SAS® Tips, Tricks and Techniques

Kirk Paul Lafler, Software Intelligence Corporation

Abstract

The base-SAS® System offers users the power of a comprehensive DATA step programming language, an assortment of powerful PROCs, a macro language that extends the capabilities of the SAS System, and user-friendly interfaces including the SAS Display Manager. This presentation highlights numerous SAS tips, tricks and techniques using a collection of proven code examples related to effectively using the SAS Display Manager and its many features; process DATA step statements to handle subroutines and code libraries; deliver output in a variety of formats; construct reusable code; troubleshoot and debug code; and an assortment of other topics.

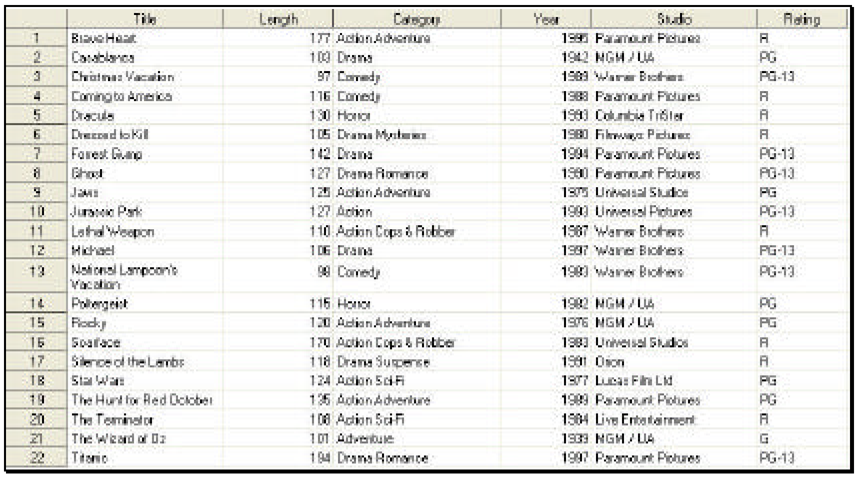
Introduction

This paper illustrates several tips, tricks and techniques related to the usage of the Base-SAS software. We will examine a variety of topics including SAS System options, DATA step programming techniques, logic conditions, output delivery and ODS, macro programming, and an assortment of other techniques.

Example Tables

The data used in all the examples in this paper consists of a selection of movies that I've viewed over the years. The Movies table consists of six columns: title, length, category, year, studio, and rating. Title, category, studio, and rating are defined as character columns with length and year being defined as numeric columns. The data stored in the Movies table is depicted below.

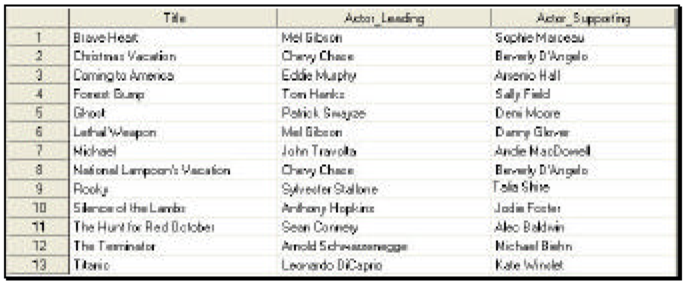
MOVIES Table



	Title	Length	Category	Year	Studio	Rating
1	Brave Heart	177	Action Adventure	1995	Paramount Pictures	R
2	Casablanca	103	Drama	1942	MGM / UA	PG
3	Christmas Vacation	97	Comedy	1989	Warner Brothers	PG-13
4	Coming to America	116	Comedy	1988	Paramount Pictures	R
5	Dracula	130	Horror	1993	Columbia TriStar	R
6	Dressed to Kill	105	Drama Mysteries	1980	Filmways Pictures	R
7	Forest Gump	142	Drama	1994	Paramount Pictures	PG-13
8	Ghost	127	Drama Romance	1990	Paramount Pictures	PG-13
9	Jaws	125	Action Adventure	1975	Universal Studios	PG
10	Jurassic Park	127	Action	1993	Universal Pictures	PG-13
11	Lethal Weapon	110	Action Cops & Robber	1987	Warner Brothers	R
12	Michael	106	Drama	1997	Warner Brothers	PG-13
13	National Lampoon's Vacation	98	Comedy	1983	Warner Brothers	PG-13
14	Poltergeist	115	Horror	1982	MGM / UA	PG
15	Rocky	120	Action Adventure	1976	MGM / UA	PG
16	Scarface	170	Action Cops & Robber	1983	Universal Studios	R
17	Silence of the Lambs	118	Drama Suspense	1991	Orion	R
18	Star Wars	124	Action Sci-Fi	1977	Lucas Film Ltd	PG
19	The Hunt for Red October	135	Action Adventure	1989	Paramount Pictures	PG
20	The Terminator	108	Action Sci-Fi	1984	Live Entertainment	R
21	The Wizard of Oz	101	Adventure	1939	MGM / UA	G
22	Titanic	194	Drama Romance	1997	Paramount Pictures	PG-13

The data stored in the ACTORS table consists of three columns: title, actor_leading, and actor_supporting, all of which are defined as character columns. The data stored in the Actors table is illustrated below.

ACTORS Table



	Title	Actor_Leading	Actor_Supporting
1	Brave Heart	Mel Gibson	Sophie Marceau
2	Christmas Vacation	Chevy Chase	Beverly D'Angelo
3	Coming to America	Eddie Murphy	Arsenio Hall
4	Forest Gump	Tom Hanks	Sally Field
5	Ghost	Patrick Swayze	Demi Moore
6	Lethal Weapon	Mel Gibson	Danny Glover
7	Michael	John Travolta	Andie MacDowell
8	National Lampoon's Vacation	Chevy Chase	Beverly D'Angelo
9	Rocky	Sylvester Stallone	Talia Shire
10	Silence of the Lambs	Anthony Hopkins	Jodie Foster
11	The Hunt for Red October	Sean Connery	Alec Baldwin
12	The Terminator	Arnold Schwarzenegger	Michael Biehn
13	Titanic	Leonardo DiCaprio	Kate Winslet

Base-SAS Tips, Tricks and Techniques

This section covers numerous base-SAS software tips, tricks and techniques. Whether you are a SAS expert who is comfortable with the many features offered in the Base SAS product or someone just getting started, these tips will make your programming experience a more rewarding one. By taking the time to learn the Base SAS fundamentals — including terminology, library structures, programming concepts, user interface, SAS Explorer, and numerous other aspects —you will get better results using this powerful product. You will learn about useful features step-by-step to help you become a more knowledgeable SAS user.

Tip #1 – Writing SAS source files included with a %INCLUDE statement to the SAS Log

By default, SAS source statements included with a %INCLUDE statement are NOT automatically written to the SAS Log. To print secondary SAS source statements included with a %INCLUDE statements, users will need to specify the **SOURCE2** system option. The default setting is **NOSOURCE2**. Either option can be specified in an OPTIONS statement, in the OPTIONS window, at SAS invocation, or in the configuration file.

Tip #2 – Specifying the most recently created data set to use in a read operation

By default, the **_LAST_=** system option is set to use the most recently created data set. This automatic feature is commonly found in procedure code when the DATA= option is omitted from a PROC. To override this built-in default, the **_LAST_=** system option can be defined as any valid temporary or permanent SAS data set name (refer to data set naming conventions). The **_LAST_=** option can be specified in an OPTIONS statement, in the OPTIONS window, at SAS invocation, or in the configuration file.

Tip #3 – Preventing SAS data sets from accidentally being replaced

Users can specify the **NOREPLACE** system option to prevent a permanent SAS data set from being accidentally replaced with another like-named data set. Conversely, by specifying the **REPLACE** system option like-named data sets can be overwritten. The REPLACE | NOREPLACE option can be specified in an OPTIONS statement, in the OPTIONS window, at SAS invocation, or in the configuration file.

Tip #4 – Executing Conditional Logic with IF-THEN / ELSE

The IF-THEN / ELSE construct enables a sequence of conditions to be constructed that when executed goes through the IF-THEN conditions until it finds one that satisfies the expression. The example shows a variable Movie_Length being assigned a character value of "Short", "Medium", or "Long" based on the mutually exclusive conditions specified in the IF-THEN and ELSE conditions. Although not required, the ELSE condition is useful to prevent missing values from being assigned to Movie_Length.

Code:

```
DATA ATTRIB_EXAMPLE;
 ATTRIB Movie_Length LENGTH=$7 LABEL='Movie Length';
 SET MOVIES;
 IF LENGTH < 120 THEN Movie_Length = 'Short';
 ELSE IF LENGTH > 160 THEN Movie_Length = 'Long';
 ELSE Movie_Length = 'Medium';
RUN;
```

Results

```
               The SAS System

Title                    Length  Movie_Length
Brave Heart                 177  Long
Casablanca                  103  Short
Christmas Vacation           97  Short
Coming to America           116  Short
Dracula                     130  Medium
Dressed to Kill             105  Short
Forrest Gump                142  Medium
Ghost                       127  Medium
```

```
Jaws                          125  Medium
Jurassic Park                 127  Medium
Lethal Weapon                 110  Short
Michael                       106  Short
National Lampoon's Vacation    98  Short
Poltergeist                   115  Short
Rocky                         120  Medium
Scarface                      170  Long
Silence of the Lambs          118  Short
Star Wars                     124  Medium
The Hunt for Red October      135  Medium
The Terminator                108  Short
The Wizard of Oz              101  Short
Titanic                       194  Long
```

Tip #5 – Executing Conditional Logic with a SELECT statement

As an alternative to a series of IF-THEN/ELSE statements, a **SELECT** statement with one or more **WHEN** conditions can be specified. The SELECT-WHEN statement enables a sequence of conditions to be constructed, that when executed goes through the WHEN conditions until it finds one that satisfies the expression. Typically a sequence of WHEN conditions are specified for a long series of conditions. The example shows a variable Movie_Length being assigned a character value of “Short”, “Medium”, or “Long” based on the mutually exclusive conditions specified in the WHEN and OTHERWISE conditions. Although not required, the OTHERWISE condition is useful to prevent missing values being assigned to Movie_Length.

Code:

```
DATA SELECT_EXAMPLE;
 SET MOVIES;
 LENGTH Movie_Length $6.;
 SELECT;
   WHEN (LENGTH < 120) Movie_Length = ‘Short’;
   WHEN (LENGTH > 160) Movie_Length = ‘Long’;
   OTHERWISE Movie_Length = ‘Medium’;
 END;
RUN;
```

Results

```
                 The SAS System

Title                         Length  Movie_Length
Brave Heart                      177  Long
Casablanca                       103  Short
Christmas Vacation                97  Short
Coming to America                116  Short
Dracula                          130  Medium
Dressed to Kill                  105  Short
Forrest Gump                     142  Medium
Ghost                            127  Medium
Jaws                             125  Medium
Jurassic Park                    127  Medium
Lethal Weapon                    110  Short
Michael                          106  Short
National Lampoon's Vacation       98  Short
```

```
Poltergeist                  115  Short
Rocky                        120  Medium
Scarface                     170  Long
Silence of the Lambs         118  Short
Star Wars                    124  Medium
The Hunt for Red October     135  Medium
The Terminator               108  Short
The Wizard of Oz             101  Short
Titanic                      194  Long
```

Case Logic

In the SQL procedure, a case expression provides a way of conditionally selecting result values from each row in a table (or view). Similar to an IF-THEN construct, a case expression uses a WHEN-THEN clause to conditionally process some but not all the rows in a table. An optional ELSE expression can be specified to handle an alternative action should none of the expression(s) identified in the WHEN condition(s) not be satisfied.

A case expression must be a valid SQL expression and conform to syntax rules similar to DATA step SELECT-WHEN statements. Even though this topic is best explained by example, let's take a quick look at the syntax.

```
CASE <column-name>
    WHEN when-condition THEN result-expression
   <WHEN when-condition THEN result-expression> …
   <ELSE result-expression>
END
```

A column-name can optionally be specified as part of the CASE-expression. If present, it is automatically made available to each when-condition. When it is not specified, the column-name must be coded in each when-condition. Let's examine how a case expression works.

If a when-condition is satisfied by a row in a table (or view), then it is considered "true" and the result-expression following the THEN keyword is processed. The remaining WHEN conditions in the CASE expression are skipped. If a when-condition is "false", the next when-condition is evaluated. SQL evaluates each when-condition until a "true" condition is found or in the event all when-conditions are "false", it then executes the ELSE expression and assigns its value to the CASE expression's result. A missing value is assigned to a CASE expression when an ELSE expression is not specified and each when-condition is "false".

Tip #6 – Conditional Execution in SQL with a Case Expression

In the following example, we will examine how a case expression actually works. Suppose a value of "Short", "Medium", or "Long" is desired for each of the movies. Using the movie's length (LENGTH) column, a CASE expression is constructed to assign one of the desired values in a unique column called Movie_Length. A value of 'Short' is assigned to the movies that are shorter than 120 minutes long, 'Long' for movies longer than 160 minutes long, and 'Medium' for all other movies. A column heading of Movie_Length is assigned to the new derived output column using the AS keyword.

SQL Code

```
PROC SQL;
  SELECT TITLE,
         LENGTH,
         CASE
           WHEN LENGTH < 120 THEN 'Short'
           WHEN LENGTH > 160 THEN 'Long'
           ELSE 'Medium'
         END AS Movie_Length
    FROM MOVIES;
QUIT;
```

Results

```
               The SAS System

Title                     Length  Movie_Length
Brave Heart                  177  Long
Casablanca                   103  Short
Christmas Vacation            97  Short
Coming to America            116  Short
Dracula                      130  Medium
Dressed to Kill              105  Short
Forrest Gump                 142  Medium
Ghost                        127  Medium
Jaws                         125  Medium
Jurassic Park                127  Medium
Lethal Weapon                110  Short
Michael                      106  Short
National Lampoon's Vacation   98  Short
Poltergeist                  115  Short
Rocky                        120  Medium
Scarface                     170  Long
Silence of the Lambs         118  Short
Star Wars                    124  Medium
The Hunt for Red October     135  Medium
The Terminator               108  Short
The Wizard of Oz             101  Short
Titanic                      194  Long
```

Output Delivery Basics

The SAS® Output Delivery System ***(ODS)*** provides many ways to format output. It controls the way output is accessed and formatted. Although ODS continues to support the creation of traditional SAS listing or monospace output (i.e., Listing), it provides many new features and greater flexibility when working with output.

An early version of ODS appeared in Version 6.12 as a way to address the inherent weaknesses found in traditional SAS output. It enables "quality" looking output to be produced without having to import it into word processors such as MS-Word. In Version 8 and 9, many new output formatting features and options are introduced for SAS users to take advantage of. Users have a powerful and easy way to create and access formatted procedure and DATA step output.

Tip #7 – ODS and "Batch" Use

Many of the ODS features found in the interactive side of the SAS Display Manager System (DMS) can also be used in batch processing. ODS has been designed to make exciting new formatting options available to users. In a windowing environment, ODS can send output to the following destinations: the output window (DMS), the listing file, HTML, SAS dataset, rich text format (RTF), postscript file, external output file (non-SAS file), or output device. The only exception for batch processing is having output sent to the output window.

Tip #8 – What if I'm Still Not Using the latest Version of SAS Software

First introduced in Version 6.12, ODS offered users the capability to format output to destinations other than traditional line printers. Version 6.12 introduced the ability to deploy output to the web, the creation of SAS datasets and rich text format (RTF) files, and DATA step interaction. ODS was designed to address the inherent weaknesses found in traditional SAS output. It enables the creation of "quality" looking output without having to import it into word processors such as MS-Word. New output enhancements were introduced in Version 8 and then in Version 9, including the ability to create postscript files and output customizations. To take full advantage of the power offered in ODS, it is recommended that users upgrade to the latest Version as early as possible to take advantage of these features.

Tip #9 – ODS and System Resources

A very important efficiency consideration is to remember that ODS currently supports the following destinations: 1) Listing, 2) HTML, 3) rich text format (RTF), 4) postscript, and 5) Output. (Note: It also provides support in the DATA step.) Each ODS destination can be open or closed at the same time. For each open destination, ODS sends output object(s) to it. System resources are used when a destination is open. As a result, make sure any and all unwanted open destinations are closed to conserve on resources.

Tip #10 – Closing Destinations before and after use

The Listing destination is open by default at SAS invocation, while the other destinations are closed. If nothing is done to suppress output to the Listing destination, your SAS programs automatically produce Listing output, just as they always have in the SAS System. If you needed to suppress printed output from being sent to the Listing destination (or DMS Output window) before the execution of a procedure step, the following ODS statement would be issued:

Code

```
ODS Listing Close;
 Proc univariate data=movies;
 Run;
ODS Listing;
```

By closing the Listing destination before the procedure code, the SAS System is actually suppressing output to that destination until it is reopened. The preceding example shows that at the end of the procedure step, the Listing destination is reopened by specifying ODS Listing; so output from subsequent steps can be sent to the Listing destination.

Tip #11 – Deleting Output from the Results Window

The Results window identifies procedure output that has been produced, providing users with an improved way to manage their output. It is customarily a good thing to remove unwanted output displayed in this window to conserve on system resources. The Results window is opened by specifying the command **ODSRESULTS** on the DMS command line or by selecting **View Results** from the pull-down menu. To delete procedure output, use the following steps:

1. Select the procedure folder you want to remove.
2. Click the Delete button on the task bar.
3. Select “Yes” to confirm the deletion of the procedure output folder.

Tip #12 – Tracing Procedure Output

Output producing procedures often create multiple pieces or tables of information. In order to discriminate between the various pieces of information, it is advantageous to know the names assigned to each piece of information. The ability to display the names of individual pieces of information generated on output is referred to as ***tracing***. The ODS statement syntax ODS trace ON / Listing; causes the SAS System to turn the trace feature on and print results to the SAS Listing destination.

Code

```
ODS Trace ON / Listing;
  Proc univariate data=movies;
  Run;
ODS Trace Off;
```

The trace record displays information about the data component, the table definition, and the output object. For example, the trace record displays the following output objects to the SAS Listing destination: 1) Moments, 2) BasicMeasures, 3) TestForLocation, 4) Quantiles, and 5) ExtremeObs. A sample trace record containing each output object’s name, label, template, and path is displayed for the Univariate procedure. Note that for each output object, the name, label, template, and path is displayed.

```
Output Added:
-------------
Name:       Moments
Label:      Moments
Template:   base.univariate.Moments
Path:       Univariate.age.Moments
-------------

Output Added:
-------------
Name:       BasicMeasures
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:       Univariate.age.BasicMeasures
-------------

Output Added:
-------------
Name:       TestsForLocation
Label:      Tests For Location
Template:   base.univariate.Location
Path:       Univariate.age.TestsForLocation
-------------

Output Added:
-------------
Name:       Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:       Univariate.age.Quantiles
-------------

Output Added:
-------------
Name:       ExtremeObs
Label:      Extreme Observations
Template:   base.univariate.ExtObs
Path:       Univariate.age.ExtremeObs
-------------
```

Selecting Output with ODS

A selection or exclusion list exists for each open ODS destination. These lists determine which output objects to send to ODS destinations. To accomplish this, ODS verifies whether an output object is included in a destination's selection or exclusion list. If it does not appear in this list, then the output object is not sent to the ODS destination. If an output object is included in the list, ODS determines if the object is included in the overall list. If it does not appear in this list, then the output object is not sent to the ODS destination. If an output object is included in the overall list then ODS sends it to the selected destination.

Tip #13 – Selecting Desired Pieces of Information

Once you know the individual names of each output object (from the trace), you can then select the desired object for reporting purposes. The syntax is:

ODS select *output-component-name*;

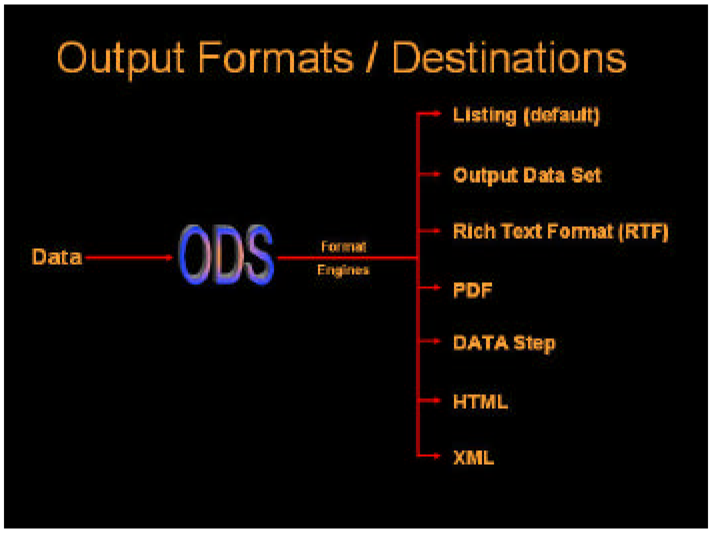
where ***output-component-name*** is the name of the desired output object. To select just the output object Moments from the Univariate procedure, the following syntax is specified:

Code

```
ODS Select Moments;
  Proc univariate data=movies;
  Run;
```

Creating Unique Output with ODS

Output Delivery System (ODS) can be used to create a variety of output formats. ODS statements are classified as global statements and are processed immediately by the SAS System. ODS statement options control what format engine(s) are turned on and in effect during the step or until another ODS statement is specified. ODS has built-in format engines (e.g., Listing, Output, RTF, PDF, DATA Step, HTML, and XML). Specifying an ODS statement and destination at a particular point in a program is important, because output-producing PROC and DATA steps will respond by sending output to the open destination.



Tip #14 – Creating SAS Output Data Sets with ODS

Occasionally, output results are needed in a SAS data set rather than in printed form such as the Listing destination. Re-directing SAS procedure output to a data set is relatively simple with ODS. The syntax is:

ODS Output *output-table-name* =
***user-defined-table-name*;**
< SAS Code >

where ***output-table-name*** is the name of the desired output table (component) containing the information you want written to a data set, such as ***Moments*** in the UNIVARIATE procedure. ***User-defined-table-name*** is the name you supply for the newly created data set. It can be defined as either a temporary or permanent data set. Once an object is selected, specify the object in the ODS OUTPUT statement. For example, the Moments from the Univariate procedure is selected and output to a SAS data set in the following code.

Code

```
ODS Listing Close;
ODS Output Moments  = Movie_Moments;
  Proc univariate data=movies;
    Run;
ODS Listing;
```

When the OUTPUT destination is no longer needed, it can be closed with the following ODS statement:

ODS OUTPUT CLOSE;

Tip #15 – Creating Rich Text Format (RTF) with ODS

Rich text format (RTF) is text consisting of formatting attributes codes, such as boldface, italics, underline, etc. It is principally used to encapsulate text and formatting attributes during copy-and-paste operations. Because word-processing programs use RTF rather than ASCII when handling data, the need to reformat is a thing of the past. The syntax to create RTF output is:

ODS RTF FILE = 'user-specified-file-name';

where user-specified-file-name references a complete and fully-qualified output location for the creation and storage of the RTF file, data, and codes. For example, the following code creates an RTF file using the Univariate procedure output. (**<u>Note:</u>** The RTF extension is required).

<u>Code</u>

```
ODS Select Moments = moments;
ODS RTF FILE='ods-rtf-univariate.rtf';
 Proc univariate data=movies;
   Title1 'Delivering RTF Output';
 Run;
ODS RTF Close;
```

The results of the RTF output are displayed below:

Delivering RTF Output
The UNIVARIATE Procedure
Variable: Year

Moments			
N	22	**Sum Weights**	22
Mean	1982.909 09	**Sum Observations**	43624
Std Deviation	15.21249 2	**Variance**	231.4199 13
Skewness	- 2.110647 3	**Kurtosis**	4.418072 09
Uncorrected SS	86507286	**Corrected SS**	4859.818 18
Coeff Variation	0.767180 51	**Std Error Mean**	3.243314 2

Tip #16 – Creating PDF Output

To share output electronically, Adobe created a proprietary format called PDF. The objective of PDF is to enable the printing of output exactly as it is seen. The significance of PDF output is that it is a great format for Web deployment since it is completely independent of any printer destination. To create PDF output from the UNIVARIATE procedure, the ODS PDF option can be specified as follows .

Code

```
ODS Listing Close;
ODS PDF FILE='ods-univariate.pdf';
  proc univariate data=movies;
    title1 'Creating PDF Output with ODS';
  run;
ODS PDF Close;
ODS Listing;
```

Output Delivery Goes Web

The Web offers incredible potential that impacts all corners of society. With its increasing popularity as a communications medium, Web publishers have arguably established the Web as the greatest medium ever created. Businesses, government agencies, professional associations, schools, libraries, research agencies, and a potpourri of society's true believers have endorsed the Web as an efficient means of conveying their messages to the world. The 24/7 model permits information to be refreshed and updated continuously as new material becomes available.

Tip #17 – Pagesize / Linesize Settings

The Options PS= and LS= have no effect when used with the HTML destination (opposed to most other output-producing steps that generate output to a print destination). If the PS= and/or LS= options are used with the HTML destination, they are simply ignored. The SAS System creates a type of "streaming" or continuous output that can be viewed with a web browser for easy navigation.

The SAS System does provide a way for users to paginate through output displayed in a body file. The HTML destination provides a way to designate an optional description of each page of the body file. The PAGE= file (when specified) recognizes each new page of output produced by ODS. What ODS does is create a section called ***Table of Pages*** containing links to the body file for easy navigation through output.

Tip #18 – Creating HTML Destination Files with ODS

Four types of files can be created with the ODS HTML destination: 1) body, 2) contents, 3) page, and 4) frame. The **Body** file contains the results from the procedure embedded in ODS-generated HTML code. Horizontal and vertical scroll bars are automatically placed on the generated page, if necessary.

The **Contents** file consists of a link to each HTML table within the body file. It uses an anchor tag to link to each table. By using your browser software, you can view the contents file directly or as part of the frame file.

The **Page** file consists of a link to each page of ODS created output. By using your browser, you can view the page file directly or as part of the frame file.

The **Frame** file displays the body file and the contents file, the page file, or both. The next example shows the creation of Web-ready Univariate procedure output using the HTML format engine.

Code

```
ODS Listing Close;
ODS HTML body='ods-body.htm'
             contents='ods-contents.htm'
                    page='ods-page.htm'
                  frame='ods-frame.htm';
proc univariate data=movies;
 Title1 'Creating HTML Output with ODS';
Run;
ODS HTML Close;
ODS Listing;
```

A snippet of the HTML output appears on the next page:

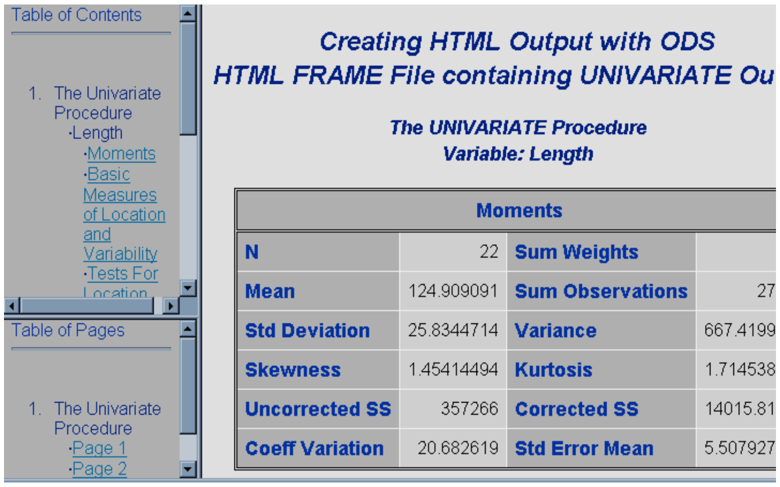


Table of Contents

1. The Univariate Procedure
·Length
·Moments
·Basic Measures of Location and Variability
·Tests For Location

Table of Pages

1. The Univariate Procedure
·Page 1
·Page 2

Creating HTML Output with ODS
HTML FRAME File containing UNIVARIATE Ou

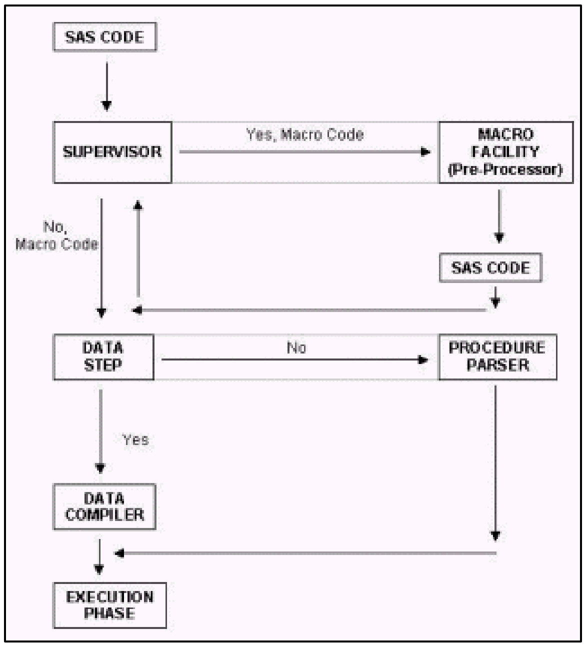
The UNIVARIATE Procedure
Variable: Length

Moments			
N	22	Sum Weights	
Mean	124.909091	Sum Observations	27
Std Deviation	25.8344714	Variance	667.4199
Skewness	1.45414494	Kurtosis	1.714538
Uncorrected SS	357266	Corrected SS	14015.81
Coeff Variation	20.682619	Std Error Mean	5.507927

Macro Language Basics

The macro language provides an additional set of tools to assist in: 1) communicating between SAS steps, 2) constructing executable and reusable code, 3) designing custom languages, 4) developing user-friendly routines, and 5) conditionally execute DATA or PROC steps.

When a program is run, the SAS System first checks to see if a macro statement exists. If the program does not contain any macro statements, then processing continues as normal with the DATA or PROC step processor. If the program does contain one or more macro statements, then the macro processor must first execute them. The result of this execution is the production of character information, macro variables, or SAS statements, which are then be passed to the DATA or PROC step processor. The control flow of a macro process appears in Figure 1 below.



Tip #19 – Debugging a Macro with SAS System Options

The SAS System offers users a number of useful system options to help debug macro issues and problems. The results associated with using macro options are automatically displayed on the SAS Log. Specific options related to macro debugging appear in alphabetical order in the table below.

SAS Option	Description
MACRO	Specifies that the macro language SYMGET and SYMPUT functions be a vailable.
MEMERR	Controls Diagnostics.
MEMRPT	Specifies that memory usage statistics be displayed on the SAS Log.
MERROR	Presents Warning Messages when there are misspellings or when an undefined macro is called.
MLOGIC	Macro execution is traced and displayed on the SAS Log for debugging purposes.
MPRINT	SAS statements generated by macro execution are traced on the SAS Log for debugging purposes.
SYMBOLGEN	Displays text from expanding macro variables to the SAS Log.

Tip #20 – Streamlining Command-line DMS Commands with a Macro

The macro language is a wonderful tool for streamlining frequently entered SAS Display Manager System (DMS) commands to reduce the number of keystrokes. By embedding a series of DMS commands inside a simple macro, you'll not only save by not having to enter them over and over again, but you'll improve your productivity as well. The following macro code illustrates a series of DMS commands being strung together in lieu of entering them individually on a Display Manager command line. The commands display and expand the SAS Log to full size respectively, and then position the cursor at the top of the log. Once the macro is defined, it can be called by entering %POSTSUBMIT on any DMS command line to activate the commands.

Macro Code

```
%MACRO postsubmit;
   Log;
   Zoom;
   Top;
%MEND postsubmit;
```

Tip #21 – Assigning a Defined Macro to a Function Key

To further reduce keystrokes and enhance user productivity even further, a call to a defined macro can be saved to a Function Key. The purpose for doing this would be to allow for one-button operation of any defined macro. To illustrate the process of saving a macro call to a Function Key, the %POSTSUBMIT macro defined in the previous tip is assigned to Function Key F12 in the KEYS window. The partial KEYS window is displayed to illustrate the process.

KEYS Window

Key	Definition
F1	help
F2	reshow
F3	end;
...	...
F10	keys
F11	command focus
F12	**%POSTSUBMIT**

Building Macro Tools

The Macro Facility, combined with the capabilities of the SQL procedure, enables the creation of versatile macro tools and general-purpose applications. A principle design goal when developing user-written macros should be that they are useful and simple to use. A macro that violates this tenant of little applicability to user needs, or with complicated and hard to remember macro variable names, are usually avoided.

As tools, macros should be designed to serve the needs of as many users as possible. They should contain no ambiguities, consist of distinctive macro variable names, avoid the possibility of naming conflicts between macro variables and data set variables, and not try to do too many things. This utilitarian approach to macro design helps gain the widespread approval and acceptance by users.

Tip #22 – Defining a Macro with Positional Parameters

Macros are frequently designed to allow the passing of one or more parameters. This allows the creation of macro variables so text strings can be passed into the macro. The order of macro variables as positional parameters is specified when the macro is coded. The assignment of values for each positional parameter is supplied at the time the macro is called.

To illustrate the definition of a two positional parameter macro, the following macro was created to display all table names (data sets) that contain the variable TITLE in the user-assigned MYDATA libref as a cross-reference listing. To retrieve the needed type of information, you could execute multiple PROC CONTENTS against selected tables. Or in a more efficient method, you could retrieve the information directly from the read-only Dictionary table COLUMNS with the selected columns LIBNAME, MEMNAME, NAME, TYPE and LENGTH, as shown. For more information about Dictionary tables, readers may want to view the “free” SAS Press Webinar by Kirk Paul Lafler at http://support.sas.com/publishing/bbu/webinar.html#lafler2 or the published paper by Kirk Paul Lafler, Exploring Dictionary Tables and SASHELP Views.

Macro Code

```
%MACRO COLUMNS(LIB, COLNAME);
  PROC SQL;
    SELECT LIBNAME, MEMNAME, NAME, TYPE, LENGTH
      FROM DICTIONARY.COLUMNS
        WHERE UPCASE(LIBNAME)="&LIB" AND
              UPCASE(NAME)="&COLNAME" AND
              UPCASE(MEMTYPE)="DATA";
  QUIT;
%MEND COLUMNS;

%COLUMNS(MYDATA,TITLE);
```

After Macro Resolution

```
PROC SQL;
  SELECT LIBNAME, MEMNAME, NAME, TYPE, LENGTH
    FROM DICTIONARY.COLUMNS
      WHERE UPCASE(LIBNAME)="MYDATA"  AND
            UPCASE(NAME)="TITLE" AND
            UPCASE(MEMTYPE)="DATA";
QUIT;
```

Output

```
Library                                         Column    Column
Name        Member Name         Column Name     Type      Length
MYDATA      ACTORS              Title           char          30
MYDATA      MOVIES              Title           char          30
MYDATA      PG_MOVIES           Title           char          30
MYDATA      PG_RATED_MOVIES     Title           char          30
MYDATA      RENTAL_INFO         Title           char          30
```

Tip #23 – Defining another Macro with Positional Parameters

Now let's examine another useful macro that is designed with a positional parameter. The following macro is designed to accept one positional parameter called &LIB. When called, it accesses the read-only Dictionary table TABLES to display each table name and the number of observations in the user-assigned MYDATA libref. This macro provides a handy way to quickly determine the number of observations in one or all tables in a libref without having to execute multiple PROC CONTENTS by using the stored information in the Dictionary table TABLES.

Macro Code

```
%MACRO NUMROWS(LIB);
  PROC SQL;
    SELECT LIBNAME, MEMNAME, NOBS
      FROM DICTIONARY.TABLES
        WHERE UPCASE(LIBNAME)="&LIB" AND
            UPCASE(MEMTYPE)="DATA";
  QUIT;
%MEND NUMROWS;

%NUMROWS(MYDATA);
```

After Macro Resolution

```
PROC SQL;
  SELECT LIBNAME, MEMNAME, NOBS
    FROM DICTIONARY.TABLES
      WHERE UPCASE(LIBNAME)="MYDATA" AND
            UPCASE(MEMTYPE)="DATA";
QUIT;
```

Output

```
Library                                       Number of Physical
Name       Member Name                              Observations
MYDATA     ACTORS                                             13
MYDATA     CUSTOMERS                                           3
MYDATA     MOVIES                                             22
MYDATA     PG_RATED_MOVIES                                    13
```

Conclusion

The base-SAS® System offers users the power of a comprehensive DATA step programming language, an assortment of powerful PROCs, a user-friendly interfaces including the SAS Display Manager, and a macro language that extends the capabilities of the SAS System. This paper illustrates numerous SAS tips, tricks and techniques using a collection of proven code examples related to effectively specifying SAS System options; process DATA step statements to handle subroutines and code libraries; produce a variety of output formats; construct reusable code; and troubleshoot and debug code.

References

Lafler, Kirk Paul, Advanced SAS® Programming Tips and Techniques; Software Intelligence Corporation, Spring Valley, CA, USA; 1987-2007.

Lafler, Kirk Paul (2007), *"Undocumented and Hard-to-find PROC SQL Features,"* Proceedings of the PharmaSUG 2007 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul and Ben Cochran (2007), *"A Hands-on Tour Inside the World of PROC SQL Features,"* Proceedings of the SAS Global Forum (SGF) 2007 Conference, Software Intelligence Corporation, Spring Valley, CA, and The Bedford Group, USA.

Lafler, Kirk Paul (2007), SAS System Macro Language Course Notes, Fourth Edition. Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2006), Exploring DICTIONARY Tables and SASHELP Views, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2006), *"A Hands-on Tour Inside the World of PROC SQL,"* Proceedings of the 31st Annual SAS Users Group International Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2005), *"Manipulating Data with PROC SQL,"* Proceedings of the 30th Annual SAS Users Group International Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2004). *PROC SQL: Beyond the Basics Using SAS*, SAS Institute Inc., Cary, NC, USA.

Lafler, Kirk Paul (2003), "*Undocumented and Hard-to-find PROC SQL Features*," *Proceedings of the Eleventh Annual Western Users of SAS Software Conference*.

Lafler, Kirk Paul, PROC SQL Programming for Beginners; Software Intelligence Corporation, Spring Valley, CA, USA; 1992-2007.

Lafler, Kirk Paul, Intermediate PROC SQL Programming; Software Intelligence Corporation, Spring Valley, CA, USA; 1998-2007.

Lafler, Kirk Paul, Advanced PROC SQL Programming; Software Intelligence Corporation, Spring Valley, CA, USA; 2001-2007.

Lafler, Kirk Paul, PROC SQL Programming Tips; Software Intelligence Corporation, Spring Valley, CA, USA; 2002-2007.

SAS® Guide to the SQL Procedure: Usage and Reference, Version 6, First Edition; SAS Institute, Cary, NC, USA; 1990.

SAS® SQL Procedure User's Guide, Version 8; SAS Institute Inc., Cary, NC, USA; 2000.

Trademark Citations

Acknowledgments

I would like to thank Curtis Mack and Pete Lund, PNWSUG 2007 Conference Chairs, for accepting this paper, as well as the PNWSUG Leadership for their support of a great Conference.

Author Information

Kirk Paul Lafler is consultant and founder of Software Intelligence Corporation and has been programming in SAS since 1979. As a SAS Certified Professional and SAS Institute Alliance Member (1996 – 2002), Kirk provides IT consulting services and training to SAS users around the world. As the author of four books including *PROC SQL: Beyond the Basics Using SAS* (SAS Institute. 2004), Kirk has written more than two hundred peer-reviewed papers and articles that have appeared in professional journals and SAS User Group proceedings. He has also been an Invited speaker at more than two hundred SAS International, regional, local, and special-interest user group conferences and meetings throughout North America. His popular SAS Tips column, "Kirk's Korner of Quick and Simple Tips", appears regularly in several SAS User Group newsletters and Web sites, and his fun-filled SASword Puzzles is featured in SAScommunity.org.

Comments and suggestions can be sent to:

Kirk Paul Lafler
Software Intelligence Corporation
World Headquarters
P.O. Box 1390
Spring Valley, California 91979-1390
E-mail: KirkLafler@cs.com