

ATTRDIFF: A Macro to Identify Differences in Variable Attributes

Frank C. Dilorio, ASG, Inc., Cary, NC

INTRODUCTION

The setting. The problem of conflicting variable attributes has long bedeviled SAS programmers. GENDER, for example, may be coded as character ("M" and "F") in one database and numeric (1 and 2) in another. A character variable may have a length of two bytes in one dataset yet be stored in three bytes in another. Other real-life scenarios are painfully easy to imagine. In all of them, the attribute differences usually cause havoc when the programmer attempts to combine the datasets.

Programming and coding standards help alleviate these problems. However, when data comes from different sources or when one deals with "legacy" data, conflicting attributes are almost inevitable.

The application. The ATTRDIFF macro attempts to simplify the life of the programmer charged with combining data from disparate sources. The macro reviews all datasets in a SAS dataset library and reports discrepancies between all like-named variables. If GENDER is character in dataset YR95 and numeric in YR94, ATTRDIFF will report the difference in a clearly labeled PRINT procedure.

ATTRDIFF does not attempt to *change* variable attributes. It is simply intended as a tool to identify differences, simplifying the programmer's life by avoiding time-consuming review of CONTENTS output for the same information.

This paper describes the ATTRDIFF macro. The logic behind the macro's organization is discussed first, followed by an example of its use.

THE MACRO

The complete macro is presented in Appendix A. It is broken into several pieces here for the sake of explanation.

First, we define the macro, giving it a name and identifying keyword parameters. The LIBNAME parameter identifies the name of the library to examine. The default is WORK. The COMPARE parameter tells ATTRDIFF which attributes to compare. Multiple choices are separated by a slash (/). Choices are:

type	type (the default)
len	length
lab	label
fmt	format
inf	informat

```
%macro attrdiff(libname=work, compare=type);
```

Identify the variables used by the macro. Their meaning corresponds to the COMPARE parameter options. The variables prefixed by "L" are used for the LABEL statement in the PRINT procedure.

```
%local type label length format informat
        lty lla lle lfo lin ;
```

Convert the user-supplied information to upper case, then see what's being compared. If the %INDEX function spots TYPE, LEN, etc. it adjusts the value of the appropriate macro variables.

```
%let libname = %upcase(&libname.);
%let compare = %upcase(&compare.);
%if %index(&compare., TYPE) > 0 %then %do;
    %let type = type ;
    %let lty = l((compared)) ;
%end;
%if %index(&compare., LEN) > 0 %then %do;
    %let length = length;
    %let lle = l((compared));
%end;
```

```
%if %index(&compare., LAB) > 0 %then %do;
    %let label = label;
    %let lla = l((compared));
%end;
%if %index(&compare., FMT) > 0 %then %do;
    %let format = format;
    %let lfo = l((compared));
%end;
%if %index(&compare., INF) > 0 %then %do;
    %let informat = informat;
    %let lin = l((compared));
%end;
```

Run the SQL procedure. First we create datasets _____DUPS. It contains relevant variable information for the LIBNAME library. We use the Dictionary Tables to get this data. We also write the path pointed to by LIBNAME into the macro variable PATH. It will be used later in the TITLE1 statement.

```
proc sql noprint;
create table work._____dups as
select libname, memname, name, label,
       type, length, format, informat
from dictionary.columns
where libname = "&libname."
order by name;
select path into :path
from dictionary.members
where libname = "&libname." ;
quit;
```

Sort the dataset, eliminating duplicates of NAME and any user-specified attributes. If, for example, the user specified COMPARE=LEN, the SORT procedure would eliminate observations with similar values of variables NAME and LENGTH. This is desirable. If, say, the variable STATUS appeared in more than one dataset in the library and it had identical values of LENGTH, only one observation would be written. That means that STATUS had the same length in each dataset — it is represented only once in _____DUPS. On the other hand, if LENGTH was 3 in one dataset and 4 in another, there would be two observations in _____DUPS.

```
proc sort nodupkey
       data=_____dups(drop=libname);
by name &type. &length. &label.
   &format. &informat.;
run;
```

Process _____DUPS, writing only observations whose sort keys occur more than once in the dataset. These are the variables with one or more conflicting attributes.

```
data _____dups;
set _____dups end=eof;
by name;
if ^(first.name & last.name) then output;
if eof then
    call symput("path",trim("&path."));
run;
```

Use SQL to see if _____DUPS has any observations. Write the value to macro variable COUNT.

```
proc sql noprint;
select nobs into :count
from dictionary.tables
where libname = 'WORK' &
      memname = '_____DUPS';
quit;
```

If we did find conflicting attributes (_____DUPS had > 0 observations), report.

```
%if &count. > 0 %then %do;
    options ls=78 pageno=1 nodate;
```

```

proc print split='|' data=___dups;
id name;
by name;
var memname type length label;
format informat;
label name = "Variable";
label memname = "Dataset";
label type = "Type&lt;y.";
label length = "Length&lt;lle.";
label label = "Label&lt;lla.";
label format = "Format&lt;lfo.";
label informat = "Informat&lt;lin.";
title1 "Libname &libname. (Path &path.)";
title2 "Comparing &type. &length. &label.
&format. &informat.";
title3 "Program run at &systime on
&sysday., &sysdate.";
run;
%end;

```

Before leaving, delete ___DUPS.

```

proc delete data=work.___dups;
run;
%mend;

```

EXAMPLE

First, let's set up some test datasets. Not surprisingly, they have variables with conflicting attributes. GENDER is defined as numeric in FIRST, character in SECOND. NUM has a format of 3.2 and Z3. CHAR's lengths are 2 and 3.

```

data first;
gender = 'm';
num = 1;
format num 3.2;

```

```

char = 'ab';
run;

data second;
gender = 1;
num = 1;
format num z3.;
char = 'abc';
run;

```

We invoke ATTRDIFF using all defaults (LIBNAME will be WORK, COMPARE will be TYPE). The results are shown in Listing 1. Notice that the title lines highlight the variable being compared.

When ATTRDIFF is run with a COMPARE value of type/label/format the listing grows. Notice that even though CHAR has a length discrepancy (2 v. 3) it isn't reported, since LEN was not coded as part of the COMPARE option. The output is presented in Listing 2.

REFERENCES

ATTRDIFF utilizes the SAS dictionary tables. These are explained in SAS Technical Report P-222, "Changes and Enhancements to Base SAS Software, Release 6.07," and Dilorio and Michal, "Data About Data: An Introduction to Dictionary Tables," SESUG '95 Proceedings (also in SUGI 21 Proceedings).

QUESTIONS? COMMENTS?

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

102 Westbury Drive
Chapel Hill NC 27516-9154

919.942.2028
7216987@mcimail.com

LISTING 1: INVOKE USING ALL DEFAULTS

```

Libname WORK (Path C:\SAS\SASWORK\#TD07015)
Comparing type
Program run at 21:02 on Tuesday, 10SEP96

```

Variable	Dataset	Type (compared)	Length	Label	Format	Informat
GENDER	FIRST	char	1			
	SECOND	num	8			

LISTING 2: USER SPECIFIES TYPE, LABEL AND FORMAT

```

Libname WORK (Path C:\SAS\SASWORK\#TD07015)
Comparing type label format
Program run at 21:02 on Tuesday, 10SEP96

```

Variable	Dataset	Type (compared)	Length	Label (compared)	Format (compared)	Informat
GENDER	FIRST	char	1			
	SECOND	num	8			
NUM	FIRST	num	8		3.2	
	SECOND	num	8		Z3.	

APPENDIX A: MACRO LISTING

```

%macro attrdiff(libname=work, compare=type);
/* Identifies conflicting variable attributes in a
library and uses PROC PRINT to display results.
Usage: attrdiff(libname=libname, compare=characteristic(s))
Parameters: libname => the libname of the library
Default is WORK.
compare => one or more attributes to compare for like-named variables
in LIBNAME's datasets. If ANY of the attributes differs, the result
will be reported. Separate the attributes with a slash (/).

```

```

Author:      Frank DiIorio, ASG, Inc., 1996
*** Copy freely so long as authorship is cited! ***
*/
%local type label length format informat
      lty lla lle lfo lin ;

%let libname = %upcase(&libname.);
%let compare = %upcase(&compare.);
%if %index(&compare., TYPE) > 0 %then %do;
  %let type    = type ;
  %let lty     = |(compared) ;
%end;

%if %index(&compare., LEN ) > 0 %then %do;
  %let length  = length;
  %let lle     = |(compared);
%end;

%if %index(&compare., LAB ) > 0 %then %do;
  %let label   = label;
  %let lla     = |(compared);
%end;

%if %index(&compare., FMT ) > 0 %then %do;
  %let format  = format;
  %let lfo     = |(compared);
%end;

%if %index(&compare., INF ) > 0 %then %do;
  %let informat = informat;
  %let lin     = |(compared);
%end;

proc sql noprint;
create table work.____dups as
select libname, memname, name, label, type, length, format, informat
  from dictionary.columns
  where libname = "&libname."
  order by name;
select path into :path
  from dictionary.members
  where libname = "&libname." ;
quit;

proc sort nodupkey data=____dups(drop=libname);
by name &type. &length. &label. &format. &informat.;
run;

data ____dups;
set ____dups end=eof;
by name;
if ^(first.name & last.name) then output;
if eof then call symput("path",trim("&path."));
run;

proc sql noprint;
select nobs into :count
  from dictionary.tables
  where libname = 'WORK' & memname = '____DUPS';
quit;

%if &count. > 0 %then %do;
  options ls=78 pageno=1 nodate;
  proc print split='|' data=____dups;
  id name;
  by name;
  var memname type length label format informat;
  label name      = "Variable";
  label memname   = "Dataset";
  label type      = "Type&lty.";
  label length    = "Length&lle.";
  label label     = "Label&lla.";
  label format    = "Format&lfo.";
  label informat  = "Informat&lin.";
  title1 "Libname &libname. (Path &path.)";
  title2 "Comparing &type. &length. &label. &format. &informat.";
  title3 "Program run at &systime on &sysday., &sysdate.";
  run;
%end;

proc delete data=work.____dups;
run;
%mend;

```