SESUG Paper 74-2021

A Quick Look at Fuzzy Matching Programming Techniques Using SAS® Software

Stephen Sloan; Data Science Senior Principal; Accenture
Kirk Paul Lafler; SAS Consultant, Application Developer, Programmer, Educator and Author

ABSTRACT

Data comes in all forms, shapes, sizes and complexities. Stored in files and data sets, SAS® users across industries know all too well that data can be, and often is, problematic and plagued with a variety of issues. Two data files can be joined without a problem when they have identifiers with unique values. However, many files do not have unique identifiers, or "keys", and need to be joined by character values, like names or E-mail addresses. These identifiers might be spelled differently, or use different abbreviation or capitalization protocols. This paper illustrates data sets containing a sampling of data issues, popular data cleaning and user-defined validation techniques, data transformation techniques, traditional merge and join techniques, the introduction to the application of different SAS character-handling functions for phonetic matching, including SOUNDEX, SPEDIS, COMPLEV, and COMPGED, and an assortment of SAS programming techniques to resolve key identifier issues and to successfully merge, join and match less than perfect, or "messy" data. Although the programming techniques are illustrated using SAS code, many, if not most, of the techniques can be applied to any software platform that supports character-handling.

Keywords: Fuzzy matching, SAS, character-handling functions, phonetic matching, SOUNDEX, SPEDIS, edit distance, Levenshtein, COMPLEV, COMPGED

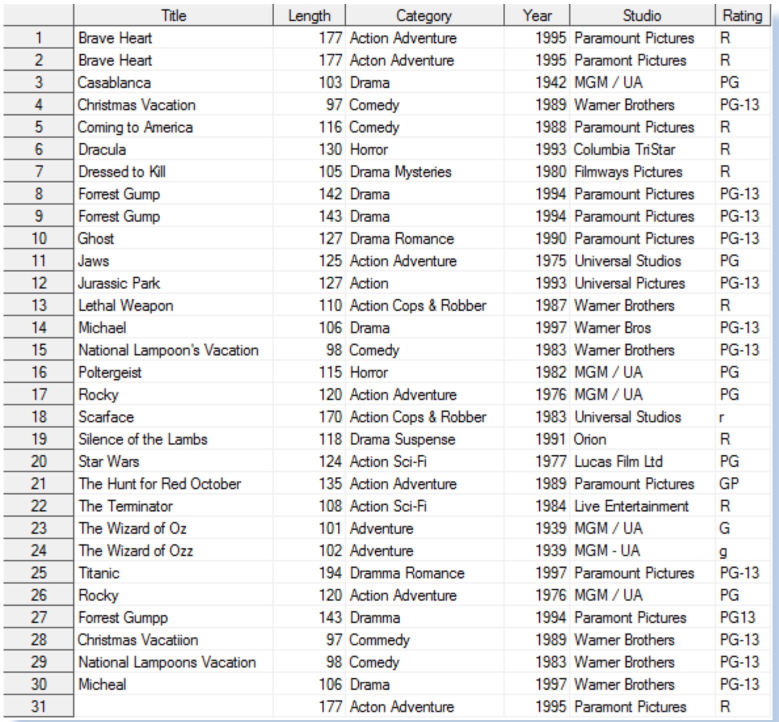
INTRODUCTION

When data sources and data sets contain consistent and valid data values, share common unique identifier(s), and have no missing data, the matching process rarely presents any problems. But, when data originating from multiple sources contain duplicate observations, duplicate and/or unreliable keys, missing values, invalid values, capitalization and punctuation issues, inconsistent matching variables, and imprecise text identifiers, the matching process is often compromised by unreliable and/or unpredictable results. When issues like these exist, SAS users must first clean and standardize any and all data irregularities before any attempts to match data records are performed. To assist in this time-consuming and costly process, users often utilize special-purpose programming techniques including the application of one or more SAS functions, the use of approximate string matching, and/or an assortment of constructive programming techniques to standardize and combine data sets together.

DATA SETS USED IN EXAMPLES

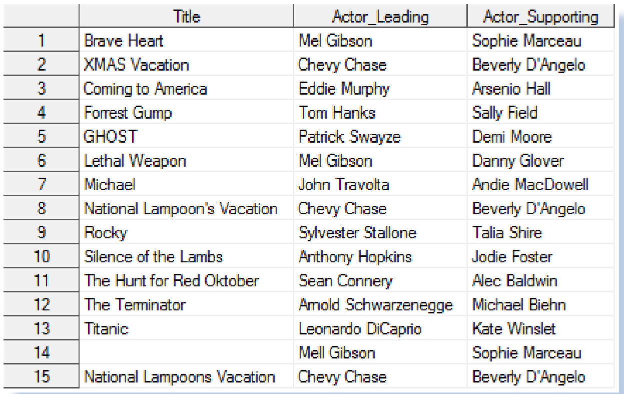
The examples presented in this paper illustrate two data sets, Movies_with_Messy_Data and Actors_with_Messy_Data. The Movies_with_Messy_Data data set, illustrated in Figure 1a, consists of 31 observations, a data structure of six variables where Title, Category, Studio, and Rating are defined as character variables; and Length and Year are defined as numeric variables. After careful inspection several data issues can be found in this data set including the existence of missing data, duplicate observations, spelling errors, punctuation inconsistencies, and invalid values.

The Actors_with_Messy_Data data set, illustrated in Figure 1b, contains 15 observations and a data structure consisting of three character variables: Title, Actor_Leading and Actor_Supporting. As with the Movies_with_Messy_Data data set, several data issues are found including missing data, spelling errors, punctuation inconsistencies, and invalid values.



	Title	Length	Category	Year	Studio	Rating
1	Brave Heart	177	Action Adventure	1995	Paramount Pictures	R
2	Brave Heart	177	Acton Adventure	1995	Paramont Pictures	R
3	Casablanca	103	Drama	1942	MGM / UA	PG
4	Christmas Vacation	97	Comedy	1989	Warner Brothers	PG-13
5	Coming to America	116	Comedy	1988	Paramount Pictures	R
6	Dracula	130	Horror	1993	Columbia TriStar	R
7	Dressed to Kill	105	Drama Mysteries	1980	Filmways Pictures	R
8	Forrest Gump	142	Drama	1994	Paramount Pictures	PG-13
9	Forrest Gump	143	Drama	1994	Paramount Pictures	PG-13
10	Ghost	127	Drama Romance	1990	Paramount Pictures	PG-13
11	Jaws	125	Action Adventure	1975	Universal Studios	PG
12	Jurassic Park	127	Action	1993	Universal Pictures	PG-13
13	Lethal Weapon	110	Action Cops & Robber	1987	Warner Brothers	R
14	Michael	106	Drama	1997	Warner Bros	PG-13
15	National Lampoon's Vacation	98	Comedy	1983	Warner Brothers	PG-13
16	Poltergeist	115	Horror	1982	MGM / UA	PG
17	Rocky	120	Action Adventure	1976	MGM / UA	PG
18	Scarface	170	Action Cops & Robber	1983	Universal Studios	r
19	Silence of the Lambs	118	Drama Suspense	1991	Orion	R
20	Star Wars	124	Action Sci-Fi	1977	Lucas Film Ltd	PG
21	The Hunt for Red October	135	Action Adventure	1989	Paramount Pictures	GP
22	The Terminator	108	Action Sci-Fi	1984	Live Entertainment	R
23	The Wizard of Oz	101	Adventure	1939	MGM / UA	G
24	The Wizard of Ozz	102	Adventure	1939	MGM - UA	g
25	Titanic	194	Dramma Romance	1997	Paramount Pictures	PG-13
26	Rocky	120	Action Adventure	1976	MGM / UA	PG
27	Forrest Gumpp	143	Dramma	1994	Paramont Pictures	PG13
28	Christmas Vacatiion	97	Commedy	1989	Warner Brothers	PG-13
29	National Lampoons Vacation	98	Comedy	1983	Warner Brothers	PG-13
30	Micheal	106	Drama	1997	Warner Brothers	PG-13
31		177	Acton Adventure	1995	Paramont Pictures	R

Figure 1a: Movies_with_Messy_Data data set.

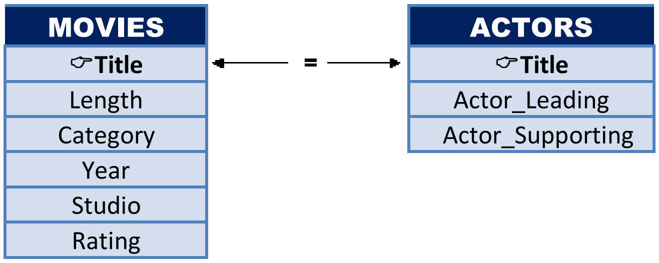


	Title	Actor_Leading	Actor_Supporting
1	Brave Heart	Mel Gibson	Sophie Marceau
2	XMAS Vacation	Chevy Chase	Beverly D'Angelo
3	Coming to America	Eddie Murphy	Arsenio Hall
4	Forrest Gump	Tom Hanks	Sally Field
5	GHOST	Patrick Swayze	Demi Moore
6	Lethal Weapon	Mel Gibson	Danny Glover
7	Michael	John Travolta	Andie MacDowell
8	National Lampoon's Vacation	Chevy Chase	Beverly D'Angelo
9	Rocky	Sylvester Stallone	Talia Shire
10	Silence of the Lambs	Anthony Hopkins	Jodie Foster
11	The Hunt for Red Oktober	Sean Connery	Alec Baldwin
12	The Terminator	Arnold Schwarzenegge	Michael Biehn
13	Titanic	Leonardo DiCaprio	Kate Winslet
14		Mell Gibson	Sophie Marceau
15	National Lampoons Vacation	Chevy Chase	Beverly D'Angelo

Figure 1b: Actors_with_Messy_Data data set.

THE MATCHING PROCESS EXPLAINED

In an age of endless spreadsheets, apps and relational database management systems (RDBMS), it's unusual to find a single sheet, file, table or data set that contains all the data needed to answer an organization's questions. Today's data exists in many forms and all too often involves matching two or more data sources to create a combined file. The matching process typically involves combining two or more data sets, spreadsheets and/or files possessing a shared, common and reliable, identifier (or key) to create a single data set, spreadsheet and/or file. The matching process, illustrated in the following diagram, shows two tables with a key, Title, to combine the two tables together.



But, when a shared and reliable key is associated with input data sources that are nonexistent, inexact, or unreliable, the matching process often becomes more involved and problematic. As cited in Sloan and Hoicowitz (2016), special processes are needed to successfully match the names and addresses from different files when they are similar, but not exactly the same. In a constructive and systematic way the authors of this paper describe a six step approach to cleansing data and performing fuzzy matching techniques.

SIX-STEP FUZZY MATCHING PROCESS

We adhere and recommend the following six step fuzzy matching process.

Step 1: Determine the Likely Matching Variables using Metadata.

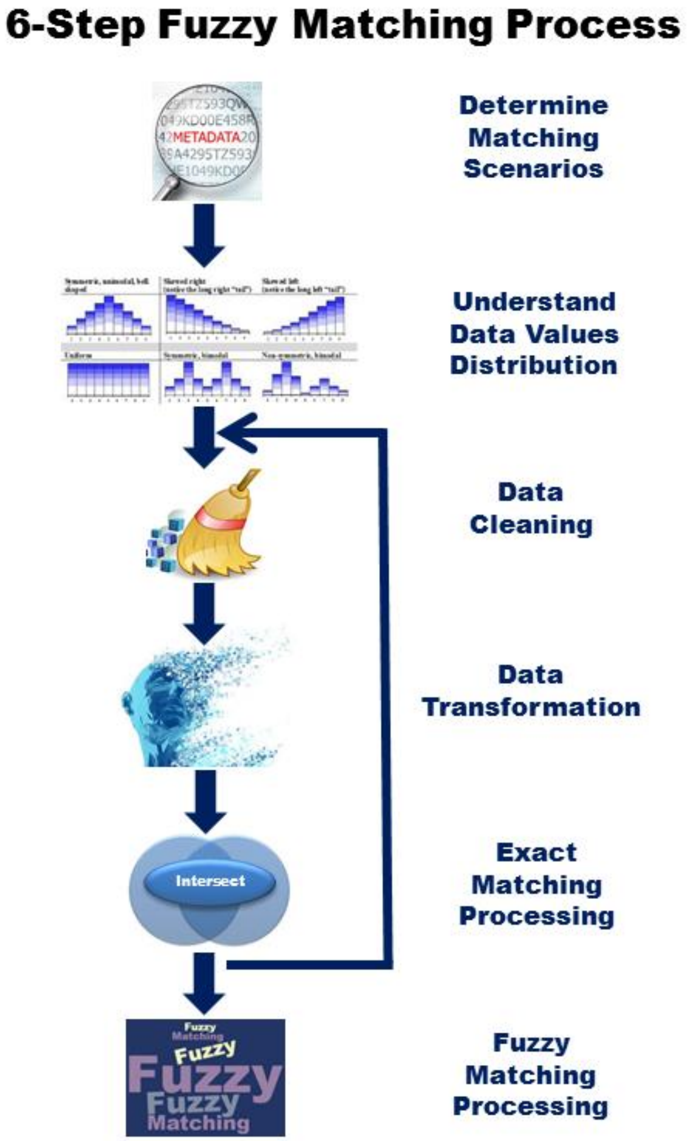
Step 2: Understand the Distribution of Data Values.

Step 3: Perform Data Cleaning.

Step 4: Perform Data Transformations.

Step 5: Process Exact Matches.

Step 6: Match Key Fields using Fuzzy Matching Techniques.


6-Step Fuzzy Matching Process
METADATA
Determine Matching Scenarios
Understand Data Values Distribution
Data Cleaning
Data Transformation
Intersect
Exact Matching Processing
Fuzzy Matching
Fuzzy Matching Processing

STEP #1: DETERMINE THE LIKELY MATCHING VARIABLES.

This first step determines whether any variables exist for matching purposes. Using the PROC CONTENTS listing, shown in Figure 2, we examine the results of the TITLE variable along with a sampling of values in greater detail to determine whether it can serve as the key for matching observations in both data sets, as well as the distribution of data values for other categorical variables.

PROC CONTENTS Code:

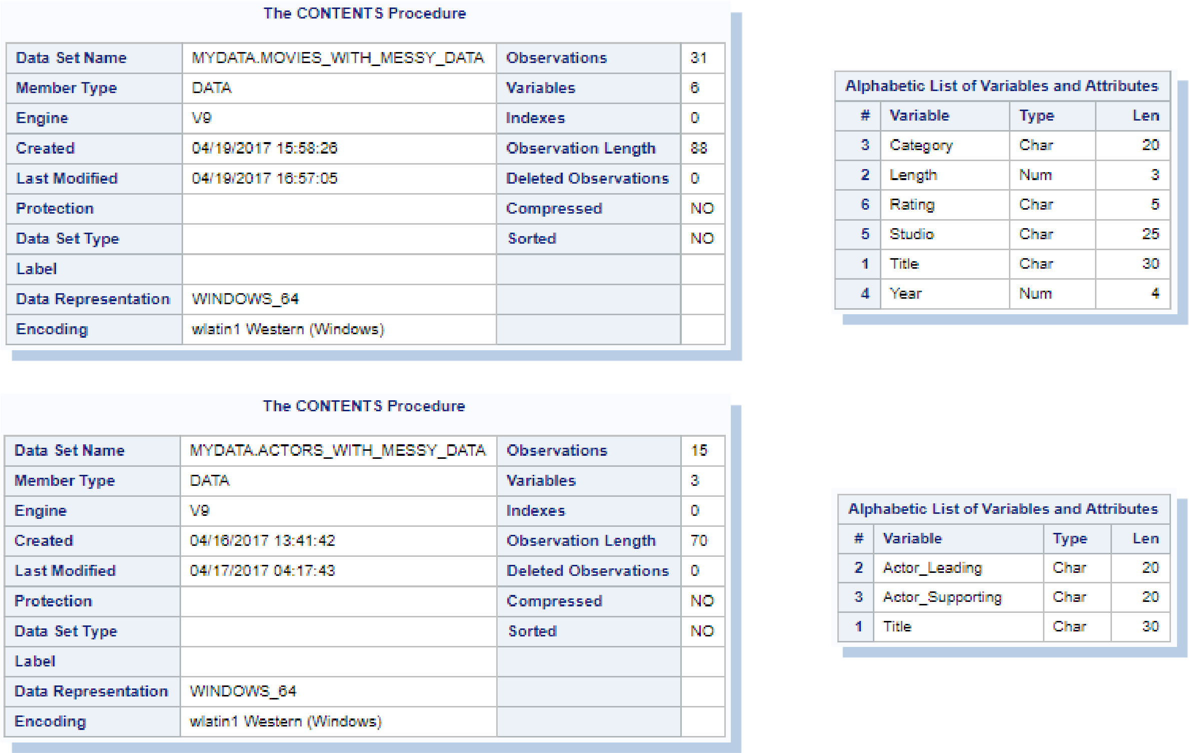
```
PROC CONTENTS DATA=mydata.Movies_with_Messy_Data ;
RUN ;
PROC CONTENTS DATA=mydata.Actors_with_Messy_Data ;
RUN ;
```

The Movies_with_Messy_Data data set's data structure consists of six variables where Title, Category, Studio, and Rating are defined as character variables; and Length and Year are defined as numeric variables. After careful inspection several data issues can be found in this data set including the existence of missing data, duplicate observations, spelling errors, punctuation inconsistencies, and invalid values.

The Actors_with_Messy_Data data set's data structure consists of three character variables: Title, Actor_Leading and Actor_Supporting. As with the Movies_with_Messy_Data data set, several data issues are found including missing data, spelling errors, punctuation inconsistencies, and invalid values.

The PROC CONTENTS output for both data sets are illustrated in Figure 2, below.

Results:



The CONTENTS Procedure

Data Set Name	MYDATA.MOVIES_WITH_MESSY_DATA	Observations	31
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	04/19/2017 15:58:26	Observation Length	88
Last Modified	04/19/2017 16:57:05	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Category	Char	20
2	Length	Num	3
6	Rating	Char	5
5	Studio	Char	25
1	Title	Char	30
4	Year	Num	4

The CONTENTS Procedure

Data Set Name	MYDATA.ACTORS_WITH_MESSY_DATA	Observations	15
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	04/16/2017 13:41:42	Observation Length	70
Last Modified	04/17/2017 04:17:43	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
2	Actor_Leading	Char	20
3	Actor_Supporting	Char	20
1	Title	Char	30

Figure 2: CONTENTS procedure Output for Movies_with_Messy_Data and Actors_with_Messy_Data data sets.

STEP #2: UNDERSTAND THE DISTRIBUTION OF DATA VALUES.

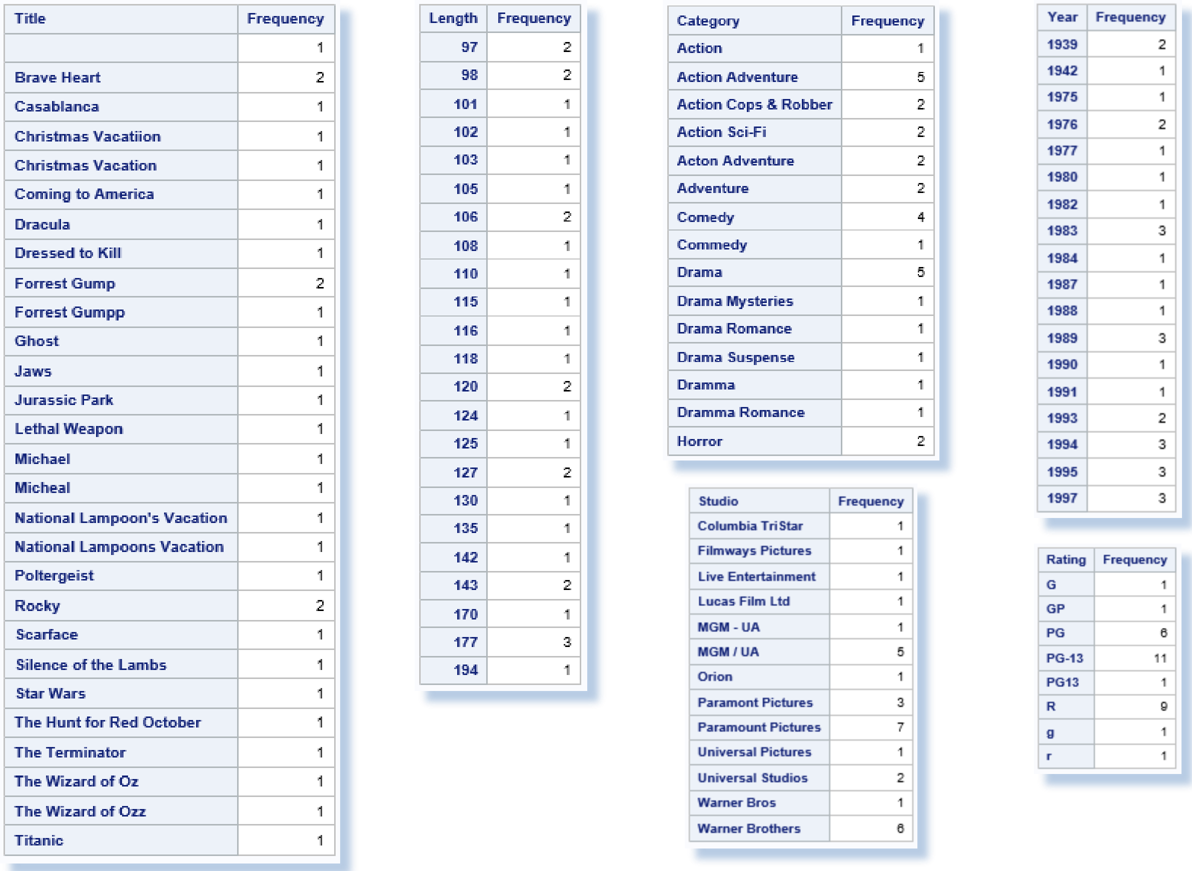
To derive a more accurate picture of the data sources, we suggest that users conduct extensive data analysis by identifying missing values, outliers, invalid values, minimum and maximum values, averages, value ranges, duplicate observations, distribution of values, and the number of distinct values a categorical variable contains. This important step provides an understanding of the data, while leveraging the data cleaning and standardizing activities that will be performed later. One of the first things data wranglers will want to do is explore the data using the SAS FREQ procedure, or an equivalent approach like Excel Pivot Tables.

PROC FREQ Code:

```
PROC FREQ DATA=mydata.Movies_with_Messy_Data ;
  TABLES _ALL_ / NOCUM NOPERCENT MISSING ;
RUN ;
```

Reviewing the results, we see an assortment of data issues including "key" values and/or record duplication, data accuracy, inconsistent values, missing values, validation, capitalization versus mixed case, and incomplete (partial) data issues, as shown in Figure 3.

Results:



Title	Frequency
	1
Brave Heart	2
Casablanca	1
Christmas Vacatiion	1
Christmas Vacation	1
Coming to America	1
Dracula	1
Dressed to Kill	1
Forrest Gump	2
Forrest Gumpp	1
Ghost	1
Jaws	1
Jurassic Park	1
Lethal Weapon	1
Michael	1
Micheal	1
National Lampoon's Vacation	1
National Lampoons Vacation	1
Poltergeist	1
Rocky	2
Scarface	1
Silence of the Lambs	1
Star Wars	1
The Hunt for Red October	1
The Terminator	1
The Wizard of Oz	1
The Wizard of Ozz	1
Titanic	1

Length	Frequency
97	2
98	2
101	1
102	1
103	1
105	1
106	2
108	1
110	1
115	1
116	1
118	1
120	2
124	1
125	1
127	2
130	1
135	1
142	1
143	2
170	1
177	3
194	1

Category	Frequency
Action	1
Action Adventure	5
Action Cops & Robber	2
Action Sci-Fi	2
Acton Adventure	2
Adventure	2
Comedy	4
Commedy	1
Drama	5
Drama Mysteries	1
Drama Romance	1
Drama Suspense	1
Dramma	1
Dramma Romance	1
Horror	2

Studio	Frequency
Columbia TriStar	1
Filmways Pictures	1
Live Entertainment	1
Lucas Film Ltd	1
MGM - UA	1
MGM / UA	5
Orion	1
Paramont Pictures	3
Paramount Pictures	7
Universal Pictures	1
Universal Studios	2
Warner Bros	1
Warner Brothers	6

Year	Frequency
1939	2
1942	1
1975	1
1976	2
1977	1
1980	1
1982	1
1983	3
1984	1
1987	1
1988	1
1989	3
1990	1
1991	1
1993	2
1994	3
1995	3
1997	3

Rating	Frequency
G	1
GP	1
PG	6
PG-13	11
PG13	1
R	9
g	1
r	1

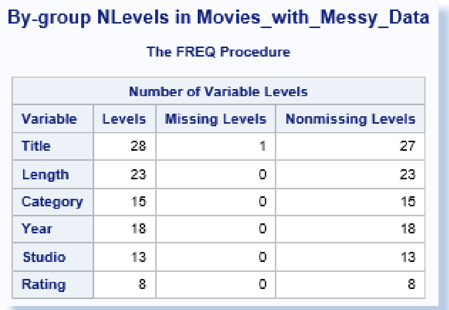
Figure 3: Distribution of Values from the FREQ Procedure.

Determining the number of distinct values a categorical variable has is critical to the fuzzy matching process. Acquiring this information helps everyone involved better understand the number of distinct variable levels, the unique values and the number of occurrences for developing data-driven programming constructs and elements. The following SAS code provides us with the number of By-group levels for each variable of interest we see in Figure 4.

PROC FREQ Code:

```
TITLE "By-group NLevels in Movies_with_Messy_Data" ;
PROC FREQ DATA=mydata.Movies_with_Messy_Data NLEVELS ;
RUN ;
```

Results:



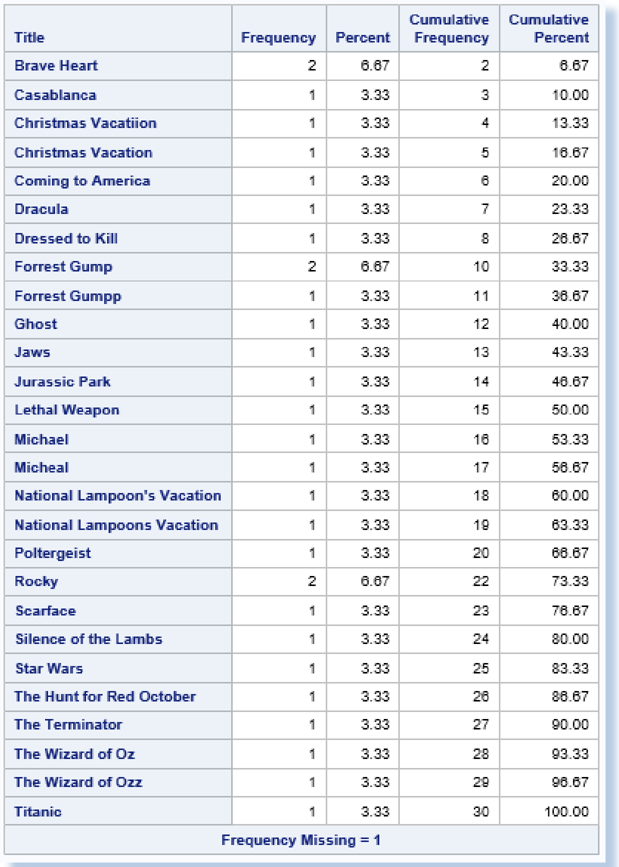
By-group NLevels in Movies_with_Messy_Data

The FREQ Procedure

Number of Variable Levels

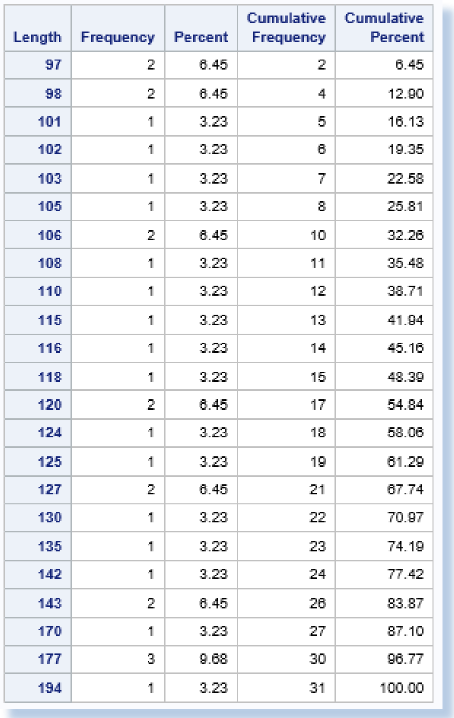
Variable	Levels	Missing Levels	Nonmissing Levels
Title	28	1	27
Length	23	0	23
Category	15	0	15
Year	18	0	18
Studio	13	0	13
Rating	8	0	8

Figure 4: The number of By-group levels for each variable of interest

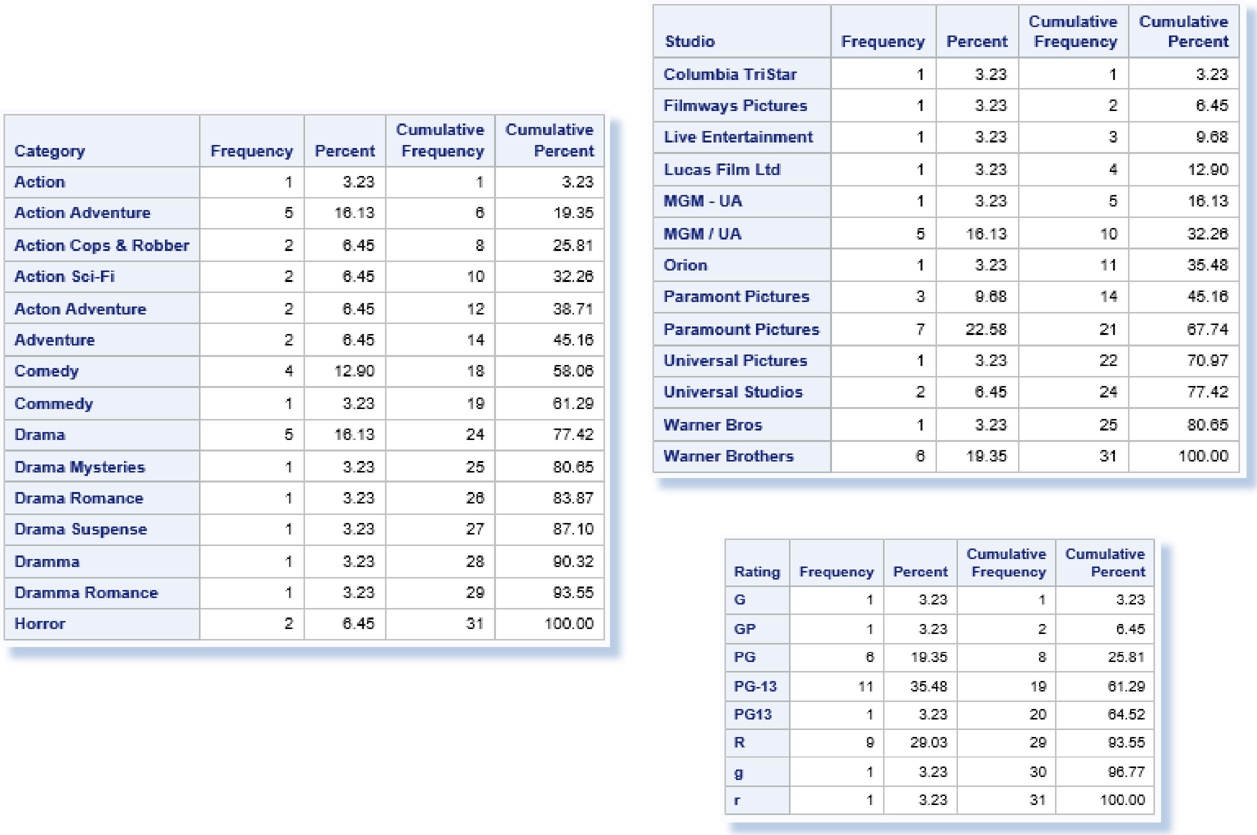


Title	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Brave Heart	2	6.67	2	6.67
Casablanca	1	3.33	3	10.00
Christmas Vacatiion	1	3.33	4	13.33
Christmas Vacation	1	3.33	5	16.67
Coming to America	1	3.33	6	20.00
Dracula	1	3.33	7	23.33
Dressed to Kill	1	3.33	8	26.67
Forrest Gump	2	6.67	10	33.33
Forrest Gumpp	1	3.33	11	36.67
Ghost	1	3.33	12	40.00
Jaws	1	3.33	13	43.33
Jurassic Park	1	3.33	14	46.67
Lethal Weapon	1	3.33	15	50.00
Michael	1	3.33	16	53.33
Micheal	1	3.33	17	56.67
National Lampoon's Vacation	1	3.33	18	60.00
National Lampoons Vacation	1	3.33	19	63.33
Poltergeist	1	3.33	20	66.67
Rocky	2	6.67	22	73.33
Scarface	1	3.33	23	76.67
Silence of the Lambs	1	3.33	24	80.00
Star Wars	1	3.33	25	83.33
The Hunt for Red October	1	3.33	26	86.67
The Terminator	1	3.33	27	90.00
The Wizard of Oz	1	3.33	28	93.33
The Wizard of Ozz	1	3.33	29	96.67
Titanic	1	3.33	30	100.00

Frequency Missing = 1



Length	Frequency	Percent	Cumulative Frequency	Cumulative Percent
97	2	6.45	2	6.45
98	2	6.45	4	12.90
101	1	3.23	5	16.13
102	1	3.23	6	19.35
103	1	3.23	7	22.58
105	1	3.23	8	25.81
106	2	6.45	10	32.26
108	1	3.23	11	35.48
110	1	3.23	12	38.71
115	1	3.23	13	41.94
116	1	3.23	14	45.16
118	1	3.23	15	48.39
120	2	6.45	17	54.84
124	1	3.23	18	58.06
125	1	3.23	19	61.29
127	2	6.45	21	67.74
130	1	3.23	22	70.97
135	1	3.23	23	74.19
142	1	3.23	24	77.42
143	2	6.45	26	83.87
170	1	3.23	27	87.10
177	3	9.68	30	96.77
194	1	3.23	31	100.00

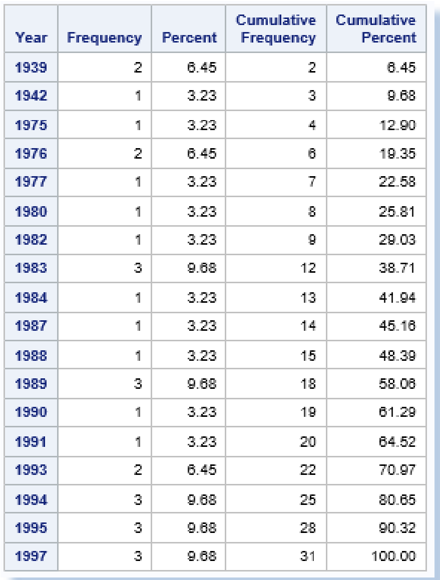


Category	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Action	1	3.23	1	3.23
Action Adventure	5	16.13	6	19.35
Action Cops & Robber	2	6.45	8	25.81
Action Sci-Fi	2	6.45	10	32.26
Acton Adventure	2	6.45	12	38.71
Adventure	2	6.45	14	45.16
Comedy	4	12.90	18	58.06
Commedy	1	3.23	19	61.29
Drama	5	16.13	24	77.42
Drama Mysteries	1	3.23	25	80.65
Drama Romance	1	3.23	26	83.87
Drama Suspense	1	3.23	27	87.10
Dramma	1	3.23	28	90.32
Dramma Romance	1	3.23	29	93.55
Horror	2	6.45	31	100.00

Studio	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Columbia TriStar	1	3.23	1	3.23
Filmways Pictures	1	3.23	2	6.45
Live Entertainment	1	3.23	3	9.68
Lucas Film Ltd	1	3.23	4	12.90
MGM - UA	1	3.23	5	16.13
MGM / UA	5	16.13	10	32.26
Orion	1	3.23	11	35.48
Paramont Pictures	3	9.68	14	45.16
Paramount Pictures	7	22.58	21	67.74
Universal Pictures	1	3.23	22	70.97
Universal Studios	2	6.45	24	77.42
Warner Bros	1	3.23	25	80.65
Warner Brothers	6	19.35	31	100.00

Rating	Frequency	Percent	Cumulative Frequency	Cumulative Percent
G	1	3.23	1	3.23
GP	1	3.23	2	6.45
PG	6	19.35	8	25.81
PG-13	11	35.48	19	61.29
PG13	1	3.23	20	64.52
R	9	29.03	29	93.55
g	1	3.23	30	96.77
r	1	3.23	31	100.00

Figure 4: The number of By-group levels for each variable of interest (continued)



Year	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1939	2	6.45	2	6.45
1942	1	3.23	3	9.68
1975	1	3.23	4	12.90
1976	2	6.45	6	19.35
1977	1	3.23	7	22.58
1980	1	3.23	8	25.81
1982	1	3.23	9	29.03
1983	3	9.68	12	38.71
1984	1	3.23	13	41.94
1987	1	3.23	14	45.16
1988	1	3.23	15	48.39
1989	3	9.68	18	58.06
1990	1	3.23	19	61.29
1991	1	3.23	20	64.52
1993	2	6.45	22	70.97
1994	3	9.68	25	80.65
1995	3	9.68	28	90.32
1997	3	9.68	31	100.00

Figure 4: The number of By-group levels for each variable of interest (continued)

STEP #3: PERFORM DATA CLEANING.

Data cleaning, referred to as data scrubbing, is the process of identifying and fixing data quality issues including missing values, invalid character and numeric values, outlier values, value ranges, duplicate observations, and other anomalies found in data sets. SAS provides many powerful ways to perform data cleaning tasks (Cody, 2017).

Use SAS Functions to Modify Data

SAS functions are an essential component of the SAS Base software. Representing a variety of built-in and callable routines, functions serve as the “work horses” in the SAS software providing users with “ready-to-use” tools designed to ease the burden of writing and testing often lengthy and complex code for a variety of programming tasks. The advantage of using SAS functions is evidenced by their relative ease of use, and their ability to provide a more efficient, robust and scalable approach to simplifying a process or programming task.

It is sometimes necessary to concatenate fields when matching files, because the fields could be concatenated in one file while separate in another. SAS functions span many functional categories, and this paper focuses on those that are integral to the fuzzy matching process. The following is a list of alternative methods of concatenating strings and/or variables together.

- Use the STRIP function to eliminate leading and trailing blanks, and then concatenate the stripped fields using the concatenation operator, and insert blanks between the stripped fields.
- Use one of the following CAT functions to concatenate fields:
  - ✓ CAT, the simplest of concatenation functions, joins two or more strings and/or variables together, end-to-end producing the same results as with the concatenation operator.
  - ✓ CATQ is similar to the CATX function, but the CATQ function adds quotation marks to any concatenated string or variable.
  - ✓ CATS removes leading and trailing blanks and concatenates two or more strings and/or variables together.
  - ✓ CATT removes trailing blanks and concatenates two or more strings and/or variables together.
  - ✓ CATX, perhaps the most robust CAT function, removes leading and trailing blanks and concatenates two or more strings and/or variables together with a delimiter between each.

Explore Data Issues with SAS’ PROC FORMAT

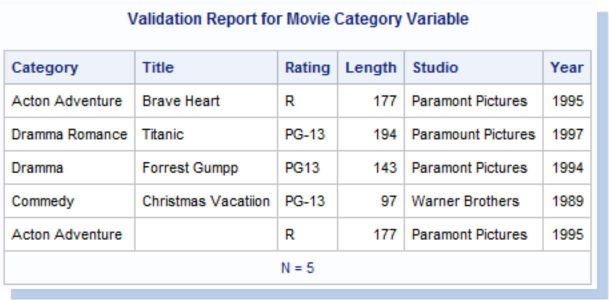
Problems with inaccurately entered data often necessitate time-consuming validation activities. A popular technique used by many to identify data issues is to use the FORMAT procedure. In the next example, a user-defined format is created with PROC FORMAT, a SAS DATA step identifies data issues associated with the Category variable, and a SAS PROC PRINT is specified to display the Category variable’s data issues, which are displayed in Figure 5.

PROC FORMAT, DATA Step and PROC PRINT Code:

```
PROC FORMAT LIBRARY=WORK ;
  VALUE $Category_Validation
    'Action'                = 'Action'
    'Action Adventure'      = 'Action Adventure'
    'Action Cops & Robber' = 'Action Cops & Robber'
    'Action Sci-Fi'         = 'Action Sci-Fi'
    'Adventure'             = 'Adventure'
    'Comedy'                = 'Comedy'
    'Drama'                 = 'Drama'
    'Drama Mysteries'       = 'Drama Mysteries'
    'Drama Romance'         = 'Drama Romance'
    'Drama Suspense'        = 'Drama Suspense'
    'Horror'                = 'Horror'
```

```
       Other                  = 'ERROR - Invalid Category'
    /* Other identified categories not listed */
    ;
  RUN ;

  DATA Validate_Category ;
    SET mydata.Movies_with_Messy_Data ;
    Check_Category = PUT(Category,$Category_Validation.) ;
    IF Check_Category = 'ERROR - Invalid Category' THEN
    DO ;
       OUTPUT ;
    END ;
  RUN ;

  PROC PRINT DATA=work.Validate_Category
             NOOBS N ;
    TITLE "Validation Report for Movie Category Variable" ;
    VAR Category Title Rating Length Studio Year ;
  RUN ;
```

Results:



Validation Report for Movie Category Variable

Category	Title	Rating	Length	Studio	Year
Acton Adventure	Brave Heart	R	177	Paramont Pictures	1995
Dramma Romance	Titanic	PG-13	194	Paramount Pictures	1997
Dramma	Forrest Gumpp	PG13	143	Paramont Pictures	1994
Commedy	Christmas Vacatiion	PG-13	97	Warner Brothers	1989
Acton Adventure		R	177	Paramont Pictures	1995

N = 5

Figure 5: Validation Report isolating Issues with the Movie Category Variable.

Once the invalid movie categories are identified with the validation report, users have the option of using one or more data cleaning techniques to manually correct, automating the process, or applying fuzzy matching techniques to correct (or handle) each invalid movie category.

Add Categories, if Available, to the Start of the Name

Doing this can eliminate matches that might occur if two businesses in the same general geographic area have the same name (for example: Smith's could describe a hardware store, a restaurant, or another type of business.) This is done in Figure 1, where Category is in the third column.

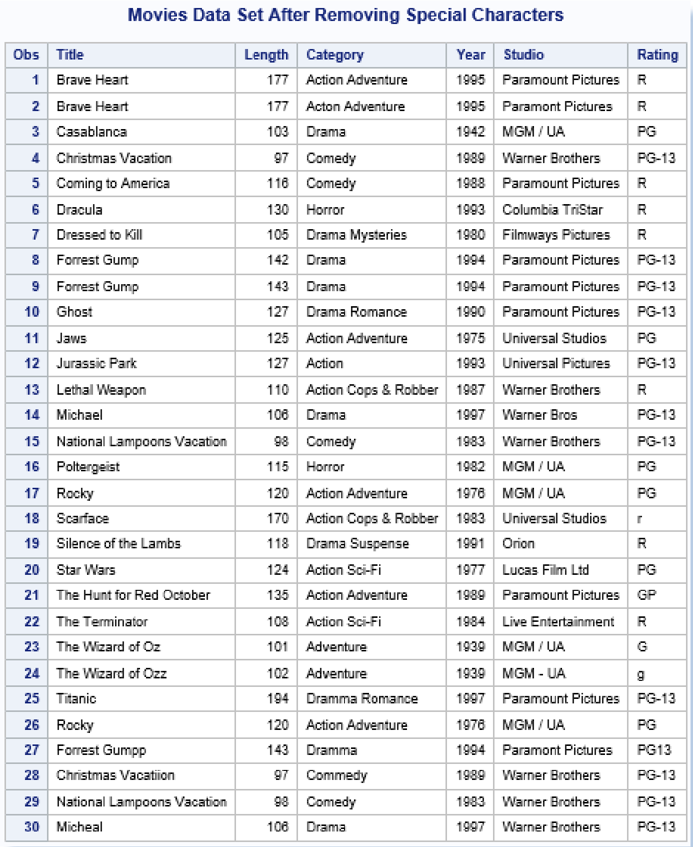
Remove Special or Extraneous Characters

Punctuation can differ even when names or titles are the same. Therefore, we remove the following characters: ‘ “ & ? – from the movie title. For example, “National Lampoon’s Vacation” and “National Lampoons Vacation” refer to the same movie title even though the former contains an apostrophe and the latter does not. Although the special characters can be removed in a number of ways, the next example shows their removal from the Title variable in both data sets using the COMPRESS function. The results are displayed in Figure 6.

Code to Remove Special Characters from Title and Perform Matching Process:

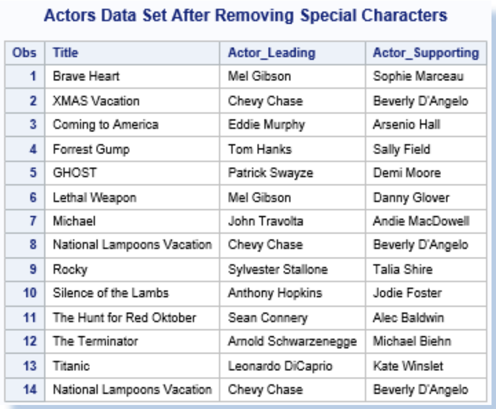
```
  data work.Movies_Cleaned ;
    set mydata.Movies_with_messy_data ;
    where title NE ‘’ ;
    title =
     compress(Title,"'""&?-") ;/*Remove special chars from Title*/
  run ;

  title "Movies Data Set After Removing Special Characters" ;
  proc print data=work.Movies_Cleaned ;
  run ;

  data work.Actors_Cleaned ;
    set mydata.Actors_with_messy_data ;
    where title NE ‘’ ;
    title =
     compress(Title,"'""&?-") ;/*Remove special chars from Title*/
  run ;

  title "Actors Data Set After Removing Special Characters" ;
  proc print data=work.Actors_Cleaned ;
  run ;

  proc sql ;
   title "Matched Rows from Movies and Actors" ;
   select DISTINCT M.Title, Rating, Length, Actor_Leading
    from work.Movies_Cleaned M,
         work.Actors_Cleaned A
     where M.Title = A.Title ;
  quit ;
```

Results:



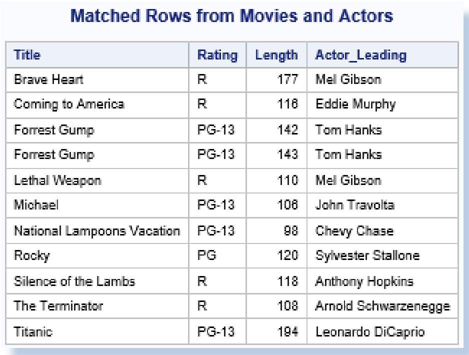
Movies Data Set After Removing Special Characters

Obs	Title	Length	Category	Year	Studio	Rating
1	Brave Heart	177	Action Adventure	1995	Paramount Pictures	R
2	Brave Heart	177	Acton Adventure	1995	Paramont Pictures	R
3	Casablanca	103	Drama	1942	MGM / UA	PG
4	Christmas Vacation	97	Comedy	1989	Warner Brothers	PG-13
5	Coming to America	116	Comedy	1988	Paramount Pictures	R
6	Dracula	130	Horror	1993	Columbia TriStar	R
7	Dressed to Kill	105	Drama Mysteries	1980	Filmways Pictures	R
8	Forrest Gump	142	Drama	1994	Paramount Pictures	PG-13
9	Forrest Gump	143	Drama	1994	Paramount Pictures	PG-13
10	Ghost	127	Drama Romance	1990	Paramount Pictures	PG-13
11	Jaws	125	Action Adventure	1975	Universal Studios	PG
12	Jurassic Park	127	Action	1993	Universal Pictures	PG-13
13	Lethal Weapon	110	Action Cops & Robber	1987	Warner Brothers	R
14	Michael	106	Drama	1997	Warner Bros	PG-13
15	National Lampoons Vacation	98	Comedy	1983	Warner Brothers	PG-13
16	Poltergeist	115	Horror	1982	MGM / UA	PG
17	Rocky	120	Action Adventure	1976	MGM / UA	PG
18	Scarface	170	Action Cops & Robber	1983	Universal Studios	r
19	Silence of the Lambs	118	Drama Suspense	1991	Orion	R
20	Star Wars	124	Action Sci-Fi	1977	Lucas Film Ltd	PG
21	The Hunt for Red October	135	Action Adventure	1989	Paramount Pictures	GP
22	The Terminator	108	Action Sci-Fi	1984	Live Entertainment	R
23	The Wizard of Oz	101	Adventure	1939	MGM / UA	G
24	The Wizard of Ozz	102	Adventure	1939	MGM - UA	g
25	Titanic	194	Dramma Romance	1997	Paramount Pictures	PG-13
26	Rocky	120	Action Adventure	1976	MGM / UA	PG
27	Forrest Gumpp	143	Dramma	1994	Paramont Pictures	PG13
28	Christmas Vacatiion	97	Commedy	1989	Warner Brothers	PG-13
29	National Lampoons Vacation	98	Comedy	1983	Warner Brothers	PG-13
30	Micheal	106	Drama	1997	Warner Brothers	PG-13



Actors Data Set After Removing Special Characters

Obs	Title	Actor_Leading	Actor_Supporting
1	Brave Heart	Mel Gibson	Sophie Marceau
2	XMAS Vacation	Chevy Chase	Beverly D'Angelo
3	Coming to America	Eddie Murphy	Arsenio Hall
4	Forrest Gump	Tom Hanks	Sally Field
5	GHOST	Patrick Swayze	Demi Moore
6	Lethal Weapon	Mel Gibson	Danny Glover
7	Michael	John Travolta	Andie MacDowell
8	National Lampoons Vacation	Chevy Chase	Beverly D'Angelo
9	Rocky	Sylvester Stallone	Talia Shire
10	Silence of the Lambs	Anthony Hopkins	Jodie Foster
11	The Hunt for Red Oktober	Sean Connery	Alec Baldwin
12	The Terminator	Arnold Schwarzenegge	Michael Biehn
13	Titanic	Leonardo DiCaprio	Kate Winslet
14	National Lampoons Vacation	Chevy Chase	Beverly D'Angelo



Matched Rows from Movies and Actors

Title	Rating	Length	Actor_Leading
Brave Heart	R	177	Mel Gibson
Coming to America	R	116	Eddie Murphy
Forrest Gump	PG-13	142	Tom Hanks
Forrest Gump	PG-13	143	Tom Hanks
Lethal Weapon	R	110	Mel Gibson
Michael	PG-13	106	John Travolta
National Lampoons Vacation	PG-13	98	Chevy Chase
Rocky	PG	120	Sylvester Stallone
Silence of the Lambs	R	118	Anthony Hopkins
The Terminator	R	108	Arnold Schwarzenegge
Titanic	PG-13	194	Leonardo DiCaprio

Figure 6: After the Removal of Special Characters and the Results from an Inner Join.

Put All Characters in Upper-case Notation and Remove Leading Blanks

Different data bases could have different standards for capitalization, and some character strings can be copied in with leading blanks. As found in our example data sets the value contained in the Title variable can be stored as all lower-case, upper-case, or in mixed-case which can impact the success of traditional merge and join matching techniques. Consequently, to remedy the issue associated with case and leading blanks, we recommend using the STRIP function to remove leading and trailing blanks along with the UPCASE function to convert all Title values to uppercase characters. For users of other popular programming languages, there is generally an equivalent function, or method, available to handle these types of issues.

Remove Words that might or might not Appear in Key Fields

Commonly used words in language, referred to as stop words, are frequently ignored by many search and retrieval processes. Stop words are classified as irrelevant and, as a result, are inserted into stop lists and are ignored. Examples include The, .com, Inc, LTD, LLC, DIVISION, CORP, CORPORATION, CO., and COMPANY. Some data base tables might include these, while others might not.

Choose a Standard for Addresses

Address fields can present a challenge when analyzing and processing data sources. To help alleviate comparison issues, decide whether to use Avenue or Ave, Road or Rd, Street or St, etc, and then convert the address fields accordingly or create a user-defined lookup process using PROC FORMAT to match the standard values.

Rationalize Zip Codes when Matching Addresses, Use Geocodes when Available

We found it useful to remove the last 4 digits of 9-digit zip codes, because some files might only have 5-digit zip codes. Since some files might have zip codes as numeric fields, and other files might have zip codes as character fields, make sure to include leading zeroes. For example, zip codes with a leading zero, as in 08514, would appear in a numeric field as 8514 requiring the leading zero to be inserted along with the specification of a Z5. informat and format being assigned to the zip code variable.

If working with US zip codes, make sure they are all numeric. This may not apply for other countries. One common mistake to watch for is that sometimes Canada, with abbreviation CA, is put in as the state CA (California) instead of the country CA. Since Canada has an alphanumeric 6-character zip code, this, hopefully, will be caught when checking for numeric zip codes.

If the program has access to geocodes, or if they are in the input data bases, geocodes can provide a further level of validation in addition to the zip codes.

Specify the DUPOUT=, NODUPRECS, or NODUPKEYS Options

A popular and frequently used procedure, PROC SORT, identifies and removes duplicate observations from a data set. By specifying one or more of the SORT procedure's three options: **DUPOUT=**, **NODUPRECS**, and **NODUPKEYS**, users are able to control how duplicate observations are identified and removed.

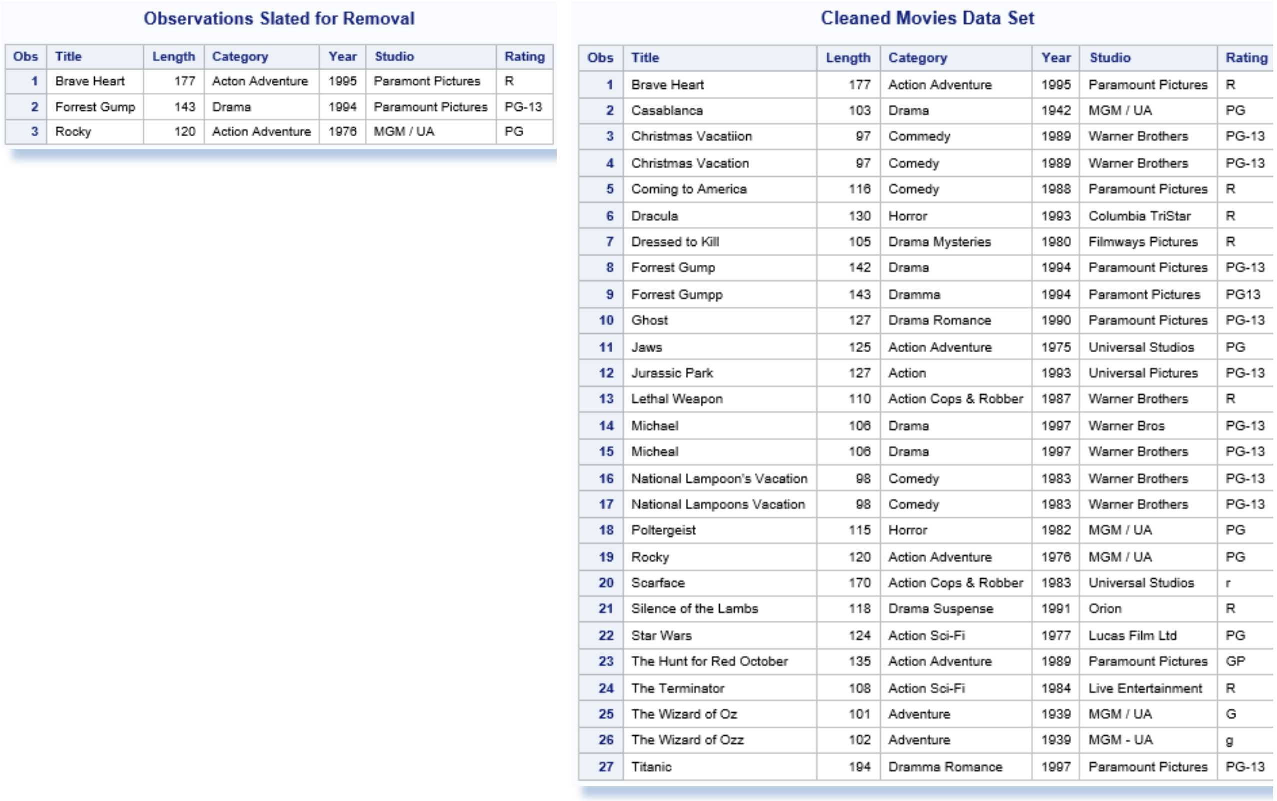
PROC SORT's DUPOUT= option is often used to identify duplicate observations before removing them from a data set. A DUPOUT= option, often specified when a data set is too large for visual inspection, can be used with the NODUPKEYS or NODUPRECS options to name a data set that contains duplicate keys or entire observations. In the next example, the DUPOUT=, OUT= and NODUPKEY options are specified to identify duplicate keys. The NODUPKEY option removes observations that have the same key values, so that only one remains in the output data set. The PROC SORT is followed by the PROC PRINT procedure so that the results can be examined.

PROC SORT and PROC PRINT Code:

```
PROC SORT DATA=mydata.Movies_with_Messy_Data
        DUPOUT=work.Movies_Dupout_NoDupkey
          OUT=work.Movies_Sorted_Cleaned_NoDupkey
        NODUPKEY ;
  BY Title ;
  WHERE Title NE “” ;
RUN ;

PROC PRINT DATA=work.Movies_Dupout_NoDupkey ;
  TITLE “Observations Slated for Removal” ;
RUN ;

PROC PRINT DATA=work.Movies_Sorted_Cleaned_NoDupkey ;
  TITLE “Cleaned Movies Data Set” ;
RUN ;
```

The results of the above SAS code are shown in Figure 7. The NODUPKEY option retains only one observation from any group of observations with duplicate keys. When Observations with identical key values are not adjacent to each other, users may first need to specify the NODUPKEY or NODUPKEYS option and sort the data set by all the variables (BY _ALL_ ;) to ensure the observations are in the correct order to remove all duplicates (SAS Usage Note 1566, 2000; Lafler, 2017).

Results:



Observations Slated for Removal

Obs	Title	Length	Category	Year	Studio	Rating
1	Brave Heart	177	Acton Adventure	1995	Paramont Pictures	R
2	Forrest Gump	143	Drama	1994	Paramount Pictures	PG-13
3	Rocky	120	Action Adventure	1976	MGM / UA	PG

Cleaned Movies Data Set

Obs	Title	Length	Category	Year	Studio	Rating
1	Brave Heart	177	Action Adventure	1995	Paramount Pictures	R
2	Casablanca	103	Drama	1942	MGM / UA	PG
3	Christmas Vacatiion	97	Commedy	1989	Warner Brothers	PG-13
4	Christmas Vacation	97	Comedy	1989	Warner Brothers	PG-13
5	Coming to America	116	Comedy	1988	Paramount Pictures	R
6	Dracula	130	Horror	1993	Columbia TriStar	R
7	Dressed to Kill	105	Drama Mysteries	1980	Filmways Pictures	R
8	Forrest Gump	142	Drama	1994	Paramount Pictures	PG-13
9	Forrest Gumpp	143	Dramma	1994	Paramont Pictures	PG13
10	Ghost	127	Drama Romance	1990	Paramount Pictures	PG-13
11	Jaws	125	Action Adventure	1975	Universal Studios	PG
12	Jurassic Park	127	Action	1993	Universal Pictures	PG-13
13	Lethal Weapon	110	Action Cops & Robber	1987	Warner Brothers	R
14	Michael	106	Drama	1997	Warner Bros	PG-13
15	Micheal	106	Drama	1997	Warner Brothers	PG-13
16	National Lampoon's Vacation	98	Comedy	1983	Warner Brothers	PG-13
17	National Lampoons Vacation	98	Comedy	1983	Warner Brothers	PG-13
18	Poltergeist	115	Horror	1982	MGM / UA	PG
19	Rocky	120	Action Adventure	1976	MGM / UA	PG
20	Scarface	170	Action Cops & Robber	1983	Universal Studios	r
21	Silence of the Lambs	118	Drama Suspense	1991	Orion	R
22	Star Wars	124	Action Sci-Fi	1977	Lucas Film Ltd	PG
23	The Hunt for Red October	135	Action Adventure	1989	Paramount Pictures	GP
24	The Terminator	108	Action Sci-Fi	1984	Live Entertainment	R
25	The Wizard of Oz	101	Adventure	1939	MGM / UA	G
26	The Wizard of Ozz	102	Adventure	1939	MGM - UA	g
27	Titanic	194	Dramma Romance	1997	Paramount Pictures	PG-13

Figure 7: Observations Slated for Removal and the Cleaned Movies Data Set.

Although the removal of duplicates using PROC SORT is a popular technique among many SAS users, an element of care should be given to using this method when processing large data sets. Since sort operations can often be CPU-intensive, the authors of this paper recommend comparing PROC SORT to procedures like SAS PROC SQL with the SELECT DISTINCT keyword and/or SAS PROC SUMMARY with the CLASS statement to determine the performance impact of one method versus another.

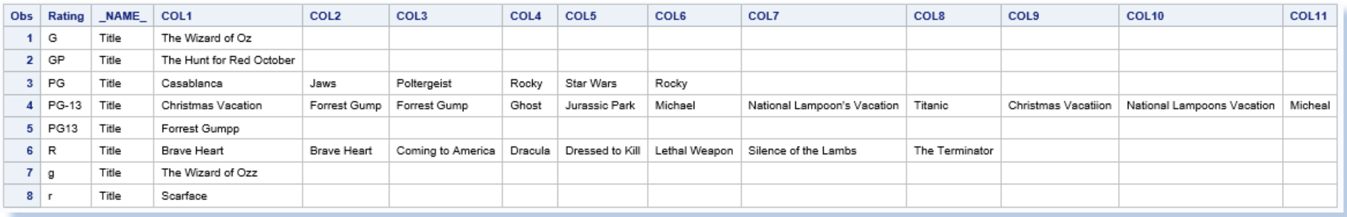
STEP #4: PERFORM DATA TRANSFORMATIONS.

Data transformations can be required to compare files. Data set structures sometimes need to be converted from wide to long or long to wide and files may need to be reconciled by having their variables grouped in different ways. When a data set's structure and data is transformed, we typically recommend that a new data set be created from the original one. SAS' PROC TRANSPOSE is handy for restructuring data in a data set, and is typically used in preparation for special types of processing like array processing. In its simplest form, data can be transformed with or without grouping. In the next example, the Movies data set is first sorted in ascending order by the variable RATING then the sorted data set is transposed using the RATING variable as the by-group variable. The result is shown in Figure 8, and it gives all of the titles within each rating.

PROC TRANSPOSE Code:

```
PROC SORT DATA=mydata.Movies_with_Messy_Data
           OUT=work.Movies_Sorted ;
  BY Rating ; /* BY-Group to Transpose */
  WHERE Title NE “” ;
RUN ;

PROC TRANSPOSE DATA=work.Movies_Sorted
           OUT=work.Movies_Transposed ;
  VAR Title ; /* Variable to Transpose */
  BY Rating ; /* BY-Group to Transpose */
RUN ;

PROC PRINT DATA=work.Movies_Transposed ;
RUN ;
```

Results:



Obs	Rating	_NAME_	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9	COL10	COL11
1	G	Title	The Wizard of Oz										
2	GP	Title	The Hunt for Red October										
3	PG	Title	Casablanca	Jaws	Poltergeist	Rocky	Star Wars	Rocky					
4	PG-13	Title	Christmas Vacation	Forrest Gump	Forrest Gump	Ghost	Jurassic Park	Michael	National Lampoon's Vacation	Titanic	Christmas Vacatiion	National Lampoons Vacation	Micheal
5	PG13	Title	Forrest Gumpp										
6	R	Title	Brave Heart	Brave Heart	Coming to America	Dracula	Dressed to Kill	Lethal Weapon	Silence of the Lambs	The Terminator			
7	g	Title	The Wizard of Ozz										
8	r	Title	Scarface										

Figure 8: Results from Performing a Data Transform with the TRANSPOSE Procedure.

STEP 5: PROCESS EXACT MATCHES.

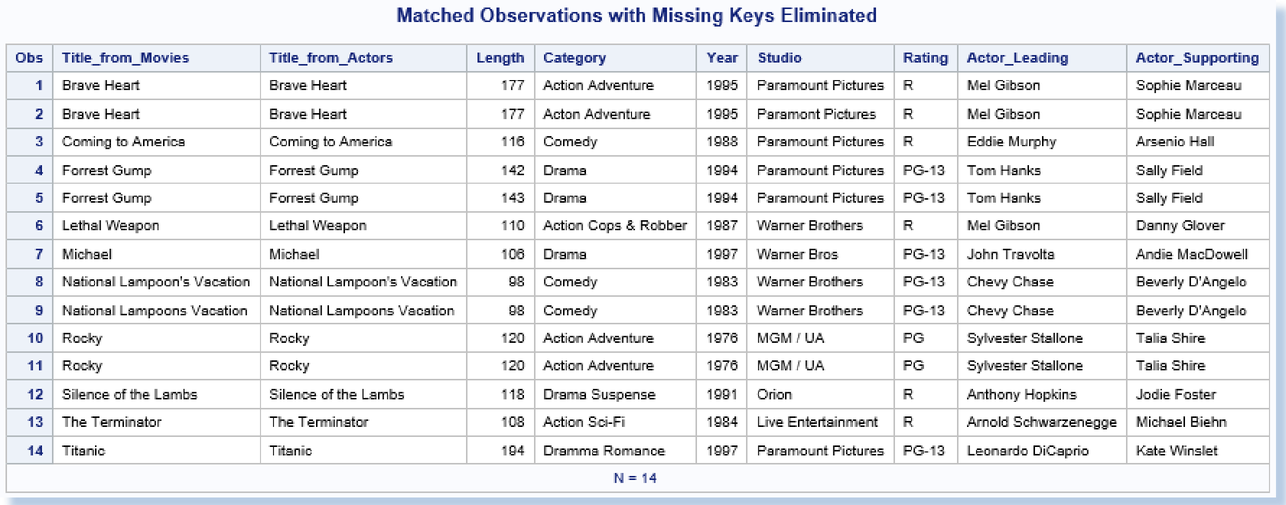
Since we are trying to match entries that do not have an exact match, we can save processing time by immediately eliminating the observations (or rows) with missing key information. This can be accomplished in a number of ways, including constructing IF-THEN/ELSE or WHERE logic to bypass processing observations with missing movie titles.

Another approach to bypass processing observations with missing movie titles could be to use the NODUP or NODUPKEY parameter with SAS' PROC SORT (more detail on these options will be presented later). Once missing observations with missing keys are eliminated, the focus can then be turned to processing observations that have exact matches on name, address, and as with our example data sets, the Title variable, as shown in Figure 9. We also process and retain the observations that have mismatches on the Title variable, as shown in Figure 10; the observations that did not have exact matches on the Title variable from the Movies data set, as shown in Figure 11; and the observations that did not have exact matches on the Title variable from the Actors data set, as shown in Figure 12.

PROC SORT, DATA Step and PROC PRINT Code:

```
proc sort data=mydata.Actors_with_messy_data
          out=work.Actors_Sorted ;
 where Title NE "" ;
 by Title ;
run ;

proc sort data=mydata.Movies_with_messy_data
          out=work.Movies_Sorted ;
 where Title NE "" ;
 by Title ;
run ;

data work.Matches(DROP=Title)
     work.MisMatches(DROP=Title)
     work.Movies_with_Unmatched_Obs(KEEP=Title Length Category
                                    Year Studio Rating)
     work.Actors_with_Unmatched_Obs(KEEP=Title Actor_Leading
                                    Actor_Supporting) ;
 merge work.Movies_Sorted (IN=M)
       work.Actors_Sorted (IN=A) ;
 by Title ;
 if M then Title_from_Movies = Title ;
 if A then Title_from_Actors = Title ;
 if M and A then output work.Matches ;
 else if NOT M or NOT A then output work.MisMatches ;
 if M and NOT A then output work.Movies_with_Unmatched_Obs ;
 else if A and NOT M then
                     output work.Actors_with_Unmatched_Obs ;
run ;

proc print data=work.Matches N ;
 title "Matched Observations with Missing Keys Eliminated" ;
 var Title_from_Movies Title_from_Actors Length Category Year
     Studio Rating Actor_Leading Actor_Supporting ;
run ;

proc print data=work.MisMatches N ;
 title "MisMatched Observations with Missing Keys Eliminated" ;
 var Title_from_Movies Title_from_Actors Length Category Year
     Studio Rating Actor_Leading Actor_Supporting ;
run ;

proc print data=work.Movies_with_Unmatched_Obs N ;
 title "Movies with UnMatched Observations" ;
 var Title Length Category Year Studio Rating ;
run ;

proc print data=work.Actors_with_Unmatched_Obs N ;
 title "Actors with UnMatched Observations" ;
 var Title Actor_Leading Actor_Supporting ;
run ;
```

Results:

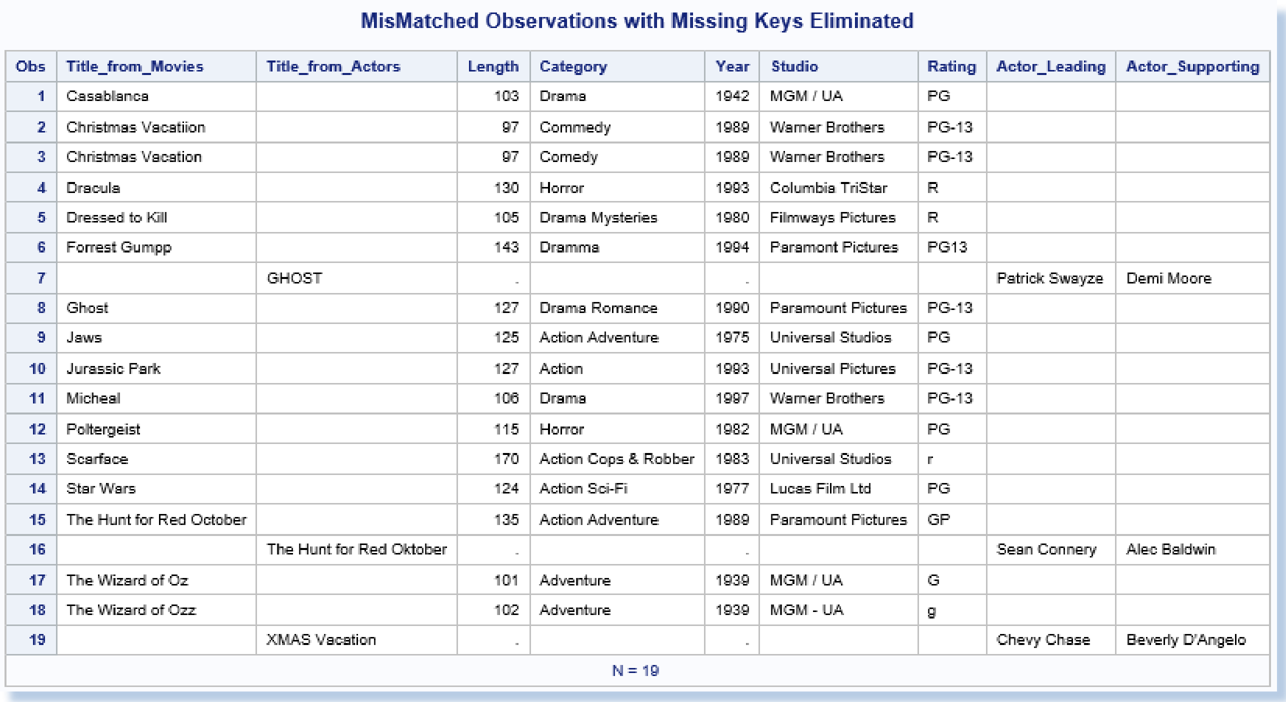


Matched Observations with Missing Keys Eliminated

Obs	Title_from_Movies	Title_from_Actors	Length	Category	Year	Studio	Rating	Actor_Leading	Actor_Supporting
1	Brave Heart	Brave Heart	177	Action Adventure	1995	Paramount Pictures	R	Mel Gibson	Sophie Marceau
2	Brave Heart	Brave Heart	177	Acton Adventure	1995	Paramont Pictures	R	Mel Gibson	Sophie Marceau
3	Coming to America	Coming to America	116	Comedy	1988	Paramount Pictures	R	Eddie Murphy	Arsenio Hall
4	Forrest Gump	Forrest Gump	142	Drama	1994	Paramount Pictures	PG-13	Tom Hanks	Sally Field
5	Forrest Gump	Forrest Gump	143	Drama	1994	Paramount Pictures	PG-13	Tom Hanks	Sally Field
6	Lethal Weapon	Lethal Weapon	110	Action Cops & Robber	1987	Warner Brothers	R	Mel Gibson	Danny Glover
7	Michael	Michael	106	Drama	1997	Warner Bros	PG-13	John Travolta	Andie MacDowell
8	National Lampoon's Vacation	National Lampoon's Vacation	98	Comedy	1983	Warner Brothers	PG-13	Chevy Chase	Beverly D'Angelo
9	National Lampoons Vacation	National Lampoons Vacation	98	Comedy	1983	Warner Brothers	PG-13	Chevy Chase	Beverly D'Angelo
10	Rocky	Rocky	120	Action Adventure	1976	MGM / UA	PG	Sylvester Stallone	Talia Shire
11	Rocky	Rocky	120	Action Adventure	1976	MGM / UA	PG	Sylvester Stallone	Talia Shire
12	Silence of the Lambs	Silence of the Lambs	118	Drama Suspense	1991	Orion	R	Anthony Hopkins	Jodie Foster
13	The Terminator	The Terminator	108	Action Sci-Fi	1984	Live Entertainment	R	Arnold Schwarzenegge	Michael Biehn
14	Titanic	Titanic	194	Dramma Romance	1997	Paramount Pictures	PG-13	Leonardo DiCaprio	Kate Winslet

N = 14

Figure 9: Matched Observations.

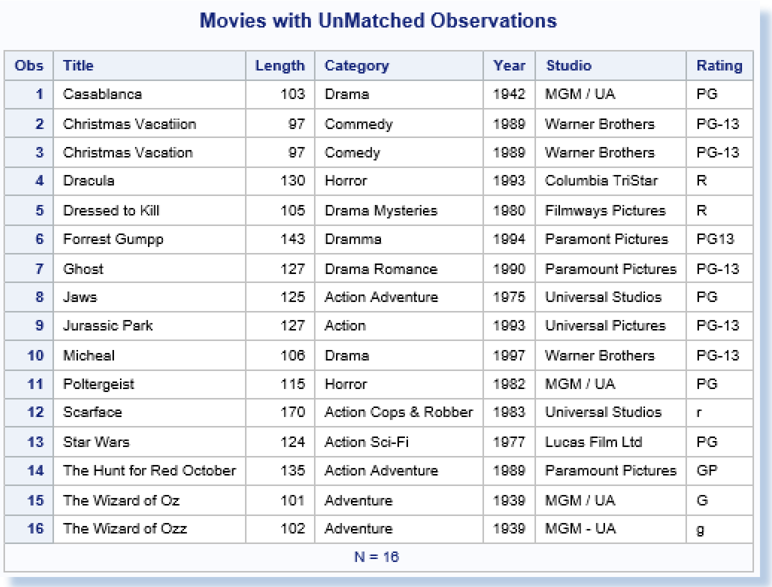


MisMatched Observations with Missing Keys Eliminated

Obs	Title_from_Movies	Title_from_Actors	Length	Category	Year	Studio	Rating	Actor_Leading	Actor_Supporting
1	Casablanca		103	Drama	1942	MGM / UA	PG		
2	Christmas Vacatiion		97	Commedy	1989	Warner Brothers	PG-13		
3	Christmas Vacation		97	Comedy	1989	Warner Brothers	PG-13		
4	Dracula		130	Horror	1993	Columbia TriStar	R		
5	Dressed to Kill		105	Drama Mysteries	1980	Filmways Pictures	R		
6	Forrest Gumpp		143	Dramma	1994	Paramont Pictures	PG13		
7		GHOST	.		.			Patrick Swayze	Demi Moore
8	Ghost		127	Drama Romance	1990	Paramount Pictures	PG-13		
9	Jaws		125	Action Adventure	1975	Universal Studios	PG		
10	Jurassic Park		127	Action	1993	Universal Pictures	PG-13		
11	Micheal		106	Drama	1997	Warner Brothers	PG-13		
12	Poltergeist		115	Horror	1982	MGM / UA	PG		
13	Scarface		170	Action Cops & Robber	1983	Universal Studios	r		
14	Star Wars		124	Action Sci-Fi	1977	Lucas Film Ltd	PG		
15	The Hunt for Red October		135	Action Adventure	1989	Paramount Pictures	GP		
16		The Hunt for Red Oktober	.		.			Sean Connery	Alec Baldwin
17	The Wizard of Oz		101	Adventure	1939	MGM / UA	G		
18	The Wizard of Ozz		102	Adventure	1939	MGM - UA	g		
19		XMAS Vacation	.		.			Chevy Chase	Beverly D'Angelo

N = 19

Figure 10: Mismatched Observations.

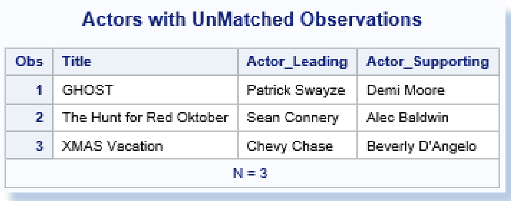


Movies with UnMatched Observations

Obs	Title	Length	Category	Year	Studio	Rating
1	Casablanca	103	Drama	1942	MGM / UA	PG
2	Christmas Vacatiion	97	Commedy	1989	Warner Brothers	PG-13
3	Christmas Vacation	97	Comedy	1989	Warner Brothers	PG-13
4	Dracula	130	Horror	1993	Columbia TriStar	R
5	Dressed to Kill	105	Drama Mysteries	1980	Filmways Pictures	R
6	Forrest Gumpp	143	Dramma	1994	Paramont Pictures	PG13
7	Ghost	127	Drama Romance	1990	Paramount Pictures	PG-13
8	Jaws	125	Action Adventure	1975	Universal Studios	PG
9	Jurassic Park	127	Action	1993	Universal Pictures	PG-13
10	Micheal	106	Drama	1997	Warner Brothers	PG-13
11	Poltergeist	115	Horror	1982	MGM / UA	PG
12	Scarface	170	Action Cops & Robber	1983	Universal Studios	r
13	Star Wars	124	Action Sci-Fi	1977	Lucas Film Ltd	PG
14	The Hunt for Red October	135	Action Adventure	1989	Paramount Pictures	GP
15	The Wizard of Oz	101	Adventure	1939	MGM / UA	G
16	The Wizard of Ozz	102	Adventure	1939	MGM - UA	g

N = 16

Figure 11: UnMatched Movies Observations.



Actors with UnMatched Observations

Obs	Title	Actor_Leading	Actor_Supporting
1	GHOST	Patrick Swayze	Demi Moore
2	The Hunt for Red Oktober	Sean Connery	Alec Baldwin
3	XMAS Vacation	Chevy Chase	Beverly D'Angelo

N = 3

Figure 12: UnMatched Actors Observations.

STEP 6: MATCH KEY FIELDS USING FUZZY MATCHING TECHNIQUES.

Once the data has been cleaned and transformed, a variety of fuzzy matching techniques are available for use. As mentioned in (Dunn, 2014), these techniques are designed to be used in a systematic way when a reliable key between data sources is nonexistent, inexact, or unreliable.

Fuzzy matching techniques are available with most, if not all, the leading software languages including R, Python, Java, and others (RosettaCode, 2018). SAS Institute offers four techniques for its users: the Soundex (phonetic matching) algorithm, and the SPEDIS, COMPLEV, and COMPGED functions to help make fuzzy matching easier and more effective (Sloan and Lafler, 2018).

APPLY THE SOUNDEX ALGORITHM

The Soundex (phonetic matching) algorithm involves matching files on words that sound alike. As one of the earliest fuzzy matching techniques, Soundex was invented and patented by Margaret K. Odell and Robert C. Russell in 1918 and 1922 to help match surnames that sound alike. It is limited to finding phonetic matches and adheres to the following rules when performing a search:

- Ignores case (case insensitive);
- Ignores embedded blanks and punctuations;
- Is better at finding English-sounding names.

Although the Soundex algorithm does a fairly good job with English-sounding names, it frequently falls short when dealing with the multitude of data sources found in today’s world economy where English- and non-English sounding names are commonplace. It also has been known to miss similar-sounding surnames like Rogers and Rodgers while matching dissimilar surnames such as Smith, Snthe and Schmitt (Foley, 1999).

So, how does the Soundex algorithm work? As implemented, SAS determines whether a name (or a variable’s contents) sounds like another by converting each word to a code. The value assigned to the code consists of the first letter in the word followed by one or more digits. Vowels, A, E, I, O and U, along with H, W, Y, and non-alphabetical characters do not receive a coded value and are ignored; and double letters (e.g., ‘TT’) are assigned a single code value for both letters. The codes derived from each word conform to the letters and values are found in Table 1.

SOUNDEX Algorithm Rules

Letter	Value
B, P, F, V	**1**
C, S, G, J, K, Q, X, Z	**2**
D, T	**3**
L	**4**
M, N	**5**
R	**6**

Table 1: Soundex Algorithm Rules

The general syntax of the Soundex algorithm takes the form of:

Variable =* “character-string”

To examine how the movie title, Rocky, is assigned a value of R22, R has a value of 6 but is retained as R, O is ignored, C is assigned a value of 2, K is assigned a value of 2, and Y is ignored. The converted code for “Rocky” is then matched with any other name that has the same assigned code.

In the next example, we use the Soundex algorithm’s **=*** operator in a simple DATA step WHERE statement with the work.Movies_with_Unmatched_Obs data set created in Step #5 earlier, to find similar sounding Movie Titles.

DATA Step Code with SOUNDEX Algorithm:

```
DATA work.Soundex_Matches ;
  SET work.Movies_with_Unmatched_Obs ;
  WHERE Title =* “Michael” ;
RUN ;

PROC PRINT DATA=work.Soundex_Matches NOOBS ;
  TITLE “Soundex Algorithm Matches” ;
RUN ;
```

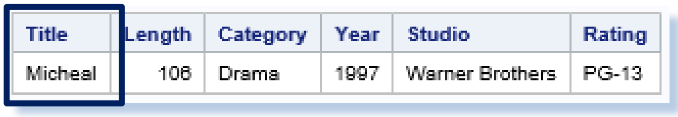
In the next example, the Soundex algorithm is illustrated using the **=*** operator in a simple SAS PROC SQL step with a WHERE-clause to find similar sounding Movie Titles.

PROC SQL Code with SOUNDEX Algorithm:

```
proc sql ;
 select *
  from work.Movies_with_Unmatched_Obs
   where Title =* "Michael" ;
quit ;
```

The results from both SOUNDEX algorithm examples are displayed in Figure 13.

Results:



Title	Length	Category	Year	Studio	Rating
Micheal	106	Drama	1997	Warner Brothers	PG-13

Figure 13: The result of the Soundex match for “Michael”

APPLY THE SPEDIS FUNCTION

The SPEDIS, or Spelling Distance, function and its two arguments evaluate possible matching scenarios by translating a keyword into a query containing the smallest distance value. Because the SPEDIS function evaluates numerous scenarios, it can experience varying performance issues in comparison to other matching techniques. The SPEDIS function evaluates query and keyword arguments returning non-negative spelling distance values. A derived value of zero indicates an exact match. Generally, derived values are less than 100, but, on occasion, can exceed 200. The authors have used and recommend using the SPEDIS function to control the matching process by specifying spelling distance values greater than zero and in increments of 10 (e.g., 10, 20, etc.).

So, how does the SPEDIS function work? As implemented, the SPEDIS function determines whether two names (or variables’ contents) are alike by computing an asymmetric spelling distance between two words. The SPEDIS function computes the costs associated with converting the keyword to the query, as illustrated in Table 2.

SPEDIS Cost Rules		
Operation	**Cost**	**Description**
Match	**0**	**No change**
Singlet	**25**	**Delete one of a double letter**
Doublet	**50**	**Double a letter**
Swap	**50**	**Reverse the order of two consecutive letters**
Truncate	**50**	**Delete a letter from the end**
Append	**35**	**Add a letter to the end**
Delete	**50**	**Delete a letter from the middle**
Insert	**100**	**Insert a letter in the middle**
Replace	**100**	**Replace a letter in the middle**
Firstdel	**100**	**Delete the first letter**
Firstins	**200**	**Insert a letter at the beginning**
Firstrep	**200**	**Replace the first letter**

Table 2: SPEDIS Cost Rules

The general syntax of the SPEDIS function takes the form of:

SPEDIS (query, keyword)

In this example, a simple DATA step with a WHERE statement shows the observations derived by the SPEDIS function for finding exact matches for the Movie Title, “Michael”.

DATA Step Code with SPEDIS Function:

```
PROC PRINT DATA=work.Movies_with_Unmatched_Obs NOOBS ;
  TITLE "SPEDIS Function Matches" ;
  WHERE SPEDIS(Title,"Michael") LE 10 ;
RUN ;
```

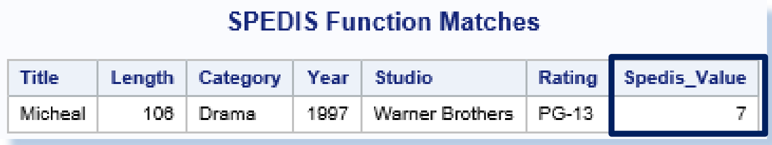
In the next example, a simple PROC SQL query with a WHERE-clause and CALCULATED keyword is specified to capture and show the observations derived by the SPEDIS function for finding exact matches for the Movie Title, “Michael”.

PROC SQL Code with SPEDIS Function:

```
PROC SQL ;
 TITLE “SPEDIS Function Matches” ;
 SELECT *,
        SPEDIS(Title,“Michael”) AS Spedis_Value
  FROM work.Movies_with_Unmatched_Obs
   WHERE CALCULATED Spedis_Value LE 10 ;
QUIT ;
```

The results from both SOUNDEX algorithm examples are displayed in Figure 14. Only “Michael” and “Micheal” were chosen. This matches the result we obtained from the SOUNDEX inquiry displayed in Figure 13.

Results:



SPEDIS Function Matches

Title	Length	Category	Year	Studio	Rating	Spedis_Value
Micheal	108	Drama	1997	Warner Brothers	PG-13	7

Figure 14: The result of a SPEDIS match for “Michael”

APPLY THE COMPLEV FUNCTION

The COMPLEV, or Levenshtein Edit Distance, function is another fuzzy matching SAS technique. COMPLEV counts the minimum number of single-character insert, delete, or replace operations needed to determine how close two strings are. Unlike the SPEDIS function and COMPGED function (discussed later), the COMPLEV function assigns a score for each operation and returns a value indicating the number of operations. The general syntax of the COMPLEV function takes the form of:

COMPLEV (string-1, string-2 <, cutoff-value> <, modifier>)

Required Arguments:

string-1 specifies a character variable, constant or expression.
string-2 specifies a character variable, constant or expression.

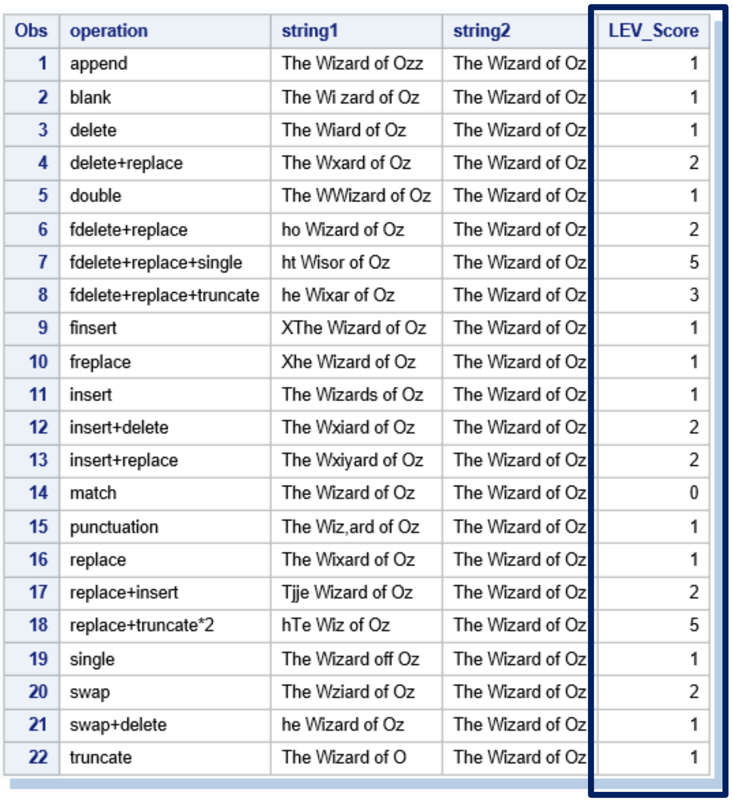
Optional Arguments:

cutoff-value specifies a numeric variable, constant or expression. If the actual Levenshtein edit distance is greater than the value of *cutoff*, the value that is returned is equal to the value of ***cutoff***.

modifier specifies a value that alters the action of the COMPLEV function. Valid modifier values are:

i or I	Ignores the case in string-1 and string-2.
l or L	Removes leading blanks before comparing the values in string-1 or string-2.
n or N	Ignores quotation marks around string-1 or string-2.
: (colon)	Truncates the longer of string-1 or string-2 to the length of the shorter string.

In Figure 15, below, we show the number of operations that are performed by the COMPLEV function as it compares string-1 with string-2. As can be seen, the smaller the LEV_Score the better the match.



Obs	operation	string1	string2	LEV_Score
1	append	The Wizard of Ozz	The Wizard of Oz	1
2	blank	The Wi zard of Oz	The Wizard of Oz	1
3	delete	The Wiard of Oz	The Wizard of Oz	1
4	delete+replace	The Wxard of Oz	The Wizard of Oz	2
5	double	The WWizard of Oz	The Wizard of Oz	1
6	fdelete+replace	ho Wizard of Oz	The Wizard of Oz	2
7	fdelete+replace+single	ht Wisor of Oz	The Wizard of Oz	5
8	fdelete+replace+truncate	he Wixar of Oz	The Wizard of Oz	3
9	finsert	XThe Wizard of Oz	The Wizard of Oz	1
10	freplace	Xhe Wizard of Oz	The Wizard of Oz	1
11	insert	The Wizards of Oz	The Wizard of Oz	1
12	insert+delete	The Wxiard of Oz	The Wizard of Oz	2
13	insert+replace	The Wxiyard of Oz	The Wizard of Oz	2
14	match	The Wizard of Oz	The Wizard of Oz	0
15	punctuation	The Wiz,ard of Oz	The Wizard of Oz	1
16	replace	The Wixard of Oz	The Wizard of Oz	1
17	replace+insert	Tjje Wizard of Oz	The Wizard of Oz	2
18	replace+truncate*2	hTe Wiz of Oz	The Wizard of Oz	5
19	single	The Wizard off Oz	The Wizard of Oz	1
20	swap	The Wziard of Oz	The Wizard of Oz	2
21	swap+delete	he Wizard of Oz	The Wizard of Oz	1
22	truncate	The Wizard of O	The Wizard of Oz	1

Figure 15: COMPLEV (Levenshtein Edit Distance) Number of Operations.

In the example below, we use the COMPLEV function to determine the best possible match with DRAMA. As illustrated in Figure 16, the COMPLEV_Number column displays the number of operations that have been performed. The lower the value the better the match (e.g., 0 = Best match, 1 = Next Best match, etc.). DRAMA matches itself for a score of 0, and DRAMMA is the next best match with a score of 1.

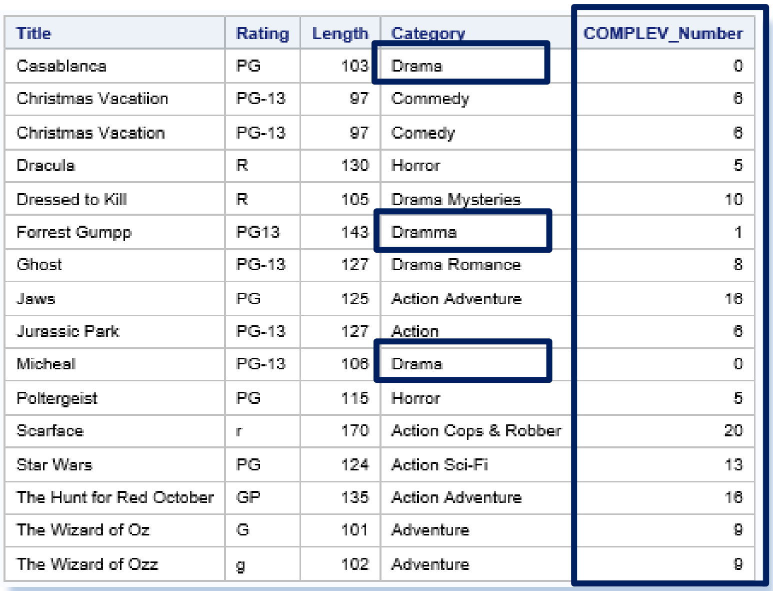
PROC SQL Code with COMPLEV Function:

```
proc sql ;
 select M.Title,
        Rating,
        Length,
        Category,
        COMPLEV(M.Category,"Drama") AS COMPLEV_Number
```

```
    from work.Movies_with_Unmatched_Obs M
     order by M.Title ;
  quit ;
```

Figure 16 shows the calculations for the Levenshtein Edit Distance for different spelling variations for ‘Drama’ in the column, CATEGORY.

Results:



Title	Rating	Length	Category	COMPLEV_Number
Casablanca	PG	103	Drama	0
Christmas Vacatiion	PG-13	97	Commedy	6
Christmas Vacation	PG-13	97	Comedy	6
Dracula	R	130	Horror	5
Dressed to Kill	R	105	Drama Mysteries	10
Forrest Gumpp	PG13	143	Dramma	1
Ghost	PG-13	127	Drama Romance	8
Jaws	PG	125	Action Adventure	16
Jurassic Park	PG-13	127	Action	6
Micheal	PG-13	106	Drama	0
Poltergeist	PG	115	Horror	5
Scarface	r	170	Action Cops & Robber	20
Star Wars	PG	124	Action Sci-Fi	13
The Hunt for Red October	GP	135	Action Adventure	16
The Wizard of Oz	G	101	Adventure	9
The Wizard of Ozz	g	102	Adventure	9

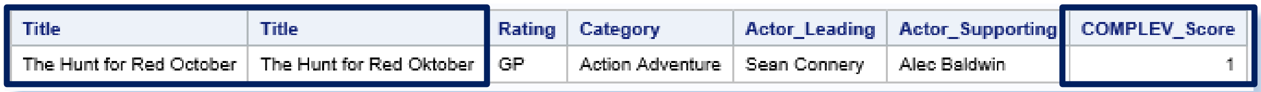
Figure 16: The results of a COMPLEV match with the category of Drama.

In the next example, the COMPLEV function’s computed value is limited to 1 or less using a WHERE-clause. The results show the observation associated with the movie “The Hunt for Red October” in the string-1 argument matches the value of “The Hunt for Red Oktober” in the string-2 argument, as shown in Figure 17.

PROC SQL Code with COMPLEV Function:

```
  PROC SQL ;
   SELECT M.Title,
          A.Title,
          Rating,
          Category,
          Actor_Leading,
          Actor_Supporting,
          COMPLEV(M.Title,A.Title) AS COMPLEV_Score
    FROM work.Movies_with_Unmatched_Obs M,
         work.Actors_with_Unmatched_Obs A
     WHERE CALCULATED COMPLEV_Score LE 1
      ORDER BY M.TITLE ;
  QUIT ;
```

Results:



Title	Title	Rating	Category	Actor_Leading	Actor_Supporting	COMPLEV_Score
The Hunt for Red October	The Hunt for Red Oktober	GP	Action Adventure	Sean Connery	Alec Baldwin	1

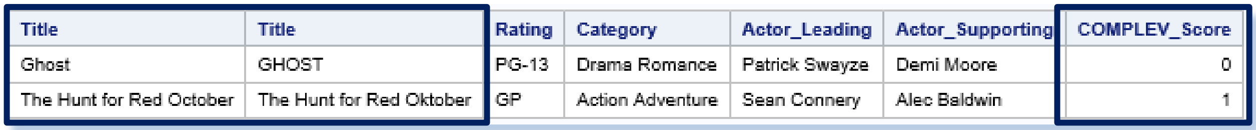
Figure 17: The results of a COMPLEV match where the Number of Operations is 1 or less.

In the next example, the COMPLEV function has a modifier value of “INL” to ignore the case, remove leading blanks, and ignore quotes around string-1 and string-2 and a value for the COMPLEV_Score of 1 or less. The results show the observation associated with the movie “Ghost” in the argument for string-1 matches the value of “GHOST” in the argument for string-2, and the observation associated with the movie “The Hunt for Red October” in the string-1 argument matches the value of “The Hunt for Red Oktober” in the string-2 argument, as shown in Figure 18.

PROC SQL Code with COMPLEV Function and Arguments:

```
PROC SQL ;
 SELECT M.Title,
        A.Title,
        Rating,
        Category,
        Actor_Leading,
        Actor_Supporting,
        COMPLEV(M.Title,A.Title,”INL”) AS COMPLEV_Score
  FROM work.Movies_with_Unmatched_Obs M,
       work.Actors_with_Unmatched_Obs A
   WHERE CALCULATED COMPLEV_Score LE 1
    ORDER BY M.TITLE ;
QUIT ;
```

Results:



Title	Title	Rating	Category	Actor_Leading	Actor_Supporting	COMPLEV_Score
Ghost	GHOST	PG-13	Drama Romance	Patrick Swayze	Demi Moore	0
The Hunt for Red October	The Hunt for Red Oktober	GP	Action Adventure	Sean Connery	Alec Baldwin	1

Figure 18: The results of a COMPLEV match with arguments “INL” specified on pairs of titles.

APPLY THE COMPGED FUNCTION

The COMPGED function is another fuzzy matching technique which is facilitated by a SAS function. It works by computing and using a Generalized Edit Distance (GED) score when comparing two text strings. The Generalized Edit Distance score is a generalization of the Levenshtein edit distance, which is a measure of dissimilarity between two strings (Teres, 2011). When using the COMPGED function to match data sets with unreliable identifiers (or keys), the higher the GED score the less likely the two strings match (Sloan and Hoicowitz, 2016). Conversely, for the greatest likelihood of a match with the COMPGED function users should seek the lowest derived score from evaluating all the possible ways of matching string-1 with string-2.

The COMPGED function returns values that are multiples of 10, e.g., 20, 100, 200, etc. It’s been our experience, as well as others, that most COMPGED scores of 100 or less are valid matches for the comparison that they are performing (Cadieux and Bretheim, 2014). The COMPGED function compares two character strings, along with optional parameters indicating whether the cases need to match, leading blanks or quotation marks need to be removed, and longer strings should be truncated. The general syntax of the COMPGED function takes the form of:

COMPGED (string-1, string-2 <, cutoff-value> <, modifier>)

Required Arguments:

string-1 specifies a character variable, constant or expression.

string-2 specifies a character variable, constant or expression.

Optional Arguments:

cutoff-value specifies a numeric variable, constant or expression. If the actual generalized edit distance is greater than the value of ***cutoff***, the value that is returned is equal to the value of ***cutoff***.

modifier specifies a value that alters the action of the COMPGED function. Valid modifier values are:

i or I	Ignores the case in string-1 and string-2.
l or L	Removes leading blanks before comparing the values in string-1 or string-2.
n or N	Ignores quotation marks around string-1 or string-2.
: (colon)	Truncates the longer of string-1 or string-2 to the length of the shorter string.

Table 3, below, shows the different point values that COMPGED assigns for changes from one character string to another.

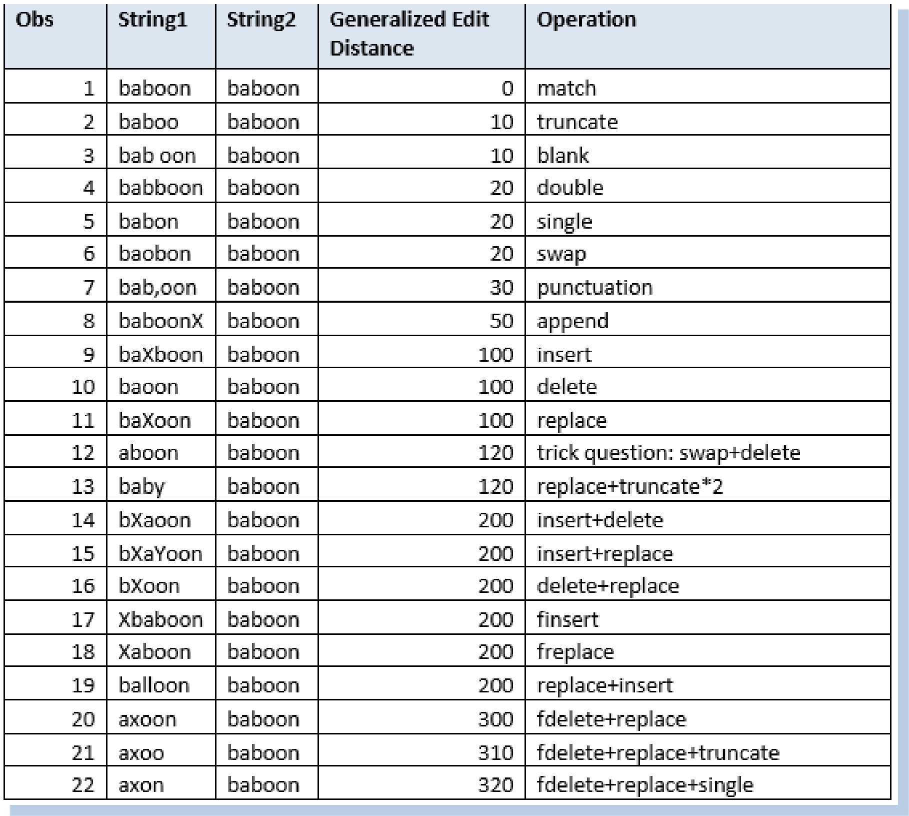
COMPGED Scoring Algorithm

Operation	Default Cost in Units	Description of Operation
APPEND	**50**	When the output string is longer than the input string, add any one character to the end of the output string without moving the pointer.
BLANK	**10**	Do any of the following: Add one space character to the end of the output string without moving the pointer. When the character at the pointer is a space character, advance the pointer by one position without changing the output string. When the character at the pointer is a space character, add one space character to the end of the output string, and advance the pointer by one position. If the cost for BLANK is set to zero by the COMPCOST function, the COMPGED function removes all space characters from both strings before doing the comparison.
DELETE	**100**	Advance the pointer by one position without changing the output string.
DOUBLE	**20**	Add the character at the pointer to the end of the output string without moving the pointer.
FDELETE	**200**	When the output string is empty, advance the pointer by one position without changing the output string.
FINSERT	**200**	When the pointer is in position one, add any one character to the end of the output string without moving the pointer.
FREPLACE	**200**	When the pointer is in position one and the output string is empty, add any one character to the end of the output string, and advance the pointer by one position.
INSERT	**100**	Add any one character to the end of the output string without moving the pointer.
MATCH	**0**	Copy the character at the pointer from the input string to the end of the output string, and advance the pointer by one position.
PUNCTUATION	**30**	Do any of the following: Add one punctuation character to the end of the output string without moving the pointer. When the character at the pointer is a punctuation character, advance the pointer by one position without changing the output string. When the character at the pointer is a punctuation character, add one punctuation character to the end of the output string, and advance the pointer by one position.

REPLACE	100	Add any one character to the end of the output string, and advance the pointer by one position.
SINGLE	20	When the character at the pointer is the same as the character that follows in the input string, advance the pointer by one position without changing the output string.
SWAP	20	Copy the character that follows the pointer from the input string to the output string. Then copy the character at the pointer from the input string to the output string. Advance the pointer two positions.
TRUNCATE	10	When the output string is shorter than the input string, advance the pointer by one position without changing the output string.

Table 3: COMPGED scoring algorithm

An example of the scoring used in the SAS COMPGED function when matching string-1 with string-2, re-sorted from an example available in the Help screen for the COMPGED function is displayed in Figure 19 (Sloan and Hoicowitz, 2016).



Obs	String1	String2	Generalized Edit Distance	Operation
1	baboon	baboon	0	match
2	baboo	baboon	10	truncate
3	bab oon	baboon	10	blank
4	babboon	baboon	20	double
5	babon	baboon	20	single
6	baobon	baboon	20	swap
7	bab,oon	baboon	30	punctuation
8	baboonX	baboon	50	append
9	baXboon	baboon	100	insert
10	baoon	baboon	100	delete
11	baXoon	baboon	100	replace
12	aboon	baboon	120	trick question: swap+delete
13	baby	baboon	120	replace+truncate*2
14	bXaoon	baboon	200	insert+delete
15	bXaYoon	baboon	200	insert+replace
16	bXoon	baboon	200	delete+replace
17	Xbaboon	baboon	200	finsert
18	Xaboon	baboon	200	freplace
19	balloon	baboon	200	replace+insert
20	axoon	baboon	300	fdelete+replace
21	axoo	baboon	310	fdelete+replace+truncate
22	axon	baboon	320	fdelete+replace+single

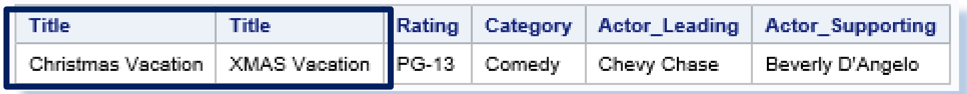
Figure 19: An example of the scoring used while matching on pairs of titles using the COMPGED function.

In the example below, traditional WHERE-clause logic with the UPCASE function is specified to equate the values of string-1 with string-2. Although this approach is far less efficient and can be more time consuming than using traditional data cleaning methods or the COMPGED function, the results show the value for the movie “Christmas Vacation” in the string-1 argument matches the value of “XMAS Vacation” in the string-2 argument, as shown in Figure 20.

PROC SQL Code with Traditional WHERE-clause logic:

```
PROC SQL ;
 SELECT M.Title,
        A.Title,
        Rating,
        Category,
        Actor_Leading,
        Actor_Supporting
  FROM work.Movies_with_Unmatched_Obs M,
       work.Actors_with_Unmatched_Obs A
   WHERE UPCASE(A.Title) = "XMAS VACATION"
     AND UPCASE(M.Title) = "CHRISTMAS VACATION"
    ORDER BY M.TITLE ;
QUIT ;
```

Results:



Title	Title	Rating	Category	Actor_Leading	Actor_Supporting
Christmas Vacation	XMAS Vacation	PG-13	Comedy	Chevy Chase	Beverly D'Angelo

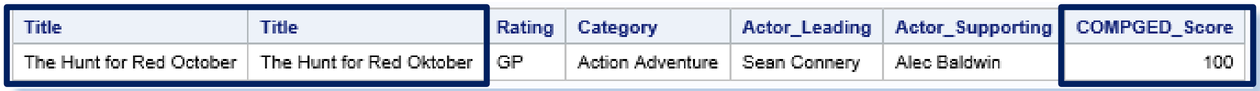
Figure 20: The results of using traditional WHERE-clause logic on pairs of titles.

In the next example, the COMPGED function has a "cutoff-value" for the COMPGED_Score set at 100. The results show the row associated with the movie "The Hunt for Red October" in the argument for string-1 matches the value of "The Hunt for Red Oktober" in the argument for string-2, as shown in Figure 21.

PROC SQL Code with COMPGED Function:

```
PROC SQL ;
 SELECT M.Title,
        A.Title,
        Rating,
        Category,
        Actor_Leading,
        Actor_Supporting,
        COMPGED(M.Title,A.Title) AS COMPGED_Score
 FROM work.Movies_with_Unmatched_Obs M,
      work.Actors_with_Unmatched_Obs A
  WHERE CALCULATED COMPGED_Score LE 100
   ORDER BY M.TITLE ;
QUIT ;
```

Results:



Title	Title	Rating	Category	Actor_Leading	Actor_Supporting	COMPGED_Score
The Hunt for Red October	The Hunt for Red Oktober	GP	Action Adventure	Sean Connery	Alec Baldwin	100

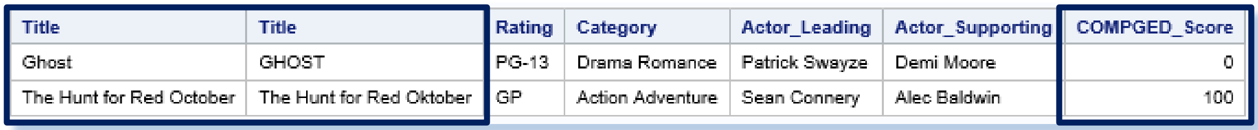
Figure 21: The results of a COMPGED match on pairs of titles.

In the next example, the COMPGED function has a modifier value of "INL" to ignore the case, remove leading blanks, and ignore quotes around string-1 and string-2 and a "cutoff-value" for the COMPGED_Score set at 100. The results show the row associated with the movie "Ghost" in the argument for string-1 matches the value of "GHOST" in the argument for string-2, as shown in Figure 22.

PROC SQL Code with COMPGED Function and Arguments:

```
  PROC SQL ;
   SELECT M.Title,
          A.Title,
          Rating,
          Category,
          Actor_Leading,
          Actor_Supporting,
          COMPGED(M.Title,A.Title,’INL’) AS COMPGED_Score
   FROM work.Movies_with_Unmatched_Obs M,
        work.Actors_with_Unmatched_Obs A
    WHERE CALCULATED COMPGED_Score LE 100
     ORDER BY M.TITLE ;
  QUIT ;
```

Results:



Title	Title	Rating	Category	Actor_Leading	Actor_Supporting	COMPGED_Score
Ghost	GHOST	PG-13	Drama Romance	Patrick Swayze	Demi Moore	0
The Hunt for Red October	The Hunt for Red Oktober	GP	Action Adventure	Sean Connery	Alec Baldwin	100

Figure 22: The results of a COMPGED match with arguments on pairs of titles.

Use the Lower Score

For those fuzzy matching techniques that are not commutative (it matters which data set is placed first and which is placed second), use the lower score that results from the different sequences.

Eliminate Entries where the Word Counts are Significantly Different

Eliminate entries where the word counts are significantly different (the level of significance will be determined based on the data sets being compared).

VALIDATION

As can be seen when comparing the SOUNDEX and SPEDIS methods, and when looking at the results of COMPLEV and COMPGED, these methods worked well on a test data set that was designed to illustrate the results. It should be noted that the authors found the COMPLEV function to be best used when comparing simple strings where data sizes and/or speed of comparison is important, such as when working with large data sets. It should also be noted that the COMPGED function required more processing time to complete because its more exhaustive and thorough capabilities.

Research was conducted on 50,000 business names to manually identify fuzzy matches using SAS’ COMPGED function (Sloan and Hoicowitz, 2016). The intent of the study was to identify false negatives by looking at an alphabetic sort of the business names. From the extracted test files the authors identified false positives. Finally, the conditions that were specified in the COMPGED function were repeated until the false positives and false negatives were significantly reduced. This then became part of the fuzzy matching process and achieved improved results efficiently and on demand.

CONCLUSION

When data originating from multiple sources contain duplicate observations, duplicate and/or unreliable keys, missing values, invalid values, capitalization and punctuation issues, inconsistent matching variables, and imprecise text identifiers, the matching process is often compromised by unreliable and/or unpredictable results. This paper demonstrates a five-step approach including identifying, cleaning and standardizing data irregularities, conducting data transformations, and utilizing special-purpose programming techniques such as the application of SAS functions, the SOUNDEX algorithm, the SPEDIS function, approximate string matching functions including COMPGED and COMPLEV, and an assortment of constructive programming techniques to standardize and combine data sets together when the matching columns are unreliable or less than perfect.

REFERENCES

Cadieux, Richard and Daniel R. Brethiem (2014). *"Matching Rules: Too Loose, Too Tight, or Just Right?"*, Proceedings of the 2014 SAS Global Forum (SGF) Conference.

Cody, Ron (2017). *"Cody's Data Cleaning Techniques Using SAS®, Third Edition"*, SAS Press, SAS Institute, Cary, NC, USA.

Dunn, Toby (2014). *"Getting the Warm and Fuzzy Feeling with Inexact Matching"*, Proceedings of the 2014 SAS Global Forum (SGF) Conference.

Foley, Malachy J. (1999). *"Fuzzy Merges: Examples and Techniques"*, Proceedings of the 1999 SAS Users Group International (SUGI) Conference.

Lafler, Kirk Paul and Stephen Sloan (2019). "Fuzzy Matching Programming Techniques Using SAS® Software", Proceedings of the 2019 Western Users of SAS Software (WUSS) Conference.

Lafler, Kirk Paul and Stephen Sloan (2017). "Fuzzy Matching Programming Techniques Using SAS® Software", Proceedings of the 2017 South Central SAS Users Group (SCSUG) Conference.

Lafler, Kirk Paul and Stephen Sloan (2017). "A Quick Look at Fuzzy Matching Programming Techniques Using SAS® Software", Proceedings of the 2017 Western Users of SAS Software (WUSS) Conference.

Lafler, Kirk Paul (2017). "Removing Duplicates Using SAS®", Proceedings of the 2017 South Central SAS Users Group (SCSUG) Conference.

Lafler, Kirk Paul (2016). *"Removing Duplicates Using SAS®"*, Proceedings of the 2016 MidWest SAS Users Group (MWSUG) Conference.

Patridge, Charles (1997). *"*The Fuzzy Feeling SAS Provides: Electronic Matching of Records without Common Keys*"*, Proceedings of the 1997 SAS Users Group International (SUGI) Conference.

Russell, Kevin (January 27, 2015). *"*How to Perform a Fuzzy Match Using SAS Functions*"*. blogs.sas.com.

Roesch, Amanda (2012). *"Matching Data Using Sounds-Like Operators and SAS® Compare Functions"*, Proceedings of the 2012 SAS Global Forum (SGF) Conference.

Sloan, Stephen and Dan Hoicowitz (2016). *"Fuzzy Matching: Where Is It Appropriate and How Is It Done? SAS Can Help."*, Proceedings of the 2016 SAS Global Forum (SGF) Conference.

Staum, Paulette (2007). *"Fuzzy Matching using the COMPGED Function"*, Proceedings of the 2007 NorthEast SAS Users Group (NESUG) Conference.

Teres, Jedediah J. (2011). *"Using SQL Joins to Perform Fuzzy Matches on Multiple Identifiers"*, Proceedings of the 2011 NorthEast SAS Users Group (NESUG) Conference.

"Transforming SAS Data Sets", (2000). http://www.rhoworld.com/pdf/ch599.pdf.

Zirbel, Douglas (2009). *"Learn the Basics of PROC TRANSPOSE"*, Proceedings of the 2009 SAS Global Forum (SGF) Conference.

ACKNOWLEDGMENTS

The authors thank Gary Moore, PharmaSUG 2020 Academic Chair and Conference Co-Chair for accepting our abstract and paper; the PharmaSUG Conference Committee; and SAS Institute for organizing and supporting a great conference!

TRADEMARK CITATIONS

CONTACT INFORMATION

Stephen Sloan has worked at Accenture in the Services, Consulting, and Digital groups and is currently a senior manager in the SAS Analytics area. He has worked in a variety of functional areas including Project Management, Data Management, and Statistical Analysis. Stephen has had the good fortune to have worked with many talented people at SAS Institute. Stephen has a B.A. in Mathematics from Brandeis University, M.S. degrees in Mathematics and Computer Science from Northern Illinois University, an MBA from Stern Business School at New York University, and a graduate certificate in Financial Analytics from Stevens Institute.

Kirk Paul Lafler is an entrepreneur, consultant, programmer, data analyst and educator, and has used SAS software since 1979. Currently, Kirk works as an independent SAS and SQL application developer and programmer; a lecturer and adjunct professor at San Diego State University; an advisor and adjunct professor at the University of California San Diego Extension; and as an independent educator teaching SAS, SQL, Python and R courses, workshops, seminars and webinars to thousands of users around the world.

Kirk also authored several books including PROC SQL: Beyond the Basics Using SAS, Third Edition (SAS Press. 2019) and Google® Search Complete (Odyssey Press. 2014); hundreds of papers and articles on a variety of SAS topics; served as an Invited speaker, educator, keynote and section leader at SAS user group conferences and meetings worldwide; and is the recipient of 25 "Best" contributed paper, hands-on workshop (HOW), and poster awards.

Comments and suggestions may be sent to:

Stephen Sloan
Senior Manager in SAS Analytics
Accenture
E-mail: Stephen.B.Sloan@accenture.com

~ ~ ~ ~ ~ ~ ~

Kirk Paul Lafler
SAS® Consultant, Application Developer, Programmer, Data Analyst, Educator and Author
E-mail: KirkLafler@cs.com
LinkedIn: https://www.linkedin.com/in/KirkPaulLafler
Twitter: @sasNerd