

Paper SD06

Old But Not Obsolete: Undocumented SAS® Procedures

Barbara B. Okerson, Health Management Corporation, Richmond, VA

ABSTRACT

Proc Spell? Proc Neighbor? Proc Browse? Through the years a number of SAS procedures have disappeared from the manuals, mainly because their functionality was picked up by newer, more robust, procedures and features of the SAS system. But sometimes, simpler can be better. This paper looks at these and other SAS procedures of the past that are not in the current documentation, but still work, and addresses situations where they can yet be useful.

Examples were developed with version 9.1.3 of SAS executing on a Windows XP platform. Except as noted, examples are not version or platform specific and can be adapted by all levels of SAS users.

INTRODUCTION

Health Management Corporation (HMC) is one of the nation's largest, most experienced providers of integrated care and total health solutions. Since 1983, HMC has offered members and clients comprehensive programs and services intended to empower members to take control of their health and see positive results. Through a population-based approach, HMC's disease management programs provide solutions for prevention, chronic condition support, lifestyle management and complex condition care. HMC's mission is to help improve the health and financial outcomes of its clients and members through innovative health solutions that consider every single member at his/her level of care.

Within HMC, the analytics group is responsible for the design, development, validation, and implementation of disease management predictive models. Additionally, the analytics group provides research, business intelligence, and program evaluation support to measure the effectiveness of the services provided, analyze potential for future products and add to general knowledge of the disease management arena. SAS is the major analytic software used by HMC and, as such, all of the examples in this paper were produced with SAS software in conjunction with this work.

UNDOCUMENTED SAS PROCEDURES THAT CAN BE USEFUL

As SAS has added functionality and robustness to current procedures, as well as adding new procedures to meet the needs of the business community to maintain their position as the leader in business intelligence software and services, some early procedures have truly become obsolete. To maintain compatibility for code and applications developed using these procedures, even most of these procedures still work, although no longer documented. Procedures in this category would include Proc FUNCAT and Proc RSQUARE from SAS/STAT. Other undocumented procedures, while their functionality has been assumed by other features or procedures, still have utility – either through ease of use, procedure assumptions, or efficiency. Procedures in this last category are the subject of this paper.

There are many more undocumented SAS procedures than are covered here and many of these may also prove useful. This paper restricts its scope to those that have been found to provide benefit for my applications.

PROCEDURE ONE – PROC SPELL

According to the SAS-L archives, SAS acquired a spell checker when they purchased the Lattice Corporation prior to rewriting the SAS package in the C programming language for V6. SAS incorporated it into their system as Proc SPELL. To use the procedure, save the text that you want to spell-check as a text file. Then invoke Proc SPELL.

```
Proc SPELL in=spelt dictionary=words.spell.mywords verify suggest;  
Run;
```

where *spelt* is a fileref for the file containing the text to be spell checked and *words* is a libref pointing to the SAS library where the spell checker list is located. The *suggest* option lists spelling suggestions from the dictionary. *Verify* is the default and is assumed unless a custom catalog is being created or updated with Proc SPELL.

To be more useful, a custom dictionary of industry relevant words that are not in SAS' dictionary can be created. This dictionary is a SAS catalog entry and is created with the spell procedure. To create a dictionary:

- Create a text file with the words that you want to define.
- Put each word on a separate line.

- Point to the SAS file that holds the dictionary catalog (if updating) or create a new catalog.
- Point to the location of the custom word list.

EXAMPLE ONE – Proc SPELL

In the health care field many standard abbreviations are used on a daily basis. In this example, a dictionary is created which includes only those health care condition abbreviations that should appear in the disease management database. The wordlist was created as condition.txt and includes the following abbreviations that many other programs read. The spelling file contains only the abbreviations.

- AST – Asthma
- CAD – Coronary Artery Disease
- CHF – Congestive Heart Failure
- COP – Chronic Obstructive Pulmonary Disease
- DIA – Diabetes
- MAT – Maternity

The code below creates the custom word catalog. This newly created catalog includes those entries in the SAS catalog in addition to the added custom words. To add words to a custom catalog change create to update.

```
Proc SPELL words="r:\bokerson\sesug 2007\condition.txt"
  create dict=work.mycatalog.spell;
Run;
```

A file containing the variables (condition1 – condition 5) were written to a text file for running against the word list to identify any spelling errors in the file. Then the following code was run:

```
Proc PRINTTO print="r:\bokerson\sesug 2007\sp_cond.txt" new;
Run;
Proc SPELL wordlist='r:\bokerson\sesug 2007\spellcase.txt' verify
  dict=work.mycatalog.spell;
Run;
Proc PRINTTO print=print;
Run;
```

Proc PRINTTO was used to divert the output to an external file. Default output writes to the output window. The output from the file r:\bokerson\sesug 2007\sp_cond.txt is below:

Unrecognized word	Freq	Line(s)
DIAB	4	14, 32, 39, 234
CHHF	1	18
COPD	5	39, 234, 235, 236, 241
HF	1	349

PROCEDURE TWO – PROC BROWSE

The browse procedure was created to allow users to read a SAS data set both interactively and in batch mode. This procedure does not allow modification of the SAS data set. When in an active SAS session, there is no need to run this procedure since the SAS data set is readily available for viewing and reading.

Statements that can be used with Proc BROWSE include:

- FIND – searches a ranged for occurrences of a variable or variables.
- LOCATE – searches for occurrences of a single value in a range of observations.
- NAME – indicates a variable to search for matches.
- SEARCH – searches for occurrences of a string or strings within a group of character variables.
- STRING – names the variable or variables to be used by the SEARCH command.
- VERIFY – displays changes according to the actions taken.
- LIST – lists the values of one or more variables in a range of observations.
- HELP – available if procedure used interactively to get help from the procedure.

Other statements that can be used with Proc Browse include Format, Informat and some positioning statements for interactive use.

The NAME and LOCATE statements work together to name a variable and a value to search a SAS data set for one or more occurrences of the value. The STRING and SEARCH statements work together to search variables for the occurrence of a string anywhere within that value.

EXAMPLE TWO – PROC BROWSE

In this example, Proc BROWSE is used in batch mode to check a data set for the existence of a health plan member. The advantage of Proc Browse for this is that it saves time. The following SAS program uses Proc BROWSE to check a data set with several million records for the occurrence of a member ID.

```
Libname r 'r:\bokerson';
Proc BROWSE data=r.medical;
NAME MemberID;
LOCATE 1,last 144490885;
Run;
```

The batch program is invoked in Windows Explorer as follows:

- Right-click on the SAS program.
- From the pop-up menu, select: *Batch Submit With SAS 9.1.*
- This SAS program file should have a .sas extension.
- The program log file with the results will given the same name with a .log extension and be located in the same folder as the .sas program.

The results are as follows (bolding is mine):

```
NOTE: Copyright (c) 2002-2003 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) 9.1 (TS1M3) Licensed to HEALTH MANAGEMENT CORP, Site XXXXXXXXXX.
NOTE: This session is executing on the NET_ASRV platform.
NOTE: SAS 9.1.3 Service Pack 4
NOTE: SAS initialization used:
      real time          0.45 seconds
      cpu time           0.28 seconds

1      libname r 'r:\bokerson';
NOTE: Libref R was successfully assigned as follows:
      Engine:           V9
      Physical Name:    r:\bokerson
2      proc browse data=r.medical;
NOTE: Welcome to the BROWSE Procedure. Begin entering commands.
3      name memberid;
4      locate 1,last 144490885;
NOTE: Found at OBS 1166639.
5      Run;
NOTE: PROCEDURE BROWSE used (Total process time):
      real time          10.31 seconds
      cpu time           3.37 seconds

NOTE: Exiting from the BROWSE Procedure.

NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time          10.78 seconds
      cpu time           3.65 seconds
```

PROCEDURE THREE – PROC NICKNAME

The nickname procedure is a tool created for the convenience of SAS system administrators. The intent of this procedure was to allow SAS administrators to use SAS aliases for adding new modules to current installations. While the utility of the nickname procedure for the original purpose is possibly still viable, the procedure can be used for a quick check of the SAS engines and access methods available. When used in this manner, Proc NICKNAME will often reveal new, experimental, and undocumented engines that can be used.

Syntax for adding a module under version 9 is listed below. In order for the following code to work it is essential for the process to have the rights to write to the SAS library SASHELP. The default privilege for this library for all versions of SAS is for READ rights only.

The code follows.

```
Proc NICKNAME cat=sashelp.core;
Add nickname=infomaps Module=sasioime Release="9"
    Desc="SAS Information Maps LIBNAME Engine" Preferred Engine;
Run;
```

Options that can be used on the Proc NICKNAME statement include:

- ACCESS – restricts search to access methods.
- AMETHOD – restricts search to access methods.
- ENGINE – restricts alias list to engines.
- CATALOG – restricts search to catalog specified. Default is sashelp.core. Order of options matters here. The catalog needs to be listed before options using this catalog.
- CALL –
- FUNCTION - sets *ffalias* as default alias type for function aliases.
- FORMAT – sets *ffalias* as fault alias type for format aliases.
- INFORMAT - sets *ffalias* as default alias type for infomat aliases.
- SUBROUTINE -

EXAMPLE Three – Proc NICKNAME

By default, Proc NICKNAME queries the sasuser.core table and provides the alias names for the various SAS entries in the catalog that are available. No write access is necessary for this application. This example illustrates this use and restricts the query to engine aliases only. The results follow the code.

```
Proc NICKNAME eng;
Run;
```

Current Catalog: SASHELP.CORE					
	Nickname	Module	Type	Fileformat	Description
P M	ACCESS	SASIOMDB	ENG	7	SAS/ACCESS Interface to PC Files
M	ACCESS99	SASECRSP	ENG		Read engine for CRSP ACCESS97 database
M	BASE	SASE7	ENG	7	Base SAS I/O Engine
P M	BLOOMBRG	SASIOBLB	ENG	9	SAS/Access Interface To Bloomberg
P M	BMDP	SASBMDPE	ENG	607	BMDP Save file engine
P M	CRSPACC	SASECRSP	ENG		Read engine for CRSP ACCESS97 database
P M	CVP	SASECVP	ENG	9	Character Variable Padding Engine
P M	DB2	SASIODBU	ENG	7	SAS/ACCESS Interface to DB2
P M	EXCEL	SASIOXLS	ENG	7	SAS/ACCESS Interface to PC Files
P M	FAMECHLI	SASEFAME	ENG		Seamless libname interface to FAME db
P M	HAVERDLX	SASEHAVR	ENG	9	Read engine for Haver Analytics DLX db
P	IMDB	SASEIMDB	ENG	9	In Memory Database Engine
P M	META	SASIOMET	ENG	7	Metadata engine
P M	MYSQL	SASIOMYL	ENG	7	SAS/ACCESS Interface to MySQL
P M	ODBC	SASIOODB	ENG	7	SAS/ACCESS Interface to ODBC
P	OLAP	SASEOLAP	ENG	9	SQL Passthru Engine for OLAP
P M	OLEDDB	SASIOOLE	ENG	7	SAS/ACCESS Interface to OLE DB
P M	ORACLE	SASIOORA	ENG	7	SAS/ACCESS Interface to Oracle
P M	OSIRIS	SASOSIRI	ENG	607	OSIRIS Data File engine
P	R3	SASIOSR3	ENG	9	SAS Engine for SAP R/3
P M	REMOTE	SASIORMT	ENG	7	SAS/SHARE Remote access engine
P M	REMOTE8	SASI8RMT	ENG	7	SAS/SHARE V8 Remote access engine
P M	REUTERS	SASEREUT	ENG	612	Reuters financial market data interface
M	SASIOOS2	SASIODBU	ENG	7	SAS/ACCESS Interface to DB2
P M	SPDE	SASSPDE	ENG	7	Scalable Performance Data Engine
P M	SPSS	SASSPSS	ENG	607	SPSS Save File engine
P	SQLVIEW	SASESQL	ENG	607	SQL view engine
M	SXLE	SASEXML	ENG	8	W3C XML input/output engine
P M	SYBASE	SASIOSYB	ENG	7	SAS/ACCESS Interface to Sybase
P M	TERADATA	SASIOTRA	ENG	8	SAS/ACCESS Interface to Teradata
P	TRACE	SASETRC	ENG	7	Version 7 trace engine
P M	V6	SASEB	ENG	607	Base SAS I/O Engine
P M	V604	SASIO602	ENG	606	Base SAS I/O Engine - 6.06 defaults
M	V607	SASEB	ENG	607	Base SAS I/O Engine
M	V608	SASEB	ENG	607	Base SAS I/O Engine
M	V609	SASEB	ENG	607	Base SAS I/O Engine
M	V610	SASEB	ENG	607	Base SAS I/O Engine
M	V611	SASEB	ENG	607	Base SAS I/O Engine
M	V612	SASEB	ENG	607	Base SAS I/O Engine
M	V7	SASE7	ENG	7	Base SAS I/O Engine
M	V701	SASE7	ENG	7	Base SAS I/O Engine
M	V8	SASE7	ENG	7	Base SAS I/O Engine
P M	V9	SASE7	ENG	7	Base SAS I/O Engine
P M	XML	SASEXML	ENG	8	W3C XML input/output engine
P M	XPORT	SASV5XPT	ENG	607	Version 5 transport datasets

In the example above, the first two columns do not have column labels. SAS does not label these columns in the output. They are interpreted as follows:

- The first column contains either a P or is blank. The P stands for preferred. Engines can have multiple nicknames and this is the preferred nickname.
- The second column contains either an M or is blank. The M indicates whether or not this nickname can be used in a filename or libname statement.

PROCEDURE FOUR – PROC NEIGHBOR

Proc NEIGHBOR is an undocumented SAS/STAT procedure that performs nearest neighbor discriminant analysis using the nearest neighbor rule and is especially suitable for classification when the classes have radically non-normal distribution. The functionality of this procedure was replaced by adding the METHOD=NPART option to Proc DISCRIM in SAS/STAT, provided the identical options and methods are selected.

As a non-parametric model, SAS procedure Proc NEIGHBOR is valuable in avoiding assumptions. As a discriminant procedure, it is based on prior probability (Mahalanobis or Euclidean distance) and posterior probability (proportional prior probability of observations in k nearest neighbors). The k-nearest neighbor rule is a method that classifies unlabeled examples based on their similarity with examples in a training set. The new observation is classified by finding the closest observation in the calibration data set, then assigning the new observation to the group from which the majority of observation's nearest neighbors came.

Statements that can be used with Proc NEIGHBOR include:

- CLASS – defines the classes (required).
- VAR – lists variables to be included.
- ID – identifies the value used for observation identification in classification results.
- PRIORS – signifies that prior probabilities are not equal.
- TESTCLASS – names the variable in the test data set that is used to determine misclassification.
- TESTID – names the observation identification variable for classification results.
- BY – requests separate analyses for groups defined by the BY variable.

Options available for the Proc NEIGHBOR statement include:

- IDENTITY – specifies use of Euclidean distances.
- K – specifies a k-value for k-nearest neighbor rule.
- LIST – prints classification results for each observation.
- LISTERR – prints only misclassified observations.
- THRESHOLD – specifies minimum possible posterior probability for classification.
- TESTLIST – lists all the observations in the testdata= data set.
- TESTLISTERR – lists only misclassified observations in the testdata= data set.

EXAMPLE Four – Proc NEIGHBOR

By default, Proc NEIGHBOR is suitable for use for classification when the classes have radically non-normal distribution. This is often true when identifying incidence of disease for disease management. In this example, patients were classified by the following criteria: known diagnosis of chronic kidney disease (CKD), gender, age group and a series of lab values. The same information for additional patients was run for additional undiagnosed patients, who were tested for CKD classification.

Code:

```
Proc NEIGHBOR k=1 data=ckd_ID testdata=testckd testlist;  
Class CKD;  
TestClass CKD;  
TESTID memberid;  
Var gender agegroup lab1-lab5;  
Run;
```

As illustrated in this example, Proc NEIGHBOR can use a combination of continuous (lab values), binary (ckd), nominal (gender), and ordinal (age group) variables. For this analysis, the nearest neighbor (k) was set to 1, for classifying the new observations, since the classes are unbalanced and a larger k does not yield better results for unbalanced classes. Mahalanobis distance (the default) was used.

The partial SAS output that follows gives a table of the numbers of correctly and wrongly classified observations.

The DISCRIM Procedure Classification Results for Test Data: WORK.TESTCKD Classification Results using Nearest Neighbor Squared Distance Function $D^2(X,Y) = (X-Y)' COV^{-1} (X-Y)$ Posterior Probability of Membership in Each ckd $m_k(X)$ = Proportion of obs in group k in nearest neighbor of X $Pr(j X) = \frac{m_j(X) PRIOR_j}{\sum_k (m_k(X) PRIOR_k)}$ Posterior Probability of Membership in ckd					
MemberID	From CKD	Classified into ckd		N	Y
2001	Y	N	*	1.0000	0.0000
2002	Y	N	*	1.0000	0.0000
2003	Y	N	*	1.0000	0.0000
2004	Y	N	*	1.0000	0.0000
2005	Y	Y		0.0000	1.0000
* Misclassified observation					

When running Proc NEIGHBOR in version 9.1.3 of SAS, the output is labeled as DISCRIM procedure, rather than as the NEIGHBOR procedure, however only Proc NEIGHBOR procedure syntax can be used.

NOTE: While this is the only undocumented SAS/STAT procedure discussed in this paper, there are others. Some other favorites of the past that are still available include:

- Proc FUNCAT – functions of categorical responses as a linear model
- Proc RSQUARE – performs all possible regressions for one or more dependent variables and a number of independent variables.
- Proc STEPWISE – provides five methods for stepwise regression.

UNDOCUMENTED SAS PROCEDURES TO USE WITH CAUTION

The procedures listed above can be used without concern for data loss or damage to the system. Some available, but undocumented, SAS procedures that can be used need to be used with great care since changes made by these procedures are permanent. Without documentation or help files to fall back on, the likelihood of fatal errors is increased. The two SAS procedures in this category discussed in this paper are Proc EDITOR and Proc DELETE.

PROCEDURE FIVE – PROC EDITOR

Proc EDITOR is a procedure that allows users to make changes to a SAS data set without running a data step. However, a new data set is NOT created with Proc EDITOR. Changes made by Proc EDITOR operate directly on the existing SAS data set and cannot be undone without commands reversing the edit.

Additionally, please note that deleted observations are set to missing rather than actually deleted.

Proc EDITOR uses all the statements listed above as available in Proc BROWSE. Additional statements unique to the editor procedure include:

- REPLACE – changes one or more values of specified variables in a range of observations.
- ADD – adds a new observation to the end of the data set.
- DELETE – sets all the variables in a specified range of observations to missing.
- DUP – duplicates observations for a specified range in the data set.

If multiple edits are requested in a single Proc EDITOR step, an END statement needs to be placed between each edit, since edits are interpreted and executed one by one.

EXAMPLE FIVE – Proc EDITOR

In any industry that maintains healthcare data for a large number of clients, keeping that client data current is imperative. In the example below, a client has changed its name from National Bank to New Bank. The code below makes this change to the correct field in the dataset in place, without creating a new dataset.

```
Libname r 'r:\bokerson\sesug.2007';
Proc EDITOR data=r.clientemployergroup;
Find 1, last employerdesc='National Bank';
Replace employerdesc='New Bank';
Run;
```

In this example, the *find* command searches a range (1 to last) for occurrences of National Bank in the clientemployergroup dataset and renames these occurrences to New Bank.

PROCEDURE SIX – PROC DELETE

The DELETE procedure was designed to delete SAS data sets from the SAS library where they are stored. This allows the space to be freed immediately for new data sets. When used for deleting datasets from the work directory during processing as datasets are no longer needed, less code is required than with proc datasets. The caution is that running the following code with no options will delete the entire contents of your work directory, which is usually NOT what is wanted.

```
Proc DELETE;
Run;
```

The DELETE procedure does permit use of the `_all_` (all datasets in a library) and `_last_` (last data set created in a library) second-level dataset names.

EXAMPLE SIX – Proc DELETE

In the process of data modeling, often a large number of temporary datasets are created or used during early stages of the process that are no longer needed after calculations are complete. These datasets take up space and can cause processing to slow down; removing them when no longer needed contributes to the efficiency of the process.

The following example shows equivalent Proc DELETE and Proc DATASETS code for deleting two temporary work datasets called test1 and test2. While the two sets of code perform equivalent functions, the Proc DELETE code can be more efficient, averaging less CPU time usage.

```
Proc DELETE data=work.test1 work.test2;
Run;
```

is equivalent to:

```
Proc DATASETS library=work nolist;
Delete test1 test2;
Run;
```

Partial Log follows:

```
146 Proc DATASETS library=work nolist;
      147 Delete test1 test2; Run;

NOTE: Deleting WORK.TEST1 (memtype=DATA).
NOTE: Deleting WORK.TEST2 (memtype=DATA).

NOTE: PROCEDURE DATASETS used (Total process time):
      real time          0.09 seconds
      cpu time           0.09 seconds

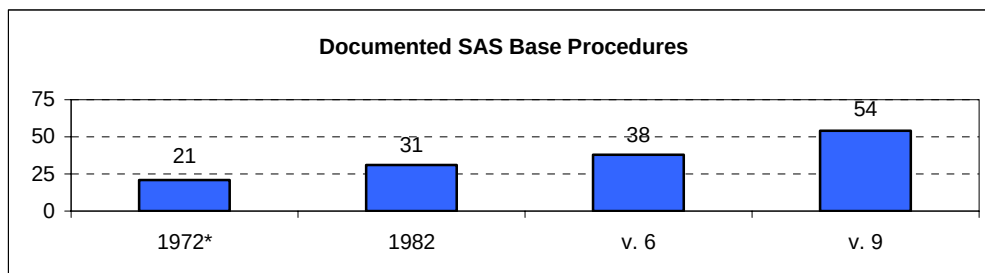
192 Proc DELETE data=work.test1 work.test2; Run;

NOTE: Deleting WORK.TEST1 (memtype=DATA).
NOTE: Deleting WORK.TEST2 (memtype=DATA).
NOTE: PROCEDURE DELETE used (Total process time):
      real time          0.07 seconds
      cpu time           0.07 seconds
```

NOTE: While not documented since version 5, Proc DELETE seems to be alive and well at SAS Institute, Inc. Current sample code on the SAS support web site still includes Proc DELETE statements. For example, see <http://support.sas.com/ctx/samples/index.jsp?sid=449>.

CONCLUSION

For the most part, software is not like a good wine. It does not get better with age. Users constantly demand enhancements to the functionality of the software itself as well as its system operability. For developers, it is often more efficient to start from scratch, i.e. write a new procedure in SAS, then to modify the original. It is difficult then to justify providing support for both a new enhanced procedure that has all the functionality of the old and much more and for the old, original procedure. But, sometimes, the old is exactly what is needed. That is true in the SAS examples above.



* statistical procedures included in base.

REFERENCES

Jaffe, Jay. *Mastering the SAS System*. New York: Van Nostrand Reinhold, 1989.

SAS Institute, Inc. *SAS Users' Guide: Basics, 1982 Edition*. SAS Institute Inc, Cary, NC, 1982.

SAS-L archives. <http://www.listserv.uga.edu/archives/sas-l.html>.

Service, Jolayne. *A User's Guide to the Statistical Analysis System*. North Carolina State University, 1972.

ACKNOWLEDGEMENTS

I would like to acknowledge the members of the HMC analytic group for their suggestions and assistance in the development of this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. For more information contact:

Barbara B. Okerson, Ph.D., CPHQ
Senior Health Information Consultant, Analytics
Health Management Corporation
8831 Park Central Drive, Suite 100
Richmond, VA 23227
Office: 804-678-2028
Fax: 804-354-2468
Email: bokerson@choosehmc.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.