

Utilizing SAS to Remedy Issues with Long Character Variables

Piotr Krzystek, ICPSR at the University of Michigan, Ann Arbor, MI

ABSTRACT

Character variables, otherwise known as string variables, contain non-numeric values and are read by a statistical program system as being in text format. The response values for character variables can be discreet and categorical in nature, such as listing single-lettered multiple-choice responses or the names of months. Other character variables can be open-ended, and each observation can possess unique values such as user-entered free text responses to questions in a survey. Some character variables can possess large lengths that may present issues for some statistical programs to read datasets properly. SAS software provides built in solutions to these issues where other statistical programs do not. The purpose of this paper is to present issues that may arise with long character variables and describe how SAS can be utilized to remedy them. The outline for this paper includes defining the types of character variables, common issues that may arise with character variables that are defined as long, and suggestions for SAS code that can be utilized to remedy these issues. The version of SAS utilized for this paper is 9.4, and users would need a basic understanding of data step syntax to edit and run the code.

INTRODUCTION

A character, or string, variable is a variable that contains non-numeric values and is read by a statistical program system as being in text format. Because of its non-numeric format, character variables can be either categorical or open-ended in their content. Character variables that are categorical or discreet tend to have fixed values for responses, examples being names of months or single-lettered multiple-choice responses (A, B, C, etc.). By contrast, open-ended character variables can possess any kind of content and usually have unique values per observations, with examples of these variables being individual responses to questions in a survey. Because the differences between values are read by a system down to punctuations and capitalizations, even minor differences will make each value be considered separate in frequency outputs.

Another aspect of character variables that can impact their versatility is their potential length. To expand on this notion, the following distinction between types of character variables will be made based on their lengths: short (12 characters or less), medium (12 to 244 characters long), and long (245 characters or more). Short character variables tend to be categorical while medium and long character variables are more often open-ended. In the context of this paper, focus will be primarily put on long character variables even though some issues discussed can be applicable to all character variables irrespective of their lengths.

POTENTIAL ISSUES WITH LONG CHARACTER VARIABLES

Numerous issues can exist with long character variables that may pose problems with further processing of given datasets. These problems can pertain to either the length of the variables or their contents. In terms of length, certain statistical packages such as Stata will not accept imported variables with lengths greater than 244 characters (creating variables within Stata though can be at maximum 2,045 characters). The maximum length for a character variable in R, while much longer than in Stata, is capped at 4,000 characters. By contrast, the maximum possible length for a character variable in SAS is 32,767 characters. This figure is the same maximum length of characters in a single cell of a Microsoft Excel file and is roughly equivalent to 18 pages worth of text.

In terms of the potential content of these variables, there can be technical (issues causing the data to not be readable) and disclosure risk factors. Technical factors would be if a long character variable contains foreign/suspicious or improperly converted characters that are not accepted by UTF-8 encoding (backtick

(') instead of the acceptable apostrophe (') mark for example). Disclosure risks would be any information that contains sensitive information about respondents' data, such as social security numbers and home addresses. Sometimes, indirect identifiers such as demographic information can also be considered a disclosure risk if the population coverage that a dataset pertains to is either small or specific [1]. Ultimately though, disclosure risk factors are at the discretion of whoever is handling the data while doing so in a manner that respects privacy. Such users can either be researchers who utilize the data in a restrictive setting but retain the full information for its usefulness or data processors who wish to make the data available publicly but need to remediate disclosive values.

Regardless of either the disclosive, technical, or length-related issues pertaining to long character variables, SAS is a program that can remedy any of these problems. In the following examples, simple procedures as well as basic functions will be utilized. The outputs from these data steps will show how easy it is to update long character variables and make the data further processable.

REMEDYING ISSUES WITH LONG CHARACTER VARIABLES

IMPORTING THE DATA

The first step in remedying problematic long character variables in SAS is by importing a dataset via PROC IMPORT. For this example, a SAV file type is being used. The syntax for the data step is as follows:

```
proc import out = <converted datafile name>
datafile = <filepath specifying dataset>
dbms = SAV
replace;
run;
```

The OUT statement specifies a working datafile name within a SAS program while the DATAFILE statement specifies the filepath of the dataset that is being imported. Next, the DBMS statement specifies the file type of the original dataset and REPLACE specifies that any preexisting working datafile of the same name that already exists will be overwritten.

EDITING VALUES WITH IF AND THEN FUNCTIONS

Once the data has been imported into SAS, the next step is to create data steps that utilize IF and THEN functions. In the following example, ID is a unique identifier variable associated with an observation, OLDVAR is the original version of a long character variable that needs to be updated, and NEWVAR serves as a temporary processing version of OLDVAR (a later data step example will show that the variable will be renamed back to OLDVAR):

```
data <processing datafile name>;
set <converted datafile name>;
IF ID = 'ABC123' THEN NEWVAR = '[TEXT > 244 CHARACTERS IN LENGTH]';
IF ID = 'DEF456' THEN NEWVAR = '[TEXT > 244 CHARACTERS IN LENGTH]';
IF ID = 'GHI789' THEN NEWVAR = '[TEXT > 244 CHARACTERS IN LENGTH]';
IF NEWVAR = '' THEN NEWVAR = OLDVAR;
run;
```

While this line of code may appear simple, it does perform a couple important processes. The first is the THEN function's specification of NEWVAR, which creates a brand-new variable. Depending on how long the first value for NEWVAR specified in the data step is listed, that value also establishes the length and format type for the new variable. The format type designation is done automatically depending on the type of characters that are included in the value, as string values will make the variable read by the system as non-numeric (it should be noted that the maximum length for numeric variables in SAS is 8 bytes or 16 digits). Lastly, the final line (IF NEWVAR = "" THEN NEWVAR = OLDVAR) copies original values in OLDVAR to what would have been empty values in NEWVAR, thus making NEWVAR a processed

version of OLDVAR. It is important when creating the data step that the longest value in a variable is listed first even if it is not in the order of a given dataset, as subsequent values specified may end up truncated in the output. From there, users can edit each value to either replace foreign characters or mask disclosive information at their own preference.

If disclosure mitigation requires masking an entire variable (also known as blanket masking), then the following syntax can be utilized:

```
IF OLDVAR = '' THEN NEWVAR = OLDVAR;
ELSE NEWVAR = '[MASKED VALUE]';
```

In the example above, all non-blank values in OLDVAR are replaced with a general masking value. It should be noted though that there can be numerous ways to blanket mask a variable, such as creating NEWVAR with all values being '[MASKED VALUE]' and deleting / replacing OLDVAR. Aside from blanket masking disclosive information, editing long character variables on a large scale can be done with simple functions that do not require listing out each problematic value per observation.

ADDITIONAL STATEMENTS AND FUNCTIONS

If it is necessary to do so, INPUT with INFORMAT, LENGTH, or ATTRIB with FORMAT statements at the beginning of the data step that specify NEWVAR and \$ (the dollar symbol establishes the variable in the character format) can be included [2]. The INPUT statement would require INFORMAT as the default length of a variable specified in INPUT alone is 8 characters. These statements are best needed if it is required to update values in NEWVAR to be longer than the original length of OLDVAR or preserve the full lengths of nonproblematic values in OLDVAR that could be truncated in NEWVAR. Additionally, while it is not necessary for processing the data, running PROC CONTENTS after this data step or creating a LENGTHC function output can serve as a useful self-check to verify the lengths and formats for NEWVAR.

Another option to edit a long character variable, as simple as the previous data step but a more advanced function, is to utilize TRANWRD. The TRANWRD function replaces all occurrences of a substring in a character string such as single characters or specific phrases [3]. In the context of dealing with long character variables, this function is useful as it can be a workaround to listing out each instance of problematic values whether there are numerous instances per variable or even if one single value possesses multiple foreign/suspicious characters. Utilizing the TRANWRD function would resemble the following:

```
data <processing datafile name>;
set <converted datafile name>;
NEWVAR = TRANWRD(OLDVAR, "?", "'");
put NEWVAR = ;
run;
```

In the example above, the new processing variable NEWVAR consists of the original variable OLDVAR which has question marks in its values replaced with apostrophes (question marks in this circumstance are improperly converted characters instead of being foreign/suspicious). For disclosure risk mitigation, the function would include specific phrases or substrings; an example being:

```
NEWVAR = TRANWRD(OLDVAR, "JOHN SMITH", "[NAME MASKED]");
```

In this example, all instances of "JOHN SMITH" are replaced by "[NAME MASKED]". It is extremely important for users to review a dataset's frequency outputs before utilizing TRANWRD, as an improper usage of the function could accidentally replace legitimately punctuated characters and nonproblematic substrings that were present in OLDVAR.

Other functions that can be utilized to fix values in a long character variable, while not falling under the main technical or disclosive issues discussed earlier, include COMPBL and COMPRESS. The COMPBL

function removes multiple successive blank spaces in a value and reduces them to just one blank space. COMPRESS not only can remove blank spaces completely from character values but also any other characters that are specified in its function. Utilizing either of these functions would resemble the following, where in this example the COMPRESS function for NEWVAR3 also removes other characters from OLDVAR such as tilde, backtrack, and question mark (the extra space at the end before the last apostrophe indicates that blank spaces should be removed from OLDVAR):

```
data <processing datafile name>;
set <converted datafile name>;
NEWVAR1 = COMPBL (OLDVAR);
NEWVAR2 = COMPRESS (OLDVAR);
NEWVAR3 = COMPRESS (OLDVAR, '~`? ');
run;
```

Should removing blank spaces only on certain ends of values be necessary, the TRIM (trailing blanks), LEFT (leading blanks), and STRIP (both trailing and leading blanks) functions can be utilized. If the capitalizations of values need to be changed, then the PROPCASE, UPCASE, and LOWCASE functions could be utilized to change all the values in a variable to sentence, upper, or lower case respectively.

FURTHER PROCESSING AND EXPORTING THE DATA

Whichever prior method is chosen with editing the long character variables, the next step is to replace the original variables in the initially imported dataset with the updated variables. Another data step, this one containing RETAIN and RENAME functions, is utilized to reorder the dataset to its original order of variables and replace the original variables with the new ones:

```
data <final datafile name>;
RETAIN ID NEWVAR;
set <processing datafile name>;
RENAME NEWVAR = OLDVAR;
run;
```

The RETAIN statement keeps the variables that were created from the last data step and RENAME changes the names of the variables, in this case reverting NEWVAR back to OLDVAR. If the TRANWRD approach is utilized, it is possible to skip the RETAIN and RENAME functions by specifying OLDVAR in the TRANWRD function instead of NEWVAR (it would automatically overwrite the previous version of the variable).

Once the data has been edited to remedy the problematic long character variables, the final part of the SAS code is to utilize PROC EXPORT to create a new file containing the desired output. The data step is as follows:

```
proc export data = <datafile name> (KEEP=)
outfile = <filepath specifying new dataset>
dbms = CSV
replace;
run;
```

The file type of choice in this example for exporting the data in the DBMS statement is CSV, as it can be utilized by any given statistical package to link variables with datasets that were previously incapable of sustaining them.

BENEFITS

The primary benefit with these data steps is that they make the data further processable and devoid of technical issues. They are additionally able to do so in a way that makes updating the data hassle-free and non-tedious. From a disclosure mitigation perspective, these data steps show the ease of removing

sensitive substrings and making otherwise restrictive datasets publicly accessible. Additionally, because of the nature of long character variables, it helps to emphasize that qualitative data is just as important with SAS as is quantitative data. As a matter of fact, the ability for SAS to perform qualitative data analysis using calculations such as frequencies, correlations, and cluster analyses adds to this notion [4].

CONCLUSION

SAS software is a tool that is proven to tackle some of the most significant issues when handling data, and remedying problems pertaining to long character variables is no exception. The ability for SAS to import the variables, edit responses associated with either individual observations or all values in a variable, and then exporting the variables to user-friendly formats shows the program's effectiveness. It is even possible to expand upon the basic statements and functions previously outlined to produce more advanced procedures with macros and even regular expressions. Because of this ability, SAS is a tool that is encouraged to be utilized by those who work with qualitative information in any capacity.

ACKNOWLEDGMENTS

The author would like to thank Dr. Amy Pienta and Dr. Scott Liening for providing helpful feedback on earlier drafts of this paper and the Curation Unit at ICPSR for their support in making this paper possible.

REFERENCES

1. ICPSR's Approach to Confidentiality. ICPSR. Retrieved July 18, 2024, from <https://www.icpsr.umich.edu/web/pages/datamanagement/confidentiality/index.html>
2. Create and Modify SAS Variables (2024). SAS Help Center. Retrieved July 18, 2024, from https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/lepg/n1rz4zvmck7guqn15fbtj56i6rxk.htm
3. TRANWRD Function (2024). SAS Help Center. Retrieved July 18, 2024, from https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/lefunctionsref/p0pgemqcsIm9uen1tvr5gcrusgrw.htm
4. Stewart, R.G. (2000). Using the SAS® System and QSR NUD*IST® for qualitative data analysis (QDA). Retrieved July 18, 2024, from <https://www8.sas.com/scholars/05/PREVIOUS/2000/pdf/p-910.pdf>

CONTACT INFORMATION

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

Piotr Krzystek
ICPSR at the University of Michigan
piotrk@umich.edu

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.