

Automated Data Converting of Character into Numeric Fields

Mila Chiflikyan, RTI International, Research Triangle Park, NC

Nick L Kinsey, RTI International, Research Triangle Park, NC

Ruben Chiflikyan, RTI International, Research Triangle Park, NC

Valentina Grouverman, RTI International, Research Triangle Park, NC

ABSTRACT

Providing high quality numeric data to be used in data analysis is a crucial component in data delivery using the SAS® system. However, because of many data transformations using various systems including scanning from paper forms, exporting to Excel and importing back, the data values often become 'contaminated' and the variables that previously contained only numeric values can appear as character variables in the final SAS datasets. After the obligatory data cleaning, one needs to check whether all values for any eligible variables does not contain alpha characters and then to convert these variables to the appropriate numeric format. This is a time-consuming process for the developer to complete, especially for datasets containing hundreds or even thousands of variables. To overcome these difficulties, we have created a highly efficient time-saving automated program for checking variables eligibility for conversion from character to a numeric data type and converting each eligible variable while maintaining all variables in the final SAS datasets in their original variable order.

KEYWORDS: macros, macro variables, macro functions, arrays, data cleaning, PROC SQL, PROC TRANSPOSE, PROC IMPORT.

INTRODUCTION

Very often SAS has to process data that is originated from an EXCEL, ACCESS or other data sources.

When using the PROC IMPORT procedure, the data is scanned and automatically read as a character or numeric fields. If the data was not cleaned before importing, many numeric fields could be treated as character fields, and this is not what a client wants to see. Thus, a first step should involve a thorough data cleaning. For data sets containing many hundreds of variables, reviewing all values for each variable and manually converting them to numeric fields would be a time-consuming process. To overcome these difficulties, we propose an efficient approach that automates the process of reviewing all variable values. If all the values for a specific variable are numeric, but for some reason have embedded symbols, e.g., blanks, special symbols, etc., then the original character field will be converted to numeric. Additionally, since temporary variables have been used when performing these transformations, one has to bring them in the original order.

In our example, there are few records, where decimal numbers are present with imbedded blanks, and it causes the variable to be defined as a character data type. We are providing two different programs to handle this situation. Before converting data to the numeric fields, the data in both programs is cleaned. For the data conversion, the first program uses the PROC TRANSPOSE procedure to list all the character fields in the form of case of identifiers, variable names, and their values. Then, the program retains a flag that indicates whether any alpha values were found in the data, and the process automatically creates temporary numeric fields, assigns the values of the original fields to the temporary fields and lastly renames them to the original names. Since the renaming process causes variables to change original variable order, the program also uses the PROC SQL procedure, macro processing and output from the PROC CONTENTS procedure to reconstruct the original order of the variables. In the second program, we are providing a more compact solution using macro strings instead of creating temporary text files. Both programs have the same functionality.

DESCRIPTION OF THE FIRST PROGRAM

We demonstrate the efficiency of this program using a subset of the raw data that was collected using paper and pencil method. Below, you can see the sample that was read from EXCEL (Figure 1.). As one sees from

the data dump, there was a problem with the initial data scanning. Sometimes, because of certain technical malfunctions of the scanner or poor quality of the paper document, specific types of systematic errors are produced. In this particular case, the scanner quite often imbedded a blank character after the decimal point, e.g., '5.1' is transformed to '5. 1'.

Caseid	Q1a	Q1b	Q1c	Q3b	Q4	Q5	Q7	Q10a	Q10b
1	5. 1	4. 1	4 .1	1. 0		0. 5	6. 5	20	3. 0
2	5	5	5	13	1. 3	3	6.5	120	50
3		5	4	13	13	3	6. 5	120	50
4	3	3. 02	2	10	2	0	8	.	
5	4	4	2	9	0	1	7	180	6 0

Figure 1. The dump of the data imported from an external data source into a SAS data set.

The PROC CONTENTS procedure in Figure 2 confirms that many variables that are expected to be numeric are imported into SAS as character data types.

#	Variable	Type	Len	Format	Informat	Label
1	Caseid	Num	8			Caseid
2	Q1a	Char	4	\$4.	\$4.	Q1a
3	Q1b	Char	4	\$4.	\$4.	Q1b
4	Q1c	Char	4	\$4.	\$4.	Q1c
5	Q3	Char	4	\$4.	\$4.	Q3
6	Q4	Char	4	\$4.	\$4.	Q4
7	Q5	Char	4	\$4.	\$4.	Q5
8	Q7	Char	4	\$4.	\$4.	Q7
9	Q10a	Num	8			Q10a
10	Q10b	Char	4	\$4.	\$4.	Q10b

Figure 2. The contents of data imported from an external data source into a SAS data set.

Below are four temporary files which are used for writing and reading SAS code outputs while the program is executing.

```
filename outlist '<path>\temp.lst';
filename out_drop '<path>\temp.txt';
filename outrn '<path>\num.txt';
filename out '<path>\sas_code.txt';
```

We do not want to make any modifications to the *ID* (case of identifier) variable. Later in the program, the ID variable in the PROC TRANSPOSE procedure will be used, and the user is requested to enter the name of the actual *ID* variable name instead of "CASEID".

```
%let id=CASEID;
libname lib '<path>';
%let indsn=lib.raw ; /*input data set*/
%let outdsn=lib.final ; /*output data set*/
%let dsn=lib.clean ; /*cleaned data set*/
```

As was mentioned earlier, the scanned data is imported into a SAS data set using the standard PROC IMPORT procedure as follows:

```
proc import out= &indsn datafile= "<path>\raw.xls"
dbms=excel2002 replace;
getnames=yes; mixed=yes; run;
```

The raw data was collected using paper and pencil, scanned, and exported into an EXCEL spreadsheet. As mentioned above, there were problems with the initial data scanning. The following data step checks to see if a field is numeric and then removes any imbedded blanks.

```
data &dsn ;
```

```

set &indsn;
/*any needed cleaning steps could be performed here*/
array ch _character_;
do over ch;
if anyalpha(ch)=0 then
ch=compress(ch,' ');
end; run;

```

After the appropriate cleaning is complete, removing the embedded blanks in this case, the data set takes the following form:

Caseid	Q1a	Q1b	Q1c	Q3b	Q4	Q5	Q7	Q10a	Q10b
1	5.1	4.1	4.1	1.0		0.5	6.5	20	3.0
2	5	5	5	13	1.3	3	6.5	120	50
3		5	4	13	13	3	6.5	120	50
4	3	3.02	2	10	2	0	8	.	
5	4	4	2	9	0	1	7	180	60

Figure 3. The embedded blanks are removed from data fields.

After data cleaning, the eligible character variables will need to be converted into numeric variables. The CHAR_TO_NUM macro checks the values for each variable and converts character variables to numeric variables if all values for that variable are numeric.

```

proc sort data=&dsn;
by &id; run;

```

The PROC TRANSPOSE procedure below transposes the values for each of the character variables.

```

proc transpose data= &dsn out=tr prefix=rt;
var _character_;
by &id; run;

```

Caseid	_NAME_	rt1
1	Q1a	5.1
1	Q1b	4.1
1	Q1c	4.1
1	Q3b	1.0
1	Q4	
1	Q5	0.5
1	Q7	6.5
1	Q10b	3.0
2	Q1a	5
2	Q1b	5
2	Q1c	5
2	Q3b	13
2	Q4	1.3

Figure 4. The data restructured by the PROC TRANSPOSE procedure.

For data review purposes, the transposed data is sorted by the variable name and variable value. To facilitate the time needed to review the data, we remove duplicate values for every variable from the transposed data set.

```

proc sort data=tr nodupkey; by _name_ rt1; run;

```

In the following DATA step, the transposed data is grouped by the variable name for reviewing data for each variable.

```

data chk;

```

```

set tr;
by _name_ ;
if _name_="&id" then delete;
/*id should be excluded from the processing*/
retain x ;
/*x is a flag that is set to '1' if any alpha characters were found in the
reviewed variable */

if first._name_ then x=0;

/*if alpha is found the flag is set to '1'*/
if anyalpha(rt1)>0 then x=1; run;

```

When the last value for the specified variable is reached, the program checks whether the flag for the reviewed variable remains 0 (no alpha values were found).

If no alpha characters were found for a variable, then the conversion of this character variable into a numeric variable takes place. The SAS statements below initialize the new variable to zero and set the value of the newly created numeric variable to the value of original variable.

```

if last._name_ then do; /* the last line for the specified variable */
_name_=trim(left(_name_));

if x=0 then do; /*check if no alpha values were found*/

file out lrecl=80 ;
put @1 'N_' @3 _name_ '=0 ;' ;
/*the new variable name has the same name with 'N_' prefix */
put @1 'N_' @3 _name_ @20 '=' @23 _name_ ';' ;

```

The following SAS statements are written to the text file reference OUT:

```

N_Q10b=0; N_Q10b=Q10b; N_Q1a=0; N_Q1a=Q1a; N_Q1b=0;
N_Q1b=Q1b; N_Q1c=0; N_Q1c=Q1c; N_Q3b=0; N_Q3b=Q3b;
N_Q4=0; N_Q4=Q4; N_Q5=0; N_Q5=Q5;

```

Afterwards, the SAS statements that drop original variable names are written to the text file reference *out_drop*, while the SAS statements which rename the newly created numeric variables to the original variable names are written to the *outrn* text file.

```

file out_drop lrecl=80;
if _n_=1 then
put @1 'drop' ;
put _name_;
file outrn lrecl=80 ;
put @1 'N_' @3 _name_ @20 '=' @23 _name_ ;
end;
end;

```

This is the sample list from *out_drop* data set:

Q10b Q1a Q1b Q1c Q3b Q4 Q5.

And this is the sample list from *outrn*:

```

N_Q10b = Q10b N_Q1a = Q1a N_Q1b = Q1b N_Q1c = Q1c
N_Q3b = Q3b N_Q4 = Q4 N_Q5 = Q5.

```

In the data step below, the data conversion takes place. The cleaned data set is read in, the fields that will become numeric data fields are dropped, and the temporarily created variables are set to the values of the corresponding character variables, thus performing the SAS conversion from character fields to numeric fields.

```

data upd;
set &dsn;
drop
%include out_drop;

```

```
%include out;
```

The output results can be seen in Figure 5.

Caseid	Q10a	N_Q10b	N_Q1a	N_Q1b	N_Q1c	N_Q3b	N_Q4	N_Q5
1	20	3	5.1	4.10	4.1	1	.	0.5
2	120	50	5.0	5.00	5.0	13	1.3	3.0
3	120	50	.	5.00	4.0	13	13.0	3.0
4	.	.	3.0	3.02	2.0	10	2.0	0.0
5	180	60	4.0	4.00	2.0	9	0.0	1.0

Figure 5. The following intermediate data set has temporary N_* variables.

In the next data step, the temporary N_* variables are renamed to the original variable names that are preserved in *outrn* text data set.

```
data &dsn._upd; set upd; rename %include outrn; run;
```

Figure 6 illustrates the contents of the resulting data set.

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Label
1	Caseid	Num	8	Caseid
8	Q4	Num	8	Q4
9	Q5	Num	8	Q5
10	Q7	Num	8	Q7
2	Q10a	Num	8	Q10a

Figure 6. All character variables containing only numeric data are converted to a numeric data type.

The output SAS data set has all the needed character variables converted to numeric, however the variables are not placed in their original order (see Figure 7).

Caseid	Q10a	Q10b	Q1a	Q1b	Q1c	Q3b	Q4	Q5
1	20	3	5.1	4.10	4.1	1	.	0.5
2	120	50	5.0	5.00	5.0	13	1.3	3.0
3	120	50	.	5.00	4.0	13	13.0	3.0
4	.	.	3.0	3.02	2.0	10	2.0	0.0
5	180	60	4.0	4.00	2.0	9	0.0	1.0

Figure 7. Variables are out of original order.

Since the variables are reordered because of the renaming process, PROC CONTENTS and PROC SQL procedures were used to place all variables in the data set into their original data set order.

```
proc contents data=&dsn noprint out= contents; run;
```

Figure 8 illustrates the dump of the contents data set that was created from the original data set.

NAME	TYPE	VARNUM
Caseid	1	1
Q4	2	6
Q5	2	7
Q7	2	8
Q10a	1	9
Q10b	2	10
Q1a	2	2
Q1b	2	3
Q1c	2	4

Q3b	2	5
-----	---	---

Figure 8. Output from proc contents - the original variables order.

Below the macro variable named `&str`, containing the comma-separated variable names in their original data set order, is created:

```
proc sort data=contents; by varnum;

proc sql noprint;
select name into : str separated by ',' from contents;
```

In the final step, after reading the data set with the converted numeric variables, placing it in original data set order, and using the SELECT statement and the macro variable named `&str =Caseid, Q1a, Q1b, Q1c, Q3b, Q4, Q5, Q7, Q10a, Q10b`, the final SAS data set shown in Figure 9 is created:

```
proc sql;
create table &outdsn as
select &str from &dsn._upd;
```

Caseid	Q1a	Q1b	Q1c	Q3b	Q4	Q5	Q7	Q10a	Q10b
1	5.1	4.10	4.1	1	.	0.5	6.5	20	3
2	5.0	5.00	5.0	13	1.3	3.0	6.5	120	50
3	.	5.00	4.0	13	13.0	3.0	6.5	120	50
4	3.0	3.02	2.0	10	2.0	0.0	8.0	.	.
5	4.0	4.00	2.0	9	0.0	1.0	7.0	180	60

Figure 9. The previous order is reconstructed.

As one can see, the eligible character fields are converted to numeric fields and all the variables are now in their original order.

THE SECOND APPROACH

This program is based on using macro strings instead of creating temporary files. Note that this approach has the same functionality as the first one. To be independent from the first approach, we used a different sample for testing of this program.

```
libname lib 'Z: ';
* sample data;
data one;
infile datalines;
input @1 caseid $1.
      @3 Q1a    $2.
      @6 Q1b    $2.
      @9 Q4     $2.
      @12 Q5    4.
      @17 Q6    4.
;
datalines;
1 10 13 14 12 14
2 14 qt 5 15 56
3 18 2 4 78 65
4 19 1 4a 56 1
5 50 2 4 45 2
;
```

Firstly, by using the OPEN function, the contents of the *one* data set is retrieved and the information about variable names and their types is written to the *allnames* and *type* variables, accordingly.

```

data one_t;
length type $1. allnames $8;
drop dsid i num ;
dsid=open("work.one","i");
num=attrn(dsid,"nvars");
do i=1 to num; *num is the number of vars;
type= vartype(dsid,i);
allnames=varname(dsid,i);
output;
end;
run;

```

The dump of the *one_t* data set is displayed in Figure 10.

type	allnames
C	caseid
C	Q1a
C	Q1b
C	Q4
N	Q5
N	Q6

Figure 10. The newly created data set has the list of all variables and their types.

To proceed further, the numeric variables and character variables are split into two data sets.

```

data one_u_char one_u_num(rename=(allnames=name3));
set one_t;
if type='C' then output one_u_char;
else if type='N' then output one_u_num;
drop type; run;

```

The macro variable *&obs* contains the number of character variables in the data set.

```

data _null_;
call symput('obs',obs);
set one_u_char nobs=obs;
stop; run;

```

The number of observations in the original data set will be kept in the macro variable named *&f*.

```

data _null_;
call symput('f',obs);
set one nobs=obs;
stop; run;

```

Here the *name2* variable is created by taking the original variable name and adding the 'g' prefix to each of variable values.

```

data one_b;
set one_u_char;
name1=allnames;
name2='g'||name1;
drop allnames;
run;

```

A dump of the data is shown in Figure 11.

name1	name2
caseid	gcaseid
Q1a	gQ1a
Q1b	gQ1b

Q4	gQ4
-----------	------------

Figure 11. Renaming of the original variables.

Next, we create the following four macro variables with variable names separated by a blank character:

- (1) *&allnames* has the string with all original variable names,
- (2) *&name1* has the string of all character variables,
- (3) *&name2* has the string of *&name1* with 'g' character added in front of the variable name, and
- (4) *&name3* has the string of all numeric variable names.

The output from the PROC CONTENTS procedure is used to preserve the original order. The 'one' data set is the original data set with the correct order of the variables.

```
proc contents data=one noprint out= contents ; run;
```

The *varnum* variable is created by PROC CONTENTS procedure and contains the order of the variable in the data set. When we sort the contents data set by the *varnum*, the list of macro variables listed sequentially in the original order is created:

```
proc sort data=contents; by varnum; run;

proc sql noprint;
select name
into : allnames separated by ' '
from contents;
select name1, name2
into : name1 separated by ' ',
: name2 separated by ' '
from one_b;
select name3
into : name3 separated by ' '
from one_u_num;
```

The code above creates four macro strings, each of them contains a list of variables in the specified order. Here is the description of their content:

- (1) macro variable *&allnames* resolves to *caseid Q1a Q1b Q4 Q5 Q6* – original data set order,
- (2) macro variable *&name1* resolves to *caseid Q1a Q1b Q4* – the list of all character variables,
- (3) macro variable *&name2* resolves to *gcaseid gQ1a gQ1b gQ4* – the list of all temporarily created variables, and
- (4) macro variable *&name3* resolves to *Q5 Q6* – list of all numeric variables.

The next step creates the flags for the numeric and character variables.

```
data two;
set one(drop=&name3);
%let d=&obs; *&obs which has # of character variables in the data set;

array xxx(&obs) $ &name1; /*array of all character variables*/
array yyy(&obs) y1-y&d; /*sequential flags for each variable */

do I=1 to &obs;
if anyalpha(yyy(I))=0 then yyy(I)=1;
else yyy(I)=0;
end; drop I; run;
```

If the variable has a numeric value, then the corresponding flag is set to 1. The listing of the resulting data is displayed in Figure 12.

caseid	Q1a	Q1b	Q4	y1	Y2	y3	y4
1	10	13	14	1	1	1	1
2	14	qt	5	1	1	0	1
3	18	2	4	1	1	1	1

4	19	1	4a	1	1	1	0
5	50	2	4	1	1	1	1

Figure 12. The flags are set for each value indicating numeric(1) and character(0) fields.

```
proc transpose data=two out=three; run;
```

Since only the numeric fields are selected automatically by the PROC TRANSPOSE procedure, the numeric flags are processed by this procedure (Figure 13).

NAME	COL1	COL2	COL3	COL4	COL5
y1	1	1	1	1	1
y2	1	1	1	1	1
y3	1	0	1	1	1
y4	1	1	1	0	1

Figure 13. The data is transposed for reviewing of all the values in each variable.

```
* assigning 1 or 0 to the flag: 1-numeric, 0-character;
data four(keep=flag);
set three;
%let g=&f; *&f has # of obs in the original data set;

if sum(of col1-col&g)=&g then flag=1;
else flag=0;
```

By adding all the columns and comparing them to the total number of observations in the data set, this step checks if all the values for the selected variable are numeric. Figure 14 displays the final *flag* in the character variables order.

flag
1
1
0
0

Figure 14. The flag variable is set to '1' if no alpha values are found.

Then the program creates the macro variable *&flag* that contains the list of all the flags separated by blank.

```
proc sql noprint;
select flag
into :flag separated by ' '
from four;
quit;

%macro tty;
data one_e;
set one;
%do I=1 %to &obs;
%if %scan(&flag,&I,' ')=1 %then %do;
%scan(&name2,&I,' ')=0;
%scan(&name2,&I,' ')=input(%scan(&name1,&I,' '),4.);
%end;
%else
%if %scan(&flag,&I,' ')=0 %then %do;
%scan(&name2,&I,' ')=%scan(&name1,&I,' ');
%end;
%end;
```

```

%end;
drop &name1;
run;

%mend tty;
%tty;

```

This macro assigns corresponding values to the 'g' fields and converts character variables to numeric variables when all the values for the variable are numeric. The contents of the created data set are displayed in Figure 15.

#	Variable	Type	Len
1	Q5	Num	8
2	Q6	Num	8
6	gQ4	Char	2
4	gQ1a	Num	8
5	gQ1b	Char	2
3	gcasid	Num	8

Figure 15. The 'g' variables need to be renamed.

The *trtr* macro below renames the newly created 'G' variables to their original names.

```

%macro trtr;
  rename
  %do I=1 %to &obs;
    %scan(&name2,&I,' ')=%scan(&name1,&I,' ')
  %end;
  ;
run;
%mend trtr;

data fin1;
set one_e ;
%trtr;
run;

proc sql;
create table fin as
select &allnames
from fin1;
quit;

```

This step puts all the variables from the converted data set into its original order. The macro variable *&allnames* = *Caseid,Q1a,Q1b,Q4,Q5,Q6* contains a list of all the comma separated variables from the original data set in their original order. The content of the final data set is displayed in the Figure 16.

Variables in Creation Order			
#	Variable	Type	Len
1	caseid	Num	8
2	Q1a	Num	8
3	Q1b	Char	2
4	Q4	Char	2
5	Q5	Num	8
6	Q6	Num	8

Figure 16. The final data set.

CONCLUSION

RTI has developed an approach to automate the checking of variables eligibility for character-to-numeric data conversion and performing the corresponding data conversion by preserving of the original variable's type and order. Technically, this approach is implemented in 2 different SAS programs having the same functionality. Easily integrated into any SAS data process, these time and cost-saving programs will significantly improve data accuracy and are therefore very helpful during the data analysis stage. These two programs have been successfully used on a number of projects to deliver high quality data to our clients.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Mila Chiflikyan
Research Triangle Institute, International
3040 Cornwallis Road, PO Box 12194
Research Triangle Park, NC, 27709-2194
Work Phone: (919)316-3360
Fax: (919)541-6178
E-mail: milachif@rti.org
Web: <http://www.rti.org>

Nick L Kinsey
Research Triangle Institute, International
3040 Cornwallis Road, PO Box 12194
Research Triangle Park, NC, 27709-2194
Work Phone: (919) 541-6791
Fax: (919)541-6178
E-mail: nlk@rti.org
Web: <http://www.rti.org>

Ruben Chiflikyan
Research Triangle Institute, International
3040 Cornwallis Road, PO Box 12194
Research Triangle Park, NC, 27709-2194
Work Phone: (919)541-6064
Fax: (919)541-6178
E-mail: rchiflikyan@rti.org

Valentina Grouverman
Research Triangle Institute, International
3040 Cornwallis Road, PO Box 12194
Research Triangle Park, NC, 27709-2194
Work Phone: (402)261-3989
Fax: (919)541-6178
E-mail: vgrouvermann@rti.org

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.