Paper SF145-2019

Old But Not Obsolete: Undocumented Procedures for SAS® University Edition

Barbara B. Okerson, Independent Contractor

ABSTRACT

Proc Spell? Proc Neighbor? Proc Browse? These and other 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. These procedures have remained available in standard SAS installations. But did you know they are also available for the University Edition? Since sometimes, simpler can be better, this paper will use the SAS University Edition to look at these and other available SAS procedures not in the current documentation and address situations where they can be useful today.

Examples were developed with the SAS University Edition. Examples are not version or platform specific and can be adapted by all levels of SAS users.

ABOUT THE SAS UNIVERSITY EDITION

While the product name is SAS University Edition, this SAS offering is not just for those at institutions of higher learning; any independent learner or researcher can download this product. The SAS University Edition includes Base SAS® (including SAS graphics), SAS/STAT®, SAS/IML®, SAS/ACCESS® Interface to PC Files and SAS® Studio and is available for your PC, Mac, or Linux workstation. It was originally designed for teaching and learning statistics and quantitative methods and for primary use in such areas as economics, psychology and other social sciences, computer science, business, medical/health and engineering. Available graphics in SAS University Edition are created using the statistical graphic (SG) procedures and the SAS Output Delivery System (ODS).

SAS University Edition can be run either as a virtual application on your computer running VirtualBox or VMWare Player or you can run it directly from the Amazon Web Services (AWS) MarketPlace. The difference between the two methods is that if you download the vApp, you must also install and run virtualization software. If you run SAS University Edition on Amazon Web Services Marketplace, you need a browser and an Internet connection. You can download SAS University Edition at: https://support.sas.com/en/software/university-edition.html. Set up instructions are also at this link. Instructions for running on AWS MarketPlace can be found at; https://support.sas.com/software/products/university-edition/faq/AWS_runvApp.htm.

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 such as Proc MATRIX. To maintain compatibility for code and applications developed using these procedures, even most of these procedures still work, although no longer documented. An example of an obsolete procedure in this category would be Proc NICKNAME. Other procedures have been swallowed by new procedures. These procedures include Proc FUNCAT replaced by Proc CATMOD and Proc STEPWISE by Proc REG. While the old procedures can still be run, there is often no advantage in doing so. But there are some major reasons why you still might want to use the old procedures:

- You have inherited working code that uses them.
- You only want the output provided by the undocumented procedure. I
- You want to save time and machine resources.

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. While undocumented SAS procedures not covered here 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.

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="/folders/myfolders/SESUG/condition.txt"
create dict=work.mycatalog.spell; Run;
```

A text file was created from a large database with five fields (condition1, condition2, condition3, condition4, condition 5) that should only contain the values in the condition.txt file. This file will be used to identify errors and error location in the large file. The first step is to create a location for the results using Proc PRINTTO.

```
Proc PRINTTO print="/folders/myfolders/SESUG/sp_cond.txt" new;
Run;
```

Then the file from the large database was run against the word list to identify any spelling errors in the file. Proc PRINTTO diverted the output to an external file. Default output writes to the output window. See code below.

```
Proc SPELL wordlist="/folders/myfolders/SESUG/spellcase.txt" verify
dict=work.mycatalog.spell;
Run;

Proc PRINTTO print=print;
Run;
```

The output from the file /folders/myfolders/SESUG/sp_cond.txt is below:

```
File: "/folders/myfolders/SESUG/spellcase.txt"

Unrecognized word    Freq  Line(s)

COND1             5   1 (5)
DIAB              4   14, 32, 39, 234
CHHF              1   18
COPD              5   39, 234, 235, 236, 241
HF                1   349
```

Display 1. Proc SPELL Output.

PROCEDURE TWO – PROC BROWSE

The browse procedure was created to allow users to read a SAS dataset both interactively and in batch mode. This procedure does not allow modification of the SAS dataset.

Statements that can be used with Proc BROWSE are:

- FORMAT – assign output formats.
- INFORMAT – assign informats for the duration of the procedure.
- 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.
- TOP – moves to first observation for multiple commands.
- BOTTOM – moves to last observation.
- UP – move pointer up a designated number of observations.
- DOWN – move pointer down a designated number of observations.

The NAME and LOCATE statements work together to name a variable and a value to search a SAS dataset 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.

In this example, Proc BROWSE is used in to check a dataset for the existence of particular member claim. The advantage of Proc Browse for this is that it saves time. With Proc BROWSE, the dataset with many million records returns a response in less than 3 seconds (real and cpu). The results are returned in the log. See code and results below:

```
libname SESUG '/folders/myfolders/SESUG/'; run;
Proc BROWSE data=SESUG.interim8;
FIND all 1,last bene_clm_num='540620968A';
Run;
```

The results are displayed in the log below as follows (bolding is mine):

```
1     OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
72
73     libname SESUG '/folders/myfolders/SESUG/';
NOTE: Libref SESUG was successfully assigned as follows:
Engine:    V9
Physical Name: /folders/myfolders/SESUG
73    !                       run;
74     Proc BROWSE data=SESUG.interim8;

NOTE: Welcome to the BROWSE Procedure. Begin entering commands.
75     FIND all 1,last bene_clm_num='540620968A';
NOTE: Found at OBS 161007.
NOTE: End of search. OBS=161892.
76     Run;
```

Display 2. Proc BROWSE Output.

PROCEDURE THREE – PROC EDITOR

Proc EDITOR is a procedure that allows users to make changes to a SAS dataset without running a data step. However, a new dataset is NOT created with Proc EDITOR. Changes made by Proc EDITOR operate directly on the existing SAS dataset 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:

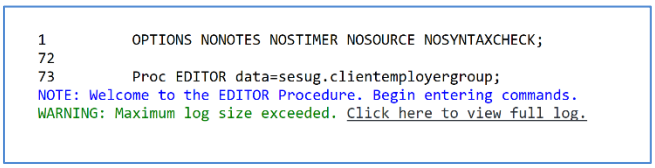
- REPLACE – changes one or more values of specified variables in a range of observations.
- ADD – adds a new single observation to the end of the dataset.
- DELETE – sets all the variables in a specified range of observations to missing.
- DUP – duplicates observations for a specified range in the dataset.

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.

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 WellPoint to Anthem. The code below makes this change to the correct field in the dataset in place, without creating a new dataset and without reading the dataset into memory:

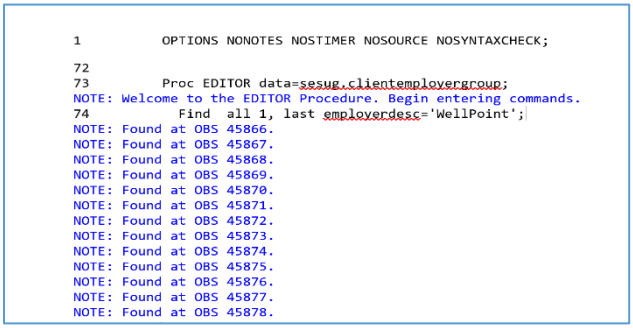
```
Proc EDITOR data=sesug.clientemployergroup;
 Find all 1, last employerdesc='WellPoint';
 Replace employerdesc='Anthem';
Run;
```

In this example, the *find* command searches a range (1 to last) for occurrences of WellPoint in the clientemployergroup dataset and renames these occurrences to Anthem. The all option is necessary. Even with a range provided, without *all,* the procedure will stop after making the first rename.
Here is the log from the procedure:



Display 3. Proc EDITOR log.

Clicking on the link displays the full log, which is a listing of the observations where changes were made in the dataset. See below the first part of this log (thousands of observations were actually changed):



Display 4. Partial Proc EDITOR Expanded Log.

If you have a very large dataset with a small number of variables, Proc Editor can be a very efficient method for adding a single observation since it adds the row in place without reading dataset into memory. Unfortunately, multiple observations need to be entered one at a time. Syntax is as follows:

```
Proc Editor data=SASdatasetname;
Add variable1=value1 variable2=value2 variable3=value3 …;
Run;
```

Proc EDITOR should be used with care as changes made are permanent. Without documentation or help files to fall back on, the likelihood of fatal errors is increased.

PROCEDURE FOUR – PROC RSQUARE

The RSQUARE procedure was designed to perform all possible regressions for one or more dependent variables and selected independent variables. By default the output is the R-squared value. Mallows C statistic can also be computed. The only statements available for the original RSQUARE procedure are MODEL and BY. However, today the procedure also takes statements such as FREQ that are available for Proc REG and other statistical procedures. Proc REG replaced this procedure. The code to run Proc RSQUARE vs Proc REG is very similar as in the example below (data generated for example). But the compute time and output are very different.

Proc RQUARE and Proc REG comparison code:

```
Proc RSQUARE;
 Model b=a;
 Freq c;
run;
```

```
Proc REG RSQUARE;
 Model b=a;
 Freq c;
run;
```

The RSQUARE Procedure
Model: MODEL1
Dependent Variable: b

R-Square Selection Method

Number of Observations Read	5
Number of Observations Used	5
Sum of Frequencies Read	23
Sum of Frequencies Used	23

Frequency: c

Number in Model	R-Square	Variables in Model
1	0.7518	a

Display 5: Proc RSQUARE Output.

As illustrated below the REG procedure includes additional statistics, parameter estimates, fit diagnostics and residuals graphics in addition to the R-squared value. Run time for Proc RSQUARE was .39 seconds as opposed to 15 seconds for Proc REG with the RSQUARE option.

The REG Procedure
Model: MODEL1
Dependent Variable: b

Number of Observations Read	5
Number of Observations Used	5
Sum of Frequencies Read	23
Sum of Frequencies Used	23

Frequency: c

Analysis of Variance

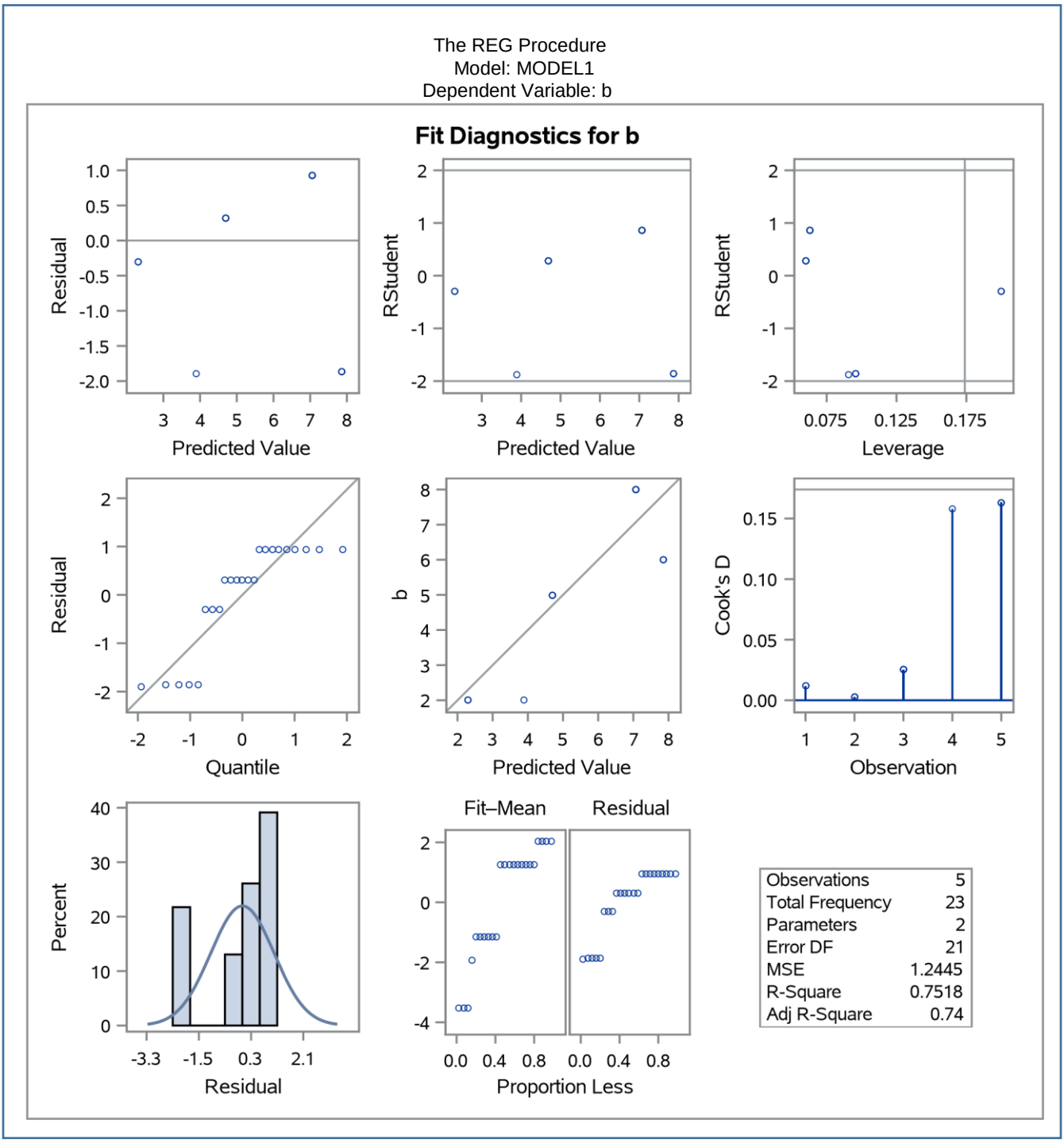
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	79.17009	79.17009	63.62	<.0001
Error	21	26.13426	1.24449		
Corrected Total	22	105.30435			

Root MSE	1.11557	R-Square	0.7518
Dependent Mean	5.82609	Adj R-Sq	0.7400
Coeff Var	19.14778		

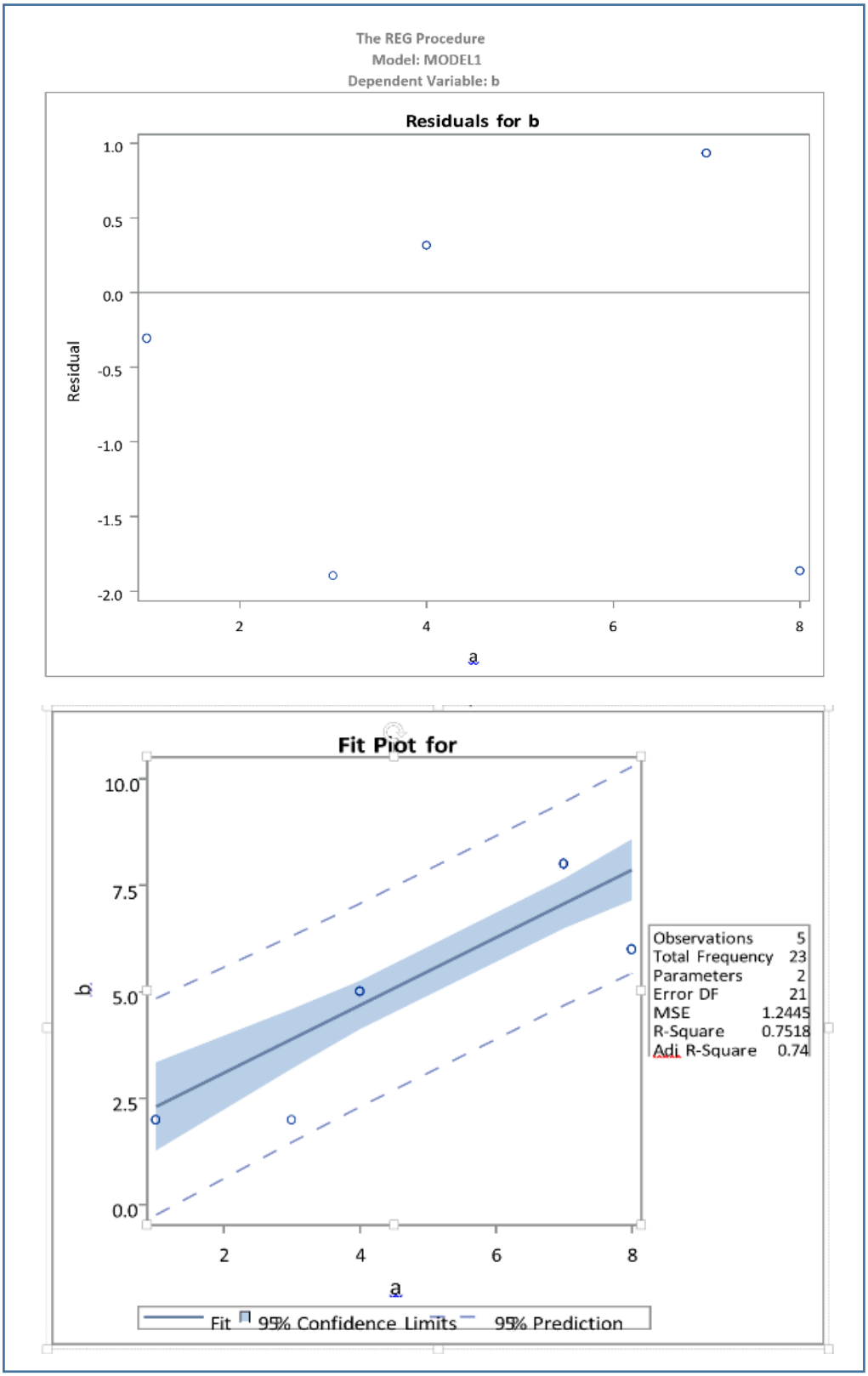
Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1.51211	0.58877	2.57	0.0179
a	1	0.79377	0.09952	7.98	<.0001

Display 6: Proc REG with RSQUARE Option First Page Output.



Display 7: Proc REG with RSQUARE Option Fit Diagnostics Output.



Display 8: Proc REG with RSQUARE Option Residuals and Fit Plot Output.

PROCEDURE FIVE – 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=NPAR 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 dataset, 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 dataset 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:

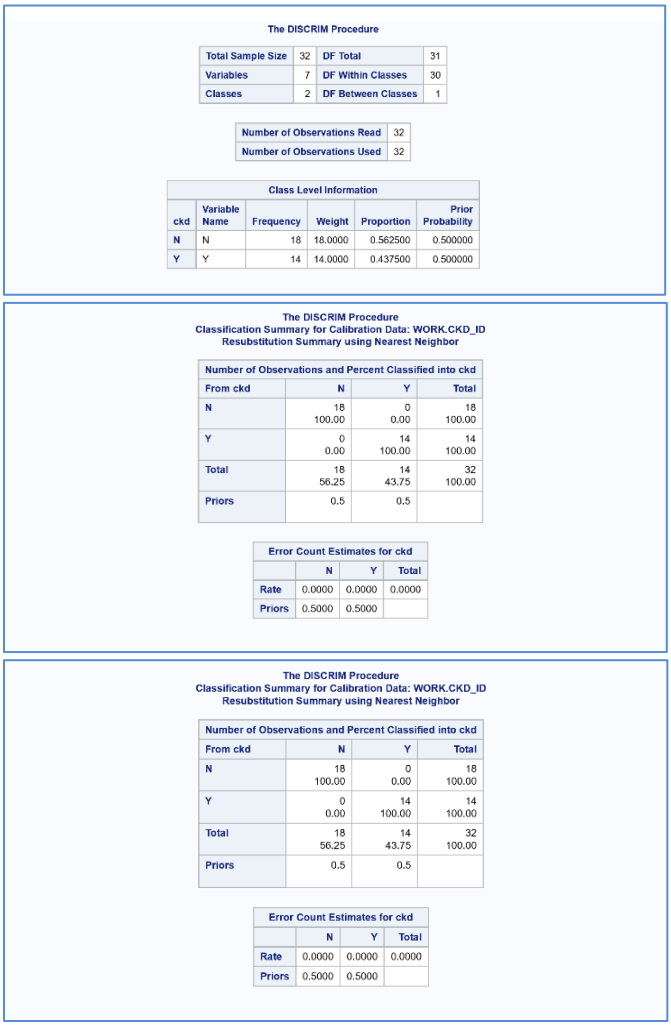
- IDENTITY – specifies use of Euclidean distances.
- K – specifies a k-value for k-nearest neighbor rule.
- LIST – prints classification results for each observation.
- LISTERR – prints only misclassified observations.
- THRESHOLD – specifies minimum possible posterior probability for classification.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

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.

```
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.

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



The DISCRIM Procedure

Total Sample Size	32	DF Total	31
Variables	7	DF Within Classes	30
Classes	2	DF Between Classes	1

Number of Observations Read	32
Number of Observations Used	32

Class Level Information

ckd	Variable Name	Frequency	Weight	Proportion	Prior Probability
N	N	18	18.0000	0.562500	0.500000
Y	Y	14	14.0000	0.437500	0.500000

The DISCRIM Procedure
Classification Summary for Calibration Data: WORK.CKD_ID
Resubstitution Summary using Nearest Neighbor

Number of Observations and Percent Classified into ckd

From ckd	N	Y	Total
N	18 100.00	0 0.00	18 100.00
Y	0 0.00	14 100.00	14 100.00
Total	18 56.25	14 43.75	32 100.00
Priors	0.5	0.5	

Error Count Estimates for ckd

	N	Y	Total
Rate	0.0000	0.0000	0.0000
Priors	0.5000	0.5000	

The DISCRIM Procedure
Classification Summary for Calibration Data: WORK.CKD_ID
Resubstitution Summary using Nearest Neighbor

Number of Observations and Percent Classified into ckd

From ckd	N	Y	Total
N	18 100.00	0 0.00	18 100.00
Y	0 0.00	14 100.00	14 100.00
Total	18 56.25	14 43.75	32 100.00
Priors	0.5	0.5	

Error Count Estimates for ckd

	N	Y	Total
Rate	0.0000	0.0000	0.0000
Priors	0.5000	0.5000	

Display 9. Proc Neighbor Output.

When running Proc NEIGHBOR in 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 RSQUARE and NEIGHBOR are the only undocumented SAS statistical procedures 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 STEPWISE – provides five methods for stepwise regression.

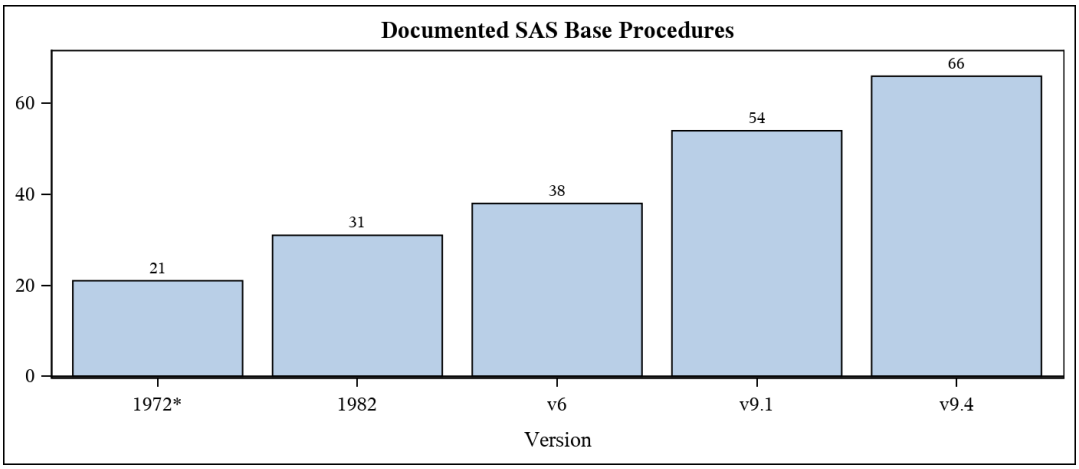
Other former statistical procedures are no longer available. An example in this category is Proc MATRIX which disappeared altogether with the advent of SAS/IML.

A NOTE ABOUT PROC DELETE

Proc DELETE was removed from SAS documentation after version 5 as SAS expected the functionality to be taken up by Proc DATASETS. Users stubbornly continued to use Proc DELETE – less code and more efficient. Proc DELETE does not need an in-memory directory to delete a dataset. SAS has now returned Proc DELETE to the version 9.4 and forward procedure documentation.

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, e.g. write a new procedure in SAS, than 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 SAS.

REFERENCES

Okerson, Barbara. 2007. “Old But Not Obsolete: Undocumented SAS Procedures.” *Proceedings of the 2007 SouthEast SAS User Group Conference.* Hilton Head, SC.

SAS Institute, Inc. https://support.sas.com/en/documentation.html. Accessed August 8, 2019.

SAS Institute, Inc. https://support.sas.com/documentation/onlinedoc/stat/indexproc.html. Accessed August 16, 2019.

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

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

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

CONTACT INFORMATION

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

Barbara B. Okerson, PhD, CPHQ, FAHM
Independent Contractor
Email: bokerson1@charter.net