Paper RF-003

WHERE, Oh, WHERE Art Thou? A Cautionary Tale for Using WHERE Statements and WHERE= Options

Britney D. Gilbert, Juniper Tree Consulting, Porter, Oklahoma

ABSTRACT

Using WHERE statements in your SAS programming can make processing data more efficient. However, the programmer needs to have a full understanding and be aware of how SAS processes multiple WHERE statements or combinations of WHERE statements and WHERE= dataset options in DATA steps and PROCEDURES. This paper explores examples of the combinations of WHERE statements and WHERE= dataset options in both DATA steps and PROCEDURES and the resulting logs and output.

INTRODUCTION

Whether we need to process large datasets or produce repeat tables, listings, or figures, the WHERE statement and/or WHERE= dataset option is one of the fundamental tools a programmer has in their toolbox. As such, it is important to understand how they work independently and in combination in both DATA steps and PROCEDURES.

To review, WHERE statements increase efficiency by only processing records that meet the conditions specified in the WHERE clause. When using multiple input datasets, a WHERE= dataset option can be used to select records for processing for that specific input dataset. But, what happens when we use a WHERE statement and WHERE= in combination or if we program multiple WHERE statements in a DATA step or PROCEDURE?

Display 1. Sample Dataset

VIEWTABLE: My_lib.Adsl

	USUBJID	SITEN	SITE	SEXN	SEX	AGE	AGEGR1N	AGEGR1	FASFL
1	MYSTUDY 002-001	002	Europe	1	Female	47	1	<= 65 Year	Y
2	MYSTUDY 003-002	003	Other	2	Male	36	1	<= 65 Year	Y
3	MYSTUDY 001-003	001	US	2	Male	36	1	<= 65 Year	Y
4	MYSTUDY 002-004	002	Europe	2	Male	30	1	<= 65 Year	Y
5	MYSTUDY 003-005	003	Other	2	Female	40	1	<= 65 Year	Y
6	MYSTUDY 001-006	001	US	1	Male	39	1	<= 65 Year	Y
7	MYSTUDY 002-007	002	Europe	1	Female	47	1	<= 65 Year	Y
8	MYSTUDY 003-008	003	Other	1	Female	45	1	<= 65 Year	Y
9	MYSTUDY 001-009	001	US	2	Male	38	1	<= 65 Year	Y
10	MYSTUDY 002-010	002	Europe	1	Female	29	1	<= 65 Year	Y
11	MYSTUDY 003-011	003	Other	2	Male	58	1	<= 65 Year	Y
12	MYSTUDY 001-012	001	US	1	Male	39	1	<= 65 Year	Y
13	MYSTUDY 002-013	002	Europe	1	Male	61	1	<= 65 Year	Y
14	MYSTUDY 003-014	003	Other	1	Male	21	1	<= 65 Year	Y
15	MYSTUDY 001-015	001	US	2	Male	36	1	<= 65 Year	Y
16	MYSTUDY 002-016	002	Europe	1	Male	47	1	<= 65 Year	Y
17	MYSTUDY 003-017	003	Other	1	Male	56	1	<= 65 Year	Y
18	MYSTUDY 001-018	001	US	1	Female	57	1	<= 65 Year	Y
19	MYSTUDY 002-019	002	Europe	1	Female	65	1	<= 65 Year	Y
20	MYSTUDY 003-020	003	Other	2	Female	49	1	<= 65 Year	Y
21	MYSTUDY 001-021	001	US	2	Female	31	1	<= 65 Year	Y
22	MYSTUDY 002-022	002	Europe	1	Male	65	1	<= 65 Year	Y
23	MYSTUDY 003-023	003	Other	2	Female	39	1	<= 65 Year	Y
24	MYSTUDY 001-024	001	US	2	Male	58	1	<= 65 Year	Y
25	MYSTUDY 002-025	002	Europe	2	Male	39	1	<= 65 Year	Y
26	MYSTUDY 003-026	003	Other	1	Female	24	1	<= 65 Year	Y
27	MYSTUDY 001-027	001	US	2	Female	49	1	<= 65 Year	Y
28	MYSTUDY 002-028	002	Europe	2	Female				Y
29	MYSTUDY 003-029	003	Other	2	Male				Y
30	MYSTUDY 001-030	001	US	1	Male				
31	MYSTUDY 002-031	002	Europe						
32	MYSTUDY 003-032	003	Other						
	MYSTUDY 001-033								

To demonstrate the behaviors of WHERE statements and WHERE= dataset option, a sample dataset and SAS® 9.3 was used. In this sample ADSL dataset, there are 100 subjects from three sites (001,002,003) in three regions (US, Europe, Other). Sex and age variables are included and an age grouping (AGEGR1) and a population flag (FASFL) has been derived.

WHERE STATEMENTS AND OPTIONS IN A DATA STEP

Using a WHERE reduces the number of records read into the DATA step as opposed to IF statements where all records are read first and then filtered by the conditions of the statement. However, unlike IF statements, multiple WHERE statements cannot be used in a single DATA step unless you use a WHERE ALSO or a WHERE SAME AND statement.

The WHERE ALSO and WHERE SAME ALSO statement will execute without a proceeding WHERE statement, but the log will indicate that the "WHERE clause has been augmented." This is exemplified in Display 2. Has the WHERE clause been augmented? One condition was specified and one condition was processed.

Display 2. WHERE Statement Log Note

Log - (Untitled)

```
215
216  Data AUG_NOTE1;
217    Set MY_LIB.ADSL;
218    Where ALSO FASFL = "Y";
NOTE: WHERE clause has been augmented.
219  Run;

NOTE: There were 99 observations read from the data set MY_LIB.ADSL.
      WHERE FASFL='Y';
NOTE: The data set WORK.AUG_NOTE1 has 99 observations and 9 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


220
221  Proc Sort Data = MY_LIB.ADSL
222                Out = AUG_NOTE2;
223    Where SAME AND FASFL = "Y";
NOTE: WHERE clause has been augmented.
224    By USUBJID;
225  Run;

NOTE: There were 99 observations read from the data set MY_LIB.ADSL.
      WHERE FASFL='Y';
NOTE: The data set WORK.AUG_NOTE2 has 99 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

If a log check program is used for review, then the programmer needs to be cautious when only looking for key phrases without a full log review of the subsequent log NOTES and record counts to verify expected results.

USING MULTIPLE WHERE STATEMENTS IN A DATA STEP

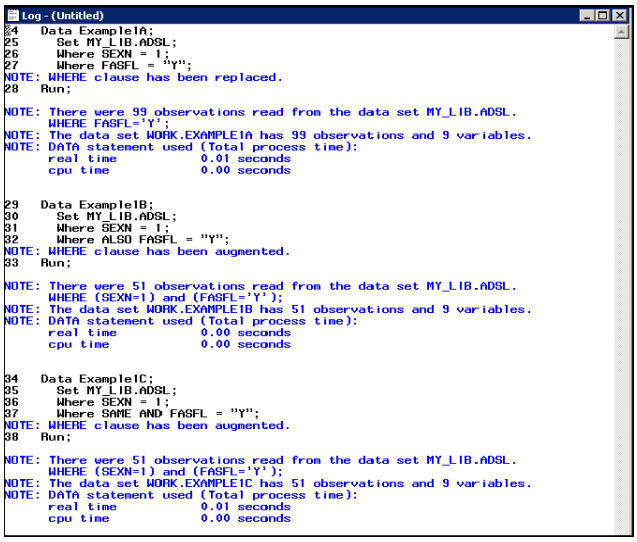
As previously mentioned, multiple WHERE statements cannot be used in a DATA step in the same manner as IF statements. When SAS® encounters a second WHERE statement in a DATA step, the first is replaced by the second.

In Display 3, the programming examples want to subset our sample ADSL dataset for the Full Analysis Set (FASFL = "Y") and Male Subjects (SEXN = 1).

At first, the program uses a DATA step with two WHERE statements: *Where SEXN = 1* and *Where FASFL = "Y"*. Reviewing the log NOTES, we see that the WHERE statement has been replaced and the resulting dataset EXAMPLE1A is only subsetted by FASFL = "Y" and has 99 observations. This is not our expected result.

Next, the program uses a DATA step with a WHERE statement followed by a WHERE ALSO statement: *WHERE SEXN = 1* and *WHERE ALSO FASFL = "Y"*. With this program, the resulting dataset EXAMPLE1B has 51 records and the log confirms that the records with both FASFL = "Y" and SEXN = 1 are read and processed. This is our expected result!

Display 3. Multiple WHERE Statements in a Data Step



```
Log - (Untitled)
24   Data Example1A;
25     Set MY_LIB.ADSL;
26     Where SEXN = 1;
27     Where FASFL = "Y";
NOTE: WHERE clause has been replaced.
28   Run;

NOTE: There were 99 observations read from the data set MY_LIB.ADSL.
      WHERE FASFL='Y';
NOTE: The data set WORK.EXAMPLE1A has 99 observations and 9 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds


29   Data Example1B;
30     Set MY_LIB.ADSL;
31     Where SEXN = 1;
32     Where ALSO FASFL = "Y";
NOTE: WHERE clause has been augmented.
33   Run;

NOTE: There were 51 observations read from the data set MY_LIB.ADSL.
      WHERE (SEXN=1) and (FASFL='Y');
NOTE: The data set WORK.EXAMPLE1B has 51 observations and 9 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


34   Data Example1C;
35     Set MY_LIB.ADSL;
36     Where SEXN = 1;
37     Where SAME AND FASFL = "Y";
NOTE: WHERE clause has been augmented.
38   Run;

NOTE: There were 51 observations read from the data set MY_LIB.ADSL.
      WHERE (SEXN=1) and (FASFL='Y');
NOTE: The data set WORK.EXAMPLE1C has 51 observations and 9 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

Finally to confirm the behavior of a WHERE SAME AND statement, a similar program is constructed using a WHERE statement followed by a WHERE SAME AND statement. The resulting dataset EXAMPLE1C has the expected 51 observations and the subsetting is confirmed in the log.

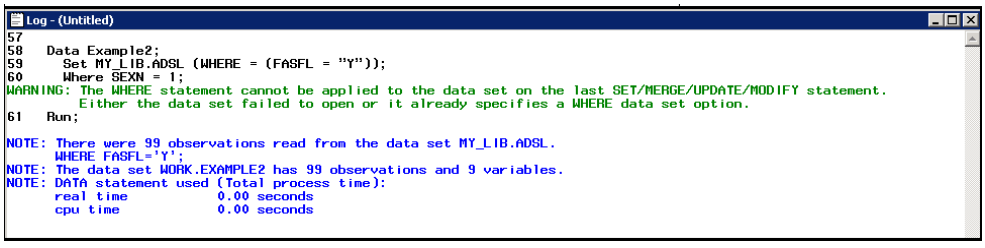
USING A WHERE STATEMENTS AND A WHERE= OPTION IN A DATA STEP

Using the combination of a WHERE statement and a WHERE= dataset option in a DATA step, is not a successful programming strategy. When a WHERE statement and WHERE= dataset option apply to the same dataset, SAS® uses the DATA step option and ignores the WHERE statement.

In Display 4, the programming example attempts to subset the ADSL for the Full Analysis Set (FASFL = "Y") and Male Subjects (SEXN = 1) using a WHERE= option, *WHERE = (FASFL = "Y")*, and a WHERE statement, *WHERE SEXN = 1*. In these cases, only the WHERE= option will be processed and the resulting dataset EXAMPLE2 has 99 observations. Again, this is not our expected result.

Reviewing the log NOTES, confirm that records with FASFL = "Y" were processed and not FASFL = "Y" **AND** SEXN = 1. Further, a WARNING appears in the log indicating that the WHERE statement cannot be applied.

Display 4. WHERE= Option and WHERE Statement in a Data Step



```
Log - (Untitled)
57
58   Data Example2;
59     Set MY_LIB.ADSL (WHERE = (FASFL = "Y"));
60     Where SEXN = 1;
WARNING: The WHERE statement cannot be applied to the data set on the last SET/MERGE/UPDATE/MODIFY statement.
         Either the data set failed to open or it already specifies a WHERE data set option.
61   Run;

NOTE: There were 99 observations read from the data set MY_LIB.ADSL.
      WHERE FASFL='Y';
NOTE: The data set WORK.EXAMPLE2 has 99 observations and 9 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

WHERE STATEMENTS AND OPTIONS IN A PROCEDURE

Because IF statements are only valid in DATA steps, WHERE statements and WHERE= dataset options are used to subset observations in SAS® PROCEDURES. However, using multiple WHERE statements and combinations of WHERE statements and WHERE= dataset options behave differently.

USING MULTIPLE WHERE STATEMENTS IN A PROCEDURE

Multiple WHERE statements in a SAS® PROCEDURE behave the same as when they are used in a DATA step. When SAS® encounters a second WHERE statement in a PROCEDURE, the first is replaced by the second.

In Display 5, the programming examples want to sort a subset our sample ADSL dataset for the Full Analysis Set (FASFL = "Y") and Male Subjects (SEXN = 1).

At first, the program uses a PROC SORT with two WHERE statements: *WHERE SEXN = 1* and *WHERE FASFL = "Y"*. Reviewing the log NOTES, we see that the WHERE statement has been replaced and the resulting dataset EXAMPLE3A is only subsetted by FASFL = "Y" before sorting and has 99 observations. Again, this is not our expected result.

Next, the program submits a PROC SORT with a WHERE statement followed by a WHERE ALSO statement: *WHERE SEXN = 1* and *WHERE ALSO FASFL = "Y"*. With this program, the resulting dataset EXAMPLE3B has 51 records and the log confirms that the records with both FASFL = "Y" and SEXN = 1 are read and sorted. Here is our expected result!

Finally to confirm the behavior of a WHERE SAME AND statement in a PROCEDURE, a similar program is constructed using a WHERE statement followed by a WHERE SAME AND statement. The resulting dataset EXAMPLE3C has the expected 51 observations and the subsetting is confirmed in the log.

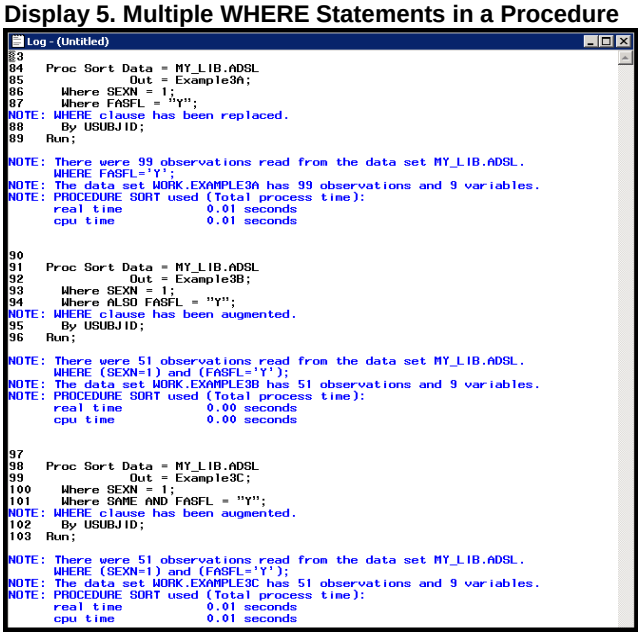
USING A WHERE STATEMENTS AND A WHERE= OPTION IN A PROCEDURE

Unlike the DATA step, in a PROCEDURE a WHERE= dataset option and a WHERE statement can be used in combination successfully.

In Display 5, the programming examples want to sort a subset our sample ADSL dataset for the Full Analysis Set (FASFL = "Y") and Male Subjects (SEXN = 1) and then sort a subset of subjects in the Full Analysis Set (FASFL = "Y"), Males (SEXN = 1), and with Age less than 65 years (AGEGR1N = 1).

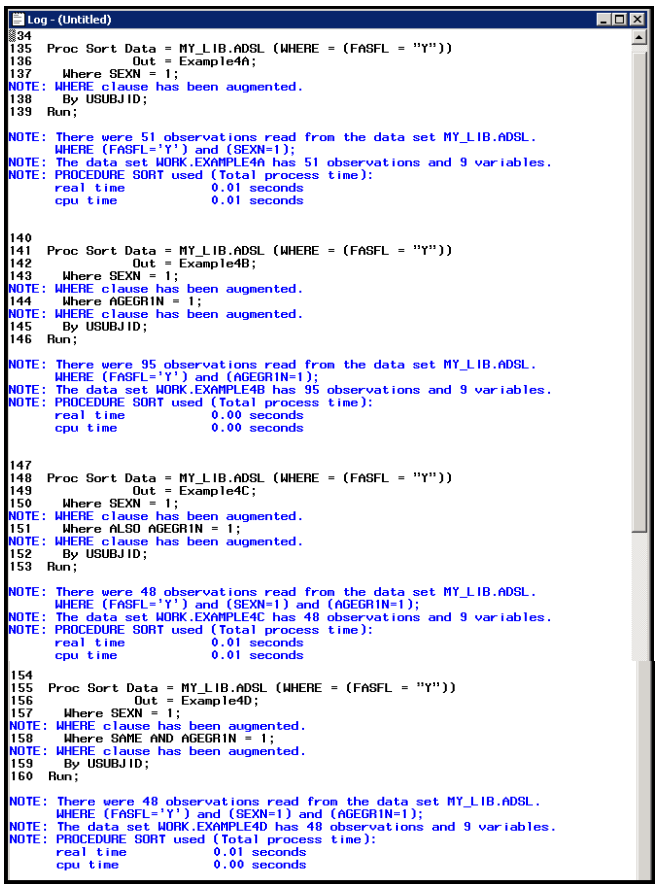
The first programming example shows a PROC SORT with a WHERE= dataset option, *WHERE = (FASFL = "Y")*, and a WHERE Statement, *WHERE SEXN = 1.* Reviewing the log shows that this is a successful programming strategy in a PROCEDURE. The resulting dataset EXAMPLE4A has the expected 51 observations

Display 5. Multiple WHERE Statements in a Procedure



```
Log - (Untitled)
83
84   Proc Sort Data = MY_LIB.ADSL
85               Out = Example3A;
86     Where SEXN = 1;
87     Where FASFL = "Y";
NOTE: WHERE clause has been replaced.
88     By USUBJID;
89   Run;

NOTE: There were 99 observations read from the data set MY_LIB.ADSL.
      WHERE FASFL='Y';
NOTE: The data set WORK.EXAMPLE3A has 99 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


90
91   Proc Sort Data = MY_LIB.ADSL
92               Out = Example3B;
93     Where SEXN = 1;
94     Where ALSO FASFL = "Y";
NOTE: WHERE clause has been augmented.
95     By USUBJID;
96   Run;

NOTE: There were 51 observations read from the data set MY_LIB.ADSL.
      WHERE (SEXN=1) and (FASFL='Y');
NOTE: The data set WORK.EXAMPLE3B has 51 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


97
98   Proc Sort Data = MY_LIB.ADSL
99               Out = Example3C;
100    Where SEXN = 1;
101    Where SAME AND FASFL = "Y";
NOTE: WHERE clause has been augmented.
102    By USUBJID;
103  Run;

NOTE: There were 51 observations read from the data set MY_LIB.ADSL.
      WHERE (SEXN=1) and (FASFL='Y');
NOTE: The data set WORK.EXAMPLE3C has 51 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Display 6. WHERE= Option and WHERE Statement in a Procedure



```
Log - (Untitled)
134
135  Proc Sort Data = MY_LIB.ADSL (WHERE = (FASFL = "Y"))
136              Out = Example4A;
137    Where SEXN = 1;
NOTE: WHERE clause has been augmented.
138    By USUBJID;
139  Run;

NOTE: There were 51 observations read from the data set MY_LIB.ADSL.
      WHERE (FASFL='Y') and (SEXN=1);
NOTE: The data set WORK.EXAMPLE4A has 51 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


140
141  Proc Sort Data = MY_LIB.ADSL (WHERE = (FASFL = "Y"))
142              Out = Example4B;
143    Where SEXN = 1;
NOTE: WHERE clause has been augmented.
144    Where AGEGR1N = 1;
NOTE: WHERE clause has been augmented.
145    By USUBJID;
146  Run;

NOTE: There were 95 observations read from the data set MY_LIB.ADSL.
      WHERE (FASFL='Y') and (AGEGR1N=1);
NOTE: The data set WORK.EXAMPLE4B has 95 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


147
148  Proc Sort Data = MY_LIB.ADSL (WHERE = (FASFL = "Y"))
149              Out = Example4C;
150    Where SEXN = 1;
NOTE: WHERE clause has been augmented.
151    Where ALSO AGEGR1N = 1;
NOTE: WHERE clause has been augmented.
152    By USUBJID;
153  Run;

NOTE: There were 48 observations read from the data set MY_LIB.ADSL.
      WHERE (FASFL='Y') and (SEXN=1) and (AGEGR1N=1);
NOTE: The data set WORK.EXAMPLE4C has 48 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds

154
155  Proc Sort Data = MY_LIB.ADSL (WHERE = (FASFL = "Y"))
156              Out = Example4D;
157    Where SEXN = 1;
NOTE: WHERE clause has been augmented.
158    Where SAME AND AGEGR1N = 1;
NOTE: WHERE clause has been augmented.
159    By USUBJID;
160  Run;

NOTE: There were 48 observations read from the data set MY_LIB.ADSL.
      WHERE (FASFL='Y') and (SEXN=1) and (AGEGR1N=1);
NOTE: The data set WORK.EXAMPLE4D has 48 observations and 9 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

and the log NOTES confirm that records with both FASFL = “Y” **AND** SEXN = 1 were processed.

The second programming example uses multiple WHERE statements with a WHERE= dataset option. Recall that previously we showed that a second WHERE statement is replaced in a PROC SORT. However, when used in combination with a WHERE= dataset option, the log returns a different log message around the WHERE statements. When reviewing the log around EXAMPLE4B, we see the WHERE clause was augmented. But, was it really? At further examination, we see 95 observations rather than the expected 48 observations. Moreover, the later log NOTES contradict the previous ones and describes the subsetting as FASFL = ”Y” and AGEGR1N = 1. The SEXN = 1 was not used in the subsetting. In this case, the SAS® log NOTES are false. The WHERE clause was not augmented it was replaced!

The last two programming examples show that by using a WHERE ALSO or a WHERE SAME AND statement will execute and correctly return our expected results with 48 observations that have FASFL = “Y” and SEXN = 1 and AGEGR1N = 1.

CONCLUSION

In this paper, the different behaviors of combining WHERE statements and WHERE= dataset options has been demonstrated. Further, misleading SAS® log messages of *NOTE: Where clause has been augmented* has been exposed. Although, WHERE statements and WHERE= dataset options are a staple programming, the simplicity of the syntax should not be overlooked in log reviews. The programmer needs to fully understand how these statements and options in combination are being processed in DATA steps and PROCEDURES. It is critical to review logs for expected results including confirming any WHERE clause augmentation or replacement and number of observations in the resulting output dataset.

ACKNOWLEDGMENTS

First, I would like to thank my Lord, Jesus Christ. It is through Him that I find my strength, patience, and resolve.

Next, I would like to thank my family: my encouraging husband, Justin, and my kids (Hope, Faith, Justin, Danny, Charity, and Paul) who are my never-ending source of happiness.

RECOMMENDED READING

- Base SAS® Language Reference
- Base SAS® Procedures Guide

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Britney Gilbert
Juniper Tree Consulting, LLC
Britney.Gilbert@JuniperTreeConsulting.com
www.JuniperTreeConsulting.com
@JuniperTree19

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

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