Paper CC138

The COMPRESS Function: Hidden Superpowers

Pamela L. Reading, Rho Inc.

ABSTRACT

Most SAS® programmers rely on the COMPRESS function for cleaning up troublesome string data. The many uses of the third ‘modifier’ argument, added in Version 9, may not be as familiar. This paper will present a quick summary of the options available and examples of their use. It will conclude with an unusual application of the ‘keep’ option to reorder characters within a string.

INTRODUCTION

If you have been programming in SAS® for any length of time, you’ve probably used the COMPRESS function to clean up input data or extract a useful tidbit from a string or variable name. Unless you are the type of person who carefully reads the documentation for each update, or browses through many conference papers, you might not be aware of the hidden superpowers you can now use. With Version 9, SAS added a third ‘modifier’ argument to the COMPRESS function – and it can do some amazing things:

- Supercharge ‘normal’ compression
- Mutate (modify or invert) ‘normal’ compression
- Rearrange characters in a variable

SUPERCHARGED COMPRESSION

Most SAS programmers first learn COMPRESS when they need to remove extraneous spaces or other troublesome characters from strings. We try to work with a string, find things we don’t want, and remove them:

```
COMPRESS(OLDSTRING,'/?!.,#')
```

This works, until the next data arrives with new extraneous characters and our code complains or crashes again. We put additional problem characters in the second argument:

```
COMPRESS(OLDSTRING,'/?!.,#$-*')
```

Now things run again, but new data may introduce additional problems and the cycle repeats….

The third argument, used wisely, can minimize this repeated modification. There are a number of options that cover whole classes of characters, so we can write generalized compression statements. Using these options alone or in combination with each other as the third argument generalizes the normal compress behavior, that is, removes the whole category from the string. Table 1 below presents the complete list of options that cover generic classes of characters.

For example, the code we had above can be generalized as:

`COMPRESS (OLDSTRING, ,'P')` – removes all punctuation (note missing second argument)

You can use as many options as you like in combination. You can leave the second argument blank if the general option is sufficient, or you can also include specific items in the second argument.

`COMPRESS (OLDSTRING,'0','AP')` – removes all punctuation, all alphabetic characters AND the digit ‘0’

Argument	**Meaning**	**Notes**
A	Alphabetic Characters	Upper and lower case and accented letters
C	Control Characters	Everything except printables
D	Digits	
F	Underscore and English Letters	No accented letters, no other punctuation
G	Graphics	Includes letters, digits, some punctuation as well as many graphic bits
H	Horizontal Tab	
L	Lowercase Letters	
N	Digits, Underscore and English Letters	No accented letters, no other punctuation
P	Punctuation	Includes some mathematical symbols, currency, copyright, etc.
S	Space Characters (Blanks, Tabs, Carriage Return, Line Feed, Form Feed)	
U	Uppercase Letters	
W	Printable Characters	Everything except control characters
X	Hexadecimal Characters	Digits and letters A-F

Table 1. Third Argument Options for Generic Compression by Category

Based on the table, if our subsequent code is designed to only handle alphabetic characters and digits, then

`COMPRESS (`***`OLDSTRING`***`, ,'PCS')` will reliably give us just what we want.

MUTANT COMPRESSION

There are also third argument options that 'mutate', that is, modify or invert, the usual behavior of the function.

- I - ***ignore*** the case of characters in argument 2 (the ones to be removed or kept). Note that most of the options in Table 1 (except L and U) include both cases. But if we need to derive a specific list of characters for removal, based on changing data, this option gives us more power. For example, instead of

  `COMPRESS(`***`OLDSTRING`***`,'ABCabc'),` we use `COMPRESS(`***`OLDSTRING`***`,'AbC',I)`
- K – ***keep*** the characters or categories specified, rather than removing them. Using 'K',

  `COMPRESS(`***`OLDSTRING`***`,,'KA')` is equivalent to `COMPRESS(`***`OLDSTRING`***`,,'PCS').`

The K option makes using COMPRESS a much more programmer-friendly experience. Instead of anticipating all the possible problem data you might encounter, you just need to decide which categories of data you want to keep. A common use of this option would be to clean out characters that would cause unexpected print behavior when dumping 'raw' text to a listing. `COMPRESS(`***`OLDSTRING`***`,,'KW')` will keep 'writable' characters. Note that some might print as blanks, but none of them should cause mysterious line feeds, page breaks, and other unfortunate output.

The other two options can be extremely useful if you are using COMPRESS with variables as arguments:

- T – ***trim*** trailing blanks in the first and second arguments
- O – ***once*** - limits the processing of the second and third arguments to one (speeds up processing in a loop if the arguments don't change once established)

SUPERPOWERS IN ACTION

The following examples are loosely based on tasks required for analysis of viral resistance data. The assumptions are that variables are named with a prefix (a combination of letters indicating the protein) and numbers (amino acid positions on that protein). Each variable can contain either blank, a limited number of symbols (?*^/) or one or more letters, the letters being single character amino acid abbreviations. Multiple letters, if present, are separated by slash (A/M/T). A given letter only appears once per variable.

A typical record might look something like this:

NXA0001	NXA0002	NXA0003	……….	NXA0245	NXA0246	NXA0247
A	?			D/X	F	C/Y/G

Table 2. Sample Record

Generalized array processing

Assume we want general code that would handle different proteins, each with a different number of variables. We want to output one record per variable (amino acid location) for any location that contains mixed (multiple character) results, stripping out any punctuation. For the example data given above, position NXA0247 contains C/Y/G, so we want the output record to look like this:

VARVAL = 'CYG' ; VARNUM='0247';

The following code will work for any case simply by changing the prefix used in the ARRAY statement.

```
DATA TEST (KEEP=USUBJID VARNUM VARVAL);
 SET SESUG.TESTDATA;

ARRAY PROTEIN(*)  NXA: ;   * creates an array of all the variables ;

DO J = 1 TO DIM(PROTEIN) ;
  VARVAL=COMPRESS(PROTEIN(J),,'KAO');            * ignore blanks and symbols;
    IF LENGTH(VARVAL)  > 1 THEN DO;              *if more than one letter;
    VARNUM=COMPRESS(VNAME(PROTEIN(J)),,'KDO');   *get the location label;
    OUTPUT;
  END;
END;

RUN;
```

Reorder characters within a string

Assume in the case above, we wanted the characters in the output variable to be in alphabetical order for further processing. This is not an easy thing to accomplish in SAS without deconstructing the string into component letters in an array. But the clever use of COMPRESS with an unexpected flip-flop of parameters makes the task easy:

```
COMPRESS('ABCDEFGHIJKLMNOPQRSTUVWXYZ',OLDSTRING,'K');
```

This would produce VARVAL = 'CGY';

Note that this code will remove duplicate letters if they occur in `OLDSTRING.`

CONCLUSION

The COMPRESS function has been greatly enhanced by the addition of the third 'modifier' argument. The superpowers will help you to clean up input data, write more generalized code, or even perform a little magic!

.

REFERENCES

Murphy, William C. 2006. “Squeezing Information out of Data”. *Proceedings of the Thirty-First Annual SAS® Users Group International Conference*. Cary, NC: SAS Institute.
Available at http://www2.sas.com/proceedings/sugi31/028-31.pdf.

ACKNOWLEDGMENTS

Many thanks to David Scocca, resident Rho® SAS guru, who pointed me to the final trick for reordering letters in a variable. It saved me many hours of programming angst.

CONTACT INFORMATION

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

Pamela L. Reading
Rho, Inc.
6330 Quadrangle Drive
Chapel Hill, NC 27517
919-595-6236
pamela_reading@rhoworld.com