Paper SM08

Unlocking the Enigma: Exploring SAS and R Strategies for Handling Non-Printable and Special Characters

Eldho Alias, Catalyst Clinical Research, Trivandrum, India

Vishal V Nair, Catalyst Clinical Research, Trivandrum, India

Pooja Pradeep Pillai, Catalyst Clinical Research, Trivandrum, India

ABSTRACT

Quality is one of the key requirements in clinical trial deliverables. The existence of non-printable and special characters in data can cause serious problems in the reports which includes incorrect counts/statistics in the deliverables, or the minor issues such as incorrect line breaks, page brakes or appearance of strange symbols in the reports.

Here comes the importance of this paper to identify and handle the non-printable and special characters using SAS® 9.4 as well as R®. This paper will portrait the concept of SAS macro/R functions which have the flexibility to either remove those characters or replace with specified characters in a single run for all the datasets in a library. Additionally, a report will be created with information including the identified/replaced characters with the associated dataset and variable name that will help the user for easy understanding. SAS/R codes will be shared along with the paper.

INTRODUCTION

Non-printable and special characters are primarily used for specific formatting purposes and are often invisible, making it challenging to identify their presence in outputs. This poses a significant issue because their presence can lead to incorrect interpretations. Therefore, effectively handling these characters is crucial in clinical deliverables.

The objective of this paper is to provide insights into the impacts of non-printable and special characters in clinical trial data. It also outlines efficient approaches that programmers can use to handle these characters without wasting much time.

NON-PRINTABLE AND SPECIAL CHARACTERS?

Non-printable and special characters play important roles in text processing and programming. Non-printable characters (ASCII 0-31) are those that do not have a visual representation; they control formatting or data flow instead. Common examples include carriage returns (CR) and line feeds (LF), which dictate how text is displayed on different lines, as well as tabs that create indentation.

In contrast, non-ASCII special characters have specific meanings or functions, often used in programming languages and markup formats. These include characters like é, ñ, ü, as well as extended punctuation marks such as ©, ®, and ™. Understanding both types of characters is crucial for effectively managing and manipulating text, ensuring that it displays and behaves as intended in various applications.

The Pinnacle 21 (P21) validation tool enforces several rules for non-ASCII characters in submission datasets to ensure data consistency and compliance with FDA requirements. One key rule is that non-ASCII characters should be removed or converted to standard ASCII characters before submission. This helps avoid potential issues with data processing and ensures that the data meets regulatory standards.

HOW DOES IT ORIGINATE AND WHO IS RESPONSIBLE?

In the context of clinical trial submissions, non-printable and special characters can arise from various processes related to data management, documentation, and electronic submissions.

i. When clinical trial data is entered into electronic databases or electronic data capture (EDC) systems, non-printable characters may inadvertently be included. This can happen during manual data entry or when transferring data between systems.

ii. Different software applications used for data collection, analysis, and submission may generate special characters as part of their formatting or coding.

iii. Data files (like CSV, XML, or JSON) might incorporate non-printable characters based on the format specifications. For example, line breaks and delimiters are often essential for data organization but may not be visible in standard text views.
iv. During data cleaning and processing, automated scripts or algorithms might introduce non-printable characters, either through errors or as part of the data manipulation process.

The responsibility typically falls on both the sponsor and the investigator/institution involved in the trial.
The sponsor is tasked with overseeing the overall conduct of the trial, including data management. They must ensure that the data is clean, accurate, and devoid of non-printable characters that could compromise data analysis and reporting.
The investigator or the institution conducting the trial is responsible for ensuring that the collected data is accurate and properly formatted. They should adhere to established guidelines for data entry and management to prevent the introduction of non-printable characters.
Both parties need to collaborate to establish clear protocols for data handling, including the identification and correction of non-printable characters, to uphold the integrity of the clinical trial data.
The data analysis or reporting team plays a crucial role in identifying any non-printable special characters in the data. Once identified, they should report these issues to the relevant team, such as the data management team within the CRO or the sponsor's data management team. This ensures that the data can be cleaned and corrected promptly, maintaining the integrity and accuracy of the clinical trial data.

WHY DOES IT NEED HANDLING?

Handling non-printable special characters before clinical trial submission is crucial. Non-printable characters can distort data, leading to inaccurate statistics or counts in deliverables, which affects the integrity of the clinical trial data. These characters can create problems such as improper line breaks, page breaks, or the appearance of strange symbols in reports, affecting both readability and the professional presentation of documents.
Regulatory agencies mandate clean and accurate data submissions. The presence of non-printable characters can result in non-compliance with these requirements, potentially delaying the approval process. Different systems and software utilized in clinical trials may not handle non-printable characters uniformly, leading to processing errors.
Ensuring that all data is clean and free from non-printable characters is essential for upholding the quality and reliability of clinical trial results, which is vital for regulatory approval and the overall success of the trial.

The following example highlights the importance of handling these characters in data. The count in Output 1 is incorrect due to the presence of a line feed character in the data. The line feed character is present in one of the values for 'Corneal epithelium defect,' causing SAS/R to treat both values as different. As a result, two records are generated in the output, each with a count of 1. In contrast, Output 2 is created by updating the source dataset to remove the line feed character using SAS macros or R functions, resulting in a single record for the value 'Corneal epithelium defect' with the correct count of 2.

(Output 1)

Catalyst CR Confidential Page 1 of 2
Protocol: XX

Characteristic	Location	Trt A	Trt B	Total
Any TEAE [a]	Ocular-Any Treated Eye	2 (16.7)	0	2 (11.1)
Eye disorders	Ocular-Any Treated Eye	2 (16.7)	0	2 (11.1)
Corneal epithelium defect	Ocular-Any Treated Eye	1 (8.3)	0	1 (5.6)
Corneal opacity	Ocular-Any Treated Eye	1 (8.3)	0	1 (5.6)
Keratic precipitates	Ocular-Any Treated Eye	1 (8.3)	0	1 (5.6)
Visual impairment	Ocular-Any Treated Eye	1 (8.3)	0	1 (5.6)
Any TEAE [a]	Ocular-All Eyes	4 (33.3)	1 (16.7)	5 (27.8)
Eye disorders	Ocular-All Eyes	4 (33.3)	1 (16.7)	5 (27.8)
Visual impairment	Ocular-All Eyes	1 (8.3)	1 (16.7)	2 (11.1)
Corneal epithelium defect	Ocular-All Eyes	1 (8.3)	0	1 (5.6)
Corneal epithelium defect	Ocular-All Eyes	1 (8.3)	0	1 (5.6)
Corneal opacity	Ocular-All Eyes	1 (8.3)	0	1 (5.6)

(Output 2)

Characteristic	Location	Trt A	Trt B	Total
Any TEAE [a]	Ocular-Any Treated Eye	2 (16.7)	0	2 (11.1)
Eye disorders	Ocular-Any Treated Eye	2 (16.7)	0	2 (11.1)
Corneal epithelium defect	Ocular-Any Treated Eye	1 (8.3)	0	1 (5.6)
Corneal opacity	Ocular-Any Treated Eye	1 (8.3)	0	1 (5.6)
Keratic precipitates	Ocular-Any Treated Eye	1 (8.3)	0	1 (5.6)
Visual impairment	Ocular-Any Treated Eye	1 (8.3)	0	1 (5.6)
Any TEAE [a]	Ocular-All Eyes	4 (33.3)	1 (16.7)	5 (27.8)
Eye disorders	Ocular-All Eyes	4 (33.3)	1 (16.7)	5 (27.8)
Visual impairment	Ocular-All Eyes	1 (8.3)	1 (16.7)	2 (11.1)
Corneal epithelium defect	Ocular-All Eyes	2 (16.7)	0	2 (11.1)
Corneal opacity	Ocular-All Eyes	1 (8.3)	0	1 (5.6)

Spotting such duplications during a review often relies on luck, which increases the risk of missing these issues. This can lead to incorrect conclusions and decisions. Therefore, it's essential to have strong data cleaning and validation processes in place to identify and correct these errors before they affect the final report. It would be beneficial to perform a final check for these characters before generating the report, even if a robust data-cleaning process has been implemented. If any issues are identified, they should be reported to the responsible team to update and refresh the data with the corrected version.
The next step is to establish a procedure to check for these characters throughout the entire dataset. It would be helpful to have a tool that reads all the data from a specified location, identifies these characters, and generates a report. This report would provide a clear overview of the issues and can be shared with the responsible team for correction.

DETAILS OF SAS MACRO AND R FUNCTION

SAS Macro:

The SAS macro '**scnprpt'** has three required macro parameters: '**inpath'**, '**outpath'**, and '**exlpath'**. The purpose of these parameters is as follows.

- **inpath** - Path of the SAS library containing the dataset from which non-printable or special characters need to be removed or replaced.
- **outpath** - Path of the SAS library where the dataset should be stored after handling non-printable or special characters.
- **exlpath** - Path of the Excel file where the data should be stored and later updated based on the requirement.

```
*************************************************************************************************
* MACRO NAME              : scnprpt.sas
* MACRO DESCRIPTION       : Macro for handling non-printable and special characters
* AUTHOR                  : Eldho Alias
* PLATFORM                : SAS 9.4 Windows 12
*************************************************************************************************
* SAMPLE CALL             : %scnprpt(inpath=%str(G:\Working Folder\Eldho\sm08\in),
                            outpath=%str(G:\Working Folder\Eldho\sm08\out),
                            exlpath=%str(G:\Working Folder\Eldho\sm08\excel));
*************************************************************************************************;
%macro scnprpt(inpath=,outpath=,exlpath=);
 options noquotelenmax;

 libname in "&inpath." access=readonly;*assigning the input library;
 libname out "&outpath.";*assigning the output library;

 *To check the existence of excel contain non-printable & special characters;
 %let filepath = &exlpath.\excl_npsc.xlsx;
```

When the macro is run for the first time, non-printable and special characters will be automatically handled for certain cases defined within the SAS program. An Excel file named '**excel_npsc**' will be created in the specified path. This

Excel file will contain the variable '**schar**' with the values of the predefined non-printable and special characters, and the variable '**rchar**' with the corresponding replacement characters.

Once the Excel file is created, only the values in the Excel file will be considered for handling non-printable and special characters. If any of the predefined values are not needed or if new values need to be added, the Excel file should be modified accordingly, and the macro should be run again. If any non-printable or special characters need to be removed without adding a space, the text '**nospace**' should be specified for those values in the '**rchar**' variable.

```
 %if %sysfunc(fileexist(&filepath)) %then %do;

  *Importing already existing excel which contains all the non-printable & special characters to be replaced;
  *Note: mention "nospace" in excel(column "rchar") if non-printable/special character needs to be replaced without
   space;

  proc import file="&exlpath\excl_npsc.xlsx"
    out=excl_npsc
    dbms=xlsx
    replace;
  run;

 %end;

  %else %do;

    *Identifying non-printable/special characters using SAS when the excel is not available;

    data excl_npsc;
      length schar rchar $7;
      do i=130 to 140, 142, 145 to 156, 158, 159, 161 to 255;
                schar=byte(i);
        rchar=" ";
        output;
      end;
      schar="'09'x";
      rchar="nospace";
      output;
      schar="'0D0A'x";
      rchar="nospace";
      output;
      schar="'0D'x";
      rchar="nospace";
      output;
      schar="'0A'x";
      rchar="nospace";
      output;
      rchar="nospace";
      keep schar rchar;
    run;

    *Numbering each non-printable/special characters, this number can later cross-checked between excel and
     report;

    data excl_npsc;
      retain RowNumber;
      set excl_npsc;
      RowNumber=_n_;
    run;

    *Exporting excl_npsc.sas into an excel format and excel can be updated based on the user requirement;

    proc export data=excl_npsc
```

```
    outfile="&exlpath\excl_npsc.xlsx"
    dbms=xlsx
    replace;
    sheet="schar";
  run;
 %end;

*Concatenating non-printable/special characters and replacing character to create a single variable;

data exc1_imp1 ;
 set excl_npsc(where=(not missing(schar))) end=last;
 if ~last then comp=strip(compress(schar))||"="||rchar||"*";
  else comp=strip(compress(schar))||"="||rchar;
 num=_n_;
run;

*Removing the duplicates;

proc sort data=exc1_imp1 nodupkey;
 by comp;
run;

proc sort data=exc1_imp1;
 by num;
run;

%let noobs=&sysnobs;

*To create a single record with columns(col1 to coln) for each combination of non-printable/special characters and
replacing character;

proc transpose data=exc1_imp1 out=exc1_imp2;
 var comp;
run;

*Concatenating col1 to coln to create a single column which contains all the combination of non-printable/special
 characters and replacing character;

data exc1_imp3;
 length compfin $32767;
 set exc1_imp2;
 compfin=cat(of col1--col%cmpres(&noobs));
 keep compfin;
run;

*Identifying the number of datasets in the specified library;

proc sql noprint;
 create table mem as
 select distinct memname as mem,memlabel from sashelp.vtable where find(libname,"IN","i");
 select count(*) into :cnt from sashelp.vtable where find(libname,"IN","i");
quit;

%do j=1 %to &cnt.;

 *passing the dataset name and the corresponding label to macro variables;

 proc sql noprint;
  select mem into:dsn from mem(firstobs=&j obs=&j);
  select memlabel into:dlabel from mem(firstobs=&j obs=&j);
 quit;
```

```
*passing all the combination of non-printable/special characters and replacing character to a macro variable;

proc sql noprint;
  select compfin into :tt from exc1_imp3;
quit;

%let prevar=%nrbquote(&tt);

*Passing the total number of unique combinations of non-printable/special characters and replacing character to a
 macro variable;

%let precnt=%sysfunc(countw(%nrbquote(&prevar),*));

data report_&j. (keep=dsname varname npsc updated excelrowno) out.%cmpres(&dsn.) %if &dlabel. ne  %then
%do; (drop=dsname varname npsc updated sub sub1 sub2 j chars_org excelrowno label=&dlabel.);%end;
  %else %do;(drop=dsname varname npsc updated sub sub1 sub2 j chars_org excelrowno);%end; *For keeping
 the input dataset label unchanged in the output dataset;

  set in.&dsn.;
  array chars[*] _character_; *Defining array for identifying all the character variables in a dataset;

  %do i=1 %to &precnt.;
    length sub sub1 sub2 npsc $ 32767;*Assigning length to the maximum;
    sub=scan("&prevar",&i,"*");
    sub1=scan(sub,1,"=");
    sub2=scan(sub,2,"=");

    do j = 1 to dim(chars);
      chars_org = chars(j);
      if strip(sub2) in ("nospace") then do; *Removing non-printable characters without space;
        if strip(sub1)="'0D0A'x" then chars(j)=transtrn(chars(j),'0D0A'x,trimn(' '));
          else if strip(sub1)="'0D'x" then chars(j)=transtrn(chars(j),'0D'x,trimn(' '));
          else if strip(sub1)="'0A'x" then chars(j)=transtrn(chars(j),'0A'x,trimn(' '));
           else if strip(sub1)="'09'x" then chars(j)=transtrn(chars(j),'09'x,trimn(' '));
           else chars(j)=compress(chars(j),strip(sub1)); *Removing special characters without space;
      end;

        else do;
          chars(j)=tranwrd(chars(j),strip(sub1),strip(sub2)); *Replacing of non-printable and special characters;
        end;

      *To identify the removed/replaced non-printable and special characters;

      if chars_org ne chars(j) and chars_org ne "" then do;
        dsname = "&dsn.";
        varname = vname(chars(j));
        npsc = strip(sub1);
        if sub2="nospace" then updated = "(replaced without space)";
          else if missing(sub2) then updated = "(replaced with space)";
          else updated=strip(sub2);
        if &i=1 then excelrowno=1;
          else excelrowno=&i.;
          label dsname="Dataset Name" varname="Variable Name" npsc="Non Printable & Special Character"
          updated="Replaced Character"
          excelrowno="Excel Row Number";
          output report_&j.;
        end;
      end;
  %end;
  output out.%cmpres(&dsn.);
run;
```

```
  *Stacking the removed/replaced non-printable and special characters from each datasets in the input library;

  data report;
    retain dsname varname npsc updated excelrowno;
    set report:;
  run;

  proc sort data=report nodupkey;
    by dsname varname npsc updated;
  run;
%end;

 *To create the pdf report of removed/replaced non-printable and special characters;

 ods pdf file="&outpath./Non Printable & Special Characters Report.pdf";
 title "Non printable & Special Characters Report";
   proc print data=report label noobs;
   run;
 ods pdf close;

 proc datasets library=work kill;run;quit;

%mend scnprpt;

****End of Program****;
```

The macro will create a macro variable containing a combination of all the non-printable/special characters and their corresponding replacement characters. Each combination will then be iterated and checked across all character variables in all datasets within the specified library. At the end, a PDF report will be generated, as shown in the screenshot below, listing the updated datasets and variables along with details of the non-printable and special characters that were replaced.

Non printable & Special Characters Report 05:24 Friday, October 11, 2024

Dataset Name	Variable Name	Non Printable & Special Character	Replaced Character	Excel Row Number
CM	CMTRT	®	(replaced with space)	40
DV	DVTERM	̇	'	14
T14_3_1_2_V	CHARACTERISTIC	'0A'x	(replaced without space)	125
TS	TSVAL	̇	'	14
TS	TSVAL1	½	(replaced with space)	55
TS	test	̈	"	15

R CODE:

The R Code creates 2 user-defined functions**: replace_special_characters** and **convert_sas_datasets**. In **replace_special_characters,** we have 3 arguments being defined where '**dataset1**' represents the Excel file or the metadata created, '**dataset2'** are the '**dataframes**' where the special characters will be replaced and '**dfname'** is the name of the dataset being processed.

Within the for loop, each function iterates over each row of dataset1. For each row, the special character and its replacement character are extracted.

```
rm(list = ls())

library(haven)

library(writexl)

library(readxl)

library(gridExtra)

library(grid)

replace_special_characters <- function(dataset2, dataset1, dfName) {

replacements_made <- c()

# For each row in df1

for (i in seq_len(nrow(dataset1))) {

    special_char <- dataset1$special_char[i]

    replacement_char <- dataset1$replacement_char[i]
```

Further, the function **replace_special_characters,** uses **lapply()** and **sapply()** in order to apply a replacement function to each column in '**dataset2'** and to check if any special characters exist. If any match is found, then the special character is replaced by the replacement character using the **gsub()** function.

```
  dataset2 <- as.data.frame(lapply(dataset2, function(column) {

     sapply(column, function(x) {

       if (!is.na(x) && grepl(special_char, x)) {

 # Print

  replacements_made <<- c(replacements_made,  paste0("Replaced '", special_char, "' with '", replacement_char,

                                  "' in value: ", x))

 # Replacement

         gsub(special_char, replacement_char, x)

       } else {

         x

       }

     }, USE.NAMES = FALSE)

  }))
```

```
 }
```

After all the columns are being processed, the function checks if any replacements are made and if so prints them to the console and the updated version of dataset2 will be returned.

```
# Print the replacements that were made

  if (length(replacements_made) > 0) {

    cat("The following replacements were made:\n")

    cat(paste(replacements_made, collapse = "\n"))

  } else {

    cat("No replacements were made.\n")

  }

 # Return the updated dataset2

  return(dataset2)

}
```

Now the Excel file will be read from the folder and if there are no expected values in the file it gets replaced with empty strings otherwise get replaced with the predefined value provided.

In the second function **convert_sas_datasets,** we have 2 arguments where '**source**_dir' represents all the input datasets, and '**target_dir**' represents the modified datasets. This function reads all the SAS datasets from the input folder, processes them and returns the output in Excel and xpt files in the target directory. If the file name has a length greater than 8, then it creates only the Excel file.

```
# Read meta Data

excl_npsc <- read_excel("www/excl_npsc.xlsx")

excl_npsc[is.na(excl_npsc)] <- ""

library_path <- "G:/Working Folder/xxx/PHUSE Paper /SAS Code/Input"  # Specify the path to the library directory

convert_sas_datasets <- function(source_dir, target_dir) {

  # Get a list of all files in the source directory

  all_files <- list.files(source_dir, full.names = TRUE)

  # Filter the list to include only SAS datasets (.sas7bdat files)

  sas_files <- all_files[grepl("\\.sas7bdat$", all_files)]
```

```
 # Create the target directory if it doesn't exist

 if (!dir.exists(target_dir)) {

   dir.create(target_dir)

 }
```

Moving forward, each of the SAS datasets will be identified from the source directory and the file name will be extracted using the **basename()** function and the earlier defined function **replace_special_characters** will be used to replace the special characters with the replacement characters predefined in the Excel. Here the **names()** is used to prefix each column name in the processed dataset with the dataset name.

```
# Read and process each SAS dataset

 result_dataframes <- list()

 for (i in seq_along(sas_files)) {

   file_path <- sas_files[i]

# Read the SAS dataset

   dataset <- read_sas(file_path)

    file_name <- tools::file_path_sans_ext(basename(file_path))

  # Remove special characters

   processed_dataset <- data.frame(replace_special_characters(dataset, excl_npsc, file_name))

 # Get the dataset name

   dataset_name <- tools::file_path_sans_ext(basename(file_path))

 # Assign names to the dataframe columns

   names(processed_dataset) <- paste0(dataset_name, "_", names(processed_dataset))

 # Save the dataframe as an Excel file

   excel_file <- file.path(target_dir, paste0(dataset_name, ".xlsx"))

   write_xlsx(processed_dataset, excel_file)

  # Save the dataframe as an XPT file

   sas_file <- file.path(target_dir, paste0(dataset_name, ".xpt"))

  processed_dataset %>% xportr_write(sas_file,  strict_checks = FALSE)
```

```
  # Store the dataframe in the result list
    result_dataframes[[dataset_name]] <- processed_dataset
  }
 # Return the resulting dataframes
  return(result_dataframes)
}
# Specify the source directory containing the SAS datasets
source_dir <- "G:/Working Folder/xxx/PHUSE Paper /SAS Code/Input"
# Specify the target directory to save the converted datasets
target_dir <- "G:/Working Folder/xxx/PHUSE Paper /R Code/Project/out"
# Call the function
converted_dataframes <- convert_sas_datasets(source_dir, target_dir)
```

CONCLUSION

Ensuring data quality in clinical trial deliverables is paramount and addressing non-printable and special characters is a critical aspect of this process. This paper demonstrates the effective use of SAS macros and R functions to identify and manage these characters, either by removal or replacement, across all datasets in a library. The generated report, detailing the identified and replaced characters along with their dataset and variable names, provides a clear and comprehensive understanding for users. By sharing the SAS and R codes, this paper equips users with practical tools to maintain the integrity and accuracy of their clinical trial reports.

CONTACT INFORMATION

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

Eldho Alias
Catalyst Clinical research
Email: eldho.alias@catalystcr.com

Vishal V Nair
Catalyst Clinical research
Email: vishal.nair@catalystcr.com

Pooja Pradeep Pillai
Catalyst Clinical research
Email: pooja.pradeep@catalystcr.com