Mailing List Management

Imelda C. Go, Richland County School District One, Columbia, SC

ABSTRACT

SAS® functions and procedures can be very helpful in managing mailing lists. For example, duplicate addresses are to be eliminated for a particular mailing. A data set has two addresses: `'501 Pelham Drive'` and `'501 Pelham Dr'`. These are logically equivalent addresses but are not recognized by SAS as the same character values. Use the TRANWRD function to change 'Drive' to 'Dr' or to change substrings within an address. The paper includes other mailing list management-related tasks and uses SAS to automate them.

INTRODUCTION

Input data can come from a variety of sources and it is not always possible to control how they are created and their quality. High quality data naturally require less troubleshooting and programming. Some strategies for preventing data processing problems are:

- Establish data entry standards. For example, only standard Postal Service abbreviations are to be entered as the street type. This would avoid the problem described in the abstract. Another example is to denote the apartment numbers consistently with '#' or 'Apt' after the street name or have a separate field for the apartment number. Consistent data facilitates programming.
- Prevent invalid values from being entered as data. For example, use lookup tables that restrict values of variables to valid values only.
- Validate the data prior to processing. Decide what happens to records with incomplete addresses. Undeliverable mail is waste.
- Plan the data's structure to be compatible with the uses of the data. For data used in applications that rely heavily on street address data (e.g., transportation routing, mapping, geographic information systems), it might be useful to decompose a street address into a number of variables (street number, street name, street type, and apartment number) instead of using one variable to represent it.

There is typically more than one SAS programming solution to a problem. A few general examples are given to illustrate the use of SAS character functions and other features to handle mailing list management issues.

EXAMPLES

Situation 1 How do I change address substrings into other values?

Inconsistent data cause problems. For example, if addresses in the data inconsistently use 'Drive' and 'Dr' to denote the street type, address comparisons become more difficult. Although 'Drive' and 'Dr' are logically equivalent, they are not equivalent based on a character-by-character comparison. Use the TRANWRD function to change substrings within an address.

```
address=tranwrd(address,'Drive','Dr');
```

Making global replacements require caution. For example, if all occurrences of `Court` were to be replaced by `Ct`, `Courtney St` would incorrectly become `Ctney St`. This can be avoided by using the INDEXW function.

```
if indexw(address,'Court')>0 then
   address=tranwrd(address,'Court','Ct');
```

The INDEXW function searches `address`, from left to right, for the first occurrence of `Court` and returns the position where it first occurs in `address`. If there are multiple occurrences, the function returns the position of the first occurrence. The function searches for patterns that are words and not parts of words. Word boundaries for INDEXW are blanks, and the beginning and the ending of `address`. If the pattern does not occur as a word, the function returns a value of 0. Otherwise, it returns a positive integer. (Caution: Punctuation marks are not considered word boundaries.)

`test=indexw(address,'Court')`

value of address	value of test
'Courtney'	0
'River Court'	7
'Court,'	0

Situation 2 There are records in my data set that have zip codes, but have no state. Can SAS can help me find the state values?

The ZIPNAME, ZIPNAMEL, and ZIPSTATE functions require a five-character expression and return a character expression. They provide the state or U.S. Territory that corresponds to the zip code.

sample statement	value of test
test=zipname('29209');	'SOUTH CAROLINA'
test=zipnamel('29209');	'South Carolina'
test=zipstate('29209');	'SC'

ZIPNAME returns the name in uppercase. ZIPNAMEL returns the name in mixed case. ZIPSTATE returns the two-character state postal code (or world-wide GSA geographic U.S. territories code).

Situation 3 How do I validate the zip codes (`zip`) and state (`state`) information?

There are four possibilities in a record:

correct zip, correct state
correct zip, incorrect state
incorrect zip, correct state
incorrect zip, incorrect state

Each state has a range of valid zip code values. The ZIPNAME, ZIPNAMEL, or ZIPSTATE functions can be used to check if the zip code corresponds to the state in the data. Whenever there is a discrepancy, either one or both of the `zip` and `state` values are incorrect. If there are parameters that govern the data, it might be possible to determine which value is invalid. For example, if all addresses in a data set are South Carolina addresses, then the zip codes for those addresses must start with the number 2. Conditional statements, PROC FORMAT, etc. can also be used to validate zip codes and other variables.

Situation 4 A single variable contains the street number and street name. How do I obtain the street name from this variable?

The SCAN function can be used to obtain the value of the street name. Both examples scan for the 2nd word in the given character expression. The word separator specified is a space. `Pelham` would be the desired street name. However, the street number `3 1/2` was not compatible with the intent of the SCAN function in the example below.

```
test=scan(address,2,' ');
```

value of address	value of test
'501 Pelham Drive'	'Pelham'
'3 1/2 Pelham Dr'	'1/2'

To prevent such an error, use the COMPRESS function to remove specified characters from a string. If no characters for removal are specified, the function removes spaces by default. The LEFT function left aligns a character expression.

```
test=left(compress(address,'1234567890/'));
```

value of address	value of test
'501 Pelham Drive'	'Pelham Drive'
'3 1/2 Pelham Dr'	'Pelham Dr'

`test=compress('exam ple')` results in `test='example'`

Situation 5 With one record per person, how do I find out how many people live in the same city?

Comparing strings is case-sensitive. That is, `'COLUMBIA'` is not the same as `'Columbia'`. When data are typed in mixed case, it can be prone to human error. For example, `'Columbia'` may have been typed as `'COLumbia'`. One solution is to use the UPCASE or LOWCASE functions.

sample statement	value of test
test=upcase('Columbia');	'COLUMBIA'
test=lowcase('COLumbia');	'columbia'

Another problem that could occur is `'West Columbia'` might be entered as `'West  Columbia'` where there are two spaces between `West` and `Columbia`. Use the COMPBL function to convert two or more consecutive blanks into one blank.

```
test=compbl(city);
```

value of city	value of test
'West Columbia'	'West Columbia'
'West Columbia'	'West Columbia'

After modifying the data to prevent problems of inconsistent data entries, use PROC FREQ or other SAS procedures to count the number of people who live in the same city.

```
proc freq;
   tables city;
```

Situation 6 Some records have duplicate addresses. How do I create a data set that only has unique addresses?

Perhaps only one piece of mail needs to go to each address that occurs more than once in the data set.

```
proc sort data=withduplicates
   nodupkey out=noduplicates;
   by address;
```

The NODUPKEY option checks for observations with the same BY values. All but one of the records with the same BY values are excluded from the output data set `noduplicates`. If no output data set is specified, the input data set is overwritten by the output data set. (Caution: In most operating systems, only the first of a set of duplicate records based on BY values will be kept.)

Situation 7 The data set has records that are exact copies of each other. How do I eliminate all but one of these duplicate records?

```
proc sort data=withduplicates
   noduprecs out=noduplicates;
   by …;
```

The NODUPRECS (alias NODUP) checks for and eliminates consecutive duplicate observations. (Caution: Because it analyzes consecutive observations, it is important to sort on variables that will cause duplicate records to be adjacent to each other. Otherwise, there is the potential to not eliminate all duplicate records.)

Situation 8 It is assumed that students who have exactly the same address are siblings. How do I separate the students with no siblings from the students with siblings?

Perhaps a separate mailing goes to each group of students. One approach is to use the FIRST. and LAST. variables.

```
proc sort data=students;
   by address;

data onlychild siblings;
   set students;
   by address;
   if first.address and last.address
      then output onlychild;
      else output siblings;
```

CONCLUSION

SAS has features that facilitate the management of address-related data. Whenever possible, create procedures that help produce high quality data instead of relying solely on programming to fix problems with the data.

REFERENCES

SAS Institute Inc., *SAS® Language Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999. 1256 pp.

SAS Institute Inc., *SAS OnLineDoc®*, *Version 8*, Cary, NC: SAS Institute Inc., 1999.

TRADEMARK NOTICE

Imelda C. Go
Office of Research and Evaluation
Richland County School District One
1616 Richland St.
Columbia, SC 29201
Tel.: (803) 733-6079
Fax: (803) 929-3873
icgo@juno.com