

VIEWS

International SAS Programmer Community



News, Hints, Tips and Information for SAS® Users

Issue 76, 2nd Quarter 2024

This newsletter is the home of Foundation SAS and brings you information about making the most of SAS 9. This newsletter issue includes a macro that makes controlling graph text characteristics easy from LeRoy Bessler, Richard Carson completes his discussion about configuring of SAS processes without editing them in the sixth part of a series of articles about KT, and I have to admit that I delayed the publication of this newsletter to tell you about my 2 new books, which were finally published on 21Jun2024 (see <https://hollandnumerics.org.uk/wordpress/book-table/>). 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.

Ask The Consultant

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

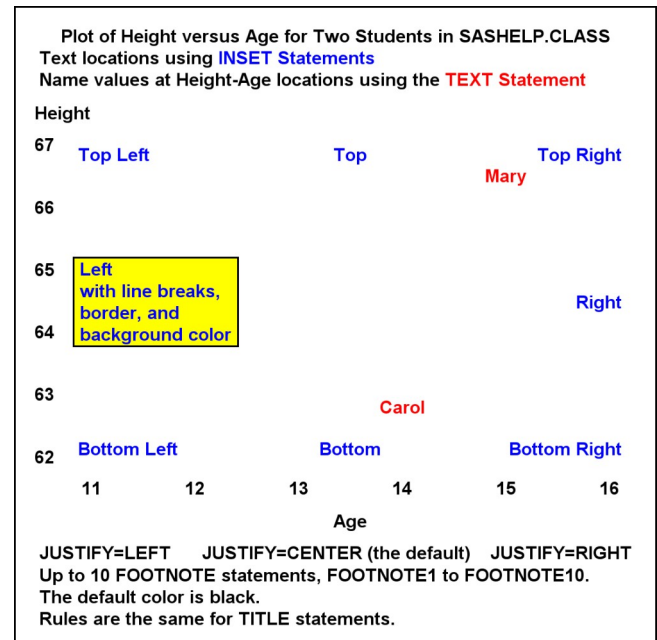
The most efficient way to manage text characteristics for a graph

Q: What is the most efficient way to manage text characteristics for a graph?

A: [SAS 9.4] The answer is a simple, but high-value, macro that I created, which has saved me more time than any other tool used in my SAS experience. Before I explain the macro, let me talk about the controls for SAS graphic text characteristics and How To Assure Text Readability.

Below is a graph that is a bona fide graph, but uses text only, and no traditional graphic elements. It is like a scatter plot, but no markers are used, and the plot points are instead identified by

text labels centred at the plot points. When talking about making graph text readable, a graph with only text is a good example. This example uses the INSET statement as an alternative to the ODS Graphics Annotate feature. In some cases, its lesser capabilities may be adequate for your needs.



Though I prefer 11pt Arial Bold, the column width limit (and, therefore, image width) of 3.65 inches required me to use 7pt Arial Bold instead. Code for this is shown later.

From my Principia Graphika (the short Latin version of the title of my book's Chapter 1, for which the long form is "Principles of Communication-Effective Graphic Design"), an extremely important Principium (or, Principle, if you prefer) is:

Make all text and numbers readable.

You say, "That's obvious". Well, the fact is that it is not always achieved. The most frequent failure is omitting to:

Assure sufficient colour contrast on a plain background.

Contrast is a key to readability. I have seen, with great difficulty, yellow text on white and dark blue text on black.

Discouraging use of backgrounds (whether gradient, or texture, or photograph) is another principium. Black on plain white for books and newspapers is not an accident.

There are three other corollaries to making all text and numbers readable.

Use sufficient size.

Bold face always enhances readability.

Sans serif text is more readable.

Bold is never harmful, or dangerous, or an obstacle to reading. Serif text is best for long text (e.g. a novel), not a graph. For a graph, my preferred text characteristics are 11pt Arial Bold.

Also included in the Principia Graphika is:

Make all text horizontal.

If the Y axis label is not too long to display at the top of the axis, include it there. Instead, the labels for both axes can be in a subtitle, if they are not already in the title itself. Axes need labels only if not identified elsewhere. Date axes never need a label, unless a special date needs a description. Axis identification in a title or subtitle is the quickest and shortest route to identification.

Title and subtitle are the first two things that any graphic viewer will read.

Though the practice is widely demonstrated in the book, omitted from the book's Principia was:

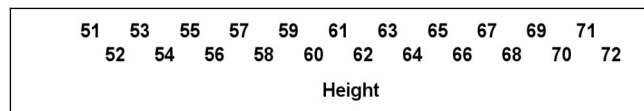
Make all text in your graph the same font, same size, and same weight, if possible.

(Doing that easily is the purpose and benefit of the macro taken up below in this article.)

X axis values are apt to be a challenge. An XAXIS alternative to using VALUEATTRS=(SIZE=SmallerSizeChoice) is

XAXIS FITPOLICY=STAGGER...;

Using it, e.g. in a plot of Weight versus Height for the students in SASHELP.CLASS produces the X axis shown below.



There are several ways to control text besides VALUEATTRS. I was reluctant to try to list them here. I might miss one. Here are the ones that I can think of: LABELATTRS, TITLEATTRS, DATA LABELATTRS, CURVELABELATTRS, SEGLABELATTRS, and HOLELABELATTRS. Note that TITLEATTRS is used for titles of graphic elements within the graph, such as a key legend.

For the graph's titles and footnotes, there are FONT= and HEIGHT= and the BOLD option. The ATTRS options above have suboptions of FAMILY (not FONT), SIZE (not HEIGHT), WEIGHT (BOLD or NORMAL), STYLE (ITALIC or NORMAL), and COLOR.

The default colour of BLACK is always safe. From Principia Color (Color is the Latin word for America's "Color" and Britain's "Colour"):

Use colour to communicate, not to decorate.

From that,

Use colour for emphasis, or for colour-coding.

Data in a GROUP variable is a common case for colour.

Before I show you a tool to make specification of graph text characteristics easy and uniform, let's look at doing it the hard way, for the text-only scatter plot above.

Rather than present all of the code, below are only some representative pieces.

```
/* Code to control characteristics of segments  
of titles and footnotes: */  
TITLE2 FONT='Arial/Bold' HEIGHT=7pt JUSTIFY=LEFT  
Text locations using COLOR=blue  
INSET Statements';  
FOOTNOTE1 FONT='Arial/Bold' HEIGHT=7pt  
JUSTIFY=LEFT JUSTIFY=LEFT'  
JUSTIFY=CENTER JUSTIFY=CENTER (the default)'  
JUSTIFY=RIGHT JUSTIFY=RIGHT';
```

```
/* Code only to control the multi-feature complicated INSET statement. All INSET statements used in this example include quoted description of the location, the POSITION option to assign the location, and the TEXTATTRS assignment as shown. For POSITION=Left, separating the assigned INSET text string into quoted pieces produces a line break between each pair of quoted pieces. */
```

```
INSET 'Left' 'with line breaks,'  
'border, and' 'background color' /  
POSITION=left  
BORDER  
BACKCOLOR=yellow  
TEXTATTRS=(COLOR=blue FAMILY=Arial SIZE=7pt  
WEIGHT=Bold);
```

```
/* Code to start the procedure step, and to
display name values for two students at the
age and height coordinates. No SCATTER
statement with markers is included. */
```

```
PROC SGPLOT DATA=sashelp.class NOBORDER
NOAUTOLEGEND;
WHERE Name IN ('Carol' 'Mary');
TEXT X=Age Y=Height TEXT=Name /
POSITION=Center
TEXTATTRS=(COLOR=red FAMILY=Arial SIZE=7pt
WEIGHT=Bold);
```

Full code for the example above, but sized bigger for my book, is in Listing A3-1 in Appendix A of the book. That code has the quirk of `POSITION=Top` for the Name text, perhaps since an earlier version included a `SCATTER` statement with markers, and I did not want the Names to overlay the markers.

After that long introduction, finally here is a simple macro to deliver a uniform font, size, and weight for everything in your graph. I actually came to develop this macro by necessity, rather than by choice. When I started my book, I was expecting to code all of my text characteristics explicitly for any aspects where I didn't like the software's defaults, i.e. I was a naïve lazy abstainer from using ODS Styles to easily control my graphs' text characteristics.

For more about gradient legends, and how best to use them, if at all, please see, e.g. Chapter 6 about Heat Maps in my book. Having to use them, I was perplexed that the GRADLEGEND statement has no VALUEATTRS option to control the text values along the colour band of the legend. SAS documentation as of May 10, 2024 confirms that this omission still persists. The KEYLEGEND statement has always had the option, and includes more. The only recourse offered by SAS Technical Support was to use an ODS Style. That's how I got here, and why I developed what I will show you.

With an ODS style, you can apply your text characteristics wishes to your graph's text by starting your graph creation SG procedure code block with

```
ODS DesiredDestination STYLE=MyStyle . . . ;
```

where DesiredDestination can be LISTING (for a standalone style to disk), PDF, EXCEL, HTML5, etc.

You can change the defaults of the macro below to whatever are your preferences in an altered version of the macro if you choose to create one, or you can simply override them when you invoke this macro.

Run the macro one time to store any desired custom ODS style permanently in your SASUSER template store, so the style can be used in the creation of any future graph.

```
%MACRO
  AllGraphTextSetup(Size, Family=Arial,
                    Weight=Bold);

/* If stored, this macro FileName must be
   AllGraphTextSetup.sas */

/* The other option for Weight is Normal. */

/* The number N specified for Size is used as
   Point size, Npt. This number is the macro's
   positional parameter. A macro may have at
   most one positional parameter, and it must be
   first in the invocation of the macro, when
   other parameters are used. */

/* Family may be used for font names with
   embedded blanks, such as Times New Roman, but
   the style created omits blanks, as in style
   name GraphFontTimesNewRoman10ptBold. Blanks
   are removed with the %LET statement below. */

%LET FamilyForStyle =
  %SYSFUNC (COMPRESS (&Family, ' '));

PROC TEMPLATE;
  DEFINE STYLE
    GraphFont&FamilyForStyle.&Size.pt&Weight;
  PARENT=styles.listing;
  CLASS GraphFonts /
    'GraphValueFont'=('&Family",&Size.pt,&Weight)
    'GraphLabelFont'=('&Family",&Size.pt,&Weight)
    'GraphDataFont'=('&Family",&Size.pt,&Weight)
    'GraphTitleFont'=('&Family",&Size.pt,&Weight)
    'GraphFootnoteFont'=('&Family",&Size.pt,&Weight);
  END;
RUN;

%MEND AllGraphTextSetup;

%AllGraphTextSetup(11);
```

Because of macro defaults, this invocation is equivalent to:

```
%AllGraphTextSetup(11, Family=Arial,
                    Weight=Bold);
```

If using a font family name with embedded blanks, it must be enclosed in quotes as in

```
%AllGraphTextSetup(11, Family='Arial Narrow',
                    Weight=Bold);
```

for which the ODS style created would be GraphFontArialNarrow11ptBold.

After running the preceding macro invocation, your SAS log includes this:

```
NOTE: STYLE 'GraphFontArial11ptBold' has been
saved to: SASUSER.TEMPLAT
```

SASUSER.TEMPLAT is your default permanent template store. Styles are written there by default. Whenever you reference a style, SASUSER.TEMPLAT is where ODS Graphics will look for it by default. SASUSER.TEMPLAT is part of your SASUSER profile. If you delete the profile in the SASUSER data library, you will lose SASUSER.TEMPLAT. If not, the styles will be there permanently.

You can define template stores at other locations than SASUSER, but that is not necessary here and is outside the scope of this article. If interested, please consult SAS documentation.

You can, of course, create other custom styles. You can permanently store the preceding macro source code, %MACRO statement through %MEND statement, in any convenient location. For example, on Windows, you can create a folder such as C:\MyMacros. The filename must be AllGraphTextSetup, and the filetype must be sas. The other option is to simply paste the macro source code into your SAS code creation and submission window.

To use a stored macro, if in C:\MyMacros, your application program must start with:

```
%INCLUDE "C:\MyMacros\AllGraphTextSetup.sas";
```

If using Arial 12pt Bold, before your SG PROC step, you must use two statements:

```
%AllGraphTextSetup(12);
ODS LISTING STYLE=GraphFontArial12ptBold;

/* For macro invocation without overrides for
   Family and Weight, the style created will be
   for Arial and Bold, as done here. */

< example-specific code here >

ODS LISTING CLOSE;
```

If you wish to use, for example, Verdana 10pt Normal, your macro invocation and your ODS LISTING statement must be

```
%AllGraphTextSetup(10, Family=Verdana,
                    Weight=Normal);
ODS LISTING STYLE=GraphFontVerdana10ptNormal;
```

If you wish to use instead Verdana 10pt Bold, you can omit Weight= from the macro invocation since Bold is the default.

NOTE: Assigning an ODS Style for your graph creation code block is not a straightjacket. It only establishes a default. Any of the ATTRS options listed above, as well as the TITLE and FOOTNOTE statements, can be used for custom overrides.

For code to list all of the styles in your default template store, or to display in the SAS log the source code that was used to create a style in SASUSER.TEMPLAT, see Chapter 3 in the book “Visual Data Insights Using SAS ODS Graphics: A Guide to Communication-Effective Data Visualization”.

LeRoy Bessler, Bessler Consulting and Research

Did You Know?

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

KT – Part 6: Final components of the COVID 19 project

What this is about

In part 1 of this series I outlined how a standard framework for recurring SAS processes could be controlled in entirety with external configuration files. In parts 2 and 3 I demonstrated how the task of importing COVID 19 data into a project could be configured externally and encapsulated in individual nodeConnect processes, and in part 4 I introduced the nodeBranch macro to initiate each task in sequence, while part 5 added initial configurations and associated macros.

My implementation, KT, has 3 core processes – nodeStem, nodeBranch and nodeConnect. In this part I will introduce the nodeStem macro and additional setup statements to run the complete project.

As in previous examples, for brevity, the code for the process is a “Lite” version, without full documentation, parameter validation and error handling.

Example 5 – Initiating the project

The nodeStem process performs 2 initiation tasks: converting all configuration files to SAS tables, and then using the projectConfig specification to implement remaining initial configurations tasks.

Here the folder= parameter points to the folder containing configuration files.

```
%MACRO nodeStemLite(folder=, configlib=work
                    ,stemConfig=configProject);
/* Identify and load all configuration data
into SAS tables */

DATA _NULL_;
  LENGTH folder fileref pathid $96
          tableid $32
          execstr $256;
  folder = SYMGET('folder');
  rc = FILENAME('fref', folder);
  did = DOPEN('fref');
  dnum = DNUM(did);
  DO k = 1 TO dnum;
    fileref = DREAD(did, k);
    IF MOPEN(did, fileref) > 0 THEN DO;
      tableid = TRANSTRN(fileref, '.csv'
                        , '');
      pathid = CATX('\', folder, fileref);
      execstr =
        CATS('readConfigFileLite(configPath='
              ,pathid, ',configFile='
              ,tableid, ')');
      CALL EXECUTE(CATS('%', execstr));
    END;
  END;
  rc = DCLOSE(did);
RUN;
```

```
/* Implement the configurations listed in the
configProject table */

DATA _NULL_;
  SST &configlib..&stemConfig.;
  doConfig = CATS(parameter, '(configdata='
                  ,value, ');');
  CALL EXECUTE(CATS('%', doConfig, ');'));
RUN;
%MEND nodeStemLite;
```

Invoking the nodeStem macro and running the project

A small segment of additional information needs to be run to invoke the nodeStem macro. The project needs to know where to find its configuration files and where to find the KT macros.

It would be possible to facilitate this by using the INITSTMT= SAS system option containing the following code, and this might be useful for a production version, but for simplicity and transparency in the Lite version the following statements need to be in stand-alone code (with appropriate edits of the folder locations):

```
/* start_COVID19ConfirmedGlobalTimeSeries.sas */
%LET configbase = C:\Projects\WorldStatistics\
COVID-19\Lite\config;
%LET KTmacros = C:\Projects\KT\Lite\macros;
%INCLUDE "&KTmacros.\*.sas";

/* Compile KT macros */

%nodeStemLite(folder=&configbase.);
%nodeBranchLite(configdata=&branchConfig);
```

Invoking nodeStem and nodeBranch individually allows either of them to be commented out during development and testing (as distinct from invoking nodeBranch within nodeStem).

This starting code needs to be saved with the project, not in the KT file tree. For a different project (in another location) make a fresh copy of the start* code, update the configbase reference and save it to the new location with an appropriate change of name.

Project Documentation

Every SAS process that is intended to be run more than once needs to be well documented, but, alas, frequently is not. In a sense every SAS job is like a story, an epic or a shaggy dog or anything in between. Sometimes the plot can be traced in the source code and sometimes the log of a successful run will provide a retrospective view (“forensic documentation”).

In the case of a KT project the story in the SAS code in the starting segment amounts to “In a certain location, something happens. The end.” All the process logic has been abstracted into configuration files. With the right choice of macro facility options the log can be more informative, especially if the nodes echo the contents of all the configuration files as they are invoked. But it should not be necessary to run the code first in order to understand what it does.

In short, the requirement for good documentation is most acute with any KT project. As a starting point a helper app could be used to navigate the process flow through the configuration files and provide a skeleton for a more readable description. This would work best if the configuration files themselves contain documentation segments as well as inline comments.

It's a wrap

This KT Lite project is a working example of building a project on a generic SAS framework that is entirely agnostic of project specific inputs, processes or outputs; with all project specific information accessed from configuration files.

The KT Lite code has of necessity (focus on key concepts and space considerations in a publication) been shorn of enhancements such as test facilitation, data validation, error handling, and internal documentation. A more comprehensive version of the code is in preparation.

Richard C Carson

Book Announcements

“SAS Programming Experiences: A How-To Guide from a Power SAS User” by Philip R Holland

I have spent the last 40+ years working with SAS software, programming in the banking, insurance, human resources, database marketing, customer relations and pharmaceutical sectors. “SAS Programming Experiences” is divided into the 3 sections, with the first 2 sections looking back at the SAS environments that have influenced my SAS programming styles. The final section looks into the future at a software application that will allow me to continue SAS programming without having to pay the SAS Institute for software licences. Altair SLC is now mature enough to run the vast majority of my published SAS programs, and can be licensed for free, and Altair Analytics Workbench uses the Eclipse platform to provide a user-friendly interface.

Topics included:

- SAS Programming Methods:
 - The Art of Defensive Programming: Coping with Unseen Data
 - Writing Reusable Macros
 - SAS Hints and Tips
 - Using SAS Software for Hobbies
- SAS Interfaces for Programming:
 - The History of SAS Editors and Installs
 - The History of SAS Enterprise Guide and SAS Studio
 - Using SAS Studio Tasks to Plot with ODS Graphics
 - Making Graphs Easier to Validate using ODS Graphics
- SAS Programming Alternatives:
 - What is Altair SLC?
 - Introduction to Altair Analytics Workbench
 - Altair SLC for Clinical Trials
 - Generating Graphics with Altair SLC
 - Using Altair SLC with R and Python

This book is available in PDF and paperback formats, and can be downloaded directly, or linked to the publishers site, from <https://hollandnumerics.org.uk/wordpress/book-table/>.

Philip R Holland, Newsletter Editor

“Altair SLC: The SAS Language Compiler” by Philip R Holland

It is possible that you have no interest in the history of SAS programming, as documented in “SAS Programming Experiences”, but need to know more about Altair SLC. If that is the case, then you can buy my second book, which is a subset of “SAS Programming Experiences”, and only contains the chapters related to Altair SLC and Altair Analytics Workbench, with information on how to access Altair SLC from Microsoft Visual Studio Code Editor.

Topics included:

- What is Altair SLC?
- Introduction to Altair Analytics Workbench
- Altair SLC for Clinical Trials
- Generating Graphics with Altair SLC
- Using Altair SLC with R and Python

This book is available in PDF and paperback formats, and can be downloaded directly, or linked to the publishers site, from <https://hollandnumerics.org.uk/wordpress/book-table/>.

Philip R Holland, Newsletter Editor

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 format **ZDw.d** converts numeric values to EBCDIC zoned decimal values, where each byte represents a digit, with the sign of the value being incorporated into the last digit. Informats **ZDw.d**, **ZDBw.d** and **ZDVw.d** read EBCDIC zoned decimal values into SAS numeric variables. **ZDVw.d** validates the input data, and **ZDBw.d** expects blanks (from IBM 1410, 1401, and 1620), instead of zeroes.

[SAS 9.4M4] The **LIBNAME** statement can now read and write JSON files using the **JSON** engine.

[SAS 9.4M5] Use the **%COPY_TO_NEW_ENCODING()** macro function to convert data sets to a new encoding. **%COPY_TO_NEW_ENCODING()** must be run on a system that matches the source-data encoding.

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 **40,700** 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 21,700 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 have been holding regular meetings in London, online, and also occasional meetings in Bristol, Dublin, Edinburgh, Leeds and Manchester. For more details and to register for future events please go to meetup.com/SUGUKI.
- Fancy joining the new SAS Global Analytics Explorers hub, then please fill in your **name**, **company email address** and **company** at <https://infl.tv/npmv>.
- Altair Analytics Workbench Community Edition provides named users with free access to Altair SLC software (formerly WPS) including a SAS language compiler with coding and workflow tools on unlimited data volumes, with interfaces to a wide range of databases, as well as integrated R and Python programming environments. A Community Edition or Personal Edition licence key will be issued that will activate the software for up to 12 months. At the end of this term, users may be invited to renew their licence and/or to upgrade to the latest available full version of the software. Download the software from www.altair.com/altair-slc. No technical support service is offered with the Community Edition, but assistance can be requested using the Community forums at community.altair.com/community. The current General Access version of Altair SLC is 5.24.1 (2024).
- Do you use Microsoft Visual Studio Code Editor?
 - You may be interested in a SAS-related extension called **SASjs - DevOps for SAS**. You can develop SAS applications and run code on SAS Viya, SAS 9 EBI or SASjs Server! Just set up the target, and click the running man to execute your SAS code. The SASjs Server could be your local SAS installation. SASjs is an open source project, and can be obtained from the Visual Studio Code Marketplace, where the current version is v2.6.0.
 - The SAS Institute has also been developing its own extension called **SAS**. You can develop SAS applications and run code on local SAS installations, SAS Viya, or SAS OnDemand for Academics! Just set up the target, and click the running man to execute your SAS code. The extension can be obtained from the Visual Studio Code Marketplace, where the current version is v1.10.0.
 - Another Microsoft Visual Studio Code Editor extension has just been released called **Altair SLC**. This extension allows you to connect to local or remote Altair SLC installations, even if, until now, you have only been able to use Altair Analytics Workbench to access Altair SLC. The current version in the Visual Studio Code Marketplace is v0.7.0 (pre-release).

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.

November 2024

10-13 **PhUSE EU Connect**, Strasbourg, France

The PHUSE Community allows you to enhance your learning and gain insights from our experts on today's most critical industry topics through a variety of engaging presentations, workshops, and networking opportunities. Individuals from all areas of the industry will be gathered in one place, providing an opportunity not only for you to advance your career but also to heighten the pharmaceutical industry's influence and findings. For full details please visit www.phuse.eu.

16-19 **MWSUG**, Waukesha, WI, USA

This will be the first post-pandemic conference of the MidWest SAS Users Group, which has been serving users in 12 states in the mid-western region of the United States since 1990. It provides educational opportunities and resources, and facilitates communication amongst SAS users and SAS. Full details can be found at www.mwsug.org.

March 2025

16-19 **PhUSE US Connect**, Orlando, FL, USA

The PHUSE Community allows you to enhance your learning and gain insights from our experts on today's most critical industry topics through a variety of engaging presentations, workshops, and networking opportunities. Individuals from all areas of the industry will be gathered in one place, providing an opportunity not only for you to advance your career but also to heighten the pharmaceutical industry's influence and findings. Full details at www.phuse.eu.

May 2025

6-9 **SAS Innovate**, Orlando, FL, USA

Whether you're a business leader, technical user or SAS Partner, SAS Innovate is a no-risk, all-reward opportunity to learn how to succeed with data and AI. Full details can be found at www.sas.com/en_us/events.

Contacts

LeRoy Bessler	Email: besslerleroy2024@gmail.com
Richard C Carson	Email: richard@arcol-analytics.com
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.