PROC SQL

How can I build sample datasets to perform fuzzy matches?

Sometimes data coming from different sources can be subtly different, resulting in a mis-match when performing a straight character comparison, but a visual comparison would suggest that the data should in fact be matched. The sort of issues which can arise are non-alphabetic characters, or issues regarding case-sensitivity.

To demonstrate how to perform the 'fuzzy' matches, we must first generate some 'dirty' data. For this exercise I have used open-source data of companies listed on the NY Stock Exchange. Reading in this data, and generating a dummy stock holding figure:

```
*** Use open-source data with details of companies listed on the NY St
ock Exchange *** ;

filename usco url 'https://raw.githubusercontent.com/datasets/nyse-
listings/master/data/nyse-listed.csv' ;

data nyse (drop = d1) ;
  infile usco dsd firstobs = 2 missover ;
  input d1    : $1.
        stock : $100.
 ;
  *** Remove some of the additional characters which can cause issues
identifying word boundaries *** ;
  stock   = compress(stock,"',)(.-/")   ;
  *** Generate a stock holding *** ;
  holding = round(ranuni(0)*10000,0.01) ;
run ;
```

The stock name has also had a number of non-alphabetic characters removed to assist in clarifying word-boundaries.

Using this master list, the text can be manipulated using PERL Regular Expressions to extract a random number of 'words' from the stock name. The stock holding is also randomly adjusted for a proportion of the records.

```
data list1 (keep = stock stock1 hold1)  ;
  set nyse ;
    *** Count the number of 'words' in the stock name *** ;
 wc = countw(stock) ;
 *** Build a PERL Regular Expression in the form /(\w+ ){04}/ i.e. fou
r words separated by spaces *** ;
 *** Select a randon number of words between 3 and all available words
 *** ;
 regexp = '/(\w+ ){'!!put(max(3,floor(ranuni(0) * wc)),z2.)!!'}/' ;
 /***
      /             Start the PERL RegExp
      (             Open Content Buffer 1
             \w            Match a 'word' character
             +             ...one or more times
                           ...followed by a space
             )             Close Content Buffer 1
             {             Quantifier Multiple of the previous 'thing'
 (in this case 'a word followed by a space')
      !!            Concatenate
             put(max(3,floor(ranuni(0) * wc)),z2.)
                    Generate a random number between 0-1 and multiply
by the number of words             : RANUNI
                    Take the integer less than or equal to the result
                                 : FLOOR
                    Return the greater of 3 or this integer (amend the
 3 to return more / fewer words) : MAX
                    As a zero-filled two-
digit text string e.g. 04                                     : PUT
      !!            Concatenate
      }             Finish Quantifier Multiple
      /             End the PERL RegExp
 ***/

 *** Parse the PERL RegExp and assign it a unique numeric identifier *
** ;
        words = prxparse(regexp) ;
 *** Endeavour to match the desired RegExp and return the Starting poi
nt (s) and Length (e) *** ;
        call prxsubstr(words,stock,s,e) ;
 *** If the RegExp was found within the target string, extract it and
convert to upper case *** ;
 if s and e then stock1 = upcase(substrn(stock,s,e)) ;
 else stock1 = stock ;
 *** Generate a random number, add 1 and round to 2 decimal places ***
 ;
 rn = 1 + round(ranuni(0),.01) ;
 *** Adjust the holding for a proportion of the records *** ;
 if rn < 1.3 then hold1 = round(holding * rn, .01) ;
 else hold1 = holding ;
run ;
```

A second list is generated which results in subtly different stock names and holdings.

```
*** Repeat the process to generate a second list *** ;
```

```
data list2 (keep = stock2 hold2)  ;
  set nyse ;
 wc = countw(stock) ;
 regexp = '/(\w+ ){'!!put(max(3,floor(ranuni(0) * wc)),z2.)!!'}/' ;
        words = prxparse(regexp) ;
        call prxsubstr(words,stock,s,e) ;
 if s and e then stock2 = upcase(substrn(stock,s,e)) ;
 else stock2 = stock ;
 rn = 1 + round(ranuni(0),.01) ;
 if rn < 1.3 then hold2 = round(holding * rn, .01) ;
 else hold2 = holding ;
run ;
```

The two lists can then be aligned to show the original stock name and the two sets of variants.

```
*** Align the two lists to show how the values have been altered *** ;
data paired ;
  set list1 ;
  set list2 ;
run ;
```

Partial output:

stock	stock1	hold1
Agilent Technologies Inc Common Stock	AGILENT TECHNOLOGIES INC	5671
Alcoa Inc Common Stock	ALCOA INC COMMON	3770
Alcoa Inc Depository Shares Representing 110th Preferred Convertilble Class B Series 1	ALCOA INC DEPOSITORY	9229
AAC Holdings Inc Common Stock	AAC HOLDINGS INC	2324
Aarons Inc Common Stock	AARONS INC COMMON	2140
Advance Auto Parts Inc Advance Auto Parts Inc WI	ADVANCE AUTO PARTS INC ADVANCE AUTO	5653
American Assets Trust Inc Common Stock	AMERICAN ASSETS TRUST	753
Advantage Oil & Gas Ltd Ordinary Shares	Advantage Oil & Gas Ltd Ordinary Shares	544

Allianceberstein Holding LP Units	ALLIANCEBERSTEIN HOLDING LP	278
ABB Ltd Common Stock	ABB LTD COMMON	6559
AbbVie Inc Common Stock	ABBVIE INC COMMON	3115
AmerisourceBergen Corporation Holding Co Common Stock	AMERISOURCEBERGEN CORPORATION HOLDING	4909
Ambev SA American Depositary Shares Each representing 1 Common Share	AMBEV SA AMERICAN DEPOSITARY SHARES EACH	4082
Asbury Automotive Group Inc Common Stock	ASBURY AUTOMOTIVE GROUP INC	453

Fuzzy matches can then be performed against the two lists - matching the character variable left -> right and stopping when the end of the shorter string is reached. The holdings can be rounded to a desired level of accuracy before comparing.

```
*** Perform a 'fuzzy' match on the two lists *** ;
*** Read the text strings from left to right and stop when the end of
the shorter one is reached *** ;
*** Compare holdings rounded to the nearest integer-
multiple of 500 (amend this for different margin) *** ;
proc sql noprint ;
  create table matched (drop = len) as
  select  stock1
         ,hold1
         ,stock2
  ,hold2
  ,min(length(stock1),length(stock2)) as len
  from  list1
       ,list2
  where upcase(substr(stock1,1,calculated len)) = upcase(substr(stock2
,1,calculated len))
  and   round(hold1,500) = round(hold2,500)
  ;
quit ;
```

SPEDIS Function

Another useful function for comparing character strings is the SPEDIS function which generates an integer value representing the SPElling DIStance between two strings. The SPEDIS function is case-sensitive - it may be necessary to transform the data prior to comparison.

The function reads the two character variables from left to right and assigns a score for each character difference. Differences at the beginning of the string count more highly than differences at the end of the string. The length of the string also impacts on the score, as a one letter spelling difference is more significant in a short string e.g.

```
data _null_ ;
length w1 w2 $ 200 ;

w1 = 'cat' ;
w2 = 'vat' ;
sped = spedis(w1,w2) ;
put _all_ ;

w1 = 'car' ;
w2 = 'cat' ;
sped = spedis(w1,w2) ;
put _all_ ;

w1 = 'catastrophic' ;
w2 = 'vatastrophic' ;
sped = spedis(w1,w2) ;
put _all_ ;

w1 = 'catastrophic' ;
w2 = 'catastrophiv' ;
sped = spedis(w1,w2) ;
put _all_ ;

run ;


w1=cat w2=vat sped=66 _ERROR_=0 _N_=1
w1=car w2=cat sped=33 _ERROR_=0 _N_=1
w1=catastrophic w2=vatastrophic sped=16 _ERROR_=0 _N_=1
w1=catastrophic w2=catastrophiv sped=8 _ERROR_=0 _N_=1
```

Using the SPEDIS function on the paired dataset created earlier to generate a measure of the distance between the two stock name variants, would allow a filter to be applied to select records which are deemed to be sufficiently similar.

```
*** For variables which have been joined by some other means a 'reason
ability' check can be performed *** ;
*** Using the PAIRED dataset the SPEDIS function assigns numeric value
 to the Spelling Distance       *** ;

data paired ;
  set paired ;
    sped = spedis(stock1,stock2) ;
```

```
run ;
```

Partial Output:

stock	stock1	hold1
Agilent Technologies Inc Common Stock	AGILENT TECHNOLOGIES INC	567
Alcoa Inc Common Stock	ALCOA INC COMMON	377
Alcoa Inc Depository Shares Representing 110th Preferred Convertilble Class B Series 1	ALCOA INC DEPOSITORY	922
AAC Holdings Inc Common Stock	AAC HOLDINGS INC	232
Aarons Inc Common Stock	AARONS INC COMMON	214
Advance Auto Parts Inc Advance Auto Parts Inc WI	ADVANCE AUTO PARTS INC ADVANCE AUTO	565
American Assets Trust Inc Common Stock	AMERICAN ASSETS TRUST	75

Unique solution ID: #1033
Author: Alan D Rudland
Last update: 2017-09-01 12:48