Paper S1-11-2013
SAS®: Tips and Tricks
Audrey Yeo, Aviva USA, West Des Moines, Iowa

Abstract
Using SAS® in the work environment is different from using SAS in the educational environment. Data in the work environment is neither as perfect nor as simple as data in the educational environment. This paper will highlight some tips and tricks that will help a new SAS user, whether a recent graduate or just new to SAS, deal with some common but challenging problems.

Introduction
While working as an Actuarial Technician, I was required to do a lot of data cleansing, data merging, and also data reporting using SAS®. While doing all these tasks, I encounter a lot of "How do I do this" questions. Hence, I started to compile notes, tips, and tricks that I find useful along the way. In this paper, I'm going to present tips on how to create reports that require a "numeric" variable to be sorted accordingly. I'm also going to present some merging tips and conclude the paper with some information on many to many merge.

Tips and Tricks #1
Working in an insurance industry, our reports produced are always dealing with numbers. However, some of these numbers are actually numerical values stored in character formats, especially when we are dealing with ranges. Thus, when creating reports using PROC REPORT, we see that the outputs for these "numerical" variables are not sorted accordingly in the numerical format. This is because they are actually character variables, and are sorted accordingly in the character format. For example, if you have a column with data 1-10, 31-35, 21-30, and 111-120, the output for the character variables in a PROC REPORT will be 1-10, 111-120, 21-30, and 31-35. Below is a trick that we can use so that the output will be reported as 1-10, 21-30, 31-35, and 111-120.

Original Code1:
```
DATA growth1;
  input blocks $7. trtmnt growth;
  datalines;
1-10    1 7.84
31-35   1 8.69
21-30   2 6.69
35-41   2 6.95
41-46   2 6.41
47-48   3 6.43
56-60   3 6.61
111-120 3 6.61
121-145 3 6.61
410-460 2 6.41
210-250 2 6.41
395-405 3 6.79
;
run;
```

```
PROC REPORT data = growth1 missing headline nowd out = growth3;
      column blocks trtmnt growth;
      define blocks / group;
      define trtmnt / group;
      define growth / analysis sum;
run;
```

Output (Original Code1):

blocks	trtmnt	growth
1-10	1	7.84
111-120	3	6.61
121-145	3	6.61
21-30	2	6.69
210-250	2	6.41
31-35	1	8.69
35-41	2	6.95
395-405	3	6.79
41-46	2	6.41
410-460	2	6.41
47-48	3	6.43
56-60	3	6.61

As you can see, the report created has the blocks column sorted in a way that is not ideal. We can use PROC FORMAT to fix this issue, as shown in the code below.

Fixed Code1:

```
PROC FORMAT;
value $blockfmt (notsorted)
                  1 = '1-10'
                  2 = '21-30'
                  3 = '31-35'
                  4 = '35-41'
                  5 = '41-46'
                  6 = '47-48'
                  7 = '56-60'
                  8 = '111-120'
                  9 = '121-145'
                  10 = '210-250'
                  11 = '395-405'
                  12 = '410-460';
run;

DATA growth2;
  input block $7. trtmnt growth;
  datalines;
1-10    1 7.84
31-35   1 8.69
21-30   2 6.69
35-41   2 6.95
41-46   2 6.41
47-48   3 6.43
56-60   3 6.61
111-120 3 6.61
121-145 3 6.61
410-460 2 6.41
210-250 2 6.41
395-405 3 6.79
;
run;
```

```
PROC REPORT data = growth2 missing headline nowd completerows out = growth4;
      column block trtmnt growth;
      define block / group preloadfmt order = data format = $blockfmt.;
      define trtmnt / analysis;
      define growth / analysis sum;
run;
```

Output (Fixed Code1):

block	trtmnt	growth
1-10	1	7.84
21-30	2	6.69
31-35	1	8.69
35-41	2	6.95
41-46	2	6.41
47-48	3	6.43
56-60	3	6.61
111-120	3	6.61
121-145	3	6.61
210-250	2	6.41
395-405	3	6.79
410-460	2	6.41

The output above shows that the variables are sorted accordingly. Besides PROC FORMAT, we could also use a PROC SORT with SORTSEQ=LINGUISTIC(NUMERIC_COLLATION=ON) as shown in Fixed Code2.

Fixed Code2:

```
DATA growth1;
  input blocks $7. trtmnt growth ;
  datalines;
1-10    1 7.84
31-35   1 8.69
21-30   2 6.69
35-41   2 6.95
41-46   2 6.41
47-48   3 6.43
56-60   3 6.61
111-120 3 6.61
121-145 3 6.61
410-460 2 6.41
210-250 2 6.41
395-405 3 6.79
;
run;

PROC SORT data=growth1 sortseq=linguistic(numeric_collation=on) out=growth2;
by blocks; run;

PROC REPORT data = growth2 missing headline nowindows out = growth3;
      column blocks trtmnt growth;
      define blocks / group order=data;
      define trtmnt / group;
      define growth / analysis sum;
run;
```

Output (Fixed Code2):

block	trtmnt	growth
1-10	1	7.84
21-30	2	6.69
31-35	1	8.69
35-41	2	6.95
41-46	2	6.41
47-48	3	6.43
56-60	3	6.61
111-120	3	6.61
121-145	3	6.61
210-250	2	6.41
395-405	3	6.79
410-460	2	6.41

Now, let's say we add another variable into the blocks column, 461+, as shown below.

Original Code2:

```
DATA growth1;
  input blocks $7. trtmnt growth ;
  datalines;
1-10    1 7.84
31-35   1 8.69
21-30   2 6.69
35-41   2 6.95
41-46   2 6.41
47-48   3 6.43
56-60   3 6.61
111-120 3 6.61
121-145 3 6.61
410-460 2 6.41
210-250 2 6.41
395-405 3 6.79
461+    5 3.24
;
run;

PROC REPORT data = growth1 missing headline nowindows out = growth3;
      column blocks trtmnt growth;
      define blocks / group;
      define trtmnt / group;
      define growth / analysis sum;
run;
```

Output (Original Code2):

blocks	trtmnt	growth
1-10	1	7.84
111-120	3	6.61
121-145	3	6.61
21-30	2	6.69
210-250	2	6.41
31-35	1	8.69
35-41	2	6.95
395-405	3	6.79
41-46	2	6.41
410-460	2	6.41
461+	5	3.24
47-48	3	6.43
56-60	3	6.61

Like the previous Original Code1, the blocks column is not sorted ideally.

Fixed Code3:

```
PROC FORMAT;
value $blockfmt (notsorted)
      1 = '1-10'
    2 = '21-30'
      3 = '31-35'
      4 = '35-41'
      5 = '41-46'
      6 = '47-48'
      7 = '56-60'
      8 = '111-120'
      9 = '121-145'
      10 = '210-250'
      11 = '395-405'
      12 = '410-460'
      13 = '461+';
run;

DATA growth2;
  input block $7. trtmnt growth;
  datalines;
1-10    1 7.84
31-35   1 8.69
21-30   2 6.69
35-41   2 6.95
41-46   2 6.41
47-48   3 6.43
56-60   3 6.61
111-120 3 6.61
121-145 3 6.61
461+    5 3.24
410-460 2 6.41
210-250 2 6.41
395-405 3 6.79
;
run;

PROC REPORT data = growth2 missing headline nowd completerows out = growth4;
      column block trtmnt growth;
      define block / group preloadfmt order = data format = $blockfmt.;
      define trtmnt / analysis;
      define growth / analysis sum;
run;
```

Output (Fixed Code3):

block	trtmnt	growth
1-10	1	7.84
21-30	2	6.69
31-35	1	8.69
35-41	2	6.95
41-46	2	6.41
47-48	3	6.43
56-60	3	6.61
111-120	3	6.61
121-145	3	6.61
210-250	2	6.41
395-405	3	6.79
410-460	2	6.41
461+	5	3.24

Fixed Code4:

```
DATA growth1;
  input blocks $7. trtmnt growth ;
  datalines;
1-10    1 7.84
31-35   1 8.69
21-30   2 6.69
35-41   2 6.95
41-46   2 6.41
47-48   3 6.43
56-60   3 6.61
111-120 3 6.61
121-145 3 6.61
410-460 2 6.41
210-250 2 6.41
395-405 3 6.79
461+    5 3.24
;
run;

PROC SORT data=growth1 sortseq=linguistic(numeric_collation=on) out=growth2;
by blocks; run;

PROC REPORT data = growth2 missing headline nowindows out = growth3;
      column blocks trtmnt growth;
      define blocks / group order=data;
      define trtmnt / group;
      define growth / analysis sum;
run;
```

Output (Fixed Code4):

blocks	trtmnt	growth
1-10	1	7.84
21-30	2	6.69
31-35	1	8.69
35-41	2	6.95
41-46	2	6.41
47-48	3	6.43
56-60	3	6.61
111-120	3	6.61
121-145	3	6.61
210-250	2	6.41
395-405	3	6.79
410-460	2	6.41
461+	5	3.24

Using both PROC FORMAT and PROC SORT with NUMERIC_COLLATION=ON, we're able to rank the variable 461+ into the rightful position.

Tips and Tricks #2

Before we are able to create our reports, we have to make sure that we have the data required to create the reports. Many times, the data required is not contained in just one dataset/file. We have to merge several datasets/files in order to get everything that we need to create the reports.

It is important to understand that there are more than one ways to solve a problem. It depends on what you have at hand at the moment and what you think is the best way of using what you have. From here onwards, I'm going to show you the equivalents of the DATA steps and the PROC SQL merge.

Given the two datasets below, we will be going through left join merge, right join merge, inner join merge, and full join merge.

Employee:

LastName	DepartmentID
Jasper	30
Rafferty	31
Jones	33
Steinber	33
Robinson	34
Smith	34

Department:

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

1. Left Join

DATA step Merge1:

```
PROC SORT data = Employee; by DepartmentID; run;
PROC SORT data = Department; by DepartmentID; run;

DATA LeftJoin;
      merge Employee (in=a) Department (in=b);
      by DepartmentID;
      if a;
run;
```

Output (DATA step Merge1):

LastName	DepartmentID	DepartmentName
Jasper	30	
Rafferty	31	Sales
Jones	33	Engineering
Steinber	33	Engineering
Robinson	34	Clerical
Smith	34	Clerical

PROC SQL Merge1:

```
PROC SQL;
      create table LeftJoin2 as
            select e.lastname, e.departmentID, d.departmentname
            from employee as e left join department as d
            on e.departmentID = d.departmentID;
quit;
```

Output (PROC SQL Merge1):

LastName	DepartmentID	DepartmentName
Jasper	30	
Rafferty	31	Sales
Jones	33	Engineering
Steinber	33	Engineering
Robinson	34	Clerical
Smith	34	Clerical

PROC SQL left join is equivalent to DATA step merge if a.

2. Right Join

DATA step Merge2:

```
PROC SORT data = Employee; by DepartmentID; run;
PROC SORT data = Department; by DepartmentID; run;

DATA RightJoin;
      merge Employee (in=a) Department (in=b);
      by DepartmentID;
      if b;
run;
```

Output (DATA step Merge2):

LastName	DepartmentID	DepartmentName
Rafferty	31	Sales
Jones	33	Engineering
Steinber	33	Engineering
Robinson	34	Clerical
Smith	34	Clerical
	35	Marketing

PROC SQL Merge2:

```
PROC SQL;
      create table RightJoin2 as
            select e.lastname, d.departmentID, d.departmentname
            from employee as e right join department as d
            on e.departmentID = d.departmentID;
quit;
```

Output (PROC SQL Merge2):

LastName	DepartmentID	DepartmentName
Rafferty	31	Sales
Jones	33	Engineering
Steinber	33	Engineering
Robinson	34	Clerical
Smith	34	Clerical
	35	Marketing

PROC SQL right join is equivalent to DATA step merge if b.

3. Inner Join

DATA step Merge3:

```
PROC SORT data = Employee; by DepartmentID; run;
PROC SORT data = Department; by DepartmentID; run;

DATA InnerJoin;
      merge Employee (in=a) Department (in=b);
      by DepartmentID;
      if a and b;
run;
```

Output (DATA step Merge3):

LastName	DepartmentID	DepartmentName
Rafferty	31	Sales
Jones	33	Engineering
Steinber	33	Engineering
Robinson	34	Clerical
Smith	34	Clerical

PROC SQL Merge3:

```
PROC SQL;
      create table InnerJoin2 as
            select e.lastname, e.departmentID, d.departmentname
            from employee as e inner join department as d
            on e.departmentID = d.departmentID;
quit;
```

Output (PROC SQL Merge3):

LastName	DepartmentID	DepartmentName
Rafferty	31	Sales
Jones	33	Engineering
Steinber	33	Engineering
Robinson	34	Clerical
Smith	34	Clerical

PROC SQL inner join is equivalent to DATA step merge if a and b.

4. Full Join

DATA step Merge4:

```
PROC SORT data = Employee; by DepartmentID; run;
PROC SORT data = Department; by DepartmentID; run;

DATA FullJoin;
      merge Employee (in=a) Department (in=b);
      by DepartmentID;
      if a or b;
run;
```

Output (DATA step Merge4):

LastName	DepartmentID	DepartmentName
Jasper	30	
Rafferty	31	Sales
Jones	33	Engineering
Steinber	33	Engineering
Robinson	34	Clerical
Smith	34	Clerical
	35	Marketing

PROC SQL Merge4:

```
PROC SQL;
      create table FullJoin2 as
            select e.lastname, e.departmentID, d.departmentname
            from employee as e full join department as d
            on e.departmentID = d.departmentID;
quit;
```

Output (PROC SQL Merge4):

LastName	DepartmentID	DepartmentName
Jasper	30	
Rafferty	31	Sales
Jones	33	Engineering
Steinber	33	Engineering
Robinson	34	Clerical
Smith	34	Clerical
	35	Marketing

PROC SQL full join is equivalent to DATA step merge if a or b.

The four joins that we went through above works only if it is a one to one merge, one to many merge, or many to one merge. While we can also use DATA step to create a many to many merge, it is a more complicated code. PROC SQL, on the other hand, provides a simpler solution. Below is an example for a many to many merge using PROC SQL.

5. Many to many join

Tablea:

userid	group
userid9	groupa
userid4	groupa
userid8	groupb
userid2	groupb

Tableb:

profile	group
profile99	groupa
profile97	groupa
profile99	groupb
profile97	groupb

Desired output needs to be:

userid	group	profile
userid2	groupb	profile97
userid2	groupb	profile99
userid4	groupa	profile97
userid4	groupa	profile99
userid8	groupb	profile97
userid8	groupb	profile99
userid9	groupa	profile97
userid9	groupa	profile99

We were given two tables as shown on the left above and we would like to merge them to create a third table (on the right). This is a many to many merge.

Many to Many Code1:

```
proc sql;
      create table manymany as
            select a.userid, a.group, b.profile
            from tablea as a, tableb as b
            where a.group = b.group
            order by a.userid, a.group, b.profile;
quit;
```

Using PROC SQL, we are able to merge the dataset into our desired output. Let us show you another example of the many to many merge using the full join function.

Many to Many Code2:

```
PROC SQL;
      create table manymany2 as
            select a.userid, a.group, b.profile
            from tablea as a full join tableb as b
            on a.group = b.group
            order by a.userid, a.group, b.profile;
quit;
```

Using a PROC SQL with a full join function, we are able to get our desired output. Comparing the Many to Many Code2 example to the Many to Many Code1 example, you can see that we did not use the full join function in the first example (instead, we used where a.group = b.group), but we are still able to get the many to many merge output.

Conclusion

To summarize, these are some tips and trick that I find very useful to me as a data technician. Especially tips and trick #2 where we go into detail the merges using DATA steps and PROC SQL.

Contact Information

Name: Audrey Yeo
Enterprise: Aviva USA
Address: 7700 Mills Civic Parkway
City, State, ZIP: West Des Moines, IA 50316
Work Phone: 515-342-3759
E-mail: audrey.yeo@avivausa.com

Trademark Citations