Paper CT06

Code Generator to Check Truncated Values in Derived Datasets

Proma Mukherjee, Parexel International, Bengaluru, India

ABSTRACT

An important part of dataset programming validation involves handling of SAS® log messages like overwritten variables, missing values, etc. Since there is no specific log message for character variable truncations, the usual validation method is manual scanning or spot checks of datasets.

The intent of this paper is to provide a robust macro script which can be used in dataset codes to check for possible truncation of character variables. It achieves this by checking the length and value of character variables in the derived and source datasets. The required inputs are the source/derived dataset names which generate 'truncate check datasets'. These macro-generated datasets contain the character variables and their maximum length in a single observation which can be visually glanced through to check for possible truncations. This is a time-saving preliminary QC for dataset programming and can significantly improve the probability of catching truncations well ahead in the programming lifecycle.

INTRODUCTION

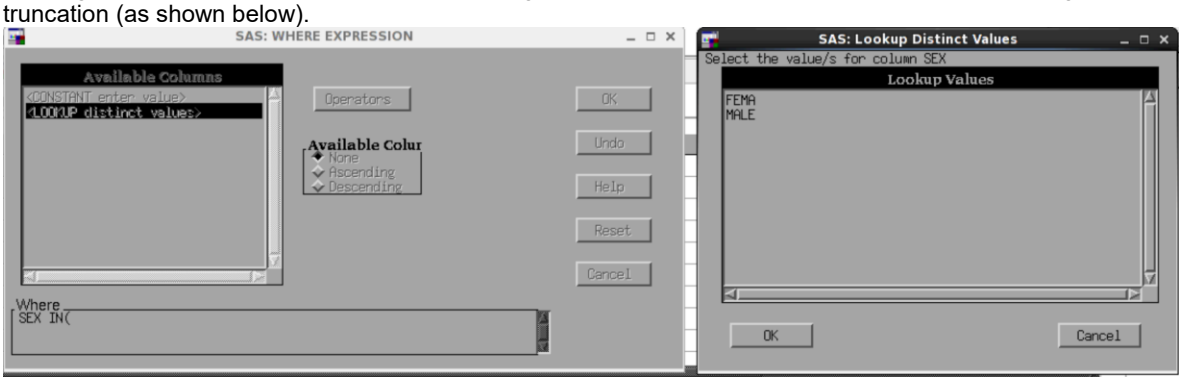
Numerous validation checks are done in a derived dataset code, both in CDISC compliant datasets as well as non-standard datasets. These checks are done while writing the code in intermediate steps and at the end. In addition to constantly checking the log and intermediate datasets for the right counts, inclusion of the needed variables, etc. there are other issues as well like missing values, uninitialized variables, repeat of by values in merge statement, etc. The SAS log takes care of most of the issues by displaying the notes for these instances. But there are scenarios such as when a character variable can get truncated without displaying any log message. These truncations can get missed out unless one is specifically looking for it or if it is clearly visible.

In real life scenarios many programmers work in multiple submissions all while using and updating the same code. In such situations it becomes a time-consuming task to check for truncations and can easily get missed out. It usually shows up when the tables/listing/figure (TLF) displays show the cropped values. Truncation can be easily missed out in multiple page listings as well and result in incorrect reporting. This paper will illustrate an improved method of checking for truncation of character values in dataset programs which can act as the first step to aid detailed validation.

CURRENT SCENARIO

Currently, validation checks for truncations usually happen in these two ways:

a. Manually scrolling through the final dataset to check if everything looks right.
b. Filtering the final dataset in SAS to check the longest value of the character variables where one might expect to find truncation (as shown below).



Both these methods can be time-consuming and error-prone when the number of variables and the observation counts are more.

The truncation check macro is a step up in that direction as it saves time and is more likely to catch truncations.

HIGH LEVEL SOLUTION

This macro checks all the character variables irrespective of their length. This takes care of smaller values which can get truncated. For example, decode of SEX variable with values 'Female' and 'Male' can get truncated to display 'Fema' and 'Male', respectively if the length is not set and the first observation has value 'Male'. For variables with longer values, the variable lengths in the source and derived datasets can be checked if it is difficult to spot truncations. Another scenario is when the length and value of variables vary from the source dataset to the derived dataset due to concatenations, substrings, etc., but that can be corroborated from within the code and its specification.

The maximum length and the corresponding value of each character variable is displayed in the resulting truncate check dataset. If there are multiple observations with the same maximum length, only the first one is displayed because is easier to scroll through a single observation for QC purpose. Also, since it is a preliminary check the objective is to spot possible truncations based on length matching and it defeats the purpose if there are different number of observations for each variable.

DETAILED SOLUTION

1. This macro script can be added at the end of the dataset code or can be run in a separate window in the same SAS session.
2. The final derived dataset and the source dataset names should be passed to the macro variable ds. In the script below, these datasets are fin and raw, respectively. Then the macro is run and the tr_chk_fin and tr_chk_raw datasets are generated.

   Note: Refrain from using the source datasets in LIBNAME.DATASETNAME format, use the first intermediate dataset instead. The reason behind this is to reduce the count of character variables by only keeping the needed variables.

PROGRAMMING STEPS

Script	Explanation

`%macro truncate_chk (ds =);`
`  proc contents data = &ds.`
`  out = trchk_&ds.2 (`
`  keep = name type memname) noprint;`
`  run;`
	▪ Beginning of the macro truncate_chk with the macro variable ds for the derived/source datasets.
	▪ Proc contents creates an intermediate dataset trchk_&ds.2 with the dataset name (memname), all the variable names (name) and type (= 1 for numerical variables and 2 for character variables).

`  data trchk_&ds.2_ (keep = type name2);`
`  set trchk_&ds.2(where = (type = 2));`
`  length name2 $40.;`
`  name2 = compress(name)||'_L';`
`  run;`
	▪ trchk_&ds.2 dataset contains a new variable name2 for all the character variables (type = 2).

`  proc transpose data = trchk_&ds.2_`
`  out = trchk_&ds.3`
`  (drop = _name_ _label_);`
`  id name2;`
`  var type;`
`  run;`
	▪ trchk_&ds.2_ dataset is transposed to create trchk_&ds.3 dataset which has numeric variables (named as character variable with '_L' added at the end) for all the corresponding character variables from trchk_&ds.2_ dataset.

`  data trchk_&ds.4 (keep = memname combo);`
`  set trchk_&ds.2 (where = (type = 2));`
`  by memname;`
`  length combo $999;`
`  retain combo;`
	▪ trchk_&ds.4 dataset is created from trchk_&ds.2 and contains a single observation with two variables: the dataset name (memname) and a new variable combo.
	▪ The variable combo contains all the character variable names concatenated together and separated by

Script	Explanation
`if first.memname then combo = compress(name); else combo = compress(catx(',', combo, name)); if last.memname; run;`	commas.

`proc sql noprint; select count(*) into :cnt from trchk_&ds.2 where type = 2;`
 	
`select combo into :cmb from trchk_&ds.4;`
 	
`create table trchk_&ds.5 as select &cmb from &ds.; quit;`
	▪ The count of character variables from trchk_&ds.2 is stored in the macro variable cnt.
	▪ The character variable combo from trchk_&ds.4 is stored in the macro variable cmb.
	▪ The dataset trchk_&ds.5 is generated from the input dataset &ds. containing all the character variables in it.

`data trchk_&ds.6; merge trchk_&ds.3 trchk_&ds.5; run;`	▪ trchk_&ds.6 dataset contains all the character variables and the corresponding numeric variables (ending with '_L').

`%let x = &cnt; proc sql noprint; select name into :nm1 - :nm&x from trchk_&ds.2 where type = 2;`
 	
`select name2 into :l1 - :l&x from trchk_&ds.2_ ; quit;`
	▪ The value from the macro variable &cnt is stored in the macro variable x.
	▪ All the character variable names from trchk_&ds.2 are being stored in the macro variables nm1 till nm&x (where x is the count of character variables).
	▪ All the numeric variable names from trchk_&ds.2_ are being stored in the macro variables l1 till l&x.

`%do i = 1 %to &x; data trchk_&ds._&i; set trchk_&ds.6; n=1; &&l&i = lengthn(&&nm&i); keep n &&l&i &&nm&i; run;`
 	
`proc sort data = trchk_&ds._&i; by n descending &&l&i; run;`
 	
`data trchk_&ds._&i._; set trchk_&ds._&i; by n descending &&l&i; if first.n; run; %end;`
	▪ The trchk_&ds._&i dataset separates each character variable and its corresponding numeric variable which captures the length of the character variable.
	▪ The trchk_&ds._&i dataset is sorted by descending length of the character values and the longest character is retained as a single observation in the trchk_&ds._&i._ dataset.

`data tr_chk_&ds.; merge %do j = 1 %to &x; trchk_&ds._&j.`	▪ All the individual datasets trchk_&ds._&j._/trchk_&ds._&i._ are merged to create

Script	Explanation

`%end;;`
`drop n;`
`run;`
`proc datasets;`
`delete trchk_&ds:;`
`quit;`
`%mend truncate_chk;`
`%truncate_chk (ds = fin);`
`%truncate_chk (ds = raw);`
…
	a single observation with all the character variables containing the longest value and their corresponding length as a numeric variable.
	▪ All the intermediate datasets are deleted.
	▪ Call the macro for the derived/source datasets. In this example, fin is the derived dataset and raw is the source dataset.

3. The next step is to glance through the character values in tr_chk_fin to look for any possible truncation. If there are any probable truncated words, check their corresponding length variable and compare them to the matching variable and length variable in tr_chk_raw dataset.

OUTPUT

Tr_chk_fin is generated for the derived dataset (with the character variables containing the value with the maximum length displayed next to their corresponding length '_L' variables):

SAS: VIEWTABLE: Work.Tr_chk_fin

File Edit View Tools Data Solutions Help

	STUDYID	STUDYID_L	USUBJID	USUBJID_L	SEX	SEX_L	REF	REF_L	VAL	VAL_L
1	123456	6	123456.987654	13	Fema	4	This is a script for comment reference for the purpose of writing a PhUSE paper.	80	This is a test for the comments value for the purpose of writing a PhUSE paper.This is a test for the comments value section for the purpose of writing this PhUSE paper. Adding text to add the length.	200

Similarly, Tr_chk_raw is generated for the source dataset:

SAS: VIEWTABLE: Work.Tr_chk_raw

File Edit View Tools Data Solutions Help

	STUDYID	STUDYID_L	USUBJID	USUBJID_L	SEX	SEX_L	REF	REF_L	VAL	VAL_L
1	123456	6	123456.987654	13	Female	6	This is a script for comment reference for the purpose of writing a PhUSE paper.	80	This is a test for the comments value for the purpose of writing a PhUSE paper.This is a test for the comments value section for the purpose of writing this PhUSE paper. Adding text to add the length.	200

In the above example, the value for variable SEX is getting truncated in tr_chk_fin, whereas it is a complete word in tr_chk_raw. We can figure this out in two ways:

a. Checking the SEX_L values in both the datasets to see if they are different.
b. Checking the actual values of the variables in both the datasets.

Similar process can be replicated for multiple source datasets.

OUT OF SCOPE

This macro is visually checking for truncations and not comparing the variables between the datasets programmatically because:

1. For the same value of a derived character variable, there can be different variables in the source dataset with the matching value. In this case, it is complex to deduce the source of the derived variable programmatically.

2. Concatenation is sometimes done on the character variables; in such cases it cannot programmatically match to any character variables in the source datasets.
3. Matching variable can be present in multiple source datasets, therefore making it complex to deduce the source.

CONCLUSION

The truncate check macro can be used to simplify and speed up the QC process of checking truncated values in the dataset codes. This will allow the programmer to quickly scan through the macro generated datasets and greatly reduce the chances of truncations being missed out, which in turn will help in maintaining the data consistency and quality.

REFERENCES

SAS 9.4 Macro Language: Reference, Fifth Edition
https://documentation.sas.com/?docsetId=mcrolref&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en

SAS 9.4 SQL Procedure User's Guide, Fourth Edition
https://documentation.sas.com/?docsetId=sqlproc&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en

CONTACT INFORMATION

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

Proma Mukherjee
Parexel International
Bengaluru
India
Email: proma.mukherjee@parexel.com
Web: www.parexel.com