PharmaSUG 2022 - Paper QT-005

You Don't Know Where That's Been! Pitfalls of Cutting-and-Pasting Directly into the Programming Work Flow

Dave Hall, Quality Data Services

ABSTRACT

It is sometimes tempting to save time by cutting-and-pasting values directly into a SAS program, a specification, or any other element involved in the life-cycle of a project. The alternative, when it involves substantial amounts of text that would otherwise need to be typed, can be tedious and time-consuming. However, inserting values from potentially unreliable sources without adequate safeguards can be a risky proposition. In the end, it may be wiser and more efficient to be more deliberate about adding such values.

Saving time by cutting-and-pasting is like finding a piece of candy on the street. Although it might look tasty, there's no way to know where it came from, where it's been, or what's in it. Putting it in one's mouth is unwise and risky. Caution is advised, because you don't know where that's been!

INTRODUCTION

Many times, values and text strings which are to be incorporated into a process are readily available from sources such as web pages or high-level files. Simply taking the value directly from such a source by cutting-and-pasting may seem to be an efficient method of placing the text where it's needed, with no apparent downside. Instead of minutes, or even hours, the task is completed painlessly in seconds. What could go wrong?

A lot could. Like that tasty piece of candy found alongside the curb of a busy street, there are risks involved in introducing to the process text taken from places such as a web page or even an Excel spreadsheet or Word document. It may contain special characters which are not visible to the human eye that can contaminate programs, specifications, and data sets, causing problems which are ultimately not worth the time that was saved by cutting-and-pasting. Or it could contain characters which are not part of the US ASCII character set, complicating Pinnacle 21 validation.

Some problems will be immediately apparent when such values are used. For instance, it is quickly evident when a text string contains a character which creates problems when displayed. Unexpected line feeds, for example, can not only distort the appearance of an otherwise orderly listing, table, or figure, but also can cause issues with titles or footnotes, and even disrupt line-counting and paging.

Other potential problems are not limited to the immediate realm of SAS processing, and may not manifest themselves until the project nears completion. In many shops, running Pinnacle 21 reports and creation of the define.xml is left toward the end of the timeline. This is a most inopportune time for problems to appear. By then, SDTM domains and ADaM data sets have been programmed and validated, as have tables, listings and figures. Correcting the problem in order to meet Pinnacle 21 requirements may require repeating some or all of these activities, creating needless anxiety and even threatening the project timeline – all for the purpose of avoiding some typing earlier in the process when there was no need to rush.

PROACTIVELY FILTERING OUT SPECIAL AND NON-US ASCII CHARACTERS

One conceivable method of minimizing the risk is to introduce SAS code to detect and eliminate undesired characters, be they special, non-printable characters or those that are not part of the US ASCII character set. Methods and SAS code for doing so have been thoroughly researched and documented. The Recommended Reading section lists several articles and papers already written on this topic, as well

as a link to a full table of US ASCII characters and their octal and hexadecimal codes, which will likely be needed to identify them programmatically.

None of this code is going to be foolproof, however. Some unwanted characters will likely slip through even the most rigorous programming. The best solution for keeping such characters out of the programming work flow is to never include them in the first place.

EXAMPLES

Following are three examples of how non-standard and special characters, which were cut-and-paste into the work flow, disrupted the process and created a need for re-work.

TEXT IN A CHARACTER VARIABLE

This example involves a PAD (Peripheral Artery Disease) Quality of Life data set, comprised of questions and responses by subjects to forty questions. The text of each question was cut-and-paste directly into a SAS program from a tab of an ADaM specification in an Excel spreadsheet, with double-quotes added around the text after pasting. Prior to that, the values in the spreadsheet tab had themselves been populated via cut-and-paste from a PDF containing the survey question text. These character strings have been all over the place. Who knows what's in them at this point? Who knows where that's been?

Here are just two of the questions as they appear in the ADaM specification spreadsheet:

Item	Description
QOL26	My PAD makes me feel 'not normal'
FACTOR5	Factor 5 – Positive Adaptation

It's obvious that the single quotes in the first row and the hyphen in the second row are not part of the US ASCII character set, which are as follows for these characters:

Description	US ASCII Character	DEC	OCTAL	HEX
Apostrophe/Single Quote	'	39	27	047
Hyphen/Minus	-	45	2D	055

The corresponding lines of the SAS program to populate the question text variable for these two, into which these character strings were cut-and-paste directly from the specification spreadsheet, looked like this:

```
else if (PARAMN=26) then PARAM = "My PAD makes me feel 'not normal'";
```

and

```
else if (PARAMN=43) then PARAM = "Factor 5 – Positive Adaptation";
```

SAS is robust enough to accept the non-standard characters and display them in a table via PROC REPORT:

Pharma Company LLC
Page 2 of 15

Protocol 1234567

Table 1.2.3 Summary of PAD Quality of Life

Factor 5 – Positive Adaptation [5]	n	36
	Mean (STD)	xx.x (xx.xx)
	Median	xx.x
	Min, Max	x, xxx
Q26. My PAD makes me feel 'not normal'	n	36
	Mean (STD)	x.x (x.xx)
	Median	x.x
	Min, Max	x, x

Problems arise, however, when the data set is converted to a SAS transport file for Pinnacle 21 validation. Transport files support only US ASCII characters. As a result, the single quotes and hyphen mentioned above (and other characters across the forty PAD Questionnaire items) will be distorted or left out completely. This means the text values in the transport file will not match values in the corresponding code list in the define.xml (unless they, too were cut-and-paste from the same sources, in which case they would not match the CRF and project documentation), which includes the quotes and hyphen from the US ASCII character set. The result will be a series of warnings in the Pinnacle 21 report:

Domain	Count	Variables	Values	Message
ADPADQOL	36	PARAM	Factor 5 Positive Adaptation	Value for PARAM not found in (ADPADQOL Parameter) user-defined codelist
ADPADQOL	36	PARAM	My PAD makes me feel not normal	Value for PARAM not found in (ADPADQOL Parameter) user-defined codelist

For these two example Questionnaire items alone, there are 72 issues (one per subject per question) in the Pinnacle 21 report which will need to be reconciled in some way. The embarrassing and sloppy error could be explained in the reviewer's guide. Otherwise, the problem must be resolved at the SDTM or ADaM level (wherever the non-standard characters were introduced) by replacing them with standard US ASCII characters. This will require regeneration and revalidation of data (SDTM and/or ADaM), followed by regeneration and revalidation of TLF, followed by rerunning of Pinnacle 21 reports. Somebody may well be staying up late in order to hit that deadline. Depending on timing and circumstances, it is quite possible that almost everything will have to be rerun.

ADAM DATA SET VARIABLE LABEL

Introduction of unfiltered cut-and-paste text can also be risky with regard to metadata that is eventually used for creation of ADaM data sets. If a special character were to be inadvertently embedded in a variable label, for instance, it can lead to a warning in Pinnacle 21 reports.

In this example, the label for ADaM variable ANL01FL was cut-and-paste from this web page (with "zz" replaced by "01"), which is displaying an excerpt from the current ADaM Implementation Guide

ANLzzFL	Analysis Record Flag zz	Char	Y	Cond	ANLzzFL is a conditionally required flag to be used in addition to other selection variables when the other selection variables in combination are insufficient to identify the exact set of records used for one or more analyses. Often one ANLzzFL will serve to support the accurate selection of records for more than one analysis.

and inserted directly into a spreadsheet containing an ADaM data set specification. It was subsequently extracted programmatically from that specification and used as a SAS data set variable label in the ADaM data set.

It may appear to the human eye as a 23-character label, well below the CDISC limit of 40. However, Pinnacle 21 stumbles on the line feed character embedded in the text string after "Record", which was required for presentation on the web page to neatly wrap the label onto a new line.

The effect of the line feed can actually be seen in the "Values" column of the Pinnacle 21 report in the ADAC rows, and is especially marked in the second one pictured. Although there is plenty of space, the label text wraps to a new line after "Record". The special character has also caused a miscount of the length of the text string, evoking the warning that it exceeds the limit of 40 when it appears to contain only 23 characters. More importantly, its presence has caused a mismatch with the standard label for ANL01FL, which obviously does not contain an embedded line feed.

Domain	Variables	Values	Pinnacle 21 ID	Message
ADAC	LABEL	Analysis Record Flag 01	AD0016	Variable label length is greater than 40 characters
ADAC	VARIABLE, LABEL	ANL01FL, Analysis Record Flag 01	AD0018	Variable label mismatch between dataset and ADaM standard

The ADaM specification and the data set itself will have to be updated and revalidated, and TLFs will have to be rerun and revalidated. Pinnacle 21 reports, of course, will also need to be regenerated. All to avoid typing the text of a variable label rather than cutting-and-pasting it from a web page.

FOOTNOTE TEXT

There is risk even at the TLF level with regard to direct, unfiltered cutting-and-pasting. In this example, footnote text was taken directly from a table mockup which was stored in a Word document. It is important to note that high level files such as Excel spreadsheets, PDFs, and Word documents routinely include non-standard characters. Here, the following footnote text was cut-and-paste from the mockup into a spreadsheet used in title and footnote processing:

[3] The leg with lowest TBI (Toe–Brachial Index).

The footnote text includes the same elongated hyphen seen in the first example (it's among the most common offenders). In this case, SAS was unable to display it as shown, perhaps because it is enclosed in parentheses. Regardless of the reason, this footnote was displayed as follows in the table, with the hyphen replaced by an open parenthesis:

```
[3] The leg with lowest TBI (Toe(Brachial Index).
```

With any luck, it will be noticed quickly, and the troublesome character replaced with a standard hyphen. Even so, the extra work could have been avoided by typing the footnote instead of cutting-and-pasting.

CONCLUSION

As has been shown, there are risks involved when it comes to cutting-and-pasting text from unreliable sources directly into the programming work flow. The cost of saving a few minutes at the front end of a project, when hours are plentiful, may end up being unacceptably high.

As time passes, familiarity and comfort with such sources, especially with regard to the ever-present internet, will only rise. The temptation to use such sources with no filter will likely grow as the sense of risk wanes. It may be wise to remember the analogy of the candy found in the street when considering costs and benefits of such an approach. It may look tasty. No danger is apparent. The saving of tedious typing beckons. But in the end, no programmer can know for sure what's in that, or where it's been!

ACKNOWLEDGMENTS

Thank you to Ellina Babouchkina for unlimited patience and kindness while dragging me into the 21st century as a SAS programmer. I could never have written nor even understood this paper if not for Ellina.

RECOMMENDED READING

- "The US ASCII Character Set", Columbia University, available at www.columbia.edu/kermit/ascii.html
- "Paper 87-2019, People Keep ASCIIng Me About These Characters", Dan Konkler, Rebecca Bowermaster, and Stephanie Ann Sanchez, Roche Molecular Solutions, Available at https://www.lexjansen.com/wuss/2019/87_Final_Paper_PDF.pdf
- "Non Printable & Special Characters: Problems and how to overcome them", Sridhar R Dodlapati, Praveen Lakkaraju, Naresh Tulluru and Zemin Zeng, Available at http://www.lexjansen.com/nesug/nesug10/ff/ff04.pdf
- "How to Remove Special Character and Non-Printable Character in SAS", Balram Chauhan, Available at https://balramchauhan.com/how-to-remove-special-character-and-non-printable-character-in-sas

CONTACT INFORMATION

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

Dave Hall
Quality Data Services
davehall@rocketmail.com