

VIEWS

International SAS Programmer Community



News, Hints, Tips and Information for SAS® Users

Issue 58, 4th Quarter 2019

General Coding Efficiencies

There are generally accepted methods of programming efficiency which have been expanded on by other papers. The following programming techniques are mentioned in Kirk Lafler's paper [1], but most often cited by others. Best practices aim to reduce your overall programming time;

- Reduce the overall data set size by only keeping the needed variables and rows needed. If your query only requires the current year and a few variables, then limit the data pulled for the query.
- Use SAS procedures to analyse data. The SAS procedures are already tuned to be as efficient as possible. It's easier to use a technique like PROC SQL or PROC SUMMARY to get answers.
- Sorting is an expensive operation. Avoid sorting data when it's not required.
- Expand or update your knowledge of the SAS language. Consider adding techniques such as HASH programming and DataStep2 to your skillset.

This is not an exhaustive list but merely a few ideas of easy ways to be more efficient.

Using Compression to Reduce Run Time

With SAS datasets, you may have data items that need to have a long length to incorporate large values, but most of the values might only need a fraction of that length. When it comes to storing that dataset, SAS makes room for the total length of each value, regardless of how much space the actual data requires. This can be very inefficient since you are using storage for blank space in your data file. Use the compress option on the dataset or program to alleviate this problem.

Dataset

name	address	phone
Ryan	10 5th Avenue	1234567890
Sam	56 King Street	2345678901
Jon	80 Elm Road	3456789012

Uncompressed

20	50	10
20	50	10
20	50	10

Compressed

4	13	10
3	14	10
3	11	10

An example of how compression works in SAS is shown in the figure above. The dataset shown has character lengths of 20, 50, and 10 for the data columns "name", "address" and "phone"

This newsletter issue contains an article from Jonathan Boase about programming efficiency, a report on the October SUGUKI meeting in London from Laura Illingworth, and a Base SAS macros article from myself. I have again included some SAS-related events publicised for the next few months, and I've also added some more formats, options and functions that you 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.

Tame Your Big Data with SAS Coding Efficiencies

Working with big data comes with its own set of challenges. Large data sets with millions of rows and columns often take a very long time to process. The impact of this lengthy processing time is that often businesses have to wait long periods of time to retrieve the results. SAS programming language's flexibility actually allows multiple ways to code a solution. However, often enough, the most obvious solution won't be the most efficient one.

You've likely experienced code execution taking forever to run. In most cases, you set those jobs to run at night so as not to waste a day of work. But what if we could optimize performance and run the said jobs in a time period that's more conducive to a day's work?

Explore the following general coding efficiencies, learn how SAS stores data, and take a peek at the tactical technique you could use to work with the database.

respectively. The Uncompressed section shows how the data file would look if each of the values were to take up their entire allocated length, no matter what the actual values are. In the Compressed section, you can see that each of the values only take up the space of their value (for example, Ryan = 4-character length). The blue space in the diagram represents the storage space saved by using compression and can be used to store other values.

As an example to see how much space can be saved, see how the compress option was used on the SASHELP.BIRD dataset.

```
data birdNoCompress;
  set SASHELP.BIRD;
run;

data birdCompress(compress=YES);
  set SASHELP.BIRD;
run;
```

This dataset only has four data items, but one column (text) has a length of 1,200 so that it can include long values even though most values only use around 20-50 characters. The code shown to the left is the code that was executed. After execution, we ran a PROC COMPARE on each dataset. The birdNoCompress dataset had a file size of 768KB and the birdCompress dataset was only 192KB. The compressed dataset was about 75% smaller than the original!

Now there are a few things to note about using compression before using the (compress=YES) option of the data step for every program:

- It can be set as on a per-dataset basis (see BIRD example above) or as a system option. The system option compresses every dataset that is created in your program.
- You can only compress numeric or character values in a dataset, not both. “compress=YES” and “compress=CHAR” compresses character values only; “compress=BINARY” does numeric values.
- Using compression decreases I/O and storage space when processing, which is good, but it also increases the program runtime, since the dataset must be uncompressed before being processed. For this reason, compression should be selectively used when working with larger datasets.

The REUSE= dataset or system option needs to be set to YES in order to use the freed-up space for new observations. If the option is not set (default is NO), then new observations are simply appended to the data file.

Adding Indexes to Your Dataset

If your data is not stored in a database, then tuning the SAS datasets can be helpful. One method to improve the query time from SAS datasets is to add an index. Indexing gives a SAS dataset direct access to specified observations. Instead of reading every record, SAS can quickly find the requested observations. In some cases, the technique can reduce the query time by 95%.

SAS Indexes come in two forms; simple and composite. A simple index is created on one key variable while a composite index is created on multiple variables. Indexes are typically used if you are filtering data frequently or joining columns.

Indexes can be created in PROC DATASETS, PROC SQL, and the DATA statement. Here's an example using the

DATASETS procedure. An index is added to the SALES_REP data set.

```
/* PROC DATASET EXAMPLE */
proc datasets library=performance;
  modify sales rep;
  index create sales rep id / unique;
run;
```

SAS indexes return observations in sorted order. This eliminates the need to sort the dataset in the subsequent data processing. Indexes may not be appropriate for all situations, so refer to the SAS documentation for more detailed information.

Be Efficient with Your Coding Techniques

It is difficult to quantify the added value to an organization when code executes in a shorter time period. Whatever the case is, whether you're testing code changes or updating a system process, if it runs in a few hours (or quicker!) rather than overnight, you not only save time but also money.

Using one or more of the coding efficiencies mentioned above, you can run jobs in a shorter time frame.

Jonathan Boase, Zencos

[1] Efficient SAS Programming Techniques, Kirk Lafler, SUGI 25

support.sas.com/resources/papers/proceedings/proceedings/sugi25/25/hands/25p146.pdf

hollandnumerics.org.uk/wordpress/wp-content/uploads/2019/12/sugi25-25p146.pdf

Ask The Consultant

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

Managing SAS Macro Variables in Macros

Q: How do you make certain macro variables created in macros don't overwrite macro variables in the calling program?

A:[SAS] In theory macro variables should be easier to control, as SAS uses the concept of “scope” for them. This means that a “global” macro variable is accessible anywhere in a SAS program, but a “local” macro variable can only be accessed within a macro and in a called macro. Macro parameters are automatically defined as local macro variables, but all new macro variables should be defined explicitly as global or local. The following code shows the three types of macro variables: global &macvar1, local %macvar1 and parameter &var, where only &macvar1 exists outside of macro %newmac, but all three macro variables exist inside:

```
%GLOBAL macvar1;
%LET macvar1 = 1;

%MACRO newmac(var=);
  %LOCAL macvar2;
  %LET macvar2 = 2;

  %PUT macvar1=&macvar1.
        macvar2=&macvar2.
        var=&var.;
%MEND newmac;
```

```
%newmac(var=Y)

%PUT macvar1=&macvar1.
      macvar2=&macvar2.
      var=&var.;
```

The log from this program will show the following after %newmac():

```
macvar1=1      macvar2=2      var=Y
```

The following is seen in the log after second %PUT:

```
WARNING: Apparent symbolic reference MACVAR2 not
resolved.
WARNING: Apparent symbolic reference VAR not
resolved.
macvar1=1      macvar2=&macvar2.      var=&var.
```

This should be fine, except that it is perfectly possible to create a global macro variable inside of a macro. If this macro is then compiled, it may not be obvious that a global macro variable has been created, with unforeseen consequences.

So how could a macro be written to create a global macro variable safely? One way would be to specify the name of the new global macro variable in a macro parameter:

```
%MACRO newglobal(global1=, value1=,
                  global2=, value2=);

  %GLOBAL &global1. &global2.;

  %if "&global1." ne "" %then %do;
    %LET &global1. = &value1.;
  %end;

  %if "&global2." ne "" %then %do;
    %LET &global2. = &value2.;
  %end;

%MEND newglobal;

%newglobal(global1=newglob, value1=Y)

%PUT global1=&global1. value1=&newglob.;
```

The log from this program will show the following:

```
WARNING: Apparent symbolic reference GLOBAL1 not
resolved.
global1=&global1. value1=Y
```

Philip R Holland (Newsletter Editor)

Extract from "Writing Reusable Macros" by Philip R Holland

Formats, Options, and Functions

This section is devoted to the description of useful, or unusual, SAS Formats, Options, and Functions.

[SAS 9.4] The date formats WEEKUw., WEEKVw. and WEEKWw. with the U, V and W indicating the name of the week algorithm used. They create different string layouts for years, weeks and days from SAS date values depending on the value of w:

- 3-4, e.g. W10
- 5-6, e.g. 19W10
- 7-8, e.g. 19W1003
- 9-10, e.g. 2019W1003
- 11-200, e.g. 2009-W10-03

There are corresponding informats WEEKUw., WEEKVw. and WEEKWw. to read these strings into SAS date variables.

[SAS 9.4M5, SAS 9.4M6] The ACCESSIBLE option on the ODS PDF statement enables accessible PDF documents following the PDF/Universal Accessibility (PDF/UA) format to be generated, and is fully supported from SAS 9.4M5. The system option ACCESSIBLEPDF switches on ODS PDF ACCESSIBLE by default from 9.4M6.

[SAS 9.3] The functions ROUND(), FLOOR() and CEIL() should be familiar to most SAS programmers, as well as the ROUNDE() which rounds to the nearest even multiple of the second argument, and all 4 functions perform extra computations to make their results agree with decimal arithmetic. However, floating point numeric values don't always represent decimal values precisely, so there are now 3 more functions ROUNDZ(), FLOORZ() and CEILZ(), which do not perform extra computations, so their results may differ from decimal arithmetic.

News

Why SAS-meet-up?

People often ask me why go to a SAS meet up? There's a very simple answer: free beer and wine. However, you will be hard pressed to find a better bunch of friendly knowledgeable people. The talks that are given are practical and mind broadening. After attending a few sessions in London, Vasilij Nevlev asked me to present, and I was honoured to do so in October 2019.

The first talk presented by Dr Mark Geldart (SAS UK) showed how D3 can be effectively used to produce impressive data visualisations. It was great to see D3 in action and how quickly it could produce visualizations of large datasets. Another highlight was seeing it integrating well into other applications.

I found presenting a great way to reflect and gain a deeper understanding of how to express statistical topics in simple and complex ways. I presented on propensity scores, they enable people to compare treatment effects over slightly different populations. The talk went well with many people asking questions. People are happy to hear and share ideas that can help in everyday SAS use. This is the reason to attend SAS meet up. If you don't believe me come on down and check it out, if you do then what are you waiting for?

Laura Illingworth, Thrombosis Research Institute

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.

February 2020

6 SUGUKI, London, UK

Speakers to be announced. For more details see www.meetup.com/SUGUKI.

March 2020

5 SUGUKI, London, UK

Speakers to be announced. For more details see www.meetup.com/SUGUKI.

8-11 PhUSE Connect US, Orlando, FL, USA

The Pharmaceutical Users Software Exchange (PhUSE) conference is now the premier programming event for the pharmaceutical industry in the USA. See www.phuse.eu for more details.

29 March-1 April SAS Global Forum, Washington, DC, USA

The SAS Global Forum (SASGF, formerly called SUGI) is held each year and focuses on the technical aspects of SAS software. See www.sasglobalforum.org for more details.

May 2020

10-13 PharmaSUG US, San Francisco, CA, USA

See the PharmaSUG web site at www.pharmasug.org for further details.

June 2020

22-23 SAS UK Forum, Birmingham, UK

See www.sas.com/uk/events for further details.

August 2020

28-29 PharmaSUG China, Beijing, China

See the PharmaSUG web site at www.pharmasug.org for further details.

November 2020

8-11 PhUSE Connect EU, Liverpool, UK

The Pharmaceutical Users Software Exchange (PhUSE) conference is now the premier programming event for the pharmaceutical industry in Europe. See www.phuse.eu for more details.

In Brief

- Back issues of *VIEWS News* are available from the VIEWS UK forum at blog.hollandnumerics.org.uk. Please register on the blog, and then request free access to the forum.
- The SAS Professional Forum group on LinkedIn, which currently has over 33,830 members. Everyone is welcome to join, but discussions are restricted to SAS-related topics: www.linkedin.com/groups?gid=70702.
- In the UK and Ireland the SUGUKI group is holding regular meetings in London, and also occasional meetings in Bristol, Dublin, Edinburgh, Leeds and Manchester. For more details and to register for future events please go to www.meetup.com/SUGUKI.

Contacts

Jonathan Boase

Web:

www.linkedin.com/in/jonathanboase

Laura Illingworth

Web:

www.linkedin.com/in/laura-illingworth

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.