Paper CC-189

When PROPCASE Isn't Proper: A Macro Supplement for the SAS® Function

Alex Buck, Rho®

ABSTRACT

Formatting is important for clarity when reporting descriptive data such as country or subject status. In comes PROPCASE to save the day. "Indonesia" and "Protocol Violation" are certainly easier to read than their UPCASE counterparts. However, PROPCASE is limited. It will convert every word in the argument without exceptions, creating phrases such as "United States Of America" and "Withdrawal By Subject". These distractions can sometimes be anticipated and fixed, but as new data are collected there is a risk of a renegade "To", "And" or "Out" hiding somewhere. This paper introduces a macro supplement for PROPCASE which contains a default list of common lowercase words and the option to add or exclude words at the user's discretion. With this macro, the user will specify lowercase words in an efficient and dynamic way, allowing confidence in formatting across deliveries.

INTRODUCTION

There are three considerations when setting a character string to use PROPCASE: lowercase exceptions, uppercase exceptions, and the first word exception. Lowercase exceptions are reserved for articles and prepositions while uppercase exceptions can include acronyms (HIV, UNC). The first word exception reflects a rule regarding titles where the first word always has the first letter uppercase. This will override any lowercase exceptions when used. For the purposes of this article, the term 'propcase' will indicate that the first letter of a word is uppercase and all subsequent letters are lowercase. This term will be used as an adjective (propcase) to describe a word and as a verb (propcasing, propcased) to describe the action of setting the first letter to be uppercase and all subsequent letters to be lowercase in a word. The purpose of this macro is to allow the user to include any of the three considerations when propcasing a character string. The user is encouraged to review documentation on regular expressions, but this is not a requirement for utilizing this macro.

THE MACRO

The macro has 5 input values:

- Var: The variable to be propcased. This is a required input.
- ExLow: A list of words to be excluded from the lowercase exceptions.
- AddLow: A list of words to be added to the lowercase exceptions.
- FFlag: A flag to indicate if the first word of the character string should be propcase. The default value is Y, to indicate that the first word of the character string should be propcase.
- UpList: A list of words for the uppercase exception.

The inputs for ExLow, AddLow, and UpList should be enclosed in %STR(). The input for AddLow should also start the list with the delimiter |. The macro has been created to be called within a data step.

The macro call is as follows:

```
%BreakIt (Var=,
          ExLow=,
          AddLow=,
          FFlag=Y,
          UpList=);
```

The macro will output a single variable, &Var._Prop, which will be the propcased version of the original variable with all user-specified exceptions.

THE MACRO PROCESS

The first step of the macro is the build the list of lowercase exceptions. The macro starts with list of lowercase exceptions referred to as ORGLOWLIST. This list includes a, an, and, at, but, by, down, for, in, of, on, or, out, over, past, so, the, to, up, with, and yet. If the input variable EXLOW is non-missing, PRXCHANGE will be used to search for any words included in both the EXLOW and ORGLOWLIST and remove them from ORGLOWLIST. Once all user-specified words are excluded, any user-specified words in AddLow are added to the list of lowercase exceptions.

```
%let LowList='';
%let OrgLowList=%str(a|an|and|at|but|by|down|for|in|of|on|or|out|over|past
                     |so|the|to|up|with|yet);
%put Orginal LowList= &OrgLowList;

%if &ExLow ne %then %do;
    %let LowList =
          %sysfunc(prxchange(s/\|(&ExLow)\b|\b(&ExLow)\|//i,-1,&OrgLowList));
    %put Lowlist with exception: &LowList;
%end;

%else %do;
    %let LowList=&OrgLowList;
%end;


%let LowList=%str(&LowList&AddLow.);
%put LowList with Additions= &LowList;
```

In PRXCHANGE a search and replace is implemented. Note that EXLOW is listed twice in the search pattern, first with the delimiter | before the word and a word boundary after the word, and next with the word boundary before the word and the delimiter | after the word. This allows the function to search and capture 'a' as well as 'yet'. The replacement pattern is left missing to indicate that those words will be removed completely with no replacements. The search is case insensitive as indicated by 'i' after the close of the replacement pattern.

Once the lowercase exception list is created the original variable is propcased while leaving any word on the exception list as lowercase. VAR is converted to lowercase for a consistent starting point. The string is then searched for any word not found in LOWLIST, allowing for word boundaries before and after the word. The first letter in each word is placed in the first capture buffer and all subsequent letters are placed in a second capture buffer. All items in the first capture buffer are converted to uppercase while all items in the second capture buffer are converted to lowercase. Again our search is case insensitive.

```
&Var._Prop =
prxchange("s/\b(?!(?:&LowList)\b)([a-z])([a-z]+)\b/\U$1\L$2/i", -1,lowcase(&Var.));
```

If the first word flag is indicated the first word is propcased regardless of it was previously part of the lowercase exception. The TRANWRD function is utilized to replace the first word of &VAR_PROP with its propcased version. The uppercase exceptions are then implemented. Any word found from UPLIST is then set to be uppercase.

```
%if &FFlag=Y %then %do;
    &Var._Prop=
       tranwrd(&Var._Prop,scan(&Var._Prop,1,' '),propcase(scan(&Var._Prop,1,' ')));

    %if &UpList ne %then %do;
          &Var._Prop=prxchange("s/\b(&UpList.)\b/\U$1/i",-1,&Var._Prop);
    %end;
%end;

%else %if &UpList ne %then %do;
    &Var._Prop=prxchange("s/\b(&UpList.)\b/\U$1/i",-1,&Var._Prop);
%end;
```

AN EXAMPLE

Let us assume we have a variable which contains the values of 'FOR UNKNOWN REASONS','WITHDREW FOR UNKNOWN REASONS', 'HCV AND/OR HIV', 'HCV OR HIVES', and 'ASTHMA ATTACK'. We want to propcase our variable while leaving 'HCV' and 'HIV' uppercase, having 'AND/OR and 'OR' lowercase, and keeping 'FOR' propcase if it is the first word in the string. Our macro call would be as below:

```
data dat_prop;
    set dat_org;
    %PropIt(Var=MYVAR,
            ExLow=,
            AddLow=,
             FFlag=Y,
            UpList=%STR(HIV|HCV);
run;
```

The table below illustrates the steps taken by the macro and the temporary values of the variable after each step. When we follow the text string 'FOR UNKNOWN REASONS' we see that after the lowercase exceptions are implemented, 'for' is now lowercase. Many readers will agree that the text string 'for Unknown Reasons' is not agreeable. This issue is rectified after implementing the first word exception. Another functionality to note is that while 'and/or' is not specifically listed in the lowercase exceptions; both words are captured in the lowercase exception step. When 'HIV' is listed as an uppercase exception we see that the word 'Hives' is left as propcase and not captured in the exception.

MYVAR	After Lowercase Exceptions	After First Word Exception	After Uppercase Exceptions
FOR UNKNOWN REASONS	for Unknown Reasons	For Unknown Reasons	For Unknown Reasons
WITHDREW FOR UNKNOWN REASONS	Withdrew for Unknown Reasons	Withdrew for Unknown Reasons	Withdrew for Unknown Reasons
HCV AND/OR HIV	Hcv and Hiv	Hcv and Hiv	HCV and HIV
HCV OR HIVES	Hcv or Hives	Hcv or Hives	HCV or Hives
ASTHMA ATTACK	Asthma Attack	Asthma Attack	Asthma Attack

CONCLUSION

The utilization of %PropIt will allow users to propcase text strings efficiently while allowing for exceptions as they see fit. The user is cautioned that overlap between the lowercase and uppercase exceptions (e.g., 'at' and 'AT') will result in unintended uppercase values. Another caution is that macro was also not designed for delimiters other than spaces and may produce unexpected results. Even with these caveats, it is the belief of the author that use of %PropIt will reduce burdensome rework while converting text strings to propcase.

REFERENCES

Childress, Spencer. 2014 "Express Yourself! Regular Expression vs SAS Text String Functions." *Proceedings of the PharmaSUG 2014 Conference*. Available at http://www.pharmasug.org/proceedings/2014/BB/PharmaSUG-2014-BB08.pdf

CONTACT INFORMATION

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

Name: Alex Buck
Enterprise: Rho
Address: 6330 Quadrangle Dr
City, State ZIP: Chapel Hill, NC 27517
E-mail: Alexandra_Buck@Rhoworld.com

APPENDIX 1: ONLINE LOCATION OF MACRO

https://github.com/RhoInc/sas-PropIt