

VIEWS

International SAS Programmer Community

News, Hints, Tips and Information
for SAS® Users

Issue 74, 4th Quarter 2023

Supporting all Foundation SAS programmers, this newsletter continues to bring you information to make the most of SAS 9, which, according to SAS Explore 2023, still has a long life ahead of it. This newsletter issue includes examples of how to make heat maps usable from LeRoy Bessler, I have asked ChatGPT to write another SAS program and Richard Carson continues his discussion about configuring of SAS processes without editing them in the fourth part of a series of articles about KT. As usual I have publicised some more SAS-related events for the next few months, and I have also added 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.

How to Make a Heat Map Usable

Q: Can heat maps be made more usable?

A: [SAS 9.4] Graphs are visual for quick, easy inference, but need precise numbers for correct, accurate inference.

While writing my book “Visual Data Insights Using SAS ODS Graphics: A Guide to Communication-Effective Data Visualization” based on my principles for graphic design and colour use, I had to get better acquainted with heat maps.

I had previously avoided them. Heat maps normally use a continuous colour gradient legend. The SAS default is Red-White-Blue, but you CAN use a single-hue light-to-dark gradient.

Counts of Car Origin and Type

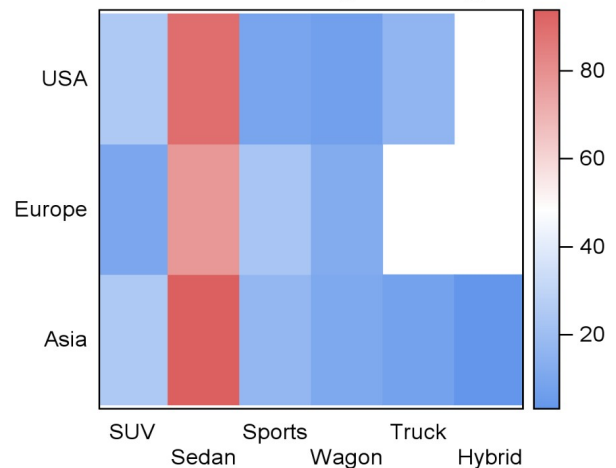


Figure 1. Uninformative Misleading Default.

The Red-White-Blue legend (a default) is equivalent to the option `COLORMODEL=ThreeColorRamp`. The empty cells can be construed to have the value of the legend's unlabelled White colour. No outline between heat map cells is a default.

Here is code used for Figure 1:

```
ODS _ALL_ CLOSE;
TITLE;
FOOTNOTE;
ODS LISTING GPATH="C:\TEMP" DPI=300;
ODS GRAPHICS / RESET=ALL SCALE=OFF
                WIDTH=3.65IN HEIGHT=3.5IN
                IMAGENAME="Figure_1";

TITLE1 JUSTIFY=CENTER
  "Counts of Car Origin and Type";
FOOTNOTE1 JUSTIFY=LEFT COLOR=Red BOLD
  "Figure 1. Uninformative Misleading Default.";

PROC SGPLOT DATA=SASHELP.Cars;
  HEATMAP X=Type Y=Origin;
  /*No outline between heat map cells is a
  default.*/
  XAXIS DISPLAY=(NOTICKS NOLABEL NOLINE)
    FITPOLICY=STAGGER;
  YAXIS DISPLAY=(NOTICKS NOLABEL NOLINE);
  GRADLEGEND / NOTITLE;
RUN;

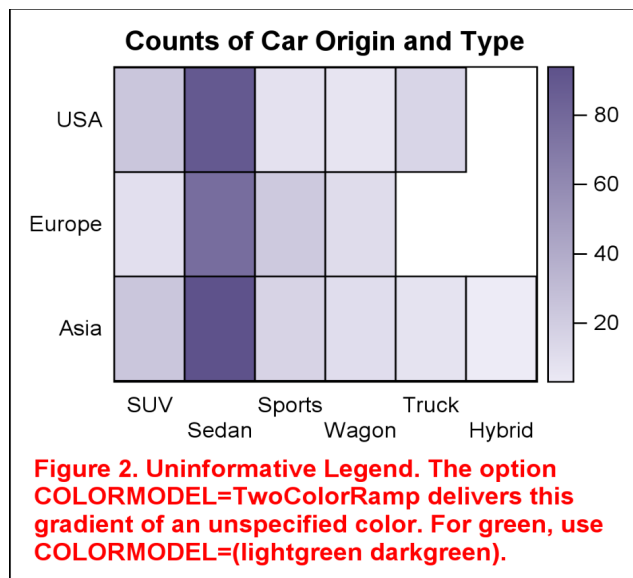
ODS LISTING CLOSE;
TITLE; FOOTNOTE;
```

When the `COLORRESPONSE` option is not used on the `HEATMAP` statement, default processing determines the frequency count, and uses that to colour the heat map.

If you DO specify `COLORRESPONSE`, then that variable's values are used to colour the heat map.

Frequency counts created by default in the procedure step and any in-step-computed statistics are not available to be used to annotate the cells.

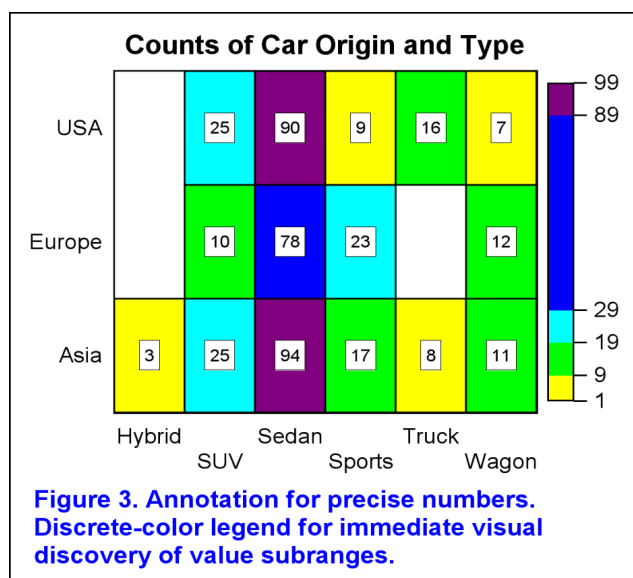
Any variable to be used for annotation, as in the case of Figure 3, must be in the input to the SGPLOT procedure step.



For Figure 2, change or omit the FOOTNOTE statement and replace the HEATMAP statement with:

```
HEATMAP X=Type Y=Origin /
  OUTLINE /*between and around cells*/
  COLORMODEL=TwoColorRamp;
```

In both Figure 1 and Figure 2, you cannot find and exactly match a heat map cell colour in the gradient of colours. Also, you cannot know the value for each colour in the gradient because the legend cannot fit a label for each colour. A continuous-colour gradient legend such as either of these is useless.



When cells are few enough and image dimensions big enough, annotation is the obvious solution to this problem, but colour coding can still be useful if done with discrete colours,

rather than a continuous colour gradient. You can use a discrete attribute map rather than the COLORMODEL option.

Here is code used for Figure 3:

```
/*Make frequency counts available as input
for the HEATMAP and TEXT statements.*/
PROC FREQ DATA=SASHELP.CARS NOPRINT;
  TABLES TYPE * ORIGIN / OUT=WORK.CarsFreq;
RUN;

/*Create the Range Attribute Map*/
DATA WORK.RattrMap_FiveColors;
  RETAIN ID "RangeID";
  LENGTH MIN MAX $ 2 COLOR $ 8;
  INPUT MIN $ MAX $ COLOR $;
  /*The INPUT statement ignores everything
  after COLOR variable value. You can use
  that area for notes.*/
  DATALINES;
1 9 CXFFFF00 Yellow
9 19 CX00FF00 Green
19 29 CX00FFFF Cyan (aka Aqua or Turquoise)
29 89 CX0000FF Blue
89 99 Purple
RUN;
```

Change or omit the FOOTNOTE statement and replace the PROC and HEATMAP statements with:

```
PROC SGPLOT DATA=WORK.CarsFreq
  /*omitting COLORMODEL=*/
  RATTRMAP=WORK.RattrMap_FiveColors;
HEATMAP X=Type Y=Origin / OUTLINE
  RATTRID=RangeID
  COLORRESPONSE=Count;
```

RangeID is the key to relevant information in WORK.RattrMap. Since an Attribute Map data set may contain multiple maps, an ID must be provided for each, even when there is only one.

For annotation, include this statement or after the HEATMAP statement:

```
TEXT X=TYPE Y=ORIGIN TEXT=Count /
  STRIP BACKFILL FILLATTRS=(COLOR=White)
  OUTLINE;
```

For other heat map examples, see Chapter 6 of my book.

LeRoy Bessler, Visual Data Insights™

LeRoy's book "Visual Data Insights Using SAS ODS Graphics: A Guide to Communication-Effective Data Visualization" can be bought and downloaded from Apress, Amazon and all good book resellers.

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 4: Managing task sequences in 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 part 2 I demonstrated how the

task of importing COVID 19 data into a project could be configured externally and encapsulated in a **nodeConnect** process, and in part 3 I added a second task and its configuration for **nodeConnect**.

My implementation, KT, has 3 core processes: **nodeStem**, **nodeBranch** and **nodeConnect**. In this part I will describe how the project tasks, with their individual instances of **nodeConnect** and associated configurations, are initiated by **nodeBranch**. Finally I will show how the project as a whole is configured using **nodeStem**.

Example 3 – Managing task sequences for global COVID 19 confirmed case data

In this demonstration so far we have created 2 tasks:

Short description	Configuration table	Task macro
Import COVID-19 data	covid19confirmed_import delimited	importDelimitedTextFile
Extract dimensions	covid19confirmed_extract Dimens	extractDimensionDataLite

The **nodeBranch** process converts this information into calls to **nodeConnect**, as many as required, with the required configuration for each instance. Here is the **nodeBranch** configuration in table covid19_branchConfigLite, using importData and extractDimens as aliases for the tasks:

Section	Parameter	Value
taskOrder	importData	1
taskOrder	extractDimens	2
importData	configdata	covid19confirmed importdelimited
importData	configLib	work
extractDimens	configdata	covid19confirmed extractDimens
extractDimens	configLib	work

To run the macro the configuration data in the table above must be imported to a table covid19_branchConfigLite.

Note: If you copy and paste the table from a PDF document into a text file, the row values will be transposed but this code will read it (change location to suit):

```
DATA covid19confirmed_extractDimens;
  INFILE "C:\Temp\covid19confirmed_extractDimens.dat"
  FLOWOVER FIRSTOBS = 2;
  LENGTH Section Parameter Value $64;
  INPUT Section Parameter Value &;
RUN;
```

For a CSV file the infile options are **DSD DLM=''**, **MISOVER FIRSTOBS=2**. Tab delimited files use the same options with the exception **DLM='09'X**.

As in previous examples for brevity the code for the **nodeBranch** process is a “Lite” version, without error handling and with the configuration data pre-loaded in SAS tables rather than interpreted from a text file.

Note that **nodeBranch** does not need to know the details about each task, just the name of its configuration data. It will use the taskOrder section to order the tasks correctly.

The code for nodeBranchLite is:

```
%MACRO nodeBranchLite(configdata =,
                      configlib = work);
  %*Retrieve task names and order and form the
  sequence of tasks;
```

```
PROC SQL NOPRINT;
  CREATE TABLE taskOrder AS
    SELECT (INPUT(value, BEST.)) AS taskOrder
      ,parameter AS task
    FROM   &configlib.&configdata.
    WHERE  UPCASE(section) = 'TASKORDER'
  ;

  SELECT DISTINCT taskOrder
    INTO   :taskSequence SEPARATED BY '~'
  FROM   taskOrder
 ORDER BY
      taskOrder
  ;

  %*Retrieve individual task configuration
  file data;
  CREATE TABLE configTasks AS
    SELECT tsk.taskOrder
      ,tsk.task
      ,cfg.parameter
      ,cfg.value
    FROM   taskOrder tsk
      ,&configlib.&configdata. cfg
    WHERE  tsk.task = cfg.section
    ORDER BY
      tsk.taskOrder
      ,cfg.parameter
  ;

  QUIT;

  %LET counter = 1;

  %*Iterate through tasks in sequence;
  %DO %UNTIL (&taskId. = 0);
    %LET taskId = %SCAN(%BQUOTE(&taskSequence.)
                      ,&counter.
                      ,~);
    %IF %LENGTH(&task.) > 0 %THEN %DO;
      PROC SQL NOPRINT;
        SELECT CATX('=', parameter,value)
          INTO   :config SEPARATED BY ', '
        FROM   configTasks
        WHERE  taskOrder = &taskId.
      ;
      QUIT;
      %nodeconnectLite(&config.);
    %END;
    %ELSE %LET taskId = 0;

    %LET counter = %EVAL(&counter. + 1);
    %IF &counter. > 9999 %THEN %LET taskId = 0;
  %END;
%MEND nodeBranchLite;
```

Here is the **nodeBranch** invocation which will run the macro:

```
%nodeBranchLite(
  configlib = work
  ,configdata = covid19_branchConfigLite);
```

In my implementation all the project configuration data is in the following folder:

C:\Projects\WorldStatistics\COVID-19\Lite\config

Conversely, all KT macros are in a separate folder tree, available to other projects as required:

C:\Projects\KT\Lite\macros

This avoids having to copy and paste the source code into individual projects, with attendant issues of version control. Additional task macros unique to the project should be contained within the project folder tree.

Richard C Carson

ChatGPT: Write a recursive SAS program

I am generally rather sceptical when ChatGPT tries to write SAS programs, but in this case it has actually generated correct and working code, which suggests it has been trained with this program, or something very similar. However, I'm now rather disappointed that no attribution has been given!

[ChatGPT] Here's an example of a recursive SAS program that calculates the factorial of a number using a recursive function:

```
/* Recursive SAS program to calculate the
   factorial of a number */
/* Define the recursive function */
%macro factorial(n);
    %if &n = 0 %then 1; /* Base case */
    %else %eval(&n * %factorial(%eval(&n - 1)));
    /* Recursive case */
%mend;

/* Call the recursive function */
%put Factorial of 5 is %factorial(5);
```

In this program, the 'factorial' macro function takes a single argument n and recursively calculates the factorial of that number. The base case of the recursion is when n is 0, in which case the function returns 1. The recursive case multiplies n by the factorial of n-1, which is calculated by calling the 'factorial' function with n-1 as the argument.

The %eval macro function is used to perform arithmetic operations on the arguments passed to the 'factorial' function. The %put statement is used to print the result of the factorial calculation to the SAS log.

[SAS 9.4 and Altair SLC 5.24 (2024)]: Surprisingly, the SAS program generated by ChatGPT runs perfectly in SAS and Altair SLC.

```
28 /* Recursive SAS program to calculate
    the factorial of a number */
29 /* Define the recursive function */
30 %macro factorial(n);
31 %if &n = 0 %then 1; /* Base case */
32 %else %eval(&n * %factorial(%eval(&n -
33 1)));
34 /* Recursive case */
35 %mend;
36 /* Call the recursive function */
37 %put Factorial of 5 is %factorial(5);

Factorial of 5 is 120
```

If you have a successful, or unsuccessful, SAS programming request from ChatGPT, then please email the request and response to me at views-uk@hollandnumerics.org.uk.

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 9.4] The SAS formats **NLDATESw.**, **NLDATEMw.** and **NLDATELw.** generate locale-specific short, medium and long date layouts, which can change depending on the available width. Set your locale using the system option **LOCALE=**, which determines date, language and currency defaults, e.g. **de_AT**, **de_CH**, **de_DE**, **en_GB**, **en_US**, **es_AR**, **es_SP**, **fr_BE**, **fr_CH**, **fr_FR**, **hu_HU**, **it_CH**, **it_IT**, **pt_BR**, **pt_PT**.

[SAS 6.07] When used with the **NOBYLINE** system option, the **TITLE** statement options **#BYLINE**, **#BYVAL** and **#BYVAR**, add the variable names and/or values from the **BY** statement.

[SAS 9.4M5] The Data Step function **DLGCDIR()** changes the current working folder of the SAS session on Windows and UNIX.

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 40,200 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 20,500 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 new SAS Global Analytics Explorers hub, then please fill in your **name**, **company email address** and **company** at <https://infl.tv/npmv>.
- 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 integrated 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 5.23.6 (2023).

Contacts

LeRoy Bessler, Visual Data Insights™	Email: Le_Roy_Bessler@wi.rr.com , Le_Roy_Bessler@spectrum.net
Richard C Carson	Email: richard@arcol-analytics.com
SAS UK	Tel: +44-(0)1628-486933 Web: www.sas.com/uk
IEWS	
Web Site	Web: blog.hollandnumerics.org.uk
Newsletter Editor	Email: views-uk@hollandnumerics.org.uk

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

Diary

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

February 2024

25–28 **PhUSE US Connect**, Bethesda, MA, USA

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.

April 2024

16–19 **SAS Innovate**, Las Vegas, NV, USA

SAS INNOVATE is a conference dedicated to analytics and AI for executives and how to transform existing data into faster, more effective decisions in an era of digital transformation, economic uncertainty, consumer demands and fierce competition. Full details at innovate.sas.com.

28–30 **MWSUG**, Milwaukee, WI, USA

MidWest SAS Users Group (MWSUG) is an official SAS Users Group which exists to serve users of SAS software in the midwestern region by providing educational opportunities and resources and by facilitating communication among SAS users and SAS. Full details at www.mwsug.org.

May 2024

19–22 **PharmaSUG**, Baltimore, MD, USA

PharmaSUG is the premiere conference of the year for software professionals in the life sciences industry. For more details see www.pharmasug.org.