

Paper SD13
Smart SAS® Tricks
Monal Kohli, Smith Hanley Consulting, Parsippany, NJ

ABSTRACT

Are you ever struck with problems like how to count the number of occurrences of a character in a string without writing a lot of code, how to check the contents of a format, how to output contents of a library to a file, what to do when formats are not resolved, how to deal with difficult dates and how to choose *retain* Vs *lag*. If you want to learn some pragmatic SAS tricks for elegant SAS programming, this paper is surely for you.

INTRODUCTION

There are times when we all are stuck with problems which we wish had simpler solutions. We'll discuss some of those problems in this paper. To start with we will see how COUNTC function works, it will help us get the count of occurrences of a character in a string. We will look into how to use "PROC FORMAT" efficiently. This will help us to find solutions for copying formats easily, looking at the contents of formats in a jiffy. We will also look at the command for copying the contents of a directory to a file for using it later in any SAS program. Finally we'll see the scenarios for usage of *retain* Vs *lag* function. So let's start learning!!!!

TRICK I : COUNTING THE OCCURENCES OF A CHARACTER

This function "COUNTC" which is new to SAS 9.0 will help us accomplish this task of counting the occurrences without writing lines of code. This function basically counts the number of specific characters that either appear or do not appear within a character string that you specify. Let's look at how this function works:

Below is the code for implementation of function COUNTC:

```
data final;
length id $5 adverse_events $40;
input id $ adverse_events $ ;
count_adverse_events=countc(adverse_events,',',
');
datalines;
1001 cough,cold,migraine
1002 fatigue,chills
1003 sinus,insomnia,wheezing,anxiety
;
run;
```

This is how the dataset "final" looks:

id	adverse_events	count_adverse_events
1001	cough, cold, migraine	2
1002	fatigue, chills	1
1003	sinus, insomnia, wheezing, anxiety	4

TRICK II : SOLUTION TO "FORMAT NOT RESOLVED"

Have you ever encountered this problem when you are working with datasets which you didn't create and they have predefined formats and you have no clue what those formats are doing? Let's look at a quick way for figuring out the contents of a predefined format.

If you are working with a character format you would need to specify a "\$" symbol in front of the format name in the SELECT statement.

```
proc format
lib=work.formats fmtlib;
select $pt;
run ;
```

If you are working with a numeric format you would need to specify a "\$" symbol in front of the format name in the SELECT statement.

```
proc format lib=work.formats
fmtlib;
select pt;
run ;
```

Many times we are struck with situations when we are trying to define a format and we have lots of data. Now if that data is present in a dataset, the task gets much easier. How??????????????? Let's look at the solution to this problem.

Suppose we have a dataset "pt_val " which has all the values for which we want to create a format. That is we want to define a format which will replace pt_term in a dataset with the respective value.

pt_term	value
Lymphadenopathy	3
Tachycardia	5
Cerumen impaction	7
Ear disorder	8
Ear pain	9
Photophobia	11
Abdominal discomfort	13
Abdominal pain	14
Abdominal pain upper	15
Abnormal faeces	16
Anal skin tags	17
Aphthous stomatitis	18
Chapped lips	19
Diarrhoea	20
Dry mouth	21
Dyspepsia	22
Glossodynia	23
Haemorrhoids	24
Lip ulceration	25
Nausea	26
Oral pain	27
Stomach discomfort	28
Tongue coated	29
Tongue disorder	30

Let's look at the steps now.

- First we need to rearrange the dataset to have a format name, start and end variables. These are required to define a format successfully.

```
data fmt;
retain fmtname "$pt_term";
set pt_val ;
```

```

start = pt_term;
label = value;
keep start label fmtname ;
run;

```

- Second we just use the dataset "fmt" to output the format pt_term like this :

```

proc format cntlin = fmt fmtlib;
select $pt_term;
run;

```

- Finally let's look at the contents of the format after it is outputted to work.formats library. Below is a snapshot of The format output.

FORMAT NAME: \$PT_TERM LENGTH: 2 NUMBER OF VALUES: 31		
MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 2 FUZZ: 0		
START	END	LABEL (VER. 8.2 26APR2007:10:43:00)
Abdominal discom	Abdominal discom	7
Abdominal pain	Abdominal pain	8
Abdominal pain u	Abdominal pain u	9
Abnormal faeces	Abnormal faeces	10
Anal skin tags	Anal skin tags	11
Aphthous stomati,	Aphthous stomati,	12
Catheter site pa	Catheter site pa	27
Cerumen impaction	Cerumen impaction	3
Chapped lips	Chapped lips	13
Chills	Chills	28

TRICK III: HOW TO HANDLE TRICKY DATES

What do you do when you have a month as a character like "JAN"? How do you create a correct date in a situation like this. Let's look an example.

DAY	MONTH	YEAR	ID
11	AUG	2006	7017
06	MAR	2006	5002
18	APR	2006	6002
16	FEB	2006	7002
03	MAY	2006	7010

In the above data all month, day and year are character variables. Now how do we convert the month to a proper numeric format so that MDY function works properly. The problem here is we have a month which is a character and it does not align with any valid character date format. So basically we need to change month to "month+year" in order to use "mmyy5" format to convert the character to numeric. For this just add "01" at the end of the month where "01" signifies the year. It could be any valid year suffix. So let's look at the solution now.

To get the month in a proper format we use this statement:

```

month_new=month(INPUT(month!!'01',monyy5.));

```

To get the complete date we use this statement::

```
date=mdy(month(INPUT(month!!'01',monyy5.)),INPUT(day,$3.),INPUT(year,$4.));
```

This is how the final data looks with the correct dates.

DAY	MONTH	YEAR	ID	month_new	date
11	AUG	2006	7017	8	8/11/2006
06	MAR	2006	5002	3	3/6/2006
18	APR	2006	6002	4	4/18/2006
16	FEB	2006	7002	2	2/16/2006
03	MAY	2006	7010	5	5/3/2006

TRICK IV: RETAIN VS LAG

Are you ever struck in a dilemma of when to use RETAIN and when to use LAG. We will see the difference by actual examples and not theory.

Let's use RETAIN first!!

I have a dataset which has subject's data with time points and other lab data. But some of the rows are missing the date of birth. Our task is to populate all the subjects with their respective date of births so that none of the observations are missing date of birth. But one of the visits surely has date of birth in it and we will use that date to populate the other missing rows.

This is how the dataset "dob" looks now:

subjid	timepoint	dob
1001	visit1	7Apr77
1001	visit2	
1001	visit3	
1001	visit4	
1002	visit1	
1002	visit2	8May78
1002	visit3	
1002	visit4	

This is a perfect scenario for using RETAIN statement. Lets try to use RETAIN statement in the above dataset.

```
proc sort data=dob; by subjid;run;
data dob;
set dob;
by subjid;
retain date;
if first.subjid then date=dob;
run;
```

Let's look at the dataset now:

subjid	timepoint	dob	date
1001	visit1	4/7/1977	4/7/1977
1001	visit2		4/7/1977
1001	visit3		4/7/1977
1001	visit4		4/7/1977
1002	visit1		
1002	visit2	5/8/1978	
1002	visit3		
1002	visit4		

Oops what happened here ?????????? Subject “1001” got the right dates but subject “1002” totally lost it. Do we know the reason why??? The observation which is used to retain the date of birth, the first observation per subject, in this case “1002” is missing the value of dob. When we use a RETAIN statement, the PDV actually stores the value of DOB for all the observations with the same subjid, but if the first observation itself is missing the date it will result in all observations having missing dates. To fix this problem we have to ensure that the observation at the top should be the one with the populated dob in it. Let's fix this problem.

```
proc sort data=dob;by subjid
descending dob;run;
data dob;
set dob;
by subjid descending dob;
retain date;
if first.subjid then date=dob;
if dob eq . then dob=date;
run;
```

This is how the dataset dob looks after implementing the above code:

subjid	timepoint	dob	date
1001	visit1	4/7/1977	4/7/1977
1001	visit2	4/7/1977	4/7/1977
1001	visit3	4/7/1977	4/7/1977
1001	visit4	4/7/1977	4/7/1977
1002	visit2	5/8/1978	5/8/1978
1002	visit1	5/8/1978	5/8/1978
1002	visit3	5/8/1978	5/8/1978
1002	visit4	5/8/1978	5/8/1978

So only use RETAIN when you do not want the PDV to reset the values to missing at the start of each iteration in the datastep.

Let's turn now to the LAG Function.

Let's look at an example which will explain when and why we should use the LAG function. We have a dataset which has the recorded weights for two subjects taken at visit1,visit2,visit3 and visit4. The weight is recorded in visits 1 and 3 .The task here is to carry over visit1 weight to visit2 and visit3 weight to visit4.

The dataset “weight” is illustrated in the table that follows:

subjid	timepoint	weight
1001	visit1	145
1001	visit2	
1001	visit3	165
1001	visit4	
1002	visit1	165
1002	visit2	
1002	visit3	165
1002	visit4	168

This is a perfect example to use LAG function. Let' see how.

```
proc sort data=weight ;by subjid timepoint;
run;
data weight;
set weight;
wt=lag(weight);
if weight eq . then weight=wt;
run;
```

This is how the dataset looks after applying the LAG function.

subjid	timepoint	weight	wt
1001	visit1	145	
1001	visit2	145	145
1001	visit3	165	
1001	visit4	165	165
1002	visit1	165	
1002	visit2	165	165
1002	visit3	165	
1002	visit4	168	165

LAG should only be used when you do want the previous observation's value carried over to the next observation.

CONCLUSION

This paper discussed some handy techniques to overcome problems associated with strings, formats and data retention in PDV. We discussed the new COUNTC function and techniques for using the PROC format efficiently. I Hope the tips discussed in this paper can prove helpful to everyone.

CONTACT INFORMATION

Author Name:	Monal Kohli
Company:	Smith Hanley Consulting
Address:	114 Emily Pl
City state ZIP	Parsippany NJ 07054
Work Phone:	5168155727
Email:	monal.kohli@gmail.com

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.