

SAS Macros and PROC SQL in Managing Chemical Plant Laboratory Results

Abstract

A large petrochemical plant uses SAS software to manage laboratory sample results. This SAS system provides for long-term storage of these results. It also provides tools to combine the results with process data from non-SAS systems, and this system can produce reports and graphic output. This paper discusses the use of SAS macros and PROC SQL in developing end-user tools; it covers design decisions, coding techniques, and system maintenance.

Topics Presented

This paper presents the following topics:

Topic	See Page
Definitions	189
The Lab Data Process	190
SAS Datasets and Tools	192
Combining Process Data and Lab Data	193
SAS Code for Transposing Lab Data	196
Viewing the Lab Datasets Online	200

Definitions

Introduction	Definitions of some terms used in this paper are listed below.
---------------------	--

Definition: Sample id	Technicians regularly take samples of chemical products from different locations in the plant. The <u>sample id</u> identifies where a given sample was collected. <u>Example:</u> all samples taken from valve #4 on unit #12 bear a <u>sample id</u> of UNIT12V4.
----------------------------------	---

Definition: Test name	Lab technicians have a number of different procedures that they can use to determine the characteristics or composition of a chemical sample. The names of these procedures are called <u>test names</u> (or just <u>tests</u>). <u>Example:</u> one of the procedures used to determine benzene and toluene concentration is called "BENZT". The <u>test name</u> is BENZT.
------------------------------	--

Definition: Result name	Each test determines one or more characteristics of a sample. The characteristics for a given test are the <u>result names</u>. <u>Example:</u> Test BENZT determines benzene and toluene concentrations. The <u>result names</u> for this test are BENZENE and TOLUENE.
------------------------------------	--

Definition: value	The actual measurement for a test result name is called the <u>value</u>. <u>Example:</u> A technician using test BENZT determines that the benzene concentration of a sample is .05. The <u>value</u> is .05.
------------------------------	--

Definition: Process data	In addition to lab data, <u>process data</u> are also collected. <u>Process data</u> are identified by an instrument name. <u>Example:</u> a thermometer continually registers the temperature of the chemical as it flows past valve #4 in unit #12. Hourly temperature values from the thermometer are stored in a <u>process data</u> database.
-------------------------------------	--

The Lab Data Process

Overview of the Process

The "Lab data process" consists of the following stages:

- Data gathering
 - Data storage
 - Data retrieval
-

Data Gathering

Samples of chemical products are regularly collected at the plant, and sent to the plant's laboratory. Each of the samples is identified by a "sample id," and the datetime the sample was collected. The laboratory runs one or more tests on these samples; these tests determine the composition or characteristics of the chemical product.

Data Storage

The results of the tests are then sent to the SAS system as a set of observations. Each observation is keyed by a concatenated key consisting of the datetime that the sample was taken, a sample id, the test run on the sample, and the particular result name for that test (some tests produce multiple result names).

Data Retrieval

Engineers interested in correlating process conditions with the lab analysis of products then retrieve the lab data.

Example: Data Gathering

- A technician takes a sample of a chemical stream from valve #4 on unit #12. This sample is tagged with a sample id of "UNIT12V4" as well as the time the sample was taken ("07JUL92:06").
 - This sample is delivered to the lab, where the lab technician runs two tests on this sample.
 - The first test, named BENZT, determines the concentrations of benzene and toluene in the sample.
 - The second test, named ZTEST, runs a more sensitive determination of toluene concentration.
-

Continued on next page

The Lab Data Process, Continued

Example: Data Storage The technician enters the results of the tests into the lab computer (which is not a SAS system) like this:

Sample id: UNIT12V4
Date: 07JUL92 Time: 06:30
Test name: BENZT
Result name: BENZENE Value: .02
Result name: TOLUENE Value: .05
Test name: ZTEST
Result name: TOLUENE Value: .049

The lab computer sends the test results to a SAS database on the IBM VM system. The sample above would enter the SAS database as the following observations:

<u>DATETIME</u>	<u>SAMPLEID</u>	<u>TEST</u> <u>NAME</u>	<u>RESULT</u> <u>NAME</u>	<u>VALUE</u>
07JUL92:06:30	UNIT12V4	BENZT	BENZENE	.02
07JUL92:06:30	UNIT12V4	BENZT	TOLUENE	.05
07JUL92:06:30	UNIT12V4	ZTEST	TOLUENE	.049

The next section describes the structure of the SAS dataset.

SAS Datasets and Tools

Dataset Design The data are stored on a VM/CMS system. The lab computer sends the observations to a utility id on the VM system that updates the master datasets using the UPDATE statement. Because of the large number of observations kept (over 100,000 per month), the datasets are kept separately by month. That is, there is one SAS dataset for all January 1992 lab results, one for February 1992, and so forth. This greatly speeds up dataset updating and data retrieval. It does mean that as each month starts, the utility program must create a new SAS dataset for that month, rather than adding observations to the previous month's dataset.

Hint: Macros To determine whether a new dataset needs to be created, I use the handy %EXIST macro from Chapter 11 of the "SAS Guide to Macro Processing" to check if the dataset for that month exists, and if not, the program creates it instead of trying to update it. There are several other handy sample macros in this book, and it is an excellent tutorial and reference to the SAS macro language.

Reading the Datasets Most of the SAS programs that retrieve lab data are written by the engineers themselves. I placed a macro in our SITEAUTO MACLIB autocall library to make coding the SET statements easier. Because the names of the datasets containing current lab data change each month, regularly scheduled jobs could not hard-code SET statements to access the lab datasets. They use a macro ("%SETLAB"), and pass it a parameter of "MTD" for the current calendar month's dataset, or "LMTD" for a concatenation of the last and current month's data.

Speeding up the Macros I inherited a large and complicated macro that I "speed up" by compressing the code before storing it in the autocall library. An XEDIT macro does the compression by removing comments, extra spaces, and spaces around special characters. The code is then "flowed" across eighty column lines, and the result looks something like this:

```
DATA &LABVAL;MERGE SNAPNAME SNAPLAB;BY TAGNAME;KEEP TAGNAME SASNAME DATETIME  
VALUE;XIF &TRANS=YXTHENXDO;PROC SORT DATA=&LABVAL;BY DATETIME;PROC TRANSPOSE  
DATA=&LABVAL OUT=SNAPTRAN;BY DATETIME;ID SASNAME;IDLABEL TAGNAME;DATA &OUT;SET  
SNAPTRAN;DROP _NAME_;IF DATETIME=-.;XEND;XEND;XELSEXDO;
```

This compression cuts about 15% off the execution time for the macro.

Combining Process Data and Lab Data

Accessing Process Data

Many of the engineers need to correlate process data with lab results. Our plants' process data is kept in a Nomad database, keyed by datetime and an instrument id. Each process observation is made of this key and a value. Because SAS does not directly access Nomad databases, a rather complicated macro (mentioned in the previous section) is used to build a Nomad request, invoke Nomad, and read the returned data into a SAS dataset.

Continued on next page

Combining Process Data and Lab Data, Continued

The Problem with Combining Process and Lab Data

Once the engineer has retrieved both the lab data and the process data into SAS datasets, there is still another problem with combining the data. The process data, being regularly scheduled observations, are stored like this:

SAS dataset PROCESS:

<u>DATETIME</u>	<u>TEMP1</u>	<u>TEMP2</u>	<u>FLOW1</u>
12JAN92 13:00	245.7	266.8	33.0
12JAN92 14:00	244.4	266.5	33.2

On the other hand, the lab data are individual events, and are stored like this:

SAS dataset LAB:

<u>DATETIME</u>	<u>SAMPLE ID</u>	<u>TEST NAME</u>	<u>RESULT NAME</u>	<u>VALUE</u>
12JAN92 13:00	PIPE33	PROP	C3	44
12JAN92 13:00	UNIT1V4	SOLU	WATER	.01
12JAN92 13:00	UNIT1V4	SOLU	NAOH	11
12JAN92 13:00	UNIT1V4	PHEN	NAOH	12
12JAN92 14:00	PIPE33	PROP	C3	45
12JAN92 14:00	UNIT1V5	PHEN	NAOH	13

The engineer wants his table to look like this:

SAS dataset COMBO:

<u>DATETIME</u>	<u>TEMP1</u>	<u>TEMP2</u>	<u>PIPE33 PROP C3</u>	<u>UNIT1V4 SOLU WATER</u>	<u>UNIT1V4 SOLU NAOH</u>	<u>UNIT1V5 PHEN NAOH</u>
12JAN92 13:00	245.7	266.8	44	.01	11	.
12JAN92 14:00	244.4	266.5	45	.	.	13

The problem is transposing the LAB dataset: what are we going to name the columns labeled "PIPE33 PROP C3", "UNIT1PT4 SOLU WATER", and so on? We can't use the labels as names, because we're restricted to eight characters.

Continued on next page

Combining Process Data and Lab Data, Continued

**The
transposition
solution:
description**

This is the solution to the transposition problem:

- encode the sample id, test name, and result name into eight characters
- make the simplifying assumption that the engineer will request no more than 99 different sample ids, none of which has more than 99 different test names or result names
- generate variable names of the format "Xaabbcc", where "aa" is a serial number identifying the sample id, "bb" a number for the test name, and "cc" a number for the result name

Note: there are thousands of different samples, test names, and result names in the entire database, so one cannot assign these numbers ahead of time, but instead, must assign them for each engineer's request.

The next section presents the SAS code for this solution.

SAS Code for Transposing Lab Data

Overview

This section presents the SAS code that transposes the Lab datasets. As mentioned earlier, lab values are identified by a concatenated key that is much longer than eight characters. This SAS program encodes the keys in a particular datasets to an eight-character value.

Form of the Lab Dataset

This is the form of the lab dataset:

Dataset LAB:

<u>Variable</u>	<u>Format</u>
DATETIME	DATETIME16.
SAMPID	\$16.
TESTNAM	\$4.
RESNAME	\$6.
VALUE	\$4.

Create unique variable names

Use PROC SQL to build lists of the different values for sample id, test name, and result name:

```
PROC SQL;  
  CREATE TABLE SAMPID AS  
    SELECT DISTINCT SAMPID FROM LAB;  
  CREATE TABLE TESTNAM AS  
    SELECT DISTINCT TESTNAM FROM LAB;  
  CREATE TABLE RESNAME AS  
    SELECT DISTINCT RESNAME FROM LAB;
```

This PROC step creates three datasets ("SAMPID", "TESTNAM", and "RESNAME") that contain the values for sample id, test name, and result name. The keyword DISTINCT removes duplicate values from each dataset.

Continued on next page

SAS Code for Transposing Lab Data, Continued

Assign a serial number to each unique value

Use data steps to assign a serial number to each value of sample id, test name, and result name:

```
DATA SAMPID;  
  SET SAMPID;  
  XSAMPID = PUT(_N_,Z2.);  
DATA TESTNAM;  
  SET TESTNAM;  
  XTESTNAM = PUT(_N_,Z2.);  
DATA RESNAME;  
  SET RESNAME;  
  XRESNAME = PUT(_N_,Z2.);
```

Example:

The dataset SAMPID now looks like this:

<u>SAMPID</u>	<u>XSAMPID</u>
PIPE33	01
UNIT1V4	02
UNIT12V4	03

Continued on next page

SAS Code for Transposing Lab Data, Continued

Add the serial numbers to the lab dataset

Now the serial numbers created in the previous step are added back into the original lab dataset:

```
PROC SQL;
CREATE TABLE LAB AS
SELECT SAMPID.XSAMPID, LAB.SAMPID, TESTNAM.XTESTNAM,
       LAB.TESTNAM, RESNAME.XRESNAME, LAB.RESNAME,
       LAB.DATETIME, LAB.VALUE
FROM SAMPID, TESTNAM, RESNAME, LAB
WHERE SAMPID.SAMPID = LAB.SAMPID AND
      TESTNAM.TESTNAM = LAB.TESTNAM AND
      RESNAME.RESNAME = LAB.RESNAME;
```

Example:

If the original lab dataset looked like this:

<u>DATETIME</u>	<u>SAMPID</u>	<u>TESTNAM</u>	<u>RESNAME</u>	<u>VALUE</u>
12JAN92:13	PIPE33	PROP	C3	44
12JAN92:13	UNIT1V4	SOLU	WATER	.01
12JAN92:13	UNIT1V4	SOLU	NAOH	11
12JAN92:13	UNIT1V4	PHEN	NAOH	12
12JAN92:14	PIPE33	PROP	C3	45
12JAN92:14	UNIT1V4	SOLU	WATER	.01
12JAN92:14	UNIT1V4	SOLU	NAOH	11
12JAN92:14	UNIT1V4	PHEN	NAOH	13

then the lab dataset now looks like this:

<u>DATETIME</u>	<u>SAMPID</u>	<u>XSAMPID</u>	<u>TESTNAM</u>	<u>XTESTNAM</u>	<u>RESNAME</u>	<u>XRESNAME</u>	<u>VALUE</u>
12JAN92:13	PIPE33	01	PROP	01	C3	01	44
12JAN92:13	UNIT1V4	02	SOLU	02	WATER	02	.01
12JAN92:13	UNIT1V4	02	SOLU	02	NAOH	03	11
12JAN92:13	UNIT1V4	02	PHEN	03	NAOH	03	12
12JAN92:14	PIPE33	01	PROP	01	C3	01	45
12JAN92:14	UNIT1V4	02	SOLU	02	WATER	02	.01
12JAN92:14	UNIT1V4	02	SOLU	02	NAOH	03	11
12JAN92:14	UNIT1V4	02	PHEN	03	NAOH	03	13

Continued on next page

SAS Code for Transposing Lab Data, Continued

Create new variables

Here the new variable names are created by concatenating an "X" and the serial numbers:

```
CREATE TABLE LAB AS
  SELECT DATETIME, VALUE,
    ('X'||XSAMPID||XTESTNAM||XRESNAME) AS VARNAME,
    (SAMPID||TESTNAM||RESNAME) AS VARLABEL
  FROM LAB
  ORDER BY DATETIME;
```

Example:

The data set in the previous block now looks like this:

<u>DATETIME</u>	<u>VARNAME</u>	<u>VARLABEL</u>	<u>VALUE</u>
12JAN92:13	X010101	PIPE33 PROP C3	44
12JAN92:13	X020202	UNIT1V4 SOLU WATER	.01
12JAN92:13	X020203	UNIT1V4 SOLU NAOH	11
12JAN92:13	X020303	UNIT1V4 PHEN NAOH	12
12JAN92:14	X010101	PIPE33 PROP C3	45
12JAN92:14	X020202	UNIT1V4 SOLU WATER	.01
12JAN92:14	X020203	UNIT1V4 SOLU NAOH	11
12JAN92:14	X020303	UNIT1V4 PHEN NAOH	13

Transpose the Dataset

Now this code transposes the dataset:

```
PROC TRANSPOSE DATA=LAB;
  BY DATETIME;
  ID VARNAME
  IDLABEL VARLABEL;
  VAR VALUE;
```

Example:

the example dataset now looks like this:

<u>DATETIME</u>	<u>X010101</u>	<u>X020202</u>	<u>X020203</u>	<u>X020303</u>
12JAN92:13	44	.01	11	12
12JAN92:14	45	.01	11	13

The transposed dataset can now be merged with the dataset PROCESS (above) to achieve the desired results. I've placed the transposing code into an autocall macro for ease of use.

The Power of PROC SQL makes all this possible

If you are not familiar with PROC SQL, just look at the ways it is used above, and try to perform the same transformations with data steps and PROC SORT. It can be done, but it is much easier with SQL.

Viewing the Lab Datasets Online

Introduction

Engineers who just want to look at the data can run a SAS program that uses a DATA step window to prompt for the calendar month that the engineer is interested in. The program then invokes PROC FSVIEW with PMENUS for a quick and easy view at the data. By using PROC FSVIEW, the data are presented in "spreadsheet" format; this type of presentation is more quickly grasped and understood by the users than anything else that I have tried.

Using macro windows for the primary screen

The first screen (a macro window) that the users see looks similar to this:

```
+QUIKLAB-----+
View... Help... Exit

      QUIKLAB

      To select an option, tab the cursor to the option
      at the top of the screen and press ENTER

      Select VIEW to view lab results (one month at a time).

      Select EXIT to leave QUIKLAB.
```

Using PMENUs to request data

Selecting the "View..." option on the screen above pulls down a dialog for the user to fill out:

```
+QUIKLAB-----+
+View...-----+
Type the two-digit year and month that you want to view:
  Year:   Month:

Tab the cursor to OK (to view), or tab to Cancel
(to return), and press ENTER:

                        (OK)                (Cancel)
```

The users enter the year and month that they want to see, and select OK.

Continued on next page

Viewing the Lab Datasets Online, Continued

Using FSVIEW FSVIEW is used to display the data on the screen:
to view the SAS
datasets

```
+FSVIEW: LABDB.X9201 (B)-----
Where Mask... Save... Print Up Down Left Right End

OBS      SAMPID      TEST  RESNAME  VALUE
1        UNIT13LL3    LOOM   IV-C10   5.632
2        UNIT32PMV12  MELT   MFR      .20
3        UNIT32PMV12  CARB   C2"      9.419
4        UNIT32PMV12  CARB   C2"RUB   57.300
5        UNIT32PMV12  MIXR   DECSOL   10.65
6        UNIT32PMV12  MIXR   IV       5.073
7        UNIT32PMV12  MIXR   IV-C10   6.025
8        UNIT32PMV12  COMP   RATIO    1.34
9        UNIT32PMV12  BOIL   PERC2    34.81
10       UNIT33LXX12A22 FLOW   MFR      .20
11       UNIT33LXX12A22 CARB   C2"      9.983
12       UNIT33LXX12A22 CARB   C2"RUB   58.710
13       UNIT33LXX12A22 BRAN   S-100    1.0
14       UNIT33LXX12A22 BRAN   S-100+   .1
15       UNIT33LXX12A22 BRAN   S-12     42.3
16       UNIT33LXX12A22 BRAN   S-20     28.0
17       UNIT33LXX12A22 BRAN   S-6      1.9
18       UNIT33LXX12A22 BRAN   S-8      26.7
```

More PMENUs (at the top of the screen) allow users to specify where clauses, print the dataset, and save the dataset.

Benefits of FSVIEW

FSVIEW allows SAS sites that don't have AF installed to provide simple, menu driven data reporting tools.