Paper 82-26

SUMMARIZING DATA WITH AN INCONSISTENT TEXT KEY? HERE'S HOW!

Sylvia Tze, Medtronic Inc., Shoreview, MN

ABSTRACT

In healthcare or insurance database systems, we work with a lot of clients. A typical database design consists of tables of data. One such table would contain a field for the client name, a unique identification number, and other important information. The identification number must be unique to serve its purpose. Problems arise when the names are mistyped or misspelled, or simply entered differently due to abbreviations. As a result, the same client would have different names and would be assigned different identification numbers on different records. This poses a challenge when summarizing the data since there is no unique field to serve as a key for grouping the records. This paper explores a methodology to deal with this problem. The idea is to remove and transform words in the name field to achieve uniformity. A group of macros are introduced that would allow users to group records together in summarizing. By using this procedure, refining it to handle different situations, over 90% of the records can be correctly identified and summarized.

INTRODUCTION

Most people have experienced the frustration of summarizing data with a name field. Other than errors and typos, there is the difference between 'CORP' and 'CORPORATION', 'LTD.' and 'LIMITED', 'CALIFORNIA' and 'CA', etc. If there is a unique identification number, the data can then be summarized with that ID number. Unfortunately, the ID number is not always unique. One possible solution is to have a macro that would remove certain words such as 'THE' and to transform, for example, 'CORP' to 'CORPORATION' to achieve uniformity in the name. Two macros, **%trxform** and **%remove**, are written to perform this task.

THE PROBLEM

Here is an example using a data set MEMDAT which, for demonstration purposes, contains 3 fields : group name GRP_NAM, group number GRP_NUM and member months MEMMON.

GRP_NUM	GRP_NAM	MEMMON
11	MINNESOTA AGRICULTURE	4400
12	MN AGRICULTURE	5400
11	MINNESOTA AGRICULTURE	4577
12	MN AGRICULTURE	5422
13	MINN AGRICULTURE	4500
20	ITT TECHNOLOGY	9800
21	ITT TECHNOLOGIES	3466
30	ACE MANUFACTURING CORP	8700
31	ACE MFG CORPORATION	7400
30	ACE MANUFACTURING CORP	2300
41	CIENA GROUP INC., THE	2700
42	THE CIENA GP INC.	7800
51	INTERNATIONAL MARKETING	3488
52	INTL MKTING	7600

We would use PROC SUMMARY to summarize the member month variable by each client.

```
proc summary data=memgrp nway;
  class grp_num;
  id grp_nam;
  var memmon;
  output out=memout sum=;
run;
```

Here is the output data set:

GRP_NAM	GRP_NUM	_FREQ_	MEMMON
MINNESOTA AGRICULTURE	11	2	8977
MN AGRICULTURE	12	2	10822
MINN AGRICULTURE	13	1	4500
ITT TECHNOLOGY	20	1	9800
ITT TECHNOLOGIES	21	1	3466
ACE MANUFACTURING CORP	30	2	11000
ACE MFG CORPORATION	31	1	7400
CIENA GROUP INC., THE	41	1	2700
THE CIENA GP INC.	42	1	7800
INTERNATIONAL MARKETING	51	1	3488
INTL MKTING	52	1	7600

Due to the inconsistency in data entry, the data was not summarized the way we had intended. To name a few inconsistencies, Minnesota was spelled out in three different ways, "Technology" and "Technologies" can easily be grouped together and "International" also exists in its abbreviated form, "Intl".

THE MACROS

%trxform takes in 2 parameters, **&from** and &to. It simply changes any occurrences of &from in the name variable to &to.

%remove was built on **%trxform** with parameter &to being the null string. It takes in 1 parameter, &arg. Any occurrence of &arg in the name variable will be removed.

The methodology is to dissect the name variable into 2 strings, before and after the occurrence of &from. The variable is then constructed back together with &to being the new string to replace &from or null, as in the case for **%remove**.

Sometimes it is necessary to include blanks before or after &from to ensure we are picking up the individual string we want instead of a substring of a bigger string. An example would be 'SCORPION CORP'. We would then want to pass &from as ' CORP' with a blank in the beginning to avoid picking up the 'CORP' in 'S**CORP**ION'. Since the length function neglects trailing blanks, a few lines of codes are written to return the number of trailing blanks.

Here is the finished macro:

```
%macro trxform(from,to);

  /* This part is to figure out the number of
trailing blanks in the &from parameter. */
  /* The length function ignores trailing
blanks. */

  nblanks=0;
  inchar=' ';
  do while (inchar eq ' ');

    /* read in &from from the back to
determine the number of trailing blanks */

    inchar=
      substr(reverse(&from),nblanks+1,1);
    if inchar=' '
      then nblanks=nblanks+1;
  end;

  /* length of the name converting */
  lenvar=length(newname);

  /* location of the &from in the name */
  loc_from=index(newname,&from);

  /* first part of the string up to &from */
  loc_fstr=loc_from-1;

 /* location of the 2nd part of the string */
  loc_tstr=loc_from+length(&from)+nblanks;

  /* length of the 2nd part of the string */
  lenstr2=lenvar-loc_tstr+1;


 /* if we found &from in the name variable */
  if (loc_from ne 0) then do;

 /* set the first part of our string from the
beginning to where we located &from */

  if (loc_fstr > 0)
    then fstr=substr(newname,1,loc_fstr);
         else fstr='';

 /* set the second part of our string to
where we left off &from to the end */

  if (lenstr2 > 0)
     then
      lstr=substr(newname,loc_tstr,lenstr2);
     else lstr='';

    /* reconstruct our name variable */
    newname=trim(fstr)||&to||lstr ;

  end;

%mend;

%macro remove(arg);
  %trxform(&arg,'');
%mend;

%macro cleanup;
  newname=compress(newname,' ,-/!?');
  drop nblanks inchar loc_from loc_fstr
lenvar loc_tstr lenstr2 fstr lstr;
%mend;
```

EXAMPLE

In order to use the macros, we need to create a new variable (newname) which would be set to the name we want to convert (GRP_NAM in our MEMDAT example).

```
data memdat;
  set memdat;
  newname=grp_nam;
run;
```

We do the transformation of the variable in a data step with the macro %cleanup at the end to compress the name of punctuations and blanks and to drop unnecessary temporary variables.

```
data memdat;
  set memdat;

  %remove('THE');
  %remove('INC.');
  %trxform('GIES ','GY ');
  %trxform('MINNESOTA','MN');
  %trxform('MINN ','MN ');
  %trxform(' MANUFACTURING ',' MFG ');
  %trxform(' CORP ',' CORPORATION');
  %trxform(' GP ',' GROUP ');
  %trxform('INTERNATIONAL','INTL');
  %trxform('MARKETING','MKTING');
  %cleanup;

run;
```

Here is the resulting summarized data set:

GRP_NAM	_FREQ_	MEMMON
THE CIENA GP INC.	2	10500
ACE MFG CORPORATION	3	18400
INTL MKTING	2	11088

```
ITT TECHNOLOGY                  2         13266
MN AGRICULTURE                  5         24299
```

The data is now summarized in the expected way and MEMMON contains the total member months for the different groups.

CONCLUSION

The macros **%trxform** and **%remove** allow us to manipulate a text variable to be used as a key in data summarization. The macros are easy to use and can be readily adapted to other applications.

ACKNOWLEDGEMENT

The author wants to thank Susan Knox for proofreading this paper.

AUTHOR CONTACT

Sylvia Tze, Senior Business Systems Analyst
Medtronic, Inc.
4000 Lexington Avenue North
Mail Stop X230
Shoreview, MN 55126-2983
Phone : (763) 514-9401
E-Mail : sylvia.tze@medtronic.com