of the incoming data, determine if variables have been added or removed, locate specific variables by keyword, and determine missing data in key datasets. We refer to this suite as the VAR family. These macros, when added to a program written to unpack a transport file or to load data from another file format, provide concise output, which enables the programmer to identify potential problems with the data. Each member of the VAR family is described below.

VARLIST

MACRO CALL:

%varlist (whatlib=*libref*, wherelib=*libref*, print=(0 | Null))

FUNCTION: To create a list of all the variable names within a given library and to list all cases where a variable exists in more than one dataset.

MECHANISM: Varlist runs on a specified library of data that is defined in the whatlib parameter. It creates a list of all the datasets in the specified library and then begins to assemble a dataset that contains the memname (dataset), variable name, type, length, label, format, and informat.

Once the dataset is assembled, a list of the dataset is printed using PROC PRINT. Then a second list is created using a DATA NULL step that prints a summary for each variable that is contained in multiple datasets. This list contains the variable name, label, and a list of all the datasets in which the variable is contained.

Varlist has two other parameters that can be used by other programs and are used by the other members of VAR family. The first is the wherelib parameter. This parameter is used to determine the library where the metadata dataset will be created. Typically WORK is the specified directory, so no permanent dataset is created. If a permanent dataset is desired, then define another libref, and varlist will output the dataset to that library.

The second parameter is the print parameter. This parameter is optional. When it has the value of 0, varlist does not create the two lists described above. When the parameter is not specified or given any value other then 0, the lists are printed. This parameter is used by the other VAR family macros so that varlist creates the necessary datasets but does not create a text output when called.

Varlist assembles its list of variable names by using PROC CONTENTS with the OUT option. A loop is created that processes the code below for each dataset.

```
proc contents data=&whatlib..&&ds&i noprint
        out=work.tmpvar
        (keep=memname name type length
              label format informat);
run;
```

Each iteration of the loop creates the PROC CONTENTS dataset for a dataset in the library of interest. This dataset called tempvar (temporary variable list) and then appends it to a dataset called mvarlst (master variable list). The dataset mvarlst is then sorted and printed to create the first list containing all the variables in the specified library.

After printing the complete list of all variables, the macro takes the mvarlst dataset and removes all records that only appear once, and prints the variables that appear in more than one dataset by using a DATA NULL step. The code that creates this listing is shown below:

```
data work.multvars;
    set &wherelib..mvarlst;
    by name;
    if first.name and last.name  then delete;
run;

data _NULL_ ;
   %if &print ne 0 %then %do;
         set work.multvars;
         by name;
         file print;
         bline=repeat ('-', 30);
         if first.name then do;
            put //@5 'Variable Name: '
                  @22 name
                / @5 'Label: '
                  @13 label;
            put @4 bline;
            put / @5 'Data Sets: '
                  @17 memname;
         end;
         if not first.name then do;
              put @17 memname;
         end;
   %end;
run;
```

EXAMPLE OUTPUT: When Varlist is run on a library called samplib, the output is shown below:

```
      Complete list of Variables from:
                    SAMPLIB

MEMNAME  NAME    TYPE LENGTH  LABEL         FORMAT

BLODSAMP DAY         1    8   STUDY DAY
DRUGADMN DAY         1    8   STUDY DAY
EXAM     DAY         1    8   DAY
MEAL     DAY         1    8   STUDY DAY
QUESTION DAY         1    8   STUDY DAY
VITALS   DAY         1    8   DAY
FORMATS  DEFAULT     1    5   DEFAULT
EXAM     DESCR1      2    40  DESC1
EXAM     DESCR2      2    40  DESC2
DEMOG    DOB         1    8   DATE OF BIRTH  DATE

Variable Name:   BIRTHD
Label:  Date of Birth
------------------------------
Data Sets:  DEMOG
            SUMMARY
```

It can be seen that the variable birthd is found in both the demog and summary datasets.

As stated above, VARLIST is called by the other members of the VAR family to create a dataset of variable information, so that the other macros can do different types of metadata analysis. A shorter description of those macros are given below:

VARFREQ

MACRO Call: %varfreq (whatlib=*libref*, thr=[0.01-1.0])

FUNCTION: To summarize from a list of variables in a library the variables that appear in a specified percentage of datasets and to create a summary list of all known formats for that library.

MECHANISM: Varfreq is an extension of Varlist. It was designed to create more directed and summarized output from the data

generated by Varlist. It has replaced Varlist as the macro we use to summarize libraries of data. While Varfreq may have replaced Varlist for reporting, it still starts with a macro call to Varlist, with the print=0 option. Varlist creates the dataset of all variable names within the specified library from the whatlib parameter.

Then PROC SQL is used to create a summary dataset from the Varlist output dataset. This new dataset contains a list of each variable in the library along with the number of datasets in which that variable is present. That number is then expressed as a percentage of the total number of datasets in the library. The percentage calculated is then evaluated against the threshold parameter (thr), and the observation is deleted if the percentage is less then the value of the threshold. This creates a dataset of variables that are present in a specified percentage of datasets. The final dataset is then listed by using PROC PRINT.

A second dataset of variable names is created based on the datasets that were above the threshold. The dataset is then printed using PROC REPORT, displaying the variable name and a list of all associated datasets in which that variable is located.

Finally, a simple PROC FREQ is done on the original Varlist dataset to generate a table of all the formats used in the library.

EXAMPLE OUTPUT: The example below shows a typical output from the Varfreq macro, exemplifying all three listings described above.

Variable Frequency For testlib
Threshold= 0.75

Variable Name	Variable Desc.	No. of Tables	Pct. of Tables
INVNUM	Inv. Number	8	100%
PT_NO	Patient No.	8	100%
STUDY	Study Name	8	100%
VISIT_ORG	Visit Name	7	88%
VISIT_DT	Visit Date	7	88%
DOB	Date of Birth	7	76%

Most Common Variable Summary
From Library testlib

Variable Name	Variable Label	Pct. Table	Data Sets
VISIT_ORG	Visit Name	88%	DEMO
			EKG
			LABS
			CONMED
			PK
			PKPKD
			VITALSGN

Format Frequency for Library: testlib
The FREQ Procedure
Variable Format

FORMAT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	42	22.95	42	22.95
$	120	65.57	162	88.52
YSN	21	11.48	183	100.00

VARCOMP

MACRO CALL: %varcomp (newlib=*libref*, oldlib=*libref*)

FUNCTION: To compare the variable lists from two libraries of data and to determine which variables have been added, which have been removed, and which are the same.

MECHANISM: Varcomp starts by creating datasets of metadata from the two libraries. This is done by calling varlist with the print=0 parameter, twice. Once the datasets have been made, they are then merged by memname and variable name and put into three datasets using the IN option: newvar, lostvar, and merglst.

The newly created list begins with a summary of the number of variables in each of the three datasets followed by a list of newvar and lostvar.

EXAMPLE OUTPUT: The example below shows a typical output from the varcomp macro where there are differences in the variables between two libraries.

VarComp Summary Report

Number of Variables that are the same: 1128

Number of Variables that are new: 9

Number of Variables that are lost: 2

Variables Not In Previous Data Set
(New Variables)

OBS	MEMNAME	NAME	LABEL
1	ADE	DRUG	
2	ADE	PAGEDONE	PAGE WAS DONE
3	DOSTUDY	DRUG	
4	EOS	DRUG	
5	MED_HX	DRUG	
6	OTHMED	DRUG	
7	SMEDACC	DRUG	
8	VITAL	DRUG	
9	VITAL	PAGENO	PAGE NUMBER

N = 9

Variables In Previous Data Set but Not In New Data Set
(Missing Variables)

OBS	MEMNAME	NAME	LABEL
1	ADE	PTINIT	PATIENT INITIALS
2	OTHMED	PTINIT	PATIENT INITIALS

N = 2

VARSCAN

MACRO CALL:
%varscan (whatlib=*libref*, keyword=*text*, field=[N,L,B])

FUNCTION: To search a list of variable names and labels in order to locate variables that match the keyword.

MECHANISM: Varscan starts by creating a dataset of metadata by calling varlist with a print=0 parameter. Then the dataset is pared down by using a where statement which uses both the sounds-like and like options to find potential matches to the keyword that is specified. The keyword has to be a single word.

The field parameter allows the user to specify which fields varscan should search. A value of N searches only the variable name field. A value of L searches only the variable label field. A

value of B searches both the variable name and label fields. The default setting is B, and if the field parameter is not included, the search is performed on both the variable name and label fields.

EXAMPLE OUTPUT: The example below shows a search for both variable names and variable labels that contain the keyword "dose."

```
       B: Variable Name Search for Labels and
          Variables that sound like: dose
                From Library: lib018

Obs    MEMNAME     NAME     LABEL

1      CONMED      DOSE     DOSE
2      DRUGADMN    DOSE     COMPOUND DOSAGE

                        N = 2

       B: Variable Name Search for Labels and
            Variables that contain: dose
                From Library: lib018

Obs    MEMNAME     NAME     LABEL

1      CONMED      DOSE     DOSE
2      DRUGADMN    DOSE     COMPOUND DOSAGE
3      CONMED      UNITS    DOSE UNITS

                        N = 3
```

Each mechanism delivers slightly different results. That is why it is best to use both mechanisms to increase the chance of locating the variable of interest.

VARSUM

MACRO CALL: %varsum (whatlib=*libref*, dset=*dataset*)

FUNCTION: To summarize a given table and determine for each variable the number of missing and non-missing values.

MECHANISM: Varsum creates a dataset of variable names from the dataset specified in the dset parameter. Using that metadata list as the list of variables in the table, varsum begins to count the blank and non-blank values for each variable. It does this by taking each variable one at a time and for each record determining if the value is missing or not missing then summing these counts. The metadata for the variable and the number of missing and non-missing values are then transferred to a new dataset as a single record.

When all the variables have been scanned, the final dataset of metadata and counts is then output as a list.

EXAMPLE OUTPUT:

```
   Variable List for dataset:  lib1.conc

Num Var     Label           Type      N  MISS
1  PROT     Protocol        Character  884    0
2  CENTER   Center #        Character  884    0
3  SUBJECT  Subject #       Character  884    0
4  SAMPLE   SampleType      Character  884    0
5  DRDTRAW  Draw Date       Character  884    0
6  DRAWDT   Draw Date (SAS) Numeric    884    0
7  VISIT    Visit Desc      Character  884    0
8  ANALYT   Analyte         Character  884    0
9  LAB      Analytical Lab  Character  884    0
10 EXTID    External ID #   Character  884    0
11 CONC     Concen.         Character  873   11
12 UNITS    Units           Character  884    0
13 COMENT   Lab Comments    Character  17   867
```

ODS HTML

Most companies have an Intranet site. This Intranet site contains pages of useful information for the employees of the company. The Data Receipt Log should be one of the pages of this site. A centrally located Log allows various team members to determine what data has arrived and when. It also gives them a historical log of all data that has arrived.

ODS, or Output Delivery System is a feature in SAS that creates formatted output that can be used by a variety of programs. In our discussion we will focus on how it is used to create HTML pages from SAS Datasets. While SAS/IntrNet® can be used to create a more interactive resource, a very simple resource can be created using some familiar PROCs and ODS HTML output. The best part about ODS HTML is that no HTML experience is necessary to create web ready reports; SAS will do all the work for you.

There are a number of predefined HTML color schemes to use, they are referred to as Styles. These styles are standard and are supplied with SAS version 8 and can be used without any HTML experience. With a little understanding of SAS and HTML you can modify those templates and create your own color schemes. The program we use in house to create our Data Receipt Log uses its own custom template, called CogD that we have modified to display our corporate colors.

In it's simplest form, a program to create HTML output of a dataset would look like this:

```
ods html file='pklist.html';
ods listing close;
ods html style=CogD;

proc print data=work.webrpt label;
    var sponsor drug study filename arrive path;
run;

ods html close;
```

The output from the Proc Print would then be transformed into a HTML file named, pklist.html. The output of which would look something like this:

The SAS System

Obs	Sponsor Code	Study Drug	Protocol	File Name	Arrival Date	Directory
1	Drug Co.	DrugX	Study-101	101.XPT	22JAN2001	/doc/drugco/drugx/dataorig/study-101/22jan2001
2	Drug Co.	DrugX	Study-102	102.XPT	22JAN2001	/doc/drugco/drugx/dataorig/study-102/22jan2001
3	Drug Co.	DrugX	Study-102	pk summary 5-18-01.xls	23MAY2001	/doc/drugco/drugx/dataorig/study-102/11apr2001>
4	Drug Co.	DrugX	Study-103	103.XPT	22JAN2001	/doc/drugco/drugx/dataorig/study-103/22jan2001
5	Drug Co.	DrugX	Study-103	blood.xpt	21MAY2001	/doc/drugco/drugx/dataorig/study-103/21may2001
6	Drug Co.	DrugX	Study-103	bld.xpt	23MAY2001	/doc/drugco/drugx/dataorig/study-103/23may2001
7	Drug Co.	DrugX	Study-104	104.XPT	22JAN2001	/doc/drugco/drugx/dataorig/study-104/22jan2001
8	Drug Co.	DrugX	Study-105	105.XPT	22JAN2001	/doc/drugco/drugx/dataorig/study-105/22jan2001
9	Drug Co.	DrugX	Study-108	SAS Data Sets (Various)	25OCT2001	/doc/drugco/drugx/dataorig/study-108/25oct2001
10	Drug Co.	DrugX	Study-108	final pk.xls (pk Results (reported))	25OCT2001	/doc/drugco/drugx/dataorig/study-108/25oct2001

You can create a more elaborate report by using PROC REPORT that includes breaks after each sponsor and other formatting. This report can be further enhanced by using an option in ODS HTML that will create a web page with frames that will have a table of contents on the left portion of the page. This table of contents will allow the user to jump to the section of the Data Receipt Log that is most important to him/her. The code below shows the changes that would need to be made to create this framed webpage:

```
ods html file='datarec_body.html'
         contents='datarec_contents.html'
         frame='datarec_frame.html'
         headtext= '<TITLE> Cognigen Data
                              Receipt </TITLE>';
ods listing close;
ods html style=CogD;

proc report data=work.webrpt nowindows wrap
           headline headskip;
     columns study arrive filename mediatyp
             filetype desc confordt path
             sendnote;
      title 'Cognigen Data Receipt Report';
      title2 "Posted On: &outdt";
      footnote;
      by sponsor drug;
      define study/order width=15 spacing=1;
      define arrive/ width=9 spacing=1;
      define filename/ width=50 spacing=1 flow ;
      define mediatyp/  width=20 spacing=1
                        flow left;
      define filetype/  width=20 spacing=1
                        flow left;
      define desc/ width=50 spacing=1 flow left;
      define confordt/ width=9 spacing=1  left;
      define path/width=100 spacing=1 flow left;
      define sendnote/ width=50 spacing=1
                       flow left;
      break after study/ dul ;
run;
ods html close;
```

The contents of the PROC REPORT step will be contained in datarec_body.html. The table of contents as determined by the break option in the PROC REPORT step will be contained in datarec_contents.html, and the frame page that will incorporate both of these web pages is named datarec_frame.html. The flow option was used a number of times on longer text fields to allow the text to wrap within the confines of the defined cell rather then letting it skew the report.

When the program is run, the HTML output looks like this:

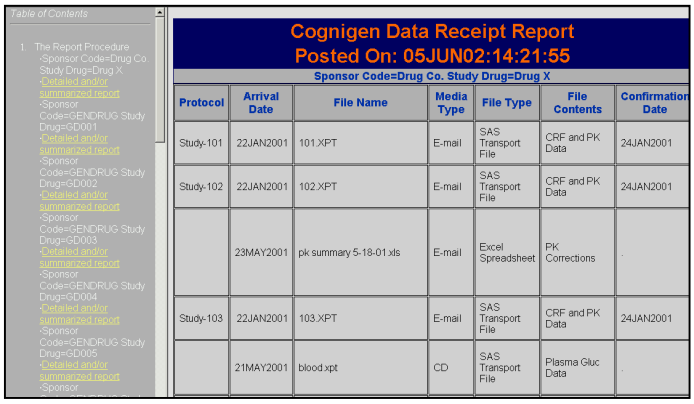

Table of Contents

1. The Report Procedure
   - Sponsor Code=Drug Co. Study Drug=Drug X
   - Detailed and/or summarized report
   - Sponsor Code=GENDRUG Study Drug=GD001
   - Detailed and/or summarized report
   - Sponsor Code=GENDRUG Study Drug=GD002
   - Detailed and/or summarized report
   - Sponsor Code=GENDRUG Study Drug=GD003
   - Detailed and/or summarized report
   - Sponsor Code=GENDRUG Study Drug=GD004
   - Detailed and/or summarized report
   - Sponsor Code=GENDRUG Study Drug=GD005
   - Detailed and/or summarized report
   - Sponsor

Cognigen Data Receipt Report
Posted On: 05JUN02:14:21:55
Sponsor Code=Drug Co. Study Drug=Drug X

Protocol	Arrival Date	File Name	Media Type	File Type	File Contents	Confirmation Date
Study-101	22JAN2001	101.XPT	E-mail	SAS Transport File	CRF and PK Data	24JAN2001
Study-102	22JAN2001	102.XPT	E-mail	SAS Transport File	CRF and PK Data	24JAN2001
	23MAY2001	pk summary 5-18-01.xls	E-mail	Excel Spreadsheet	PK Corrections	
Study-103	22JAN2001	103.XPT	E-mail	SAS Transport File	CRF and PK Data	24JAN2001
	21MAY2001	blood.xpt	CD	SAS Transport File	Plasma Gluc Data	

You can see that the left column now has links to take you to each section of the report. The right side is the PROC REPORT. By using the break option, the study values do not repeat and using a BY value in the PROC REPORT the web page is broken up by Sponsor and Drug.

We use SAS ODS HTML to post our Data Receipt Log in a format very similar to the output shown above. We use PROC REPORT to create a custom report with breaks between every sponsor/drug combination, and then use the frame option of ODS HTML to create a table of contents page for the HTML output. This page is then posted to our Intranet site, and made available for everyone to use.

The value of ODS HTML is that this type of report can be created with very little code, and when posted to an intranet site is a valuable resource for keeping track of data.

EMAIL VIA SAS

The SAS system can generate and send its own email using a modified filename statement. This email is then generated automatically by the program and sent to the recipients defined in the program, as the program is run.

This is a great way to disseminate critical information in a timely manner. The program can be a macro that is triggered at some time during the data transfer process, or it can be a program that is automatically run by the job manager in your operating system, such as the UNIX® CRON command.

To program SAS to send an email, there is a special filename statement that is used to define for SAS that this file references an email. A DATA NULL step is then used to report out the information to be sent. This DATA NULL step uses the email filename in the File option; the records are printed out as defined in the body of the DATA NULL step. When the step is completed an email is generated and sent.

We employ a program that makes a weekly mailing of all data received for that week to the directors of each department, informing them of the data that came in for the week. Excerpts of the program we use are shown below:

```
filename emailout email 'pv@cognigencorp.com'
         to=("pv@cognigencorp.com"
             "rtz@cognigencorp.com"
             "gdg@cogngencorp.com"
             "dn@cognigencorp.com");
```

This statement defines that emailout is to be an email that will be sent to everyone listed in the TO option. The syntax of the statement requires that a single address be included after the email statement, but that address is overridden by the TO option. The body of the email is handled by this code:

```
data _null_;
     set work.webrpt end=last;
     by sponsor drug study;
     rptdt= put(&begwk, date9.);
     file emailout
          Subject= 'Weekly Data Receipt Report';
     bline= repeat('-',60);

     if _n_=1 then do;
         put /@20 'Cognigen Weekly Data
                   Receipt Report';
         put /@30  "For:"
         put  @35  rptdt ;
     end;
    if First.sponsor and first.drug and
          first.study then do;
        put   /@4 ' ';
        put   /@4 ' ';
        put   /@4 bline
              /@5 'Sponsor'
               @15 'Drug'
               @40 'Study'
              /@4 bline;

        put  /@5 sponsor
              @15 drug
```

```
             @40 study
            /@4 bline;

       put /@5 'File Name'
            @40 'Description'
           /@5 '---------'
            @40 '-----------';

       put /@5 filename
            @40 desc;
    end;
  if not first.sponsor and not first.drug and
           not first.study then do;
       put /@5 filename
          @40 desc;
     end;

     if last then do;
        put @5 ' ';
        put @5 ' ';
        put /@20 'This report brought
                 to you by SAS.';
end;
run;
```

When this program runs, it generates an email with a body like this:

```
     Cognigen Weekly Data Receipt Report
                For:02JUN2002
-----------------------------------------------
Sponsor   Drug                    Study
-----------------------------------------------
Drug Co.  Drug X                 Study 101
-----------------------------------------------
File Name                       Description
---------                       -------------
Patient 103                   CRF Data(various)
Patient 104                   CRF Data(various)
Patient 105                   CRF Data(various)

-----------------------------------------------
Sponsor   Drug                    Study
-----------------------------------------------
GENDRUG   GD001                   02-14
-----------------------------------------------
 File Name                        Description
---------                         -----------
GD001_02-14_AGE.xls              Patient Ages


       This report brought to you by SAS.
```

This makes a great summary report that can be generated and automatically sent from an existing Data Receipt Log via SAS.

WHERE PROCESS MEETS PROGRAMMING

The true power of SAS comes when programming is integrated with process. This section will pull together the data transfer process we have discussed along with the programming techniques to create a more automated process. It will be discussed in the context of Cognigen's data transfer practices.

RECEIPT OF DATA

There is not much automation that SAS can provide at this level. Basically, the sender has sent data to the recipient. That data is not yet available to be analyzed by SAS.

PROCESSING OF DATA

This is the most crucial stage to detect data content errors. At this stage, SAS can interface with the data via a SAS dataset. As a SAS dataset there is now metadata associated with each dataset. This metadata can be accessed using the Data Dictionary tables or the PROC CONTENTS OUT option.

We use %datasum to create a listing with metadata about the library (from VMEMBER) and the individual datasets (from VCOLUMNS). This is especially true if the transmission of data was in the form of a SAS Transport file and contains 20 or more datasets. Often we will receive control totals from our senders, telling us how many observations are in each dataset. In the past we would take that list and compare it to the log file of the program that unpacked the data. The log file is often cluttered with additional notes, and picking out the line for each dataset unpacked could be tedious. With %datasum we are able to produce a single, easy to read table that makes verification easy and quick.

At this stage of data transfer, it is important to ascertain if any data is missing in the form of missing datasets or missing variables. Since we often get our data in the form of transport files, we are concerned about what variables are contained in the datasets provided and how complete is each dataset. If there are missing variables or too many missing observations, we may not be able to properly assemble and analyze the data. By using the VAR family of macros we can quickly look at the metadata of all the datasets in a library and determine:

- Missing variables by using VARSCAN to look for keywords such as dose, conc, age, sex, race.
- Newly added or deleted variables using VARCOMP to compare the data from the previous transmission to the current transmission.
- Merge Variables by using VARFREQ to find all the variables that appear in a significant number of datasets.
- Types of formats used by using VARFREQ to generate a complete listing of formats used in the library.
- Completeness of data by using VARSUM to look at the number of missing and non-missing values in a given table.

Using all these tools on newly arrived data allows us to identify problems as we receive new data, rather than during the data assembly phase. Problems that are found at this stage can be communicated to the sender as well as the internal team. This tight turnaround allows the sender to correct the problems and re-transmit quickly.

EXTERNAL CONFIRMATION

It is important to communicate back to the sender exactly what was sent. In the case of missing or incomplete data, problems can often be found and corrected if the sender is aware of exactly what was received. In many cases the error is that the wrong dataset was sent. In this case we use %datasum again to provide an easy to read table of all received datasets along with their number of observations and variables. The output from %datasum is pasted into a boilerplate receipt email and sent to the Sender.

If any errors are found in any of the datasets using the VAR macros we will communicate those queries to the sender. For instance, we once received locked PK concentration data from a CRO. This data was to be used at once in a data analysis. The Excel® file of concentrations was processed and transformed into a SAS® dataset and a varsum call was added to the end of the program. From the output of varsum, it was shown that 11 records had missing concentration values. Because this data was locked and our analysis of it was about to proceed, we contacted the sender one hour after the data arrived and confirmed with them

that the missing values could not be recovered.

Had we not found the missing variables with varsum, they would have been found in the middle of data assembly, and would have delayed data assembly until the above confirmation was made.

INTERNAL CONFIRMATION

As important as it is for the sender to know when data has been received, it is equally important for the internal team members to know that data has arrived, and to give them any additional information about the data based on the findings during the processing phase.

At Cognigen we send a receipt to the sender and a Data Summary Report to the internal team members. The Data Summary Report consists of output from %datasum as well as output from the VAR family macros, typically %varsum and %varcomp. This report provides information about the library, as well as the individual datasets, and the variables within the datasets. The report is easy to read and rich in information. It is easy to assemble this report because the macros do all the work within the program that unpacked or transformed the data.

In addition to our Data Summary Report, we post our Data Receipt Log to our Intranet site. That Data Receipt Log is produced using ODS HTML. This central source for our Log allows all internal company users to look up what data has been received, giving them filenames, descriptions, and locations of data.

In order to keep Management informed of what data arrives weekly, we used the Email filename to create a summary report of all the data that has been received for the week. The program that generates this message has been put into our CRON program and every Friday afternoon the program is run and the report is sent without any human intervention.

CONCLUSION

The data transfer process is an important part of any CRO or departments' activities. It is a multi-step process that requires a combination of procedure and programming. SAS can be used to create programs that can automate the collection and analysis of the metadata. From this analysis of metadata, potential problems that could delay data assembly and analysis can be identified and resolved.

SAS can also be used to facilitate the reporting of received data. Web pages of data receipt information, in the form of SAS datasets, can be created easily using ODS HTML. Email containing information from SAS datasets can also be sent out from SAS, creating an automated system that increases the overall level of communication within a team or department.

Data Transfer should be a transparent process, but that does not mean that it should be a silent process. There is a wealth of information that can be gained by looking at the metadata. This information can be harvested by the programmers assembling the data to speed up assembly and preparation of the data for analysis.

Because of the length and cost of the Drug Development process, any system that contributes to a reduction in time for data analysis has an impact in the time to bring a new compound to market. By decreasing the time invested in data transfer and increasing the information that can be gained from data transfer, data assembly can occur more quickly. This leads to more time for data analysis. It is during data analysis where discoveries are made and information based decisions are made.

REFERENCES

Vecchione, Phil (2002), "Rapid Assessment of Data Set Structures", Proceedings from the 2002 Conference of the Pharmaceutical Industry SAS Users Group, Cary, NC: SAS Institute Inc., 177-180.

ACKNOWLEDGMENTS

The Cognigen data transfer process and programs would not be possible without the trust and guidance of Kathy Reitz, the Manager of Data Management, who let me derive both the process and programming, which has formed our data transfer system.

In addition I have to thank the SAS programmers in the Data Management department who answered endless questions and reviewed program after program as this system evolved. Without their knowledge and feedback this system could not exist.

CONTACT INFORMATION

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

Phil Vecchione
Cognigen Corporation
395 Youngs Road
Buffalo, NY 14221-5831
Work Phone: 716-633-3463 (ext. 260)
Fax: 716-633-7404
Email: phil.vecchione@cognigencorp.com
Web: www.cognigencorp.com