

VIEWS

International SAS Programmer Community



News, Hints, Tips and Information for SAS® Users

Issue 62, 4th Quarter 2020

This newsletter issue contains an article from LeRoy Bessler about SparkLine alternatives, a statistics article about rounding from Anders Sköllermo, and I have added an article on code indenting 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.

Annotated SparseLine As a SparkLine Alternative and an Enhanced SparkLine

The Annotated SparseLine and my Enhanced SparkLine are implementations of my oft-repeated graphic design mantra:

“Show them what’s important”.

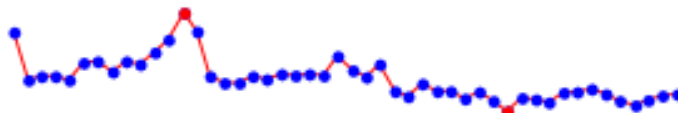
Wikipedia explains the origin around January 1998 of the sparkline: “The concept was developed by interaction designer Peter Zelchenko in conversation with programmer Michael Medved, while Medved was developing the QuoteTracker application. The product was later sold to E-Trade.”.

Below is a table (also known as a SparkTable) using my Enhanced SparkLines. The code is available via email.

Monthly Closing Price 1999-2001: Start, Min, Max, END, Latest Monthly Change, and SparkLine
Any intermediate Min or Max point is Red

Stock	Start	Min	Max	END	Latest Monthly Change	
IBM	99, JAN98	85, DEC00	209, APR99	121, DEC01	+5	
Intel	81, JAN98	20, SEP01	141, JAN99	31, DEC01	-2	
Microsoft	149, JAN98	43, DEC00	175, JAN99	66, DEC01	+2	

This **differs** from typical SparkTables because the tabular data focuses on the critical points of the history, and the images helpfully identify the minima and maxima in the SparkLines. Also, these SparkLine “micrographs” are usually created without highlighting any maximum or minimum. The red dots are not easy to see when image is shrunk to fit here. I could have made the red dots much bigger, if they overlaid the adjacent blue dots that might not be construed as a catastrophe since the micrograph is not offered as an instrument for detailed examination, merely as a picture of the trend that can be fit in a table with data. Here is an enlargement of the SparkLine for the Microsoft Monthly Closing Stock Price:



Back in 1990, I had come to the conclusion that on a trend line or time series plot, often, if not almost always, the points of most, if not only, interest are the Start, End, any intermediate maximum, and any intermediate minimum.

A fresh rethink led me to conclude that an accessory fact of interest is likely to be the change since the second-from-last history point. Another critical point, not a minimum or maximum, would be a point where the trend permanently changed, from, e.g. rapidly increasing (decreasing) to slowly increasing (decreasing), or slowly to rapidly.

For a multi-line multi-year trend plot, you might be looking for seasonality, in which case, in principle, all points are of interest. Nevertheless, you would be especially looking for coincidences, or near coincidences, of maxima and/or minima.

Knowing with certainty the precise coordinate values of a point on a conventional plot is never possible. Running one’s eye from a plot point to an axis is never reliable and then the next step is to interpolate between two tick mark values to estimate the value. Compound guesswork is no route to the knowledge and certainty required for wise conclusions, inferences, and decisions.

An on-page companion table is better than nothing, but running your eye from the plot points to the table cells slows the interpretation process and diminishes the utility of the plot.

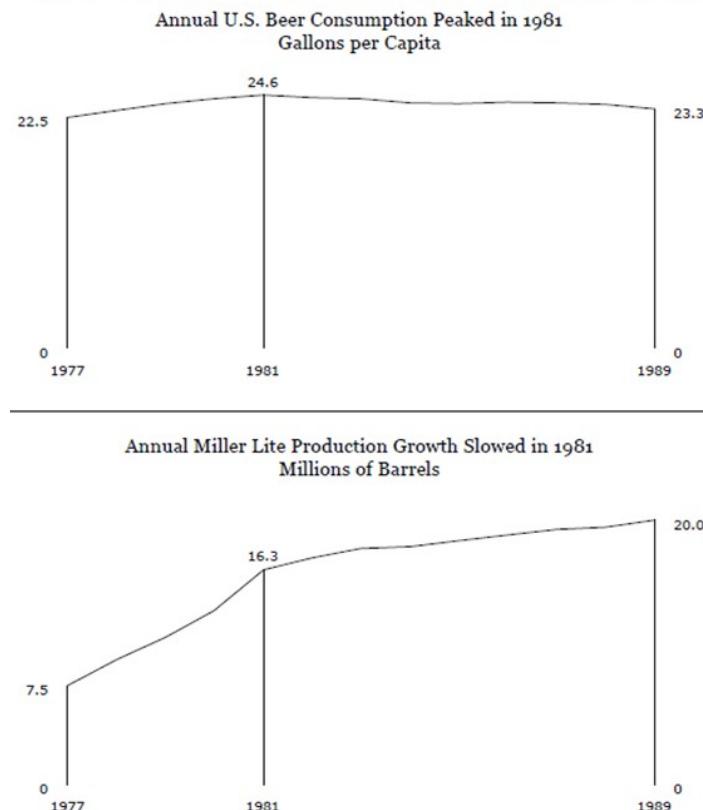
A web-enabled plot, of course, can make precise plot coordinates available with mouse work, but those numbers

vanish as soon as the mouse moves, and the viewer is required to take notes for easy reference.

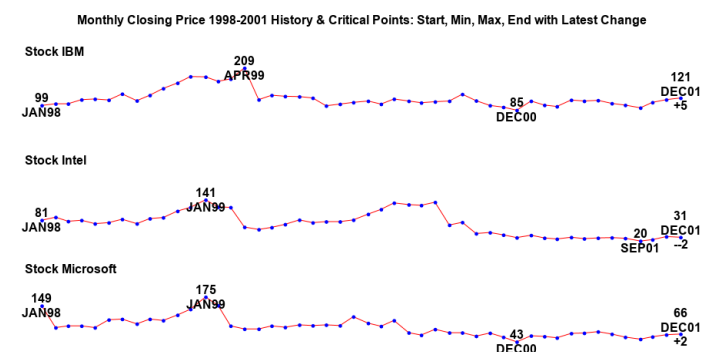
Permanent annotation of the plot is the best solution, but a full annotation of the plot line is impossible when the points are too dense, and annotation of a multi-line plot is impossible without collisions between annotations and/or between annotations and plot lines.

For a single-line plot, sparse line annotation of only the critical points is my solution from 1990. For 2020, I will deal with sparsely annotating what might have been a multi-line plot.

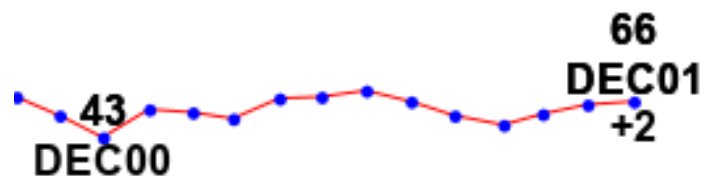
Here is a picture of that 1990 Sparse Line Annotation:



For this particular data, the non-critical history points are clearly not very interesting, especially in the upper plot. Below is my 2020 enhanced update to Sparse Line Annotation, but for a multi-line plot, which is not feasible as an annotated multi-line overlay plot. This Multiple SparseLine Plot uses the same input as the SparkTable.

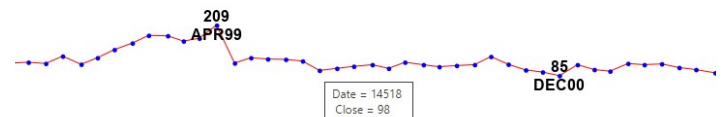


Here is a blow-up to show the annotation more clearly:



Here is an excerpt from a screen capture of a **web-enabled** version of the above Multiple SparseLine Plot:

Monthly Closing Price 1998-2001 History & Critical Points: Start, Min, Max, End with Latest Change
(If desired, use the mouse to display the value for any point that is not annotated.)



Here is the code to create the Multiple SparseLine Plot:

```
DATA CloseByStockByMon;
  FORMAT Date 5. Close 3.;
  SET SASHELP.STOCKS (KEEP = Stock Date Close);
  IF 1998 LE YEAR(Date) LE 2001;
  Close = ROUND(Close); /* eliminate pennies */
RUN;

PROC SORT DATA = CloseByStockByMon;
  BY Stock Date;
  /* in case input is in reverse
  chronological order, which happens to be
  true for SASHELP.STOCKS */
RUN;

PROC MEANS DATA = CloseByStockByMon
  MIN MAX NOPRINT;
  BY Stock;
  VAR Close;
  OUTPUT OUT = MinMax;
RUN;

DATA MinMaxByStock;
  RETAIN MinY MaxY;
  SET MinMax;
  BY Stock;
  IF _STAT_ EQ 'MIN' THEN MinY = Close;
  ELSE IF _STAT_ EQ 'MAX' THEN MaxY = Close;
  IF Last.Stock THEN OUTPUT;
RUN;

DATA Labels (KEEP = Stock xTop yTop xBottom
  yBottom LabelTop LabelBottom);
  LENGTH LabelBottom LabelTop $16;
  RETAIN PredecessorToLastY;
  MERGE CloseByStockByMon
    MinMaxByStock (KEEP = Stock MinY MaxY);
  BY Stock;
  IF First.Stock THEN DO;
    LabelTop = PUT(Close, 3.);
    xTop = Date;
    yTop = Close;
    LabelBottom = PUT(Date, MONYY5.);
    xBottom = Date;
    yBottom = Close;
    OUTPUT;
  END;
  IF NOT Last.Stock
    THEN PredecessorToLastY = Close;
```

```

IF Last.Stock THEN DO;
  LabelTop = TRIM(LEFT(PUT(Close, 3.))) ||
    ' ' || PUT(Date, MONYY5.);
  xTop = Date;
  yTop = Close;
  Change = Close - PredecessorToLastY;
  IF Change GE 0
    THEN LabelBottom = '+' ||
      TRIM(LEFT(PUT(Change, 3.)));
    ELSE LabelBottom = '-' ||
      TRIM(LEFT(PUT(Change, 3.)));
  xBottom = Date;
  yBottom = Close;
  OUTPUT;
END;
ELSE IF PUT(Close, 12.8) EQ MinY THEN DO;
  LabelTop = PUT(Close, 3.);
  xTop = Date;
  yTop = Close;
  LabelBottom = PUT(Date, MONYY5.);
  xBottom = Date;
  yBottom = Close;
  OUTPUT;
END;
ELSE IF PUT(Close, 12.8) EQ MaxY THEN DO;
  LabelTop = PUT(Close, 3.);
  xTop = Date;
  yTop = Close;
  LabelBottom = PUT(Date, MONYY5.);
  xBottom = Date;
  yBottom = Close;
  OUTPUT;
END;
RUN;

DATA CloseByStockByMonAndLabels;
  SET CloseByStockByMon
    Labels;
RUN;

ODS RESULTS = OFF;
ODS _ALL_ CLOSE;

ODS LISTING STYLE = Listing GPATH = "C:\temp";

ODS GRAPHICS ON
  / RESET = ALL SCALE = OFF
  /* NOBORDER turn off border around image */
  WIDTH = 9.5IN HEIGHT = 5IN
  /* Height can be less and still fit content,
    but at some point of reduction
    the lines become boringly nearly flat. */
  IMAGENAME = 'MultipleSparseLinePlot';

TITLE1 HEIGHT = 12PT "Monthly Closing Price
1998-2001 History & Critical Points: Start, Min,
Max, End with Latest Change";
TITLE2 HEIGHT = 6PT " "; /* extra white space */

PROC SGPPANEL DATA = CloseByStockByMonAndLabels
  NOAUTOLEGEND;
  PANELBY Stock
  / LAYOUT = ROWLATTICE NOBORDER NOHEADER;
  SERIES x = Date y = Close
  / MARKERS MARKERATTRS =
    (COLOR=Blue SYMBOL=CIRCLEFILLED SIZE=5)
  LINEATTRS = (COLOR=CXFF0000); /* Red */

```

```

TEXT x = xTop y = yTop TEXT = LabelTop
  / POSITION = TOP TEXTATTRS =
    (FAMILY=Arial WEIGHT=BOLD SIZE=12
    COLOR=Black)
  SPLITCHAR = ' ' /* text part separator */
  SPLITPOLICY = SPLITALWAYS;
  /* The top text will split ALWAYS,
    but only when it contains multiple parts,
    which is true only for the end points. */
TEXT x = xBottom y = yBottom
  TEXT = LabelBottom
  / POSITION = BOTTOM
  TEXTATTRS = (FAMILY=Arial WEIGHT=BOLD
    SIZE=12 COLOR=Black);

INSET Stock
  / POSITION = TOPLEFT
  TEXTATTRS = (SIZE=12PT WEIGHT=BOLD);
ROWAXIS DISPLAY = NONE;
COLAXIS DISPLAY = NONE;
RUN;

ODS LISTING CLOSE;

```

Below are the only changes to the previous code to create the Multiple SparseLine Plot:

- use of ODS HTML instead of ODS LISTING
- change the IMAGENAME and add the IMAGEMAP option in the ODS GRAPHICS statement to turn on mouseover text
- change TITLE2 statement to notify the viewer about the mouseover text option

```

/* Same PROC SGPPANEL input preparation code */
ODS HTML STYLE = Minimal GTITLE GFOOTNOTE
  PATH = "C:\temp" (URL=NONE) BODY =
    "WebEnabled MultipleSparseLinePlot.html"
  (TITLE="Web-Enabled Multiple SparseLine Plot");

ODS GRAPHICS ON
  / RESET = ALL SCALE = OFF WIDTH = 9.5IN
  HEIGHT = 5IN
  IMAGENAME='WebEnabled_MultipleSparseLinePlot'
  IMAGEMAP; /* for mouseover text capability */

/* Same TITLE1 statement */
TITLE2 HEIGHT = 12PT COLOR = CX6666FF BOLD
  "(If desired, use the mouse to display the
  value for any point that is not annotated.)";

/* Same PROC SGPPANEL code,
  but include a FORMAT statement
  to properly display the SAS dates
  in the default mouseover text. */
FORMAT Date MONYY5.;

ODS HTML CLOSE;

```

NOTE: To customize the mouseover text, there are these options, which are not used here:

- **TIP = (var_1 var_2 ...)** to specify the variables whose values are to be displayed
- **TIPFORMAT = (format_1 format_2 ...)** to specify the displayed formats of TIP variables
- **TIPLABEL = (label_1 label_2 ...)** to specify quoted text strings to be displayed as labels of the TIP variables

If you want to accept the default format or label for any variable, use the keyword **AUTO** in place of the format or label.

This has been another episode in my continuing quest to find and show ways to deliver images and precise numbers.

- images = quick, easy inference
- precise numbers = accurate, reliable inference

LeRoy Bessler, Visual Data Insights™

Is your table suitably rounded OR Are all your data values suitably rounded?

Original	Rounded data	Rounded table
33.3	33	33
33.4	33	34
33.3	33	33
100.0	99 or 100?	100

I do not have any good general solution to this problem. Below I try to discuss different ways to handle this problem, in a suitable and correct way. In real life the tables are often much bigger. If the original data have two or three decimals (in the “Original” column), the basic problem is still the same.

Rounding up/down always means that we introduce errors in the data presented. The error, usually the relative error, should be made as small as possible. Consider the following columns:

Rounded data: Round each data, independent of the other data. Each data value is then correctly rounded. However the sum of the data values is, in these examples, usually not correct.

Shall we write 99 or 100 as the sum of the rounded data?

Rounded table: Round the data, in such a way as to make the smallest relative error in the data and at the same time the sum of all the data is still 100. The values that have been explicitly changed by the user, are marked as bold below.

We have used “<” to denote that the value has been explicitly “rounded up” by the user. Likewise we use “>” to denote “rounded down”. Rounding always introduces errors, so we must verify that these errors are acceptable. The exact rules for how the rounding is made must be designed by the user. The rules should also be explained in the user’s text.

Marked and rounded table: A “Rounded table” where we take into account what the data values stand for. We have to look at the meaning of the numbers. We have introduced the notation “← invalid” for rows, where the original values are out of range.

Note that the rounded value might be within the accepted range, but the opposite may also be true. The original values are within the allowed range, but rounding may produce numbers that are out of range.

1)	Original	Rounded data	Rounded table	Marked and rounded table
	11.4	11	11	11
	23.4	23	23	23
	65.2	65	< 66	< 66
	100.0	99 or 100?	100	100

2)	Original	Rounded data	Rounded table	Marked and rounded table
	12.2	12	12	12
	22.6	23	23	23 ← invalid
	65.2	65	65	65
	100.0	100	100	100

3)	Original	Rounded data	Rounded table	Marked and rounded table
	11.4	11	11	11
	22.4	22	22	22 ← invalid
	66.2	66	< 67	< 67 ← invalid
	100.0	99 or 100?	100	100

Suppose, in the examples above, that the second and third rows are measures of certain chemical quantities. Industry regulations may say that the minimum and maximum recommended values are 23 and 66. That is, all values in the second row should be ≥ 23 (greater than or equal). Also all values in the third row should be ≤ 66 (less than or equal).

Rounding may also introduce another type of error. In 2) the value 22.6 is rounded to 23. This may look good. However, the original value is still 22.6 which is too low, so it is invalid. The effect of the rounding is that the rounded value seems to be valid.

In 3) the value 66.2 is rounded up to 67, which is invalid. However the situation is not as bad as it looks, since the original value is 66.2. The user must verify what is valid.

The problem originates from SCB (Statistics Sweden).

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.

Annoying SAS Programming Styles: Part 3

Q: Why should I indent my code?

A:[SAS] After programming in SAS since 1981 I can now read and understand SAS programs, but only if they are formatted logically:

1. Unfortunately code written like the block below (which I created using the TF line command in the Program Editor) is almost impossible for me to understand, because I can’t immediately see the logic used:

```
data fred; set sashelp.class; where
name =: 'J' and age > 11; if substr(
name, 2, 1) = 'a' then do; agewt = age
* weight; if agewt > 13*80 then big =
'Y'; else big = 'N'; end; else do;
ageht = age * height; if ageht > 13*60
then big = 'Y'; else big = 'N'; end;
run;
```

2. The rearranged code below is easier to understand, but the nesting is still nowhere near to being clear enough, so some of the detail is still not obvious at first sight:

```
data fred;
set sashelp.class;
where name =: 'J' and age > 11;
if substr(name, 2, 1) = 'a' then do;
agewt = age * weight;
if agewt > 13*80 then big = 'Y';
else big = 'N';
end;
else do;
ageht = age * height;
if ageht > 13*60 then big = 'Y';
else big = 'N';
end;
run;
```

3. Finally, the indentation of the `do...end` clauses makes the nesting much clearer, explaining the logic behind the `big` values:

```
data fred;
set sashelp.class;
where name =: 'J' and age > 11;
if substr(name, 2, 1) = 'a' then do;
    agewt = age * weight;
    if agewt > 13*80 then big = 'Y';
    else big = 'N';
end;
else do;
    ageht = age * height;
    if ageht > 13*60 then big = 'Y';
    else big = 'N';
end;
run;
```

All 3 programs generate the same SAS data set:

Name	Sex	Age	Height	Weight	agewt	big	ageht
James	M	12	57.3	83	996	N	.
Jane	F	12	59.8	84.5	1014	N	.
Janet	F	15	62.5	112.5	1687.5	Y	.
Jeffrey	M	13	62.5	84	.	Y	812.5
John	M	12	59	99.5	.	N	708
Judy	F	14	64.3	90	.	Y	900.2

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

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 **TIMEw.d** is frequently used to format time values and will format values greater or equal to 24 hours as expected, but for time values less than 10 hours will include a leading blank. To display time values there are several SAS formats available to use, which have different results. For a time value of 25 hours 15 minutes and 10.2 seconds ('25:15:10.2'**T**), **TIME11.2 (TIMEw.d)** will display the time as "25:15:10.20", **TOD11.2 (TODw.d)**, meaning time of day) as "01:15:10.20" (including a leading zero when the hours shown are less than 10), **HHMM8.2 (HHMMw.d)** as "25:15.17", **MMSS10.2 (MMSSw.d)** as "1515:10.20", and **HOURL5.2 (HOURLw.d)** as "25.25".

[SAS 9.4] Have you ever wanted to create animations in SAS? If so, then you will probably find these SAS options useful: **ANIMATION=START | STOP** (default) starts or stops creating animation frames (**ANIMATE=** can be used instead), **ANIMDURATION=** sets the seconds that each frame in the animation is held in view, **ANIMLOOP=YES** (default) | **NO** | **n** sets whether the animation loop is played continuously, is played once, or the number of times that the animation loop is repeated, and **ANIMOVERLAY** (default) | **NOANIMOVERLAY** sets whether animation frames are overlaid or whether they are played sequentially.

[SAS] The well-known macro "functions" **%LEFT()**, which left-justifies macro values, **%RIGHT()**, which right-justifies macro values, and **%TRIM()**, which removes trailing spaces from macro values, appear to be very similar. However, **%LEFT()** and **%TRIM()** are actually SAS-supplied macros (with user-readable programs!), and **%RIGHT()** is the only real macro function.

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 VIEWS UK forum.
- The SAS Professional Forum group on LinkedIn, which currently has over 33,300 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 12,400 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.

Late Spring 2021

Virtual SAS Global Forum

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

May 2021

23-26 **PharmaSUG US**, Baltimore, MA, USA

The 2020 conference was cancelled due to COVID-19. See the PharmaSUG web site at **pharmasug.org** for further details.

August 2021

27-28 **PharmaSUG China**, Shanghai, China

The 2020 conference was cancelled due to COVID-19. See the PharmaSUG China web site at **pharmasug.org/china** for further details.

Contacts

Anders Skölleremo

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

LeRoy Bessler

Email: **le_roy_bessler@wi.rr.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.*