VIEWS
International SAS Programmer Community

News, Hints, Tips and Information for SAS® Users

Issue 48, 4th Quarter 2009

It is always good to hear from old friends. All the articles in this issue of *VIEWS News* are from regular contributors, but Andrew Ratcliffe, who handed over the editorial reins to me in 2002, last wrote an article for us in issue 17, so I'm particularly pleased to welcome him back with an extract from his new blog. Along with Andrew, our regular contributors LeRoy Bessler, Murphy Choy and the Amadeus Software Consultancy Team have written articles on measuring time in SAS jobs, encryption algorithms and outliers, and SAS procedures FCMP and SUMMARY, respectively. Finally, my regular Formats, Options, and Functions section has more unusual features of SAS software.

I will be presenting at the PSI Graphics Day in December (see the Diary section for details), and I have a report from the recent PhUSE Conference in PhUSE News.

For a change I've been blessed with more articles than I could fit in a single issue. However, 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 **newsletter@views-uk.org**. A list of subject suggestions for your articles can be found on the *VIEWS* web site, and anyone is very welcome to add to that list by sending emails to **newsletter@views-uk.org** with your own questions.

Philip R Holland (Newsletter Editor)

*Membership of VIEWS is **free** and you can register for notification of all future VIEWS events by emailing us at **membership@views-uk.org**, making certain you include your name and postal address. If you would like to receive email notification whenever a new issue of the free quarterly VIEWS News is published on the web site, please remember to also include your email address.*

Ask The Consultant

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

Showing absent classification combinations with PROC SUMMARY

Q: How can I include all possible classification combinations in summaries?

A: PROC SUMMARY with the NWAY option will produce a summary of a dataset based on all the combinations of the CLASS variables found in the data.

```
PROC SUMMARY DATA = spend_info NWAY;
 CLASS accnum date;
 VAR spend;
 OUTPUT OUT = summ_spend (DROP = _FREQ_ _TYPE_)
        SUM =;
RUN;
```

The example above produces the following table:

```
accnum     date       spend
111111     JAN2009    147.56
111111     FEB2009    385.99
111111     APR2009    252.04
222222     JAN2009    70
222222     FEB2009    302.1
222222     MAR2009    682.84
222222     APR2009    552.31
222222     JUN2009    486.55
333333     FEB2009    315.44
......     .......    .....
```

The table produced lists all of the combinations of accnum and date that occur within the dataset, looking at the data it is possible to see that there are dates for some accounts that are not listed against others (i.e. 'MAR2009' for accnum '111111').

You may wish to see all possible combinations of the values within the classification variables listed, even if that combination does not appear in the data so that a row will appear for the combination where date is 'MAR2009' and accnum is '111111'.

This can be achieved using the COMPLETETYPES option as in the example below:

```
PROC SUMMARY DATA = spend_info
             COMPLETETYPES NWAY;
 CLASS accnum date;
 VAR spend;
 OUTPUT OUT = summ_spend (DROP = _FREQ_ _TYPE_)
        SUM =;
RUN;
```

PROC SUMMARY has listed out all of the possible combinations of the classification variables based on the values occurring throughout the whole dataset, where a particular combination does not actually appear in the data, a missing value is generated in the analysis variable.

```
accnum     date       spend
111111     JAN2009    147.56
111111     FEB2009    385.99
111111     MAR2009    .
111111     APR2009    252.04
111111     MAY2009    .
111111     JUN2009    .
111111     JUL2009    .
......     .......    .....
```

Here we can see that the date 'MAR2009' is matched with accnum '111111', even though there is no data for that combination in the dataset.

Amadeus Software Consultancy Team

When did my SAS program run, and what were its Elapsed Time and Total CPU Time?

Q: When did my SAS program run, and what were its elapsed time and total CPU time?

A: Though the SAS log displays the real time (elapsed time) and the CPU time for each DATA step and each PROC step, it does not answer the questions of when your program ran, and, unless it was run in batch, what the elapsed time and CPU time were for the entire block of code submitted, which is usually multi-step. The %LogTimer macro is an easy-to-use tool that answers those questions.

Without it, the only timing content in the SAS log, from SAS Display Manager or SAS Enterprise Guide, is of this form:

```
NOTE: DATA statement used (Total process time):
      real time           0.41 seconds
      cpu time            0.02 seconds
```

If your program runs in batch, step-level information like that above will be provided, but also the SAS log will end with a NOTE of this form:

```
NOTE: The SAS System used:
      real time           16.13 seconds
      cpu time            2.06 seconds
```

In SAS 9.2, if you turn on the system option FULLSTIMER, your SAS log will display the date/time at the end of each step. The %LogTimer macro works in all versions of SAS, and does not require FULLSTIMER to be turned on.

Before the SAS log content for the execution of the main body of your code, the %LogTimer macro inserts something like this:

```
****************************************
Started at 4:42:02 on 27OCT2009
****************************************
```

and after the SAS log content for the execution of the main body of your code, the %LogTimer macro inserts something like this:

```
****************************************
Ended at 4:42:04 on 27OCT2009
Elapsed Time (hours:minutes:seconds) = 0:00:02
CPU Time (hours:minutes:seconds) = 0:00:01
****************************************
```

NOTE: As stated in the macro's comments, some parts of the code which captures CPU time are dependent on the specific version of Windows on which the macro is running.

Below are code for the %LogTimer macro and an example of its use:

```
%MACRO LogTimer(StartOrEnd);
%LET StartOrEnd = %UPCASE(&StartOrEnd.);
%GLOBAL LogTimerWasStarted;
%GLOBAL SaveStartDateTime;
%GLOBAL SaveStartCPUtime;

DATA _NULL_;
 DateTime = DATETIME();
 CALL SYMPUT('LogTimerDate',
  TRIM(LEFT(PUT(DATEPART(DateTime), DATE9.))));
 CALL SYMPUT('LogTimerTime',
  TRIM(LEFT(PUT(TIMEPART(DateTime), TIME8.))));
 IF "&StartorEnd" = 'START' THEN
   CALL SYMPUT('SaveStartDateTime',
               PUT(DateTime, 22.10));
 /* maximum precision for saved Start datetime
    to avoid possible negative elapsed time */
 ELSE DO;
  ElapsedSeconds =
     DateTime -
     INPUT(SYMGET('SaveStartDateTime'), 22.10);
  CALL SYMPUT('LogTimerElapsedTime',
       TRIM(LEFT(PUT(ElapsedSeconds, TIME.))));
 END;
RUN;

%LET TaskListCommand = %STR('tasklist /v');

FILENAME TaskList PIPE &TaskListCommand.;

DATA _NULL_;
 INFILE TaskList LRECL = 224 PAD END = LastOne;
 /* LRECL varies by Windows version.
    224 is appropriate for Windows XP.
    229 is for Windows 2003 Advanced Server. */
 INPUT @1 CommandResponse $CHAR224.;
 IF _N_ GE 4;
 IF INDEX(CommandResponse, "&sysjobID.") NE 0;
 Cumulative_CPUtime =
         TRIM(LEFT(SUBSTR(CommandResponse,
                          140, 12)));
 /* offset of Cumulative_CPU_time
    varies by Windows version.
    140 is appropriate for Windows XP.
    145 is for Windows 2003 Advanced Server. */
 Cumulative_CPU_seconds =
         INPUT(Cumulative_CPUtime, HHMMSS12.);
 IF "&StartorEnd." = 'START' THEN
   CALL SYMPUT('SaveStartCPUtime',
         TRIM(LEFT(Cumulative_CPU_seconds)));
 ELSE DO;
  CPU_seconds =
         Cumulative_CPU_seconds -
         INPUT(SYMGET('SaveStartCPUtime'), 8.);
  CALL SYMPUT('LogTimerCPUtime',
         TRIM(LEFT(PUT(CPU_seconds, TIME.))));
 END;
RUN;

%PUT ****************************************;

%IF &StartOrEnd. = START %THEN %DO;
 %PUT Started at &LogTimerTime. on
      &LogTimerDate.;
 %LET LogTimerWasStarted = YES;
%END;
%ELSE %IF &StartOrEnd. = END %THEN %DO;
 %IF &LogTimerWasStarted. EQ YES %THEN %DO;
  %PUT Ended at &LogTimerTime. on
       &LogTimerDate.;
  %PUT Elapsed Time (hours:minutes:seconds) =
       &LogTimerElapsedTime.;
  %PUT CPU Time (hours:minutes:seconds) =
       &LogTimerCPUtime.;
 %END;
```

```
 %ELSE %DO;
  %PUT LogTimer Macro User ERROR: Invocation
       Value was &StartOrEnd.;
  %PUT But there was no prior invocation with
       Start;
 %END;
%END;
%ELSE %DO;
 %PUT LogTimer Macro User ERROR: Invocation
      Value was &StartOrEnd.;
 %PUT Must be Start or End;
%END;

%PUT ****************************************;

%MEND LogTimer;
```

Here is how to use it:

```
%LogTimer(Start);

<your code here>

%LogTimer(End);
```

Invocation of the macro with "End" before "Start" or invocation of the macro with any other value will result in an appropriate error message.

NOTE: The "End" invocation of the %LogTimer macro can be used, if desired, at multiple points within a code submission, in which case it, of course, displays the cumulative Elapsed Time and CPU Time as processing progresses. Discrete Elapsed Time and CPU Time for each constituent DATA step or PROC step are provided by SAS itself in the usual manner.

LeRoy Bessler

Did You Know?

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

Detecting outliers and missing values

Abstract

Outliers and missing values are important aspects of data. Very often, data mining model never consider the aspect of outliers and missing values. Outliers are usually values which deviate drastically from normality while missing values are usually due to inability to capture the information. Occasionally, such value detection is difficult in large datasets. Here we will discuss a way to look for outliers and missing values.

Introduction

Data preparation work is something that is very important for modelling work. Most of the modelling techniques currently do not handle outliers and missing values, which are very common in most data. Given the inability to handle such data, preparation work has to be done to ensure that the data can reflect the truth and that modelling work results are valid. In order to come up with corrective measures, one must first be able to detect them. In the following sections, we will explore the different approaches to detecting such data anomalies.

Outliers

Outliers are defined in a variety of ways. In most cases, it is defined as the group of values which deviates significantly from the rest of population where the measurements are taken (Grubb, 1969). The motivation behind the attempts to detect outliers lies in the fact that some of the most common statistical techniques are not inherently robust against outliers. The most commonly quoted example is the one on mean and median. The mean is a statistical measure that is easily influenced by the presence of outliers, while median is more robust against outliers.

The below example demonstrates the problem of outliers in mean and how median helps.

Assuming the following readings:

```
X1 = 1
X2 = 2
X3 = 3
X4 = 4
X5 = 40
Mean = (1+2+3+4+40)/5 = 10
Median = 3
```

If you notice how much the single outlier has changed the mean, which itself is much bigger than 80% of the values. However, median is still within acceptable range compared to the values. Thus we noticed that a single outlier can change the entire aspect of the statistical measures. As regression models are mainly based on means, they are easily affected by the presence of outliers and thus outlier elimination is very critical. There are 2 types of outliers, the first form being the known outliers form, and the second form being the unknown outliers form.

Known outliers

Known outliers are usually values in the population which simply cannot exist due to logical contradiction. An example is the age where it is impossible for a living human to be more than 150 years old and any person with an age greater than 100 would have been considered to be an outlier. Thus in this case it is possible for us to filter out values which will be considered to be outliers. In cases where they are known, the solution to removing them is considerably easier than cases where they are unknown.

PROC FORMAT

PROC FORMAT is one of the common procedures that we use often to create formats for the data itself. In this case, it can also be used to detect values which are outliers as well. Below is an example of some age data. We can see how to look for the outlier values here in this section.

```
DATA age;
 INPUT age;
 DATALINES;
12
15
-19
30
45
60
70
90
1000
RUN;
```

From the above, we can see that there is a negative value, which is invalid, and another value which is way too big. Due to the size of the data, we can see the error easily. But in situations where the dataset is much bigger, such detection will be difficult

to achieve manually, or by eye-balling. Let us see how a simple PROC FORMAT can help in finding these invalid values.

```
PROC FORMAT;
 VALUE valida
  1-99 = 'VALID'
  OTHER = 'INVALID'
  ;
RUN;
```

The PROC FORMAT in this case creates a format that indicates values which are valid and invalid. By applying the format on the data itself, it will allow us to identify data which are not valid. Below is an example of the code used to identify the records which are invalid.

```
DATA _NULL_;
 SET age;
 test = PUT(age, valida.);
 PUT age= @12 test=;
RUN;
```

After running this code, the result will be published to the log section.

```
age=12     test=valid
age=15     test=valid
age=-19    test=invalid
age=30     test=valid
age=45     test=valid
age=60     test=valid
age=70     test=valid
age=90     test=valid
age=1000   test=invalid
NOTE: There were 9 observations read from the
data set WORK.AGE.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.01 seconds
```

We can see that the 2 invalid values have been identified. However, printing a few million records to the screen will be disastrous. For bigger data, we can create a separate dataset for those records which are invalid.

Unknown outliers

Unknown outliers are more common in measurements which are not common knowledge, or do not necessarily have a fixed range. These types of outliers are more difficult to find and requires more effort to test for them. As we cannot determine the limits of normality, an additional step is required to determine the limits which often rely on some form of testing. Most of the statistical tests rely on parametric assumptions such as normality. Most of these assumptions are not upheld in the real world, it is dangerous to use them for outlier removal as actual true values might be removed as well. To avoid such precarious situation, a non parametric approach is utilized in this case.

Box plots

Box plots are graphical means for examining the distribution of the data. It uses several measures such as quartiles, median and inter-quartile range to describe the distribution of the data. As it does not rely on any assumptions of the distribution of the data, it is considered to be a non-parametric approach. Box plots define any values beyond 1.5 times of the inter-quartile range from either the first quartile or the third quartile to be outliers. As the quartiles are not affected by outliers, they are robust in nature and the inter-quartile range also indicates the spread of data within the population. However, there are no fixed functions to perform this kind of outlier removal in SAS. To remedy the problem, we will be using a macro to do such filtering. Below is the macro.

```
%MACRO box_outliers(input=, xvar=, output=);
/*Univariate function: Extract critical info*/
PROC UNIVARIATE DATA = &input. NOPRINT;
 VAR &xvar.;
 OUTPUT OUT = &xvar. QRANGE = iqr
        Q1 = p25str Q3 = p75str;
RUN;

/*Setting the limits*/
DATA _NULL_;
 SET &xvar.;
 ub = p75str + 1.5 * iqr;
 lb = p25str - 1.5 * iqr;
 CALL SYMPUT('ub', ub);
 CALL SYMPUT('lb', lb);
RUN;

/*Outputting rejects and accepts*/
DATA &output. rejects;
 SET &input.;
 IF &lb.<=&xvar.<=&ub. THEN OUTPUT &output.;
                       ELSE OUTPUT rejects;
RUN;
%MEND;
```

Missing values and MISSING() function in SAS

Missing values in SAS are very common and occur naturally when the data read in does not have information in those observations. It can also occur when logical contradictions occur in the data itself. However, as compared to outliers, missing values are more easily discerned. In SAS we have a very useful function called MISSING(), which allows us to detect missing values on both numeric and character variables.

Conclusion

SAS provides a variety of ways for users to handle missing and outliers. It is crucial that modellers use these tools to make life easier for themselves.

Murphy Choy, University College Dublin

Debugging Nested Loops in SAS 9.2

[SAS 9.2] One of the earliest posts on my new blog at **notecolon.info** describes a new feature of SAS 9.2. Judging by the number of hits, the post has been very popular with the blog readership, so I thought it worth repeating here.

I discovered a DATA statement argument that's new in SAS 9.2 today. The NESTING argument instructs SAS to print a note to the log at the beginning and end of every DO/END and SELECT/END pairing.

This is jolly handy for debugging recalcitrant DATA steps with unbalanced nesting.

Andrew Ratcliffe, RTSL.eu

PROC FCMP for Scrabble

[SAS 9.2] Now that the RX family of functions is being retired, we need new ways of calculating how many points a word scores at Scrabble, and which of several possible words scores highest. One way of doing this uses PROC FCMP to define some

user-written functions. In SAS 9.2 such functions can be used in data steps. Here is how to define a function to score a single word:

```
PROC FCMP OUTLIB = work.funcs.words;
 FUNCTION score(word $);
  points = COUNTC(word, 'AEILNORSTU') +
           2 * COUNTC(word, 'DG') +
           3 * COUNTC(word, 'BCMP') +
           4 * COUNTC(word, 'FHVWY') +
           5 * COUNTC(word, 'K') +
           8 * COUNTC(word, 'JX') +
           10 * COUNTC(word, 'QZ');
  RETURN(points);
 ENDSUB;
QUIT;
```

This function definition will be stored as part of a package called WORDS within a data set WORK.FUNCS. The function is called **score()** and takes a single parameter, a character string called **word**. Since **score()** is a function (and not a subroutine), parameters are passed by value. The COUNTC() function is used to count how many letters in the word score one point each, and so on, and the calculation is performed in the obvious way.

Now we define a subroutine called **highest()**, which will use **score()** to calculate which of several words scores highest:

```
OPTIONS CMPLIB = work.funcs;

PROC FCMP OUTLIB = work.funcs.words;
 SUBROUTINE highest(string $, topword $,
                    topscore);
  OUTARGS topword, topscore;
  nw = COUNTW(string);
  DO I = 1 TO nw;
   thisscore = score(SCAN(string, 1));
   IF thisscore > topscore THEN DO;
    topscore = thisscore;
    topword = SCAN(string, i);
   END;
  END;
 ENDSUB;
QUIT;
```

This subroutine is going to call the **score()** function, so it needs to know where to find it. This is the purpose of the CMPLIB= option. Note that the package is not mentioned here. Since **highest()** is a subroutine, parameters are passed by reference, and some of them can be updated. Those that are to be updated are declared in an OUTARGS statement.

highest() uses the COUNTW() function, new in SAS 9.2, to count the number of words in the input string, and then loops around using **score()** to evaluate each.

Here is a program that shows this subroutine in use:

```
OPTIONS CMPLIB = work.funcs;

DATA test;
 LENGTH instr $80 word $40;
 INPUT instr $ 1-80;
 newstr = UPCASE(COMPRESS(instr, , 'p'));
 CALL highest(newstr, word, score);
 PUT 'In "' instr +(-1) '",';
 PUT 'the highest-scoring word in '
     word 'which is worth ' score;
 DATALINES;
Which of these words scores highest, do you
think:
benzoxycamphors, or sacerdotalizing, or
squandermaniacs?
RUN;
```

As before, the CMPLIB= option is needed, to specify the location of the user-written functions and subroutines. COMPRESS() is used to eliminate all punctuation marks (**'p'**) from the input string. The output from the program is:

```
In "Which of these words scores highest, do you
think:",
the highest-scoring word is WHICH which is
worth 16
In "benzoxycamphors, or sacerdotalizing, or
squandermaniacs?",
the highest-scoring word is BENZOXYCAMPHORS
which is worth 45
```

Amadeus Software Consultancy Team

Classical Cipher in SAS: Caesar's Cipher

Caesar's cipher is one of the earliest known ciphers that is being used for encryption purposes. As its name suggested, it is supposedly to be created and used by the great Roman general, Julius Caesar. The fundamental workings of the cipher is very simple and it is basically a shift in the in the alphabet order and substituting the values in the original string with the values in the new string. The operation can be expressed in a mathematical form which is modulus function.

The encryption function is expressed below as:

$$E_n(x) = (x + n) \mod 26.$$

The decryption function is expressed below as

$$D_n(x) = (x - n) \mod 26.$$

Due to the nature of this shift, it is easily broken by a simple shift of the alphabet series. At the same time, given that the maximum shift is 26, it is possible to use a brute force approach to break through the cipher. Thus this cipher provides very little encryption protection.

```
/*********************************************
Caesar's Cipher
---------------------------------------------
Parameter  Description
---------------------------------------------
input      Input data
var        Variable to be encrypted/decrypted
shift      Shifting degree
mode       ENCRYPT/DECRYPT
*********************************************/

%MACRO caesar_cipher(input=, var=,
                     shift=, mode=);
DATA _NULL_;
 /*Alphabetical string: 26 characters*/
 original = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
 /*Shifting the string by number of character*/
 cipher = COMPRESS(SUBSTR(original, &shift.+1)
%DO i = 1 %TO &shift.;
          || SUBSTR(original, &i., 1)
%END;
          );
```

```
 /*Assign macro var for subsequent use*/
 CALL SYMPUT('original', COMPRESS(original));
 CALL SYMPUT('cipher', COMPRESS(cipher));
RUN;

/*Encryption or decryption of data*/
DATA &input.;
 SET &input.;
 /*ENCRYPTION*/
%IF %UPCASE(&mode.) = ENCRYPT %THEN %DO;
 encrypted_&var. =
    TRANSLATE(&var., "&original.", "&cipher");
%END;
 /*DECRYPTION*/
%ELSE %IF %UPCASE(&mode.) = DECRYPT %THEN %DO;
 decrypted_&var. =
    TRANSLATE(encrypted_&var., "&cipher",
              "&original.");
%END;
 /*NO ACTION*/
%ELSE %DO;
 &var. = &var.;
%END;
RUN;
%MEND;
```

The macro requires all of the the parameters to be supplied:

```
%caesar_cipher(input=class, var=name,
               shift=5, mode=ENCRYPT);
```

This is probably the easiest cipher that can be implemented in SAS. While it is not the best cipher available to us, it nevertheless provides a simple tool for us to work with.

Murphy Choy, University College Dublin

In Brief

- Back issues of *VIEWS News* are available from the *VIEWS* web site, and also from the *VIEWS News* page at **www.sasCommunity.org/wiki/VIEWS_News**
- Andrew Ratcliffe's excellent email newsletter "NOTE:" has been converted into a blog at **notecolon.info**. Recent updates have included links to *VIEWS News*!
- More SAS hints and tips can be found on the SAS FAQ pages **www.hollandnumerics.com/sasfaq** and also on the Amadeus Software Tips pages at **www.amadeus.co.uk/TipsList.aspx?ST=1**
- There is a social networking groups for SAS users on Ning:
  - **SAS Professionals**, intended for SAS users in the UK, at **www.sasprofessionals.net**, but currently totalling more than 1,350 members worldwide.
- Phil Holland has also started a group page on Squidoo for other pages related to SAS, and would like to make it a central resource for SAS hints, tips, links and books at **www.squidoo.com/groups/SAS_users**
- SAS users who use Twitter can now join a "twibe" at **twibes.com/SAS**, which filters your tweets down to those containing "**SAS**", "**EnterpriseGuide**" and "**PROC**".
- There is a new forum for the pharmaceutical sector, with SAS-related streams at **www.phenix-mtk.com/forums.**
- Phil Mason has a SAS tips newsletter, which can be requested by going to **www.woodstreet.org.uk** and then clicking on Tips. Previous tips can also be found on the web site.

Formats, Options, and Functions

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

[SAS 9.2] The **CHAR()** function returns a string of length 1 from the specified position of the string argument.

[SAS 9.2] The **TRAILSGNw.** informat reads numeric values with trailing plus or minus signs. This informat existed in earlier versions of SAS, but it now accepts numeric values containing commas.

[SAS 9.2] The **ODS GRAPHICS** statement can be instructed to send images to different file formats using the **IMAGEFMT=** option. This option can have the following values, most of which are familiar and self-explanatory: **STATIC** (default, uses the best quality static image format), **BMP**, **DIB**, **EMF**, **EPSI**, **GIF**, **JFIF**, **JPEG**, **PBM**, **PCD**, **PDF**, **PICT**, **PNG**, **PS**, **TIFF**, **WMF**, **XBM** or **XPM**.

PhUSE News

2009 PhUSE Conference

This year's PhUSE conference was held in Basel, Switzerland on 19-21 October 2009. The conference continues to grow with further development of relationships across companies and countries, and saw participation increase to 400. There were 114 presentations in 11 streams and a University Day.

The main theme this year was "Accelerating Clinical Development" and an impressive and much referred to keynote speech was given by Diane Jorkasky, the video of which can be viewed on the PhUSE website, **www.phuse.eu**. The main theme also brought forth a most successful discussion club. All papers will be available on the PhUSE website in the near future. Any feedback from the conference is most welcome, please feel free to communicate to us what you liked this year or would like to change next year.

Thank you for contributing to the success of PhUSE and we hope that you will be able to join us at the conference in 2010 which will take place in Berlin, Germany. Please note that the Call for Papers will be open at the slightly earlier date of January 2010.

PhUSE Committee

PhUSE Single-Day Event in December 2009

The ESUG committee is pleased to announce that it will be meeting at Brunel University Conference Centre, Uxbridge on 10 December 2009, to focus on the implementation of CDISC standards. You can register for "Implementing CDISC standards from protocol to p-value" at **www.phuse.eu/1-day-events.aspx**.

PhUSE Committee

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.

December 2009

1 **SAS Programming Tips and Tricks**, Oxfordshire
See the Amadeus Software web site **www.amadeus.co.uk** for further details.

8 **PSI Graphics Coding Challenge**, London
For further details see the PSI Events Summary web page **https://services.psiweb.org/site/events/events_summary.asp**

10 **PhUSE/ESUG Single Day Event**, Middlesex
For further details please go to the PhUSE web page at **www.phuse.eu/1-day-events.aspx**.

January 2010

21 **SAS Programming Tips and Tricks for Base SAS**
This is a webinar. For further details see the Amadeus Software web site **www.amadeus.co.uk/Webinars.aspx**

27 **SAS Programming Tips and Tricks for the Output Delivery System**
This is a webinar. For further details see the Amadeus Software web site **www.amadeus.co.uk/Webinars.aspx**

February 2010

8-9 **Best Practices in SAS Statistical Programming in Regulatory Submission**, New Brunswick, NJ, USA
These courses are intended for anyone directly or indirectly responsible for the creation, content or validation of summary tables, data lists and graphs used to support research, drug or medical device efficacy and safety in a regulatory submission. See **tinyurl.com/5ayrmq** for more details of the courses from Sunil Gupta.

16 **SAS Programming Tips and Tricks for Base SAS**
This is a webinar. For further details see the Amadeus Software web site **www.amadeus.co.uk/Webinars.aspx**

25 **SAS Programming Tips and Tricks for the Output Delivery System**
This is a webinar. For further details see the Amadeus Software web site **www.amadeus.co.uk/Webinars.aspx**

March 2010

18 **SAS Programming Tips and Tricks for Base SAS**
This is a webinar. For further details see the Amadeus Software web site **www.amadeus.co.uk/Webinars.aspx**

24 **New and Enhanced Features of SAS 9.2**, Oxfordshire
See the Amadeus Software web site **www.amadeus.co.uk** for further details.

30 **SAS Programming Tips and Tricks for the Output Delivery System**
This is a webinar. For further details see the Amadeus Software web site **www.amadeus.co.uk/Webinars.aspx**

April 2010

11-14 **SAS Global Forum**, Seattle, WA, USA
The SAS Global Forum (SGF, formerly called SUGI) is held each year and focuses on the technical aspects of SAS software. See **www.sasglobalforum.org** for more details.

May 2010

5 **SAS Programming Tips and Tricks**, Oxfordshire
See the Amadeus Software web site **www.amadeus.co.uk** for further details.

June 2010

10 **From Data to Knowledge: A Journey into SAS9**, Oxfordshire
See the Amadeus Software web site **www.amadeus.co.uk** for further details.

October 2010

18-20 **PhUSE Conference**, Berlin, Germany
The Pharmaceutical Users Software Exchange (PhUSE) conference is the premier programming event for the pharmaceutical industry in Europe. See **www.phuse.eu** for more details.

All VIEWS event information will be posted and/or emailed to registered members of VIEWS, and will include an event application form. Please send your queries about any VIEWS events to **event@views-uk.org**, *and don't forget to look at the web site for the latest news.*

Contacts

Amadeus Software	Email: **info@amadeus.co.uk** Tel: +44-(0)1993-878287 Web: **www.amadeus.co.uk**
LeRoy Bessler	Email: **le_roy_bessler@wi.rr.com**
Murphy Choy University College Dublin	Email: **goladin@gmail.com**
Andrew Ratcliffe RTSL.eu	Email: **enquiries@rtsl.eu** Web: **www.rtsl.eu**
PhUSE	Email: **office@phuse.eu** Web: **www.phuse.eu**
SAS UK	Tel: +44-(0)1628-486933 Web: **www.sas.com/uk**
VIEWS	
Web Site	Web: **www.views-uk.org**
Chairman	Email: **chair@views-uk.org**
Newsletter Editor	Email: **newsletter@views-uk.org**
Membership secretary	Email: **membership@views-uk.org**

VIEWS News *is published quarterly by VIEWS International SAS Programmer Community. VIEWS is a not-for-profit organisation. Our Mission Statement and Rules can be viewed on our web site together with a list of current committee members.*