stack — Stack data

Syntax

\[ \text{stack } \texttt{varlist} \ [ \text{if} ] \ [ \text{in} ], \ \{ \text{into(} \texttt{newvars} \text{)} | \text{group(} \# \text{)} \} \ [ \text{options} ] \]

options Description

Main
* into(\texttt{newvars}) identify names of new variables to be created
* \texttt{group(} \# \texttt{)} stack \# groups of variables in \texttt{varlist}
\texttt{clear} clear dataset from memory
\texttt{wide} keep variables in \texttt{varlist} that are not specified in \texttt{newvars}

* Either into(\texttt{newvars}) or group(\#) is required.

Menu

Data > Create or change data > Other variable-transformation commands > Stack data

Description

\texttt{stack} stacks the variables in \texttt{varlist} vertically, resulting in a dataset with variables \texttt{newvars} and \( N \cdot (N_v / N_n) \) observations, where \( N_v \) is the number of variables in \texttt{varlist} and \( N_n \) is the number in \texttt{newvars}. \texttt{stack} creates the new variable \_stack identifying the groups.

Options

\texttt{into(} \texttt{newvars}) identifies the names of the new variables to be created. into(\texttt{)} may be specified using variable ranges (for example, into(\texttt{v1-v3})). Either into(\texttt{)} or group(\texttt{)}, but not both, must be specified.

\texttt{group(} \# \texttt{)} specifies the number of groups of variables in \texttt{varlist} to be stacked. The created variables will be named according to the first group in \texttt{varlist}. Either group(\texttt{)} or into(\texttt{)}, but not both, must be specified.

\texttt{clear} indicates that it is okay to clear the dataset in memory. If you do not specify this option, you will be asked to confirm your intentions.

\texttt{wide} includes any of the original variables in \texttt{varlist} that are not specified in \texttt{newvars} in the resulting data.
Example 1: Illustrating the concept

This command is best understood by examples. We begin with artificial but informative examples and end with useful examples.

. use http://www.stata-press.com/data/r13/stackxmpl
. list
   a  b  c  d
  1.  1  2  3  4
  2.  5  6  7  8

. stack a b c d, into(e f) clear
. list
   _stack  e  f
   1.      1  1  2
   2.      1  5  6
   3.      2  3  4
   4.      2  7  8

We formed the new variable e by stacking a and c, and we formed the new variable f by stacking b and d. _stack is automatically created and set equal to 1 for the first (a, b) group and equal to 2 for the second (c, d) group. (When _stack==1, the new data e and f contain the values from a and b. When _stack==2, e and f contain values from c and d.)

There are two groups because we specified four variables in the varlist and two variables in the into list, and 4/2 = 2. If there were six variables in the varlist, there would be 6/2 = 3 groups. If there were also three variables in the into list, there would be 6/3 = 2 groups. Specifying six variables in the varlist and four variables in the into list would result in an error because 6/4 is not an integer.

Example 2: Stacking a variable multiple times

Variables may be repeated in the varlist, and the varlist need not contain all the variables:

. use http://www.stata-press.com/data/r13/stackxmpl, clear
. list
   a  b  c  d
  1.  1  2  3  4
  2.  5  6  7  8

. stack a b a c, into(a bc) clear
a was stacked on a and called a, whereas b was stacked on c and called bc.

If we had wanted the resulting variables to be called simply a and b, we could have used

```
. stack a b a c, group(2) clear
```

which is equivalent to

```
. stack a b a c, into(a b) clear
```

Example 3: Keeping the original variables

In this artificial but informative example, the `wide` option includes the variables in the original dataset that were specified in `varlist` in the output dataset:

```
. use http://www.stata-press.com/data/r13/stackxmpl, clear
. list
```

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

```
. stack a b c d, into(e f) clear wide
. list
```

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2 . .</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6 . .</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>. 3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>. 7</td>
<td>8</td>
</tr>
</tbody>
</table>

In addition to the stacked e and f variables, the original a, b, c, and d variables are included. They are set to missing where their values are not appropriate.

Example 4: Using wide with repeated variables

This is the last artificial example. When you specify the `wide` option and repeat the same variable name in both the `varlist` and the `into` list, the variable will contain the stacked values:
Example 5: Using stack to make graphs

We want one graph of \( y \) against \( x_1 \) and \( y \) against \( x_2 \). We might be tempted to type `scatter y x1 x2`, but that would graph \( y \) against \( x_2 \) and \( x_1 \) against \( x_2 \). One solution is to type

```
. save mydata
. stack y x1 y x2, into(yy x12) clear
. generate y1 = yy if _stack==1
. generate y2 = yy if _stack==2
. scatter y1 y2 x12
. use mydata, clear
```

The names `yy` and `x12` are supposed to suggest the contents of the variables. `yy` contains \((y,y)\), and `x12` contains \((x1,x2)\). We then make `y1` defined at the \( x_1 \) points but missing at the \( x_2 \) points—graphing \( y_1 \) against \( x_2 \) is the same as graphing \( y \) against \( x_1 \) in the original dataset. Similarly, `y2` is defined at the \( x_2 \) points but missing at \( x_1 \)—graphing \( y_2 \) against \( x_12 \) is the same as graphing \( y \) against \( x_2 \) in the original dataset. Therefore, `scatter y1 y2 x12` produces the desired graph.

Example 6: Plotting cumulative distributions

We wish to graph \( y_1 \) against \( x_1 \) and \( y_2 \) against \( x_2 \) on the same graph. The logic is the same as above, but let’s go through it. Perhaps we have constructed two cumulative distributions by using `cumul` (see [R] `cumul`):

```
. use http://www.stata-press.com/data/r13/citytemp
   (City Temperature Data)
. cumul tempjan, gen(cjan)
. cumul tempjuly, gen(cjuly)
```

We want to graph both cumulatives in the same graph; that is, we want to graph `cjan` against `tempjan` and `cjuly` against `tempjuly`. Remember that we could graph the `tempjan` cumulative by typing

```
. scatter cjan tempjan, c(l) m(o) sort
   (output omitted)
```
We can graph the \texttt{tempjuly} cumulative similarly. To obtain both on the same graph, we must stack the data:

\begin{verbatim}
. stack cjuly tempjuly cjan tempjan, into(c temp) clear
. generate cjan = c if _stack==1
  (958 missing values generated)
. generate cjuly = c if _stack==2
  (958 missing values generated)
. scatter cjan cjuly temp, c(l l) m(o o) sort
(output omitted)
\end{verbatim}

Alternatively, if we specify the \texttt{wide} option, we do not have to regenerate \texttt{cjan} and \texttt{cjuly} because they will be created automatically:

\begin{verbatim}
. use http://www.stata-press.com/data/r13/citytemp, clear
  (City Temperature Data)
. cumul tempjan, gen(cjan)
. cumul tempjuly, gen(cjuly)
. stack cjuly tempjuly cjan tempjan, into(c temp) clear wide
. scatter cjan cjuly temp, c(l l) m(o o) sort
(output omitted)
\end{verbatim}

\textbf{Technical note}

There is a third way, not using the \texttt{wide} option, that is exceedingly tricky but is sometimes useful:

\begin{verbatim}
. use http://www.stata-press.com/data/r13/citytemp, clear
  (City Temperature Data)
. cumul tempjan, gen(cjan)
. cumul tempjuly, gen(cjuly)
. stack cjuly tempjuly cjan tempjan, into(c temp) clear
. sort _stack temp
. scatter c temp, c(L) m(o)
(output omitted)
\end{verbatim}

Note the use of \texttt{connect}’s capital \texttt{L} rather than lowercase \texttt{l} option. \texttt{c(L)} connects points only from left to right; because the data are sorted by \_\texttt{stack temp}, \texttt{temp} increases within the first group (\texttt{cjuly} vs. \texttt{tempjuly}) and then starts again for the second (\texttt{cjan} vs. \texttt{tempjan}); see \cite{connectstyle}. 

\section*{Reference}

Baum, C. F. 2009. \textit{An Introduction to Stata Programming}. College Station, TX: Stata Press.

\section*{Also see}

[D] \texttt{contract} — Make dataset of frequencies and percentages  
[D] \texttt{reshape} — Convert data from wide to long form and vice versa  
[D] \texttt{xpose} — Interchange observations and variables