

VIEWS

International SAS Programmer Community

News, Hints, Tips and Information
for SAS® Users

Issue 72, 2nd Quarter 2023

Even though Foundation SAS is being pushed to one side by SAS Viya, this newsletter continues to bring you information about making the most of SAS 9. This newsletter issue includes another way to draw butterfly plots from LeRoy Bessler, I asked ChatGPT to write a SAS program, and Richard Carson continues his discussion about configuring of SAS processes without editing them in the second part of a series of articles about KT. 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.

Butterfly Plot Using ODS Graphics

Q: How do you create a butterfly plot?

A: [SAS 9.4] In VIEWS News 70, Philip R Holland showed how to draw a butterfly plot, using Graphic Template Language (GTL). It was a two-side bi-directional stacked horizontal bar chart, with data labels inside the bar segments.

Phil had concerns about the asymmetric width of the two wings of the butterfly plot and the order of the legend entries. The alternative shown here has equal-sized wings (but with a more informative “butterfly body” between them). Legend order becomes irrelevant when a legend substitute is provided here.

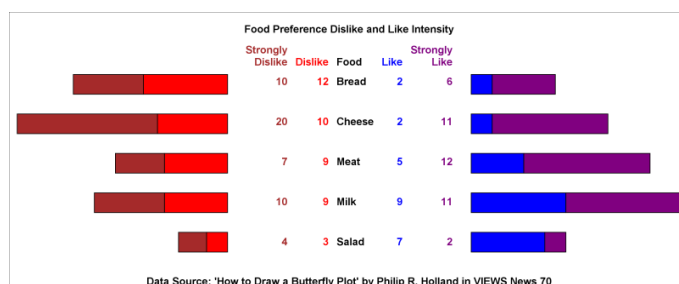
For me, the paramount concern is that data labels with more digits might not fit inside bar segments. This is a common

problem for standard stacked bar charts, whether horizontal or vertical, not just a possibility for butterfly plots.

One of the most important design principles in my new book “Visual Data Insights Using SAS ODS Graphics: A Guide to Communication-Effective Data Visualization” is that though visuals are the key to quick, easy inference, the associated precise numbers are essential for correct, reliable inference.

I wanted a butterfly plot that can present the precise values even for bigger numbers. So I decided to use ODS Graphics, rather than Graph Template Language as Phil did. Both of his concerns are addressed. For the examples here the same one- and two-digit values are used as input, but the solution is immune to the fit problem for higher-digit values.

Below is the ODS Graphics alternative to the GTL butterfly plot.



I would have liked to use Light Red and Light Blue, as the colour codes for the Dislike and Like columns of the table (with Red and Blue for the Strong intensities), but Light Red would have been more difficult to read.

The image output from ODS PRINTER cannot directly be provided a border. So, post-processing the image file with the ApplyBorderToAnyImage macro is used to provide a border for the composite. You can use this macro for any image, not just the composite created in this article.

For this butterfly plot a challenge is to align the rows of the table with the bars of the bar charts. As you can see in the examples, the alignment is imperfect, but it is sufficient to enable the viewer to associate the values with their bars.

The CELLHEIGHT for ROWHEADER and DATA in the PROC TEMPLATE code to create the ODS style ToBeDeleted is adjusted to control row spacing to align the table rows with the bar chart bars.

By colour-coding the columns, their column labels identify the bar segments and the legend can be eliminated. When a legend is unnecessary, the image viewer can interpret the visual with less eye travel and effort, always a benefit.

Here is the code used for this example:

```
ODS RESULTS OFF; /* nothing to OUTPUT window */
```

```

DATA work.Input;
  INFILE DATALINES DLM = ',';
  INPUT Food $
         Dislike DislikeStrongly Like LikeStrongly;
  DATALINES;
Bread, 12, 10, 2, 6
Cheese, 10, 20, 2, 11
Meat, 9, 7, 5, 12
Milk, 9, 10, 9, 11
Salad, 3, 4, 7, 2
RUN;

DATA work.InputWithGroups;
  KEEP Food DislikeIntensity LikeIntensity Count
        DislikeStrongly Dislike Like LikeStrongly;
  LENGTH DislikeIntensity LikeIntensity $ 8;
  SET work.Input;
  LikeIntensity = ' ';
  DislikeIntensity = 'Strongly';
  Count = 0 - DislikeStrongly;
  OUTPUT;
  DislikeIntensity = 'Dislike';
  Count = 0 - Dislike;
  OUTPUT;
  DislikeIntensity = ' ';
  LikeIntensity = 'Strongly';
  Count = LikeStrongly;
  OUTPUT;
  LikeIntensity = 'Like';
  Count = Like;
  OUTPUT;
RUN;

```

Prepare discrete attribute maps.

```

DATA work.ButterFlyChartDatAttrMapLeft;
  RETAIN ID 'GroupColors';
  LENGTH ID $ 11 Value $ 15 FillColor $ 8;
  Value = 'Strongly'; FillColor = 'Brown';
  OUTPUT;
  Value='Dislike'; FillColor='Red';
  OUTPUT;
RUN;

DATA work.ButterFlyChartDatAttrMapRight;
  RETAIN ID 'GroupColors';
  LENGTH ID $ 11 Value $ 15 FillColor $ 8;
  Value = 'Like'; FillColor = 'Blue';
  OUTPUT;
  Value = 'Strongly'; FillColor = 'Purple';
  OUTPUT;
RUN;

```

NOTE: Since the image might require code tuning to get to a satisfactory result, gather controls that are likely to need change up here to the top of the code.

```

%LET FontFamily = Arial;
%LET FontPointSize = 11;
%LET FontWeight = Bold;

```

In the %LET statements below, the controls for common title and footnote text that is inlaid with PROC ODSTEXT and the P statement are different from those used on TITLE and FOOTNOTE statements, which are used for PROC PRINT & PROC SGPLOT.

The following controls are identical, but they could be different:

```

%LET CommonTitleAttrs =
  FONTFAMILY = "&FontFamily."
  FONT_SIZE = &FontPointSize.pt
  FONT_WEIGHT = &FontWeight.
  JUST = CENTER COLOR = Black;

%LET CommonFootnoteAttrs =
  FONTFAMILY = "&FontFamily."
  FONT_SIZE = &FontPointSize.pt
  FONT_WEIGHT = &FontWeight.
  JUST = CENTER COLOR = Black;

%LET CommonTitle = %STR(Food Preference Dislike
and Like Intensity);

%LET CommonFootnote = %STR(Data Source: 'How to
Draw a Butterfly Plot' by Philip R. Holland in
VIEWS News 70);

PROC TEMPLATE;
  DEFINE STYLE ToBeDeleted;
    PARENT=Styles.PRINTER;
    CLASS GraphFonts /
      'GraphValueFont' = ("&FontFamily.",
        &FontPointSize.pt, &FontWeight.)
      'GraphLabelFont' = ("&FontFamily.",
        &FontPointSize.pt, &FontWeight.)
      'GraphDataFont' = ("&FontFamily.",
        &FontPointSize.pt, &FontWeight.)
      'GraphTitleFont' = ("&FontFamily.",
        &FontPointSize.pt, &FontWeight.)
      'GraphFootnoteFont' = ("&FontFamily.",
        &FontPointSize.pt, &FontWeight.)
    ;
    /* Classes below are for non-graph output */
    CLASS Fonts /
      'TitleFont' = /*table titles footnotes*/
        ("&FontFamily.", &FontPointSize.pt,
        &FontWeight.)
      'HeadingFont' = /*column and row headers*/
        ("&FontFamily.", &FontPointSize.pt,
        &FontWeight.)
      'DocFont' = /*table cells*/
        ("&FontFamily.", &FontPointSize.pt,
        &FontWeight.)
    ;
    CLASS HEADER /
      BACKGROUND_COLOR = White COLOR = Black;
    CLASS ROWHEADER /
      BACKGROUND_COLOR = White COLOR = Black;
    CLASS DATA /
      BACKGROUND_COLOR = White COLOR = Black;
    CLASS TABLE / RULES=NONE FRAME=VOID;
    CLASS ROWHEADER / CELLHEIGHT = 42pt;
    CLASS DATA / CELLHEIGHT = 42pt;
    /* 42pt aligns table rows with bars */
  END;
RUN;

OPTIONS CENTER NONUMBER NODATE;
OPTIONS PAPER_SIZE=(3.34in 3.5in);
ODS _ALL_ CLOSE;
ODS PRINTER STYLE=ToBeDeleted DPI=300
  FILE="C:\temp\Table.png" PRINTER=PNG300;
TITLE1;

PROC PRINT DATA=work.Input NOOBS LABEL;
  VAR DislikeStrongly / STYLE(HEADER) =
    [COLOR=Brown] STYLE(DATA)=[COLOR=Brown];

```

```

VAR Dislike / STYLE(HEADER) =
  [COLOR=Red] STYLE(DATA)=[COLOR=Red];
VAR Food / STYLE(HEADER) =
  [COLOR=Black] STYLE(DATA)=[COLOR=Black];
VAR Like / STYLE(HEADER) =
  [COLOR=Blue] STYLE(DATA)=[COLOR=Blue];
VAR LikeStrongly / STYLE(HEADER) =
  [COLOR=Purple] STYLE(DATA)=[COLOR=Purple];
LABEL
  DislikeStrongly = 'Strongly Dislike'
  Dislike = 'Dislike'
  Like = 'Like'
  LikeStrongly = 'Strongly Like';
RUN;

ODS PRINTER CLOSE;

PROC TEMPLATE;
  DELETE ToBeDeleted / STORE=SASUSER.TEMPLAT;
RUN;

```

Defaults are DATE NUMBER CENTER.

```

OPTIONS NODATE NONUMBER;

```

Composite WIDTH and HEIGHT.

```

OPTIONS PAPER SIZE = ("10in" "4.1in");
ODS _ALL_ CLOSE;

```

ODS PRINTER creates the composite image. ODS LAYOUT places the right items in the right places.

```

ODS PRINTER PRINTER = PNG DPI = 300
  FILE = "C:\temp\NoBorder.png";
ODS LAYOUT ABSOLUTE;

```

Nullify any leftover TITLE or FOOTNOTE statements from earlier in the SAS session.

```

TITLE; FOOTNOTE;

```

Inlay a common title.

```

ODS REGION X = 0in y = 0in
  WIDTH = 10in HEIGHT = 0.6in;
PROC ODSTEXT;
  P " " / STYLE={FONT_SIZE = 0.1in}; /*space*/
  P "&CommonTitle." /
    STYLE={&CommonTitleAttrs.};
RUN;

```

Set up for BOTH images:

```

ODS GRAPHICS / RESET = all NOBORDER
  SCALE=OFF WIDTH = 3.33in HEIGHT = 3.5in;
ODS REGION X = 0in Y = 0.35in
  WIDTH = 3.33in HEIGHT = 3.5in;
ODS GRAPHICS / RESET = INDEX;

```

No title needed, but preserves the vertical alignment of top bar with top of table.

```

TITLE1 JUSTIFY = LEFT FONT = &FontFamily.
  &FontWeight. HEIGHT = &FontPointSize.pt ' ';
PROC SG PLOT DATA = work.InputWithGroups
  NOBORDER NOAUTOLEGEND
  DATTRMAP = work.ButterFlyChartDattrMapLeft;

```

```

HBAR Food / RESPONSE = Count Y2AXIS
  ATTRID = GroupColors BARWIDTH = 0.5
  GROUP = DislikeIntensity
  GROUPDISPLAY = STACK
  DISPLAYBASELINE = OFF FILL FILLTYPE = SOLID
  OUTLINE OUTLINEATTRS = (COLOR = Black);
Y2AXIS DISPLAY = NONE;
XAXIS DISPLAY = NONE;
FORMAT Count 2.;
RUN;

ODS REGION X = 6.67in Y = 0.35in
  WIDTH = 3.33in HEIGHT = 3.5in;
ODS GRAPHICS / RESET = INDEX;
TITLE1 JUSTIFY = LEFT FONT = &FontFamily.
  &FontWeight. HEIGHT = &FontPointSize.pt ' ';
PROC SG PLOT DATA = work.InputWithGroups
  NOBORDER NOAUTOLEGEND
  DATTRMAP = work.ButterFlyChartDattrMapRight;
HBAR Food / RESPONSE = Count
  ATTRID = GroupColors BARWIDTH = 0.5
  GROUP = LikeIntensity GROUPDISPLAY = STACK
  DISPLAYBASELINE = OFF FILL FILLTYPE = SOLID
  OUTLINE OUTLINEATTRS = (COLOR = Black);
YAXIS DISPLAY = NONE;
XAXIS DISPLAY = NONE;
FORMAT Count 2.;
RUN;

```

Insert the image of the table between the two bar charts, below 0.35 inches of space that includes the common title.

```

ODS REGION X = 3.33in y = 0.35in
  WIDTH = 3.34in HEIGHT = 3.5in;

```

Use the SAS Report Writing Interface:

```

DATA _NULL_;
  DCL ODSOUT OBJ();
  OBJ.IMAGE(FILE: "C:\temp\Table.png");
RUN;

```

Inlay a common footnote.

```

ODS REGION X = 0in Y = 3.85in
  WIDTH = 10in HEIGHT = 0.25in;
PROC ODSTEXT;
  P "&CommonFootnote." /
    STYLE={&CommonFootnoteAttrs.};
  /* P "    &CommonFootnote." /
    STYLE = {ASIS = ON &CommonFootnoteAttrs.};
  If using JUST = LEFT in CommonFootnoteAttrs,
  ASIS = ON preserves the three leading blanks
  for an indent, such as three spaces here. */
RUN;

ODS LAYOUT END;
ODS PRINTER CLOSE;

```

Defining the %ApplyBorderToAnyImage macro:

```

%MACRO ApplyBorderToAnyImage(in=, out=,
                             height=, width=);

  ODS RESULTS OFF;
  ODS _ALL_ CLOSE;
  TITLE; FOOTNOTE;

  OPTIONS CENTER NONUMBER NODATE;
  OPTIONS PAPER SIZE = (&width. &height.);
  GOPTIONS DEVICE = PNGT TRANSPARENCY;

```

```

ODS PRINTER PRINTER = PNG300 DPI = 300
FILE = "&out.";
ODS LAYOUT ABSOLUTE;
ODS REGION X = 0in Y = 0in
WIDTH = &width. HEIGHT = &height.;

DATA _NULL_;
DCL ODSOUT OBJ();
OBJ.IMAGE(FILE: "&in.");
RUN;

ODS REGION X = 0in Y = 0in
WIDTH = &width. HEIGHT = &height.;

PROC GSLIDE BORDER;
RUN; QUIT;

ODS LAYOUT END;
ODS PRINTER CLOSE;
%MEND ApplyBorderToAnyImage;

```

Call the %ApplyBorderToAnyImage macro:

```

%ApplyBorderToAnyImage(in=C:\temp\NoBorder.png,
out=C:\temp\ButterFlyPlotImbeddedTableNoLegend.png,
width = 10in,
height = 4.1in);

```

NoBorder.png and Table.png can be deleted.

The code used to create this composite uses machinery that can be adapted for much richer creations, which cannot be delivered with simpler code. The method can be adapted to deliver Anything Anywhere in a Composite Image, i.e. any collection of graphs, tables, and text (and, if desired, non-graph images) arranged any way that you want it. See, for example, Figure 13-7 in my book for a different example. With the methods demonstrated there and in this article you have the tools to deliver any composite for reporting, or to create a complex combination of visuals and text that has come to be called an Infographic.

LeRoy Bessler, Visual Data Insights™

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 2 Integrating tasks with nodeConnect

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. The advantages of this approach are that the framework is robust and changes to the process can be made without needing to edit its SAS code.

My implementation, KT, has 3 core processes: nodeStem, nodeBranch and nodeConnect. In this part I will describe a typical process task, importing data, and how this can be configured in nodeConnect. Later I will show how multiple instances of nodeConnect and different tasks can be controlled through nodeBranch. Finally I will show how the project as a whole is configured using nodeStem.

Example 1: Importing global COVID 19 confirmed case data

The data source is derived from the 2019 Novel Coronavirus Visual Dashboard operated by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE), as

transposed by the (UN) Humanitarian Data Exchange (HDX) at <https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases?>. The transposed data adds ISO 3166 alpha-3 country and region codes, and is downloadable as a CSV file.

Assuming the file has been downloaded the task is to import the data into a SAS table. The problem with the CSV data interchange file format is that all metadata other than the column name is not preserved. PROC IMPORT and the Enterprise Guide file import process, for example, use in-process discovery to rectify this omission, but the results may require editing and the process is inefficient for recurring projects. This demonstration uses PROC IMPORT, but it is more than 100 times slower than a Data Step making use of metadata derived from a pre-process discovery.

work.covid19confirmed_importdelimited: this is the configuration table containing process data for the CSV file. Note the values of **datarow** and **guessingrows** differ from the default values for PROC IMPORT because there is a second header row, and because of the number of initial rows with null values in some columns, and the length of some country names near the bottom of the data file.

Section	Parameter	Value
process	macroname	importDelimitedTextFile
process	configlib	work
process	configdata	covid19confirmed_importdelimited
sourcefile	folder	C:\Projects\WorldStatistics\COVID-19\
sourcefile	filespec	time_series_covid19_confirmed_global_narrow.csv
structure	dbms	CSV
structure	delimiter	'
structure	getnames	YES
structure	datarow	3
structure	guessingrows	MAX
outdata	tablename	covid19_confirmed_global
outdata	library	work
outdata	overwrite	REPLACE

Here **configlib** and **configdata** are input parameters for nodeConnect and are thus redundant.

The nodeConnect step uses the configuration table to identify the task macro and the location of the configuration data, which is passed on to the task macro. The remaining configuration values are consumed within the macro. This is a 'Lite' version of nodeConnect, shorn of its parameter validation and error handling duties. Note the macro call **%¯oname** is not a misprint.

```

%MACRO nodeConnectLite(configlib=, configdata=);
  %* Extract task macro name;
  DATA _NULL_;
    SET &configlib.&configdata.;
    WHERE UPCASE(parameter) = 'MACRONAME';
    CALL SYMPUTX('macroname', TRIM(value), 'L');
  RUN;

  %* Call macro and pass config parameters;
  %&macroname.(configlib=&configlib.
               ,configdata=&configdata.);
%MEND nodeConnectLite;

```

Here is the delimited text demo macro.

```
%MACRO importDelimitedTextFile(configlib=
                                ,configdata=);
  %LOCAL folder filespec dbms delimiter getnames
  dataRow guessingrows overwrite
  tablename library;

  %* Store inidata file parameters as macro
  variables;
  DATA _NULL_;
    SET &configlib.&configdata.;
    WHERE UPCASE(section)
      IN ('SOURCEFILE' 'STRUCTURE' 'OUTDATA');
    CALL SYMPUTX(parameter, value, 'L');
  RUN;

  %* Import the data;
  PROC IMPORT DATAFILE=&folder.\&filespec."
    DBMS=&dbms.
    OUT=&library.&tablename.
    &overwrite.;
    GUESSINGROWS=&guessingrows.;
    DELIMITER=&delimiter.;
    GETNAMES=&getnames.;
    DATAROW=&datarow.;
  RUN;
%MEND importDelimitedTextFile;
```

Copy the table above to a text file `covid19confirmed_importdelimited.txt` and in EG use the file import task, in the interim, or use PROC IMPORT with DBMS=TAB. Assuming the configuration data has been loaded to `work.covid19confirmed_importdelimited` the task to extract the data can be invoked as follows:

```
%nodeConnectLite(configlib=work
,configdata=covid19confirmed_importdelimited);
```

This will be the statement issued from nodeBranch as one step in the project.

Richard C Carson

Is ChatGPT going to replace Programmers?

Request:

Can you write a SAS program to draw a heart shape?

ChatGPT:

Yes, here is an example SAS program to draw a heart shape using the SGPLOT procedure:

```
/*Create a SAS dataset with x and y coordinates
of heart shape*/
data heart;
  pi = constant('pi');
  do t = 0 to 2*pi by 0.01;
    x = 16*sin(t)^3;
    y = 13*cos(t) - 5*cos(2*t) - 2*cos(3*t)
      - cos(4*t);
    output;
  end;
run;
```

```
/*Plot the heart shape using SGPLOT procedure*/
proc sgplot data=heart noautolegend;
  scatter x=x y=y /
    markerattrs=(symbol=heart filledcolor=red
    size=100);
  xaxis display=none;
  yaxis display=none;
run;
```

The program creates a SAS dataset called "heart" which contains x and y coordinates of the heart shape. The heart shape is generated using parametric equations for a heart shape. The SGPLOT procedure is used to plot the heart shape by creating a scatter plot with the heart symbol as markers. The x-axis and y-axis are hidden to create a clean heart shape.

Does it work?:

[SAS 9.4M8] The code looks like a SAS program, and the formula for a heart shape is broadly correct, but the program fails to run because of errors:

```
32      x = 16*sin(t)^3;
                                22
ERROR 22-322: Syntax error, expecting one of the
following: <, <=, =, >, >=, EQ, GE, GT, LE, LT, NE,
NG, NL, ^=, ~=.
```

ChatGPT is hallucinating, and borrowing syntax from another programming language! In SAS ^ means "Not", so the SAS parser is looking for another operator, whereas ** in SAS means "To The Power Of", which was what was needed.

Replacing ^ with ** exposes another failure:

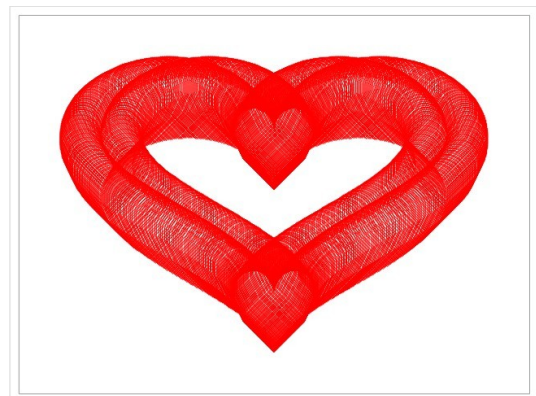
```
42      markerattrs=(symbol=heart filledcolor=red
                                22
ERROR 22-322: Syntax error, expecting one of the
following: ), COLOR, SIZE, SYMBOL.
43      size=100);
```

ChatGPT is hallucinating again! In ODS Graphics there is no FILLED COLOR= option.

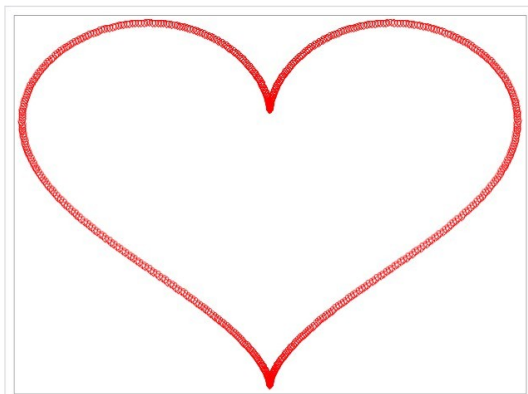
Replacing FILLED COLOR= with COLOR= exposes a NOTE about unsupported syntax. This begs the question: Why would a symbol shaped like a heart be selected to outline a heart shape?

NOTE: SYMBOL=HEART is not fully supported, and may not work well in all cases.

Also the generated image is too large to display inside Enterprise Guide, and has to be viewed in an external browser:



Changing SIZE=100 to SIZE=10 finally gives an image that can be viewed in Enterprise Guide:



Conclusions:

- ChatGPT can write SAS programs, but does not know how to write syntactically correct code.
- Weird assumptions and hallucinations can affect the generated programs, and may not make them fail, but could impact their output.
- Never trust a SAS program written by ChatGPT until it has been fully tested!
- I believe human SAS programmers are safe (for now!), because they will, at least, be required to validate any generated code.

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 **\$BINARYw.** displays character values in binary representation. The **BINARYw.** format converts numeric values to binary representation.

[WPS 4] There are a number of options related to **PROC PYTHON** and **PROC R** that control whether the environment in the procedure is retained or deleted when the procedure ends. The system options **PYTHONKEEP** and **RKEEP** retain the environment, and **NOPYTHONKEEP** and **NORKEEP** delete the environment. However, the **KEEP** and **TERMINATE** options of **PROC PYTHON** and **PROC R** override the system options.

[SAS 8.1] The functions **HARMEAN()** and **HARMEANZ()** returns the harmonic mean of the non-null or nonmissing values. **HARMEANZ()** does not fuzz the values of the arguments that are approximately zero.

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 39,600 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 18,100 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. 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.
- Do you use Microsoft Visual Studio Code Editor? You may be interested in a SAS-related extension called **SASjs**. 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.
- 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 R and Python programming environments. A Community 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 5.23.0 (2023).

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.

September 2023

11-14 **SAS Explore**, Las Vegas, NV, USA

Immerse yourself in four days of jam-packed learning alongside thousands of like-minded technology peers. Learn best practices and see real-world examples of how AI is transforming the way we work. More details at explore.sas.com.

November 2023

5-8 **PhUSE EU Connect**, Birmingham, UK

PhUSE Connect events bring together our global Community to collaborate on hot topics, share perspectives and discuss different approaches to guiding the healthcare industry landscape, both now and in the future. Full details at www.phuse-events.org.

Contacts

LeRoy Bessler, Visual Data Insights™

Email: Le_Roy_Bessler@wi.rr.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.*