Techniques for Merging Datasets Based on Key Text Fields

Steve Fleming, National Center for Educational Achievement, Austin, TX

Abstract

Have you ever had to merge two datasets but the only matching field was a long text field such as a name? Have you found that the name from one of the datasets was not built in quite the same way as the name in the other dataset? If so, this presentation is for you. I will review techniques from simple changes in case to complex applications of the PRX functions. A motivating example concerning matching schools from a national dataset with a state dataset will be presented followed by a description of each technique including SAS code.

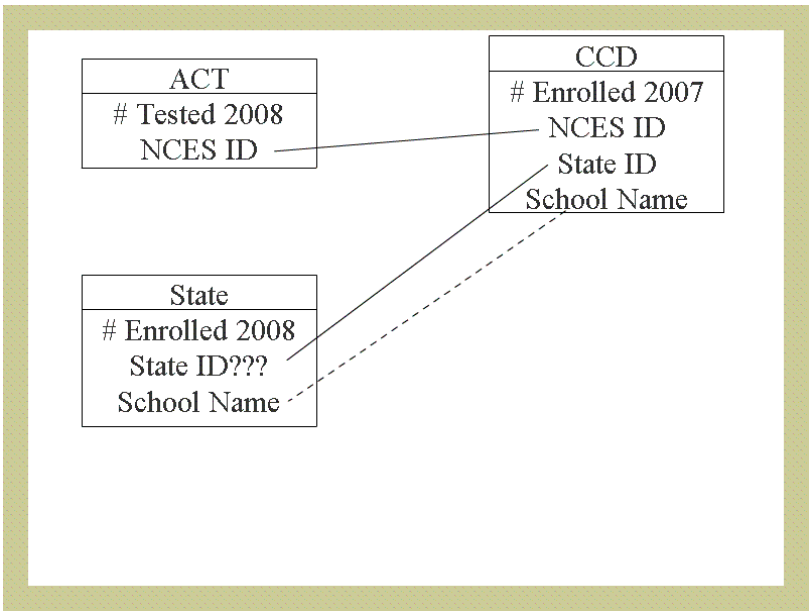
Introduction

One key task undertaken by the National Center for Educational Achievement (NCEA, www.nc4ea.org) is to provide schools with a national comparison to similar or more disadvantaged schools through our CoreWork Diagnostics school improvement tool. This national comparison is on results from ACT's EPAS (Educational Planning and Assessment System) exams. This system consists of the EXPLORE assessment usually taken in grade 8, the PLAN assessment usually taken in grade 10, and the ACT assessment usually taken in either grade 11 or 12.

To insure fair comparisons, an important component is to estimate an approximate percentage of students in the grade that were tested on the particular EPAS test. Then school comparisons are only made to schools that have a high percentage of students taking the assessment. The idea being that schools with lower percentages of tested kids may only have their most advantaged kids taking the exam thus inflating their results over what the school would really achieve if all students took the assessment.

Since ACT does not collect the number of students at the tested grade level in each school, the best national source of such data is the Common Core of Data (CCD) dataset administered by the National Center of Education Statistics (NCES). This dataset provides detailed information about every public school in the United States [Sable and Chen 2009, nces.ed.gov/ccd/pubschuniv.asp]. ACT maintains an ID for each tested school that matches the federal school ID in the CCD data base (Figure 1).

Figure 1: Data Relationships



However, the CCD dataset is at least one and sometimes two years out of date. To get the most accurate value of the number of students enrolled in a grade in each school we download publically available enrollment data from state departments of education. Fortunately, the CCD dataset contains a field that captures each state's ID for the particular school. In theory, this allows us to easily obtain the current year enrollment (Figure 1). In reality, not all states report their school state ID with their publically available data. When this happens, we usually have to match schools by name. This is problematic because the school names reported in the CCD dataset are not always the same when reported by the state. For example, "J. S. Smith High School" in one dataset might be known as "John Stuart Smith HS" in the other.

Data analysts at NCEA developed SAS code to maximize the number of correct school matches between the federal and state datasets when school name was the best common variable to match. This paper explores some of these coding techniques and other considerations that might be useful in text matching. For this application, the cost of a false positive (incorrect match) was considered higher than a false negative (missed correct match). To simplify the presentation, two small datasets will be used to demonstrate the techniques. The State dataset (Table 1) contains two fields, *sname:* the school name and *sdname:* the school district name. The federal dataset contains three fields, *fid:* the federal school ID, *fname:* the school name, and *fdname:* the school district name.

Table 1: State Dataset

sname	*sdname*
Austin Elementary	Central
Brownsville School	Coastal
Corpus Christi High	Coastal
Dallas Middle - North	Blackland
Elm Street Middle	Blackland
Elm Street Middle	Caprock
Fleming El PS 238	Caprock
Garland Elmentary	Prairie
Highland Elementry	Prairie
Johnson High	Central
Johnston High	Central

Tips for Starting

Familiarity with the data is one of the most important aspects of quality data matching. After perusing a sample of each dataset, a helpful next step is to try matching the data directly with no manipulation of the data values. Although this rarely achieves the final goal, it often yields clues that point to fruitful techniques. In this presentation, PROC SQL is used to merge the datasets (Sample 1), but this work could also be done in a DATA step. One advantage of PROC SQL is that merging can be done even when the field names on the two datasets are not the same. A FULL JOIN is used in the merge so that all records from each dataset are kept even if they do not match. This helps identify some of the reasons observations do not match (Output 1).

Table 2: Federal Dataset

fid	*fname*	*fdname*
03	AUSTIN ELEMENTARY	CENTRAL
95	BROWNSVILLE SCH	COASTAL
66	CORPUS CHRISTI HS	COASTAL
85	DALLAS MIDDLE NORTH	BLACKLAND
22	ELM STREET MIDDLE	BLACKLAND
28	ELM STREET MIDDLE	CAPROCK
49	PS 238 FLEMING EL	CAPROCK
79	GARLAND ELEMENTARY	PRAIRIE
35	HIGHLAND ELEMENTARY	PRAIRIE
02	L B JOHNSON HIGH	CENTRAL
50	JOHNSTON HIGH	CENTRAL

Sample 1: Merge Based on Equality Only

```
proc sql;
select f.fid, f.fname, s.sname
      from state as s
            full join
             federal as f
            on s.sname = f.fname ;
quit;
```

Case Matters

It is apparent that some of the difficulty in merging the two datasets is simply due to the federal dataset using all uppercase and the state dataset using mixed case (Output 1). For example, a false negative occurred when "AUSTIN ELEMENTARY" on the federal dataset did not match "Austin Elementary" on the state dataset. Case matters when merging by text fields! A lower case letter is not the same as an upper case letter. A simple modification eliminates the problem through the use of the UPCASE function (Sample 2). The results indicate that our first successful match has occurred for Austin Elementary (Output 2).

Standardization

Some of the remaining problems with merging are due to differing abbreviations and punctuation used in the two datasets (Output 2). For example, the Federal dataset uses "SCH" for "school" and "HS" for High School. To address such issues two name standardization macros were developed. The **fixAbbr** macro accepts two parameters, *oldAbbr* which is the abbreviation we want to change from and *newAbbr* which is the abbreviation we want to change to (Sample 3). The macro uses the INDEXW function to avoid identification of the old abbreviation within words in the string. For example, we would not want to change the "HS" in "TRUTHS" to yield "TRUTHIGH". The **changeName** macro creates a variable called *matchName*. This macro is a shortened version of the one used by NCEA to standardize school names. For this presentation, it first converts the *matchName* to upper case. "SCHOOL" and "SCH" are removed from the name, "HS" is converted to "HIGH", and finally all punctuation and spaces are stripped from the name using the 'p' and 's' modifiers of the COMPRESS function [Murphy 2006]. The modified PROC SQL statement now uses the matchName variable to merge the datasets. Brownsville School, Corpus Christi High, and Dallas Middle North are all successfully matched by this updated code (Output 3).

Output 1: Merge Based on Equality

fid	*Fname*	*sname*
03	AUSTIN ELEMENTARY	
		Austin Elementary
95	BROWNSVILLE SCH	
		Brownsville School
66	CORPUS CHRISTI HS	
		Corpus Christi High
85	DALLAS MIDDLE NORTH	
		Dallas Middle - North
28	ELM STREET MIDDLE	
22	ELM STREET MIDDLE	
		Elm Street Middle
		Elm Street Middle
		Fleming El PS 238
79	GARLAND ELEMENTARY	
		Garland Elmentary
35	HIGHLAND ELEMENTARY	
		Highland Elementry
50	JOHNSTON HIGH	
		Johnson High
		Johnston High
02	L B JOHNSON HIGH	
49	PS 238 FLEMING EL	

Sample 2: Merge Using UPCASE Function

```
proc sql;
select f.fid, f.fname, s.sname
      from state as s
            full join
             federal as f
            on upcase(s.sname) = upcase(f.fname) ;
quit;
```

Output 2: Merge Using UPCASE Function

fid	*fname*	*sname*
03	AUSTIN ELEMENTARY	Austin Elementary
95	BROWNSVILLE SCH	
		Brownsville School
		Corpus Christi High
66	CORPUS CHRISTI HS	
		Dallas Middle - North
85	DALLAS MIDDLE NORTH	
28	ELM STREET MIDDLE	Elm Street Middle
22	ELM STREET MIDDLE	Elm Street Middle
28	ELM STREET MIDDLE	Elm Street Middle
22	ELM STREET MIDDLE	Elm Street Middle
		Fleming El PS 238
79	GARLAND ELEMENTARY	
		Garland Elmentary
35	HIGHLAND ELEMENTARY	
		Highland Elementry
		Johnson High
50	JOHNSTON HIGH	Johnston High
02	L B JOHNSON HIGH	
49	PS 238 FLEMING EL	

Sample 3: Standardization Macros

```
%macro fixAbbr(oldAbbr=, newAbbr=);
      %let oldLen = %length(&oldAbbr.);

      start = indexw(matchName, "&oldAbbr.", " ./-:()");

      do while (start ne 0);
            select (start);
                  when (1)  /* beginning of string */
                        matchName = "&newAbbr." ||
                                    substr(matchName,start+&oldLen.);
                  when (length(matchName) - &oldLen. + 1) /* end of string */
                        matchName = substr(matchName,1,start-1) ||
                                    "&newAbbr." ;
                  otherwise  /* middle of string */
                        matchName = substr(matchName,1,start-1)
                                    || "&newAbbr."
                                    || substr(matchName,start+&oldLen.);
            end;

            start = indexw(matchName, "&oldAbbr.", " ./-:()");
      end;
%mend fixAbbr;
```

Sample 3: Standardization Macros (continued)

```
%macro changeName(varold=);
      matchName = upcase(&varold.);

      %fixAbbr(oldAbbr=SCHOOL, newAbbr=);
      %fixAbbr(oldAbbr=SCH, newAbbr=);
      %fixAbbr(oldAbbr=HS, newAbbr=HIGH);

      /* remove space and punctuation characters */
      matchName = compress(matchName,,'ops');

      drop start ;
%mend changeName;

data state;
      set state;
      length matchName $30;
      %changeName(varold=sname);
run;

data federal;
      set federal;
      length matchName $30;
      %changeName(varold=fname);
run;

proc sql;
select f.fid, f.fname, s.sname
      from state as s
            full join
             federal as f
            on s.matchName = f.matchName ;
quit;
```

Duplicates

As more observations are merged, we may end up making more matches than are justified. Such matches can be termed as false positives. In an anticipated one-to-one merge, when any observations from one dataset match more than one observation from the other dataset a false positive has almost certainly occurred. A check for duplicate matches is always a good idea (although false positives may occur without duplicates). PROC SQL can be used (Sample 4) by finding the number of records (nRecs) that have the same Federal ID (fid). Notice the subquery in parentheses which essentially does the same merge found in Sample 3. A similar check for duplicates from the State dataset would also be advised.

The schools with fid 22 and 28 were each matched to two observations from the state dataset (Output 4) because there are two separate schools named “Elm Street Middle” in our data. To untangle duplicates such as these, additional information is usually required. In this instance we can resolve the issue by using the school district as an additional key in the merge (Sample 5). Other possibilities might be address or phone number. In this case using district name resolves the duplicates.

Output 3: Standardization Macros

fid	*fname*	*sname*
03	AUSTIN ELEMENTARY	Austin Elementary
95	BROWNSVILLE SCH	Brownsville School
66	CORPUS CHRISTI HS	Corpus Christi High
85	DALLAS MIDDLE NORTH	Dallas Middle - North
28	ELM STREET MIDDLE	Elm Street Middle
22	ELM STREET MIDDLE	Elm Street Middle
28	ELM STREET MIDDLE	Elm Street Middle
22	ELM STREET MIDDLE	Elm Street Middle
		Fleming El PS 238
79	GARLAND ELEMENTARY	
		Garland Elmentary
35	HIGHLAND ELEMENTARY	
		Highland Elementry
		Johnson High
50	JOHNSTON HIGH	Johnston High
02	L B JOHNSON HIGH	
49	PS 238 FLEMING EL	

Sample 4: Search for Duplicates

```
proc sql;
select fid, count(*) as nRecs
      from (select f.fid
                  from state as s
                        full join
                         federal as f
                        on s.matchName = f.matchName )
      where not missing(fid)
      group by fid
      having nRecs > 1;
quit;
```

Output 4: Search for Duplicates

fid	*nRecs*
22	2
28	2

Sample 5: Resolving Duplicates

```
proc sql;
select f.fid, f.fname, s.sname, f.fdname, s.sdname
      from state as s
            full join
             federal as f
            on s.matchName = f.matchName
           and upcase(s.sdname) = upcase(f.fdname);
quit;
```

Output 5: Resolving Duplicates

fid	*fname*	*sname*	*fdname*	*sdname*
03	AUSTIN ELEMENTARY	Austin Elementary	CENTRAL	Central
95	BROWNSVILLE SCH	Brownsville School	COASTAL	Coastal
66	CORPUS CHRISTI HS	Corpus Christi High	COASTAL	Coastal
85	DALLAS MIDDLE NORTH	Dallas Middle - North	BLACKLAND	Blackland
22	ELM STREET MIDDLE	Elm Street Middle	BLACKLAND	Blackland
28	ELM STREET MIDDLE	Elm Street Middle	CAPROCK	Caprock
		Fleming El PS 238		Caprock
79	GARLAND ELEMENTARY		PRAIRIE	
		Garland Elmentary		Prairie
35	HIGHLAND ELEMENTARY		PRAIRIE	
		Highland Elementry		Prairie
		Johnson High		Central
50	JOHNSTON HIGH	Johnston High	CENTRAL	Central
02	L B JOHNSON HIGH		CENTRAL	
49	PS 238 FLEMING EL		CAPROCK	

Regular Expressions

Thus far we have covered some of the basic tools for text matching. Now we will begin exploring more advanced features. Regular Expressions are a powerful set of tools that can seek out and manipulate very complex patterns. For example, in the federal dataset, school 49 is called “PS 238 Fleming El” while in the state dataset it is known as “Fleming El PS 238”. What if we could recognize when the pattern PS followed by some numbers is present? Regular expressions give us ways to do this. In SAS, Perl regular expressions are available through the PRX function starting in version 9.

A DATA step to modify the state dataset was constructed (Sample 6). Upon reading the first record in the dataset, the PRXPARSE function compiles the pattern of the letters “PS” followed by one to three digits “\d{1,3}” which is retained for subsequent observations in the variable *psDigits*. The CALL PRXSUBSTR routine checks for a match to the expression within the matchName string. If one is found the values of start and len are filled in the starting position and the length of the pattern respectively. If this occurs at the beginning of the string we leave it alone, otherwise we rearrange such that the pattern comes at the beginning of the string. Once this change has been made we are able to match Fleming Elementary (Output 6). There are many other useful elements besides the ones mentioned here that can be included regular expressions [Cody 2005].

Sample 6: PRX Functions

```
data state(drop=psDigits start len);
      set state;
      retain psDigits;

      if _n_ = 1 then psDigits = prxparse("/PS\d{1,3}/");
            /* look for a pattern the characters PS followed by 1 to 3 digits */
      /* Determine the position and length of match */
      call prxsubstr(psDigits, matchName, start, len);
      /* If found but not at beginning rearrange to standard form */
      if start > 1 then do;
            if start + len - 1 = length(matchName)
                  then matchName = cats(substr(matchName,start,len),
                                        substr(matchName,1,start-1) );
            else matchName = cats(substr(matchName,start,len),
                                  substr(matchName,1,start-1),
                                  substr(matchName,start+len,length(matchName)));
      end;
run;
```

Output 6: PRX Functions

fid	*Fname*	*sname*	*fdname*	*sdname*
03	AUSTIN ELEMENTARY	Austin Elementary	CENTRAL	Central
95	BROWNSVILLE SCH	Brownsville School	COASTAL	Coastal
66	CORPUS CHRISTI HS	Corpus Christi High	COASTAL	Coastal
85	DALLAS MIDDLE NORTH	Dallas Middle - North	BLACKLAND	Blackland
22	ELM STREET MIDDLE	Elm Street Middle	BLACKLAND	Blackland
28	ELM STREET MIDDLE	Elm Street Middle	CAPROCK	Caprock
79	GARLAND ELEMENTARY		PRAIRIE	
		Garland Elmentary		Prairie
35	HIGHLAND ELEMENTARY		PRAIRIE	
		Highland Elementry		Prairie
		Johnson High		Central
50	JOHNSTON HIGH	Johnston High	CENTRAL	Central
02	L B JOHNSON HIGH		CENTRAL	
49	PS 238 FLEMING EL	Fleming El PS 238	CAPROCK	Caprock

Similarity Functions

SAS provides a many other functions that check for the similarity between text strings. We will explore the SOUNDEX and COMPGED functions.

SOUNDEX

The SOUNDEX function implements an algorithm patented in 1922 that attempts to indentify text strings that "sound" alike. Often this can catch minor misspellings especially those involving vowels. In the state data someone had trouble typing "ELEMENTARY" correctly. Matching using the SOUNDEX function (Sample 7) allows "ELMENTARY" and "ELEMENTRY" to be matched to "ELEMENTARY" (Output 7).

Sample7: SOUNDEX

```
proc sql;
select f.fid, f.fname, s.sname, f.fdname, s.sdname
      from state as s
            full join
             federal as f
            on soundex(s.matchName) = soundex(f.matchName)
         and upcase(s.sdname) = upcase(f.fdname);
quit;
```

Output 7: Soundex

fid	*Fname*	*sname*	*fdname*	*sdname*
03	AUSTIN ELEMENTARY	Austin Elementary	CENTRAL	Central
95	BROWNSVILLE SCH	Brownsville School	COASTAL	Coastal
66	CORPUS CHRISTI HS	Corpus Christi High	COASTAL	Coastal
85	DALLAS MIDDLE NORTH	Dallas Middle - North	BLACKLAND	Blackland
22	ELM STREET MIDDLE	Elm Street Middle	BLACKLAND	Blackland
28	ELM STREET MIDDLE	Elm Street Middle	CAPROCK	Caprock
79	GARLAND ELEMENTARY	Garland Elmentary	PRAIRIE	Prairie
35	HIGHLAND ELEMENTARY	Highland Elementry	PRAIRIE	Prairie
50	JOHNSTON HIGH	Johnston High	CENTRAL	Central
		Johnson High		Central
02	L B JOHNSON HIGH		CENTRAL	
49	PS 238 FLEMING EL	Fleming El PS 238	CAPROCK	Caprock

COMPGED

COMPGED and the similar functions COMPLEV and SPEDIS calculate an edit or spelling distance between two strings [Code 2005]. The one remaining unmatched school is due to the presence of the initials “L B” before the name of the school. Since you can choose the threshold at which to declare two strings a match, COMPGED can be fine-tuned to maximize the proportion of correct matches.

First we try an edit distance of 10 as the cutoff (Sample 8). This fails to match the schools that the SOUNDEX function caught (Output 8a). However, increasing the cutoff to 100 causes a false positive match between JOHNSTON and JOHNSON HIGH (Output 8b) because deleting a letter in the middle of a string adds 100 points to the edit distance. By contrast, it costs 400 points of edit distance to turn “LBJOHNSON” into “JOHNSON” because deleting the first letter in a string costs 200 points in edit distance. In some situations it would be worth exploring the CALL COMPCOST routine which allows modification of the default edit costs used by COMPGED.

Manual Matches

As increasingly complex SAS functions take care of the low hanging fruit of potential matches, there usually comes a point where the coding effort becomes greater than the cost of programming time particularly when deadlines are tight. We have found that the last few matches usually need to be made manually because the code required to make the match would be too specific to a situation and difficult to maintain. In these cases it makes sense just to change a text string manually (Sample 9).With this change, all of the schools in this example have now been matched (Output 9).

Sample 8: COMPGED Function

```
proc sql;
select f.fid, f.fname, s.sname, f.fdname, s.sdname
      from state as s
            full join
             federal as f
            on compged(s.matchName,f.matchName) <= 10
         and upcase(s.sdname) = upcase(f.fdname)
      order by f.fid;
quit;
```

Output 8a: COMPGED Function (Threshold = 10)

fid	*Fname*	*sname*	*fdname*	*sdname*
		Highland Elementry		Prairie
		Garland Elmentary		Prairie
		Johnson High		Central
02	L B JOHNSON HIGH		CENTRAL	
03	AUSTIN ELEMENTARY	Austin Elementary	CENTRAL	Central
22	ELM STREET MIDDLE	Elm Street Middle	BLACKLAND	Blackland
28	ELM STREET MIDDLE	Elm Street Middle	CAPROCK	Caprock
35	HIGHLAND ELEMENTARY		PRAIRIE	
49	PS 238 FLEMING EL	Fleming El PS 238	CAPROCK	Caprock
50	JOHNSTON HIGH	Johnston High	CENTRAL	Central
66	CORPUS CHRISTI HS	Corpus Christi High	COASTAL	Coastal
79	GARLAND ELEMENTARY		PRAIRIE	
85	DALLAS MIDDLE NORTH	Dallas Middle - North	BLACKLAND	Blackland
95	BROWNSVILLE SCH	Brownsville School	COASTAL	Coastal

Output 8b: COMPGED Function (Threshold = 100)

fid	*Fname*	*sname*	*fdname*	*sdname*
02	L B JOHNSON HIGH		CENTRAL	
03	AUSTIN ELEMENTARY	Austin Elementary	CENTRAL	Central
22	ELM STREET MIDDLE	Elm Street Middle	BLACKLAND	Blackland
28	ELM STREET MIDDLE	Elm Street Middle	CAPROCK	Caprock
35	HIGHLAND ELEMENTARY	Highland Elementry	PRAIRIE	Prairie
49	PS 238 FLEMING EL	Fleming El PS 238	CAPROCK	Caprock
50	JOHNSTON HIGH	Johnston High	CENTRAL	Central
50	JOHNSTON HIGH	Johnson High	CENTRAL	Central
66	CORPUS CHRISTI HS	Corpus Christi High	COASTAL	Coastal
79	GARLAND ELEMENTARY	Garland Elmentary	PRAIRIE	Prairie
85	DALLAS MIDDLE NORTH	Dallas Middle - North	BLACKLAND	Blackland
95	BROWNSVILLE SCH	Brownsville School	COASTAL	Coastal

Sample 9: Manual Changes

```
data state;
      set state;

      if upcase(sname) = "JOHNSON HIGH" then matchName = "LBJOHNSONHIGH";
run;
```

Output 9: Manual Changes

fid	*Fname*	*sname*	*fdname*	*sdname*
02	L B JOHNSON HIGH	Johnson High	CENTRAL	Central
03	AUSTIN ELEMENTARY	Austin Elementary	CENTRAL	Central
22	ELM STREET MIDDLE	Elm Street Middle	BLACKLAND	Blackland
28	ELM STREET MIDDLE	Elm Street Middle	CAPROCK	Caprock
35	HIGHLAND ELEMENTARY	Highland Elementry	PRAIRIE	Prairie
49	PS 238 FLEMING EL	Fleming El PS 238	CAPROCK	Caprock
50	JOHNSTON HIGH	Johnston High	CENTRAL	Central
66	CORPUS CHRISTI HS	Corpus Christi High	COASTAL	Coastal
79	GARLAND ELEMENTARY	Garland Elmentary	PRAIRIE	Prairie
85	DALLAS MIDDLE NORTH	Dallas Middle - North	BLACKLAND	Blackland
95	BROWNSVILLE SCH	Brownsville School	COASTAL	Coastal

Conclusion

From simple to complex, techniques for matching text strings from two different sources have been demonstrated. May you find something useful to apply to your work.

References

Cody, Ron (2005), SAS Functions by Example, SAS Publishing, July 2005.

Murphy, William C (2006), Squeezing Information out of Data, Paper 028-31, SUGI 31.

Sable, Jennifer and Lee Hoffman (2009), Documentation to the NCES Common Core of Data Public Elementary/Secondary School Universe Survey: School Year 2006-07 Revised Version 1c May 2009, retrieved from http://nces.ed.gov/ccd/pdf/psu061cgen.pdf, September 15, 2009.

SAS Institute (2007), Online Documentation for Version 9.1.3, available at http://support.sas.com/onlinedoc/913/docMainpage.jsp.

Author Contact Info

Steve Fleming
National Center for Educational Achievement
4030-2 West Braker Ln
Austin, TX 78759
sfleming@nc4ea.org