Renaming SAS® Variables

Imelda C. Go, South Carolina Department of Education, Columbia, SC

ABSTRACT

This paper discusses a number of ways to rename variables. Its topics include the RENAME statement used in DATA steps, the RENAME= data set option, the AS keyword for PROC SQL, using macros, and using the DATA _NULL_ step.

A "quick and dirty" way to change the name of a variable is shown below. The variable \( x \) is assigned to variable \( y \) and \( x \) is dropped from the data set. This is not really renaming a variable because it involves creating a copy of the variable to be renamed and the copy has the desired new name. This practice, although it achieves the desired result, is not optimal.

```sas
data two;
  set one;
  y=x;
  drop x;
```

Changing a variable name is an inevitable fact of programming. There are several ways to accomplish this depending on the types of SAS statements involved. Consider the following SAS code.

```sas
data one;
  input x z;
  y=x+1;
  cards;
  1 2
;
```

The variables acquire their names through the INPUT statement (variables \( x \) and \( z \) in data set \( one \)), an assignment statement (variable \( y \) in data set \( one \)), or automatically in procedures when the user does not specify variable names. It is important to be aware of a PROC's features and how a PROC might accommodate variable name changes.

```sas
proc means data=one n mean;
  var x y;
  output out=stats;
```

The PROC MEANS results are:

The MEANS Procedure

<table>
<thead>
<tr>
<th>Variable</th>
<th>N</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1.0000000</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>2.0000000</td>
</tr>
</tbody>
</table>

Data set \( stats \) is shown below. The \( n \) and \( mean \) values are listed under variables named after the corresponding analysis variable. The type of statistic in each observation is indicated by the \_STAT\_ variable.

```
Obs _TYPE_ _FREQ_ _STAT_  x  y
1   0   1   N   1 1
2   0   1   MIN 1 2
3   0   1   MAX 1 2
4   0   1   MEAN 1 2
5   0   1   STD  . .
```

Suppose that the following code is used instead.

```sas
proc means data=one;
  var x y;
  output out=stats
    n=countx county
    mean=avex avey;
```
The results are shown below and even the data set structure differs from the previous one due to the user-specified options.

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>countx</th>
<th>county</th>
<th>avex</th>
<th>avey</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**RENAME (DATA Step Statement)**

A straightforward way of renaming a variable is to use the RENAME statement. The syntax for \( n \) variables is:

```plaintext
rename oldvarname1=newvarname1
oldvarname2=newvarname2
...
oldvarnamen=newvarnamen;
```

In the example below, the variable \( x \) is renamed to variable \( y \) and the variable \( z \) is renamed to variable \( a \).

```plaintext
data two;
set one;
rename x=y z=a;
```

**RENAME= (The Data Set Option)**

Another way of renaming the variables is to use the RENAME option with the DATA option in a SET statement.

```plaintext
data two;
set one (rename=(x=y z=a));
```

Procedures provide default names for variables in output data sets.

```plaintext
proc freq data=one;
tables x/out=xcounts;
```

By default, count is the variable name for the frequencies.

```
The FREQ Procedure

<table>
<thead>
<tr>
<th>x</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
</tbody>
</table>
```

Suppose the default name is not satisfactory, one can do the following.

```plaintext
data xcounts;
set xcounts;
rename count=n;
```

However, the variable count can be renamed within PROC FREQ using the RENAME= data set option.

```plaintext
proc freq;
tables x/out=xcounts (rename=(count=n));
```
PROC SQL

The AS keyword can be used in PROC SQL to change the variable name or assign a name to a computed value. In the example below, several variables are renamed and the difference of `group1.ss-group2.ss` is named `ssdiff`.

```
proc sql;
   select group1.grade, group1.name as name1,
       group2.name as name2, group1.ss as ss1,
       group2.ss as ss2, sex,
       group1.ss-group2.ss as ssdiff
   from group1, group2
   where group1.grade=group2.grade and
       group1.lunch=group2.lunch;
```

Sample results are shown below:

```
GRADE  NAME1            NAME2           SS1   SS2  SEX    SSDIFF
----------------------------------------------------------------
 2      Taylor, Liz      Ryan, Meg       245   234  F          11
```

Macros Can Facilitate Renaming

One can also use a macro to rename variables that are part of a range, such as `x1-x100`.

```
%macro rename;
   rename
   %do i=1 to 100;
      x&i=y&i
   %end;
%end rename;
```

This macro writes the code to rename `x1-x100` to `y1-y100` respectively. However, the variable names are not always in such a convenient sequence.

Variable Names from PROC CONTENTS Output Data

If the variable names need to be renamed in a predictable manner, data from PROC CONTENTS and the DATA _NULL_ step can be used to create the RENAME statement.

Consider the following data set.

```
data three;
   infile externalfile;
   input x y z f a1-a3;
```

When PROC CONTENTS is applied to a data set and an output data set is created, the output data set will have the variable names of the data set the PROC was applied to. In the example below, data set `varnames` contains the variable names from data set `three`.

```
proc contents data=three out=varnames;
```

There are many other variables in data set `varnames`. The variable characteristics and labels are shown by the PROC CONTENTS output below.

```
###Alphabetic List of Variables and Attributes###
# Variable Type Len Pos Format      Label
--------------------------------------------------------------------------------
32 CHARSET  Char   8 802             Host Character Set
33 COLLATE  Char   8 810             Collating Sequence
28 COMPRESS Char   8 791             Compression Routine
20 CRDATE   Num    8  80 DATETIME16. Create Date
22 DELOBS   Num    8  96             Deleted Observations in Data Set
36 ENCRYPT  Char   8 824             Encryption Routine
19 ENGINE   Char   8 760             Engine Name
27 FLAGS    Char   3 788             Update Flags (Protect
```
### PROC PRINT Output for Data Set `varnames`

The variable names are in a field called `name`.

#### Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Informat</th>
<th>Format</th>
<th>Format Length</th>
<th>Number of Format Decimals</th>
<th>Use of Informat in Indexes</th>
<th>Password Protection</th>
<th>Sorted and/or Validated</th>
<th>Data Set Label</th>
<th>Library Member Name</th>
<th>Library Name</th>
<th>Generation Number</th>
<th>Next Generation Number</th>
<th>Maximum Number of Generations</th>
<th>Sort Option: No Duplicate Keys</th>
<th>Sort Option: No Duplicate Records</th>
<th>Point to Observations</th>
<th>Position of Variable in Sortedby Clause</th>
<th>Special Data Set Type (From TYPE=)</th>
<th>Observations in Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK ONE</td>
<td>a1</td>
<td>1</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>0</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>V8</td>
</tr>
<tr>
<td>WORK ONE</td>
<td>a2</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>40</td>
<td>0</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>40</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>WORK ONE</td>
<td>f</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>24</td>
<td>0</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>24</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>WORK ONE</td>
<td>x</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>WORK ONE</td>
<td>y</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>WORK ONE</td>
<td>z</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>16</td>
<td>V8</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>
Using a DATA _NULL_ Step

Suppose there are pre-test data in a data set and the data set has exactly the same variable names as in data set `varnames` above. The task is to put the suffix of `pre` to all variable names for the pre-test data.

The results of the following DATA _NULL_ step are written in the SAS log.

```sas
data _null_; set varnames end=eof;
if _n_=1 then put "rename ";
newvarname=trim(name)||'pre';
put name '= ' newvarname;
if eof then put ';';
```

The LOG will have the RENAME statement created by the code above.

```
466 data _null_;
467 set varnames end=eof;
468 if _n_=1 then put "rename ";
469 newvarname=trim(name)||'pre';
470 put name '= ' newvarname;
471 if eof then put ';';
```

```sas
rename
a1 = a1pre
a2 = a2pre
f = fpre
x = xpre
y = ypre
z = zpre
;
```

NOTE: There were 6 observations read from the dataset WORK.VARNAMES.

NOTE: DATA statement used:
real time 0.05 seconds

%INFILE Statement

This RENAME statement can be copied from the SAS log into a SAS program. This would not be ideal because of the manual intervention. One may choose to write the results into an external file (`pretest.txt`). Later this external file can be referenced in the program using the %INFILE statement.

```sas
filename pre 'c:\example\pretest.txt';

data _null_;
file pre;
set varnames end=eof;
if _n_=1 then put "rename ";
newvarname=trim(name)||'pre';
if eof then put ';';

data pretest;
set unrenamedpretestdata;
%infile pre;
```

The previous example showed how to use the DATA _NULL_ step to create the SAS statements needed to rename variables. However, the example involved a simple renaming rule where a suffix is systematically added to each old variable name. If the renaming rules are not so simple but a pattern can be detected, then it is a programmable situation.

An alternative to programming the type of renaming described above is to use a data set with the old and new variable names. The data set is then matched by the old variable name with the data set obtained from PROC CONTENTS. After that, a DATA _NULL_ step, similar to the above, can be used to write a RENAME statement for renaming the variables.
REFERENCES


TRADEMARK NOTICE
SAS is a registered trademark or trademark of the SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Imelda C. Go (icgo@juno.com)
South Carolina State Department of Education
Columbia, SC