SESUG 2022 Paper 197

Learning Fun(damental) Character String Cleaning and Parsing Methods in SAS®!

William Zachary Smith, RTI International

ABSTRACT

Cleaning and parsing character strings is an important step in data processing activities to allow the data to be as analytically useful as possible. This often requires character strings to be summarized into discrete numeric categories (coded) or requires using character strings to fuzzy match records between datasets. In order to do these activities, a number of cleaning and parsing techniques are needed. This paper will examine different SAS functions and custom macros to clean character string data, remove or replace unwanted characters and symbols, and parse single character strings into separate strings based on user-defined characters and delimiters. The functions and macros explored within are easy to understand and fundamental for all SAS users at any skill level.

INTRODUCTION

RTI International is responsible for many large-scale surveys across the country, and delivering a high-quality data product to clients requires various data processing steps that often utilize many different character string cleaning and parsing methods in SAS. Coding is one of these steps that occurs on many different large-scale surveys, and requires a multitude of string manipulation techniques. Here, coding refers to the process of summarizing character strings into discrete numeric categories.

For example, one of many surveys that requires a large amount of coding is the Survey of Earned Doctorates (SED). Contracted by the National Science Foundation (NSF), the SED survey is an annual census of all individuals receiving a research doctorate from an accredited U.S. institution in a given academic year, and collects information such as educational history, demographic characteristics and postgraduation plans. Within the educational history section, respondents are asked to provide their field of study (FOS) information for every valid degree they have earned from an academic institution. Respondents are able to select from a list of FOS codes and corresponding labels provided by the SED, or they are able to provide a string for their FOS if the respondent feels the FOS codes and labels do not accurately reflect their FOS. It is then the responsibility of SED staff at RTI International to code every FOS string received from respondents to the list of official FOS codes and labels provided by the SED.

Because the volume of FOS strings received from doctorate recipients every year is high, the coding process is broken into two steps – autocoding and manual coding. Autocoding is the first step in this process, and involves utilizing a vast array of character string cleaning and parsing techniques in SAS to match the FOS strings from respondents to an FOS label. It is of high importance to autocode as many FOS strings as possible so that the burden of manual coding, which involves staff manually reviewing and coding every FOS string that could not be autocoded, is as low as possible. This paper will examine the different character string cleaning and parsing techniques used in various FOS autocoders at RTI to match verbatim strings from respondents to official FOS labels, and reduce the burden of manual coding. The methods used in this paper are simple, easy to learn, and relevant to SAS users of all levels.

CLEANING CHARACTER DATA

One major obstacle in being able to match respondent FOS string data to an official list of FOS labels is that respondent data is rife with misspellings, non-alpha-numeric characters, and inconsistent capitalization. In order to clean up this data, the FOS autocoder uses four different SAS functions – the FIND function, the TRANWRD function, the UPCASE function, and the COMPRESS function. Below is a small fake data set called "mock_data" that contains mock FOS data that needs to be cleaned.

	respondent_string
1	chemisty
2	computer informaition
3	astrophyics
4	health administraton
5	BiOLogy
6	economics, math, statisstics
7	3-D printing science
8	GENDER and Women studies

Figure 1. Sample of Mock FOS Data That Needs Cleaning

The first step in cleaning the data above involves correcting some common misspellings found throughout multiple years of processing FOS data on multiple studies. This involves using the FIND function and the TRANWRD function. The FIND function is usually used to count how many instances a substring occurs in a string, and is used in the FOS autocoder to identify any strings that may contain common spellings. There are two arguments used in the FIND function– the variable that contains the strings of interest, and then the substring (in this case misspelling) you are looking for. The output of this function is a count representing how many times the specified substring occurs in the string of interest.

The following DATA step code can be used to identify the strings above that contain common misspellings and create an indicator variable called "misspelled" that has a value of "1" when a string contains a common misspelling:

```
data mock_data2;
set mock_data;
if find(respondent_string, 'chemisty' ) ge 1 then mispelled=1;
if find(respondent_string, 'informaition') ge 1 then mispelled=1;
if find(respondent_string, 'phyics') ge 1 then mispelled=1;
if find(respondent_string, 'administraton') ge 1 then mispelled=1;
if find(respondent_string, 'statisstics') ge 1 then mispelled=1;
run;
```

The resulting data set from running this code is then shown in Figure 2.

	respondent_string	mispelled
1	chemisty	1
2	computer informaition	1
3	astrophyics	1
4	health administraton	1
5	BiOLogy	.
6	economics, math, statisstics	1
7	3-D printing science	.
8	GENDER and Women studies	.

Figure 2. Sample of Mock FOS Data With Identified Misspellings

After using the FIND function to identify misspellings, the TRANWRD function is then used to replace the misspellings with the correct spellings. The TRANWRD function in this example utilizes three arguments – the variable containing the string of interest, the misspelling we want to correct, and the correction itself. The following DATA step code is used to implement the correction of these misspellings:

```
data mock_data3;
 set mock_data2;

 if find(respondent_string, 'chemisty' ) ge 1 then
 cleaned_string=tranwrd(respondent_string, 'chemisty' , 'chemistry' );

 if find(respondent_string, 'informaition' ) ge 1 then
 cleaned_string=tranwrd(respondent_string, 'informaition' ,
 'information' );

 if find(respondent_string, 'phyics' ) ge 1 then
 cleaned_string=tranwrd(respondent_string, 'phyics', 'physics');

 if find(respondent_string, 'administraton') ge 1 then
 cleaned_string=tranwrd(respondent_string, 'administraton',
 'administration');

 if find(respondent_string, 'statisstics') ge 1 then
 cleaned_string=tranwrd(respondent_string, 'statisstics',
 'statistics');

 run;
```

The resulting data set from running this code is then shown in Figure 3.

	respondent_string	mispelled	cleaned_string
1	chemisty	1	chemistry
2	computer informaition	1	computer information
3	astrophyics	1	astrophysics
4	health administraton	1	health administration
5	BiOLogy	.	
6	economics, math, statisstics	1	economics, math, statistics
7	3-D printing science	.	
8	GENDER and Women studies	.	

Figure 3. Sample of Mock FOS Data With Corrected Misspellings

Once the common misspellings have been identified and corrected, the strings then need to be standardized by making all strings uppercase, removing spaces, and stripping strings of non-alphanumeric characters. These three steps are achieved using the UPCASE function and the COMPRESS function. The UPCASE function only requires one argument that specifies the string you want to be fully capitalized. The COMPRESS function in this example removes all unwanted characters, and uses three arguments – the variable containing the string of interest, a full list of characters that are allowed to exist in the string, and then a "modifier" argument that specifies how SAS should handle characters that are present in the string of interest, but are not present in the list of characters allowed in the string in the 2nd argument of the function. In this example, the modifier "k" is used to instruct SAS to remove any characters that exist in the string of interest, but are not present in the list of characters allowed in the string. A full list of modifiers available with the COMPRESS function, and their corresponding meanings, can be found on the SAS Help Center website. The following DATA step code is used to create a variable called "matchfld" utilizing the two functions mentioned above:

```
data mock_data4;
set mock_data3;
if cleaned_string='' then cleaned_string=respondent_string;
matchfld=upcase(cleaned_string);
matchfld=compress(matchfld,"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789","k");
run;
```

The resulting data set from running this code is then shown in Figure 4.

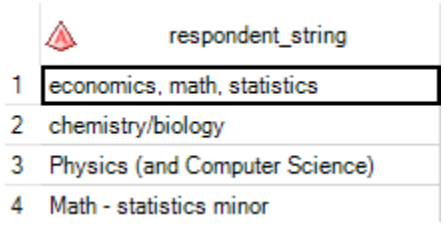
	respondent_string	mispelled	cleaned_string	matchfld
1	chemisty	1	chemistry	CHEMISTRY
2	computer informaition	1	computer information	COMPUTERINFORMATION
3	astrophyics	1	astrophysics	ASTROPHYSICS
4	health administraton	1	health administration	HEALTHADMINISTRATION
5	BiOLogy	.	BiOLogy	BIOLOGY
6	economics, math, statisstics	1	economics, math, statistics	ECONOMICSMATHSTATISTICS
7	3-D printing science	.	3-D printing science	3DPRINTINGSCIENCE
8	GENDER and Women studies	.	GENDER and Women studies	GENDERANDWOMENSTUDIES

Figure 4. Sample of Mock FOS Data With Final Matching Variable

From here, the "matchfld" variable is then used to do direct matches using a simple DATA step merge to the official FOS labels that have undergone similar string standardizing procedures in SAS prior to matching. This pre-cleaning and direct match procedure outlined above acts as the main step for many FOS autocoders at RTI International. Though these examples provided a simple situation for the usage of these four SAS functions, they can be combined and used in varying ways to implement complex character data manipulations and should be considered as building blocks in the repertoire of all SAS programmers dealing with character string data.

PARSING CHARACTER DATA

Another obstacle in being able to match respondent FOS string data to a list of official FOS labels is that sometimes respondents report multiple FOS within a single string. These strings need to be parsed out so that we can identify respondents who obtained a single degree in more than one FOS. This situation most prominently occurs when respondents report their double major bachelor's degree as a single FOS. Figure 5 below shows an example of some mock FOS strings that need to be parsed into separate strings so that they can be each individually matched to an official FOS label.



	respondent_string
1	economics, math, statistics
2	chemistry/biology
3	Physics (and Computer Science)
4	Math - statistics minor

Figure 5. Sample of Mock Double Major FOS Data

As seen in Figure 5, many respondents report a double major where the multiple FOS fields reported can be split out by special character delimiters (commas, dashes, parenthesis, etc.). A custom macro titled %DELIM is used to create a "matchfld" variable for each FOS reported by a respondent that can be parsed out by special delimiters. The macro uses the TRANWRD, UPCASE and COMPRESS functions described in the previous section, and also uses the SCAN function.

The SCAN function can be used to parse character string data based on a specific character and uses three arguments in this example– the original string of interest, a numeric constant that specifics the "number of the word" in the character string that you want SCAN to select, and the character that is being used as a delimiter to parse the string into different substrings. "Number of the word", often called *n,* refers to the *n*th "word" from the character string of interest, where a "word" means any section of the string between two delimiters.

The full code for this macro, including the usage of the SCAN function, can be found in Appendix A. The SAS code below then shows multiple callings of the %DELIM macro using the mock data above:

```
%delim (set=mock_data_doublemaj,del=%str(,),label=comma);
%delim (set=mock_data_doublemaj,del=%str(/),label=slash);
```

```
%delim (set=mock_data_doublemaj,del=%str(-),label=dash);
%delim (set=mock_data_doublemaj,del=%str(%(),label=paren);

data mock_data_doublemaj2 ;
set stringsplitcomma
stringsplitslash
stringsplitdash
stringsplitparen;
drop matchfld;
run;
```

Running the four calls of %DELIM macro above, and the proceeding DATA step to combine the results from the four different runs of the %DELIM macro results in the following dataset shown in Figure 6.

	respondent_string	matchfld1	matchfld2	matchfld3
1	economics, math, statistics	ECONOMICS	MATH	STATISTICS
2	chemistry/biology	CHEMISTRY	BIOLOGY	
3	Math - statistics minor	MATH	STATISTICSMINOR	
4	Physics (and Computer Science)	PHYSICS	ANDCOMPUTERSCIENCE	

Figure 6. Sample of Mock Double Major FOS Data After Parsing

From here, each "matchfld" variable shown above in Figure 6 can then be used to match against official FOS labels via a merge in a DATA step so that each FOS reported by respondents in a single string can be identified and coded individually.

CONCLUSION

As shown, SAS has a multitude of built-in character manipulation functions that should be imperative to any SAS programmer who has to process character string data by cleaning, editing, or parsing string-based data. These functions can all be used in a variety of ways to achieve multiple goals, and can be built upon each other to achieve more complex and advanced character string manipulations all within a single DATA step. This paper is limited in its scope so that its contents are accessible and easily digestible for novice SAS users, and does not cover a wide array of other SAS string functions that deal with concatenating different character strings, stripping and trimming unneeded spaces, and converting numeric data into character data.

APPENDIX A

```
%macro delim (set=, del = , label = );
data stringsplit&label.;
set &set.;
delim ="&del.";
if delim not in ("&","+") then do;
matchfld = tranwrd(respondent_string,"&","AND");
matchfld = tranwrd(matchfld,"+","AND");
end;
drop delim;
```

```
matchfld = upcase(matchfld);
matchfld1 = scan(matchfld,1,"&del.");
matchfld2 = scan(matchfld,2,"&del.");
matchfld3 = scan(matchfld,3,"&del.");
matchfld1 =
compress(matchfld1,"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789","k");
matchfld2 =
compress(matchfld2,"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789","k");
matchfld3 =
compress(matchfld3,"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789","k");
if matchfld2^='';
run;
%mend delim;
```

CONTACT INFORMATION

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

William Zachary Smith
RTI International
919-541-6987
wzsmith@rti.org