ABSTRACT

This is a simple macro that will examine all fields in a SAS® dataset that are stored as character data to see if they contain real character data or if they could be converted to numeric. It then performs the conversion and reports in the log the names of any fields that could not be converted. This allows the truly numeric data to be analyzed by PROC MEANS or PROC UNIVARIATE. It makes use of the SAS dictionary tables, the SELECT INTO syntax, and the ANYALPHA function.

INTRODUCTION

When working with credit data there are very few fields that are really character data, but often data sets are received with numeric data stored as character. Even though the files have the same fields, the data type can change from one month to the next for any given field and that needs to be checked every month. To be able to use PROC MEANS or PROC UNIVARIATE the data must be converted to numeric. This macro will identify them and perform the conversion.

The majority of the fields that are truly character data are Y/N type fields that can easily be converted to 1/0 without losing any information. This macro will identify and convert those as well.

The only parameters are lib for the library and dsn for the data set name. A future possible enhancement will be the addition of an exclude list if there are any fields (such as an account number) that is strictly numbers but should not be converted.

USING SASHELP DATA TO IDENTIFY THE CHARACTER FIELDS

The SASHELP.VCOLUMN table has metadata with one row for every field in every SAS data set in every library in use by the current SAS session. A temporary data set is created with only the rows about fields in the specific data set indicated by the macro parameters. It is limited to just those rows with type = ‘char’.

```
data temp_vars;
  set sashelp.vcolumn(keep=libname memname type length name);
  if libname=upcase("&lib") and memname=upcase("&dsn") and type='char';;
run;
```

PROC SQL is used to get the count of such fields and uses SELECT INTO: syntax to store it in a macro called num_char. SELECT INTO: is also used to store the names of the fields in a series of macro variables called c_var1 to c_var9999999. Despite the large number, SAS will only generate as many macro variables as it actually needs. The number provided should be larger than what is really needed because SAS will stop at that number if it doesn’t run out first.

```
proc sql noprint;
  select count(name) into :num_char from temp_vars;
  select name into :c_var1 thru :c_var9999999 from temp_vars;
quit;
```

EVALUATE THE DATA FOR THE EXISTANCE OF LETTERS

A temporary data set is created from the original data set with only those fields that are stored as character. A %do loop from i=1 to &num_char is used to generate the KEEP= statement.

```
data temp_char;/*only char vars here*/
  set &lib..&dsn(keep= %do i=1 %to &num_char;
    &c_var&i
  %end;)
```


The ANYALPHA function returns the position of the first letter it finds and a 0 if it finds none. This is stored in a series of new fields.

```
%do i=1 %to &num_char;
    &c_var&i.._i=anyalpha(&c_var&i);
%end;
run;
```

PROC SQL is used to create a new data set with the sum of each of the new fields across the whole data set. If the ANYALPHA function returned 0 for all records then the sum will be 0 and this indicates the field can be converted to numeric.

```
proc sql noprint;
    create table char_sums as
    select
        %do i=1 %to &num_char;
            sum(&c_var&i.._i) as &c_var&i,
        %end;
    1 as junk
    from temp_char;
quit;
```

PROC TRANSPORT rotates the data set produced by PROC SQL so that the field names are now in one column and the sums are in another.

```
proc transpose data=char_sums out=temp_names;
run;
```

Two data sets are created, one with the field names that are numeric and can be converted and those that are character and need more work.

```
data num_names char_names;
    set temp_names;
    if _name_ ne 'junk';
        if col1=0 then output num_names;
        else output char_names;
    drop col1;
run;
```

**REEXAMINE THE TRUE CHARACTER DATA FOR Y/N FIELDS**

Those fields that really contain character data are then analyzed to see which ones are just Y/N that can be converted to 1/0.

Once again PROC SQL and the SELECT INTO: syntax are used to create a new series of macros with just those that are true character data.

```
proc sql noprint;
    select count(_name_) into :num_tchar from char_names;
    select _name_ into :tc_var1 thru :tc_var9999999 from char_names;
quit;
```

A temporary data set is created with just the true character data fields.

```
data temp_YNs;
    set temp_char(keep= %do i=1 %to &num_tchar;
        &tc_var&i
    %end;
);`
Those fields are checked for being Y/N or null. New fields are created that are indicators. If Y/N/null is all that is found a 0 is stored, otherwise a 1 is stored.

```%
%do i=1 %to &num_tchar;
&tc_var&i.._i=&tc_var&i not in ('Y', 'N', ''); /*if it is Y/N/null I want 0 so I can sum up and find columns that total 0*/
%end;
run;
```

PROC SQL is used to create a new data set with the sums for each true character field and PROC TRANSPOSE is once again used to rotate the data set.

```proctime
proc sql noprint;
create table YN_sums as
select %do i=1 %to &num_tchar;
    sum(&&tc_var&i.._i) as &&tc_var&i,
%end;
1 as junk
from temp_yns;
quit;
proc transpose data=YN_sums out=temp_names;
run;
```

Two data sets are created, one with the list of names that are Y/N and one with the true character data that cannot be converted.

```proctime
data YN_names char_names;
set temp_names;
if _name_ ne 'junk';
if col1=0 then output YN_names;
else output char_names;
drop col1;
run;
```

**REPORTING FIELDS THAT CANNOT BE CONVERTED**

PROC SQL is used to see if there are any fields that cannot be converted and stores the field names in a series of macro variables.

```proctime
proc sql noprint;
select count(_name_) into :issues from char_names;
select _name_ into :f_var1 thru :f_var9999999 from char_names;
quit;
```

If there are any, then a message listing them is written to the log.

```%if &issues %then %do;
    data _null_; 
    put '*******************************';
    put '*******************************';
    put '*******************************';
    put 'there are issues with char data';
    %do i=1 %to &issues;
        put "&&f_var&i";
    %end;
    put '*******************************';
    put '*******************************';
    put '*******************************';
    run;
%end;```
PREPARING TO CONVERT THE DATA

The names of the numeric fields that will be converted are merged into one dataset with the metadata from SASHELP.VCOLUMN. The length of the character data is needed to have an idea of what informat to apply with the INPUT function.

```sas
proc sort data=temp_vars;
   by name;
run;

proc sort data=num_names;
   by _name_;  
run;

data temp;
   merge num_names (in=want rename=(_name_=name))
      temp_vars (keep=name length);
   if want;
      by name;
run;
```

PROC SQL is used to count how many fields will be converted from character to numeric and save the names and lengths of those fields in a series of macros. The number of fields that will be converted from Y/N to 1/0 are similarly counted and those names are stored in a separate series of macros.

```sql
proc sql noprint;
   select count(name) into :num_c2n from temp;
   select name   into :c2n_var1 thru :c2n_var9999999 from temp;
   select length into :c2n_len1 thru :c2n_len9999999 from temp;
   select count(_name_) into :num_yn from yn_names;
   select _name_ into :yn_var1 thru :yn_var9999999 from yn_names;
quit;
```

DATA CONVERSION

The original data set is SET, and a RENAME= statement is generated for all the fields being converted so the new version of the data set will have the same field names as before, only the data type will have changed.

```sas
data &lib..&dsn;
   set &lib..&dsn (rename=(
      %do i=1 %to &num_c2n;
      &&c2n_var&i = temp&i
      %end;
      /*y/n's*/
      %do i=1 %to &num_yn;
      &&yn_var&i = temp_yn_&i
      %end;
   ));

   /*y/n's*/
   %do i=1 %to &num_yn;
      &&yn_var&i = temp_yn_&i
   %end;
));
```

The last step is to write the conversion code. Those fields going from character to numeric directly use an INPUT statement.

```sas
%do i=1 %to &num_c2n;
   &&c2n_var&i = input(temp&i,&&c2n_len&i...);
%end;
```
The Y/N fields are converted with simple IF/THEN/ELSE logic.

```sas
%do i=1 %to &num_yn;
  if temp_yn_&i="Y" then &&yn_var&i =1;
  else if temp_yn_&i="N" then &&yn_var&i =0;
  else if temp_yn_&i=""  then &&yn_var&i =.;
%end;

drop temp;;
run;
```

CONCLUSION

This macro results in the original data set having few character fields, if any, those fields will be listed in the log, and will have at least one record with character data. It does this through the use of the SASHHELP tables to identify potential fields, the use of SELECT INTO: to take data values and store them in macros, and the ANYALPHA function to check the data for the presence of letters.

COMPLETE CODE

```sas
%macro conversion(lib,dsn);

data temp_vars;
  set sashelp.vcolumn(keep=libname memname type length name);
  if libname=upcase("&lib") and memname=upcase("&dsn") and type='char';;
run;

proc sql noprint;
  select count(name) into :num_char from temp_vars;
  select name into :c_var1 thru :c_var9999999 from temp_vars;
quit;

data temp_char;/*only char vars here*/
  set &lib..&dsn(keep=
    %do i=1 %to &num_char;
      &&c_var&i
    %end;
  )
/*first pass, look for fields that are all numbers*/
/*create a 1/0 indicator as to if the record contains a letter or not*/
  %do i=1 %to &num_char;
    &&c_var&i.._i=anyalpha(&&c_var&i);
  %end;
run;
/*sum the columns*/
proc sql noprint*;
  create table char_sums as
    select
      %do i=1 %to &num_char;
        sum(&&c_var&i.._i) as &&c_var&i,
      %end;
    1 as junk
    from temp_char;
quit;

proc transpose data=char_sums out=temp_names;
run;
/*sort out those that are all num from those that need more work*/
data num_names char_names;
  set temp_names;
  if _name_ ne 'junk';
```

```
if col1=0 then output num_names;
else output char_names;
drop col1;
run;

/*for those that are char, find the ones that are only Y/N/null*/
proc sql noprint;
select count(_name_) into :num_tchar from char_names;
select _name_ into :tc_var1 thru :tc_var9999999 from char_names;
quit;
data temp_YNs;
set temp_char(keep=
%do i=1 %to &num_tchar;
&&tc_var&i
%end;);
%do i=1 %to &num_tchar;
&&tc_var&i.._i=&&tc_var&i not in ('Y', 'N', '')
%end;
run;

proc sql noprint;
create table YN_sums as
select %do i=1 %to &num_tchar;
sum(&&tc_var&i.._i) as &&tc_var&i,
%end;
1 as junk
from temp_yns;
quit;
proc transpose data=YN_sums out=temp_names;
run;
/*sort out those that are Y/N from those that can't be converted*/
data YN_names char_names;
set temp_names;
if _name_ ne 'junk';
if col1=0 then output YN_names;
else output char_names;
drop col1;
run;
/*see if there are any fields that are really unexpected*/
proc sql noprint;
select count(_name_) into :issues from char_names;
select _name_ into :f_var1 thru :f_var9999999 from char_names;
quit;
%if &issues %then %do;
data _null_; put '*******************************';
put '*******************************';
put '*******************************';
put 'there are issues with char data';
%do i=1 %to &issues;
put "&f_var&i";
%end;
put '*******************************';
put '*******************************';
put '*******************************';
run;
%end;
proc sort data=temp_vars;
  by name;
run;

proc sort data=num_names;
  by _name_; 
run;

data temp;
  merge num_names (in=want rename=(_name_=name))
    temp_vars (keep=name length);
  if want;
    by name;
run;

proc sql noprint;
  select count(name) into :num_c2n from temp;
  select name into :c2n_var1 thru :c2n_var9999999 from temp;
  select length into :c2n_len1 thru :c2n_len9999999 from temp;
  select count(_name_) into :num_yn from yn_names;
  select _name_ into :yn_var1 thru :yn_var9999999 from yn_names;
quit;

/*convert char to numeric where applicable*/
data &lib..&dsn;
  set &lib..&dsn (rename=(
    %do i=1 %to &num_c2n;
      &&c2n_var&i = temp&i
    %end;
    /*y/n's*/
    %do i=1 %to &num_yn;
      &&yn_var&i = temp_yn_&i
    %end;
  ));
  %do i=1 %to &num_c2n;
    &&c2n_var&i = input(temp&i,&&c2n_len&i...);
  %end;
  /*y/n's*/
  %do i=1 %to &num_yn;
    if temp_yn_&i="Y" then &&yn_var&i =1;
    else if temp_yn_&i="N" then &&yn_var&i =0;
    else if temp_yn_&i="" then &&yn_var&i =.;
  %end;
  drop temp.;
run;
%mend;

REFERENCES
CONTACT INFORMATION

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

Name: Andrea Wainwright-Zimmerman
Email: AndreaNWZimmerman@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.