Paper 3963-2019

Dynamically Assigning Length to Transposed Variables

Ethan Ritchie, Sarah Cannon

ABSTRACT

When using PROC TRANSPOSE to transform "narrow" data (a single subject variable stored in many rows) into "wide" data (one row with subject values stored as distinct variables), the length of the original subject variable is used as the length for each new variable. As a result, variable lengths may be much larger than necessary. This paper will demonstrate a method to automatically assign the smallest length necessary to each variable once the "narrow" data is transposed into "wide" data. The method is currently used on large sample survey data that is updated daily and accounts for the possibility that the optimal length of a variable may change from day to day.

INTRODUCTION

There are many applications where data are collected and stored in a “narrow” format and must be transposed before being used for analysis. The variety of values may change over time and the number of variables may also change over time. Survey data collections are a prime example of this since not all items are necessarily administered to each respondent and the variety of responses is always increasing. In this paper you will learn how to assign lengths to transposed character variables without being hindered by the number of variables transposed or changes in the maximum length of variables over time.

EXAMINING THE PROCESS

A simple “narrow” dataset will include an identification field, a variable name field, and a variable value field. In the context of survey data collection, these fields correspond to the survey respondent, the survey question, and the respondent’s answer to that question, respectively. PROC PRINT output for example data are displayed below in Output 1 and PROC CONTENTS output for the data is shown in Output 2 below.

Output 1 – PROC PRINT output for raw data

ID	Variable	Value
1001	First	Joseph
1001	Last	Mattheson
1001	Sex	M
1001	City	Charlotte
1001	State	NC
1002	First	Ian
1002	Last	Cummings
1002	Sex	M
1002	City	Ithaca
1002	State	NY
1003	First	Jessica
1003	Last	Porter
1003	Sex	F
1003	City	Chancellorsville
1003	State	VA
1004	First	Amanda
1004	Last	Wright
1004	Sex	F
1004	City	Provo
1004	State	UT

Output 2 – PROC CONTENTS output for raw data

Variables in Creation Order			
#	Variable	Type	Len
1	ID	Num	8
2	Variable	Char	50
3	Value	Char	200

TRANSPOSING THE DATA

While storing data in a "narrow" format is the most flexible way to store data during collection, it is not typically the desired format for analytical purposes. These data need to be transposed into a "wide" dataset. The name of the dataset shown in Output 1 is "step1", so the SAS® code to transpose the data would be:

```
proc transpose data=step1 out=step2 (drop=_NAME_);
   id Variable; by ID; var value;
run;
```

The transposed data are shown below in Output 3. The PROC CONTENTS output shown in Output 4 below indicates that each transposed variable has a length of 200. In large scale data collections, this is problematic since this results in larger files than necessary. An efficient and dynamic method is needed to reduce the lengths of these variables.

Output 3 – PROC PRINT output for transposed data

ID	First	Last	Sex	City	State
1001	Joseph	Mattheson	M	Charlotte	NC
1002	Ian	Cummings	M	Ithaca	NY
1003	Jessica	Porter	F	Chancellorsville	VA
1004	Amanda	Wright	F	Provo	UT

Output 4 – PROC CONTENTS output for transposed data

Variables in Creation Order			
#	Variable	Type	Len
1	ID	Num	8
2	First	Char	200
3	Last	Char	200
4	Sex	Char	200
5	City	Char	200
6	State	Char	200

GENERATING A LIST OF VARIABLE NAMES

The first step required after transposing the data is to generate a macro variable containing the list of variable names (not including the ID variable) and a corresponding macro variable that contains the number of variables included in the list. The following SAS code will assign the list of variables to a macro variable named “varNames” and the number of listed variables to another macro variable called “numVars”.

```
proc contents data=step2 out=varNames noprint order=varnum; run;

%let varNames = ; %let numVars = ;
data _NULL_;
   length names $20000;
   set varNames end=finally;
   retain names '' counter 0;
   if Type = 2 then do;
      names = catx(' ',names,NAME);
      counter + 1;
   end;
   if finally then do;
      call symput('varNames',names);
      call symput('numVars',counter);
   end;
run;
```

The SAS code first uses PROC CONTENTS to create a dataset containing a list of variable names contained in the transposed data. The following data step iterates through the list and adds each name to a single string variable that is retained until the last observation. A count of the number of variable names added to the string variable is also maintained. Variables added to the list are limited to character variables (TYPE = 2) since those are the only variables that need reduced length. On the last observation, CALL SYMPUT is used to assign the value of the retained string variable to &varNames and the corresponding variable count to &numVars. The values assigned by the SAS code to the two macro variables in this example are as follows:

&varNames = First Last Sex City State

&numVars = 5

These two macro variables will be used in the next step of the process which determines the maximum length for each variable.

GENERATING FORMATTING CODE TO ASSIGN LENGTH

The second step required is determining the maximum length of each character variable. The following code determines the maximum length for each variable and creates an additional macro variable that contains a formatting statement to be used in a subsequent step:

```
%let formatCode = ;
data _NULL_;
   length formatCode $20000;
   set step2 end=finally;
   array names(&numVars) &varNames;
   array maxLen(&numVars) _TEMPORARY_;
   do i=1 to &numVars;
      if _N_ = 1 then maxLen(i) = 2;
      if length(names(i)) > maxLen(i) then maxLen(i) = length(names(i));
   end;
   if finally then do;
      do j=1 to &numVars;
         convert = input(strip(maxLen(j)),$5.);
         formatCode = catx(' ',formatCode,vname(names(j)),cats('$',convert,'.'));
      end;
      call symput('formatCode',formatCode);
   end;
run;
```

The macro variable &varNames created earlier is used to define the elements of an array called “names”. The macro variable &numVars is used to define the number of elements in the “names” array as well as in a temporary array called “maxLen”. The temporary array is used to retain the maximum length of each variable through each iteration. A minimum length of 2 is set for each variable. On the last observation, a string variable is created that contains each variable and its new format. CALL SYMPUT is then used to assign the value of the string variable to a new macro variable called “formatCode”. The values assigned by the SAS code to the new macro variable in this example is as follows:

&formatCode = First $7. Last $9. Sex $2. City $16. State $2.

This code will be used in the next step to redefine the length of each character variable.

ASSIGNING REDUCED LENGTH

The final step is to use the previously created code to assign a reduced length to each character variable:

```
data finalStep;
   format ID 8. &formatCode;
   set step2;
run;
```

The SAS code uses the format statement that is stored in &formatCode to assign length to the character variables before setting the new dataset to the transposed data. PROC CONTENTS output (see Output 5 below) shows that the new lengths of the variables match the longest value from the raw data.

Output 5 – PROC CONTENTS output for final data

Variables in Creation Order				
#	Variable	Type	Len	Format
1	ID	Num	8	8.
2	City	Char	16	$16.
3	First	Char	7	$7.
4	Last	Char	9	$9.
5	Sex	Char	2	$2.
6	State	Char	2	$2.

CONCLUSION

The method for assigning length to transposed character variables presented in this paper truly is dynamic because it works independent of the number of variable and values transposed. The SAS code can be used on any dataset, but its value is in reducing the length of variables in datasets that have been transposed from “narrow” to “wide” datasets. A related method to convert variables solely containing numeric values to numeric formats is typically used in conjunction with the process presented in this paper, but that is outside the scope of this paper.

CONTACT INFORMATION

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

Ethan Ritchie
Sarah Cannon
Ethan.Ritchie@SarahCannon.com