

Beyond the Data Dictionary – Database Consistency

Sheree Hughes, Fred Hutchinson Cancer Research Center, Seattle, WA

ABSTRACT

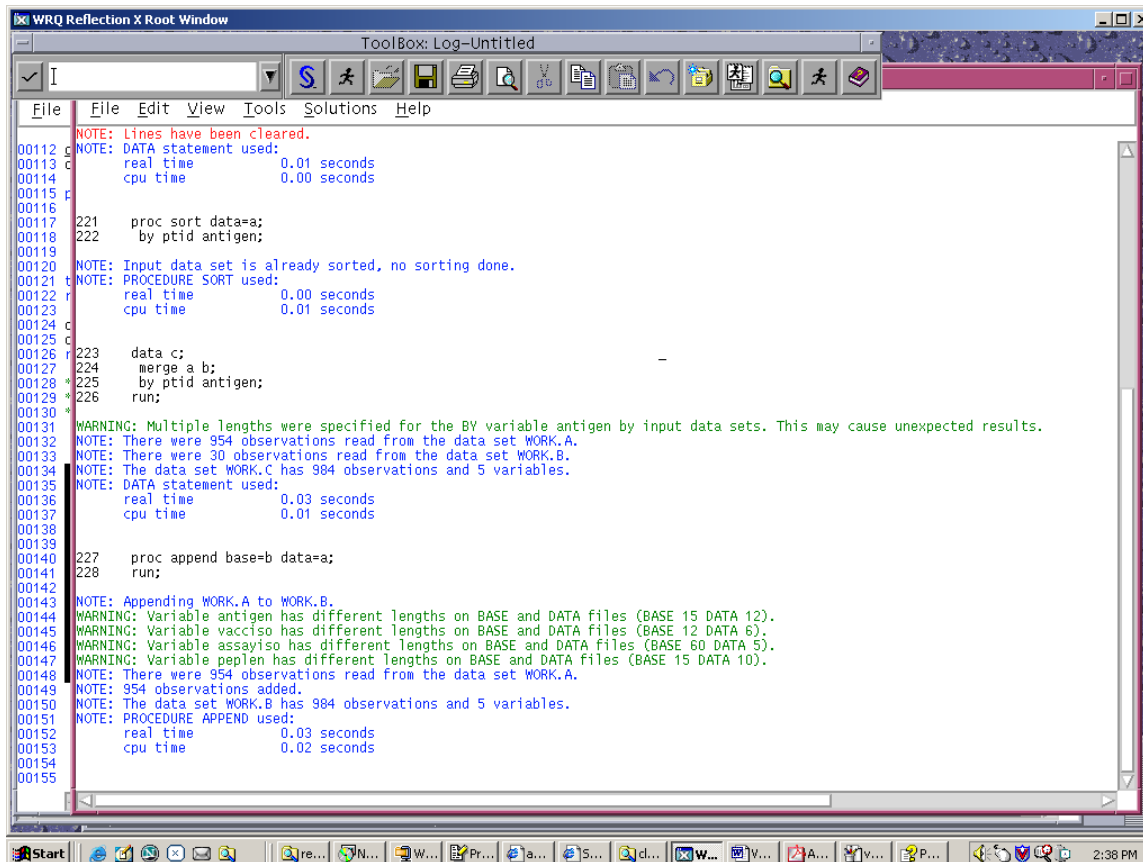
How often do you get a LOG file surprise telling you that the variable length of similarly named variables differs in more than one data source when you are trying to merge, or append them? Data Dictionaries serve a critical role in helping a user know his data within a dataset, but they do not enable a user to get the bird's eye view that is also needed across a database. This paper details an effective way to determine variable name, length, type, and format consistency across multiple SAS datasets in a database. It uses the features of PROC CONTENTS, The Data Step, and PROC TABULATE to produce a report that indicates, at a glance, any discrepancies in name, length, type, or format characteristics. The "Variable Dictionary", like the "Data Dictionary" is a tool that every database manager should not be without! In a few easy steps you can keep your data clean, and know what to fix, if it is not. In an enhanced version of the application it is also possible to indicate KEY variables used across all SAS datasets in your database to determine observation uniqueness, and possible merge combinations.

INTRODUCTION

The value of a data dictionary has long been apparent. The previous work of several of our SAS colleagues have shown the need to know our data more intimately than what PROC CONTENTS, PROC PRINT, PROC FREQ, and PROC UNIVARIATE can provide separately. Combining the information from these procedures yields a tool that enables analysts to work confidently with a specified dataset. This paper extends the concept to database integrity, by defining common variables across multiple datasets and insuring their compatibility as to data type, length, format, and label.

PROBLEM DEFINED

One of the many strengths of SAS is the ability to merge, concatenate, interleave, update, and otherwise combine one or more SAS datasets. How often is it that reference to the LOG informs us that incompatibilities have been detected and either prevents the step from executing, or warns us that we may get unexpected results? An example of the type of errors, and or warnings I refer to is shown in this LOG excerpt:



DATABASE COMPATIBILITY BY DESIGN

When combining datasets by common variables it is rewarding to know that not only does the step execute, but also that we will obtain the result we expect. This can be accomplished through the strategy of designing the database such that common variables are indeed common, as to their attributes.

As a data manager of a SAS clinical trial laboratory results database, it quickly became apparent that I needed a tool to guide the building of multiple datasets with compatible variable attributes. Thus was born the Variable Dictionary. This is a reference document that serves as a tool not only for database managers, but also for all users of the data. An example of a Variable Dictionary is shown below:

Table of HIVT Data Base Variables

13:50 Saturday, August 28, 2004 1

Var Name	Label	Length	Var Type	Format	ADC	CTL	ILP	ELS	HLA	ICS	IL2	IVC	LPA	NAB	NAP
incubate	Incubation Period	3	Num					x					Key		
inunits	Infectious Units	6	Char											x	x
isolate	Isolate	12	Char											Key	Key
labfile	File Name	23	Char		x	x	x	x	x	x	x	x	x	x	x
labid	Lab ID	2	Char		Key	Key	Key	Key	Key	Key	Key	Key	Key	Key	Key
labpos	Lab Positivity Call	3	Num							x					
lytpct	Percent Lysis	8	Num		x										
method	Count Method 1=Guava 2=Hemoey	1	Char					x		x					
mercur	Microcuries	8	Num										x		
moddt	Date Resord was Modified	8	Num	DATE	x	x	x	x	x	x	x	x	x	x	x
neutpct	Neutralization Percent	8	Num												x
nout	No antigen Reading	8	Num				x				x				
noantran	No Antigen Range	8	Num				x				x				
nrepl	# Replicate	3	Num							Key					
nsub	# of Subset Cells Analyzed	8	Num							x					
outlier	Outlier (Y/N)?	1	Char									x			
petpos	% Cytoline Positive Cells in Subset	8	Num							x					
petsub	% of Lymphocytes from Subset	8	Num							x					
pepcnc	Peptide Conc.	8	Num					x							
peplen	Peptide Length/Overlap	15	Char					Key		Key					
plateid		18	Char					x							
poscrit	Positive Criteria (%)	8	Num											x	
protocol	Protocol Number	8	Num		Key	Key	Key	Key	Key	Key	Key	Key	Key	Key	Key
ptid	Participant ID	8	Num		Key	Key	Key	Key	Key	Key	Key	Key	Key	Key	Key
reading	Reading	8	Num				x				x				

/scharp/lab_tools/tn/code/sas/ShereeTest/var_dictionary

Note the facility of displaying the information in tabular form. A user can quickly scan this reference document, and glean all critical information relating to the variables contained in the database, i.e. which variables exist in one or more datasets, and which variables are keys. If variables exist in one or more datasets they are candidates for combining data through MERGE, or SET processing.

The Variable Dictionary is created from the output of PROC CONTENTS, some Data Step manipulation, including: variable creation, and value transformation is required. The results are displayed using the features of PROC TABULATE, and formats created in PROC FORMAT that distinguishes the “key” variables within a dataset, and variable type.

WHY PROC TABULATE?

PROC TABULATE is a powerful procedure that displays n-way relational information in the 2 dimensions we can view on output. It provides the visual aid of automatic grids.

VARIABLE DICTIONARY CODE

```
*****;
*
*      Program : variable_dictionary.sas
*
*      Creation Date: 02/06/04
*
*      Primary client: Statisticians & LTP
*
*      Purpose : Get list of all variables used in all datasets
*
*****;
```

```

*      Location: /scharp/lab_tools/vtn/code      *;
*      *;
*      Author: Sheree Hughes      *;
*      *;
*      Project : Across all assays      *;
*      Fred Hutchinson Cancer Research Center      *;
*      *;
*      Inputs:      *;
*      - rawdata.m_ASSAYTYPE_new SAS datasets      *;
*      *;
*      - SAS contents:      *;
*      *;
*      *;
*      Outputs:      *;
*      - Report of all Variables, types, lengths, & formats      *;
*      *;
*      Usage: sas82 Get_New_Files      *;
*      *;
*      Special Notes:      *;
*      *;
*      Revisions:   added labels 5/20/04      *;
*      *;
*****;
footnote "/scharp/lab_tools/vtn/code/sas/ShereeTest/var_dictionary";

```

The code required to produce this reference tool is remarkably simple. Begin with PROC FORMAT.

This mapping of values, through the format: varpl., determines whether a variable is a key, exists in the dataset (x), or does not exist in the dataset (missing). Also the format: vtype. names the variable type.

```

*****;
* Set up formats to map values to appropriate representation in final *;
* report      *;
*****;
proc format;
  value varpl
    1= ' x '
    2=' Key '
    other=' ';
  value vtype
    1='Num'
    2='Char';

```

Next run PROC CONTENTS using the keyword `_all_` on the database of interest, and specify an output location of the results, with a KEEP dataset option to keep relevant variables.

```

*****;
* Create output dataset from proc contents for each assay dataset      *;
*****;

proc contents data=rawdata._all_
  out=ALLVARS(keep=memname name label format type length
  where=(memname='M_')) noprint;

```

Combine all the individual datasets with a MERGE using a WHERE dataset option identifying the member name, and the IN dataset option to designate the source. MERGE the dataset components by variable name, format, length, and type, remembering that they are pre-sorted by PROC CONTENTS.

```

*****;
* Create master dataset from merged contents *;
* Associate assaytype with each assay, & define the key fields *;
*****;
data dictionary;
  merge ALLVARS (where=(memname='M_ADC') in=inADC)
        ALLVARS (where=(memname='M_CTL') in=inCTL)
        ALLVARS (where=(memname='M_ELP') in=inELP)
        ALLVARS (where=(memname='M_ELS') in=inELS)
        ALLVARS (where=(memname='M_HLA') in=inHLA)
        ALLVARS (where=(memname='M_ICS') in=inICS)
        ALLVARS (where=(memname='M_IL2') in=inIL2)
        ALLVARS (where=(memname='M_IVC') in=inIVC)
        ALLVARS (where=(memname='M_LPA') in=inLPA)
        ALLVARS (where=(memname='M_NAB') in=inNAB)
        ALLVARS (where=(memname='M_NAP') in=inNAP);
  by name format length type;

```

Set up an explicit array of variables, which identifies each dataset by name. In this example the dataset names are: ADC, CTL, ELP, etc. Then based upon what position in the array the dataset is, define the key variables and set the value to the dataset variable to 2, using the array reference PLACE. All other variables are given the value of the dataset indicators, either 0, or 1, to indicate not in the dataset, or in the dataset, respectively.

```

array finame{11} inADC inCTL inELP inELS inHLA inICS inIL2 inIVC inLPA inNAB inNAP;
array place{11} 3 ADC CTL ELP ELS HLA ICS IL2 IVC LPA NAB NAP;
do i=1 to 11;
  if (name in ('labid','protocol','visitno','ptid','subtype')) then place{i}=2;
  else if (i=1 & name in ('dilution','target')) then place{1}=2;
  else if (i=2 & name in ('effector','target')) then place{2}=2;
  else if (i=3 & name in ('antigen','titer')) then place{3}=2;
  else if (i=4 & name in ('antigen','assayiso','vacciso')) then place{4}=2;
  else if (i=6 & name in ('antigen','assayiso','vacciso')) then place{6}=2;
  else if (i=7 & name in ('dilution','titer')) then place{7}=2;
  else if (i=8 & name in ('chaldose')) then place{8}=2;
  else if (i=9 & name in ('antigen','cellwell','effector','viriso')) then place{9}=2;
  else if (i=10 & name in ('isolate','assaytyp','celltype','cutoff')) then place{10}=2;
  else if (i=11 & name in ('isolate','assaytyp','celltype','serdilu')) then place{11}=2;
  else place{i}=finame{i};
end;
format type vtype.;

```

Use PROC TABULATE to display the information. The class variables correspond to the variable attributes: name, label, length, type and format. The table is defined as the attributes in the first, or vertical dimension, and the dataset indicators in the second, or horizontal dimension. Format the values in the table using the varpl. format created earlier.

```

*****;
* Tabulate final report with formatting to produce output which *;
* indicates all variables that exist for a given assay & whether it is *;
* a key field *;
*****;
ods trace on;
ods pdf file="/ssharp/lab_tools/vtn/assay_results/reports/Variable_Dictionary.pdf";

proc tabulate data=dictionary(where=(name>='in' & name<='re')) format=8.0 missing;
  class name label length type format;
  var ADC CTL ELP ELS HLA ICS IL2 IVC LPA NAB NAP;
  table name='Var Name'* label='Label' * length='Length'* type='Var Type' * format='Format',
        (ADC CTL ELP ELS HLA ICS IL2 IVC LPA NAB NAP)*(sum='
*f=varpl.)/rts=57;
  format ADC CTL ELP ELS HLA ICS IL2 IVC LPA NAB NAP varpl.;
title "Table of HVTN Data Base Variables";
run;

```

```
ods pdf close;  
ods trace off;  
run;
```

```
*****;  
* END PROGRAM      *;  
*****;
```

ADAPTATION

It is possible to group sets of variables together within the database by adding another categorical variable as a class variable in PROC TABULATE. Another application allowed easy grouping of related variables by this technique.

EPILOGUE

Remember that SAS is only limited by the imagination of the user!