Paper CC40

All Data Are (Most Likely) *Not* Created Equal: A SAS® Macro to Compare Structure and Data Across Multiple Datasets

Jason L. Salemi, Baylor College of Medicine, Houston, TX

ABSTRACT

In nearly every discipline, from Accounting to Zoology, whether you are a student-in-training or an established professional, a central tenet of interacting with information is to “Know Thy Data”. Hasty compilation and analysis of inadequately vetted data can lead to misleading if not erroneous interpretation, which can have disastrous consequences ranging from business downfalls to adopting health interventions that worsen rather than improve the longevity and quality of people’s lives. In some situations, knowing thy data involves only a single analytic dataset, in which case review of a data dictionary to explore attributes of the dataset supplemented with univariate and bivariate statistics will do the trick. This has been discussed extensively in the literature and certainly in the SAS Global Forum and User’s Groups. In other scenarios, there is a need for comparing the structure, variables, and even values of variables across two datasets. Again, in this case, SAS offers a powerful COMPARE procedure to compare pairs of datasets, and many papers have offered macros to add additional functionality, refine the comparison, or simplify the analytic output. However, imagine the following scenario: you are provided with or download a myriad of datasets, perhaps which are produced quarterly or annually. Each dataset has a corresponding data dictionary and you might even be fortunate enough to have been provided with some code to facilitate importation into SAS. Your initial goal, perhaps a “first date” with your new datasets, is to understand whether variables exist in every dataset, whether there are differences in the type or length of each variable, the absolute and relative missingness of each variable, and whether the actual values being input for each variable are consistent. This paper describes the creation and use of a macro, “**compareMultipleDS**”, to make the first date with your data a pleasant one. Macro parameters through which the user can control which comparisons are performed/reported as well as the appearance of the generated “comparison report” are discussed, and use of the macro is demonstrated using two case studies that leverage publicly-available data.

INTRODUCTION

Unfortunately, Thomas Jefferson’s immortal declaration that “All men are created equal” does not translate well to the practical world of working with data. In nearly every discipline, from **A**ccounting to **Z**oology, whether you are a student-in-training or an established professional, a central tenet of interacting with information is to “Know Thy Data”. Hasty compilation and analysis of inadequately vetted data can lead to misleading if not erroneous interpretation, which can have disastrous consequences ranging from business downfalls to adopting health interventions that worsen rather than improve the longevity and quality of people’s lives. In some situations, *knowing thy data* involves only a single analytic dataset, in which case review of a data dictionary (and perhaps the CONTENTS or DATASETS procedures) to explore attributes of the dataset supplemented with univariate and bivariate statistics (UNIVARIATE, MEANS, and FREQ procedures) will do the trick. This has been discussed extensively in the literature and certainly in the SAS Global Forum and User’s Groups. In other scenarios, there is a need for comparing the structure, variables, and even values of variables across two datasets. This could be to compare a new data system to an old one, to assess competing algorithms for computing variables, to evaluate the validity and reliability of data collection based on a double data entry protocol, or to investigate two datasets prior to performing a match merge. Again, in this case, SAS offers a powerful COMPARE procedure to compare pairs of datasets, and many papers have offered macros to add additional functionality, refine the comparison, or simplify the analytic output. However, imagine the following scenario: you are provided with or download a myriad of datasets, perhaps which are produced quarterly or annually. Each dataset has a corresponding data dictionary and you might even be fortunate enough to have been provided with some code to facilitate importation into SAS. Your initial goal, perhaps a “first date” with your new datasets, is to understand not only the number of observations and variables in each dataset, but also whether:

- Variables exist in every dataset; if not, how many and in which datasets do they appear
- There are differences in the type (character, numeric, date/time) or length of variables that appear in more than one dataset
- The absolute and relative missingness of variables is consistent across datasets
- Variables (e.g., gender) have the same number of unique levels across all datasets
- The actual values being input for a particular variable are consistent (e.g., Male/Female vs. M/F vs. 1/0)

That is the purpose of this paper and the SAS macro upon which it is based. Currently, most dataset comparison tools have focused only on a few variable characteristics, or a detailed observation-level comparison of values with datasets compared in a pairwise fashion. In a 2013 SCSUG paper, a macro utility was presented to compare one or more pairs of data on different structural elements (labels, types, lengths); however, this macro presents only dichotomous indicators of agreement as opposed to the detail necessary to understand each dataset. It also was not designed to compare the unique values of variables entered across multiple datasets, which is vital to assess prior to stacking or merging. What is often done is to scan multiple data dictionaries (if they exist), or pour over the output from multiple CONTENTS and FREQ/MEANS/UNIVARIATE procedures to understand similarities and differences across all datasets. Attempts to combine the datasets first may be futile due to differences the types, lengths, or other characteristics of same-named variables. This macro, "**compareMultipleDS**", makes that first date with your data a pleasant one, and facilitates understanding with a report that highlights all of the details important to the user.

MACRO PARAMETERS AND USER INPUT

In order to facilitate user control over the comprehensiveness of the assessment and appearance of the report, there exist 19 parameters that can be specified by the user when calling *%compareMultipleDS* **(Table 1)**. The parameters can be divided into three major categories: 1) inputs that identify the identity and location of the datasets being compared, as well as the output log and report; 2) inputs that control which assessments are performed and/or produced in the comparison report; and 3) inputs that are primarily intended to control the appearance of the final report.

Macro parameter	What it controls	Default value	Example value inputs
Required user inputs			
`dsNames`	The list of datasets to be compared. Place the values inside %str() separating dataset names with a space.	*n/a*	%str(data1 data2, data3)
`libRef`	The library in which the datasets being compared are located.	work	myData, nchs, faers
`fileLocation`	The path to the folder in which you would like to store the output report (and log if `redirectLog` =1). Make certain to enter the final slash (\) at the end of the path.	*n/a*	C:\myProject\
`fileNamePrefix`	The name of the output report (and log if `redirectLog` =1). The current date will be added to the end of the file name.	compare	nchsReport
Optional user inputs that control what assessments are performed/reported			
`compareVars`	Enter 1 to compare variable-level characteristics; enter 0 to only compare the number of observations and variables.	1	0 or 1
`only1VarList`	Enter 1 to restrict variable comparison to a single summary table in the report; enter 0 to include subtables.	0	0 or 1
`compareValues`	Enter 1 to compare values of variables across datasets; enter 0 to exclude this comparison.	0	0 or 1
`ignoreCase`	Enter 1 to ignore case during value comparison; enter 0 to treat values like "boy", "Boy", and "BOY" as different.	1	0 or 1
`maxCharLen`	Controls the maximum length of values for character variables that will be compared and displayed. If the length of the value exceeds this number, it will be truncated to this length prior to comparison. Specifying a reasonable number (e.g., <=50) also makes the output more 'readable'.	50	5 – 100
`maxNumLevels`	Controls the maximum number of levels to consider in comparison of values across variables. This will prevent comparison of unique identifiers or other variables for which there may be as many levels as there are observations.	20	5 – 200
`runExceedMaxLevels`	Enter 1 to compare variables whose levels exceed `maxNumLevels` (only the first *n* levels are compared where *n*=`maxNumLevels`); enter 0 to skip any comparison of these variables (this is the most time consuming task when comparing large datasets).	0	0 or 1
`pctMissOnly`	Enter 1 to display only the percent (%) missing, but not the frequency; enter 0 to display both the frequency and %.	0	0 or 1
`pctValueOnly`	Enter 1 to display only the percent (%) of the time each value appears in the dataset, but not the frequency; enter 0 to display both the frequency and %.	0	0 or 1
`valueOrderFreq`	Enter 1 to order the values of all variables by their frequency across all datasets; enter 0 to order variables alphabetically. *NOTE:* All variables with a number of levels that is >= `maxNumLevels` will be presented in descending order of their frequency, regardless of this parameter's value.	0	0 or 1
Optional user inputs that control the appearance of the final report			
`mySpacing`	Controls the CELLSPACING style option. Smaller numbers will allow more to fit on a single page, but reduces readability.	8	2 – 15
`myPadding`	Controls the CELLPADDING style option. Smaller numbers will allow more to fit on a single page, but reduces readability.	2	0 – 4

Macro parameter	What it controls	Default value	Example value inputs
`myFontSize`	Controls the FONTSIZE style option. Smaller numbers will allow more to fit on a single page, but reduces readability.	1	0.5 – 3
`fileType`	Controls whether the report is produced in RTF or PDF format.	rtf	rtf, pdf
Other optional user inputs			
`redirectLog`	Enter 1 to redirect the log to an external file; enter 0 to use the default log window. It would be important to redirect the log if you have many datasets with many variables being compared – prevents the log from filling up.	0	0 or 1

Table 1. Required and optional parameters for the compareMultipleDS macro

Any number of datasets can be specified in the `dsNames` parameter meaning that any number of datasets can be compared using this macro. Since the user specifies a single value for the `libRef` parameter, all datasets being compared must reside in the same folder (although this aspect of the macro could easily be modified). The user will specify a location to which the comparison report is saved (`fileLocation` parameter) and will provide the report with a name (`fileNamePrefix` parameter), unless the default name of ***compare*** is sufficient. Depending on the number and size of the datasets being compared, as well as the specification of parameters that control the comparison details, the log generated by this macro can be extensive and the log can fill up easily if running the macro in interactive mode. There are many approaches to avoid interruption of the macro due to a full log, this macro offers one option; namely, diverting the SAS log to an external file by setting the `redirectLog` parameter equal to 1. The log will be saved in the same location as the comparison report and will have the same name. Both the comparison report and log (if saved as an external file) will also have the date added as a suffix to the name. For example, if the user specifies “MyData” as the `fileNamePrefix` parameter, and the date is March 9, 2015, the file saved will be titled “MyData_2015-03-09”.

By default, the comparison report will include only a basic dataset-level comparison of the number of observations and variables in each dataset (described later); however, the user has a series of indicator parameters (where 1 and 0 are valid values) that control conditional sections of the macro and ultimately what is entailed in the data comparison being performed. The user, by specifying `compareVars` = 1 can request variable-level comparison of the presence/absence, type, length, and number of unique levels of each variable across datasets. If the user specifies `only1VarList` = 1, only a single summary table is generated, which lists all variables that appear in at least one dataset. If `only1VarList` = 0, then up to 7 additional tables are produced to categorize variables based on the nature of their characteristics. With this specification of `compareVars` = 1, a missingness report is also generated, and the `pctMissOnly` parameter can control whether the count and percentage of missing observations for each variable are reported or the percentage alone.

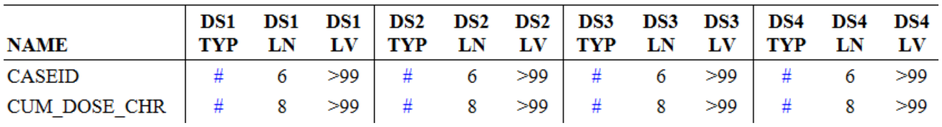
One of the most useful aspects of this macro is the ability to drilldown to the level of the values for each variable; similar to running a FREQ procedure for each variable in each dataset and comparing the results across all datasets in an easy-to-read format. These value-level comparisons can be generated by setting `compareValues` = 1. Depending on the size of the datasets, and the nature of the variables, there are a number of additional options the user should consider. Not every variable should have a comparison of each unique value; in some cases it is not necessary (e.g., household income down to the dollar), and in others it is ill-advised because of the unwieldy length of the comparison report (e.g., a unique identifier for each observation or person-level identifier such as a social security number). Therefore, a number of parameters work in concert to dictate how each variable is processed to compare its values. The most important of these parameters is `maxNumLevels`, which establishes a limit on the number of levels under which a variable must have in order to have each of its unique values compared. This pertains to variables that are character, numeric, or date/time. For example, if `maxNumLevels` = 60, then any variable that has 60 unique levels or less will have each unique value, along with its frequency and percent listed in the comparison report. If a numeric variable representing the state of birth (in the US) had 51 levels (each of the 50 states, plus a missing level of -99), then each value would be included and compared in the report. For variables whose number of unique values exceeds the number specified by `maxNumLevels`, the nature of the comparison run depends on 1) whether the type of the variable is character, numeric, or date/time; and 2) the value of the `runExceedMaxLevels` parameter:

- For numeric or date/time fields (e.g., date of surgery), the comparison will consist of a comparison of minimum and maximum values entered in each dataset. This is included in two separate tables in the comparison report, one for numeric values and one for date/time fields. Each variable will only take up one line in the report, summarizing its range in each dataset.
- For character variables, no value-level comparison is run when `runExceedMaxLevels` = 0, processing terminates. If the user specifies `runExceedMaxLevels` = 1, a value-level comparison proceeds as describe earlier for variables whose number of unique levels is ≤ `maxNumLevels` with one major difference:

the number of levels that will appear in the comparison report will be restricted to the number specified in `maxNumLevels`. Which values are presented is dictated by their relative frequency – values are listed in, selected, and then presented in order of descending frequency (more common value listed first). This is useful for variables that the user might want to get a "feel" for how they are entered (e.g., if a decimal is being entered for ICD-10 diagnosis codes), or understand the most common values across datasets.

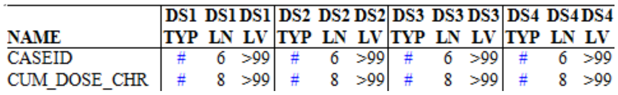
For all value-level comparisons of character variables, there are three additional variables that control aspects of the comparison. If the user specifies `ignoreCase` = 1, then values that only differ by their case ("UNITED STATES", "united states", "United States") will be treated as one unique value; otherwise, they will be treated as different values. The value of `maxCharLen` will determine how many characters in each value are compared and subsequently displayed. Some variable values can be extremely long and their full display in the comparison report can lead to extremely poor readability and comprehension. Although restricting each variable to a manageable length (<100) can improve readability, caution should be taken. For example, consider a character variable representing race/ethnicity has the following values: "non-Hispanic white", "non-Hispanic black", "Hispanic", and "non-Hispanic other". If the user specifies `maxCharLen` = 12, then the truncated (and subsequently compared) unique values will be "non-Hispanic" and "Hispanic" and the differentiation between black, white, and other will be lost. Lastly, although the order in which character variables with a number of levels > `maxNumLevels` is predetermined (listed in order of descending frequency), the user has control over the order in which values for character variables with a number of levels ≤ `maxNumLevels` will be displayed. All levels for those variables will appear; if `valueOrderFreq` = 0 then values will appear alphabetically, if `valueOrderFreq` = 1 they will appear in descending order of their frequency.

The remaining parameters that can be specified by the user collectively control the appearance of the comparison report that is produced. These options become important as the number of datasets being compared increases, or the `maxCharLen` becomes large. The user can "play" with the values of `mySpacing`, `myPadding`, and `myFontSize` to control ODS style attributes and make the report easiest to read. Consider the examples below:



NAME	DS1 TYP	DS1 LN	DS1 LV	DS2 TYP	DS2 LN	DS2 LV	DS3 TYP	DS3 LN	DS3 LV	DS4 TYP	DS4 LN	DS4 LV
CASEID	#	6	>99	#	6	>99	#	6	>99	#	6	>99
CUM_DOSE_CHR	#	8	>99	#	8	>99	#	8	>99	#	8	>99

Figure 1a. Example of ODS RTF output with `mySpacing`=15, `myPadding`=3, and `myFontSize`=1.5



NAME	DS1 TYP	DS1 LN	DS1 LV	DS2 TYP	DS2 LN	DS2 LV	DS3 TYP	DS3 LN	DS3 LV	DS4 TYP	DS4 LN	DS4 LV
CASEID	#	6	>99	#	6	>99	#	6	>99	#	6	>99
CUM_DOSE_CHR	#	8	>99	#	8	>99	#	8	>99	#	8	>99

Figure 2b. Example of ODS RTF output with `mySpacing`=3, `myPadding`=0, and `myFontSize`=0.8

MACRO PROCESSING

Although there is a great level of additional detail in the macro, what is presented herein offers an understanding of macro processing that falls in line with the overarching tasks being accomplished. The macro begins with a series of checks to ensure that user-entered values for macro parameters are valid. For example, the macro first assess whether the library specified in the `libRef` parameter and the datasets listed in the `dsNames` parameter exist. Once all validation checks are passed, the macro determines the number of datasets that have been specified in the comparison and then begins a dataset-level DO loop using the macro variable nDS as the counter.

```
data _null_;
    call symput('dsNames', compbl("&dsNames."));
    call symput('nDS',      strip(put(countw("&dsNames.", " ,"),4.)));
run;
```

At the dataset level of processing, the CONTENTS procedure is used to generate an output dataset with the number of observations in the dataset, as well as the name, type, length, format, and informat of each variable in the dataset. The FREQ procedure with the NLEVELS option is used to determine the number of unique levels for each variable. Subsequent DATA step processing is used to determine the total number of variables in the dataset, to differentiate between numeric and date or date/time variables using the nature of the formats and informats, and to assign dataset-specific labels and formats that will be important in the final report. Next, the SQL procedure is used to create several macro variables that will store the names of variables from the dataset being processed. Each macro variable contains variables with different characteristics and that will require different comparison strategies and "processing". For example, the `FREQ1vars` macro variable stores the names of all variables in which the number of unique levels is less than or equal to the number specified by the user in the `maxNumLevels` macro parameter. These variables are processed and analyzed very differently from other character variables with more than the maximum specified unique variables (`FREQ2vars`), numeric variables that exceed the maximum (`MEANSvars`), or date fields that exceed the maximum (`DATEvars` and `DATETIMEvars`).

```
proc sql noprint;
%*Store names of all variables with NLEVELS <= maxNumLevels in macro variable FREQvars;
    select name into :FREQ1vars separated by ' '
    from _vars&i.
    where (nlev&i. <= &maxNumLevels.);
quit;
```

Then, at the variable level, for each variable list described above, different techniques were used in analysis. For numeric and date or date/time variables whose number of unique values listed within the dataset exceed the maximum number allowed by the user, the processing is relatively simple, using the UNIVARIATE procedure to output both 1) the min and max nonmissing values documented within the dataset, and 2) the frequency and proportion of missing values.

For all variables whose number of unique values (i.e., the number of levels), does not exceed the maximum number of levels specified by the user, processing is more complex as it includes not only an assessment of missingness, but also a value-level comparison for each variable. A FREQ procedure first determines the frequency and proportion of the time each value was entered. Conditional %IF %THEN %DO processing then begins to incorporate some of the user-specified inputs:

- If the user specifies `ignoreCase` = 1, then "boy", "Boy", and "BOY" should all represent the same value, and thus, the values of all variables in addition to have leading and trailing blanks removed using the STRIP function, will be capitalized using the UPCASE function.
- Based on what the user specifies for `maxCharLen`, the value of each variable will be truncated to that length using the SUBSTR function prior to performing any dataset-to-dataset comparisons and displaying.

For character variables whose number of unique values exceeds the maximum number of levels specified by the user, processing is similar to the process described in the previous paragraph, but with one additional consideration if the user desires to process these variables (if he/she specifies `runExceedMaxLevels` = 1`).` The macro will process and compare, from each dataset, only the first *n* levels of each variable in which *n* = `runExceedMaxLevels`. However, based on the value of `valueOrderFreq` the levels included will either be based on alphabetical order or relative frequency (only the *n* most frequent for a given dataset are considered).

After the dataset-, variable-, and value-level data are generated for a given dataset (one loop), the resultant files are merged in with the datasets that were processed previously. After the final dataset has been processed, DATA steps are used to calculate summary variables that summarize comparisons being made across all datasets. For example, in order to compare the presence (in the dataset or not), type and length of same-named variables across all datasets, we want to 1) count the number of datasets in which each variable appears, 2) create a flag indicating whether the variable appears in every dataset, and 3) create flags to denote whether there are differences in type or length, but not penalizing if the variable did not appear in a dataset. In this macro, that is accomplished by the code segment below.

```
*Create indicator flags;
inAllDS = 1; numIn = 0; sameType = 1; sameLength = 1;
*Create variables to use in determination of flags;
currentType = .; currentLength = .;
array _typeVars(&nDS.)    type1-type&nDS.;
array _lengthVars(&nDS.) length1-length&nDS.;
do i=1 to &nDS.;
  if (_typeVars(i) ^= .) then do;
    numIn = numIn + 1;
    if (currentType = .) then currentType = _typeVars(i);
      else if (currentType ^= .) AND (_typeVars(i) ^= currentType) then sameType = 0;
  end;
  if (_lengthVars(i) ^= .) then do;
    if (currentLength = .) then currentLength = _lengthVars(i);
      else if (currentLength ^= .) AND (_lengthVars(i) ^= currentLength) then sameLength = 0;
  end;
end;
if numIn ^= &nDS. then inAllDS = 0;
```

Once the indicator flags have been created for each desired comparison, the remainder of the macro uses ODS statements, combined with PRINT procedures and specific style attributes for each table, header, and data component of the output. A version of traffic lighting and color scheming is used to draw attention to areas of importance (e.g., when there are differences in type or length of variables; when there is 100% missingness of a variable in a given dataset). All temporary datasets generated by the macro are then deleted.

COMPARISON REPORT

The final output of the macro is that report that highlights your “first date” with your datasets. Depending on the macro parameter inputs specified by the user, the report will include a number of tables (up to 12) that summarize various dataset-, variable-, and value-level comparisons across datasets. These outputs are summarized in **Table 2**, described briefly below, and exemplified in the next section.

COMPARISONS AT THE DATASET LEVEL

All comparison reports will include a single dataset-level table that provides a very basic comparison of the number of observations and number of variables in each of the datasets included in the comparison. Since the name of the original datasets can be extremely lengthy, dataset “code names” are created by the macro (DS1 through DS*n*) and are displayed in this table. The fields that appear are:

- Dataset Name: Original name of the dataset
- Coded Name: The coded name of the dataset (see above)
- # Obs: The number of observations in the full dataset
- # Vars: The number of variables in the full dataset

The translation of which table each code name represents will also be displayed in the footnote of each table in the entire comparison report.

COMPARISONS AT THE VARIABLE LEVEL

If `compareVars` = 1, the comparison report will include at least one table that lists each variable that appears in at least one dataset. The fields that appear in this report are:

- Name: the name of the variable whose values are being compared (all names are capitalized).
- TYP: the type of the variable for a given dataset, entered as “$” for character variables, “#” for numeric variables (colored in blue), and “dt” for date/time variables (colored in green). The dataset code name will appear above “TYP”.
- LN: the length of the variable for a given dataset. The dataset code name will appear above “LN”.
- LV: the number of unique values (levels) of the variable for a given dataset. The dataset code name will appear above “LV”.
- “IN ALL”: this is an indicator variable that will either be “Y” or “N”. The “Y” represents variables that appear in each dataset. “N” will be denoted otherwise and all “N” values are colored in red.
- “NUM IN”: represents the number of datasets in which that variable appeared. If “IN ALL” = “Y” then “NUM IN” will equal the total number of datasets being compared and will be less than that number if “IN ALL” = “N”.
- “SAME TYPE”: this is an indicator variable that will either be “Y” or “N”. The “Y” represents variables whose type is always the same *in datasets in which the variable appears*. “N” will be denoted otherwise and all “N” values are colored in red. A variable does NOT have to appear in every dataset to have a “Y” value.
- “SAME LEN”: this is an indicator variable that will either be “Y” or “N”. The “Y” represents variables whose length is always the same *in datasets in which the variable appears*. “N” will be denoted otherwise and all “N” values are colored in red. A variable does NOT have to appear in every dataset to have a “Y” value.
- “MIN LN”: the minimum length of that variable across all datasets in which it appeared. This is useful for assessing potential truncation that may results when combining this variable from all datasets.

- "MAX LN": the maximum length of that variable across all datasets in which it appeared. This is useful for assessing potential truncation that may results when combining this variable from all datasets.

COMPARISONS AT THE VALUE LEVEL

If `compareValues` = 1, the comparison report will include a single table that lists each variable that appears in at least one dataset, and each value of that variable that appears in at least one dataset, up to the maximum number of values specified by the `maxNumLevels` parameter. The fields that appear in the this report are:

- Variable name: the name of the variable whose values are being compared, followed by the number of unique values (i.e., levels) that are being displayed. The variable name will only appear with the first value to avoid redundancy.
- Value: the actual value of the variable being compared across datasets. If `ignoreCase` = 1, then all values will be capitalized; otherwise, they will appear in their original case.
- Count: the frequency with which each value appears in each dataset being compared.
- Percentage: the proportion of the time (%) with which each value appears in each dataset being compared. Percentages below 1% (but not exactly 0) will appear in blue, and if 100% of the observations for a particular dataset have that value, the 100% will be colored red.
- "IN ALL": this is an indicator variable that will either be "Y" or "N". The "Y" represents values of variables that appear in each dataset, regardless of its frequency. "N" will be denoted otherwise and all "N" values are colored in red.
- "NUM IN": represents the number of datasets in which that value of that variable appeared, regardless of its frequency. If "IN ALL" = "Y" then "NUM IN" will equal the total number of datasets being compared and will be less than that number if "IN ALL" = "N".

Table #	Level of comparison	Table description	Produced when	Modifiers
1	Dataset	Introductory table that provides the name of the original dataset included in the comparison, a shortened (coded) dataset name (e.g., DS1-DS*n*), and the number of observations and variables in each dataset.	All reports	None
2	Variable	Provides an alphabetical listing of each variable that appears in at least one of the datasets being compared. Compares the type ($=characters, #=numeric, dt=date/time), length, and number of levels for each variable. Provides variables that summarize the comparison of these characteristics across all datasets.	`compareVars=1`	None

3	Variable	Similar to table 2, but restricts to variables that appear in every dataset and that have the same type and length. Variables in this table would be combined easily across datasets.	`compareVars=1`
			`only1VarList=0`
				None

4	Variable	Similar to table 2, but restricts to variables that appear in every dataset and that have different types (e.g., character variable in DS1-DS4, but numeric in DS5-DS8).	`compareVars=1`
			`only1VarList=0`
				None

5	Variable	Similar to table 2, but restricts to variables that do not appear in every dataset (but do appear in more than one). This could be important for preventing data truncation when combining datasets.	`compareVars=1`
			`only1VarList=0`
				None

6	Variable	Similar to table 3, but only includes variables that do not appear in every dataset (but do appear in more than one). Variables still have the same name, type, and length.	`compareVars=1`
			`only1VarList=0`
				None

7	Variable	Similar to table 4, but only includes variables that do not appear in every dataset (but do appear in more than one). Same-named variables differ on their type across datasets.	`compareVars=1`
			`only1VarList=0`
				None

8	Variable	Similar to table 5, but only includes variables that do not appear in every dataset (but do appear in more than one). Same-named variables have the same type, but different lengths.	`compareVars=1`
			`only1VarList=0`
				None

9	Variable	Similar to table 2, but restricts to variables that appear in only one dataset. No summary comparisons across datasets are made.	`compareVars=1`
			`only1VarList=0`
				None

Table #	Level of comparison	Table description	Produced when	Modifiers
10	Variable	Provides an alphabetical listing of each variable that appears in at least one of the datasets being compared. Compares the missingness across all datasets in which the variable appears. If the variable does not appear in a dataset, "---" is listed for the frequency and "." for the percent missingness. *Note*: only true missing values are considered missing, actual values that may represent a missing response (e.g., -99) are considered valid values since these codes may differ depending on the variable and dataset.	All reports	`pctMissOnly`
11	Variable	Provides an alphabetical listing of selected* numeric variables that appears in at least one of the datasets being compared. Compares the min and max values across all datasets. If the variable does not appear in a dataset, "---" is listed for the min and max. Provides variables that summarize the comparison of these characteristics across all datasets.	All reports	`maxNumLevels`
11	Variable	Provides an alphabetical listing of selected* date variables that appears in at least one of the datasets being compared. Compares the min and max formatted date values across all datasets. If the variable does not appear in a dataset, "---" is listed for the min and max. Provides variables that summarize the comparison of these characteristics across all datasets.	All reports	`maxNumLevels`

12	Value	Provides an alphabetical listing of each variable that appears in at least one of the datasets being compared. For each variable, then lists the unique values entered for that variable across all datasets. Can order values alphabetically or based on frequency. Compares the frequency and proportion of the time each value appeared. Additional options are described in the text.	`compareValues` = 1	`maxCharLen`
				`maxNumLevels`
				`runExceedMaxLevels`
				`valueOrderFreq`

Table 2. Basic summary of tables produced in the report generated by the compareMultipleDS macro

**The determination of whether numeric and date variables are compared in the variable-level numeric and date tables comparing only extreme values (min/max) or the value-level table comparing the frequency and proportion of unique values is described in the paper's text.*

APPLICATION OF THE MACRO TO PUBLICLY-AVAILABLE DATA

The purpose of this section is to illustrate two basic examples of how the "**compareMultipleDS**" macro is intended to be used. As an epidemiologist and public health researcher, my case studies stem from this arena; however, are easily translated into virtually any other scenario is which the ultimate goal is to implement a comparison of multiple datasets. The examples illustrate variation in user input of macro parameter values that control the comprehensiveness of the comparison and appearance of the comparison report.

CASE STUDY #1: MORTALITY DATA FROM THE NATIONAL CENTER FOR HEALTH STATISTICS

The National Center for Health Statistics (NCHS) is the principal health statistics agency in the United States. Their mission is to provide statistical information that will guide actions and the health of population they serve. NCHS provides online portals that offer interactive online data access tools and downloadable public use data files (http://www.cdc.gov/nchs/data_access/Vitalstatsonline.htm). For this case study, we are interested in conducting a series of analyses on trends in the causes of death in the United States from 2000-2005. However, vital records such as death certificates, from which these mortality files are generated, have undergone state and national-level changes during our analytic period, and our first task is to understand those differences prior to combining the datasets for analysis. The datasets are available for download as annual zipped text files and are supplemented with annual data dictionaries (file documentation) from NCHS. The documentation contains the name of each variable, the position of each variable in the text file, a description of the variable, and if the variable contains distinct levels, a listing of each code and what the code is intended to represent (e.g., W for marital status represents "widowed"). This can be used to develop SAS code to import each annual file, develop variable names, labels, and value formats. By comparing multiple data dictionaries, one can also begin to understand differences in the datasets over time; however, this process can be arduous and time consuming. Although the macro described in this paper is not a replacement for a detailed review all documentation, it can help to simplify the process and summarize some of the important variation in the mortality datasets over time.

Since this paper is not intended to discuss the creation of SAS code to import the text files into SAS, we will work from SAS files already created from the raw NCHS public-use mortality data files. For this, the author thanks Jean Roth and the National Bureau of Economic Research for providing free access to SAS programs used to import the raw data files into SAS, and well as the SAS files themselves. They are available from http://www.nber.org/data/vital-statistics-mortality-data-multiple-cause-of-death.html. The mort*yyyy*.sas7bdat.zip (where *yyyy* = 2000-2005) were downloaded and unzipped into a common folder.

The "**compareMultipleDS**" macro was called using the code below:

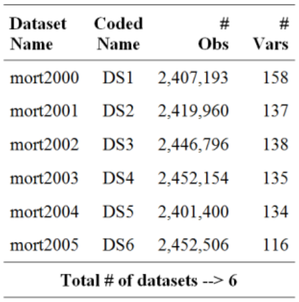
```
%compareMultipleDS(
  dsNames= %str(mort2000 mort2001 mort2002 mort2003 mort2004 mort2005), libRef= nchs,
  fileLocation= F:\DATA\NCHS\Multiple Cause Mortality\, fileNamePrefix= compareMORTvars,
  compareVars=1, only1VarList=0, compareValues=1, ignoreCase=1, maxCharLen=40,
  maxNumLevels=30, runExceedMaxLevels=1, pctMissOnly=0, pctValueOnly=0, valueOrderFreq=1,
  mySpacing=15, myPadding=3, myFontSize=1, fileType=rtf, redirectLog=1)
```

This macro recall is indicative of a user that wants to generate all available assessments (evidenced by `compareVars=1, only1VarList=0, compareValues=1`). The final report will be called **compareMORTvars_*yyyy-mm-dd*.rtf**.

So what can we learn from the report? Figures 2-6 are excerpts from various reports produced by the macro. Figure 3 begins to identify variables that were either added to mortality files (e.g., AUTOPSY [autopsy indicator], BRACE [bridged race indicator], CMSARES [consolidated metropolitan statistical area], all added in 2003), or variables that were no longer collected after 2002 (e.g., DIVSTOC [division/state code of occurrence], DIVSTRES [division/state code of residence]). The city of residence variable (CITYRES) was increased from 3-5 characters in 2003, which is important if attempting to combine this variable across all datasets. The full report identifies a host of other changes that reflect a new version of the death certificate appearing in states across the U.S. Figure 4 documents the missingness present for each variable over time. The excerpt presented reveals that the ACTIVITY variable, which captures the activity that the decedent was engaging in at the time of death, may be present is each dataset, but is missing in >93% of death records, meaning it may be of limited utility in analysis. In contrast, AGE is never "missing"; however, a subsequent investigation of the table represented in Figure 5 reveals that a missing code was used as opposed to a missing value. It also reveals that the missing code changed from 999 in 2000-02 to 9999 in 2003-05. That is why these reports must be considered in concert and with subsequent referencing of data dictionaries that may be provided. Again, this macro simply facilitates an initial understanding of that data contained in multiple datasets simultaneously.

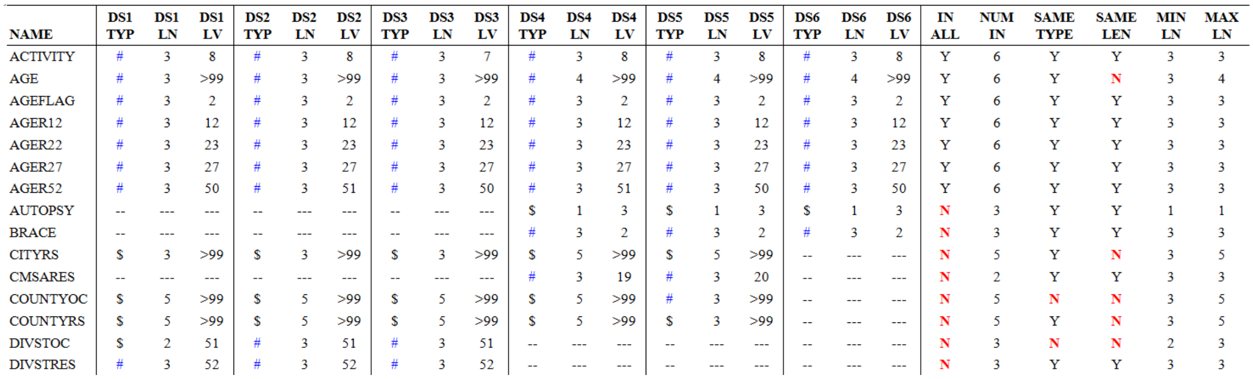
Figure 6 contains the value-level and most informative table produced by the macro. As dictated by the `valueOrderFreq` parameter, the values of all variables are presented in descending order of their frequency. Since missing values are included, this table provides a good overview of "pure missingness" and the relative frequencies of each code. So, not only can this report reflect new codes/values that emerge and ones that may be terminated, but can also permit an expedient way of determining if population characteristics change across dataset (which in this case means over time). For example, the AGER12 variable represents the decedent's age recoded into one of 12 categories (confirmed by reviewing the file documentation). Looking at the percentages of each value within each dataset gives the user a good sense that there have not been any shifts in the age distribution of the population over time, according to these categories, and assuming their definitions have not changed over time.

This report, comparing 6 datasets with approximately 2.5 million observations and 140 variables each, with value-level comparisons being made took approximately 8 minutes. Restriction to only variable-level comparisons reduces the time taken to run the report to approximately 1 minute.



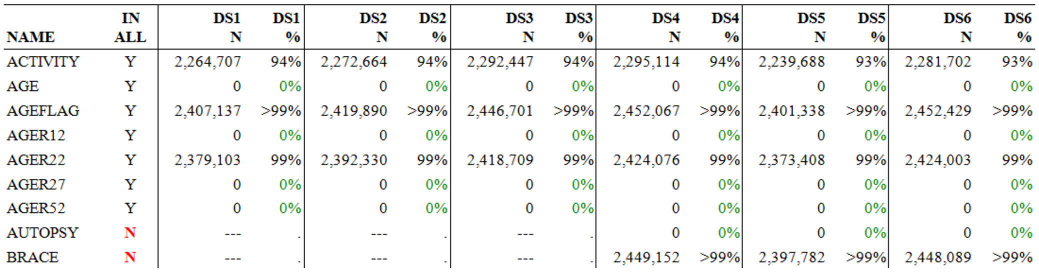
Dataset Name	Coded Name	# Obs	# Vars
mort2000	DS1	2,407,193	158
mort2001	DS2	2,419,960	137
mort2002	DS3	2,446,796	138
mort2003	DS4	2,452,154	135
mort2004	DS5	2,401,400	134
mort2005	DS6	2,452,506	116
Total # of datasets --> 6			

Figure 2. Comparison report: dataset-level comparison table



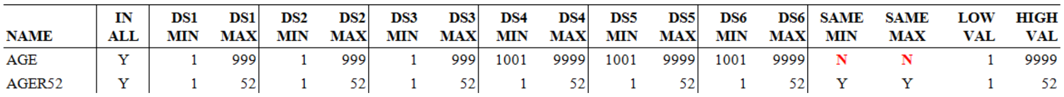
NAME	DS1 TYP	DS1 LN	DS1 LV	DS2 TYP	DS2 LN	DS2 LV	DS3 TYP	DS3 LN	DS3 LV	DS4 TYP	DS4 LN	DS4 LV	DS5 TYP	DS5 LN	DS5 LV	DS6 TYP	DS6 LN	DS6 LV	IN ALL	NUM IN	SAME TYPE	SAME LEN	MIN LN	MAX LN
ACTIVITY	#	3	8	#	3	8	#	3	7	#	3	8	#	3	8	#	3	8	Y	6	Y	Y	3	3
AGE	#	3	>99	#	3	>99	#	3	>99	#	4	>99	#	4	>99	#	4	>99	Y	6	Y	N	3	4
AGEFLAG	#	3	2	#	3	2	#	3	2	#	3	2	#	3	2	#	3	2	Y	6	Y	Y	3	3
AGER12	#	3	12	#	3	12	#	3	12	#	3	12	#	3	12	#	3	12	Y	6	Y	Y	3	3
AGER22	#	3	23	#	3	23	#	3	23	#	3	23	#	3	23	#	3	23	Y	6	Y	Y	3	3
AGER27	#	3	27	#	3	27	#	3	27	#	3	27	#	3	27	#	3	27	Y	6	Y	Y	3	3
AGER52	#	3	50	#	3	51	#	3	50	#	3	51	#	3	50	#	3	50	Y	6	Y	Y	3	3
AUTOPSY	--	---	---	--	---	---	--	---	---	$	1	3	$	1	3	$	1	3	N	3	Y	Y	1	1
BRACE	--	---	---	--	---	---	--	---	---	#	3	2	#	3	2	#	3	2	N	3	Y	Y	3	3
CITYRS	$	3	>99	$	3	>99	$	3	>99	$	5	>99	$	5	>99	--	---	---	N	5	Y	N	3	5
CMSARES	--	---	---	--	---	---	--	---	---	#	3	19	#	3	20	--	---	---	N	2	Y	Y	3	3
COUNTYOC	$	5	>99	$	5	>99	$	5	>99	$	5	>99	#	3	>99	--	---	---	N	5	N	N	3	5
COUNTYRS	$	5	>99	$	5	>99	$	5	>99	$	5	>99	$	3	>99	--	---	---	N	5	Y	N	3	5
DIVSTOC	$	2	51	#	3	51	#	3	51	--	---	---	--	---	---	--	---	---	N	3	N	N	2	3
DIVSTRES	#	3	52	#	3	52	#	3	52	--	---	---	--	---	---	--	---	---	N	3	Y	Y	3	3

Figure 3. Comparison report excerpt: variable-level characteristics table



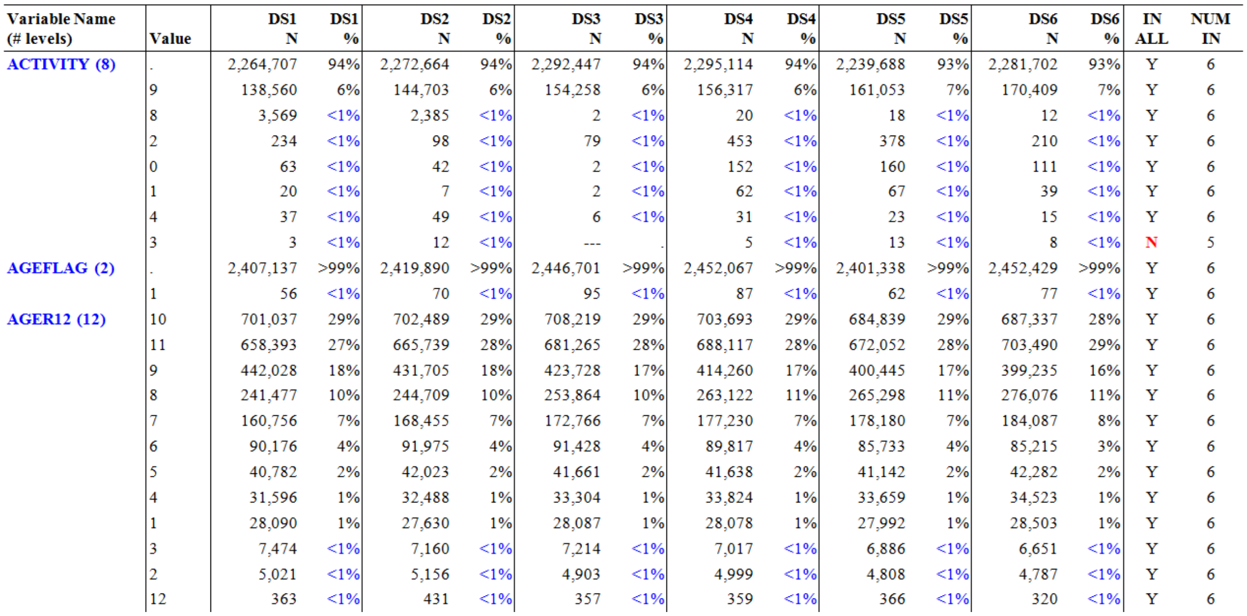
NAME	IN ALL	DS1 N	DS1 %	DS2 N	DS2 %	DS3 N	DS3 %	DS4 N	DS4 %	DS5 N	DS5 %	DS6 N	DS6 %
ACTIVITY	Y	2,264,707	94%	2,272,664	94%	2,292,447	94%	2,295,114	94%	2,239,688	93%	2,281,702	93%
AGE	Y	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
AGEFLAG	Y	2,407,137	>99%	2,419,890	>99%	2,446,701	>99%	2,452,067	>99%	2,401,338	>99%	2,452,429	>99%
AGER12	Y	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
AGER22	Y	2,379,103	99%	2,392,330	99%	2,418,709	99%	2,424,076	99%	2,373,408	99%	2,424,003	99%
AGER27	Y	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
AGER52	Y	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
AUTOPSY	N	---	.	---	.	---	.	0	0%	0	0%	0	0%
BRACE	N	---	.	---	.	---	.	2,449,152	>99%	2,397,782	>99%	2,448,089	>99%

Figure 4. Comparison report excerpt: variable-level missingness table



NAME	IN ALL	DS1 MIN	DS1 MAX	DS2 MIN	DS2 MAX	DS3 MIN	DS3 MAX	DS4 MIN	DS4 MAX	DS5 MIN	DS5 MAX	DS6 MIN	DS6 MAX	SAME MIN	SAME MAX	LOW VAL	HIGH VAL
AGE	Y	1	999	1	999	1	999	1001	9999	1001	9999	1001	9999	N	N	1	9999
AGER52	Y	1	52	1	52	1	52	1	52	1	52	1	52	Y	Y	1	52

Figure 5. Comparison report excerpt: variable-level table for numeric variable with a number of levels that exceeds `maxNumLevels`



Variable Name (# levels)	Value	DS1 N	DS1 %	DS2 N	DS2 %	DS3 N	DS3 %	DS4 N	DS4 %	DS5 N	DS5 %	DS6 N	DS6 %	IN ALL	NUM IN
ACTIVITY (8)	.	2,264,707	94%	2,272,664	94%	2,292,447	94%	2,295,114	94%	2,239,688	93%	2,281,702	93%	Y	6
	9	138,560	6%	144,703	6%	154,258	6%	156,317	6%	161,053	7%	170,409	7%	Y	6
	8	3,569	<1%	2,385	<1%	2	<1%	20	<1%	18	<1%	12	<1%	Y	6
	2	234	<1%	98	<1%	79	<1%	453	<1%	378	<1%	210	<1%	Y	6
	0	63	<1%	42	<1%	2	<1%	152	<1%	160	<1%	111	<1%	Y	6
	1	20	<1%	7	<1%	2	<1%	62	<1%	67	<1%	39	<1%	Y	6
	4	37	<1%	49	<1%	6	<1%	31	<1%	23	<1%	15	<1%	Y	6
	3	3	<1%	12	<1%	---	.	5	<1%	13	<1%	8	<1%	N	5
AGEFLAG (2)	.	2,407,137	>99%	2,419,890	>99%	2,446,701	>99%	2,452,067	>99%	2,401,338	>99%	2,452,429	>99%	Y	6
	1	56	<1%	70	<1%	95	<1%	87	<1%	62	<1%	77	<1%	Y	6
AGER12 (12)	10	701,037	29%	702,489	29%	708,219	29%	703,693	29%	684,839	29%	687,337	28%	Y	6
	11	658,393	27%	665,739	28%	681,265	28%	688,117	28%	672,052	28%	703,490	29%	Y	6
	9	442,028	18%	431,705	18%	423,728	17%	414,260	17%	400,445	17%	399,235	16%	Y	6
	8	241,477	10%	244,709	10%	253,864	10%	263,122	11%	265,298	11%	276,076	11%	Y	6
	7	160,756	7%	168,455	7%	172,766	7%	177,230	7%	178,180	7%	184,087	8%	Y	6
	6	90,176	4%	91,975	4%	91,428	4%	89,817	4%	85,733	4%	85,215	3%	Y	6
	5	40,782	2%	42,023	2%	41,661	2%	41,638	2%	41,142	2%	42,282	2%	Y	6
	4	31,596	1%	32,488	1%	33,304	1%	33,824	1%	33,659	1%	34,523	1%	Y	6
	1	28,090	1%	27,630	1%	28,087	1%	28,078	1%	27,992	1%	28,503	1%	Y	6
	3	7,474	<1%	7,160	<1%	7,214	<1%	7,017	<1%	6,886	<1%	6,651	<1%	Y	6
	2	5,021	<1%	5,156	<1%	4,903	<1%	4,999	<1%	4,808	<1%	4,787	<1%	Y	6
	12	363	<1%	431	<1%	357	<1%	359	<1%	366	<1%	320	<1%	Y	6

Figure 6. Comparison report excerpt: value-level table for all variables with a number of levels that is less than or equal to `maxNumLevels`

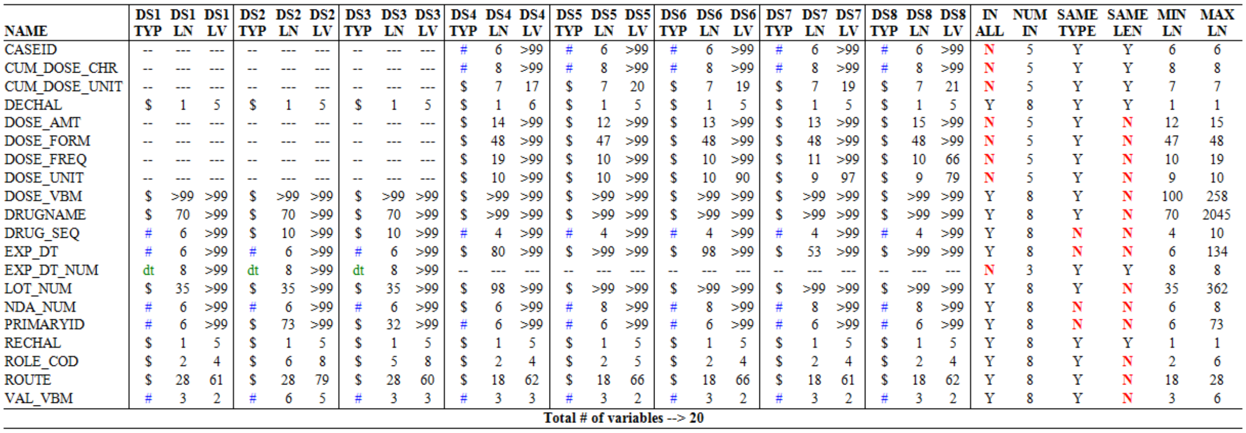
CASE STUDY #2: DATA FROM THE FDA ADVERSE EVENT REPORTING SYSTEM (FAERS)

In the U.S., the Food and Drug Administration is the federal agency responsible for regulation and oversight of food safety, tobacco products, dietary supplements, medications, vaccines, and a host of other exposures that impact the public's health. The FDA hosts an FDA Adverse Event Reporting System (FAERS), a database that tracks information reported voluntarily by healthcare professionals and consumers on adverse events and medication error reports submitted to the FDA, at http://www.fda.gov/Drugs/GuidanceComplianceRegulatoryInformation/Surveillance/AdverseDrugEffects/. FAERS databases are made available to the public, for free, in the form of quarterly relational databases consisting of de-identified, individual case safety reports (the databases included in any given quarter are: DEMO*yyQq*, DRUG*yyQq*, INDI*yyQq*, OUTC*yyQq*, REAC*yyQq*, RPSR*yyQq*, and THER*yyQq*). Each contains different types of information on demographics, mediciations, outcomes, etc. Similar to our previous case study, the FAERS databases are supplemented with data dictionaries that describe the structure and coding within each of the relational tables. After downloading 8 quarters worth of FAERS data from 2012-2013, the "**compareMultipleDS**" macro was used to compare each of the relational database components across all 8 quarters.

The "**compareMultipleDS**" macro was called using the code below:

```
%compareMultipleDS(
  dsNames= %str(drug2012q1 drug2012q2 drug2012q3 drug2012q4 drug2013q1 drug2013q2 drug2013q3
  drug2013q4), libRef= faers, fileLocation= F:\DATA\FAERS\, fileNamePrefix= compareFAERSdrug,
  compareVars=1, only1VarList=0, compareValues=1, ignoreCase=0, maxCharLen=30,
  maxNumLevels=40, runExceedMaxLevels=1, pctMissOnly=1, pctValueOnly=1, valueOrderFreq=1,
  mySpacing=6, myPadding=1, myFontSize=0.8, fileType=rtf, redirectLog=0)
```

For the purposes of presentation, which is primarily to draw distinctions between this and the previous report based on user-specified macro parameters, we focus only on the very simple DRUG datasets. First, since more datasets were compared in this case study (8) versus the last (6), the parameters controlling the CELLSPACING, CELLPADDING, and FONTSIZE style options were lowered to ensure all output for a variable appeared on one line (**Figure 7**). This (tweaking style options) may, at times, be an iterative process based on an initial run of the macro.

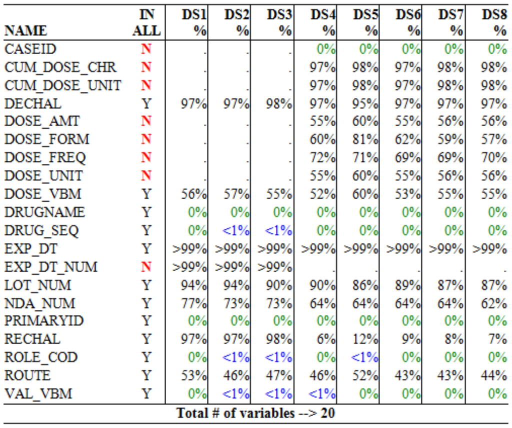


NAME	DS1 TYP	DS1 LN	DS1 LV	DS2 TYP	DS2 LN	DS2 LV	DS3 TYP	DS3 LN	DS3 LV	DS4 TYP	DS4 LN	DS4 LV	DS5 TYP	DS5 LN	DS5 LV	DS6 TYP	DS6 LN	DS6 LV	DS7 TYP	DS7 LN	DS7 LV	DS8 TYP	DS8 LN	DS8 LV	IN ALL	NUM IN	SAME TYPE	SAME LEN	MIN LN	MAX LN
CASEID	--	---	---	--	---	---	--	---	---	#	6	>99	#	6	>99	#	6	>99	#	6	>99	#	6	>99	N	5	Y	Y	6	6
CUM_DOSE_CHR	--	---	---	--	---	---	--	---	---	#	8	>99	#	8	>99	#	8	>99	#	8	>99	#	8	>99	N	5	Y	Y	8	8
CUM_DOSE_UNIT	--	---	---	--	---	---	--	---	---	$	7	17	$	7	20	$	7	19	$	7	19	$	7	21	N	5	Y	Y	7	7
DECHAL	$	1	5	$	1	5	$	1	5	$	1	6	$	1	5	$	1	5	$	1	5	$	1	5	Y	8	Y	Y	1	1
DOSE_AMT	--	---	---	--	---	---	--	---	---	$	14	>99	$	12	>99	$	13	>99	$	13	>99	$	15	>99	N	5	Y	N	12	15
DOSE_FORM	--	---	---	--	---	---	--	---	---	$	48	>99	$	47	>99	$	48	>99	$	48	>99	$	48	>99	N	5	Y	N	47	48
DOSE_FREQ	--	---	---	--	---	---	--	---	---	$	19	>99	$	10	>99	$	10	>99	$	11	>99	$	10	66	N	5	Y	N	10	19
DOSE_UNIT	--	---	---	--	---	---	--	---	---	$	10	>99	$	10	>99	$	10	90	$	9	97	$	9	79	N	5	Y	N	9	10
DOSE_VBM	$	>99	>99	$	>99	>99	$	>99	>99	$	>99	>99	$	>99	>99	$	>99	>99	$	>99	>99	$	>99	>99	Y	8	Y	N	100	258
DRUGNAME	$	70	>99	$	70	>99	$	70	>99	$	>99	>99	$	>99	>99	$	>99	>99	$	>99	>99	$	>99	>99	Y	8	Y	N	70	2045
DRUG_SEQ	#	6	>99	$	10	>99	$	10	>99	#	4	>99	#	4	>99	#	4	>99	#	4	>99	#	4	>99	Y	8	N	N	4	10
EXP_DT	#	6	>99	#	6	>99	#	6	>99	$	80	>99	$	>99	>99	$	98	>99	$	53	>99	$	>99	>99	Y	8	N	N	6	134
EXP_DT_NUM	dt	8	>99	dt	8	>99	dt	8	>99	--	---	---	--	---	---	--	---	---	--	---	---	--	---	---	N	3	Y	Y	8	8
LOT_NUM	$	35	>99	$	35	>99	$	35	>99	$	98	>99	$	>99	>99	$	>99	>99	$	>99	>99	$	>99	>99	Y	8	Y	N	35	362
NDA_NUM	#	6	>99	#	6	>99	#	6	>99	$	6	>99	#	8	>99	#	8	>99	#	8	>99	#	8	>99	Y	8	N	N	6	8
PRIMARYID	#	6	>99	$	73	>99	$	32	>99	#	6	>99	#	6	>99	#	6	>99	#	6	>99	#	6	>99	Y	8	N	N	6	73
RECHAL	$	1	5	$	1	5	$	1	5	$	1	5	$	1	5	$	1	5	$	1	5	$	1	5	Y	8	Y	Y	1	1
ROLE_COD	$	2	4	$	6	8	$	5	8	$	2	4	$	2	5	$	2	4	$	2	4	$	2	4	Y	8	Y	N	2	6
ROUTE	$	28	61	$	28	79	$	28	60	$	18	62	$	18	66	$	18	66	$	18	61	$	18	62	Y	8	Y	N	18	28
VAL_VBM	#	3	2	#	6	5	#	3	3	#	3	3	#	3	2	#	3	2	#	3	2	#	3	2	Y	8	Y	N	3	6

Total # of variables --> 20

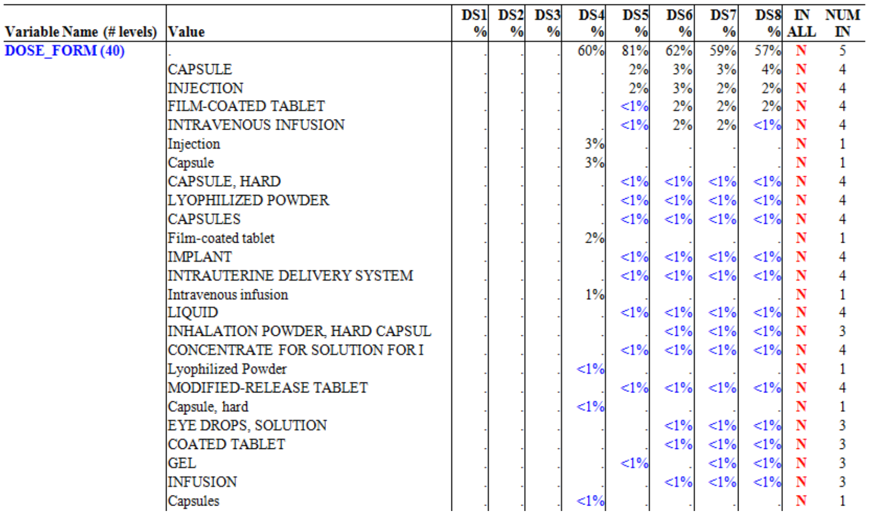
Figure 7. Comparison report: variable-level characteristics table

Furthermore, in both **Figure 8** (missingness table) and **Figure 9** (value-level comparison table), the specification of `pctMissOnly=1` and `pctValueOnly=1` result in only percentages (and not frequencies) being displayed in the table. This can be done to conserve space or simplify the output. Lastly, although only a few values of the DOSE_FORM variable are presented, we can immediately see the impact of the `ignoreCase=0` option. Notice in **Figure 9** that the values “CAPSULE” and “Capsule” are treated as separate values. Depending on your objective in comparing the datasets, it may or may not be desirable to ignore case. Also note, however, that “Capsule, hard” and “Capsules” also appear as values. The user may ultimately want to combine all of these values into one, and this macro can help identify some (but not all) of these issues prior to combining and analyzing datasets.



NAME	IN ALL	DS1 %	DS2 %	DS3 %	DS4 %	DS5 %	DS6 %	DS7 %	DS8 %
CASEID	N	.	.	.	0%	0%	0%	0%	0%
CUM_DOSE_CHR	N	.	.	.	97%	98%	97%	98%	98%
CUM_DOSE_UNIT	N	.	.	.	97%	98%	97%	98%	98%
DECHAL	Y	97%	97%	98%	97%	95%	97%	97%	97%
DOSE_AMT	N	.	.	.	55%	60%	55%	56%	56%
DOSE_FORM	N	.	.	.	60%	81%	62%	59%	57%
DOSE_FREQ	N	.	.	.	72%	71%	69%	69%	70%
DOSE_UNIT	N	.	.	.	55%	60%	55%	56%	56%
DOSE_VBM	Y	56%	57%	55%	52%	60%	53%	55%	55%
DRUGNAME	Y	0%	0%	0%	0%	0%	0%	0%	0%
DRUG_SEQ	Y	0%	<1%	<1%	0%	0%	0%	0%	0%
EXP_DT	Y	>99%	>99%	>99%	>99%	>99%	>99%	>99%	>99%
EXP_DT_NUM	N	>99%	>99%	>99%	.	.	.	.	.
LOT_NUM	Y	94%	94%	90%	90%	86%	89%	87%	87%
NDA_NUM	Y	77%	73%	73%	64%	64%	64%	64%	62%
PRIMARYID	Y	0%	0%	0%	0%	0%	0%	0%	0%
RECHAL	Y	97%	97%	98%	6%	12%	9%	8%	7%
ROLE_COD	Y	0%	<1%	<1%	0%	<1%	0%	0%	0%
ROUTE	Y	53%	46%	47%	46%	52%	43%	43%	44%
VAL_VBM	Y	0%	<1%	<1%	<1%	0%	0%	0%	0%
Total # of variables --> 20									

Figure 8. Comparison report: variable-level missingness table



Variable Name (# levels)	Value	DS1 %	DS2 %	DS3 %	DS4 %	DS5 %	DS6 %	DS7 %	DS8 %	IN ALL	NUM IN
DOSE_FORM (40)	.	.	.	.	60%	81%	62%	59%	57%	N	5
	CAPSULE	.	.	.	.	2%	3%	3%	4%	N	4
	INJECTION	.	.	.	.	2%	3%	2%	2%	N	4
	FILM-COATED TABLET	.	.	.	.	<1%	2%	2%	2%	N	4
	INTRAVENOUS INFUSION	.	.	.	.	<1%	2%	2%	<1%	N	4
	Injection	.	.	.	3%	.	.	.	.	N	1
	Capsule	.	.	.	3%	.	.	.	.	N	1
	CAPSULE, HARD	.	.	.	.	<1%	<1%	<1%	<1%	N	4
	LYOPHILIZED POWDER	.	.	.	.	<1%	<1%	<1%	<1%	N	4
	CAPSULES	.	.	.	.	<1%	<1%	<1%	<1%	N	4
	Film-coated tablet	.	.	.	2%	.	.	.	.	N	1
	IMPLANT	.	.	.	.	<1%	<1%	<1%	<1%	N	4
	INTRAUTERINE DELIVERY SYSTEM	.	.	.	.	<1%	<1%	<1%	<1%	N	4
	Intravenous infusion	.	.	.	1%	.	.	.	.	N	1
	LIQUID	.	.	.	.	<1%	<1%	<1%	<1%	N	4
	INHALATION POWDER, HARD CAPSUL	.	.	.	.	.	<1%	<1%	<1%	N	3
	CONCENTRATE FOR SOLUTION FOR I	.	.	.	.	<1%	<1%	<1%	<1%	N	4
	Lyophilized Powder	.	.	.	<1%	.	.	.	.	N	1
	MODIFIED-RELEASE TABLET	.	.	.	.	<1%	<1%	<1%	<1%	N	4
	Capsule, hard	.	.	.	<1%	.	.	.	.	N	1
	EYE DROPS, SOLUTION	.	.	.	.	.	<1%	<1%	<1%	N	3
	COATED TABLET	.	.	.	.	.	<1%	<1%	<1%	N	3
	GEL	.	.	.	.	<1%	.	<1%	<1%	N	3
	INFUSION	.	.	.	.	.	<1%	<1%	<1%	N	3
	Capsules	.	.	.	<1%	.	.	.	.	N	1

Figure 9. Comparison report excerpt: value-level table for all variables with a number of levels that is less than or equal to `maxNumLevels`

CONCLUSION

When it comes to data, first impressions count. Depending on those first impressions, you may be more adept and savvy regarding how best to get to “Know Thy Data”. As someone who works with a myriad of databases that often require combining over time or across multiple versions, and someone who has heard the complaints and difficulties of others who do the same, I decided to develop a simple tool that, to this author’s knowledge, does not exist in this capacity, to facilitate the initial understanding of multiple databases simultaneously. Despite the existence of a powerful COMAPRE procedure and a host of user-developed macros that cover dataset comparison, this macro serves a unique niche that includes not only dataset- and variable-level comparison, but also missingness reports and value-level investigation in one fell swoop. The macro was designed to allow user-defined refinements to the mechanics of the comparison, as well as the appearance of the comparison report, but without overwhelming the user with an unmanageable number of input parameters.

REFERENCES

Katare, Anurag. 2011. "'Compare Me' a SAS® Datasets Comparison Tool." Proceedings of the MidWest SAS® Users Group: Paper 76505-2011. Available at http://www.mwsug.org/proceedings/2011/appdev/MWSUG-2011-AD09.pdf.

Krishnan, Krish. 2013. "Macro utility to compare multiple SAS® datasets." Proceedings of the South Central SAS Users Group. Available at http://www.scsug.org/wp-content/uploads/2013/11/Macro-utility-to-compare-multiple-SAS-data-sets-Krish-Krishnan.pdf.

Kuligowski, Andrew T. 2013. "Know Thy Data: Techniques for Data Exploration." Proceedings of the SAS® Global Forum: Paper 145-2013. Available at http://support.sas.com/resources/papers/proceedings13/145-2013.pdf.

Wang, Hui. 2014. "A SAS® Macro Tool for Visualizing Data Comparison Results in an Intuitive Way." Proceedings of the Pharmaceutical Industry SAS® Users Group: Paper CC22. Available at http://www.pharmasug.org/proceedings/2014/CC/PharmaSUG-2014-CC22.pdf.

ACKNOWLEDGMENTS

The author would like to acknowledge that this macro was developed initially after working with data to support the "*Healthy Start Initiatives-Eliminating Racial/Ethnic Disparities*" project, supported by grant number H49MC12793 from the Health Resources and Services Administration (HRSA). The examples presented within this paper are not based on data from this project since illustration was deemed to be better served by use of publicly-available data that could be downloaded by users wishing to test the macro's functionality. The content is solely the responsibility of the author and does not necessarily represent the official views or approved approaches of the Health Resources and Services Administration, ReachUp, Inc., or the Baylor College of Medicine.

The author would also like to thank my friend and colleague, **Jean Paul Tanner, MPH** at the University of South Florida College of Public Health, for his insights regarding the design of the macro's final report and the inclusion of selected macro parameters for improved user control of the comprehensiveness of the report, for testing the functionality of the macro on a myriad of real-world databases, and for editorial review of this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jason L. Salemi, PhD, MPH
Assistant Professor of Epidemiology and Biostatistics
Department of Family and Community Medicine
Baylor College of Medicine
Houston, TX 77098
Jason.Salemi@bcm.edu
http://JasonSalemi.strikingly.com