One Trainer's Method of Workshop Design

Teresa Wilson, Clemson University, SC

Keywords: PC SAS® training, workshops, workbooks

Abstract: The freshwater fisheries biologists in the South Carolina Department of Natural Resources desired to learn programming SAS® software in order to complete some of their basic data analyses independently. The author designed a series of workshops involving a hands-on programming experience for the biologists, as well as workbooks with examples and exercises relevant to fisheries research and management.

Introduction

One of the goals of the freshwater fisheries biologists in the South Carolina Department of Natural Resources was to obtain a working knowledge of SAS programming in the personal computer environment (Wilson and Foltz 1997). I had worked with the biologists for over a year as SAS programming consultant, so I knew the programming capabilities of the participants ranged from none to intermediate. Though most of the biologists programming skills were very limited, all knew how to negotiate and use SAS commands and execute programs written by others.

After talking with the biologists, we decided that I would design the workshops in-house using examples relevant to fisheries research and management that the biologists would encounter in their jobs. Since we remember 10% of what we hear, 40% from the way it’s said, 50% from what we see and 90% from what we do (Nelson and Howard 1997), hands on programming was chosen as the method for presenting the information over a two to three day period.

The biologists had indicated difficulty in reading/interpreting SAS published documentation, therefore I felt it was worth the time and effort to create a specialized workbook they could use as a reference.

Three of the workshops used SAS System 6.04. The last workshop used SAS System 6.08.

Workshop Design

For the first workshop, I surveyed and talked with the biologists to determine what they wanted to learn. All agreed that to be able to read in data from dBASE and text files and to perform simple descriptive statistics would be the first step. I then proceeded to collect small data files in dBASE and ASCII that would typically be used by the biologists. From these I created exercises to demonstrate a “building block” concept of SAS. Based on these exercises, I designed the workbook with this “building block” concept as far as sequencing information. The examples created to demonstrate the SAS statements used examples familiar to the biologists. Sample programs were included in the work book as examples in utilizing multiple concepts (i.e., taking data from PROC MEANS and utilizing in another data step). Each workshop was limited to one biologist per computer, with approximately 30 minutes of lecture and 45-60 minutes for hands-on programming.

Subsequent workshops were similarly designed. The second workshop was based on the biologists’ desires to know how to use and interpret three of the linear statistical procedures. The third workshop concentrated on introducing additional options and statements in the data step and select procedures, and introducing two additional procedures. By the time of the fourth workshop, version 6.08 was available, so this workshop highlighted what was new in SAS for Windows.
Workbook Design

The Basics of SAS

The first workshop (Figure 1a) was developed for the least experienced users. To avoid confusion and information overload, I selected statements and options that would most likely be used in any analysis the biologists might perform. The introduction described DOS path structure and filename structures, the SAS display manager and terminology used in SAS documentation. The workbook was then arranged into three chapters:

Data, where does it come from? Described the different ways of storing data, how to read ASCII files, dBASE files and spreadsheets (Figure 2) and using data in the SAS program itself. dBASE had been used for several years for data base management and was the data entry application of choice. Temporary vs. permanent SAS data sets were also described in this section. Exercises were included after the ASCII discussion and the dBASE discussion.

Data, how to manipulate it. This chapter listed the arithmetic, comparison and logical operators and the rules of using these operators. It also described how to create new variables, both numerical and character and four numeric SAS functions the biologists would likely use: log10, abs, sqrt and int. After describing informat and formats, a series of exercises were included to re-enforce these concepts. There was also a discussion on how SAS handles missing values.

Data, how to use descriptive statistical procedures. This chapter began by describing the FREQ procedure and the TABLES, WEIGHT and BY statements and some options available in these statements. Two concepts that were stressed were how to use previously created SAS data sets with the data= options and how to output the results with the out= option. Two exercises followed this discussion on PROC FREQ.

The section on the MEANS procedure listed the statistical keywords, discussed the options, data=, maxdec and noprnt, in the PROC MEANS statement, discussed the VAR, BY, CLASS, OUTPUT and ID statements. Two exercises were provided after this section.

The biologists were introduced to UNIVARATE procedure as an alternative to PROC MEANS since additional statistics (e.g., quartiles, mode, median) would be available. The options data=, noprnt, plot and freq used in the PROC statement were discussed, as were VAR, BY, and ID statements. The OUTPUT statement listed select statistical keywords that would be typically used. Example programs were included as well as an exercise.

Statistics Useful to Fisheries

The second workshop (Figure 1b) was based on the REG, ANOVA and GLM procedures. Again, to avoid information overload and confusion, I selected statements and options most likely to be used. It was explicitly stated that this was not a workshop in statistics, but the biologists should be able to write a simple program based on information given by a statistician.

The REG procedure (Figure 3) looked at the data=, noprnt, simple options in the proc statement and described what statistical estimates outest= stored in a SAS data set. We looked at how to use these statistics from outest= in another data step to perform calculations using the INTERCEPT and regression coefficients. The section on MODEL statements described how to set up a model and some options (p, r, clm, cli) available. The model selection option was also included. It was explained how and when to use a FREQ statement. Also included the OUTPUT statement and options available based on input in MODEL statement. An example program (Figure 3) included demonstrating using the outest= statistics and using regression coefficients and how to use these to determine the weight of fish based on the log10 of length. The SAS output was included as well as interpretation of the output. Another example was included demonstrating stepwise regression, with SAS code, output and notes on interpreting the output. One exercise on regression was used to reinforce the information from this chapter.
PROC ANOVA - To begin this chapter, the following generic statements were included in the workbook:

```sas
proc anova options;
class variable(s);
model dependent(s) = effect(s);
means effect(s) / options;
by variable(s);
quit;
```

The introduction described dependent vs independent variables, that ANOVA tested for differences between means for classification variables; how to write MODEL statements and identifying main effects vs interactions; using the MEANS statement. To avoid misuse of means separation tests, I recommended the biologists contact a statistician to determine which test would be appropriate for their analysis. An example typical of fisheries was included along with output and explanation of output. One exercise was included.

PROC GLM - the generic statements included were:

```sas
proc glm options;
class variable(s);
model dependent(s) = independent(s);
output out=SASdataset keyword=name(s);
means effect(s) / options;
by variable(s);
quit;
```

This chapter described the two types of independent variables, classification and continuous, as well as dependent variables. The types of models were introduced (e.g., polynomial, multivariate analysis). Options included in the PROC statement included data=, outstat, and NOPRINT. The following selective options were included with the OUTPUT discussion: out=, p=, r=, 195m, u95m, 195, u95, stdp, stdr, and stdi. The options included in LSMEANS were pdiff and stderr. Options included in the MEANS statement were the most commonly used: duncan, t, line. There was one example and three exercises for PROC GLM.

More Basics of SAS

The third workshop (Figure 1c) included functions that would be useful. I presented these as a problem that may occur with the data (Figure 4) or as something that may need to be done with the data. Character functions that would be useful for “cleaning” up data (e.g. left, right, upcase) and for joining two (or more) values (substr, trim).

For numeric functions, we looked at the difference between round and int. And with date/time functions, we looked at how to extract values (e.g. month, year) and combine values (mdy). Also included were the special functions lag, put, input, uniform and ranuni. An exercise was included to use some of these functions.

We looked at using dates and/or times in IF/THEN statements, the importance of using IF/THEN/ELSE statements, and using IF/IN statements.

New data statements introduced were ATTRIB, RETAIN, OUTPUT (in the data step and in DO groups), and simple and iterative DO groups, DO/WHILE, DO/UNTIL, DO/OVER and ARRAY statements. Examples that would be useful to the biologists (such as generating random numbers) were used to illustrate the use of each of these statements.

As anticipated, the discussion on DOs and ARRAYS was confusing for the biologists. For the exercise in using DOs and ARRAYS, I had the biologists type an example program (Figure 5) into the program editor and submit the program. This way they could see how the data was entered and how to use SAS code to restructure the data to be used for further analysis.

I had started using the %LET macro statement in some of the programs I was writing for the biologists and decided to introduce the terms “macro” and “global variable” and their usefulness in this workshop.

Although the WHERE statement can be used in a data step, I limited the discussion to using it in procedures. Many times a biologist wanted to print out a select species or lake from a permanent SAS data set and this allowed them to do that without re-reading the data and doing a proc print.
Two new procedures were introduced: PROC FORMAT and PROC TABULATE. Additional statements in PROC PRINT (SUM and PAGEBY) were included. In PROC FORMAT we looked at the VALUE statement to assign a value to an input value (e.g., where lmb in the data would be spelled out as largemouth bass) and to create ranges.

In PROC TABULATE, we used the following:

```sas
proc tabulate options;
  class variable(s);
  freq variable;
  format variable(s) format. ...;
  label variable = 'label' ...;
  by variable(s);
  table [expression,] [expression,] expression /options;
  keylabel keyword = 'text' ...;
```

Options used in the PROC statement were data=, format= and order=. Options looked at in the TABLE statement included misstext=, printmiss= and condense. The statistics most likely to be used were also listed. Because of the numerous permutations of tables, I gave an example of each along with the TABLE statement for ease in reference.

The final chapter described how to create output from a SAS data set using DATA _NULL_ and also how to create ASCII files from SAS data sets. The final exercise allowed the biologists to write SAS code using %LET, PROC FORMAT, and create a special formatted output using DATA _NULL_.

SAS for Windows

The fourth and final workbook in this series combined the three previous workbooks into one, documented new options and items in version 6.08 and items that were no longer available. Two additional procedures were included: PROC PLOT and PROC CHART. Since PROC DBF was no longer available in version 6.08, an appendix was included with instructions on how to export ASCII files from applications being used by the biologists (e.g., dBASE, LOTUS123, WordPerfect and Paradox).

Conclusion

After each workshop I handed out a survey to determine if the workshop meet the expectations/goals of the biologists. They all agreed that the hands-on method helped them learn the material, prepared them to write SAS code as well as a better understanding of the SAS programs I had written for them.

References


Acknowledgments

The author would like to thank South Carolina Department of Natural Resources for continuing financial support for this project.

Trademark Information

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicated USA registration.

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

Author contact

Teresa Wilson
Clemson University
Dept. of Aquaculture, Fisheries & Wildlife
G08 Lehotksy Hall
Clemson, SC 29634-0362
864.656.3063
twlsn@clemson.edu
### Basics of SAS

**Table of Contents**

I. Introduction
II. Data - where does it come from and how does one access it?
   - Introduction
   - SAS data sets
   - dBASE files
   - ASCII files
   - DIF files
III. Converting SAS data sets to LOTUS123 worksheets
   - Converting LOTUS123 worksheets to SAS data sets
IV. Data - how to manipulate it
   - Introduction
   - SAS Operators
   - Creating new variables
   - Informats and formats
   - Missing values
   - SAS statements
   - Proc Sort
   - Proc Print
V. Data - how to use the statistical procedure
   - Proc Freq
   - Proc Means
   - Proc Univariate

### Basic Statistics Useful in Fisheries

**Table of Contents**

I. Introduction
II. Proc Reg
III. Proc Anova
IV. Proc GLM

### More Basics of SAS

**Table of Contents**

I. Introduction
II. Data Manipulation & SAS Functions
   - Character functions
   - Numeric functions
   - Date function
   - Special functions
III. Data Statements
   - IF/THEN
   - ATTRIB
   - OUTPUT
   - DOARRAY statements
IV. %LET macro statement
V. Procedures
   - WHERE statement
   - Proc Print
   - Proc Format
   - Proc Tabulate
VI. Generating Reports with DATA _NULL_

---

**Figure 1a**

**Figure 1b**

**Figure 1c**
II. DATA - where does it come from and how does one access it?

Introduction

The first thing to look at is the data. What form is it in? What are the parameters, what modulate it in, etc. The data we will be working with today are in four forms, dBASE, ASCII, DBF, and SAS data sets, and are found on the floppy distributed earlier.

In this documentation, the form of SAS statements are specified with these conventions:
- buildup indicates to use the same spelling and form as shown.
- italic means to supply your own information.

The FILENAME statement (S3A.L2:367) will be used frequently throughout this document. Unless a file is in your current directory, you will have to use the FILENAME statement. It also makes it convenient to identify all external input and output data files at the beginning of a program even though the FILENAME statement can be used anywhere. The following is a generic FILENAME statement and examples:

```
FILENAME 'drive' \f ilename 'ext';
FILENAME 'drive' \filename 'ext';
FILENAME 'drive' \l iename 'ext';
FILENAME 'drive' \f ilename 'ext';
```

The filename is any eight character name you want to give the file, think of it as a nickname. Then in single quotes specify the drive (A, B, or C) and then the name, you want to include the directory name if the file is stored in a directory. If the data file is in your current directory (that is, the directory where you are working with SAS), you only need to use the filename and extension in single quotes.

The DATA statement is used to tell the SAS system you want to create a SAS data set. The DATA statement signals the beginning of a DATA step and the same with the DATA statement identifies the SAS data set. The SAS statements within the DATA step creates the SAS data set. For example:

```
data test;
```

```
data test; run;
```

SAS data sets

Data in SAS data sets are in a special form that only SAS can read. SAS data sets can be temporary or permanent. Temporary files last for the duration of the SAS session and have a one level name as found in:

```
data temporary;
```

```
data permanent;
```

Permanent files are stored on disks and have an .SDS extension. In order to access permanent SAS data sets you have to know where they are and reference their location by using the LSNAME statement in the SAS program (S3A.L2:367).

The LSNAME statement uses what SAS calls a libref (library reference) so the .SDS file on the disk can be accessed and indentication of the drive where the .SDS files are located. The libref then proceeds any filename without the .SDS extension (i.e.,)

The following are examples that you familiar with:

```
libname creel ' ';
libname creel 'CREEL';
```

SAS data sets

The first LSNAME statement indicates creel is the libref and the SAS data sets are in your current directory. The second one indicates the SAS data sets are on drive A: and the DATA statement is creating a SAS data set called sample that will be stored on the floppy in the A drive. (NOTE: on the floppy the file will be called sample.dat.) You could also think of the libref as a nickname. The LSNAME statement should be at the beginning of your SAS program.

REMEMBER: the SAS data sets on disk have an .SDS extension. The SAS program does not know to access the files. By using the LSNAME statement, the SAS program knows where to go.

Exercises: dBASE files

Before beginning the exercises, I want to discuss the SAS statements RUN; and PROC PRINT. The RUN; statement is used to tell SAS to execute the preceding statements, data steps or procedure (PROCs). The basic PROC PRINT; procedure will be used throughout these exercises, so you can see what is happening to the data. PROC PRINT will be discussed in detail in Chapter 3.

REMEMBER: For all of these exercises, all input files are on the floppy and all output files are to be stored on the floppy. THIS IS DRIVE B: ON THE ZENITH.

1. In the space provided below write out the statements necessary to convert the dBASE file JOBS.BAS to a permanent SAS data set. Include the statement(s) necessary to print the file.

2. Now enter these statements onto the SAS Program Editor. Now submit the program (F10). When your program runs successfully, save it onto the floppy and call it PGM_1.SAS.

3. Change the appropriate statements to convert the dBASE file in Exercise 1 to a permanent SAS data set and print out the file. The SAS data set should be stored on the floppy. Now submit the program. When your program runs successfully, save it onto the floppy and call it PGM_2.SAS.
Figure 3. From: Statistics Useful in Fisheries

CHAPTER 3
REGRESSION

PROC REG is a general purpose procedure for simple regression (SGS:266-
268). There are several other regression procedures that have more specialized
applications, which we will discuss later. PROC REG fits linear square
estimates to linear regression models. There are various model selection
methods available as well. (Note: transformations of dependent and
independent variables can produce non-linear analyses, e.g., logistic curves, etc.
This will be covered in an advanced workshop in the future).

All dependent and independent (regressed) variables used in PROC REG are
continuous variables, e.g., length, weight, pH, temperature, etc.
The following statements are used with the REG procedure:

ods reg optotypes; model dependent = regressor / option(s);
ods out=dataset (keyword=variable(s)); quit;

PROC REG statement. The following options are available in the PROC REG
statement:

- BY -- requests the SAS data set be used. If the data set is not
specified, REG uses the most recently created SAS data set.
- Emissary +dataset -- requests that parameter estimates and optional
statistics be output to this data set (see the printout at the bottom of
page 11). This can be a permanent or temporary data set. The variables found in this
output data set are as follows:
  - the BY variables, if any
  - XMODEL, a character variable containing the label of the corresponding
    MODELS statement, if any was specified
  - _TYPE_, a character variable with the WORD PARM for every
    observation
  - _RESP_, the name of the dependent variable
  - _RMSE, the root mean squared error or the estimate of the standard
    deviation of the error term
  - _INTERCEPT, the estimated intercept
  - all the variables listed in any MODELS statement containing the
    estimated regression coefficients for the model. A variable that does not appear in the model corresponding to a given observation has a missing
    value in that observation.

- FREQ -- suppresses the normal output.
- HTML -- prints the "simple" descriptive statistics for each variable used
  in REG.
- MODEL statement. At least one MODEL statement is required to identify the
  dependent and independent variables in the regression model. The dependent
(responses) variables are specified immediately after the word MODEL, followed
by an equal sign and the independent (regressor) variables. The options listed
here are only a small portion of those found in SGS:278-281; however, these are
the ones you would most probably use:
  - X -- suppresses the normal printout or regression results. You do not
    need this if you used the NOPRINT option in the PROC REG statement.
  - P -- calculates predicted values from the input data and the estimated
    model. The printout includes the observation number, the first ID variable, if
    specified, the actual and predicted values, and the residuals.
  - R -- requests that the residual be analyzed. The printed output includes
    everything requested by the p option plus the standard errors of the
    predicted and residual values, the studentized residual, and Cook's D
    statistic.
  - CI -- prints the 95% upper- and lower-confidence limits for the expected
    value of the dependent variable (error) for each observation. This is not a
    prediction interval (see the ci1 option) because it takes into account only
    the variation in the parameter estimates, not the variation in the error term.
  - ci1 -- requests the 95% upper- and lower-confidence limits for an
    individual predicted value. The confidence limits reflect variation in the
    error, as well as variation in the parameter estimates.

Options for model selection:
- selection -- specifies the model-selection method as summarized below:
  forward (FD) -- This starts with no variables in the model. Each
  independent variable that is added to the model and a F-statistic is computed. If
  the variable's contribution to the model is significant (p < 0.10), then the
  variable remains in the model. This evaluation continues until no remaining
  variable produces a significant F-statistic. Once a variable is in the model it
  stays.
  backward (BD) -- This starts with all independent variables and delete
  them one by one until variables remaining in the model produce a significant
  F-statistic (p < 0.10). For each evaluation, the variable with the smallest
  contribution is deleted.
  stepwise -- If a variable is determined significant (p < 0.15), it is added to
  the model. The model is re-evaluated and any variables no longer significant
  (p > 0.15) are deleted. This method continues until variables
  outside the model are not significant and variables inside are. In this
  method, the "best" variable may be deleted without considering what adding
  the "best" remaining variable might accomplish.
  once -- Tries to find the best one variable model, the best 2-variable model,
  etc. by defining the best (maximum) R2. Unlike stepwise, once
  evaluates all switches before adding/eliminating variables.
  twice -- Works similar to once, except looks to produce the smallest R2.
  The effect of adding/eliminating variables is also evaluated.
  improve -- determines the R2 value for all possible combinations of the
  independent variables. The model with the largest R2 is always identified.
  The F-statistic significance level default can be changed with _stats_.
  _stats_ specifies the significance level for entry into forward or stepwise models, always
  specifies the significance level for a variable staying in backward or
  stepwise models.

As with all statistical models, the above options cannot identify "true"
models, but are useful exploratory tools for model building. However, coupled
with sound theory, feasible models can be produced. See SGS:272-274 for
details.

FREQ statement. If a variable in the data set represents the frequency of
occurrences for the other values in the observation, include the variable's name in a
FREQ statement. PROC REG will treat the data set as if each observation
appears n times, where n is the value of the FREQ variable for the observation.

ID statement. The ID statement specifies one variable to identify observations
instead of observation numbers. The ID statement can only be used if the
MODELS statement contains options p, a, e, and/or c. By default (omitting the ID
statement), SAS uses observation numbers to identify the observations.

OUTPUT statement. The OUTPUT statement specifies an output data set
containing statistics calculated for each observation. One OUTPUT statement
can follow each model statement. For each statistic, specify the keyword,
the group name, and a variable name for the statistic in the output data set. The
OUTPUT data set contains all the variables in the input data set, including any
BY variables, and ID variables, and variables named in the OUTPUT statement
that contain statistics. Some of the statistics keywords and their meanings are:

- _predicted_ or _pred_ -- predicted values
- _residual_ or _err_ -- residuals calculated as actual minus predicted
- _lower bound_ of 95% confidence interval for the expected
  value(mean) of the dependent variable.
- _upper bound_ of 95% confidence interval for the expected
  value(mean) of the dependent variable.
- _LSE_ -- lower bound of a 95% confidence interval for an
  individual prediction.
  This includes the variance of the error, as well as the variance of the
  parameter estimates.
- _UCL_ -- upper bound of a 95% confidence interval for an
  individual prediction.
- _SDF_ -- standard error of the mean predicted value.
- _RES_ -- standard error of the residual.
- _RDP_ -- standard error of the individual predicted value.

BY statement. See discussion Chapter 1.

Missing values. Any missing observations are excluded from all estimates.
Use _quote_ instead of _raw_ to tell SAS to stop executing PROC REG.
Figure 3 (cont.) From: Statistics Useful in Fisheries.

Example 1:

```sas
libname fish 'a:';
proc reg outest=regr_est;
model log10_wt=log_tl;
by spe:ifical run;
run;
```

```sas
*---------------------------------------------------------------/
```
Figure 3 (con’t). From: Statistics Useful in Fisheries.

PRINTOUT of the example 1 SAS program:
REGRESSION FOR (LOG) LENGTH VS (LOG) WEIGHT OF FISH
The following will briefly describe the processing printed output. The bold words are the headings found on the printout.
1. The dependent variable is identified form (LOG WT).
2. Source of variation. Where the Error is the deviation of the dependent variable (LOG WT) from predicted values. Model reflects deviations between the predicted values and the mean.
3. DF is degrees of freedom associated with the source (i.e. the number of observations).
4. Sum of squares for each source (in %), i.e., the sum of squared deviations of each LOG WT from the mean.
5. Mean square (sample variance, at ) is calculated by:
   sum squares - DF
6. The F-value tests the hypothesis that all parameters are zero except the intercept. This is calculated by:
   mean square model - mean square error.
   The larger this value the better.
7. PROB > F is the significance probability. The is the probability of obtaining as F value as large or larger than the one obtained. If this value is less than .05 probability the regression did a “good job” of describing the data.
8. Root MSE (Mean Standard Error) is the standard deviation of the error term or square root of the error variance.
9. DEP MEAN is the mean of the dependent variable.
10. CV is the coefficient of variation which expresses the variation in non-zero values. CV is calculated by next mean - dep mean / dep mean * 100.
11. The R-square is the square of generalized correlation coefficient. This tells you the % of variation in the dependent variable that can be accounted for by variation in the independent variables and vice versa.

Figure 4. From: More Basics of SAS

CHARACTER FUNCTIONS

PROBLEM: you have a dBASE data set where the species code looks like this when printed out with SAS:

ael
ael
ael

SAS does not recognize them as the same because the second one has a blank at the beginning. You can go in and change the records in the dBASE file, but if there are many it would be more feasible to use the LEFT functions which will align character values to the left.

LEFT(variablename)

ex: spcode=left(spcode);

For those who use ASCII files as input data, this is not a problem unless you’ve assigned a column format for the variable. If the characters fall outside the specified columns they could be read as part of the next variable.

Figure 5. From: More Basics of SAS

/***********************************************/
/* This program restructures data */
/* for statistical analyses. */
/* Each wk is a measure of fish */
/* hemacritc. */
/***********************************************/

data one;
  input trtmt wk1 wk2 wk3 wk4 wk5;
array week (i) wk1-wk5;
do over weekn;
  week=i;
  hema=weekn;
  output;
end; *of DO;
drop wk1-wk5 i;
cards;
1 .25 .30 .50 .75 .80
1 .20 .30 .45 .60 .79
2 .26 .30 .40 .65 .70
2 .86 .45 .48 .68 .73
3 .15 .20 .30 .45 .50
3 .25 .45 .36 .45 .58
4 .55 .30 .50 .65 .60
4 .56 .39 .45 .67 .70;

proc print; run;