Paper CC05

Automatically Right-Size Every Character Column after Importing Your Data to SAS: From the 255-byte Default to the Real Requirements

LeRoy Bessler, Assurant Health, Milwaukee, USA, bessler@execpc.com

Abstract

If you use the SAS Enterprise Guide Import Data function, during the Import process you do have the option to override the default length of 255 bytes for character columns. However, in many, if not most, real situations you may encounter *numerous* character columns for a potentially huge number of rows, with no practical way of guessing their actual length requirements, or of determining it by inspection. The macro solution presented here was validated in SAS 8. A demo of Import Data in SAS 9 / SAS Enterprise Guide 3 did not show any relief for the problem. The author has also encountered over-sized columns from some other SAS import methods. Besides right-sizing the columns, the macro optionally provides length-appropriate formats and informats for the character columns.

The Macro Solution

To demonstrate the problem, export (Save As) sashelp.class to, e.g., C:\class.xls. Then use the SAS Enterprise Guide Import Data function to get the data back, saving it as work.FromExcel. During the Import process, you are given the option to override the default length of 255 bytes for character columns. In this case, accept the default.

The most efficient and effective solution is to first determine which columns are character type, then read through the entire data set to determine the maximum length of each, and finally resize the data with that knowledge. This is best handled, as a repetitive utility function, using a macro. Here is its definition:

```
%macro RightSizeColumnsInTable(
       ShiftLeftAndTrim=YES,
       MatchingFormat=YES,
       MatchingInformat=YES,
       DataLib=,
       Table=,
       RightSizedDataLibTable=);

* One of the macro parameter default assignments
  is to shift column content left and trim it.
  Sometimes character data is justified right upon input to the macro.
  The macro design assumption, sometimes wrong and therefore subject to override,
  is to justify all character data left before right-sizing it. *;

%let DataLib = %upcase(&DataLib);
%let Table   = %upcase(&Table);
%let ShiftLeftAndTrim = %upcase(&ShiftLeftAndTrim);

* capture all the character column names in a SAS data set *;

proc sql;
create table work.CharacterColumns
as
select
name as Column format=$32.
from dictionary.columns
where libname EQ "&DataLib"
      and
      memname EQ "&Table"
         and
      type EQ 'char';
quit;

* store them as an indexed list of macro variables in the symbol table,
  along with a count of how many there are *;
```

```
data _null_;
set work.CharacterColumns end=Last;
call symput('Col'||trim(left(_N_)),trim(left(Column)));
if Last;
call symput('ColCount',_N_);
run;

* Read through the entire data set,
  retaining the maximum length of each character column.
  After reading the last observation,
  store the maximum length of each (and a correspondingly sized format and informat)
  as an indexed list of macro variables in the symbol table. *;

data _null_;
%do i = 1 %to &ColCount;
retain ColLength&i 0;
%end;
set &DataLib..&Table
(keep=
%do i = 1 %to &ColCount;
      &&Col&i
%end;
             ) end=Last;
%do i = 1 %to &ColCount;
ColLength&i = max(ColLength&i,length(&&Col&i));
%end;
if Last;
%do i = 1 %to &ColCount;
call symput("ColLength&i"  ,ColLength&i);
call symput("ColFormat&i"  ,'$'||trim(left(ColLength&i))||'.');
call symput("ColInformat&i",'$'||trim(left(ColLength&i))||'.');
%end;
run;

* Create a right-sized data set, by reading the original data set
  and applying the correct length (and optionally format and informat),
  and optionally applying LEFT and TRIM functions, to each character column. *;

data &RightSizedDataLibTable;
%do i = 1 %to &ColCount;
length   &&Col&i $ &&ColLength&i;
  %if %upcase(&MatchingFormat) eq YES
  %then %do;
format   &&Col&i &&ColFormat&i  ;
  %end;
  %if %upcase(&MatchingInformat) eq YES
  %then %do;
informat &&Col&i &&ColInformat&i;
  %end;
%end;
set &DataLib..&Table;
%if &ShiftLeftAndTrim EQ YES %then %do;
  %do i = 1 %to &ColCount;
&&Col&i = trim(left(&&Col&i));
  %end;
%end;
run;

%mend RightSizeColumnsInTable;
```

To see what this complicated-looking code does, invoke the macro so that the resolved code appears in the SAS log:

```
OPTIONS MPRINT; /* show resolved code */
```

```
%RightSizeColumnsInTable(
  DataLib=work,
  Table=FromExcel,
  RightSizedDataLibTable=work.RightSized);
```

From the SAS log, the PROC SQL step and the first DATA _NULL_ step are omitted here since they are not different from the macro definition code, except for resolution of &DataLib and &Table. Here are the interesting parts of the SAS log:

```
data _null_;
retain ColLength1 ColLength2 0;
set WORK.FROMEXCEL (keep= Name Sex ) end=Last;
ColLength1 = max(ColLength1,length(Name));
ColLength2 = max(ColLength2,length(Sex));
if Last;
call symput("ColLength1"  ,ColLength1);
call symput("ColFormat1"  ,'$'||trim(left(ColLength1))||'.');
call symput("ColInformat1",'$'||trim(left(ColLength1))||'.');
call symput("ColLength2"  ,ColLength2);
call symput("ColFormat2"  ,'$'||trim(left(ColLength2))||'.');
call symput("ColInformat2",'$'||trim(left(ColLength2))||'.');
run;

data work.RightSized;
length   Name $ 7;
format   Name $7. ;
informat Name $7.;
length   Sex $ 1;
format   Sex $1. ;
informat Sex $1.;
set WORK.FROMEXCEL;
Name = trim(left(Name));
Sex  = trim(left(Sex));
run;
```

The DATA _NULL_ step above delivers the column names and the lengths, formats, and informats needed to right-size the data in the final DATA step. In the case of the particular data used, the TRIM and LEFT functions were not really needed, but accepted as the macro invocation defaults.

Prior Publication

An earlier edition of this macro, without ShiftLeftAndTrim, was first published in *VIEWS News*, Issue 30, 2nd Quarter 2005, available at http://www.views-uk.demon.co.uk, the web site for the "VIEWS International SAS Programmer Community".

Notices

Author Information

Your questions, comments, suggestions, and other solutions are welcome.

LeRoy Bessler PhD
Email: bessler@execpc.com
Phone: 1 414 351 6748 (evenings and weekends—time is six hours earlier than GMT)

Dr. LeRoy Bessler has special interests in: Software-Intelligent application development, which yields SAS solutions that are reliable, reusable, maintainable, and extendable; and communication-effective design and construction of reports, tables, graphs, maps, spreadsheets, and web pages.