Creating a Mixed Case Format for Monocase Data Using an Input Control Data Set with Proc Format

Lisa A. Soberano, PPD Development, Inc., Morrisville, NC

BIOGRAPHY

Lisa Soberano is a Programmer Analyst at PPD Development, and is pursuing her Master's in Statistics part time. She has a degree in Applied Mathematics from UNC Wilmington. She has served as an instructor at SAS, where she taught Programming I, Proc SQL, and the SAS Access class. Previous pharmaceutical experience includes work at Caremark as a Data Analyst and PharmaResearch as a Statistical Programmer.

ABSTRACT

PROC FORMAT is a useful Base SAS tool for displaying data in a more preferable manner, without altering the data itself. Applying formats to our data is like using “make-up”. We want the data to look its best, whether presenting data to a client or internally.

This article assumes a basic familiarity with the DATA step, PROC FORMAT, and the LIBNAME statement. We will review a commonly overlooked function of PROC FORMAT, which is how to create a format using an input control data set (the CNTLIN= option on the PROC FORMAT statement), as well as how to output formats to a data set (CNTLOUT= option). The example application in this article is also handy for displaying large amounts of monocase characters. We will create a mixed case format for character data that is stored in either all upper or all lowercase. Although we have the use of the UPCASE and LOWCASE functions available through SAS, the code covered in this text will give you a starting point to developing your own mixed case function.

The SAS functions that are incorporated in the mixed case example are UPCASE, LOWCASE, TRANWRD, SCAN, LENGTH, SUBSTR, TRIM, LEFT, COMPRESS, and COMPBL.

INTRODUCTION

Across industries, it is common practice to enter data into a database in monocase. However, the programmer may desire to present the data in a mixed case display. This article will deal with an example where a few hundred data records are stored in one case and the goal is to display the data in mixed case, without creating a new variable or retyping the data by hand on a VALUE statement in PROC FORMAT. To accomplish this, we will give the reader some basic tools on how to develop one's own mixed case function. Although there are many ways to accomplish the same goal, we will present one approach to create mixed case in this article. Also, in the case that an additional permanent variable is not desired on the original data set, we will cover the CNTLIN= and CNTLOUT= options on the PROC FORMAT statement that will allow us to create a format using a special data set created by our mixed case function. The covered techniques will come in handy in many different applications.

It will be assumed the reader has familiarity with the following SAS functions:
UPCASE
LOWCASE
TRANWRD
SCAN
LENGTH
SUBSTR
TRIM
LEFT
COMPRESS
COMPBL

GETTING TO KNOW YOUR DATA

First things first. Let's take a look at the data we will be using for this exercise. Assuming the data is a permanently stored SAS data set, you will need to specify the location of the data in a LIBNAME statement.
As mentioned earlier, we are using data that has been entered in all one case. In this example, the uppercase data collected is a preferred term for concomitant medications of a patient. Our goal may be to display a table that counts the number of concomitant medications within a certain body system, but we want to see the concomitant medication displayed in mixed case. For teaching purposes, we will just work with a simple data set. The data set has two variables, a patient identifier, PATID, and the medication preferred term, MPTERM. Here is a partial view of the CONMED data set:

<table>
<thead>
<tr>
<th>plid</th>
<th>MPTERM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BLOOD CELLS, PACKED HUMAN</td>
</tr>
<tr>
<td>1</td>
<td>CALCIUM</td>
</tr>
<tr>
<td>1</td>
<td>CEFTRIAXONE SODIUM</td>
</tr>
<tr>
<td>1</td>
<td>CISATRACURIUM BESILATE</td>
</tr>
<tr>
<td>1</td>
<td>DIPHENHYDRAMINE HYDROCHLORIDE</td>
</tr>
<tr>
<td>1</td>
<td>DOBUTAMINE</td>
</tr>
<tr>
<td>1</td>
<td>DOXYCYCLINE HYCLATE</td>
</tr>
<tr>
<td>1</td>
<td>EPINEPHRINE</td>
</tr>
<tr>
<td>1</td>
<td>FAMOTIDINE</td>
</tr>
<tr>
<td>1</td>
<td>FENTANYL</td>
</tr>
<tr>
<td>1</td>
<td>FUROSEMIDE</td>
</tr>
<tr>
<td>1</td>
<td>GENTAMICIN SULFATE</td>
</tr>
<tr>
<td>1</td>
<td>HEPARIN</td>
</tr>
<tr>
<td>1</td>
<td>HYDROCORTISONE</td>
</tr>
<tr>
<td>1</td>
<td>IBUPROFEN</td>
</tr>
<tr>
<td>1</td>
<td>INSULIN</td>
</tr>
<tr>
<td>1</td>
<td>LACRI-LUBE</td>
</tr>
<tr>
<td>1</td>
<td>LEVOFLOXACIN</td>
</tr>
<tr>
<td>1</td>
<td>LORAZEPAM</td>
</tr>
<tr>
<td>1</td>
<td>MAGNESIUM</td>
</tr>
<tr>
<td>1</td>
<td>MAGNESIUM SULFATE</td>
</tr>
<tr>
<td>1</td>
<td>METHYLPROPIONOLONE SODIUM SUCCINATE</td>
</tr>
<tr>
<td>1</td>
<td>METRONIDAZOLE</td>
</tr>
<tr>
<td>1</td>
<td>MIDAZOLAM HYDROCHLORIDE</td>
</tr>
<tr>
<td>1</td>
<td>NEUTRA-PHOS</td>
</tr>
<tr>
<td>1</td>
<td>PARACETAMOL</td>
</tr>
<tr>
<td>1</td>
<td>PHYTOMENADIONE</td>
</tr>
<tr>
<td>1</td>
<td>PLASMA PROTEIN FRACTION [HUMAN]</td>
</tr>
</tbody>
</table>

**CREATING MIXED CASE FUNCTION**

Now that we have familiarized ourselves with the data and we know we would like to display our data in a mixed case format, it is necessary to determine how we want to create the mixed case. Let's assume that prepositions will be displayed in lowercase. Also, other than prepositions, let's assume that we want the first letter of a word capitalized and the rest of the word in lowercase.

Because we want prepositions to be lowercase, we need to go ahead and standardize our data so we start with lowercase values. Then we will only need to worry about capitalizing the first letter of words that aren't prepositions. We will use the LOWCASE function, in a DATA STEP, to do this (see Appendix A for complete code). In the case that you know for sure that all the data is in lowercase, you can skip this step and alter the function appropriately.

```
lowterm = LOWCASE(mpterm);
```

Now we need to determine what delimiters will distinguish a new word. Most of the time we assume there will be a space between words. However, words in this CONMED data set are not always separated by spaces. Other delimiters used in this data are forward slashes, dashes, commas, or other characters. It is very important that each programmer determine how their data is stored and how to distinguish new words. We will see these delimiters reappearing later in our code in the
TRANWORD and SCAN functions.

As a simple example, let's say that our data has two words separated by a forward slash, with no space in between. Let lowterm="word1/word2". In the function we create, we will do the following (note: see Appendix A for complete SAS code):

1. Add a space after the forward slash using the TRANWRD function.
   
   $$\text{TRANWRD}(\text{lowterm}, "/", "/" ) \rightarrow \text{word1/ word2}$$

2. This will allow the SCAN function to find each new word in the string. Using a space as the word delimiter, the SCAN function will treat the "/" as part of the value, rather than a delimiter. We do this because the delimiter is "lost" from the value in the use of the SCAN function.
   
   $$\text{SCAN}(\text{lowterm},1,'') \rightarrow \text{word1/}
   
   \text{SCAN}(\text{lowterm},2,'') \rightarrow \text{word2}$$

3. We will then isolate the first letter of each word using the SUBSTR function, uppercase it using the UPCASE function, while the remainder of the word will maintain lowercase.
   
   $$\text{TRIM(LEFT(UPCASE(SUBSTR(wrd1,1,1))}} \rightarrow \text{W}
   
   \text{TRIM(LEFT(SUBSTR(wrd1,2)))} \rightarrow \text{ord1/}$$

4. Using the concatenation operator, ||, concatenate the first word back together. Repeat for word2. Then concatenate word 1 to word2 with a space between each word, recreating your text string. We call this new variable MPREF. Adding the space accounts for terms where the original delimiter was a space to begin with. Recall that the delimiter is lost in the use of the SCAN function.
   
   $$\text{TRIM(LEFT(UPCASE(SUBSTR(wrd1,1,1)))) || TRIM(LEFT(SUBSTR(wrd1,2))) ||' ' ||}
   
   \text{TRIM(LEFT(UPCASE(SUBSTR(wrd2,1,1)))) || TRIM(LEFT(SUBSTR(wrd2,2)))} \rightarrow \text{Word1/ Word2}$$

5. If a space was added to the original term in order to separate words, (in this case we added a space after the forward slash), remove it using the TRANWRD function.
   
   $$\text{TRANWRD(mpref,"/ ","/) \rightarrow Word1/Word2}$$

BUILDING THE DO LOOP

We need to determine the number of words in each string, and use this value in a DO loop, which will be the core of our mixed case function. As a reminder, we need to assure that there is at least one space between each word in our string and we have added spaces where necessary. We can tell how many words are in the string by taking the positive difference between the number of characters in the string, compressing multiple blanks to one blank using the COMPBL function, and the number of characters in the compressed string, eliminating all blanks using the COMPRESS function. This will give us a value one less than the number of words (because there is no space before the first word). So, we add one to this number to give us the accurate number of words in the string. Here is the SAS code to accomplish this:

$$\text{numwrds} = \text{LENGTH(COMPBL(term)) - LENGTH(COMPRESS(term, ' '))} + 1;$$

We will need to pass the number of words in the string as the max value for the DO loop. The logic of the loop should be something like the following:

- Use SCAN function to pick out the first word of the string.
- Use the SUBSTR function to pick out the first letter of the word.
- Capitalize the first letter of the word.
- Concatenate the capitalized first letter back with the rest of the word, which is in lowercase.
- Use the SCAN function to pick out the second word of the string.
- Repeat the process with the second word.
- Concatenate the second word to the first word, with a space in between them.
- Use the SCAN function to pick out the third word of the string.
- Repeat the above process for the third word and for each word in the string

Remember that we want prepositions to be all lowercase letters. So, inside of our loop, we will use conditional logic to skip the
process of capitalizing the first letter of prepositions.

Here is the DO loop used:

```sas
do j = 1 to numwrds;
    wrdp = scan(lowterm,j,' ');
    /* Add more prepositions to the list if your data calls for it. */
    if wrdp not in ('with' 'for' 'and' 'of')
        then do;
        mpref = TRIM(mpref)||" "||TRIM(LEFT(UPCASE(SUBSTR(wrdp,1,1))))
               ||TRIM(LEFT(SUBSTR(wrdp,2)));
    end;
    else do;
        mpref = trim(mpref)||" "||left(trim(wrdp));
    end;
end;
```

Now that the text string has been converted to mixed case, we need to remove any blanks that we added. We can do this using the TRANWRD function again, but in the reverse direction, as demonstrated previously. Also, we can use the same function to re-capitalize any acronyms, roman numerals, or other special cases that our mixed case function did not take into account. Of course, one could incorporate this in other ways.

```sas
mpref = TRANWRD(mpref,"/ ","/");
mpref = TRANWRD(mpref,"Vii","VIII");
mpref = TRANWRD(mpref,"K-lyte","K-Lyte");
mpref = TRANWRD(mpref,"Tpn","TPN");
```

You will notice that we did not use the dash as a delimiter. We did this because our data contained hyphenated words that only required capitalization on the first letter of the hyphenated word. There was one word, "K-Lyte", which needed the second part of the word capitalized, so we took care of that with the TRANWRD function. See Appendix A for the complete data step that was used to create the CM data set, which includes the original variable MPTERM, and the variable MPREF, which stores the mixed case values of MPTERM.

**CREATING THE INPUT CONTROL DATA SET FOR PROC FORMAT**

It may be the case that you do not need to create a format because a new variable is sufficient to get the job done. If that is the case, you may not see how the input control data set is valuable in this example. But the concepts we will cover in this section could be valuable to you in other scenarios.

Now that we know how to use our existing SAS functions to create our personalized mixed case function, we can use this to create an input control data set for PROC FORMAT. First let's define what the input control data set is and how it interfaces with PROC FORMAT.

Usually we think of PROC FORMAT as a procedure that allows us to create formats and informats using a value, invalue, or picture statement. We also have the option to create formats and informats using a preexisting data set that we read into PROC FORMAT using the CNTLIN= option on the PROC FORMAT statement. In order to do this, we will need to create the input control data set. The input control data set requires at least three variables: FMTNAME, START, and LABEL. FMTNAME stores the name of the format, START stores the starting value of the range, and LABEL stores the formatted or informatted value or the name of a standard SAS informat/format. A variable named TYPE is needed if you are creating a character format/informat or a PICTURE format. The value of TYPE would be a character value indicating what type of format/informat you are creating. Possible values are: "C" for a character format, "I" for a numeric informat, "J" for a character informat, "N" for a numeric format (excluding pictures), and "P" for a picture format.

Refer to SAS documentation for information on the other variables that you may create in the input control data set to provide you with more control. If a range is being created, and the range values are noninclusive, there are other variables that are required. However, the case we will study is inclusive and is a character format. Multiple formats can be created using an input control data set, but the observations corresponding to that format must be grouped together. Another thing to keep in mind is that you can still use the value, invalue, or picture statements in conjunction with the CNTLIN= option. However, if one of your statements creates a format/informat of the same name in your input control data set, the statement will take precedence over the same named format/informat in the data set.
Now that we know what variables we need in our input control data set, let’s begin creating them. We will go ahead and sort the CM data set by MPTERM, which isn’t necessary in this case, but is easier on the eye. Also we want to ensure that there are no duplicate records in the data. In the PROC SORT step, we keep only MPTERM and MPREF (dropping all unnecessary variables), get rid of duplicate records, and rename MPTERM to START and MPREF to LABEL.

```sas
PROC SORT DATA = cm(KEEP=mpterm mpref) NODUPKEY
   OUT = cmpref(RENAME = (mpterm=start mpref=label));
   BY mpterm mpref;
RUN;
```

Now we need to create the FMTNAME variable and store this data set in a permanent library. This is easily done in a data step:

```sas
DATA here.cmfmts;
   SET cmpref;
   RETAIN fmtname '$mpterm' type 'c';
   end=start;
RUN;
```

Notice that we use the RETAIN statement. In this data step, the RETAIN statement creates the variables FMTNAME and TYPE. For each record in the CMFMTS data set, the value of FMTNAME will be retained as "$mpterm", and the value of TYPE will be retained as "c". In the case that we want to create more than one format, we would need to reset this value to the new format/informat name for the appropriate records.

A third new variable, END, is created. Since we don’t have a range of values, the END value should be set equal to the START value. In the case that you are also creating a range format/informat to be added to this data set, you would assign START to the beginning of the range and END to the last value in the range. For more details on how to set up ranges using an input control data set, please see SAS documentation. Here are some partial views of the CMFMTS data set:

<table>
<thead>
<tr>
<th>start</th>
<th>label</th>
<th>fmtname</th>
<th>type</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCUZYME</td>
<td>Accuzyme</td>
<td>$mpterm</td>
<td>c</td>
<td>ACCUZYME</td>
</tr>
<tr>
<td>ACETAZOLAMIDE</td>
<td>Acetazolamide</td>
<td>$mpterm</td>
<td>c</td>
<td>ACETAZOLAMIDE</td>
</tr>
<tr>
<td>ACETYLCLYSTEINE</td>
<td>Acetylcysteine</td>
<td>$mpterm</td>
<td>c</td>
<td>ACETYLCLYSTEINE</td>
</tr>
<tr>
<td>ACETYLSALICYLIC ACID</td>
<td>Acetylsalicylic Acid</td>
<td>$mpterm</td>
<td>c</td>
<td>ACETYLSALICYLIC ACID</td>
</tr>
<tr>
<td>ACICLOVIR</td>
<td>Aciclovir</td>
<td>$mpterm</td>
<td>c</td>
<td>ACICLOVIR</td>
</tr>
<tr>
<td>ADENOSINE</td>
<td>Adenosine</td>
<td>$mpterm</td>
<td>c</td>
<td>ADENOSINE</td>
</tr>
<tr>
<td>ALBUMIN HUMAN</td>
<td>Albumin Human</td>
<td>$mpterm</td>
<td>c</td>
<td>ALBUMIN HUMAN</td>
</tr>
<tr>
<td>ALBUMIN HUMAN SALT-POOR</td>
<td>Albumin Human Salt-poor</td>
<td>$mpterm</td>
<td>c</td>
<td>ALBUMIN HUMAN SALT-POOR</td>
</tr>
<tr>
<td>ALPRAZOLAM</td>
<td>Alprazolam</td>
<td>$mpterm</td>
<td>c</td>
<td>ALPRAZOLAM</td>
</tr>
<tr>
<td>AMIKacin</td>
<td>Amikacin</td>
<td>$mpterm</td>
<td>c</td>
<td>AMIKacin</td>
</tr>
<tr>
<td>AMIKacin SULFATE</td>
<td>Amikacin Sulfate</td>
<td>$mpterm</td>
<td>c</td>
<td>AMIKacin SULFATE</td>
</tr>
<tr>
<td>AMINOSYN</td>
<td>Aminosyn</td>
<td>$mpterm</td>
<td>c</td>
<td>AMINOSYN</td>
</tr>
<tr>
<td>AMIODARONE</td>
<td>Amiodarone</td>
<td>$mpterm</td>
<td>c</td>
<td>AMIODARONE</td>
</tr>
<tr>
<td>AMoxicillin</td>
<td>Amoxicillin</td>
<td>$mpterm</td>
<td>c</td>
<td>AMoxicillin</td>
</tr>
<tr>
<td>AMPHOTERICIN B</td>
<td>Ampicillin</td>
<td>$mpterm</td>
<td>c</td>
<td>AMPHOTERICIN B</td>
</tr>
<tr>
<td>AMPHOTERICIN B, LIPOSOME</td>
<td>Ampicillin</td>
<td>$mpterm</td>
<td>c</td>
<td>AMPHOTERICIN B, LIPOSOME</td>
</tr>
<tr>
<td>AMPICILLIN</td>
<td>Ampicillin</td>
<td>$mpterm</td>
<td>c</td>
<td>AMPICILLIN</td>
</tr>
<tr>
<td>K-LYTE</td>
<td>K-Lyte</td>
<td>$mpterm</td>
<td>c</td>
<td>K-LYTE</td>
</tr>
<tr>
<td>KAOPECTATE</td>
<td>Kaopectate</td>
<td>$mpterm</td>
<td>c</td>
<td>KAOPECTATE</td>
</tr>
<tr>
<td>KETAMINE</td>
<td>Ketamine</td>
<td>$mpterm</td>
<td>c</td>
<td>KETAMINE</td>
</tr>
<tr>
<td>KETOROLAC</td>
<td>Ketorolac</td>
<td>$mpterm</td>
<td>c</td>
<td>KETOROLAC</td>
</tr>
<tr>
<td>KETOROLAC TROMETHAMINE</td>
<td>Ketorolac Tromethamine</td>
<td>$mpterm</td>
<td>c</td>
<td>KETOROLAC TROMETHAMINE</td>
</tr>
<tr>
<td>LABETALOL</td>
<td>Labetalol</td>
<td>$mpterm</td>
<td>c</td>
<td>LABETALOL</td>
</tr>
<tr>
<td>LACRI-LUBE</td>
<td>Lacri-lube</td>
<td>$mpterm</td>
<td>c</td>
<td>LACRI-LUBE</td>
</tr>
</tbody>
</table>
APPLYING MIXED CASE FORMAT

Now that we have the input control data set created, we can read this data set into PROC FORMAT using the CNTLIN= option on the PROC FORMAT statement. The code to do this would look like the following:

```sas
PROC FORMAT CNTLIN = here.cmfmts;
RUN;
```

Now you can apply the format $mpterm to format the display of your original uppercase data, the MPTERM variable in the CONMED data set, as mixed case. If needed, see SAS documentation for more details on using formats.

CONCLUSION

Although we have only discussed one very specific case, the techniques that were covered could easily be altered to accommodate any monocase data. Keep in mind that although we covered how to create a format using the mixed case function, if creating a new variable is sufficient, then the code to do so (Section 1 in Appendix A) is short and sweet.

Because the CONMED data used was only a few hundred records, efficiency was not a huge concern. Although it may be more efficient to sort and eliminate duplicate records before any of the data is processed, it is not necessary to do so for the mixed case function. We introduced the PROC SORT step before the input control data set was created in order to avoid errors in the log after submitting the PROC FORMAT step.

The method of creating the input control data set does save the programmer the task of painstakingly typing out each label for the VALUE statement in PROC FORMAT and avoids any human errors such as typos. It is suggested that the reader also look into the CNTLOUT= option, which will create an output control data set of all formats used/created in that PROC FORMAT step. The output control data set will be organized similar to the input control data set that we created above, and therefore, it is ready to be read into another PROC FORMAT step using CNTLIN= option.

REFERENCES

SAS V8 Online Documentation

ACKNOWLEDGMENTS

I would like to thank PPD Development, Inc. for encouraging and supporting me and a special thanks to my mother Kathleen Rosa and Jennifer Bridgers for reviewing and editing this paper.

CONTACT INFORMATION

Lisa Soberano
PPD Development, Inc.
Lisa.Soberano@rtp.ppdi.com

APPENDIX A

```sas
LIBNAME bios " location of conmed data ";
LIBNAME here " location where format data should be stored ";

**********************************************
*  Section 1 - Creating Mixed case Function  *
**********************************************;

DATA cm;
  SET here.conmed;
  LENGTH mpref $200;

  /* put spaces after characters where we would like a word break */
  lowterm = LOWCASE(mpterm);
  term = TRANWRD(lowterm, "/", "/ ");
  term = TRANWRD(term,"/","/ ");
  term = TRANWRD(term,""," ");

  numwrds = LENGTH(COMPBL(term)) - LENGTH(COMPRESS(term,"'")) + 1;
  /* numwrds stores the number of words in the string */
```
DO j = 1 TO numwrds;

    wrdp=SCAN(lowterm,j,' ');

    /* don't capitalize prepositions - add to list if data calls for it*/
    IF wrdp NOT IN ('with' 'for' 'and' 'of') THEN DO;
        mpref = TRIM(mpref)||" "
                ||TRIM(LEFT(UPCASE(SUBSTR(wrdp,1,1))))
                ||TRIM(LEFT(SUBSTR(wrdp,2))));
    END;
    ELSE DO;
        mpref = TRIM(mpref)||" "||LEFT(TRIM(wrdp));
    END;

END;

mpref = TRANWRD(mpref,"/ ","/");
mpref = TRANWRD(mpref,"Viii","VIII");
mpref = TRANWRD(mpref,"K-lyte","K-Lyte");
mpref = TRANWRD(mpref,"Tpn","TPN");

RUN;

**********************************************************************
*   Section 2 - Creating Mixed case Format   *
**********************************************************************;

PROC SORT DATA = cm(KEEP=mpterm mpref) NODUPKEY
    OUT  = cmpref(RENAME=(mpterm=start mpref=label));
BY mpterm mpref;
RUN;

DATA here.cmfmts; /*Creates a data set that stores a character format
named $mpterm to be used with the variable mpterm*/
    SET cmpref;
    RETAIN fmtname '$mpterm' type 'c';
    end=start;
RUN;

PROC FORMAT CNTLIN = here.cmfmts;
RUN;