Automating Character Variable Length Updates Using SAS Macros

Identifying and resizing character variable lengths in SAS to produce smaller, more efficient datasets

Carleigh Jo Crabtree, SAS Technical Training Consultant

THE PROBLEM

Wasted Space

SAS allocates the full declared length for every character variable in every observation — even when the actual stored value is far shorter. The unused space is padded with blanks, inflating file size and slowing processing.

HOW LENGTH WORKS

Character Variable Length

A character variable with length 200 uses 200 bytes per row, regardless of the value. In the English language, standard letters and digits use 1 byte each. If the longest real value is only 30 characters, 170 bytes per row are wasted.

THE SOLUTION

Two Macros

Run **%charCheck** to produce a comparison table of defined vs. actual character variable lengths.
Run **%charResize** to create a copy of your data set with reduced-sized character columns — no integrity lost.

%CHARCHECK

Inspects defined vs. actual lengths

Creates a table of all character variables in a specified data set and lists the longest value in that variable, along with the variables defined length

Variable	DefinedLength	MaxLength
Name	200	28
Gender	50	6
Blood_Type	10	3
Medical_Condition	300	12
Doctor	200	28
Hospital	300	35
Insurance_Provider	200	16
Admission_Type	100	9
Medication	300	11
Test_Results	500	12

%CHARRESIZE

Rebuilds with Reduced Lengths

Uses %CharCheck to create a new data set with updated lengths if a character variables' **MaxLength** is smaller than the **DefinedLength**

Variable	Type	Len
Admission_Type	Char	9
Age	Num	8
Billing_Amount	Num	8
Blood_Type	Char	3
Date_of_Admission	Num	8
Discharge_Date	Num	8
Doctor	Char	28
Gender	Char	6
Hospital	Char	35
Insurance_Provider	Char	16
Medical_Condition	Char	12
Medication	Char	11
Name	Char	28
Room_Number	Num	8
Test_Results	Char	12

Using the Macros

STEP 1 %CHARCHECK

Call the macro with your library and data set name. The output table shows all character variables alongside their defined length and longest value actually stored.

```
%charCheck(lib=psug, data=healthcare, out=char_lengths);
```

lib= library where your data set lives, defaults to WORK if omitted

data= name of the data set to analyze

out= name for the output summary table, defaults to CHAR_LENGTHS if omitted

Variable	DefinedLength	MaxLength
Name	200	28
Gender	50	6

Rows where **DefinedLength** > **MaxLength** are candidates for resizing.

Using the Macros

STEP 2 %CHARRESIZE

Once you've reviewed the output from **%charCheck**, call **%charResize**. It calls **%charCheck** and creates a new data set with the reduced-sized lengths.

```
%charResize(lib=psug,data=healthcare,outLib=psug,out=healthcare_new);
```

- **lib=** library where your data set lives, defaults to WORK if omitted
- **data=** name of the data set to analyze
- **outLib=** library where the *new resized* data set will be written
- **out=** name for the output summary table, defaults to CHAR_LENGTHS if omitted

What happens inside

1. **%charCheck** is called and produces an internal summary table
2. Columns where **DefinedLength** > **MaxLength** are identified as candidates.
3. A LENGTH statement is built dynamically using CATX and stored in **&len_stmt**.
4. A DATA step writes the new data set, applying updated lengths where needed.

SAS Server: /export/viya/homes/carleighjo.crabtree@sas.com/PharmaSUG/CharMacro/CharCheck.sas

```
1  /* %charCheck creates a data set of all character variables in a specified data set along with */
2  /* their defined length and the length of the longest value in the variable */
3
4  %macro charCheck(lib=work, data=, out=char_lengths);
5
6      %local dsid nvars i varname vartype varlen rc;
7
8      /* Open the dataset in input mode */
9      /* If the data set exists and opens successfully, open returns a positive integer like 1, 2, 3... */
10     %let dsid = %sysfunc(open(&lib..&data,i));
11     /*  If the data set opens successfully, &dsid>0, the THEN DO block executes */
12     %if &dsid %then %do;
13
14         /* Queries the number of variables in the data set referred to by dsid */
15         %let nvars = %sysfunc(attrn(&dsid,nvars));
16
17         /* Create a dataset to store results */
18         data &out;
19             length Variable $32. DefinedLength MaxLength 8.;
20             stop;
21         run;
22
23         /* Loop over variables and creates &i */
24         %do i = 1 %to &nvars;
25             /* Stores whether the current variable is N (numeric) or C (character) */
26             %let vartype = %sysfunc(vartype(&dsid,&i));
27             /* If the value of vartype is C (character) then do... */
28             %if &vartype = C %then %do;
29                 /* Stores the name of the i-th variable */
30                 %let varname = %sysfunc(varname(&dsid,&i));
31                 /* Stores the defined length of the i-th variable */
32                 %let varlen  = %sysfunc(varlen(&dsid,&i));
33
34                 proc sql noprint;
35                     /* Appends a new row to the data set, each character variable will be its own row */
36                     insert into &out
37                             /* Puts the variable name into a column named Variable */
38                     select "&varname" as Variable,
```

%charCheck

Thank you!

Carleigh Jo Crabtree, SAS Technical Training Consultant

Full programs on GitHub:

Let's connect on LinkedIn!