PharmaSUG 2018 - Paper AD-17

Managing Table Preparation and Related Coding Task with Multilingual Settings by Using SAS® Unicode Version

Kai Koo, Abbott Medical Devices, Santa Clara, CA

ABSTRACT

To apply the approval of drugs and medical devices outside the US, it is common that all the supporting documents, including statistical analysis tables and listings of trial results should use or be translated to the official languages of specific countries. Although SAS provides localized versions to support statistical analyses and table outputs in different languages, the SAS codes containing specific character encodings may display correctly in one SAS language version, but fail to work normally in other localized versions. To simplify the code maintenance and output table management in multilingual settings, SAS Unicode version, which can handle most of the writing system, is a favorable solution to produce similar tables and listings with different languages in single code. Furthermore, the SAS codes written in different localized versions can be translated from different character encoding systems to Unicode programmatically through the SAS system option - ENCODING.

INTRODUCTION

Some languages based on the Latin characters use different combinations of limited number of alphabets to form vocabularies. It is easy for single-byte encoding systems, such as ASCII or EBCDIC, to handle those languages. However, for other languages containing large numbers of characters and cannot be presented by the single-byte encoding system, using multi-byte encoding system is a necessary. Since 1970s, difference countries and regions started to define and implement their own multi-byte encoding systems. Although, those multi-byte encoding systems solved the problem of processing non-Latin character based languages in computers, uses worked in different encoding environments suffered the compatibility issue. For example, sometimes, different encoding systems define the same character to different byte combination [e.g., 中 : D6D6 (GB2321), A4A4 (Big5), 9286 (Shift JIS)], or compile different characters to the same data bytes [e.g., D6D6: 中 (GB2321), 笃 (Big5)]. This issue made it impossible for computers configurated in one encoding system to open electronic document encoded in another system normally without appropriate code mapping.

In 1980, efforts have been made to solve this compatible issue by designing a new encoding system to include the character sets for most of the writing systems. The original Unicode standard was announced in 1991 (https://unicode.org/). Currently, this encoding system has included 100,000 different characters, and almost all modern computer operating systems, program languages and applications support Unicode environment. SAS also provides Unicode support version in its additional language support package. This Unicode supported version makes generating the same analytical tables by using different language settings in one code possible.

CONCEPTS IMPLEMENTATIOIN

As a biometric team, which supports global clinical trials, we keep our capacity to generate tables, listings and figures in different languages with the help of our global team members. Recently, we received more and more requests from regulatory agencies outside the US to translate the tables prepared for US trials (originally submitted to FDA) to other languages, if we want to use those US trial table sets as one of the supporting documents for product approval in foreign countries.

To meet those requirements, we can either hire translation companies to translate those tables from English to other languages, or regenerate all tables by replacing original titles, column/row labels and even footnotes to languages other than English programmatically. Using translation companies is a straightforward approach; however, sometimes, it is not the most time/cost saving method. Re-run tables in different SAS localized versions or re-define SAS internationalization settings sound efficient; however, it needs extra effort to maintain two or more versions of the codes for same analysis and even switch back and forth of the contents in SAS configuration files. After intensive research and tests, we feel that SAS Unicode version is a good solution to process multilingual requests in one SAS session.

CONCEPT 1 – One code for all different languages:

The following SAS sample code will generate the same baseline demographic table which provided 5 basic statistics (mean, standard deviation, median, minimum and maximum) of patients' age in 3 different languages. This program containing characters of different languages can be opened, edited, saved and executed without any problems in SAS Unicode supported version.

```
dm 'out; clear; odsresults; clear; log; clear;';
libname ads 'c:\temp';

proc means data=ads.dm_test(keep=subjid age);
  var age;
  output out=agx mean=age_avg std=age_std min=age_min max=age_max
                 median=age_mid p25=age_q1 p75=age_q3;
run;

proc format;
  value rowlab_en /* for English version */
  1='Age (Years)'
  101='(n)'
  102='Mean ^\u177; SD'
  103='Median'
  104='[Q1, Q3]'
  105='Range [min, max]'
  201='Characteristic'
  202='Registry 1';

  value rowlab_cn /* for Chinese (simplified font) version */
  1='年龄 (岁)'
  101='(例数)'
  102='均数 ^\u177; 标准差'
  103='中位数'
  104='[Q1, Q3]'
  105='极差 [最小值, 最大值]'
  201='特征'
  202='试验 1';

  value rowlab_jp /* for Japanese version */
  1='年齢 （歳）'
  101='(n)'
  102='平均 ^\u177; 標準偏差'
  103='中央値'
  104='[Q1, Q3]'
  105='範囲 [最小, 最大]'
  201='特性'
```

```
  202='治験実施 1';
run;

%macro tab_multiplang(lx=, tit1=);
  %local cf1 cf2;
  data _null_;
    call symput('cf1', put(201, rowlab_&lx..));
    call symput('cf2', put(202, rowlab_&lx..));
  run;

  data tab1(keep=c1 c2);
    set agx;
    length c1 c2 $40.;

    c1=put(1, rowlab_&lx..);
    c2='';
    output;

    c1=put(101, rowlab_&lx..);
    c2=strip(put(_freq_, 5.));
    output;

    c1=put(102, rowlab_&lx..);
    c2=strip(put(age_avg, 5.1))||' ^\u177; '||strip(put(age_std, 5.1));
    output;

    c1=put(103, rowlab_&lx..);
    c2=strip(put(age_mid, 3.));
    output;

    c1=put(104, rowlab_&lx..);
    c2=strip(put(age_q1, 3.))||', '||strip(put(age_q3, 3.));
    output;

    c1=put(105, rowlab_&lx..);
    c2=strip(put(age_min, 3.))||', '||strip(put(age_max, 3.));
    output;

    label c1="&cf1" c2="&cf2";
  run;

  ods escapechar='^';
  ODS LISTING CLOSE;
  options nodate nonumber;
  ods rtf file="c:\temp\DM_test_&lx..rtf";

  title1 j=c "&tit1";

  proc report data=tab1 nowd split="\" style(report)={width=5 in};
    column c1 c2;
    define c1 / flow style(column)=[just=L cellspacing=0 cellwidth=40%];
    define c2 / flow style(column)=[just=c];
  run;
  ods rtf close;

%mend tab_multiplang;
```

```
%tab_multiplang(lx=en, tit1=%str(Table 1. Baseline Demographics - ITT
Population));

%tab_multiplang(lx=cn, tit1=%str(表 1. 基线人口统计学特征 – 意向治疗人群));

%tab_multiplang(lx=jp, tit1=%str(表 1. 手技前の人口統計学的特性 - ITT解析));
```

In this short table code, the PROC FORMAT section containing all row labels and column names of different languages is the key section of multilingual processing. Through a PUT statement [e.g., "row label/column name" = put(xxx, rowlab_&lx..)] in DATA STEP, the appropriate text string is transformed from FORMAT section to the final row labels and column names of the table. Users can add as many languages of row labels and column names as they need. Theoretically, the only limit is the character sets that Unicode has currently collected. No specific coding logics or statements need to be taken care of due to any language changes. It is not necessary to add LOCALE option in entire SAS code, unless users really want to change the displayed messages from original English to other languages.

In this sample code, it only uses a macro variable "LX" to define the specific language (e.g., en = English, cn = Chinese (simplified), or jp = Japanese) used, and pass the correct table title through "TIT1" variable. The programming team can start coding tasks from any language for table labels and titles in SAS Unicode version, and later add other language modules (i.e., insert new FORMAT sections) under this Unicode supporting environment per requests. In comparison with using different "localized" SAS versions or by adjusting the National Language Support (NLS) settings to generate the same table in different languages, this clean and simple approach is quite straightforward, and easy for code maintenance and version control.

CONCEPT 2 – Convert SAS codes written in different language versions to Unicode

The sample code in concept 1 demonstrates a simple solution for multilingual request in table preparation. However, it does not mean the old SAS code prepared in non-Unicode environments, such as GB2312, Shift-JIS etc., are totally useless under Unicode environment. Actually, the old SAS programs prepared in non-Unicode environments can be transferred to Unicode, especially UTF-8, easily.

Almost all current major computer operation systems are Unicode friendly. They provide system tools for users to convert text files between Unicode and non-Unicode encodings. In UNIX®/LINUX, uses can call the "iconv" command for file conversion. The Microsoft® Windows also has MultiByteToWideChar / WideCharToMultiByte functions to map Unicode/Non-Unicode strings after users defined the correct parameters and encoding information.

SAS users can take advantages of the fact that SAS also has its own method to convert text file (including SAS program file) from non-Unicode to Unicode. Users can simply use SAS ENCODING system option to accomplish this task. There is no need to use system tools or other third party encoding applications for text file conversion. The small but fully functioned (under SAS Unicode version) macro %TranUnicode listed below is an example.

```
%macro TransUnicode(fn1=, fn2=, ecx=); /* for SAS Unicode version */

  filename extfile "&fn1" encoding="&ecx";

  data dmcode;
    length tx $1000.;
```

```
    infile extfile truncover;
    input tx $ 1-1000;
  run;

  data _null_;  /* Output UTF-8 encoded text file */
    set dmcode;
    file "&fn2" lrecl = 1000;
    put tx;
  run;
%mend TransUnicode;

%TransUnicode(fn1=%str(c:\temp\cn_basedm.sas),
fn2=%str(c:\temp\cn_basedm_u.sas), ecx=GB2312);
```

The center of this small macro is the FILENNAE statement and system option, ENCODING. For example, we can convert a SAS program file written in SAS Chinese (simplified) version which is GB2312 encoded to a new file encoded in UTF-8 format. The user just passes the original encoding information through the macro variable “ecx” to the FILENAME statement with the ENCODING=”&ecx” option, SAS Unicode version will read this GB2312 encoded text file from INFILE statement of a DATA step and compiled each line of text string to a UTF-8 encoded character variable “tx”. After output the content of “tx” to an output file by PUT statement from the second DATA step, the original GB2312 encoded SAS program is converted to a UTF-8 encoded SAS program file. Using this small macro, programmers just input the correct encoding information and any SAS codes prepared from different localized versions will be translated to Unicode (UTF-8) without any problem.

DISSCUSSION AND CONCLUSION

In the recent years, many articles and presentations related to SAS National Language Support (NLS) or Unicode versions have been presented in different conferences of the SAS community. They provide details in system settings or instructions of specific function calls. Those articles delivered promising messages in using SAS Unicode version or LOCALE / ENCODING system options for multilingual data processing. Compared to those publications, this article helps to bridge the gap between promising messages and real world applications through two small but practical examples. SAS users can use those simple examples to test, evaluate, and later expand or modify them to fit their own needs.

Based on the settings introduced in this article, one table code can generate same table with titles, row labels and column names in different languages without many coding changes. This approach made version control and code maintenance simple, because only one table code section is needed for different language settings. SAS programming team members located in different geographical areas can produce tables by adding new language module to PROC FORMAT section without touching other parts of the SAS program. The global team can handle different languages in one code under Unicode environment without further modification of system ENCODING option in configuration file (e.g., SASV9.cfg – in Windows environment) and change the LOCALE settings. Furthermore, the old SAS codes previously developed in different SAS language versions can be reused efficiently by translating those codes to Unicode (UTF-8) directly by a simple code conversion macro under SAS Unicode support version.

As Unicode becomes one of the main streams in the computer operating systems and programming languages, the SAS Unicode support version should have the potential to become the core of SAS system. SAS uses should investigate the possibility to migrate current codes and even datasets to Unicode. It should benefit many SAS users who need a simple and efficient way to process data and generate tables / listings for regulatory submissions to many countries in different languages.

REFERENCES

Mickaël Bouedo (2012) Write Once Run Anywhere! How to Make Your SAS® Applications Speak Many Languages. SASGF 2012, Paper 254-2012.

Xiaojin QinThe (2014) SAS® LOCALE Option Win an international passport for your SAS code. PharmaSUG 2014, Paper AD-05.

Chao Wang (2014) The Power of SAS National Language Support - Embrace Multilingual Data. PharmaSUG-China 2014, Paper PT02.

CONTACT INFORMATION

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

Name: Kai Koo
Enterprise: Abbott Medical Devices
Address: 3200 Lakeside Dr.
City, State ZIP: Santa Clara, CA 95054
E-mail: kai.koo@abbott.com