Advanced Coding Skills: Renaming & Constructing Complex Variables

Hsueh-Sheng Wu
CFDR Workshop Series
May 22, 2017
Outline

• Why do we need to learn better coding techniques?
• Two types of data construction tasks:
  o Transformation on each of several variables
  o Transformation on the format of the data
• Key SAS or Stata commands for the tasks
  o Syntax rules of –array- in SAS
  o Syntax rules of –foreach- and –forvalues- in Stata
• Sample SAS and Stata codes
• Family structure codes using the do … loop commands
• Conclusions
Why Do We Need to Learn Better Coding Techniques?

• Better coding techniques give you:
  o Efficiency: You use fewer lines of codes to complete tasks
  o Accuracy: Fewer lines of codes reduce the possibility of making errors and make it easy to spot coding errors

• We focus on the following tasks on multiple variables:
  o Transform each of several variables
    ▪ Rename variables
    ▪ Recode the values of variables
    ▪ Create new variables
  o Transform the whole data set
    ▪ Change data from wide to long format
    ▪ Change data from long to wide format

• The use of -array- in SAS or -foreach- and –forvalues- in Stata are promising tools for better coding
Syntax Rules of -array- in SAS

• All variables specified within an array must be of the same type

• Variables specified within an array do not need to be already existing variables

• Syntax of specifying an –array-in SAS:

array array-name{n} <$> <length> array-elements;
Syntax Rules of –array- in SAS

• After defining an array, you can specify what to do with each of the elements in the array

• Example:

(1) an one-dimension array, called “number,” contains five numeric variables from number1 to number5. The length of each of these variable has 3 digits

array number{5} 3  number1-number5;
array number{*} 3  number1-number5;
array number{1:5} 3  number1-number5;
array number{5} 3;
# Syntax Rules of `array` in SAS

<table>
<thead>
<tr>
<th>Array Reference</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>number{1}</code></td>
<td><code>number1</code></td>
</tr>
<tr>
<td><code>number{2}</code></td>
<td><code>number2</code></td>
</tr>
<tr>
<td><code>number{3}</code></td>
<td><code>number3</code></td>
</tr>
<tr>
<td><code>number{4}</code></td>
<td><code>number4</code></td>
</tr>
<tr>
<td><code>number{5}</code></td>
<td><code>number5</code></td>
</tr>
</tbody>
</table>
Syntax Rules of -array- in SAS

• Examples:

(2) an one-dimension array, called “character,” contains three string variables from string1 to string3. The length of each of these variable has 2 characters.

array character{3} $ 2 string1-string3;

(3) an two-dimension array, called “season,” contains 12 numeric variables that reflecting twelve months. The length of each of these variable has 1 digit.

array season{4,3} 1 January February March
April May June
July August September
October November December;
Syntax Rules of –foreach- in Stata

• There is no –array- command in Stata
• The -foreach- command has similar functions as one-dimension array in SAS
• Syntax of –foreach-

```
foreach lname {in|of varlist} variables {
    commands referring to `lname'
}
```

• Tell Stata to invoke –foreach-command
• Create an index name to refer to each variable specified
• Specify whether the list of variables are generic variables, existing variables, or new variables
• The list of variables

The open brace indicates the beginning of specifying the list of variables and need to be on the same line as –foreach-

The close brace must appear on a line by itself and signal the end of the –foreach- command

All the data-construction commands that use the variables specified in the –foreach- command should be placed between the open brace and the close brace.
Syntax Rules of \texttt{forvalues} in Stata

- Syntax of \texttt{forvalues}:

\begin{verbatim}
forvalues \textit{iname} range \{ \\
\quad \textit{commands referring to `iname'} \\
\}
\end{verbatim}

- Tell Stata to invoke \texttt{forvalues} command
- Create an index name to refer to each variable specified
- Specify the range of values
- The open brace indicates the beginning of specifying the list of variables and need to be on the same line as \texttt{-forvalues-}.
- The close brace must appear on a line by itself and signal the end of the \texttt{-forvalues-} command
- All the data-construction commands that use the variables specified in the \texttt{-forvalues-} command should be placed between the open brace and the close brace.
Syntax Rules of -foreach- and –forvalues- in Stata

- Examples: An one-dimension array contains five existing numeric variables from number1 to number5. We use an index, i, to indicate the elements of this array:

1. `foreach i in number1 number2 number3 number4 number5 {
   display "\`i\"
   list `i`
   gen `i' =1
}

2. `foreach i of varlist number1 number2 number3 number4 number5 {
   display "\`i\"
   list `i`
   gen `i' =1
}

3. `forvalues i = 1(1)5 {
   list number`i'
}

Sample SAS Codes

A. Recoding, renaming, and generating variables

```
array var{5} var1-var5;
array name{5} name1-name5;
array newvar(5) newvar1-newvar5;
do k=1 to 5;
  if var1 = 99 then var1 = .;
  if var2 = 99 then var2 = .;
  if var3 = 99 then var3 = .;
  if var4 = 99 then var4 = .;
  if var5 = 99 then var5 = .;
  rename var1 = name1;
  rename var2 = name2;
  rename var3 = name3;
  rename var4 = name4;
  rename var5 = name5;
  newvar1 = name1;
  newvar2 = name2;
  newvar3 = name3;
  newvar4 = name4;
  newvar5 = name5;
end;
run
```

```
if var{k} = 99 then var{k} = .;
rename var{k} = name{k};
newvar{k} = name{k};
end;
run;
```
Sample Stata Codes

<table>
<thead>
<tr>
<th>Without using foreach or forvalues</th>
<th>using foreach</th>
</tr>
</thead>
<tbody>
<tr>
<td>use example1.dta, clear</td>
<td>use example1.dta, clear</td>
</tr>
<tr>
<td>replace var1 =. If var1 ==99</td>
<td>foreach i in var1 var2 var3 var4 var5 {</td>
</tr>
<tr>
<td>replace var2 =. If var2 ==99</td>
<td>replace <code>i</code> =. if <code>i</code> ==99</td>
</tr>
<tr>
<td>replace var3 =. If var3 ==99</td>
<td></td>
</tr>
<tr>
<td>replace var4 =. If var4 ==99</td>
<td></td>
</tr>
<tr>
<td>replace var5 =. If var5 ==99</td>
<td></td>
</tr>
<tr>
<td>rename var1 newvar1</td>
<td>rename <code>i</code> new<code>i</code></td>
</tr>
<tr>
<td>rename var2 newvar2</td>
<td></td>
</tr>
<tr>
<td>rename var3 newvar3</td>
<td></td>
</tr>
<tr>
<td>rename var4 newvar4</td>
<td></td>
</tr>
<tr>
<td>rename var5 newvar5</td>
<td></td>
</tr>
<tr>
<td>gen n_newvar1 = newvar1</td>
<td>gen n_new<code>i</code> = new<code>i</code></td>
</tr>
<tr>
<td>gen n_newvar2 = newvar2</td>
<td></td>
</tr>
<tr>
<td>gen n_newvar3 = newvar3</td>
<td></td>
</tr>
<tr>
<td>gen n_newvar4 = newvar4</td>
<td></td>
</tr>
<tr>
<td>gen n_newvar5 = newvar5</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Stata -foreach- command for recoding and renaming variables
Sample Stata Codes

Table 3. Stata -forvalues- command for recoding and renaming variables and for generating new variables

<table>
<thead>
<tr>
<th>Without using foreach or forvalues</th>
<th>using foreach</th>
</tr>
</thead>
<tbody>
<tr>
<td>use example1.dta, clear</td>
<td>use example1.dta, clear</td>
</tr>
<tr>
<td>replace var1 =. If var1 ==99</td>
<td>forvalues i = 1(1)5 {</td>
</tr>
<tr>
<td>replace var2 =. If var2 ==99</td>
<td>replace var<code>i' =. If </code>i' ==99</td>
</tr>
<tr>
<td>replace var3 =. If var3 ==99</td>
<td>}</td>
</tr>
<tr>
<td>replace var4 =. If var4 ==99</td>
<td>rename var<code>i' newvar</code>i'</td>
</tr>
<tr>
<td>replace var5 =. If var5 ==99</td>
<td>gen n_newvar1 =newvar1</td>
</tr>
<tr>
<td></td>
<td>gen n_newvar2 = newvar2</td>
</tr>
<tr>
<td></td>
<td>gen n_newvar3 = newvar3</td>
</tr>
<tr>
<td></td>
<td>gen n_newvar4 = newvar4</td>
</tr>
<tr>
<td></td>
<td>gen n_newvar5 = newvar5</td>
</tr>
<tr>
<td>rename var1 newvar1</td>
<td>}</td>
</tr>
<tr>
<td>rename var2 newvar2</td>
<td></td>
</tr>
</tbody>
</table>
Change Data Structure

• Data in wide and long format

<table>
<thead>
<tr>
<th>name</th>
<th>marriage at Wave 1</th>
<th>marriage at Wave 2</th>
<th>marriage at Wave 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tom</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5. Data in Wide Format

<table>
<thead>
<tr>
<th>name</th>
<th>wave</th>
<th>marriage</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>John</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>John</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Mary</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Mary</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Tom</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Tom</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Tom</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6. Data in Long Format
libname in 'c:\temp\array';

data in.long2;
set in.wide;
array status[3] marriage1 marriage2 marriage3;
do i=1 to 3;
Marriage = status[i];
output;
keep name i marriage;
end;
PROC SORT DATA=in.long2 OUT=in.long3;
   BY name i;
RUN;
DATA in.wide2;
   SET in.long2;
   BY name;
   KEEP name marriage;
   RETAIN marriage1- marriage3;
   ARRAY wave(1:3) marriage1 – marriage3;
   IF first.name THEN DO;
      DO i = 1 to 3 ; wave (i ) = . ;
      END;
   END;
   Wave(year)= marriage;
   IF last.name THEN OUTPUT ;
RUN;
Change Data Format in Stata

• Change data from wide to long format
  – `reshape long marriage, i(name) j(wave)`

• Change data from long to wide format
  – `reshape wide marriage, i(name) j(wave)`
Family Structure Codes Using the do … loop Commands

• See the accompany text file
Conclusions

• SAS and Stata have different commands for performing the same data construction on multiple variables.

• When working on few variables, you may not find it necessary to use these commands. However, when working on many variables at the same time, the necessity of using these commands becomes obvious.

• SAS users can also use the –array- command to transform the data between wide and long, while Stata users can use the –reshape- command, rather than –foreach- or -forvalues- to do so.

• The do…loop is another flexible tool to perform tasks on multiple variables

• It will be difficult at first to visualize how to use these commands to complete data construction tasks. When you have errors in your codes and do not know how to fix the codes, please contact me (wuh@bgsu.edu) or stop by my office during the office hours.