

VIEWS

International SAS Programmer Community



News, Hints, Tips and Information for SAS® Users

Issue 63, 1st Quarter 2021

This newsletter issue contains articles from Anders Skölleremo on batch processing and missing values, and I have added an article on SQL and macro code layout specifically to encourage discussion. As usual I have publicised some SAS-related events for the next few months, and I have also added some more SAS formats, options and functions that you may or may not have come across before, but you might find interesting, and even useful.

If you would like to contribute an article, to re-visit and improve an existing article, or just discuss the possibility of doing so, please feel free to send an email to me at views-uk@hollandnumerics.org.uk.

Philip R Holland (Newsletter Editor)

Membership of VIEWS is free and you can register for notification of all future VIEWS News issues by registering at blog.hollandnumerics.org.uk, and then emailing views-uk@hollandnumerics.org.uk, or using the blog site menu, to request access to the VIEWS UK forum. Members will receive an email notification whenever a new issue of the free quarterly VIEWS News is published on the web site, please inform us when you change your email address, otherwise your registration may be cancelled.

Did You Know?

If you have a useful hint or tip, please send it to the Editor, and share it with the VIEWS membership.

Logic for safe Batch production jobs

Consider the following situation: You have a Data Library which is regularly updated (daily, weekly or monthly) using a batch job. The supervisor program for the batch job may find fatal errors. The user programmed data checks may find errors, when reading the new data, processing them or adding them the existing data.

If fatal errors have been found you want the entire host batch job to stop execution, with a suitable error message. No update or addition to the original Data Library should take place. The batch job should complete the execution, with a host system message like "ERROR ABEND xxxxxxxx".

If no fatal errors are found when executing the user written batch program, the host batch job ends in a normal way, with a host system message like "SUCCESSFULLY COMPLETED". This also applies if some user programmed data checks reports errors, when executing the user written batch program. An update or addition to the original data Library shall in this case take place.

Below I have tried to write the logic for the different steps in such a batch job. The original situation was IBM mainframe and SAS as the programming tool. However these thoughts are applicable to many computer environments and many software packages.

First step: Create a host system variable MY_EXT_VAR, with the value "ERROR".

Second step: Copy the original Data Library to another location with a suitable name.

Third step: Start executing the user written program. Immediately create a Macro variable MY_INT_VAR and assign it the value "OK". There are three quite different situations when executing the user written program:

1) The program supervisor finds errors in the compilation/execution and ends the users batch program with an ABEND. The variable MY_EXT_VAR still has the value "ERROR".

If fatal errors are found during execution of the users program, the programs end, without changing the value of this variable. The system variable MY_EXT_VAR still has the value "ERROR".

Fourth step: Conditionally executed only if MY_EXT_VAR has the value "OK". Thus it is not executed. The entire batch job ends with a standard host system error message like "ERROR ABEND xxxxxxxx".

2) Everything is quite OK during program compilation and execution. The user programmed tests, that are executed, do not signal any error. The variable MY_INT_VAR has the value "OK" to the end of the execution. This value is exported to the external variable MY_EXT_VAR. The program ends normally.

If no fatal errors are found during program execution, the value ("OK") of the variable MY_INT_VAR shall be exported, at the end of the program, to the system variable MY_EXT_VAR.

Fourth step: Conditionally executed only if MY_EXT_VAR has the value "OK". Thus it is executed. The new Data Library is copied to the original Data Library and replaces it. The entire host batch job ends with a standard host system message like "ALL IS WELL".

3) The user programmed tests find errors and assigns the value "ERROR" to MY_INT_VAR.

This value is immediately exported to the external variable MY_EXT_VAR. The user programmed tests probably also writes messages to a user error log file and writes "strange data" to a user error Data Library. The purpose is to provide information to the programmer for error analysis.

If errors are found using the user programmed data checks, the value "ERROR" is assigned to MY_INT_VAR. This value is

exported, at the end of the program, to the system variable MY_EXT_VAR.

Fourth step: Conditionally executed only if MY_EXT_VAR has the value "OK". Thus it is not executed. The entire batch job ends with a standard host system message like "ALL IS WELL". If possible, change this message into something like "USER FOUND ERRORS".

Benefits:

The original data are always there. Never destroyed.

The programmer can define several more or less complex "Error conditions". This means that the execution shall continue to the end, but the data shall not be rewritten. This makes it possible to look at system log files, and the error reports and error data Libraries created by the programmer.

Anders Skölleremo, Retired

Ask The Consultant

This part of VIEWS News is where you can get your technical questions answered. Send your questions to the Editor.

Logically speaking, all this is correct

Q: How can I cope with missing values in my data?

A:[SAS] A SAS-programmer wanted to report of positive and negative feelings among the users at a SAS Users Conference. Four different persons reported different results, after having asked the participants. Let us have a look at his/her program below:

```
DATA work.Evaluate;
  INPUT Pos Neg;
  IF Pos - Neg THEN PUT "Great";
  ELSE PUT ".Fail";
  CARDS;
100 0
1 99
100 .
. 100
50 50
RUN;
```

with the result:

```
Great
Great
.Fail
.Fail
.Fail
```

Obviously the program does not work in the intended way. The programmer was hoping that the software should solve his/her problems. But that is never the case! The programmer must know how the software works, and write a program that uses this knowledge to solve his/her problems. This means planning, implementing and testing.

In SAS software zero (0) and missing (often represented by only a period .) are logically speaking False values, and all other values are True (even -1 and -999).

The SAS program above contains an arithmetic expression

```
IF Pos - Neg THEN
```

First the value of the expression is evaluated, and produces a number. Then this number is used as a logical value. Maybe the result is "not quite" what the programmer had expected:

Case	Pos	Neg	Pos - Neg	Printout	Comment
1	100	0	100	Great	Great – Of Course!
2	1	99	-98	Great	Ugh! But negative values are logically True.
3	100	.	.	.Fail	No. of negative responses is unknown.
4	.	100	.	.Fail	No. of positive responses is unknown.
5	100	100	0	.Fail	Rather "Neither Great nor Fail".

In case 3, the programmer must check if "the missing number for negative response", is correct.

Perhaps the missing value means zero (0) negative responses, as in case 1?

The printout will say ".Fail", but that is perhaps not true. Rather, the person who collected the data, could not decide how many were negative. Perhaps we can say that data collection "Failed".

The way logic values work in SAS, combined with how the program is written, mean that, in this case, that "any number of negative responses (e.g. 0 or 100)" will give the printout "Great". Perhaps the SAS program is not in this case so "Great".

In case 4) is "the number of positive responses" really unknown? Perhaps it is zero (0). This gives the expression Pos - Neg the value -100, which is True. So the Printout will be "Great".

In case 5) the expression Pos - Neg has the value zero (0), which is False. The Printout will be ".Fail". However 50% of the respondents liked the Conference, so a better printout would rather be "Neither Great nor Fail".

What the programmer should have done: He/she must look closely at all the data values and in each case how data shall be interpreted. Especially values like zero or missing. A missing value is often interpreted as "no values could be registered". This usually means that a further analysis is very difficult to make.

The next step for the programmer is to use SAS Software (or some other software) to implement these thoughts. It is then necessary to have a good basic knowledge of the software. SAS has its way of defining True and False, and how to use logical values in arithmetic expressions. It has its advantages and disadvantages. Other software packages probably have other definitions, and other advantages and disadvantages.

Interesting alternative IF-statements:

```
IF Pos GT Neg THEN ...
IF Pos - Neg GT 0 THEN ...
```

Neither of them is a perfect solution.

An example of a slightly better SAS program:

```
DATA work.Evaluate;
  INPUT Pos Neg;
  IF Pos GE 0 AND Neg GE 0 THEN DO;
    IF Pos EQ Neg THEN PUT "Neither";
    ELSE IF Pos GT Neg THEN PUT "Great";
    ELSE IF Pos LT Neg THEN PUT "Fail";
  END;
  ELSE PUT "! Data?"; /* Strange Data values */
  CARDS;
100 0
1 99
100 .
. 100
50 50
RUN;
```

In the WPS Analytics software the value 1.E-307 is one of the smallest, and that is of course True. Slightly smaller still is the missing value, which is False. The difference between the True value and the False value, is in many cases very small. In this case it is “smaller than 1.E-308”.

Anders Skölleremo, Retired

Annoying SAS Programming Styles: Part 4

Q: How can I write PROC SQL code to make it easier to maintain?

A:[SAS] In the following examples, the PROC SQL (and the SAS macro) code is exactly the same. But, in my opinion, coding layout 3 is easier to maintain, because all the lines have the same layout and you can insert or delete lines with a reduced risk of introducing syntax errors:

1. This is a prose layout, which is quick to write, but can be a nightmare to maintain:

```
PROC SQL;
CREATE TABLE new AS SELECT a.col1, b.col2,
a.col3 FROM a, b WHERE a.col1 = b.col1 AND
a.col2 = b.col2;
QUIT;

%macrocall (dsn=sashelp.class, invar1=height,
factor1=2.54, outvar1=heightcm, invar2=weight,
factor2=1/2.2, outvar2=weightkg)
```

2. This split layout is easier to read, but inserting and deleting lines of code can introduce syntax errors (be very careful when deleting or duplicating red or green lines):

```
PROC SQL;
  CREATE TABLE new AS
    SELECT a.col1,
           b.col2,
           a.col3
  FROM a,
       b
  WHERE a.col1 = b.col1 AND
        a.col2 = b.col2;
QUIT;
```

```
%macrocall (dsn=sashelp.class,
            invar1=height,
            factor1=2.54,
            outvar1=heightcm,
            invar2=weight,
            factor2=1/2.2,
            outvar2=weightkg)
```

3. This is a comma-first layout, which is easier to read and which lets you insert and delete lines of code relatively safely. Note that blue lines are always written with commas and operators at the beginning of each line so that they can be safely duplicated or deleted. Only the black and red lines need to be treated with care:

```
PROC SQL;
  CREATE TABLE new AS
    SELECT a.col1
           ,b.col2
           ,a.col3
  FROM a
        ,b
  WHERE a.col1 = b.col1
        AND a.col2 = b.col2
;
QUIT;

%macrocall (
  dsn=sashelp.class
  , invar1=height
  , factor1=2.54
  , outvar1=heightcm
  , invar2=weight
  , factor2=1/2.2
  , outvar2=weightkg
)
```

When formatting your programs please remember that someone else may have to understand and maintain it in the future, and that person may be you!

Philip R Holland (Newsletter Editor)

Adapted from “SAS Programming and Data Visualization Techniques” by Philip R Holland (Apress, 2015)

Formats, Options, and Functions

This section is devoted to the description of useful, or unusual, SAS Formats, Options, and Functions. Please read the SAS documentation to find out more about them.

[SAS] The SAS informat **\$CHARw.** works like the **\$w.** informat, in that it can be used to read character strings. However, **\$CHARw.** can be used to read character strings with embedded blanks, whereas blanks are treated as string terminators by **\$w.**

[SAS 9.0] In SAS version 7 the maximum allowed length of character variables was increased to 32,767, but the SAS Log would still warn the user if the variable length exceeded 262. This is still the default action, because **QUOTELENMAX** is set as the default option. This warning can be suppressed using the SAS option **NOQUOTELENMAX**.

[SAS 6] These are not strictly functions, but the PROC SQL summary functions can include **DISTINCT**, or its alias **UNIQUE**, to find unique values, and the SELECT clause can use these keywords to find unique records.

In Brief

- Back issues of *VIEWS News* are available from the VIEWS UK forum at blog.hollandnumerics.org.uk. Please register for free membership on the blog, and then request free access to the VIEWS UK forum.
- The SAS Professional Forum group on LinkedIn, which currently has over 35,900 members. Everyone is welcome to request to join, but discussions here are restricted to SAS-related topics: www.linkedin.com/groups?gid=70702.
- The SAS Opportunities group on LinkedIn, which currently has over 12,900 members. Everyone is welcome to request to join, but discussions here are also restricted to SAS-related topics: www.linkedin.com/groups?gid=805217.
- In the UK and Ireland the SUGUKI group had been holding regular meetings in London, and also occasional meetings in Bristol, Dublin, Edinburgh, Leeds and Manchester. Unfortunately lockdown has necessitated running webinars instead. For more details and to register for future events please go to meetup.com/SUGUKI.
- Fancy joining the SAS EMEA Analytics Explorers, then please send your **country, name, email address** and **company** with your permission to pass these details on to SAS to views-uk@hollandnumerics.org.uk. Other regions include Asia Pacific, Canada, Latin America and US, but to join these communities based on where you live, you will now need to provide this information to a current member of these communities, instead of me. Sorry, but this is a change from the information in *VIEWS News* issue 61!
- WPS Community Edition provides users with free access to WPS Analytics software including a SAS language compiler with coding and workflow tools on unlimited data volumes. A Community Edition licence key will be issued that will activate the software for up to six months. At the end of this term, users may be invited to renew their licence and to upgrade to the latest available version of the software. No technical support service is offered with the Community Edition. Download from www.worldprogramming.com/try-or-buy/editions. Note that the software is made available without warranty or indemnity, and is not made available to US residents.

Diary

Are you organising an event that would be of interest to the VIEWS readership? Let us know as we are interested in all non-profit making events related to SAS.

May 2021

18-20 Virtual SAS Global Forum

For more details about times and speakers please go to the SAS Global Forum web site at www.sasglobalforum.org.

24-27 Virtual PharmaSUG US

For more details about times and speakers please go to the PharmaSUG web site at www.pharmasug.org.

June 2021

14-18 Virtual PhUSE US Connect

For more details about times and speakers please go to the PhUSE web site at www.phuse-events.org.

August 2021

27-28 PharmaSUG China, Shanghai, China

For more details about times and speakers please go to the PharmaSUG China web site at www.pharmasug.org/china.

October 2021

29 PharmaSUG North Carolina, Chapel Hill, NC, USA

For more details about times and speakers at this Single Day Event please go to the PharmaSUG web site at www.pharmasug.org.

November 2021

15-19 Virtual PhUSE EU Connect

For more details about times and speakers please go to the PhUSE web site at www.phuse-events.org.

Contacts

Anders Skölleremo

Email: anders.skolleremo@one.se
Web: www.linkedin.com/in/anders-skolleremo-a497964/

SAS UK

Tel: +44-(0)1628-486933
Web: www.sas.com/uk

VIEWS

Web Site

Web: blog.hollandnumerics.org.uk

Newsletter Editor

Email: views-uk@hollandnumerics.org.uk

VIEWS News is published quarterly for the VIEWS International SAS Programmer Community by Holland Numerics Limited.