Observations®
THE TECHNICAL JOURNAL
for SAS Software Users
www.sas.com/obs

The Fuzzy Feeling SAS® Provides: Electronic Matching of Records without Common Keys

Charles Patridge

Charles has been using SAS software since 1979 in various positions, companies and consulting assignments. He has worked for Hartford Steam Boiler Inspection and Insurance Co., LIMRA, CIGNA, Aetna, Coldwell Banker Relocation, Bayer Inc., SNETCO, State of Connecticut, Automobile Association of British Columbia, Applied Psychological Techniques, Arthur Andersen Consulting and currently, for The Hartford (formerly ITT Hartford). In addition, he operates his own company (PDPC, Ltd.) specializing in SAS and "fuzzy match" applications as well as statistical analysis applications in a variety of industries. His area of SAS expertise lies in BASE, STAT, AF, FSP, Macros, and ETS.

In 1983 Charles founded HASUG (Hartford, CT Area SAS User Group) and is currently a member of its steering committee. He is founder and director of SAS CONSIG (Consultant Special Interest Group), and creator and Webmaster for the SAS CONSIG Web site. He has presented numerous papers and accepted speaking engagements to HASUG, BASUG, NYSUG, NESUG and SUGI as well as given presentations at the Institute for Graphic Communications.

Since March 1994, Charles has been informally operating a network (now on the Internet) to link companies, agencies, consultants and SAS professionals with employment opportunities, both contract and full-time positions. To date, his efforts have directly and indirectly placed over 70 individuals with assignments throughout the United States and overseas. The Internet address that makes his effort successful is **http://pages.prodigy.com/SASCONSIG.**

Charles holds a B.S. degree in mathematics from Central Connecticut State University, has accumulated 45 hours of graduate studies in statistics, and has completed parts I & II of the Society of Actuaries, HIAA and LOMA exams.

Abstract

This article briefly describes a sample application where the normal SAS merge or SQL join would fail to provide the kind of results needed or hoped for. By using a set of SAS functions, custom programs and macros, and simple scoring techniques and common sense, the author developed a working model that matches two unlinked files that contain no common keys. The fuzzy matching process developed entailed over 1500 hours of development, testing, and revisions after being applied to several different projects and assignments.

Contents

Introduction

Why would anyone want to use this Fuzzy Match/Merge routine?

1. Your marketing department has purchased an outside vendor's file of possible customers and you need to determine which records you already have in your own client database.
2. You would like to delete duplicate records from your client database.
3. You need to determine which hospital patient has used internal services and outside vendor services of an outside vendor for special treatments, in order to satisfy a Federal Request for such information.
4. As an effort to detect insurance fraud, several insurance companies are sharing their claim information database to pinpoint potential fraudulent claims.
5. As a Property & Casualty Insurance Carrier, you would like to know which and how many of your insured locations are in the Federal Super Fund CleanUp database to ensure claim reserves are adequate.
6. Your company has just merged with another company and you would like to combine both client databases to market products to clients common to both companies.

For any of the above reasons, most files containing client information will most likely be stored differently but with common fields such as Name, Address, City, State as well as Zip Code.

Example

Company A's Client File	Company B's Client File
Charles S. Patridge PDPC, Ltd. 172 Monce Road Burlington, CT 06012 860-673-9278	Patridge, Chuck Lot #4, Monce Rd. Unionville, CT 06085 203-673-9278

You can easily see the information in the above example is quite similar between files but it is not exactly the same and would prevent most merge/join routines from matching the two records.

In addition, you determine that switching the names around in one of the files still will not ensure an exact match due to the way the names and addresses are stored. As you can see, the two example records are indeed the same person/entity. However, neither SAS MERGE nor PROC SQL will match/join these two records.

What do you do? It would be like finding "**a needle in a haystack**" given the size of common client databases contain more than 100,000 records.

This is where the fuzzy match routine will help. This set of routines is by no means an exact science nor without its troubles of mismatching records (false hits). However, it will reduce the size of the "haystack" and minimize the amount of labor needed to determine which client is located in both files or is possibly duplicated in the same file.

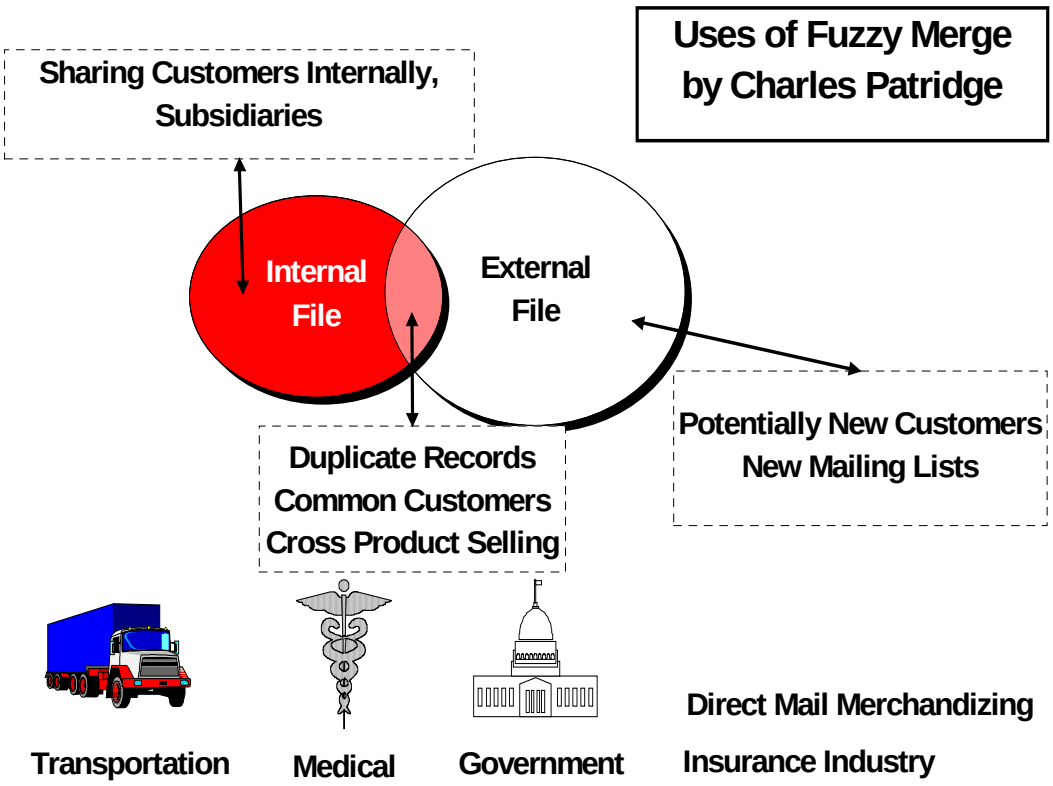
The Fuzzy match/merge routine has been utilized in past projects such as:

a. Match Internal Company Client base with a Commercial Mail Marketing Database:
   - which clients are in both files
   - which clients are the commercial mail file but not an internal client
b. Match a file against itself:
   - reduce mass mailing costs by finding duplicate records for same entity/person/household
c. Company acquisition of another company:
   - products from one company are an add-on to the products of second company. Client base is market for both companies. Find/define the three distinct sets of clients.
d. Insurance company needed to determine which of its insurance agents sold their products directly or sold their products through another insurance company (re-insurance).
e. Insurance company had a list of its insured locations and needed to match it against the Super Fund locations maintained by the Federal/State Governments.
f. State agency needed to match one of its files to another State agency's file.

See Figure 1 for a visual concept of how and where the Fuzzy Match/Merge Routine could be used within your organization. On average it takes approximately 30 hours to tailor the Fuzzy Merge/Match application to a specific application. The amount of time depends on the number available fields used in determining whether a record is a match or a duplicate, and on the building of standardized files.

The more criteria (fields) the application requires, the more accurate and less labor intensive it is in determining if there really is a match on another record. If fewer matching fields are available, the more creative the rules of matching must be to reduce the manual effort and increase the speed of the matching process.

Figure 1: Applications and Industries That Use the Fuzzy Match Process



The next section briefly describes two actual situations where this application has been implemented. I will provide a better understanding of when and where Electronic Fuzzy Matching could be effectively used. The remainder of the paper discusses the techniques and gives an example of the data being applied to the Fuzzy Match/Merge process. Caution should be used in thinking these routines are 100% accurate and fool proof and will produce desired results. As stated in the title, the logic is fuzzy and was not intended to be a 100% matching solution for every possible situation.

Application A: The Mailing Lists

The Vice President of Marketing (VP) creates a list of clients from Dunn & Bradstreet (D&B) that are potentially ideal candidates for a special marketing product campaign. The VP asks you which ones on the list are and are not current clients. You ask if this list of clients exists on a floppy diskette, as you can see this list encompasses several hundred entries. The VP says "Sure, here is the diskette and this piece of paper describes what, where and how the fields are contained on the diskette".

You are relieved you do not have to key in these several hundred entries. You put the diskette into your "hot little" PC and start to peruse the file. You notice several things about the way the data is stored on this PC file:

1. There is no common key (such as social security number) in this PC file that you can use to directly link these entries with your company's client file.

2. This file has name, address, city, state and zip code for each entry.

You conclude that a SAS program could read your client base file and the D&B file, sort the name of client by zip code and use a MERGE statement matching both files by zip code and name of client.

After developing the initial SAS program and running your merge statement, you discover you have SUCCESSFULLY matched 21% of the D&B file with the company client file. You deliver your report to the VP and state you have performed magic with the D&B file by matching 21% of these records. The VP states he appreciates your efforts but believes there should be more than 50% of D&B entries in the current company client file. You ask "How, can that be? I matched those records with a SAS program based on name of client with in zip code!". The VP says "I had my secretary take the first 100 entries and print them out, and she cross referenced them manually with our client list." You then ask to see her list and those she matched. You review the secretary's list quickly and discovered the name of clients on the D&B list are spelled or positioned differently than on the company's client list.

Example

D&B list	**Company Client List**
Chuck Patridge P D P C, Ltd. 172 Monce Road Burlington, Ct. 06013	Patridge, Charles PDPC, Ltd. Monce Rd. Burlington, Ct. 06013-2545

You realize by looking at the two similar entries, that they are in fact the same. Yet, your SAS program with the MERGE statement has failed to match records, because the name of the client is stored or spelled differently from your internal company client file versus the D&B external file. With a puzzled looked, you reply to the VP "I can't match these two files in this way because D&B does not store their clients' names the same way our company does! We would need to purchase a specialized matching software package to perform your request". The VP says "How much would that cost?". You reply "About $60,000?". VP asks "Can't we develop our own matching program?" You reply "Sure, but I would need 3 or 4 months to develop such a program. But I will not be able to get to it for another 3 months unless you say it is OK to put the current projects on the back burner". The VP answers by saying "No, continue with the current projects. I'll have my secretary match the lists manually".

Application B: Keeping Track of Insurance Agents

Company A is a direct and reinsurance company, and its clients are other primary insurance companies. Each month each primary insurance company sends Company A a list of insureds, as well as the name of the insurance agent who placed the business. Company A also sells its own products to these same insurance agents. Company A would like to know which agents of the primary companies are also their own agents and which are not. Company A's agent list has more than 55,000 members (agents). Each monthly submission to Company A could easily be one to two thousand insureds from the primary companies. And as before, each primary company has its own way of storing mailing data about its agency force.

How the Fuzzy Match/Merge Application was Developed

The main purpose of this application is its ability to match records from two separate files which do *not* have any common key to link records together. Instead, the files have common fields but can not be matched exactly by the contents of these fields due to the way the data is keyed or maintained. For instance, the name in file A is keyed as “Charles Patridge;” in file B the name is “Chuck Patridge.” A computer cannot conclude that these two data fields are, in fact, the same person. However, an individual can create a computer program(s) to consider these two data fields to be close to matching. And that is what I have attempted to create.

As with most applications, the critical aspect to this particular problem was becoming familiar with the data. To accomplish this task, I took approximately 1,200 records from an actual file and started to match each record with a file consisting of more than 55,000 similar type records. As I matched each record, I analyzed the data from both files and eventually determined an algorithm that could be created to simulate what the human brain does when comparing two similar character strings.

To duplicate this human thought process and utilize the current tools built within the base SAS system, I began the initial program one module at a time. First, I decided the SCAN function would be crucial to the success of my program. In using SCAN, I did not have to worry about the order in which a client’s name is stored. That is, Charles Patridge is the same name as Patridge, Charles. All that is important if that the "words" in one field are "contained" in a character data string of another field. In addition, I realized not all words from one field should or have to be contained in another field. That is, "Charles Patridge" is the same person as "Patridge, Chuck." The difference between these two comparisons is "Chuck" and "Charles."

To compensate for these nicknames and/or abbreviations, I built a SAS macro to "normalize/standardize" the data. For instance, when dealing with addresses, there are certain words which are commonly abbreviated: St for Street, Rd for Road, Dr for Drive, etc. Hence, I had to build several SAS macros to "normalize" my data depending on the field's use. For example, addresses need one kind of normalizing, and the names of clients another kind. I have found that data contents have a certain characteristic based on industry and/or application. One example is the insurance agent application described above. There are numerous insurance agents throughout the United States whose business/corporate name contain the word "AGENCY." However, in the Direct Mail Marketing Company (DMMC) example above, "AGENCY" can be a rather unique word and be useful in the matching process. The point is "normalizing" data should be done with care for each application to maximize the matching (hit ratio) process and minimize “false” hits.

The last ingredient needed for my application is the ability to read one record from one SAS dataset and hold it open while reading through another SAS dataset to look for potential matches with the record from the first file. To accomplish this task, I used the POINT option of the SET statement. A side benefit of the POINT option is the ability to reduce the amount of I/O and CPU time by breaking down the master file (file to be used to determine if a match exists) into subsets based on the application. For instance, the example in Application B tries to match insurance agents from all across the United States. I decided to subset the master file into 50 datasets based on the state in which the agent's address is located. This allows the program to search records based on state instead of searching through all states for a possible match. This search method reduces I/O, CPU time, and execution time.

To enhance the matching process and reduce the manual effort in matching records, I developed several additional techniques. First, I knew I needed to prioritized those matches found. That is, there could be numerous matches in a given situation, and I wanted the most probable match to occur first by some determined method. The method I chose was to count the number of words found in each possible

occurrence, and then rank these matches by the number of words contained in descending order. In real life applications, there could be numerous matches (especially in Metro areas) for a given client, and I did not want the computer program to select just one match. I needed to list all possible matches for a person to make the final decision as to which one(s) are truly the real matches. This is an application where the computer can do a lot of the grunt work but is **limited in making the right/accurate choice**. Hence, I needed to sort the final matches according to the master record with the most words found.

Another technique I added to the matching process was to delete vowels within the data. Due to human error, I determined many words are misspelled because of vowels. For example, "Charlie" sounds the same as "Charley" but is spelled differently because of the vowels. So, to enhance the matching process, I match each "normalized" word by first looking for a possible hit. And then I delete the vowels and perform the same routine looking for a hit by using only the consonants. For example, "Charlie" is not contained in the string "Charley". However, after removing the vowels, "Chrl" is contained in "Chrl." And I just have increased my hit ratio by one word.

A side note for SAS users of release 6.07: I tried using the SOUNDEX() function in my application. It would have required parsing every word in the source file, and would have added a significant amount of processing to the matching process. The gains I received did not seem to warrant utilizing it. However, it could be used for those cases where it seems to make sense, such as "SMITH" vs "SMYTHE", "JONES" vs "BONES", “KAROL” vs “CAROL”, etc.

Having discussed the basic concepts and necessary ingredients to the application, it is time to discuss the actual program using an example I have created. First, assume both files (master and transaction) are matched, have been "normalized/standardized", and edited for obvious errors (state code matches zip code, all characters are upper case, etc.). However, the Fuzzy program does not really need the state or city, as I use the zip code as the primary key to determine if a possible match record is within a postal boundary. More specifically, I use only the first three digits of the zip code to determine if a record is within a geographic area I am trying to match. Once the files are in order, the process of matching the transaction records against the master list begins.

After all transaction records have been through the matching process, the matched records are then accumulated and sorted by the most probable match first. These records are then printed and saved to a SAS dataset (**DUPS**) for an on-line selection system. Using a printed report one can visually select which records are indeed matched. Once manually marked, an operator can use the SAS system (FSEDIT) and to mark/update the DUPS file with the appropriate matches. And then the DUPS file can be used to go back and match up with the original transaction file for further processing.

I will present a visual tour of the entire process from start to finish with supporting exhibits to better illustrate the Fuzzy Match/Merge routine and to required steps to make the ultimate goal of matching as successful as possible under the Visual Tutorial section of this article.

I hope this article and the examples listed above have demonstrated the usefulness and flexibility of the SAS system, as well as the fuzzy feeling SAS can provide! I thank you very much for your time and desire to learn about Fuzzy matches. I would be delighted to hear from you on how this routine has been used within your environment, and the results it produced. You can reach me for consultation on the software routine at the address listed near the end of this article.

A Visual Tour of The Fuzzy Match/Merge Routine

The Transaction file (Figure 2) displays the 10 records that you will try to match against three records in the Master file (Figure 3). The amount of data used is small and purely for tutorial purposes to illustrate the Fuzzy Match/Merge process.

Figure 2: Transaction File in its Original Form before Conversion and Sorting

1	**C Patridge**	**172 Monce Road**	**Burlington**	**Connecticut**	**06013**
2	**Partridge Charlie**	**Monce Road Box 172**	**Lake Garda Area**	**CT.**	**06035**
3	**Richard Gibson**	**123 George Washington Turnpike**	**Burlington**	**CT.**	**06013**
4	**Jim Good Night**	**1 SAS Circle**	**Cary**	**NC**	**27513**
5	**Good-Knight James**	**SAS Campus Drive**	**Cary**	**NC**	**27513**
6	**Angell Elizabeth**	**SAS Campus Drive**	**Cary**	**NC**	**27513**
7	**Charles Todd**	**176 Monce Road**	**Burlington**	**CT.**	**06013**
8	**David Congdon**	**168 Monce Road**	**Burlington**	**CT.**	**06013**
9	**Alice Smith**	**123 George Washington Turnpike**	**Burlington**	**CT.**	**06085**
10	**William Smythe**	**675 G. Washington Tpke**	**Burlington**	**CT.**	**06013**

Figure 3: Master File in its Original Form without Sorting or Conversions

1	**Charles Patridge**	**172 Monce Road**	**Burlington**	**CT**	**06013**
2	**James Goodnight**	**SAS Campus Drive**	**Cary**	**NC**	**27513**
3	**Betty Angell**	**SAS Campus Drive**	**Cary**	**NC**	**27513**

You will notice in the Transaction file (Figure 2) that records 1 and 2 and records 4 and 5 are similar (possible duplicates) and will be used later to demonstrate how you can find duplicate records within the same file. The Master file is very small and does not contain duplicate records.

The first task at hand is to perform a frequency distribution of all words in both (word is defined as a character string within a SAS variable that contains no special characters) the Transaction and Master files by joining the two files and executing the TOKENIZE program. You can see by Figure 4 that this is a simple Proc Freq report that lists all possible words encountered in both files. Normally, this report is very large and takes considerable manual effort to review and find which words need to be "standardized" or "normalized". The TOKENIZE program reads a character field and creates a record for every word found in that character field for all records; hence, the reason for such a large listing.

Figure 4: Frequency Distribution of Words – Names and Streets

Frequency Distribution of Name

OBS	TOKEN	COUNT	PERCENT
1	ALICE	1	3.70370
2	ANGELL	2	7.40741
3	BETTY	1	3.70370
4	CHARLES	2	7.40741
5	CHARLIE	1	3.70370
6	CONGDON	1	3.70370
7	DAVID	1	3.70370
8	ELIZABETH	1	3.70370
9	GIBSON	1	3.70370
10	GOOD	2	7.40741
11	GOODNIGHT	1	3.70370
12	JAMES	2	7.40741
13	JIM	1	3.70370
14	KNIGHT	1	3.70370
15	NIGHT	1	3.70370
16	PARTRIDGE	1	3.70370
17	PATRIDGE	2	7.40741
18	RICHARD	1	3.70370
19	SMITH	1	3.70370
20	SMYTHE	1	3.70370
21	TODD	1	3.70370
22	WILLIAM	1	3.70370

Frequency Distribution of Street

OBS	TOKEN	COUNT	PERCENT
1	123	2	4.8780
2	168	1	2.4390
3	172	3	7.3171
4	176	1	2.4390
5	675	1	2.4390
6	BOX	1	2.4390
7	CAMPUS	4	9.7561
8	CIRCLE	1	2.4390
9	DRIVE	4	9.7561
10	GEORGE	2	4.8780
11	MONCE	5	12.1951
12	ROAD	5	12.1951
13	SAS	5	12.1951
14	TPKE	1	2.4390
15	TURNPIKE	2	4.8780
16	WASHINGTON	3	7.3171

Each field used for the matching process (Name and Street) needs to be run through the TOKENIZE routine separately and analyzed. From these lists and analyses, the program constructs the entries in the STDNAME (Figure 5) and STDADDR (Figure 6) files. The first column (Name) in each file is the original form of the data as found in the Proc Freq report. The second column (Standard Name) is the conversion form of the word which will be used during the matching process. If there is a "_" in the second column, then the original word will in essence be removed (i.e., **street** gets converted to **st**, and then **st** gets converted to "_"). The reason for removing specific words has to do with minimizing the possibility of "**false**" hits due to common words found in fields containing such items as address. Many people have an address containing the word "**street**", "**drive**", "**road**", etc., and these words would cause the fuzzy match/merge routine to excessively match records on such words. A similar process for handling peoples' first names is required ("**Charles**" to "**Chas**", "**Charlie**" to "**Chas**", "**Elizabeth**" to "**Liz**", "**Betty**" to "**Liz**", "**Bettie**" to "**Liz**", etc.).

Figure 5: Standardizing First Names

Name	Standard Name	Remove "Noise"	Name	Standard Name	Remove "Noise"
Charles	Chas		Jimmie	Jim	
Charlie	Chas		Jimmy	Jim	
Charley	Chas		Jamie	Jim	
Chuck	Chas		Elizabeth	Liz	
Richard	Dick		Betty	Liz	
Rick	Dick		Beth	Liz	
Rickie	Dick		William	Bill	
Rickey	Dick		Will	Bill	
Ricky	Dick		Willy	Bill	
Rich	Dick		Willie	Bill	
James	Jim				
Jame	Jim				

In Figure 5, no words are being eliminated, as all the words are needed in the matching process.

Figure 6: Standardizing Addresses

Address	Standard Address	Remove "Noise"
AVENUE	AVE	_
BOULEVARD	BLVD	_
DRAWER	BOX	_
P O BOX	BOX	_
POBOX	BOX	_
CIRCLE	CIR	_
CENTER	CTR	_
DRIVE	DR	_
EAST	E	_
FLOOR	FL	_
FORT	FT	_
HIGHWAY	HWY	_
LANE	LN	_
NORTH	N	_
PARK	PK	_
ROAD	RD	_
SOUTH	S	_
SQUARE	SQ	_
STREET	ST	_
TURNPIKE	TPKE	_
TERRACE	TR	_
WEST	W	_
CROSSING	XING	_
TURNPIKE	TPKE	_

In Figure 6, these address words are completely removed during the matching process as marked with an "_" in the 'Remove "NOISE"' column.

When dealing with Names of Companies, you have again the high probability of common words used in the legal name of a company (for example, The ABC Company, The Hartford, XYZ Limited). Such words as "**The**", "**and**", "**of**", "**Company**", "**Co**" are too common and need to be removed or "standardized" for the matching process. Looking at the STDBUSN data file (Figure 7) provides you with an idea of the standardizing that needs to be done when dealing with company names.

Figure 7: Standardizing Company Names

Company Name	Standard Company Name	Remove "Noise"	Company Name	Standard Company Name	Remove "Noise"
The	The	_	Enterprize	Entpze	
And	And	_	Incorp	Inc	_
Of	Of	_	Incorporated	Inc	
Agency	Agy		Insurance	Ins	
Business	Busn		Limited	Ltd	_
Casualty	Cslty		Management	Mgmt	
Company	Co	_	Managing	Mgmt	
Corporate	Corp		Manufacture	Mfg	
Corporated	Corp		Manufacturing	Mfg	
Enterprise	Entpze		Office	Ofce	

In Figure 7, some words will be eliminated, as they would cause too many "false" hits. Many companies include these special words in their legal name.

This process of standardizing names and addresses is the "**REAL**" key to making the fuzzy match/merge process successful. If this stage of the process is overlooked or not given sufficient attention, your results from the Fuzzy Match/Merge routine can easily produce too many "**false**" hits or not catch matches when there should be matches. Remember that this Fuzzy Match/Merge routine is not 100% accurate. Webster's definition of "fuzzy" is "not clear", "indistinct", "confused" and "not clearly worked out".

After the **STDNAME** and **STDADDR** data files have been completed, it is time to execute the **STDLIST** program which will take the Transaction and Master Files and run them through the conversion process for the Name and Address fields.

Figure 8: Transaction File After Conversion

1	ALICE SMITH	123 GEORGE WASHINGTON	BURLINGTON	CT	06085
2	ANGELL LIZ	SAS CAMPUS	CARY	NC	27513
3	C PATRIDGE	172 MONCE	BURLINGTON	CT	06013
4	CHAS TODD	176 MONCE	BURLINGTON	CT	06013
5	DAVID CONGDON	168 MONCE	BURLINGTON	CT	06013
6	GOOD KNIGHT JIM	SAS CAMPUS	CARY	NC	27513
7	JIM GOOD NIGHT	1 SAS	CARY	NC	27513
8	PARTRIDGE CHAS	MONCE 172	LAKE GARDA AREA	CT	06035
9	DICK GIBSON	123 GEORGE WASHINGTON	BURLINGTON	CT	06013
10	BILL SMYTHE	675 G WASHINGTON TPKE	BURLINGTON	CT	06013

Figure 8 shows how the sequence of records has changed from its original position due to a sort by original name

Figure 9: Master File in Converted Form

1	LIZ ANGELL	SAS CAMPUS	CARY	NC	27513
2	CHAS PATRIDGE	172 MONCE	BURLINGTON	CT	06013
3	JIM GOODNIGHT	SAS CAMPUS	CARY	NC	27513

Figure 9 shows the converted form of the master file, sorted by the original name.

By looking at figures 8 and 9 you can see what this conversion process did to each of the files.

- All characters were converted to uppercase.
- Each file was sorted by the original name.
- The conversion of words for the Name and Address fields has taken place.
- Special characters (/ \ , . - +, etc) were replaced with blanks.

Figure 10: Transaction File Before and After Conversion

	NAME	Address	City	State	Zip Code
1	C Patridge	172 Monce Road	Burlington	Connecticut	06013
	C PATRIDGE	*172 MONCE*	*BURLINGTON*	*CT*	*06013*
2	Partridge Charlie	Monce Road Box 172	Lake Garda Area	CT.	06035
	PARTRIDGE CHAS	*MONCE 172*	*LAKE GARDA AREA*	*CT*	*06013*
3	Richard Gibson	123 George Washington Turnpike	Burlington	CT.	06013
	DICK GIBSON	*123 GEORGE WASHINGTON*	*BURLINGTON*	*CT*	*06013*
4	Jim Good Night	1 SAS Circle	Cary	NC	27513
	JIM GOOD NIGHT	*1 SAS*	*CARY*	*NC*	*27513*
5	Good-Knight James	SAS Campus Drive	Cary	NC	27513
	GOOD KNIGHT JIM	*SAS CAMPUS*	*CARY*	*NC*	*27513*
6	Angell Elizabeth	SAS Campus Drive	Cary	NC	27513
	ANGELL LIZ	*SAS CAMPUS*	*CARY*	*NC*	*27513*
7	Charles Todd	176 Monce Road	Burlington	CT.	06013
	CHAS TODD	*176 MONCE*	*BURLINGTON*	*CT*	*06013*
8	David Congdon	168 Monce Road	Burlington	CT.	06013
	DAVID CONGDON	*168 MONCE*	*BURLINGTON*	*CT*	*06013*
9	Alice Smith	123 George Washington Turnpike	Burlington	CT.	06085
	ALICE SMITH	*123 GEORGE WASHINGTON*	*BURLINGTON*	*CT*	*06085*
10	William Smythe	675 G. Washington Tpke	Burlington	CT.	06013
	BILL SMYTHE	*675 G WASHINGTON TPKE*	*BURLINGTON*	*CT*	*06013*

Figure 10 shows the Transaction File in its original form and its *converted form (in italicized red)* after being processed by the STDLIST routine.

Seeing the file before and after the conversion makes the comparison even easier to see.

- In some records, first names were converted.
- In the address field, certain words (Road, Drive, Box, Turnpike, Circle) were eliminated.
- All character fields were converted to upper case, and allspecial characters were removed.
- In the state field, used ZIPSTATE function to get standard state postal code.

Figure 11 shows the Master File in its original form and in its converted form (in italicized red) after being processed by the STDLIST routine.

Figure 11: Master File Before and After Conversion

	Name	Address	City	State	Zip Code
1	Charles Patridge	172 Monce Road	Burlington	CT	06013
	CHAS PATRIDGE	*172 MONCE*	*BURLINGTON*	*CT*	*06013*
2	James Goodnight	SAS Campus Drive	Cary	NC	27513
	JIM GOODNIGHT	*SAS CAMPUS*	*CARY*	*NC*	*27513*
3	Betty Angell	SAS Campus Drive	Cary	NC	27513
	LIZ ANGELL	*SAS CAMPUS*	*CARY*	*NC*	*27513*

- *First names were converted.*
- *Certain words in the Address field were eliminated (Road, Drive).*
- *All character fields were converted to upper case.*

Now that each file has been "normalized/standardized", it is time to execute the **Fuzzy Match/Merge** program. Because the Transaction and Master file are distinct files, the macro variable "**DUP**" in the Fuzzy program should be assigned a value to anything but "**within**" indicating the matching process is not to consider finding duplicate records within the same file and is trying to find records in the Transaction file that may be a match to one or more records in the Master file. Any value for "**DUP**" will work when trying to find matches between two distinct files except "**within**".

The resulting report (Figure 12) shows all transaction records that could be possible matches to any record in the Master file. The records shown in blue are the "good" hits, and those in red are "false" hits. In each boxed area of the report, the first 3 lines are the Transaction Record, and the second 3 lines are the possible match of a Master record. At the bottom of each box are the words which were found to be common in both records. Notice that record 2 from the Transaction file (ANGELL LIZ) matched with record 1 (LIZ ANGELL) of the Master file. The words found to be common between these two records were ANGELL, CAMPUS, LIZ, LZ, NGLL and SAS (sorted alphabetically). The words LZ and NGLL are the result of deleting the vowels from LIZ (LZ) and ANGELL (NGLL).

Figure 12 shows the Transactions Records that potentially could match Master Records. The 1st 3 lines are from the Transaction File. The 2nd 3 lines are from the Master File. The Last line shows the words found that determined possible match. Words with * were caught with the soundex function.

Figure 12: Fuzzy Match/Merge Report

2 ANGELL LIZ **SAS CAMPUS** **CARY NC 27513** **1 LIZ ANGELL** **SAS CAMPUS** **CARY NC 27513** **ANGELL,CAMPUS,LIZ,LZ,NGLL,SAS,**	**3 C PATRIDGE** **172 MONCE** **BURLINGTON CT 06013** **1 CHAS PATRIDGE** **172 MONCE** **BURLINGTON CT 06013** **172,MONCE,PATRIDGE,PTRDG,**
4 CHAS TODD **176 MONCE** **BURLINGTON CT 06013** ***1 CHAS PATRIDGE*** **172 MONCE** **BURLINGTON CT 06013** ***CHAS,CHS,MONCE,***	**6 GOOD KNIGHT JIM** **SAS CAMPUS** **CARY NC 27513** **2 JIM GOODNIGHT** **SAS CAMPUS** **CARY NC 27513** **CAMPUS,GD,GOOD,JIM,JM,KNIGHT*,SAS,**
7 JIM GOOD NIGHT **1 SAS** **CARY NC 27513** **2 JIM GOODNIGHT** **SAS CAMPUS** **CARY NC 27513** **GD,GOOD,JIM,JM,NGHT,NIGHT,SAS,**	**8 PARTRIDGE CHAS** **MONCE 172** **LAKE GARDA AREA CT 06035** **1 CHAS PATRIDGE** **172 MONCE** **BURLINGTON CT 06013** **172,CHAS,CHS,MONCE,PARTRIDGE*,**

Figure 12 shows the results of matching the Transaction. The records in blue are possible matches and the records in red are "false" hits. Words with an "*" indicate that they were used during the SNDSLIKE routine. Forcing the elimination of vowels added the number of words found between any two records and could increase the number of "false" hits.

In Figure 12 (2nd Column, 2nd Row), you will see that GOOD KNIGHT JIM matched with JIM GOODNIGHT using the following words: CAMPUS, GD, GOOD, JIM, JM, KNIGHT*, and SAS. The word **KNIGHT** does not occur in both records but **KNIGHT does sound like NIGHT**, and the reason for the "*" next to KNIGHT signifies the word was caught using the **SNDSLIKE** macro which incorporates **SOUNDEX( )** function.

That is how the Fuzzy Match/Merge Routine works and how it produces reports! The reports often change an appearance based on the user's needs and desires. I choose to use PROC FORMS as an easy way to display one record with multiple SAS variables ordered in a special way to illustrate two distinct records.

Before I demonstrate the actual word for word matching process, I would like to illustrate how by using the same program (**FUZZY**) you can find duplicate records within the same file. For simplicity, **copy the Transaction file and name the copy, MASTE**R. The reason is the FUZZY program needs two files to compare records with even when the two files are identical. Then in the Fuzzy program assign the macro variable "**DUP**" to "**within**" and re-execute the **FUZZY** program.

Figure 13 illustrates the output of looking for duplicates within the Transaction File. The macro variable "DUP" informs the program that the two files (TRANSACTION and MASTER) are in fact the same file, and when comparing records between both files the program ignores all records in both files, that contains the same record number. This prevents the output from getting larger, since every record should always match itself.

The figure shows the transactions records that potentially could match Master Records. The 1st three lines are from the Transaction File. The 2nd three lines are from the Master File. The last line shows the words found that determined a possible match. Words with * were caught with the soundex function.

Figure 13: Matching Transaction File with Itself for Duplicate Records

1 ALICE SMITH 123 GEORGE WASHINGTON BURLINGTON CT 06085 10 BILL SMYTHE 675 G WASHINGTON TPKE BURLINGTON CT 06013 SMTH,	3 C PATRIDGE 172 MONCE BURLINGTON CT 06013 8 PARTRIDGE CHAS MONCE 172 LAKE GARDA AREA CT 06035 PATRIDGE*,
4 CHAS TODD 176 MONCE BURLINGTON CT 06013 8 PARTRIDGE CHAS MONCE 172 LAKE GARDA AREA CT 06035 CHAS,CHS,MONCE,	6 GOOD KNIGHT JIM SAS CAMPUS CARY NC 27513 7 JIM GOOD NIGHT 1 SAS CARY NC 27513 GD,GOOD,JIM,JM,KNIGHT*,SAS,
7 JIM GOOD NIGHT 1 SAS CARY NC 27513 6 GOOD KNIGHT JIM SAS CAMPUS CARY NC 27513 GD,GOOD,JIM,JM,NGHT,NIGHT,SAS,	8 PARTRIDGE CHAS MONCE 172 LAKE GARDA AREA CT 06035 4 CHAS TODD 176 MONCE BURLINGTON CT 06013 CHAS,CHS,MONCE,
8 PARTRIDGE MONCE 172 LAKE GARDA AREA CT 06035 3 C PATRIDGE 172 MONCE BURLINGTON CT 06013 PARTRIDGE*,	10 BILL SMYTHE 675 G WASHINGTON TPKE BURLINGTON CT 06013 1 ALICE SMITH 123 GEORGE WASHINGTON BURLINGTON CT 06085 SMTH,

By making the Master File the same as the Transaction File (make a copy), you can easily look for duplicates within the same file. As you can see in red the "False" hits, and in blue, the possible duplicate records within the Transaction File. And because this is an internal (within the same file) matching process, no record is ever matched against itself.

In Figure 13, you will notice that record 1 in the Transaction file matches record 10 of the Master file (first row, firstcol). And looking at the fourth row and second col., you can see that Record 10 in the Transaction file matches Record 1 in the Master file. **IT SHOULD** because the Transaction and Master File are the same! Hence for every matched set of records (**1,10 | 4,8 | 7,6 | 8,3**) there will be an exact set of matches with the ordered pair reversed (**10,1 | 8,4 | 6,7 | 3,8**).

Now is the time to look at the actual matching process in more detail.

Details

Look at Figure 14. In the left column is the Transaction Record that will try to match the record in the far right column (Master Record). The middle column (Word Scan Process) is the step by step process of analyzing each word in the transaction record.

Figure 14: Step by Step Processing of Each Word

Matching Process		
Incoming Record	**Word Scan Process**	**Master Record**
Chuck Partridge Box 172, Moncee Road Unionville, CT 06013		Charles Steven Patridge 172 Monce Road Burlington, CT 06013
Chuck	? = Charles	NO
	? = Steven	NO
	? = Patridge	NO
Chck	? = Chrls	NO
	? = Stvn	NO
	? = Ptrdg	NO
Partridge	? = Charles	NO
	? = Steven	NO
	? = Patridge	NO
	? *sound like Patridge*	*YES*
Prtrdg	? = Chrls	NO
	? = Stvn	NO
	? = Ptrdg	NO
Skip Box		
172	? = *172*	*YES*
Moncee	? = Monce	NO
	? = *Mnc, after vowels removed*	*YES*
Skip Road		
Skip City and State		
Get Next Incoming Record		

1. Start with the first word, **Chuck**.
   - Does it match Charles ? NO
   - Does it sound like Charles ? NO
   - Does it match Steven ? NO
   - Does it sound like Steven ? NO
   - Does it match Patridge ? NO
   - Does it sound like Patridge ? NO

2. If no match so far, try the same word without the vowels (**Chck**).
   - Does it match Chrls ? NO
   - Does it sound like Chrls ? NO
   - Does it match Stvn ? NO
   - Does it sound like Stvn ? NO
   - Does it match Ptrdg ? NO
   - Does it sound like Ptrdg ? NO

3. Now go to the second word, **Partridge,** and go through the same process.
   Does it match Charles ? NO
   Does it sound like Charles ? NO
   Does it match with Steven ? NO
   Does it sound like Steven ? NO
   Does it match Patridge ? NO
   Does it sound like Patridge ? YES

4. Now eliminate the vowels in the second word (**Prtrdg**) and try to find a match.
   Does it match Chrls ? NO
   Does it sound like Chrls ? NO
   Does it match Stvn ? NO
   Does it sound like Stvn ? NO
   Does it match Ptrdg ? NO
   Does it sound like Ptrdg ? NO

5. Get the next word, **BOX**. Box, however, has been deleted through the standardization process and will not be used.

6. Next go to the Address field and do a similar process as with Name.

The rest is self explanatory! Once finished with the record, go and get the next record and repeat the above process.

How Long Does It Take?

A question I often receive from clients or from an audience on the Fuzzy Match/Merge process is "how long does it take". That is hard to say. It depends on several independent/dependent variables.

- First, the number of records in the Transaction and Master files have a direct impact on execution time.
- Second, are there any variables (fields) such as Zip Code that can be used to select a much smaller set of records from the Master File to use in matching with the Transaction record?
- If there isn't some way to select a smaller group of records in the Master File, then maybe another rule can be applied.
  - Maybe another variable can be used, for example, SIC code, sales volume, country code, state code.
  - Even rules like taking the first letter of the first three words in the Transaction record to select only those records in the Master File that start with those three letters.
  - When dealing with files about people, maybe gender and/or birth dates can be used to select records from the Master File.
  - Anything reasonable should be sought in order to reduce the number of records to match against provided it does not prohibit you from the desired results.

If you are forced to read the Master File for each record in the Transaction File it is easy to calculate how many times the Master file will be re-read, and you can see that execution time will increase dramatically. In the example illustrated in this Visual Tutorial, you can see that I am using the first three positions (see Figure 15) of the Zip Code field to select a subset of the Master File to use in matching with the Transaction File.

Figure 15: Using Zip Code to select a subset of records in the Master File

Incoming Record	Zip Code Region	Master Record
Chuck Partridge **Box 172, Monce Rd** **Unionville, CT 06013**	**06013** **06085 YES**	**Charles Steven Patridge** **172 Monce Road** **Burlington, CT 06085**
	06013 **06085 YES**	**Howard Miller** **School Street** **Farmington, CT 06085**
		James Fusco **Whispering Rod Road** **Farmington, CT 06035**
	06013 **06101 NO**	**Bill Clinton** **One White House Road** **Avon, CT 06101**

By narrowing down the Zip Code Region, you can increase the speed of the search process, but you give up the possibility of finding possible matches or duplicate records. There is a trade off of speed or accuracy as well as the manual effort to review a larger set of likely "false" hits.

On a recent project, the transaction file consisted of more than 100,000 records and the Master file contained 22,000 records. To standardize these files took approximately 10 hours of CPU time on an Alpha 2100 machine. Running the same files through the same process took less than 3 hours on a Dual Pentium 300 Mhz desktop computer. So this Fuzzy Match/Merge routine can execute with significantly varied times depending on the operating system and the activity taking place on the machine doing the process. This Fuzzy Match/Merge routine is intensely CPU bound and less bound to I/O. By the reviewing the statistics generated by the FULLSTIMER host option, you can determine your system performance. In most execution tests, the clock time is usually very close to the CPU time.

Figure 16 summarizes the key features needed to make the Fuzzy Match/Merge Routine work successfully.

Figure 16

Key SAS Features needed to make Fuzzy Match Work Properly	Reason(s)
1. Open two SAS files simultaneously	POINT= of a SET statement * to increase Efficiency/Reduce CPU
2. Split the Master file by zip code region	SET with use of a "Where" clause * to increase Efficiency/Reduce IO
3. Search a character string efficiently	INDEX(source, excerpt) * does source contain excerpt - "hit"
4. Loop through a character string, word by word	SCAN(argument, N, Delimiters) * search character string word by word
5. Eliminate unnecessary characters in a string	COMPRESS(argument, 'characters to remove') * eliminate vowels
6. Eliminate multiple embedded blanks	COMPBL(argument)
7. Replace a Word with another Word	TRANWRD(String, Original, Convert)
8. Translate a character with another character	TRANSLATE(String, Replacement, Character to Replace)
9. Use of Custom Macros	REMOVEDB – remove multiple consecutive characters DROPCHAR – remove a string from end of a word SNDSLIKE – use SOUNDEX function for like sounding words EMPTYYN – find number of OBS within a SAS Dataset
Key Process to Make the Fuzzy Match Routine Successful	
1. Tokenize key fields (name and address)	TOKENIZE – split a character field into words
2. From step 1, build stdname, stdaddr, stdbusn files	See individual files (stdname, stdaddr, stdbusn) For example of standardization words; those words converted to "_" Are eliminated from the contents of the character variable being processed.
3. Use the Fuzzy program to search for possible matches or duplicate records	Scanning word by word, record by record build a transaction file of possible matching words, using vowel and sound like processing on both name and street variable as long as at least 1 word in name is found.

Some Precautionary Thoughts When Using the Fuzzy Match/Merge Routine

1. Remember this is not an exact science and is prone to "fuzzy" results. I recommend not using the output results as an automated input source to another application without first reviewing the results.
2. The purpose of this application is to find records that may be duplicated or matched to another file and to minimize the manual effort to determine which ones. Ideally this routine will significantly help in finding "a needle in the haystack" by reducing the size of the haystack to a manageable level and isolating which straws to look under for the "needle".

3. When dealing with domestic and foreign names, pay special attention to the foreign spelling of names (i.e., Associacion /Association, Nationale / National, Les / The, De / Of, etc.) as well as addresses. Frequently in dealing with foreign names, the ending of words can vary and need to be accounted for.
4. Often in Company Names, endings of words such as "s", "ers", "ies", "ing", and "tion", etc. need to be converted or dropped. The macro program **DROPCHAR** can be used to handle such situations. Be careful in using this macro as the order in which you drop endings can provide some weird results. You should use a small sample of data to test your sequence of dropping endings to ensure the order you choose is appropriate.
5. Another situation to be wary of is dealing with an outside Vendor's file such as Dunn & Bradstreet or Compustat. These vendors as well as others tend to code words in their own unique abbreviations which you need to understand and compensate for in building the STDNAME or STDBUSN files. Examples are HLDG for Holding, LTD for Limited, Natl for National, etc.
6. As for execution time, the more data fields you have for subsetting the Master file, the faster the matching process will run. When trying to match people, fields using zip code, gender, or birth dates are recommended. Make these fields available in both the Master and Transaction files. As for companies, the full legal name and address including zip code tends to be enough in finding matches.
7. **It is highly recommended to use copies of your data and not the original files as data transformations are not reversible!**

Conclusion

In conclusion, I hope that I have provided sufficient discussion and illustrations to inform you when the Fuzzy Match/Merge could be used and the varied areas to watch out for in trying to apply these programs to your own data. From my experience, it takes about 30-40 hours to build the STDNAME/STDBUSN and STDADDR files, which are essential and critical to the success of the matching programs. I am available for consultation and advice should you find yourself in need of the FUZZY Match/Merge routines.

References:

SAS Institute Inc. (1987), *SAS Applications Guide, 1987 Edition*

SAS Institute Inc. (1990), *SAS Guide to Macro Processing: Version 6, Second Edition*

SAS Institute Inc. (1990), *SAS Language: Reference, Version 6, First Edition*

SAS Institute Inc. (1991), SAS Technical Report P-222, *Changes and Enhancements to Base SAS Software, Release 6.07*

SAS Institute Inc. (1993), *SAS Macro Language: Course Notes*

SAS Institute Inc. (1994), *SAS Macro Facility Tips and Techniques, Version 6, First Edition*

SAS Institute Inc. (1997), *SAS Macro Language: Reference, First Edition*

Appendix A -- Data Listings of Files used in Fuzzy Match

```
SOURCE LISTING
FILE REF = PRTFUZZ AND FILE TYPES = CSV

  -------------------------- FILEREF=PRTFUZZ --------------------------

OBS      SRCCODE

1        MASTER.CSV
2        STDADDR.CSV
3        STDBUSN.CSV
4        STDNAME.CSV
5        TRANSACT.CSV



MEMBER = MASTER.CSV

OBS   SRCCODE

1     Charles Patridge,172 Monce Road,Burlington,CT,06013
2     James Goodnight,SAS Campus Drive,Cary,NC,27513
3     Betty Angell,SAS Campus Drive,Cary,NC,27513



MEMBER = STDADDR.CSV

OBS    SRCCODE

 1    Avenues,Ave
 2    Avenue,Ave
 3    Ave,_
 4    Boulevard,Blvd
 5    Blvd,_
 6    Circle,Cir
 7    Cir ,_
 8    Crossing,Xing
 9    Xing,_
10    Drawer,Box
11    Draw ,Box
12    Box,_
13    Drive,Dr
14    Dr,_
15    East,E
16    East,_
17    Floor,Fl
18    Floor,_
19    Fort,Ft
20    Ft,_
21    Heights,Hgt
22    Height,Hgt
23    Hgt,_
24    Highway,Hwy
25    Hwy,_
26    Lane,Ln
```

```
27    Ln,_
28    North,N
30    Park,Pk
31    Pk,_
32    Post Office Box,Box
33    P O Box,Box
34    PO Box,Box
35    POBox,Box
36    Box,_
37    Pointe,Pt
38    Point,Pt
39    Pt,_
40    Road,Rd
41    Rd,_
43    Rt,_
44    South,S
45    S,_
47    Square,_
48    Street,St
49    Street,_
50    Terrace,Tr
51    Tr,_
52    Turnpike,Tpk
53    Tpk,_
54    West,W
55    West,_
```

MEMBER = STDBUSN.CSV

```
OBS    SRCCODE

 1    Agency,Agy
 2    Business,Busn
 3    Casualty,Cslty
 4    Company,Co
 5    Co,_
 6    Corporate,Corp
 7    Corporated,Corp
 8    Enterprise,Entpze
 9    Enterprize,Entpze
10    Incorp ,Inc
11    Incorporated,Inc
12    Insurance,Ins
13    Limited,Ltd
14    Ltd,_
15    Management,Mgmt
16    Managing,Mgmt
17    Manufacture,Mfg
18    Manufacturing,Mfg
19    Office,Ofce
```

```
MEMBER = STDNAME.CSV

OBS    SRCCODE

 1    Charles,Chas
 2    Charlie,Chas
 4    Chuck,Chas
 5    Richard,Dick
 6    Rick,Dick
 7    Rickie,Dick
 8    Rickey,Dick
 9    Ricky,Dick
10    Rich ,Dick
11    James,Jim
12    Jame ,Jim
13    Jimmie,Jim
14    Jimmy,Jim
15    Jamie,Jim
16    Elizabeth,Liz
17    Betty,Liz
18    Beth,Liz
19    William,Bill
20    Will ,Bill
21    Willy,Bill
22    Willie,Bill
23    Billy,Bill
24    Billie,Bill



MEMBER = TRANSACT.CSV


OBS    SRCCODE

 1    C  Patridge,172 Monce Road,Burlington,Connecticut,06013
 2    Partridge Charlie,Monce Road Box 172,Lake Garda Area,CT.,06035
 3    Richard Gibson,123 George Washington Turnpike,Burlington,CT.,06013
 4    Jim Good Night,1 SAS Circle,Cary,NC,27513
 5    Good-Knight James,SAS Campus Drive,Cary,NC,27513
 6    Angell Elizabeth,SAS Campus Drive,Cary,NC,27513
 7    Charles Todd,176 Monce Road,Burlington,CT.,06013
 8    David Congdon,168 Monce Road,Burlington,CT.,06013
 9    Alice Smith,123 George Washington Turnpike,Burlington,CT.,06085
10    William Smythe,675 G. Washington Tpke,Burlington,CT.,06013
```

Appendix B – Source Listings of SAS Programs/Macros for Fuzzy Match

```
/*****************************************************************/
/*       Appendix B - SAS Code files for Obswww15 - Patridge     */
/*                                                               */
/*                                                               */
/*     'Source Listings of SAS Programs/Macros for Fuzzy Match'  */
/*                          SOURCE LISTING                       */
/*                                                               */
/* -------------------- FILEREF=PRTFUZZ ----------------------   */
/*                                                               */
/*                        OBS      SRCCODE                       */
/*                                                               */
/*                         1     DROPCHAR.SAS                    */
/*                         2     EMPTYYN.SAS                     */
/*                         3     FUZZY.SAS                       */
/*                         4     NORECDS.SAS                     */
/*                         5     READSTD.SAS                     */
/*                         6     REMOVEDB.SAS                    */
/*                         7     SNDSLIKE.SAS                    */
/*                         8     STDLIST.SAS                     */
/*                         9     TOKENIZE.SAS                    */
/*****************************************************************/
```

```
/*************************************************************/
/*** MACRO DROPCHAR.SAS                                    ***/
/***                                                       ***/
/***  Delete a string of characters from the end of any word ***/
/*************************************************************/

%macro dropchar( string, char );
 zzcnt = 0;
 zzl = length( "&char" );
 do zzw = 1 to 100;/*** how many words are there in string ***/
    if ' ' ne scan( &string, zzw) then zzcnt + 1;
 end;
 zzw = 0;
 length zzwordo $ 200 zzword $ 200 ;
 do zzw = 1 to zzcnt; /*** loop for as many words as found ***/
  zzwordo = scan(&string,zzw); /*** keep original word ***/
  zzword  = zzwordo;           /*** new word to be modified ***/
    /*** check the last char of the word and set to blank ***/
    zzword = left(reverse(zzword));
    if substr(zzword,1,zzl)=reverse("&char") then substr(zzword,1,zzl)=' ';
    zzword = left(reverse(zzword));
  zzword  = compress(zzword, ' '); /*** remove all internal blanks ***/
            /*** replace original word with new word ***/
            /*** and remove multiple blanks ***/
  &string = compbl (tranwrd( &string,trim(zzwordo),trim(zzword)));
 end;
  drop zzl zzw zzcnt zzwordo zzword; /*** drop these vars ***/
%mend dropchar;
```

```
 ********************************************************************;
 **  PROGRAM: EMPTYYN  (MACRO)                                       ;
 **  AUTHOR: CHUCK PATRIDGE                                          ;
 **  DATE: 10/12/94                                                  ;
 **  PURPOSE: DETERMINE IF A SAS DATASET IS EMPTY.                   ;
 **  INPUT PARAMETER: DSNAME (NAME OF SAS DATA SET)                  ;
 **                   ALSO CAN BE BLANK ( USE LAST DATASET CREATED)  ;
 **  OUTPUT: WILL CREATE THE FOLLOWING GLOBAL MACRO VARIABLES        ;
 **            EMPPTYYN              Y=EMPTY, N=NONEMPTY             ;
 **            NUMOBS             WHICH PROVIDES NUMBER OF RECORDS   ;
 **            DSN                WHICH PROVIDES NAME OF DATA SET    ;
 ** SAMPLE CALL: %EMPTYYN(DATASET)                                   ;
 **                                                                  ;
 ********************************************************************;

 %GLOBAL EMPTYYN NUMOBS DSN;
 %MACRO EMPTYYN(DSNAME);
   DATA _NULL_;
    IF "&DSNAME " = " " THEN CALL SYMPUT('DSNAME','_LAST_');
   RUN;

   DATA _NULL_;
    IF 0 THEN SET &DSNAME NOBS=NUMOBS;
    IF NUMOBS > 0 THEN EMPTYYN = 'N';
                  ELSE EMPTYYN = 'Y';
    CALL SYMPUT('EMPTYYN',PUT(EMPTYYN, $1.));
    CALL SYMPUT('NUMOBS' ,PUT(NUMOBS , BEST.));
    CALL SYMPUT("DSN"    ,PUT("&DSNAME" , $VARYING17.));
   RUN;
  %MEND EMPTYYN; RUN;
 ********  END OF Program - Emptyyn ***********;


/*****************************************************************/
/***fuzzy.sas                                                  ***/
/***                                                           ***/
/***  Author: Charles Patridge                                 ***/
/***          PDPC, Ltd.                                       ***/
/***          172 Monce Road                                   ***/
/***          Burlington, CT 06013                             ***/
/***          Home: 860-673-9278 or 860-675-9026               ***/
/***          Email: TVJB41A@prodigy.com                       ***/
/***          Email: Charles_S_Patridge@prodigy.com            ***/
/***                                                           ***/
/***  Copyrighted 1984, 1989, 1993, 1994, 1996, 1998           ***/
/***  With Author's permission, you may use this program       ***/
/***  as agreed upon by Author.  It is illegal to distribute   ***/
/***  this program for profit or misrepresent the original     ***/
/***  owner/author of this Fuzzy Merge Routine(s).             ***/
/***                                                           ***/
/***  Match potential records between two files , duplicates   ***/
/***  After Files have been standardized Names, Addresses, State***/
/***  Find number of words found in Name and Address fields    ***/
/***  Find Pct of words over all words                         ***/
/***                                                           ***/
/***  Ignore finding a record against itself (dup macro var)   ***/
/***  Search Only records that have the same 1st "3"           ***/
/***    characters of zipcode (USA zip codes)                  ***/
/***    and have same gender and birthdate                     ***/
/***                                                           ***/
/***  To Determine Duplicates, First Name or Last Name         ***/
/***    must be matched (order of appearance makes no          ***/
/***    difference).  Initials will throw off a match.         ***/
/***                                                           ***/
/***  This routines employs the use of the soundex function    ***/
/***    to compensate for like sounding names.                 ***/
/***                                                           ***/
/*****************************************************************/
 Libname  merge  "C:\download\merge"; run;
 filename macros "C:\download\merge"; run;
 %include macros( 'emptyyn.sas'  ); /*** Determine NOBS of a DATASET ***/
 %include macros( 'sndslike.sas' ); /*** Soundex Function Macro ***/
 %include macros( 'norecds.sas'  ); /*** display no dups found notification ***/

 %let trnxs   = transact ; /***Transaction Data File - after standardization ***/
 %let master  = master   ; /***Master Data File - after standardization ***/
 %let dup     = within   ; /***if checking for duplicates and Master/Transact are same File ***/
****%let dup     = external ; /*** if checking for matches and Master/Transact are different Files***/
```

```
    %let ziparea = 3; /*** How much of a Zip Area to search duplicates ***/
                      /*** smaller # means larger area, larger # means smaller area ***/
                      /*** ranges from 1 to 5 ***/

    %let maxword = 20; /*** maximum number of words in a given sas variable ***/

    data merge.&trnxs; /*** Transaction File ***/
      set merge.&trnxs;
      recdno = _n_; /*** get record position - used as an identifier for future use ***/
    run;

    data merge.&master; /*** Master File ***/
     set merge.&master;
     recdno = _n_;  /*** get record position - used as an identifier for future use ***/
    run;

    %emptyyn( merge.&trnxs ); /*** how many transaction records ***/
    %let notnxs = &numobs ; /*** set notnxs to number of trnxs records ***/

    proc datasets library=work; delete matches; run; /*** start with empty matches file ***/

    %macro match;
      %do loopk = 1 %to &notnxs ; /*** Loop thru Transaction File record by record ***/
       data _null_;
        recd = &loopk ;
        set merge.&trnxs point=recd; /*** get record by physical pointer ***/
        call symput( 'zipcode' , substr( zipcode, 1, &ziparea) ); /*** get partial of zipcode ***/
        stop;
       run;

       /*** Get Master File Records where Zipcode is similar to Transaction Record ***/
       data srchfile(drop=name street city state zipcode recdno);
        set merge.&master(where=(substr(zipcode,1,&ziparea)="&zipcode") );
        _recdno = recdno; /*** use as an identifier ***/
                        /*** re-assign variable names so as to be able to match words ***/
        _name   = name;
        _street = street;
        _city   = city;
        _state  = state;
        _zipcode= zipcode;
       run;

       %emptyyn( work.srchfile );
       %let nosrch = &numobs ;


data selrecds(drop=ss nn nss);
        length word $ 200. ;
        retain checkstr;
        recd = &loopk ;
        set merge.&trnxs point=recd;
        ss      = 0;
             Do ss = 1 to &nosrch ;
                checkstr = 0;
                recdss = ss ;
                set srchfile point=recdss;
                if "&dup" = 'within' and recdno = _recdno then goto skiprecd; /*** same record ***/
                do nn = 1 to &maxword;

                   /*** try to match part of name   ***/
                   noname  = 0;
                   noname2 = 0;
                   nostre  = 0;
                   word = scan( name  , nn );
                   if word eq ' ' then goto skiprecd; /*** no more words to process ***/
                   if (word ne ' ')            and
                      (length(trim(word)) > 1) and
                      0 ne index( _name  , trim(word) ) then noname=1;
                   /*** check to see if first or last name match ***/
                     if nn in (1 2) and noname = 1 then do;
                        if nn = 1 then word2nd = scan( name, nn+1);
                        if nn = 2 then word2nd = scan( name, nn-1);
                        if (word2nd ne ' ')      and
                           (length(trim(word2nd)) > 1) and
                           0 ne index( _name  , trim(word2nd) ) then noname2=1;
                        if noname  = 1 or  noname2 = 1 then checkstr + 1;
                     end;
```

```
                if noname  = 1 then output;

                noname = 0; /*** force vowel processing ***/
                /*** if no hit, squeeze vowels and try again ***/
                if noname = 0 then do;
                  word = compress( word, 'AEIOUY');
                  if (word ne ' ')            and
                     (length(trim(word)) > 1) and
                     0 ne index( compress(_name, 'AEIOUY')  , trim(word) ) then noname=1;

                  /*** Now use the SoundEx function as last resort ***/
                     if noname = 0 then do;
                       likeword = scan( name, nn );
                       word = trim(likeword) || '*';
                       do inn = 1 to &maxword; /*** use soundex function if vowels failed ***/
                          likeness = scan( _name, inn);
                          if likeness = ' ' then goto skipsnds;
                         if likeword ne ' ' and likeness ne ' ' then do;
                            %sndslike( likeword, likeness );
                         end;
                       end;
                      skipsnds:;
                     end;

                /*** check to see if first or last name match ***/
                  if nn in (1 2) and noname > 0 then do;
                     word2nd = compress( word2nd, 'AEIOUY');
                     if (word2nd ne ' ')      and
                        (length(trim(word2nd)) > 1) and
                        0 ne index( compress(_name, 'AEIOUY') , trim(word2nd) ) then noname2=1;
                     if noname  > 0 or  noname2 = 1 then checkstr + 1;
                  end;
                  if noname  > 0 then output;
                end;

              if checkstr > 1 then do; /*** now check street address for possible words ***/
                do nss = 1 to &maxword;
                  nostre = 0;
                  noname = 0;
                  /*** try to match part of street ***/
                  word = scan( street,  nss);
                  if word eq ' ' then goto skipstre;
                  if (word ne ' ')            and
                     (length(trim(word)) > 1) and
                     0 ne index( _street, trim(word) ) then nostre=1;
                  if nostre  = 1 then output;

                  /*** if no hit, squeeze vowels and try again ***/
                  if nostre = 0 then do;
                    word = compress( word, 'AEIOUY');
                    if (word ne ' ')            and
                       (length(trim(word)) > 1) and
                       0 ne index( compress(_street, 'AEIOUY')  , trim(word) ) then nostre=1;
                    if nostre  = 1 then output;
                  end;
                  /*** do not bother with SoundEx function on street address ***/
                end;
               end;
              skipstre:;
              end;
            skiprecd:;
           end;
     stop; ;
    run;
    proc append base = matches  data=selrecds; run; /*** add found words to matches ***/
   %end;
%mend match;

%match;

proc sort data=matches out=matches;
 by recdno _recdno word;
run;

/***count number of words found for name and street and total words ***/
/***create pct words found and rank records ***/
data dups(drop=word i noname nostre checkstr noname2 word2nd likeword inn likeness
                     sdf lsdf sds lsds);
 length matchstr $ 200. ;
```

```
     retain matchstr;
     set matches;
     by recdno _recdno word;
     if first._recdno then matchstr = ' ';
     if first._recdno then do;
        cntname  = 0;
        cntstre  = 0;

     end;
     /*** build a character string of words used to find possible match ***/
     if first.word and word ne ' ' then matchstr = trim( matchstr ) || trim(word) || ',';
     cntname + noname ;
     cntstre + nostre ;
     if last._recdno then do; /*** keep only the last duplicate record of master record ***/

cntword = 0;
        do i = 1 to &maxword;
           if ' ' ne scan( name  , i ) then cntword + 1;
           if ' ' ne scan( street, i ) then cntword + 1;
        end;
        pctword = sum(of cntname cntstre) / cntword ; /*** calculate pct words ***/
        matchstr = left( substr(right(matchstr),1,200));
        output;
     end;
    run;

    proc sort data=dups out=merge.dups;
     by recdno descending pctword; /*** rank by descending PCTWORD ***/
    run;

    proc datasets library=work;
     delete srchfile selrecds matches dups; /*** delete unwanted datasets ***/
    quit;

    /*** Now create a simple report of records found as a possible match ***/
    proc printto file="c:\download\merge\dups.lis" new; run;
    options ls=72 nodate center number pageno=1;
    title1 "Transactions Records that potentially could match Master Records";
    title2 "1st 3 lines are from Transaction File";
    title3 "2nd 3 lines are from Master File";
    title4 "Last Line are words found that determined possible match";
    title5 "Words with * caught with soundex function";

    %emptyyn( merge.dups );
    %norecds; /*** Display Report if No Duplicate Records Exist ***/

    proc forms data=merge.dups width=120 lines=9 between=1 pagesize=60;
    line 1   recdno  name; /*** Transaction Record ***/
    line 2            street / indent=13;
    line 3            city  state  zipcode / pack indent=13;
    line 4  _recdno  _name ; /*** Master Record ***/
    line 5           _street / indent=13;
    line 6           _city _state _zipcode / pack indent=13;
    line 7           matchstr ;
    line 8 '_____________________________';
    run;

    proc printto; run;
    /*** end of program - fuzzy ***/



    /*****************************************************************/
                        /***     MEMBER = NORECDS.SAS   ***/

    %macro norecds;
         %if &numobs = 0 %then %do;
          data _null_;
           file print;
           put "Found No Records that could potentially be duplicated";
           put "Hence, there are no records to display at this time";
         run;
         %end;
    %mend norecds;


    /*****************************************************************/
```

```
          /***     MEMBER = READSTD.SAS   ***/


/*** readstd.sas ***/
   /*** read file with words that need to be eliminated ***/

    %macro readstd ( fnames );
         filename stdfiles "c:\download\merge\&fnames..csv"; run;
         data &fnames ;
          length orginal $ 200. convert $ 200. ;
          infile stdfiles delimiter=',' dsd missover;
          input  orginal convert ;
          orginal = upcase(orginal);
          convert = upcase(convert);
         run;
    %mend readstd;

    /*** end of program - readstd ***/



    /*************************************************************/
    /*** MACRO REMOVEDB.SAS                                    ***/
    /***                                                       ***/
    /***  Parameter STRING is the character string you wish to ***/
    /***  have doubles removed from.                           ***/
    /*************************************************************/

    %macro removedb( string );
     zzcnt = 0;
     do zzw = 1 to 100;/*** how many words are there in string ***/
        if ' ' ne scan( &string, zzw) then zzcnt + 1;
     end;
     zzw = 0;
     length zzwordo $ 200 zzword $ 200 ;
     do zzw = 1 to zzcnt; /*** loop for as many words as found ***/
      zzwordo = scan(&string,zzw); /*** keep original word ***/
      zzword  = zzwordo;           /*** new word to be modified ***/
          do zzl = 1 to length(trim(left(zzword)))-1; /*** loop length of word ***/
           /*** set first occurrence of double letter to blank ***/
           if substr(zzword ,zzl,1)=substr(zzword ,zzl+1,1) then
              substr(zzword ,zzl,1)=' ';
          end;
      zzword  = compress(zzword, ' '); /*** remove all internal blanks ***/
                /*** replace original word with new word ***/
                /*** and remove multiple blanks ***/
      &string = compbl (tranwrd( &string,trim(zzwordo),trim(zzword)));
     end;
      drop zzl zzw zzcnt zzwordo zzword; /*** drop these vars ***/
    %mend removedb;



    /*****************************************************************/
                      /***     MEMBER = SNDSLIKE.SAS     ***/


    %macro sndslike( first, second );
         sdf= soundex(&first);
        lsdf= substr(sdf,2);
         sds= soundex(&second);
        lsds= substr(sds,2);
          if length(trim(lsdf)) > 1 and length(trim(lsds)) > 1 then do;
           if ( 0 ne index(trim(lsdf), trim(lsds))) or
              ( 0 ne index(trim(lsds), trim(lsdf))) then noname + 1;
          end;
     %mend sndslike;
```

```
 /*****************************************************************/
 /***stdlist.sas                                                  ***/
 /***                                                             ***/
 /***  Author: Charles Patridge                                   ***/
 /***          PDPC, Ltd.                                         ***/
 /***          172 Monce Road                                     ***/
 /***          Burlington, CT 06013                               ***/
 /***          Home: 860-673-9278 or 860-675-9026                 ***/
 /***          Email: TVJB41A@prodigy.com                         ***/
 /***          Email: Charles_S_Patridge@prodigy.com              ***/
 /***                                                             ***/
 /***  Copyrighted 1984, 1989, 1993, 1994, 1996, 1998             ***/
 /***  With Author's permission, you may use this program         ***/
 /***  as agreed upon by Author.  It is illegal to distribute     ***/
 /***  this program for profit or misrepresent the original       ***/
 /***  owner/author of this Fuzzy Merge Routine(s).               ***/
 /***                                                             ***/
 /***  In order to make Fuzzy Match Process Work, standardization ***/
 /***  Names and Street Addresses is the critical key to success  ***/
 /***  Names are converted to shorten versions where possible     ***/
 /***  Addresses are shorten and common words like road, street   ***/
 /***   drive, etc. are eliminated from address field to reduce   ***/
 /***   the number of possible false hits.                        ***/
 /***                                                             ***/
 /***  When dealing with Company Names, standardization is also   ***/
 /***  critical.  Common Words for Company Names need to be re-   ***/
 /***  moved; such COMPANY, LIMITED, ASSOCIATES, etc              ***/
 /***  In addition, word endings such as S, ES, IES, ING, TION    ***/
 /***  should be dropped using the DROPCHAR routine.              ***/
 /***  Also, it might be useful to remove any double letters found***/
 /***  within a company name and use REMOVEDB routine to do this  ***/
 /***                                                             ***/
 /***  Also need to remove special characters and make every      ***/
 /***  word UPPER CASE to get exact matching made easier          ***/
 /***                                                             ***/
 /***  Suggest making each field longer than necessary in order to***/
 /***  actually split a word into two words,                      ***/
 /***  like HONGKONG set to HONG KONG to increase matching hits   ***/
 /*****************************************************************/
  libname  merge     "c:\download\merge" ; run;
  filename macros    "c:\download\merge" ; run;

  %include macros( 'emptyyn.sas' );
  %include macros( 'dropchar.sas' );
  %include macros( 'removedb.sas' );
  %include macros( 'readstd.sas' );

 /*** Read File ***/
%macro stdlist( fname );
  filename &fname    "c:\download\merge\&fname..csv"; run;
  DATA &fname;
  length name $200 street $200 city $200 state $200 zipcode $5 ;

  INFILE &fname missover delimiter=',' dsd ;
  input name street city state zipcode;
  name   = upcase(name  );
  street = upcase(street);
  city   = upcase(city  );
  *** state  = upcase(state );
  state = zipstate( zipcode ); /*** use zipcode to get State Abbreviation ***/
  run;

  /*** eliminate exact duplicate records of names using NODUPKEY ***/
  /*** no sense in processing matching process for 2 or more identical records ***/
  PROC SORT DATA=&fname OUT=&fname NODUPKEY; by name street zipcode; run;
```

```
data &fname(drop=i spechars);
      set &fname;
     spechars = ',!@#$%^&*()_+-={}[]:;"|\<>.?/~'' || "'";

     do i = 1 to length(spechars);
        name    = translate(name   , ' ' , substr(spechars,i) ); /*** make spec chars blank ***/
        street  = translate(street , ' ' , substr(spechars,i) ); /*** make spec chars blank ***/
     end;

     name      = compbl( name      );  /*** remove multiple internal blanks ***/
     street    = compbl( street    );  /*** remove multiple internal blanks ***/
     run;

    %emptyyn( &fname ); /*** how many rawdata records ***/
    %let notnxs = &numobs ; /*** set notnxs to number of records in file ***/

    %readstd( stdname ); /*** Read Standardized Name File ***/
    %emptyyn( stdname ); /*** How many Standardized Names are there ***/
    %let noname = &numobs ; /*** set noname to number of standardized name records ***/

    %readstd( stdaddr ); /*** Read Standardized Address File ***/
    %emptyyn( stdaddr ); /*** How many Standardized Addresses are there ***/
    %let noaddr = &numobs ; /*** set noaddr to number of standardized address records ***/

    /*** if dealing with company names, then add code to process standardized company names ***/
    /*** use stdbusn file to do this ***/

    %macro stdized;
                    /*** orginal will be made into what is in convert ***/
       data merge.&fname(drop=in is orginal convert );
        set &fname;
              in = 0;
           %do in = 1 %to &noname;
             recd = &in ;
             set stdname point=recd;
             name = ' ' || trim(name); /*** add a blank to front of word ***/
             name = left( tranwrd(name ,' ' || trim(orginal) || ' ' ,' '||trim(convert)||' ' ));
             name = compress( name    , '_' ); /*** _ means removes word ***/
             name = compbl ( name );  /*** remove multiple adjacent blanks ***/
           %end;

              is = 0;
           %do is = 1 %to &noaddr;
             recd = &is ;
             set stdaddr point=recd;
             street= ' ' || trim(street); /*** add a blank to front of word ***/
             street= left( tranwrd(street ,' ' || trim(orginal) || ' ' ,' '||trim(convert)||' ' ));
             street= compress( street    , '_' ); /*** _ means removes word ***/
             street= compbl( street ); /*** remove multiple adjacent blanks ***/
           %end;

         *** %removedb( name ); /*** remove double letters - may not want to do this***/

         *** %dropchar( name, S   ); /*** drop the following characters from the ending of words ***/

       run;


proc datasets library=work ; delete stdname stdaddr; quit; /*** delete unwanted datasets ***/
 %mend stdized;

    %stdized; /*** standardize file ***/

   %mend stdlist;


%stdlist( master );   /*** Standardize Master File ***/

   %stdlist( transact ); /*** Standardize Transaction File ***/
```

```
 /*****************************************************************/
                     /***     MEMBER = TOKENIZE.SAS   ***/


 /*** Tokenize.sas ***/
 OPTIONS LS=80  CENTER DATE NUMBER pagesize=60;
 filename datafile "c:\download\merge\transact.csv"; run;

DATA TOKEN  (KEEP=TOKEN);
 length string $ 200 name $ 200 street $ 200 cityst $ 200 zipcode $ 5;
 INFILE DATAFILE missover delimiter=';' dsd ;
 input  name street cityst zipcode ;
 string = compress( upcase(name), '.');

 DO I = 1 TO 100;
   specchar = " ,<.>/?:;'\|[{]}!@#$%^&*()_-+=~'" || '"';
   TOKEN = SCAN(STRING,I, ' '      );
   IF TOKEN = ' ' THEN GOTO FINISH;
   if length(trim(token)) < 2 then goto skip; /*** length < 2 ***/
   OUTPUT TOKEN;
   skip: ;
 END;
 FINISH: ;
RUN;

PROC SORT DATA=TOKEN OUT=TOKEN; BY TOKEN; RUN;

DATA TOKEN; SET TOKEN; BY TOKEN;
 IF FIRST.TOKEN AND LAST.TOKEN THEN DELETE;
RUN;

proc printto file = "tokenize.lis" new; run;

PROC FREQ DATA=TOKEN;  ***ORDER=FREQ;
TABLES TOKEN / MISSING out=freq noprint;
RUN;

proc print data=freq; run;

/*** end of program - tokenize ***/
```

Questions and comments should be directed to:

Charles S. Patridge
PDPC, Ltd.
172 Monce Road
Burlington, CT 06013
Home: 860-673-9278 or 860-675-9026
Email: Charles_S_Patridge@prodigy.com
Website: http://pages.prodigy.com/SASCONSIG
(URL is CaSe SenSiTive, type it exactly as shown)