

NAME MATCHING USING SASTM LIST INPUT

Ron Vorp, Miami-Dade Community College, Miami, FL

ABSTRACT

Match-merging files based on persons' names can present some unique problems. Names may not be in a common format, names are easily misspelled, and individuals' names may be slightly different on the files. This paper describes coding using SASTM LIST input and other coding techniques to achieve approximately 99% accuracy on a name match between Miami-Dade Community College and Dade County Public School enrollment files.

INTRODUCTION

Miami-Dade Community College (M-DCC), receives files annually from Dade County Public Schools (DCPS) containing names and other information for seniors enrolled during that year. These are matched against M-DCC's enrollment files the following year to determine the percent draw and the draw from the individual schools. The data provide M-DCC personnel with information on the number and percent of recent DCPS graduates enrolling at M-DCC. DCPS student data are also useful in determining characteristics of direct entry M-DCC students.

The actual files used to illustrate the methods described were the DCPS file of 1994-1995 students qualified for graduation and the Summer 1995, Fall 1995, and Winter 1996 M-DCC enrollment files.

METHODS

Original attempts were made at matching by student name. This was the one variable most unique to students on the two files. These initial attempts revealed several problems including:

- separate last name/first name fields vs. one field for the entire name
- differences in input of hyphenated names (e.g. Fils-Aime vs. FilsAime)
- only one file containing titles (e.g. Jr.)
- differences in use of first names (e.g. Jim vs. James)
- two or more students with identical names.

The data were then examined for common variables in addition to the name. In this case, both files contain a numeric high school code and birth date. Including these along with the name as BY variables in the match-merge reduces the chance of false positive matches.

The probability of false positives is also reduced by restricting the size of the M-DCC data set. The target data set in this case is created from M-DCC 1995-1996 enrollment files, and the DCPS finder data set is created from 1994-1995 seniors. The target data set is restricted and still allowed a window large enough to reduce the possibility of false negatives by including only M-DCC students who entered M-DCC later than Fall Term 1993, or show a high school graduation date later than 1993, and have a M-DCC high school code indicating graduation from a Dade County Public School. These statements are not shown in the M-DCC DATA step.

Preliminary examination indicated differences in use of first names (e.g. Jim vs. James, Robert vs. Bob) were too numerous to use as a matching variable. However, they are input, along with the address, to assist in verification. The first name is also used in a subsequent merge to find matching observations missed in the initial merge.

All variables are input from the two files using different variable names and used within the DATA steps to create common variables. All are retained to assist in verification.

The primary difference between the DCPS and M-DCC files is the inputted format of the name. The M-DCC file has one 25 character field for the entire student name and often includes a middle name. The DCPS file has one 14 character field for the last name and a separate 10 digit field for the first name. No field is available for the middle name. The result in the record layout is:

```
MDCC FILE = JONES JAMES
DCPS FILE = JONES JAMES
```

SAS LIST input and FORMAT and TRIM statements are used to compensate for the difference in record layouts¹. The FORMAT statement is used to set all last name variables (LASTDCPS, LASTMDCC, LASTNAME) to the maximum of 14 characters allowed on the DCPS file. The FORMAT statement is necessary because the default maximum length is 8 characters with LIST input. This also results in any DCPS file last names exceeding 14 characters being truncated to 14, as they are on the M-DCC file.

LIST input is invoked by specifying the variable in the INPUT statement with no length parameter. This results in the variable being read starting from the specified starting column until the specified delimiter is encountered. The default delimiter is a blank space, meaning the last name variable from both files will be read from the starting column to the first blank space in the input field. The TRIM function is used to remove trailing blanks resulting from the name being shorter than the formatted value.

All first name variables (FRSTDCPS, FRSTMDCC, FRSTNAME) are set at 5 characters. This length was arbitrarily chosen through trial and error as the most useful. The TRIM function is also used for the FRSTNAME. The starting column for the first name is always column 15 on the DCPS file and the pointer control is used. The first name begins in the first column after the blank delimiter on the M-DCC file. This varies among records and the pointer control cannot be used to specify a starting column. The pointer control is used after the first name to input the high school code. This also bypasses any middle name which is present.

The resulting DATA steps, including the high school code (HSCODE), birth date (BDATE) and address (DCPSADDR and MDCCADDR) are:

¹ see INPUT, LIST command in SAS LANGUAGE: REFERENCE VERSION 6 for alternate methods of accomplishing the same FORMAT and TRIM results.

```

DATA DCPS ;
  INFILE DCPSFILE ;
  FORMAT LASTDCPS $14.
    LASTNAME $14.
    FRSTDCPS $ 5.
    FRSTNAME $ 5. ;
  INPUT @1 LASTDCPS $
    @15 FRSTDCPS $
    @26 DCPSCODE $ 6.
    DCPSDATE 8.
    DCPSADDR $35. ;
  LASTNAME = TRIM(LASTDCPS) ;
  FRSTNAME = TRIM(FRSTDCPS) ;
  BDATE = DCPSDATE ;
  HSCODE = DCPSCODE ;

```

```

DATA MDCC ;
  INFILE MDCCFILE ;
  FORMAT LASTMDCC $14.
    LASTNAME $14.
    FRSTMDCC $ 5.
    FRSTNAME $ 5. ;
  INPUT @1 LASTMDCC $
    FRSTMDCC $
    @26 MDCCCODE $ 6.
    MDCCDATE 8.
    MDCCADDR $35. ;
  LASTNAME = TRIM(LASTMDCC) ;
  FRSTNAME = TRIM(FRSTMDCC) ;
  BDATE = MDCCDATE ;
  HSCODE = MDCCCODE ;

```

The data sets are then sorted and merged. Matching observations are output to the DCPSMDCC data set, non-matches are output to the ONLYMDCC and ONLYDCPS data sets. Using the KEEP option allows the data sets to retain only the originally input variables. This means the common variables must be re-created in subsequent DATA steps. Otherwise, each data set will retain the same value for the common variables and values will be overwritten during subsequent merges.

```

PROC SORT DATA = DCPS ;
  BY HSCODE BDATE LASTNAME ;

```

```

PROC SORT DATA = MDCC ;
  BY HSCODE BDATE LASTNAME ;

```

```

DATA DCPSMDCC
  ONLYMDCC (KEEP = all input MDCC variables)
  ONLYDCPS (KEEP = all input DCPS variables) ;
  MERGE DCPS (IN=D)
    MDCC (IN=M) ;
  BY HSCODE BDATE LASTNAME ;
  IF D AND M THEN OUTPUT DCPSMDCC ;
  ELSE IF D = 1 AND M = 0 THEN OUTPUT ONLYDCPS ;
  ELSE IF D = 0 AND M = 1 THEN OUTPUT ONLYMDCC ;

```

```

PROC DATASETS ;
  DELETE MDCC DCPS ;

```

Additional statements were added to correct for the use of hyphenated names and titles in the two files. The DCPS files contains hyphens within names while the MDCC file omits these. On the other hand, the MDCC files include the addition of titles (e.g. JR., SR, etc.) which are not included on the DCPS files. Examples include:

```

MDCC FILE = FILSAIME JACQUES
DCPS FILE = FILS-AIME JACQUES

```

```

MDCC FILE = JONES JR JAMES
DCPS FILE = JONES JAMES

```

The DELIMITER option was added to both INFILE statements and subroutines added to each DATA step to compensate. The DELIMITER option allows the use of delimiters with LIST input other than the default blank. However, since the delimiters are now specified, the default value of blank is no longer assumed and must be included. A new variable (BACKUP = -1) is also created. BACKUP and the trailing @ are used in the INPUT statement to test the value of the delimiter. The sub-routines are entered based on the value of the delimiter. The INPUT statement is then resumed.

```

DATA DCPS HYPHENS ;
  INFILE DCPSFILE DELIMITER = ',' ;
  FORMAT LASTDCPS $14.
    LASTNAME $14.
    FRSTDCPS $ 5.
    FRSTNAME $ 5. ;
  BACKUP = -1
  INPUT @1 LASTDCPS $
    + BACKUP
    DELIM $ 1.0 @ ;
  IF DELIM = ',' THEN DO ;
    INPUT LASTPART $ @ ;
    HOLENAM = TRIM(LASTDCPS)||LASTPART ;
    LASTDCPS = HOLENAM ;
    OUTPUT HYPHENS ;
  END ;
  INPUT @15 FRSTDCPS $
    @26 DCPSCODE $ 6.
    DCPSDATE 8.
    DCPSADDR $ 35. ;
  LASTNAME = TRIM(LASTDCPS) ;
  FRSTNAME = TRIM(FRSTDCPS) ;
  ADDRESS = DCPSADDR ;
  BDATE = DCPSDATE ;
  HSCODE = DCPSCODE ;

```

```

DATA MDCC COMMAS ;
  INFILE MDCCFILE DELIMITER = ',' ;
  FORMAT LASTMDCC $14.
    LASTNAME $14.
    FRSTMDCC $ 5.
    FRSTNAME $ 5. ;
  BACKUP = -1 ;
  INPUT @1 LASTNAME $
    + BACKUP
    DELIM $ 1.0 @ ;
  IF DELIM = ',' THEN DO ;
    INPUT TITLE $ @ ;
    OUTPUT COMMAS ;
  END ;
  INPUT FRSTMDCC $
    @26 MDCCCODE $ 6.
    MDCCDATE 8.
    MDCCADDR $35. ;
  LASTNAME = TRIM(LASTMDCC) ;
  FRSTNAME = TRIM(FRSTMDCC) ;
  ADDRESS = MDCCADDR ;
  BDATE = MDCCDATE ;
  HSCODE = MDCCCODE ;

```

The match-merge is performed with the same BY HSCODE BDATE LASTNAME statement.

Subsequent match-merges using different criteria are performed with the non-matching data sets. The matching observations are then used to update the DCPSMDCC data set. Each time the non-matching observations are output to different data sets. These are then used for the next match-merge. The following sorts and matchings are used:

1. The common variable formed from the first name variables in each data set are sorted and matched by:

```
BY HSCODE LASTNAME FRSTNAME ;
```

2. A new variable is formed by truncating LASTNAME TO 5 characters in each data set:

```
FORMAT LASTNAM5 $5. ;
```

```
LASTNAM5 = TRIM(LASTNAME) ;
```

and sorted and matched by:

```
BY HSCODE BDATE LASTNAM5 FRSTNAME ;
```

Final examination revealed 26 duplicated observations in the matched data set. These resulted from what were apparently twins on the DCPS file. If only one was in the M-DCC enrollment file, the single observation matched with two observations on the DCPS file. These were eliminated by unduplicating the matched data set with:

```
BY HSCODE BDATE LASTNAME FRSTNAME;
```

RESULTS

For the files used for this description, the 1995 DCPS finder data set consists of 14,833 students. The number of matched M-DCC observations for each BY match-merge is:

```
BY HSCODE BDATE LASTNAME ;          n = 4,855
BY HSCODE LASTNAME FRSTNAME ;       n = 150
BY HSCODE BDATE LASTNAM5 FRSTNAME ; n = 39.
```

M-DCC's total draw within one year from DCPS 1995 seniors is 5,044 (34%).

Examination of a 25% sample of matched observations indicates an approximate 0.5% false positive rate for the entire group.

Examination of a 25% sample of non-matched observations indicates an approximate 0.6% false negative rate for the entire group.

CONCLUSION

Although match-merging by persons' names presents challenges, possibilities are available using SASTM programming to minimize errors and produce useful results. Possibilities for minimizing errors include:

- initially creating data sets as similar as possible,
- using several matching variables,
- using different combinations of matching variables,
- assuring observations are formatted as similar as possible.

BIBLIOGRAPHY

SAS Institute, Inc.(1987). SAS Language Reference, Version 6, First Edition. Cary, NC: Author.

ACKNOWLEDGMENT

I would like to thank Miami-Dade Community College, Institutional Research personnel for support and Hal Corson for assistance in learning SAS programming.

Ron Vorp
Miami-Dade Community College
Institutional Research
300 N.E Second Avenue
Miami, FL 33132-2297

Phone: (305) 237-7481

vorpr@mdcc.edu