Paper No. 007-2008

Start Improving Your Data Quality with %FreqAll

Toby Dunn, Dunn Consulting

ABSTRACT

As the age of business computing matures, the adage "garbage in garbage out" is truer than ever before. A 2002 global study of businesses by Price Waterhouse Coopers concluded that poor data quality cost over $1.4 billion dollars and caused roughly one-third of the companies to either delay or scrap a project. In today's fast-paced business world, there has never been a greater need for closer attention to the quality of data. In modern global commerce, questions need to be answered immediately and with no mistakes.

The first step in understanding and improving the quality of data requires knowledge of the composition of the tables within a database. What follows is an effective yet simple macro called %FreqAll, designed to deliver a fast and easy first glance at the data values and quality.

INTRODUCTION

How good is your data quality? Unless you already have gone down the path of developing and implementing a Data Quality Management Plan (DQMP), I can guarantee your data isn't nearly as good as you would like to believe it is. The problems with data can be as simple as multiple spellings of the same value or as complex as determining if there is a sufficient time period between the birth date of child and their parents. Regardless of the type or how complex the error, it is an error none-the-less and begs the question, whether that business decision or study conclusion based on the data was a wise choice or even accurate. (The previous sentence used to be grammatically incomplete.)

Data Profiling was developed in order to find these data errors and correct them. Data Profiling relies on five types of analysis to determine the accuracy of the data: column property, structural, simple data rule, complex data rule, and value rule analysis. The culmination of performing this analysis is to create a report of the errors in the data. Although performing this full analysis is outside the scope of this paper and requires substantial time and resources, a frequency analysis is one of the simplest to perform and is the focus of this paper.

In the world of Data Profiling, frequency analysis falls mainly under the Column Property analysis and to a smaller extent the Value Rule analysis. Frequency analysis is used to gather basic information about what actually exists in the data, rather than relying on often outdated and inaccurate metadata. It shows the number of distinct values and the maximum number of occurrences that any one value can be expected to have. This type of analysis shows the extreme values for a variable and allows the programmer to easily review the values for reasonableness. A frequency analysis exposes misspellings as well as valid alternative spellings for a value; this is useful not only in dealing with the data but also shows possible weaknesses in the data gathering and entry. Finally, it will show values that appear in data (both valid and invalid) but not in the metadata, as well as those values which are in the metadata but not in the actual data.

A frequency analysis is easy and cheap to execute and can have a huge impact on any project when used correctly. Using the output and exercising a little thought about the data allows the programmer to detect and clean up errors that wreak havoc with joins or merges, or even something as simple a summary report. There is no excuse for not knowing and cleaning your data, the consequences for not doing so far outweigh the cost of doing a little work upfront.

The %FreqAll Macro

The %FreqAll macro presented in this paper is a derivative of a macro created by Ron Fehd, a good friend and fellow SAS® programmer. Ron sent me the macro to review and I gave it a quick perusal until I could dedicate the time for a proper review. Later on I was reading a book on data profiling and it occurred to me that Ron's macro would be a perfect solution for solving some of the data problems presented in the book. While reviewing the macro I decided to make a few changes to the original code, which was written prior to SAS version 9. Also, I wanted more flexibility in the interface of the macro to increase usability. Finally, I added some error checking to give the user accurate and timely feed back for the most common problems. The result is the macro presented here.

The goal of the %FreqAll macro is to create a data set of a customized frequency so the programmer can later create a report that shows all the pertinent information that a programmer needs to analyze the values of the variables. The

macro is customized so that not only the frequency of a value is given but also the variable's name, type, length, and label. Since the output from this macro is a data set, the frequency report can be customized by the programmer each time the %FreqAll macro is called.

%FreqAll uses keyword parameters to utilize default values. They are ordered such that parameter values for input values come first and parameters affecting output come last. This methodology was used to ensure that the macro is user friendly. Allowing default values produces a set of values that will most often be used for certain parameters so that the user does not have to specify them when calling the macro If they wish to use one of the other valid values then all they need to do is specify them. Ordering the parameters in this allows the parameters to map to the mind of user, thus making them easier to remember.

The basic %FreqAll parameters are:

```
%Macro FreqAll( Lib = Work , Mem = , Vars = , ExcludeVars = ,
                LevelLimit = , NObs2View = , Format = Yes ,
                Order = Internal , DataOut = )
```

Lib

Lib represents the library name to be used for the input data set. The default value is WORK, but any valid library name can be used.

Mem

The name of any valid data set that resides in the directory specified in the Lib parameter.

Vars

A user supplied list of space separated valid variable names to be used in the frequency analysis.

Valid values:
All = Specifies both numeric and character variables
Numeric = Specifies all numeric variables
Character = Specifies all character variables

ExcludeVars

A space separated list of valid variables to exclude from the frequency analysis. This allows the use of _All_, _Numeric_, and _Character_ in the Vars parameter, and excludes certain variables that would otherwise be used if they were not specifically excluded. An example is to run the macro for all numeric variables but to exclude the BirthDate and Age variables from the analysis. The macro call would look like:

```
%Macro FreqAll( Mem         = MyData        ,
                Vars        = _Numeric_     ,
                ExcludeVars = BirthDate Age ,
                DataOut     = MyOutData      )
```

LevelLimit

This parameter allows the user to specify a numeric value of the maximum number of unique values that can appear for any one variable. When the limit is exceeded, the macro will truncate the variables frequency to the top and bottom number of unique values specified in the NObs2View parameter. This parameter was added since many

variables may have an extremely long output. In these cases it is not very useful or efficient to have the full frequency for these variables, as only the extreme values are considered.

NObs2View

This parameter specifies the number of Top and Bottom most unique values, when the total number of unique values for a variable is greater than or equal to the value specified in LevelLimit.

Format

Format specifies whether the frequency procedure will use a variable's associated format. The default is set to Yes, and valid values are Yes and No.

Order

The order determines how the frequency output for a variable will be reported. Valid values are Internal, Data, Formatted, and Freq. Internal is the default value for this parameter, ordering the values in their unformatted form as if proc sort had been utilized. Data will order the values according to their order in the data set, Formatted orders the values according to the variable formatted value, and finally, Freq orders the values by descending frequency counts.

DataOut

This parameter refers to the output data set which will contain the final custom frequency analysis.

Example 1.

```
%FreqAll( Lib         = SASHELP   ,
          Mem         = CLASS     ,
          Vars        = _ALL_     ,
          ExcludeVars = Name      ,
         DataOut     = MyFreq     )



Options NoByLine NoDate NoNumber ;

Title1 "DataSet - SASHELP.CLASS" ;
Title2 "Var #ByVal1" ;
Proc Print
 Data = MyFreq NoObs ;
 By Attribute NotSorted ;
 Var Val Frequency Percent Level ;
Run ;
```

```
       DataSet - SASHELP.CLASS
           Var Age  Num :8

Val    Frequency    Percent    Level

 11            2      10.53        1
 12            5      26.32        2
 13            3      15.79        3
 14            4      21.05        4
 15            4      21.05        5
 16            1       5.26        6
```

In the above example, the %FreqAll macro is called to produce a frequency analysis for the SASHELP.Class data set with the exception of the variable Name. I chose to set the Var parameter to _All_, as I may accidentally misspell a variables name. I excluded Name from the analysis by adding it to the ExcludeVar parameter. I then used proc print to produce a modest but informative report from the output data set created by the %FreqAll macro.

CONCLUSIONS

%FreqAll is simple, efficient, and effective in helping the programmer get a better understanding of their data. It is a tool like many others, but remember, there is no substitution to thinking about and understanding your data. Tools like the %FreqAll macro makes the process of understanding your data easier, but by no means should it be the only tool you use to this end. A better understanding of your data hopefully will help get your projects done on time and correctly.

References

Data Quality The Accuracy Dimension By Jack E. Olson

Global Data Management Survey, 2002 Price Waterhouse Coopers

Thanks to: Ron Fehd for sending the original version of this macro. Paul StLouis and Dianne Piaskoski for their invaluable editing ability. All the people on SAS-L.

Contact Information

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

Toby Dunn
550 Heimer Rd.
San Antonio, Tx. 78232
TobyDunn@HotMail.com

```
%Macro FreqAll( Lib= Work   , Mem=   , Vars= , LevelLimit=   , NObs2View= ,
                Format= Yes , Order= Internal , DataOut=  , ExcludeVars=  , ) ;
/*************************************************************************/
/** Macro Name : FreqAll                                                **/
/**                                                                     **/
/**                                                                     **/
/** Purpose    : The FreqAll macro creates a customized frequency listing  **/
/**              of specified vars.                                     **/
/**                                                                     **/
/**                                                                     **/
/** Parameters : Lib   ~ Libname of the Data Set containing the data to run **/
/**                     the frequencies on.  Default value is 'WORK'.   **/
/**                                                                     **/
/**              Mem   ~ Name of the Input Data Set. No Defualt Value.  **/
/**                                                                     **/
/**              Vars  ~ Space Seperated List of variables to perform the  **/
/**                     Frequency On.                                   **/
/**                     Valid values are: _All_       - All Vars        **/
/**                                       _Numeric_   - Numeric Vars Only  **/
/**                                       _Character_ - Character Vars Only**/
/**                                       User Supplied List of Var Names  **/
/**                                                                     **/
/**              LevelLimit ~ Numeric Limit At Which Frequecy For A Variable**/
/**                           Will Be Truncated To The Level Specified In  **/
/**                           NObs2View Parameter.                      **/
/**                                                                     **/
/**              NObs2View ~ A number representing the Top and Bottom   **/
/**                          frequencies to be included in the report.  **/
/**                          If no value is specified then all of the   **/
/**                          frequency values will be included in the   **/
/**                          output Data Set.  No default value.        **/
/**                                                                     **/
/**              Format    ~ Specifies whether to print the frequency   **/
/**                          values by their associated formats.        **/
/**                          Valid Values are 'Yes','No',’N’, and ’Y’.  The **/
/**                          Default  Value is 'YES'.                   **/
/**                                                                     **/
/**              Order     ~ Specifies how SAS should group the data in the **/
/**                          frequency analysis.                        **/
/**                          Valid Values: Internal, Data, Formatted, Freq  **/
/**                          Default value is Internal.                 **/
/**                                                                     **/
/**              DataOut   ~ The output data set name that will store the  **/
/**                          Frequiencies.                              **/
/**                                                                     **/
/**              ExcludeVars ~ Space Seperated List Of Valid Variable Names **/
/**                            To Exclude From The Frequency Output.    **/
/**                            User Defined List.                       **/
/**                                                                     **/
/**                                                 ****************    **/
/**                                                 * Make It Right *    **/
/**                                                 ****************    **/
/*************************************************************************/

%Local DSID Close DoWindow ;

%If ( %Length( %SuperQ(Vars) ) > 0 ) %Then %Do ;
  %Let Vars = %SysFunc( Compbl( &Vars ) ) ;
%End ;


%If ( %Length( %SuperQ(ExcludeVars) ) > 0 ) %Then %Do ;
  %Let ExcludeVars = %SysFunc( Compbl( &ExcludeVars ) ) ;
%End ;

 /*******************************************/
 /** Check The Existence Of The Input DataSet **/
```

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

%If ( %Sysfunc( Exist( &Lib..&Mem ) ) Eq 0 ) %Then %Do ;
       %Put ;
       %Put ;
       %Put ERROR: The Data Set [&Lib..&Mem] Does Not Exist!!! ;
       %Put ERROR: Please Check The Value Of Lib= or Mem= Parameters ;
       %Put ;
       %Put ;
       %Return ;
 %End ;


 /*********************************************/
 /** Check The Value Of Parm Vars To Ensure **/
 /** That It Meets One Of The Required Types**/
 /*********************************************/

 %If ( %SysFunc( IndexW( _ALL_ _CHARACTER_ _NUMERIC_ , %UpCase( &Vars ) ) ) EQ 0 ) Or
     ( %Length( %SuperQ(Vars) ) = 0 ) %Then %Do ;
      %Put ;
      %Put ;
      %Put ERROR: (&Vars) Is Not A Valid Value For Parameter Vars ;
      %Put ERROR: The Valid Values Are : _All_, _Character_, _Numeric_, Or ;
      %Put ERROR: A User Supplied List Of Valid Variable Names ;
      %Put ;
      %Put ;
      %Return ;
 %End ;


 /*********************************************/
 /** Check The Value Of Parm Format To Ensure  **/
 /** That It Is One Of The 4 Acceptable Values **/
 /*********************************************/

 %If ( %SysFunc( IndexW( YES Y NO N , %UpCase( &Format ) ) ) EQ 0 ) Or
     ( %Length( %SuperQ(Format) ) = 0 ) %Then %Do ;
      %Put ;
      %Put ;
      %Put ERROR: (&Format) Is Not A Valid Value For Parameter Format ;
      %Put ERROR: The Valid Values Are : Yes, Y , No , N ;
      %Put ;
      %Put ;
      %Return ;
 %End ;


 /*********************************************/
 /** Check The Value Of Parm Order To Ensure   **/
 /** That It Is One Of The 4 Acceptable Values **/
 /*********************************************/

 %If ( %SysFunc( IndexW( INTERNAL FORMATTED DATA FREQ , %UpCase( &Order ) ) ) EQ 0 )
     Or
     ( %Length( %SuperQ(Order) ) = 0 ) %Then %Do ;
      %Put ;
      %Put ;
      %Put ERROR: (&Order) Is Not A Valid Value For Parameter Order ;
      %Put ERROR: The Valid Values Are : Internal, Formatted, Data, Or Freq ;
      %Put ;
      %Put ;
      %Return ;
 %End ;


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

```
 /** Check To See If A Variable Listed In Either  **/
 /** The Vars Or ExcludList Parameters Are In The **/
 /** Specified Input Data Set.                    **/
 /************************************************/


%Let DSID = %SysFunc( Open( &LIB..&MEM , IS ) ) ;

%If ( &DSID = 0 ) %Then %Do ;
  %Put ;
  %Put ;
  %Put ERROR: Error Trying To Open Data Set ( %UpCase(&DataIn) ) ;
  %Put ERROR: %SysFunc( SysMsg( ) ) ;
  %Put ;
  %Put ;
  %Return ;
%End ;

/*********************/
/** Check Parm Vars **/
/*********************/
 %If ( %SysFunc( IndexW( _ALL_ _CHARACTER_ _NUMERIC_ , %UpCase( &Vars ) ) ) EQ 0 )
%Then %Do ;

   %Let VarStop = %Eval( %SysFunc( CountC( %Str(&Vars) , %Str( ) ) ) +
                         ( %Length( &Vars ) > 0 ) ) ;

   %Do I = 1 %To &VarStop ;
     %If %Eval( %SysFunc( VarNum( &DSID , %Scan( %Str(&Vars) , &I , %Str( ) ) ) ) = 0 )
     %Then %Do ;
       %Put ;
       %Put ;
       %Put ERROR: Can Not Find The Following Variable ( %UpCase( %Scan( %Str(&Vars) ,
       &I , %Str( ) ) ) ) ;
       %Put ERROR: In The DataSet &LIb..&Mem ;
       %Put ERROR: Please Specify A Variable In &Lib..&Mem ;
       %Put ;
       %Put ;
       %Return ;
     %End ;
   %End ;
%End ;


/****************************/
/** Check ExcludeList Parm **/
/****************************/

%If ( %Length( %SuperQ(ExcludeVars) ) > 0 ) %Then %Do ;

   %Let ExlStop = %Eval( %SysFunc( CountC( %Str(&ExcludeVars) , %Str( ) ) ) +
                         ( %Length( &ExcludeVars ) > 0 ) ) ;

   %Do K = 1 %To &ExlStop ;
     %If %Eval( %SysFunc( VarNum( &DSID , %Scan( %Str(&ExcludeVars) , &K ,
       %Str( ) ) ) ) = 0 ) %Then %Do ;
       %Put ;
       %Put ;
       %Put ERROR: Can Not Find The Following Variable
                  (%UpCase( %Scan( %Str(&ExcludeVars) , &K , %Str( ) ) ) ) ;
       %Put ERROR: In The DataSet %UpCase(&LIb..&Mem) ;
       %Put ERROR: Please Specify A Variable In %UpCase(&Lib..&Mem) ;
       %Put ;
       %Put ;
       %Return ;
     %End ;
   %End ;
```

```
%End ;

%Let Close = %SysFunc( Close( &DSID ) ) ;

%If ( &Close Ne 0 ) %Then %Do ;
  %Put ;
  %Put ;
  %Put ERROR: Error Trying To Close Data Set ( %UpCase(&DataIn) ) ;
  %Put ERROR: %SysFunc( SysMsg( ) ) ;
  %Put ;
  %Put ;
  %Return ;
%End ;


/*****************************************/
/** Check That The Values Of The Format   **/
/** And Order Parameters To Not Logically **/
/** Conflict.                             **/
/*****************************************/

%If ( ( %SysFunc( IndexW( NO N , %UpCase( &Format ) ) ) > 0 ) And
      ( %Upcase( &Order ) = FORMATTED ) ) %Then %Do ;
      %Put ;
      %Put ;
      %Put WARNING: You Have Specified Parm Format = (&Format) And Order = (&Order) ;
      %Put WARNING: NonFormatted Values Will Be Used ;
      %Put ;
      %Put ;
%End ;


/*********************************************/
/** Check That The Values Of The LevelLimit   **/
/** And NObs2View Parameters To Not Logically **/
/** Conflict.                                 **/
/*********************************************/

%If ( ( %Length( %SuperQ(LevelLimit) ) = 0 )  And ( %Length( %SuperQ(NObs2View) ) Ne
         0 ) )
    Or
    ( ( %Length( %SuperQ(LevelLimit) ) Ne 0 ) And ( %Length( %SuperQ(NObs2View) ) =
        0 ) )
    Or
    ( &LevelLimit < &NObs2View ) %Then %Do ;
      %Put ;
      %Put ;
      %Put WARNING: You Have Specified Parm LevelLimit = (&LevelLimit) And NObs2View =
                    (&NObs2View) ;
      %Put WARNING: All Levels Will Be Created ;
      %Put ;
      %Put ;
%End ;




/*******************************/
/** Create Frequency Data Set **/
/*******************************/

 ODS Listing Close ;
 ODS Output OneWayFreqs = Freqs ;

 Proc Freq
  Data = &Lib..&Mem
     %If ( %Length( %SuperQ( ExcludeVars ) ) > 0 ) %Then %Do ;
```

```
        ( Drop = &ExcludeVars )
       %End ;
 Order = &Order ;
 Table &Vars / Missing ;
    %If ( ( %UpCase( &Format = NO ) ) Or ( %UpCase( &Format ) = N ) ) %Then %Do ;
       Format _All_ ;
       %End ;
Run ;

ODS Listing ;


/*********************************/
/** Create Final Data Structure **/
/*********************************/

Data NewFreqsA ( Keep = Var Val Frequency Percent Level ) ;
Set Freqs ;
By Table NotSorted ;

If First.Table Then Level = 0 ;

Level + 1 ;

Var = PropCase( Compress( Scan( Table , 2 , ' ' ) ) ) ;

Val = Strip( VValueX( Var ) ) ;

Run ;


/*******************************************/
/** Attach Variable Attributes And Truncate **/
/** Data If LevelLimit And NObs2View Are    **/
/** Specified.                              **/
/*******************************************/

Proc SQL ;
Create Table NewFreqsB As
 Select Freqs.* , CatX( '  ' , PropCase( Freqs.Var ) , PropCase( Dict.Attr ) ) As
        Attribute
   From NewFreqsA As Freqs
        Left Join
        ( Select Name , ( PropCase( Type ) || ':' || PropCase( Strip( Put( Length , 8.
          -L ) ) ) || '   ' || PropCase( Label ) ) As Attr Format=$50.
            From Dictionary.Columns
              Where LibName = %UpCase( "&Lib" ) And
                    MemName = %Upcase( "&Mem" ) ) As Dict
        On Freqs.Var = Dict.Name

        %If ( %Length( %SuperQ( LevelLimit ) ) > 0 ) And
            ( %Length( %SuperQ( NObs2View ) ) > 0 ) %Then %Do ;
           Group By Freqs.Var
             Having ( Max( Freqs.Level ) < &LevelLimit   ) Or
                    ( ( Freqs.Level <= &Nobs2View  ) Or
                    ( Freqs.Level  > ( Max( Freqs.Level ) - &Nobs2View ) ) )
               %End ;
           Order By Freqs.Var , Freqs.Level ;
Quit ;


/*********************/
/** Output Data Set **/
/*********************/
 Proc Append
   Base = &DataOut
     Data = NewFreqsB Force ;
```

```
 Run ;


/************************/
/** Clean Up Work Space **/
/************************/

 Proc Datasets NoList ;
 Delete NewFreqsA
        NewFreqsB
           Freqs ;
 Quit ;

 %Mend FreqAll ;
```