Paper TT-10

Configuring SAS to Work for You, Part 2: More Programming Shortcuts

Kim Truett, KCT Data, Inc., Alpharetta, GA

ABSTRACT

Several years ago, I presented "DO NOT EDIT BELOW THIS LINE" (Configuring SAS to work for you). That paper contained a variety of tips for tweaking SAS options to change the way it works so that programmers can be more efficient. This paper presents new tip and tricks that I have learned over the past three years. Most of these are Windows based SAS tricks and tips. This paper doesn't present any new or undocumented features; rather it is a compilation of some programming shortcuts and hints.

Introduction

SAS is loaded with cool tricks, easy short-cuts, and new features with each release. These features can help make programming more accurate, reduce typing or do things that were, previously, quite complex to program. This paper is a summary of some of these programming shortcuts that I have learned about and think are most useful. This paper doesn't present any new or undocumented features; rather it is a compilation of programming shortcuts and hints that I have learned from many sources.

I. Programming shortcuts

A. PROPCASE WITH A SECOND ARGUMENT

SAS help states that a second argument can be used with Propcase which "specifies one or more delimiters that are enclosed in quotation marks. The default delimiters are blank, forward slash, hyphen, open parenthesis, period, and tab."

This means that Propcase will capitalize any characters that follow the delimiters that are specified.

The default `PROPCASE(variable)` capitalizes each word after a space – creating a Title.

```
variable="this is the title of my new paper.";
PROPCASE(variable): This Is The Title Of My New Paper.
```

Omit the space and change the delimiter and this creates sentence case:

```
PROPCASE(variable,'/'): This is the title of my new paper.
```

In this example, the c is not capitalized by this method, because it follows a period, rather than a space.

```
variable=Mr. Smith Goes To Washington D.c.
PROPCASE(variable,' .'):  Mr. Smith Goes To Washington D.C.
```

One final example:

```
variable="my friend, ruby-fae o'connell, wishes computers would capitalize her name
correctly.";

PROPCASE(variable,' '): variable=My Friend, Ruby-fae O'connell, Wishes Computers Would
Capitalize Her Name Correctly.

propcase(variable," -'"): variable=My Friend, Ruby-Fae O'Connell, Wishes Computers Would
Capitalize Her Name Correctly.
```

B. THISPAGE, LASTPAGE and PAGEOF FOR ODS RTF FILES

For inserting page numbers onto RTF output, SAS now has three escape character commands. In the code below, ^ is the RTF escape character, then the bracketed {} commands insert the page numbers when the RTF file is created.

- thispage – inserts the current page number
- lastpage – inserts the number of pages in the document
- pageof – inserts "x of y" style text

(&blank_ inserts a long series of blanks once, and keeps the title lines themselves shorter.)

```
%let blank_ = %str(                                                                    ) ;
TITLE1 J=l "My New Protocol Listings  &blank_ Page ^{thispage} of ^{lastpage}";
```
O r
```
TITLE1 J=l "My New Protocol Listings  &blank_ Page ^{pageof}";
```

C. CATS AND MORE CATS

Many of us are familiar with concatenating variables by using the old || trick such as:

```
Var1='Abyssinian';
Var2='Siamese';
newvar = (trim(left(var1))||' – '||(trim(left(var2));
newvar: "Abyssinian – Siamese"
```

But the four CAT functions do the same thing – but simpler:
CAT: Concatenates character strings without removing leading or trailing blanks
CATS: Concatenates character strings and removes leading and trailing blanks
CATT: Concatenates character strings and removes trailing blanks
CATX: Concatenates character strings, removes leading and trailing blanks, and inserts a specified separator

```
newvar = CAT(var1, ' – ', var2);
newvar: "Abyssinian – Siamese"
```

and CATX allows separators to be specified, such as:
```
newvar = CATX(' –', var1, var2);
newvar: "Abyssian – Siamese"
```

One advantage of this method is that CAT can utilize variable lists (var1-var8) which simplifies specifying a lengthy list of variables:
```
Var1='Abyssinian';
Var2='Burmese';
Var3='Chartreux';
Var4='Havana Brown';
Var5='Korat';
Var6='Ocicat';
Var7='Russian Blue';
Var8='Siamese';
```

as compared to
```
newvar = trim(left(var1))||', '|| trim(left(var2))||', '|| trim(left(var3))||', '||
trim(left(var4))||', '|| trim(left(var5))||', '|| trim(left(var6))||', '||
trim(left(var7))||', '|| trim(left(var8));
```

try: `newvar = CATX(', ', var1-var8);`

using either SAS code the results are the same:

```
newvar: "Abyssinian, Burmese, Chartreux, Havana Brown, Korat, Ocicat, Russian Blue,
Siamese"
```

(in case you are curious, there is no DOG function)

D. INSERTING INFORMATION ABOUT A VARIABLE'S RANGE INTO LABELS

For categorical variables, sometimes the range of *possible* values is larger than the range of *actual* values in the data. This can be identified with the min and max functions as part of the results, but here is a way to simplify the output.

This bit of code extracts the actual range of variable values and puts it into a variable:

```
PROC SQL NOPRINT;
  SELECT CATX('-', min(lotest_1), max(lotest_1)) into: range1 from task_1;
QUIT;
```

Then used (for example) as: `"Cortical Opacification (Patient range is %trim(&range1))"`

To create:

```
Cortical Opacification (Patient range is 0-5)
    N
    Mean
    Std
    Median
```

Sure – this can be presented with a min and max below the median – but this method eliminates two lines, and adds the information in a very user friendly method.

II. Better Programming Practices

A. ATTRIB - A BETTER WAY TO DEFINE DATASETS

When setting up a new dataset, the most common method is:

```
DATA MYLIB.DM;
LENGTH SUBJID 8. BRTHDT  $19. SEX  $1. ETHNIC  $12. RACE  $45. ;
LABEL SUBJID = " Subject Identifier for the Study ";
      BRTHDTC = "Date/Time of Birth";
      SEX     = "Sex";
      ETHNIC  = "Ethnicity";
      RACE    = "Race";
```

But this method shown below is easier to track and to debug – because it keeps the format and the label together.

```
DATA MYLIB.DM;
ATTRIB SUBJID  format=$25. label="Subject Identifier for the Study"
       BRTHDTC format=$19. label="Date/Time of Birth"
       SEX     format=$2.  label="Sex"
       RACE    format=$41. label="Race"
       ETHNIC  format=$22. label="Ethnicity"
```

B. OBTAINING DATASET DATA – WHAT VERSION OF THE DATASETS WAS THIS GENERATED FROM?

A good programming practice for any table or listing is to include the date of the datasets in the footer so that the version of the dataset is documented as well as the name of the dataset. The SAS dictionary tables are invaluable for obtaining this information quickly. The datastep inserts the date of one dataset into a macro variable, which can then be called in the footer.

```
data _null_;
   set sashelp.vtable(where=(libname = "CRF" and memname = "DM"));
   call symput('modate',put(datepart(modate), date9.));
run;

Footnote "Generated using datasets dated %str(&modate)";
```

C. EXTRACTING DATA ABOUT THE VARIABLES

The SAS dictionary tables can also be accessed via SAS functions to extract metadata about SAS variables.
SAS documentation has a complete list, but here are the ones that I think are most useful:

VNAME = name of variable
VLABEL = label of variable
VFORMAT = format of variable
VINFORMAT = informat of variable
VLENGTH = length of variable
VTYPE = type (character or numeric) of variable

For example:

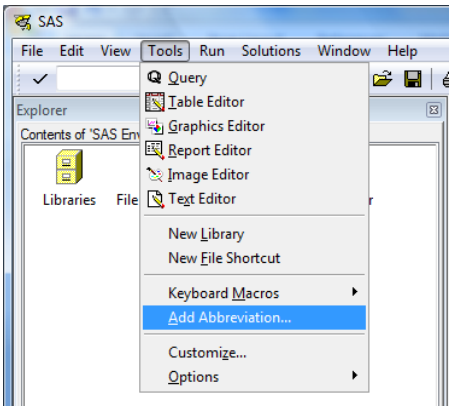
```
TexttoPrt = VNAME(varnameC)||': '||VLABEL(varnameC);
```

Uses for these functions might include clarifying the content of tables and listings or producing dataset documentation.

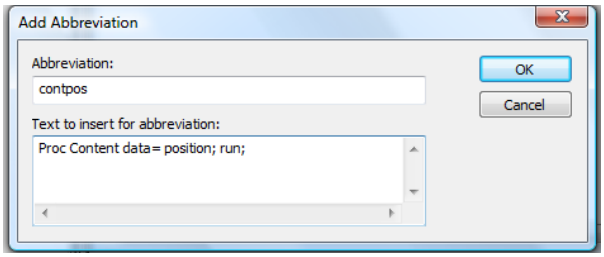
D. ADD ABBREVIATIONS

Programming can become more accurate with less typing and fewer errors using abbreviations. These are programming shortcuts that will insert pre-specified bits of SAS code.

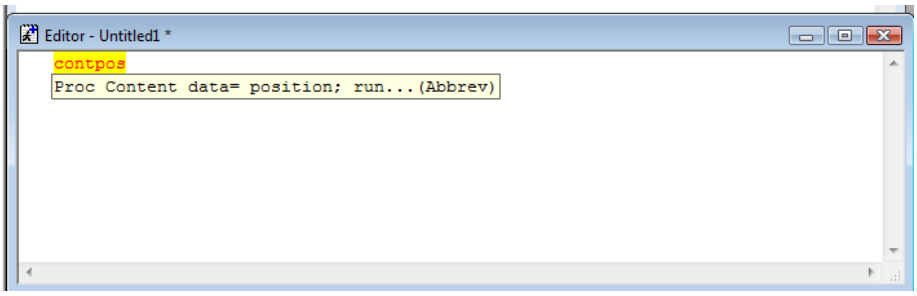
- select Tools-Add Abbreviations



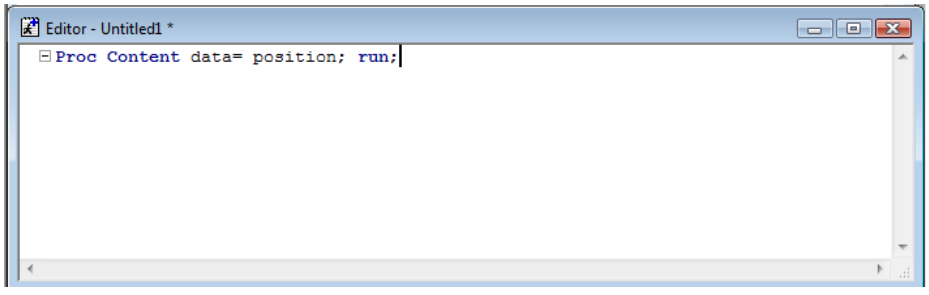
Top box – type the abbreviation (this is case sensitive – so be aware of this)
Second box – type the code that should appear



Then whenever the string is typed into the SAS editor, SAS will suggest the longer code,



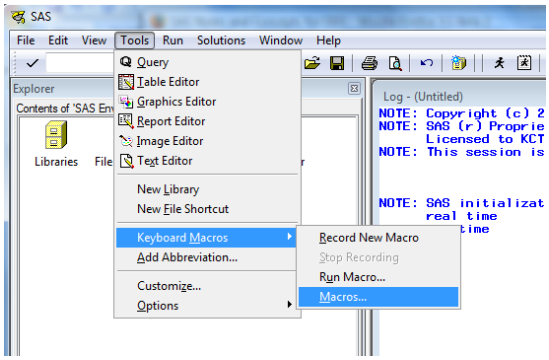
and pressing 'Enter' will insert the code or the programmer can continue typing, if the suggested text is not appropriate.

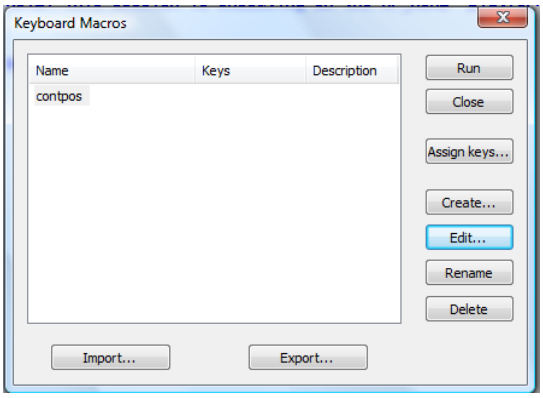


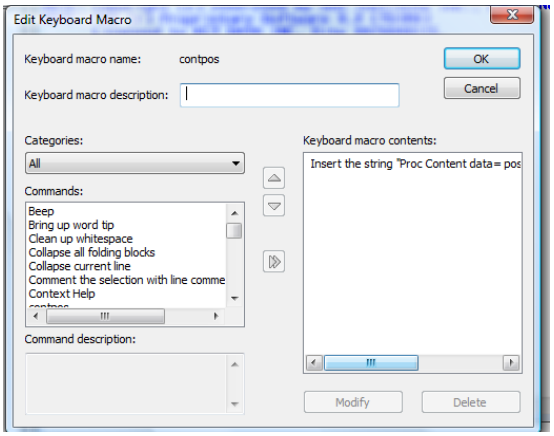
Now, just insert the dataset name, and the code is already written.

To edit (or recall) abbreviations that were previously defined, the abbreviations are stored with macros.

- select Tools-Keyboard Macros – Macros; highlight the abbreviation and select 'Edit'







This shows that the Abbreviations are actually Macros that 'Insert the string…'

And from these screens, abbreviations can be renamed, edited or deleted.

E. USING BOOKMARKS

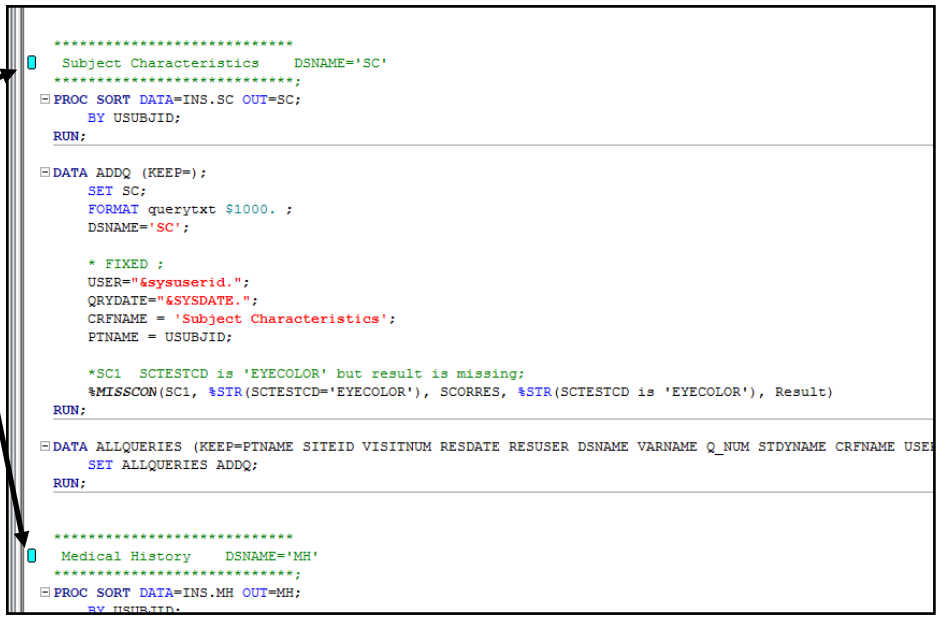
If a program is long, but divided into distinct sections, adding book marks can help the programmer move efficiently from section to section. So, for example, for a long program that runs edit checks for each dataset, bookmarks could be placed at the top of each dataset section within the program to allow rapid navigation through the sections.

Bookmarking is assigned to the F2 key:

Highlight the location where the bookmark should be added. Crtl+F2 both bookmarks and un-bookmarks lines.

Small marks are inserted to show bookmarks are inserted.

F2 and shift F2 move forwards and backwards through the bookmarks that are defined.



```
*****************************
 Subject Characteristics    DSNAME='SC'
*****************************;
PROC SORT DATA=INS.SC OUT=SC;
     BY USUBJID;
RUN;

DATA ADDQ (KEEP=);
     SET SC;
     FORMAT querytxt $1000. ;
     DSNAME='SC';

     * FIXED ;
     USER="&sysuserid.";
     QRYDATE="&SYSDATE.";
     CRFNAME = 'Subject Characteristics';
     PTNAME = USUBJID;

     *SC1  SCTESTCD is 'EYECOLOR' but result is missing;
     %MISSCON(SC1, %STR(SCTESTCD='EYECOLOR'), SCORRES, %STR(SCTESTCD is 'EYECOLOR'), Result)
RUN;

DATA ALLQUERIES (KEEP=PTNAME SITEID VISITNUM RESDATE RESUSER DSNAME VARNAME Q_NUM STDYNAME CRFNAME USE
     SET ALLQUERIES ADDQ;
RUN;


*****************************
 Medical History    DSNAME='MH'
*****************************;
PROC SORT DATA=INS.MH OUT=MH;
     BY USUBJID;
```

III. New SAS 9.2 Functions

A. SAS_EXECFILENAME and SAS_EXECFILEPATH

Copying or reusing a program with a new name, required that the path and program name information in footers had to be updated.

Old way:

```
footnote5 J=l "Program: C:\Study\Listings\VisitList.sas";
footnote6 J=l "Output: C:\Study\Listings\pgms\VisitList_&sysdate..rtf";
```

But, SAS v9.2 now stores the name of the path and file in environmental variables for the program which is executing and this metadata can be obtained using a %sysget as shown here.

```
data _null_;
* For file name of SAS program;         %put %sysget(SAS_EXECFILENAME);
* For path name where program located;  %put %sysget(SAS_EXECFILEPATH);
run;
```

These functions return a null value if the program is not yet saved.

New way:

```
footnote5 J=l "Program: %sysget(SAS_EXECFILEPATH) %sysget(SAS_EXECFILENAME)";
footnote6 J=l "Output: %sysget(SAS_EXECFILEPATH)%sysget(SAS_EXECFILENAME)_&sysdate..rtf";
```

Now when the program is moved, copied or reused the footnote will still be correct.

B. INDSNAME

SAS now also can save the name of the input dataset using the INDSNAME option on the set statement. This eliminates the need to use the IN option and set the name manually;

Old Way:

```
data aes;  set p_meds (in=inp)  c_meds (in=inc);
  if inp then dsn = p_meds;
  if inc then dsn = c_meds;
run;
```

New Way:

```
data aes; set p_meds c_meds indsname=dsn;
```

More new features can be found in the online document published by SAS at: http://support.sas.com/documentation/whatsnew/index.html

Conclusion

We all get into programming patterns – copying programs already written and reusing SAS code. While this saves time and can reduce errors (if the original programs were properly validated). I would encourage programmers at all levels to continue to review other programmer's code, subscribe to mailing lists like SAS-L list and, of course, attending programming meetings in order to learn better ways to program and keep skills sharp.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:
Kim Truett
11585 Jones Bridge Road; Suite 420-115
Alpharetta, GA 30022
770.372.0989
Email: KCTData@comcast.net