Paper 1752-2015

Fixing Lowercase Acronyms Left Over from the PROPCASE Function

Joe DeShon

ABSTRACT

The PROPCASE function is useful when cleansing a database of names and addresses in preparation for mailing. But it doesn't know the difference between a proper name (in which proper case should be used) and an acronym (which should be left all upper case). This paper describes an algorithm that determines with reasonable accuracy if a word is an acronym and converts it to upper case if it is.

INTRODUCTION

Most databases store names and addresses in all upper case for ease of storage and retrieval. The database used for the preparation of this paper contained a mixture of veterinary clinics, retail stores, and farm operations. Persons' names are mixed with company names with little to distinguish the two.

SAS provides many functions and tools to adjust the case of the names to something more aesthetic than all upper case. Since every situation is unique, it is impossible for general tools to fix everything correctly. The macro described by this paper goes a long way toward correcting common case mistakes that are left over from using the PROPCASE function.

THE PROBLEM

Most databases store names and addresses in all upper case for ease of storage and retrieval. The database that was used to create and test this process was a mixture of veterinary clinics, retail stores, and farm operations. In many cases, persons' names were mixed with company names with little to distinguish the two.

Examples of customer names:

```
JOHN SMITH
SMITH VETERINARY CLINIC
JBC FARMS
SPRINGFIELD ANIMAL HEALTH CENTER
```

When names and addresses are used where aesthetics is important, such as in a mail-merged document or a mailing list, it is usually desirable to convert the values to proper case, that is, all lower case except for the first letter of each word.

```
name = 'JOHN SMITH';
prop_name = propcase(name);
putlog name= prop_name=;

name=JOHN SMITH prop_name=John Smith
```

But the PROPCASE function doesn't understand context. Therefore, it doesn't understand that acronyms should be left as upper case. If John Smith is a veterinarian, the PROPCASE function yields a strange result:

```
name = 'JOHN SMITH, DVM';
prop_name = propcase(name);
putlog name= prop_name=;

name=JOHN SMITH, DVM prop_name=John Smith, Dvm
```

One solution would be to follow the PROPCASE function with a TRANWRD function to change the offending acronym back to upper case:

```
name = 'JOHN SMITH, DVM';
prop_name = propcase(name);
prop_name = tranwrd(prop_name,'Dvm','DVM');
putlog name= prop_name=;

name=JOHN SMITH, DVM prop_name=John Smith, DVM
```

That works, but it requires you to anticipate every possible acronym in the data and write code that looks for and corrects that condition.

A better solution would be if an algorithm could determine with a reasonable degree of accuracy if a given word is an acronym and if it should be rendered as upper case or proper case.

THE BETTER SOLUTION

A word frequency study was done on the names and addresses in the test database. Two interesting patterns emerged:

- Words that contained a vowel were most likely best rendered in proper case.
- Words that contained only consonants were most likely best rendered in upper case.

Of course, there are exceptions. But it was decided to write a routine to look for words without vowels that should be rendered as upper case. The exceptions would be easier to manage than trying to anticipate every possible acronym to determine which case it should be in.

This paper describes a macro that can be called from a variety of programs. The macro takes one parameter: a string variable that has previously been changed to proper case. The macro scans the variable, looking for individual words that contained only consonants. Those words are converted to upper case.

PASSING THE PARAMETER

The macro is named %upper_consonants.sas. It receives one parameter: the variable name to be processed.

```
%macro upper_consonants(whatvar);
%mend;
```

THE SCAN FUNCTION FINDS THE WORDS

The SCAN function is used to isolate the individual words in the string. If the SCAN function returns a non-blank value, it has isolated a word. If the returned string is blank, we have reached the end of the variable and there is no need for further processing

```
%macro upper_consonants(whatvar);
%do i = 1 %to 100;
  if scan(&whatvar,&i) = ' ' then do;
    end;
  else
  <check to see if it contains only consonants>
%end;
%mend;
```

Notice that this code generates a lot of SAS code, since it loops 100 times at compile time -- generating new SAS code with every loop. For that reason, it is best to run this macro with the options NOSYMBOLGEN and NOMPRINT, to prevent the creation of a very large SAS log.

Since the code is created at compile time -- rather than dynamically with each invocation of the macro -- it is necessary to limit to the number of iterations. The macro has an arbitrarily-large limit of 100; it will work only with data of no more than 100 individual words. Since the logic escapes when it fails to find a word, it runs very quickly.

DETERMINE IF THE WORD CONTAINS NO VOWELS

The macro uses the FINDC function with the “i” option to scan for vowels.

```
if not (findc(scan(&whatvar,&i),'AEIOUY','i')) then ...
```

The FINDC function finds an occurrence of any of the letters in the second parameter. The “i” option finds the vowel regardless of the case.

The “not” logic may seem awkward, but it’s easier to search for six vowels than for twenty consonants.

For the purpose of this paper, the letter “Y” is considered a vowel; the letter “W” is not.

THE SUBSTR FUNCTION ISOLATES THE LETTERS TO BE CONVERTED TO UPPER CASE

The SUBSTR function is unique in that it can be place on the left side of an equation. So the substring of the target variable can be converted to upper case if it can be isolated.

The syntax is straightforward:

```
substr(&whatvar,begin,length) = upcase(scan(&whatvar,&i));
```

The macro needs to fill in the beginning position and the length. The beginning position can be determined by using the FINDW function to search for the word found by the SCAN function:

```
begin = findw(&whatvar,trim(scan(&whatvar,&i)))
```

The length is simply the length of the current word:

```
length = length(trim(scan(&whatvar,&i)))
```

The upper case version (on the right of the equation) is the upper case of the word:

```
upcase(trim(scan(&whatvar,&i)))
```

Putting it all together gives a scary-looking, but functional equation:

```
substr(&whatvar,
       findw(&whatvar,trim(scan(&whatvar,&i))),
       length(trim(scan(&whatvar,&i))))
       = upcase(trim(scan(&whatvar,&i)));
```

THE CODE SO FAR

Most of the code has now been written. When the macro was written, the first version tested looked like this:

```
%macro upper_consonants(whatvar);
%do i = 1 %to 100;
  if scan(&whatvar,&i) = ' ' then do;
    end;
  else
    if not (findc(scan(&whatvar,&i),'AEIOUY','i'))
      then do;
        substr(&whatvar,
               findw(&whatvar,trim(scan(&whatvar,&i))),
               length(trim(scan(&whatvar,&i))))
                 = upcase(trim(scan(&whatvar,&i)));
        end;
    %end;
%mend;
```

TESTING AND HANDLING EXCEPTIONS

This code works very well, and incredibly fast, considering the extensive character manipulation involved.

The next step is to determine under which circumstance the code does not work, and figure out the best way of handling those problems

SPECIAL HANDLING OF ORDINALS

In one special case, ordinals (not spelled out, but those with numbers) confused the algorithm. For example, after the PROPCASE function and the %upper_consonant macro:

```
12TH STREET BANK
```

is rendered as:

```
12TH Street Bank
```

The macro sees that the first "word" -- actually the ordinal number "12TH" -- contains no consonants, so the macro converts it to upper case. An exception needs to be made. If the word is a numeric ordinal, the macro should leave it unchanged.

Although there are an infinite number of ordinals, there are a finite number of word endings that generally identify ordinals. Simply by examining the last three bytes of the word for just a few strings will find every ordinal:

```
if (upcase(subpad(scan(&whatvar,&i),
                  length(scan(&whatvar,&i))-2,
                  3)) in ('1ST'
                          '1TH'   /** Eleventh   **/
                          '2TH'   /** Twelfth    **/
                          '2ND'
                          '3RD'
                          '3TH'   /** Thirteenth **/
                          '4TH'
                          '5TH'
                          '6TH'
                          '7TH'
                          '8TH'
                          '9TH'
                          '0TH'
                          ))
      then do;
        end;
```

The SUBPAD function is used instead of the SUBSTR function, because it automatically pads words with spaces if they are shorter than the string that is being looked for.

Now the rendering is correct:

```
12th Street Bank
```

EXCEPTIONS THAT SHOULD BE LOWER CASE

There are some exceptions where the macro misidentifies words as acronyms when they are actually abbreviations that should be in proper case. This name:

```
DR JOHN SMITH, DVM
```

is rendered as:

```
DR John Smith, DVM
```

The macro accurately identifies "DVM" as an acronym that should be in upper case. But "DR" isn't an acronym; it's an abbreviation for "Doctor" and should be in proper case.

A list of exceptions must be created. If the word is in that list, the macro should bypass the word, leaving it proper case.

For ease of maintenance and for code-readability, the list is in a macro variable. The list can be compiled through inspection of your specific data. A full list would be quite comprehensive; only a portion is shown here:

```
%let make_proper =
  'DR' 'DRS' 'JR' 'MR' 'MRS' 'RD' 'SVC' 'ST' 'WM';
```

EXCEPTIONS THAT SHOULD BE UPPER CASE

In some instances, the reverse is true: a word contains a vowel, but should be rendered in upper case. Another exception list is for that purpose. Here is a short version of the list:

```
%let make_upper =
  'ABC' 'AFB' 'APO' 'DBA' 'II' 'III' 'IV' 'NE' 'SE' 'USDA' 'XYZ';
```

THE FULL CODE

After inserting the two exception lists, the full code is now complete:

```
%macro upper_consonants(whatvar);

%let make_proper =
  'DR' 'DRS' 'JR' 'MR' 'MRS' 'RD' 'SVC' 'ST' 'WM';

%let make_upper =
  'ABC' 'AFB' 'APO' 'DBA' 'II' 'III' 'IV' 'NE' 'SE' 'USDA' 'XYZ';

%do i = 1 %to 100;
  if scan(&whatvar,&i) = ' ' then do;
    end;
  else
  if upcase(scan(&whatvar,&i)) in (&make_proper) then do;
    end;
  else
  if (upcase(subpad(scan(&whatvar,&i),
                    length(scan(&whatvar,&i))-2,
                    3)) in ('1ST'
                            '1TH'   /** Eleventh   **/
                            '2TH'   /** Twelfth    **/
                            '2ND'
                            '3RD'
                            '3TH'   /** Thirteenth **/
                            '4TH'
                            '5TH'
                            '6TH'
                            '7TH'
                            '8TH'
                            '9TH'
                            '0TH'
                            ))
      then do;
        end;
  else do;
    if
     (upcase(scan(&whatvar,&i)) in (&make_upper))
                OR
      (NOT (findc(scan(&whatvar,&i),'AEIOUY','i')))
         then do;
           substr(&whatvar,
                   findw(&whatvar,trim(scan(&whatvar,&i))),
                   length(trim(scan(&whatvar,&i))))
                     = upcase(trim(scan(&whatvar,&i)));
           end;
    end;
  %end;
%mend;
```

The exception list is actually much larger than what is shown in this paper. You should periodically scan your records, searching for other exceptions to add to the list. Since every company's data is different, every exception list will be different for different circumstances.

CONCLUSION

Macros such as this can be a powerful tool to aid in your data cleansing. Every set of data is presents itself with unique challenges. The specific set of tools and how they are used will vary greatly, depending on the circumstances.

Since most name and address data is dirty, you may choose to live with some inaccuracies. For example, without sophisticated context evaluation, it's difficult to tell if "CT" is an abbreviation for "Connecticut" (in which case it should be upper case) or for "Court" (in which case it should be proper case).

Also, some conjunctive words such as "and" and "of" should be lower case. Those may be dealt with in another macro.

Having the right tools available -- and knowing under which circumstances to use them -- is an important part of any data cleansing project.

CONTACT INFORMATION

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

Joe DeShon
joedeshon@yahoo.com