

As the new Millennium starts so does the third year of *VIEWS News*. I trust that you continue to find the newsletter to be a useful contributor to your SAS® software knowledge. We've included a questionnaire that we hope you'll find the time to complete and return to us. We're not resting on our laurels and would like to make *VIEWS News* even better.

We'd like to offer a big thank you to ASA recruitment and contract consultants who begin their third year of sponsorship for *VIEWS News*. ASA have been staunch supporters of *VIEWS News* since its beginning and we are extremely grateful.

Finally, I must apologise and draw your attention to the incorrect email addresses I gave in the article about SAS-L in the last issue. Please see the corrective article in this issue for details.

Andrew Ratcliffe (Editor)

Did You Know?

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

Reordering variables in a data set

Sometimes I want to reorder the variables in a data set. Often when I interactively browse, edit or print a dataset I want the ID variables to appear first. I could of course use ID and VAR statements or the SHOW ID command to do this each time I access the data. However, if the variables are correctly ordered within the data set I do not need to carry out these steps. I find it particularly useful to order variables in frequently accessed data sets.

When I am producing a data set for others, I am sometimes requested to have the variables appear in the data in the same order as in the documentation.

I have found that SQL provides a simple way of reordering variables. Try the following code.

```
DATA testdata; a=1; b=2; c=3; d=4; e=5; RUN;

PROC SQL;
  CREATE TABLE testdata
  AS SELECT e, d, *
  FROM testdata;
RUN;
```

Below is the data set with variables reordered.

OBS	E	D	A	B	C
1	5	4	1	2	3

Note that variables *e* and *d* are now first. The '*' in the 'AS SELECT' statement selects the remaining variables in their original order.

Reordering variables does not in general affect SAS programmes. The exception being the use of '--' variable selection lists, for example, 'KEEP id1 -- c;'.
David Trenery

Tidy Format Regeneration

Those who use format catalogs regularly will be aware their log files are often generated with warning messages, informing us that each format already exists ("WARNING: Format FRUIT is already on the library"). This occurs when altering or adding a format and we need to regenerate the catalog in batch mode.

To avoid the log file containing unnecessary warning messages, add the following short macro to the top of the formats code:

```
Libname fmtlib 'my format library';

%Macro delforms();

%If %SysFunc(CExist(fmtlib.formats)) %Then
%Do;
  Proc Datasets NoList Nodetails MT=Catalog Library=fmtlib;
    Delete Formats;
  Quit;
%End;

%Mend delforms;

%Delforms;
```

The macro utilises the %SYSFUNC macro function and CEXIST SCL function to determine the existence of the format catalog. If found, the DATASETS procedure is executed and the formats catalog is deleted. The format procedure can now be executed without generating warning messages.

This principle can be extended to test for specific entries within formats catalogs and catalogs of other types.

David Shannon

Accepting Pounds

Unlike the mainframe platform the dollar format and comma informat on MS-Windows and UNIX don't tolerate a £(GBP) symbol. Although picture formats can be created to generate the sterling equivalent of the 'dollar.' format, until recently there was no alternative solution for reading text with £(GBP) prefixed money, other than parsing-out that symbol.

But now there is. The 'euro.' informat (mentioned in issue 5 of *VIEWS News*) will tolerate whichever prefix symbol you select for a euro, and that choice is yours by using the DEBUG system option (mentioned in issue 6).

```
208 %let sym=%sysfunc( putc( £, $hex2 )); *get hex of your symbol;
209 options debug="euro=&sym" ; *apply your choice;
210
211 data et;
212 infile cards truncover;
213 input money euro12. @1 text $char10.;
214 put text= @10 money best8. @20 money euro8.2;
215 cards;

TEXT=12      12      £12.00
TEXT=.5      0.5      £0.50
TEXT=£50     50      £50.00
TEXT=£ 5.    5       £5.00
TEXT=$5.2    5.25    £5.25
TEXT=£.5     0.5     £0.50

NOTE: The data set WORK.ET has 6 observations and 2 variables.
NOTE: The DATA statement used 0.05 seconds.

222 ;
```

Oddly, I notice that a \$ prefix is also supported by the 'euro.' informat (see the fifth example above).

Euro support is not a standard part of V6.12 of SAS software. SAS Institute has made a "Euro Package" available for free download from their web site (or on request from SAS Institute). You will find full details about V6.12 euro support at [software/distribution/readme/europack_final.612/europackv6.html](http://www.sas.com/software/distribution/readme/europack_final.612/europackv6.html) at the SAS Institute web site. Note that the package has moved from "beta" to "final" since it was mentioned in issue 5.

Peter Crawford

The Consultant

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

Sectioned Data Tables

Q: In issue 3 of *VIEWS News* you described how to create a 'sectioned' report using PROC REPORT. Is it possible to display something similar with a SAS/AF® data table? I've tried using the (experimental) Report Viewer widget but it doesn't seem very stable - and it has no documentation.

A: I think they misspell the word experimental, it should read temperamental! However, the reliable 'Data Table' will do the job for you. You need to use the Row/Column Data Vector's `_SET_NAME_JUST_` method.

I don't have much space here for details, so I've created a simple demonstration. Create a frame named `SASUSER.VN9.SECT.FRAME` and add a Data Table widget named `DTABLE`. Use the following as SCL for the frame.

```
INIT:
/* Create d/s with extra rows for "headers" */
submit continue status;
data work.newmem;
  length _first_ 3;
  set sashelp.vmember;
  by libname notsorted;
  if first.libname then do;
    _first_ = 1;
    output;
    _first_ = .;
  end;
  output;
run;
endsubmit;

/* Identify our d/s and hide 1st two columns */
call notify('DTABLE', '_SET_DATASET_', 'WORK.NEWMEM');

call notify('DTABLE', '_HIDE_COLUMN_', '_FIRST_', 'LIBNAME');

/* Get the ID of the Table Data Model object so that */
/* we can use _SET_INSTANCE_METHOD_ method */
call send(_frame_, '_GET_WIDGET_', 'DTABLE', pDtable);

pDmodel = getnitem(pDtable, 'DATAID');

call send(pDmodel, '_SET_INSTANCE_METHOD_', '_GET_ROW_INFO_',
          , 'SASUSER.VN9.DT.SCL'
          , 'GRI', 'AFTER' );

RETURN;
```

Save and compile the frame and its SCL. Now create an SCL entry named `SASUSER.VN9.DT.SCL` using the following code.

```
length _method_ $200 seod $1 ;

GRI:
method pRcdv 8;

/* Find our row number */
lCoord = makelist();
call send(pRcdv, '_GET_COORDINATES_', lCoord);
rownum=getitem(lCoord);
rc=dellist(lCoord, 'Y');

/* Make sure we've not gone beyond end of d/s */
call send(pRcdv, '_GET_SOFT_END_OF_DATA_', seod);

if seod eq 'N' then do;

  /* Open data set to get values */
  dVmember=open('WORK.NEWMEM');

  rc=fetchobs(dVmember, rownum);

  vnFirst=varnum(dVmember, '_FIRST_');
  first=getvarc(dVmember, vnFirst);

  if first then do; /* This is a header row */

    vnLibname=varnum(dVmember, 'LIBNAME');
    libname=getvarc(dVmember, vnLibname);

    /* This text shouldspan the whole row */
    call send(pRcdv, '_SET_NAME_JUST_', -1);

    /* Set the text, left-justify and reverse-video */
    call send(pRcdv, '_SET_TEXT_', libname);
    call send(pRcdv, '_SET_HJUST_', 'LEFT');
    call send(pRcdv, '_SET_REVERSE_', 'Y');

  end; /* This is a header row */

  else
  do; /* Ordinary row */
    call send(pRcdv, '_SET_TEXT_', '');
  end; /* Ordinary row */

  rc=close(dVmember);

end;

ENDMETHOD;
```

Compile and save the SCL entry and run the application (don't TESTAF it, it submits code). You'll see a frame similar to the accompanying screen-shot.

I'll offer some explanation for what's happening. Firstly the frame. We create a data set that has additional rows representing our section breaks and their labels. The fact that the row is a section break is represented by a value of 1 for `_FIRST_`. After creating the data set we can tell the Data Table to display it and to hide the two variables that should not be displayed as columns. Finally, we need to tell the Data Table that we want to add some extra processing of our own when it is preparing row labels (the grey area down the left-side of the Data Table). We do this with the `_SET_INSTANCE_METHOD_` and tell it to run our code at after its own.

In our own code for `_GET_ROW_INFO_` (called once for each row in turn) we read the requisite row from our data set and check to see if it's a section break row. If not, we do nothing more. For section break rows we use the `_SET_NAME_JUST_` method with a value of -1 to indicate that we want the row label to extend across the whole of the width of the table. At the same time we set the text that we want and apply a little formatting.

So, very briefly, that's it! This example is very simplistic and is for demonstration purposes only, but I hope it has shown the general principles of how to use this style of table. It can be used in very many different situations and provides a welcome degree of flexibility to add to the otherwise mundane looking table.

Andrew Ratcliffe

Refreshing User Defined Formats in a SAS/AF® Application

Q: Changes made to a user-defined format during the execution of a SAS/AF® application are often not loaded until the application is shutdown and restarted. How can I see the results of changed formats during the execution of my application?

A: The reason for this is that a user-defined format is loaded into memory when it is first used and therefore, any changes that are made to that format are not loaded again until the application is restarted and the format reloaded.

There is a rather simple solution to this problem that does not require the application to be shutdown and restarted. The answer lies in being able to run multiple SAS/AF applications at the same time. Using the following as an example we can show the solution in action:-

Our application consists of two simple frames. Frame 1, a data selection frame and Frame 2, a data viewing frame containing a data table.

The INIT section of Frame 2 generates a common format `$COMMON.`, whose values are dependent upon the data selection made by the user in Frame 1, via some PROC FORMAT code written into a submit continue block. Unfortunately, upon entering Frame 2 the second and subsequent times, the `$COMMON.` format will contain those values that were placed in it when it was created upon entering Frame 2 the first time. The only way to get the format to be updated is to close the application and start it again going to Frame 2 by selecting a new set of data in Frame 1. That is, until we know the alternative way!!

As mentioned earlier, SAS software has the ability of running more than one AF Application at the same time. So, if we call Frame 2 by issuing an AFA command rather than a Call Display routine, we shall in effect have two applications running at the same time, Frame 1 as one application that invokes Frame 2 as a separate application. Frame 2 will now successfully regenerate and reload the `$COMMON.` macro each time as it is continuously closed down and restarted. Closing Frame 2 simply returns you back to the other application (Frame 1) which is sitting in the background.

The AFA command can be issued from the SCL behind Frame 1 in the following manner:-

```
Call Execcmmdi('afa c=system.main.frame2.frame');
```

The CALL EXECCMDI routine issues SAS commands immediately (this is different to the CALL EXECCMD routine which issues the command

after the SCL label has completed). And the AFA command invokes an application in its own Application Workspace. Note, however, that the user can click on Frame 1 and continue interacting with it whilst Frame 2 is still active. You may wish to add code to prevent this.

Amadeus SAS Consultancy Team

Options?

Q: I'm familiar with the use of the OPTIONS statement and parameters like PAGESIZE and DATE, but I can't get some of the options parameters to work (and some I simply don't understand). For instance, WHERE and IN. A: Your confusion is caused by the fact that SAS software has three different types of options: system options, data set options, and statement options. I'll explain the three individually.

System options are instructions that control the way that the SAS System operates. System options remain in effect for all subsequent DATA steps and PROCs in a SAS session until specifically changed again. System options are specified with the OPTIONS statement (or the OPTSETx functions).

Data set options are set in brackets after data set names. They specify actions that apply only to the processing of the SAS data set with which they appear. For example, they let you rename variables, specify variables to be included or dropped in later processing, and select only the first or last n observations for processing.

Statement options are specified only in a given SAS statement or statements, and they affect only that statement or step. For example, the HEADER= option in the FILE statement is a statement option.

The following short example shows usage of the PAGESIZE system option and the INDEX, KEEP, and RENAME data set options.

```
Options pagesize=40;
Data citidata (index=(date));
  Set sashelp.citiday (keep=date snydjcm rename=(snydjcm=index));
  ;
  etc...
```

Andrew Ratcliffe

Formats, Options, and Functions

The \$QUOTEw. format writes data values enclosed in double quotation marks.

Contacts

Contributors and Contacts

Amadeus Software Ltd	Tel: 01993-775454 / Fax: 01993-700577 Email: enquiries@amadeus.co.uk
Peter Crawford	Crawford Software Consultancy Ltd Tel: 020-8656-1794 Email: Peter@crawfordsoftware.demon.co.uk
David Shannon	Quintiles, Battle Tel: 01424-776229 Email: david.shannon@qbat.quintiles.com
David Trenery	Aventis Pharma Ltd, Denham UK Tel: 01895 837803 / Fax: 01895 837765 Email: david.trenery@aventis.com
VIEWS Web Site	www.views-uk.demon.co.uk
Phil Mason Chairman	Tel/Fax: 01491-834615 Email: phil_mason@email.com 16 Wood Street, Wallingford, Oxfordshire. OX10 0AY
Bruce Bovill SAS Liaison	Tel: 01628-404347 Fax: 01628-404303 Email: Bruce.Bovill@suk.sas.com Web: www.assisi.demon.co.uk
Andrew Ratcliffe Newsletter Editor	Tel/Fax: 01322-525672 Email: andrew@ratcliffe.demon.co.uk Web: www.ratcliffe.demon.co.uk

The SORTEDBY= data set option enables you to specify how the data set is currently sorted. This information is stored with the data set; use PROC CONTENTS to see how a data set is sorted. The SAS System uses this information to avoid unnecessary sorting of data.

The COMPBL() function removes multiple blanks between words in a character string. Each occurrence of two or more consecutive blanks is translated into a single blank.

News

SAS-L: Correction

The article on SAS-L in the last issue of *VIEWS News* contained incorrect email addresses for subscribing to the list. Please accept my apologies for this error; thank you to those who pointed-out the mistake.

To subscribe to SAS-L, send an email to listserv@listserv.uga.edu, and (as correctly stated in the previous issue) include the following text in the body (not the title) of the message: subscribe sas-l. You may add your name to the end of the subscribe command if you wish, but that is optional.

While we're on the subject, you might like some more information. To get general help about using Listserv software, send an email to the above address and include just the following in the body of the message: help. The email response gives brief information on subscribing, unsubscribing (signing-off), querying your subscription options, and getting more information about lists.

However, a word of warning should follow this information. SAS-L is a high bandwidth conference, i.e. you may be inundated with email. I don't personally use SAS-L in this way. I use it from a newsreader (CIX/Ameol in my case, but MS-Outlook Express and others are fine too). That has two advantages: a) it keeps your mail client's Inbox tidy, and b) you see some threading of conversations so that it is easier to follow a conversation. The SAS news group (**news:comp.soft-sys.sas**) accessed by my newsreader is "mirrored" with SAS-L, i.e. all SAS-L messages are sent to the news group, and all of the news group's messages are sent to SAS-L

In Brief

- Check-out the new resources site at www.sashelp.com for some very useful SAS-related items. The site was created by *VIEWS News* contributor David Ward
- For some code to accurately calculate age in only one line (and for a useful discussion of true leap years too) see service/doc/periodicals/sascomm/4q98/SC4Q98_38.htm on the SAS Institute web site
- This year's SUGI in Indianapolis is only two months away. If you haven't already booked, remember that last year's event was ranked 5th most valuable of all user conferences by Computerworld in September 1999!

Views News is published quarterly by VIEWS - the UK's independent SAS user group. VIEWS is a non-profit organisation. Our Mission Statement and Rules can be viewed on our web site together with a list of current committee members

Diary of Events

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 software.

February 2000	May 2000
? DM SIG sub-group (provisional)	1 VIEWS News 10th Edition
April 2000	7-10 Pharma SUG Seattle, Washington, USA
9-12 SUGI, Indianapolis, USA Pharma SIG (provisional)	June 2000
	20-23 SEUGI, Dublin, Ireland
	September 2000
	? VIEWS#6 conference (provisional)

SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASA**SAS**@**AS**ASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA
SASASASAS@ASASASASA

Looking for a contract? Or looking for quality people?
Call us now - we know SAS® software inside out!

Official sponsors of

VIEWS
The Independent
UK SAS User Group

<http://www.asaltd.co.uk>


recruitment & contract consultants

ASA Ltd. First Floor, East Wing, Auckland House, New Zealand Avenue, Walton Upon Thames, Surrey KT12 1PL.
Contact Richard Field at ASA on Tel: 01932 701801 Fax: 01932 701800 Email: sas@asaltd.co.uk

SAS is a registered trademark of SAS Institute Inc. Cary, North Carolina, USA.