**Syntax**

**Overview**

<table>
<thead>
<tr>
<th>long</th>
<th>wide</th>
</tr>
</thead>
<tbody>
<tr>
<td>[i \quad j \quad stub]</td>
<td>[i \quad stub1 \quad stub2]</td>
</tr>
<tr>
<td>1 \quad 1 \quad 4.1</td>
<td>1 \quad 4.1 \quad 4.5</td>
</tr>
<tr>
<td>1 \quad 2 \quad 4.5</td>
<td>2 \quad 3.3 \quad 3.0</td>
</tr>
<tr>
<td>2 \quad 1 \quad 3.3</td>
<td></td>
</tr>
<tr>
<td>2 \quad 2 \quad 3.0</td>
<td></td>
</tr>
</tbody>
</table>

To go from long to wide:

\[
\text{reshape wide \hspace{1em} stub, \hspace{1em} i(i) \hspace{1em} j(j)}
\]

To go from wide to long:

\[
\text{reshape long \hspace{1em} stub, \hspace{1em} i(i) \hspace{1em} j(j)}
\]

To go back to long after using `reshape wide`:

\[
\text{reshape long}
\]

To go back to wide after using `reshape long`:

\[
\text{reshape wide}
\]

**Basic syntax**

*Convert data from wide form to long form*

\[
\text{reshape long \hspace{1em} stubnames, \hspace{1em} i(varlist) \hspace{1em} [options]}
\]

*Convert data from long form to wide form*

\[
\text{reshape wide \hspace{1em} stubnames, \hspace{1em} i(varlist) \hspace{1em} [options]}
\]

*Convert data back to long form after using reshape wide*

\[
\text{reshape long}
\]
Convert data back to wide form after using reshape long
reshape wide

List problem observations when reshape fails
reshape error

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*i(varlist)</td>
<td>use varlist as the ID variables</td>
</tr>
<tr>
<td>j(varname [values])</td>
<td>long→wide: varname, existing variable wide→long: varname, new variable optionally specify values to subset varname</td>
</tr>
<tr>
<td>string</td>
<td>varname is a string variable (default is numeric)</td>
</tr>
</tbody>
</table>

* i(varlist) is required.

where values is #[#] [ # ... ] if varname is numeric (default)
"string" [ "string" ... ] if varname is string

and where stubnames are variable names (long→wide), or stubs of variable names (wide→long), and either way, may contain @, denoting where j appears or is to appear in the name.

Advanced syntax

reshape i varlist
reshape j varname [values] [, string]
reshape xij fvarnames [, atwl(chars)]
reshape xi [varlist]
reshape [query]
reshape clear

Menu
Data > Create or change data > Other variable-transformation commands > Convert data between wide and long

Description
reshape converts data from wide to long form and vice versa.

Options
i(varlist) specifies the variables whose unique values denote a logical observation. i() is required.
j(varname [values]) specifies the variable whose unique values denote a subobservation. values lists the unique values to be used from varname, which typically are not explicitly stated because reshape will determine them automatically from the data.
string specifies that \( j() \) may contain string values.

\texttt{atwl(chars)}, available only with the advanced syntax and not shown in the dialog box, specifies that \texttt{chars} be substituted for the @ character when converting the data from wide to long form.

### Remarks and examples

Remarks are presented under the following headings:

- Description of basic syntax
- Wide and long data forms
- Avoiding and correcting mistakes
- \texttt{reshape long} and \texttt{reshape wide} without arguments
- Missing variables
- Advanced issues with basic syntax: \texttt{i()}
- Advanced issues with basic syntax: \texttt{j()}
- Advanced issues with basic syntax: \texttt{xij}
- Advanced issues with basic syntax: String identifiers for \texttt{j()}
- Advanced issues with basic syntax: Second-level nesting
- Description of advanced syntax

See Mitchell (2010, chap. 8) for information and examples using \texttt{reshape}.

### Description of basic syntax

Before using \texttt{reshape}, you need to determine whether the data are in long or wide form. You also must determine the logical observation (i) and the subobservation (j) by which to organize the data. Suppose that you had the following data, which could be organized in wide or long form as follows:

\[
\begin{array}{cccc}
\text{i} & \ldots & X_{ij} & \ldots \\
id & \text{sex} & \text{inc80} & \text{inc81} & \text{inc82} \\
1 & 0 & 5000 & 5500 & 6000 \\
2 & 1 & 2000 & 2200 & 3300 \\
3 & 0 & 3000 & 2000 & 1000 \\
\end{array}
\quad
\begin{array}{cccc}
\text{i} & \text{j} & X_{ij} \\
id & \text{year} & \text{sex} & \text{inc} \\
1 & 80 & 0 & 5000 \\
1 & 81 & 0 & 5500 \\
1 & 82 & 0 & 6000 \\
2 & 80 & 1 & 2000 \\
2 & 81 & 1 & 2200 \\
2 & 82 & 1 & 3300 \\
3 & 80 & 0 & 3000 \\
3 & 81 & 0 & 2000 \\
3 & 82 & 0 & 1000 \\
\end{array}
\]

Given these data, you could use \texttt{reshape} to convert from one form to the other:

\[
\begin{align*}
\text{. reshape long inc, i(id) j(year)} & \quad /* \text{goes from left form to right} */ \\
\text{. reshape wide inc, i(id) j(year)} & \quad /* \text{goes from right form to left} */
\end{align*}
\]

Because we did not specify \texttt{sex} in the command, Stata assumes that it is constant within the logical observation, here \texttt{id}.

### Wide and long data forms

Think of the data as a collection of observations \( X_{ij} \), where \( i \) is the logical observation, or group identifier, and \( j \) is the subobservation, or within-group identifier.
Wide-form data are organized by logical observation, storing all the data on a particular observation in one row. Long-form data are organized by subobservation, storing the data in multiple rows.

Example 1

For example, we might have data on a person’s ID, gender, and annual income over the years 1980–1982. We have two $X_{ij}$ variables with the data in wide form:

```
.use http://www.stata-press.com/data/r13/reshape1
.list
```

<table>
<thead>
<tr>
<th>id</th>
<th>sex</th>
<th>inc80</th>
<th>inc81</th>
<th>inc82</th>
<th>ue80</th>
<th>ue81</th>
<th>ue82</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5000</td>
<td>5500</td>
<td>6000</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2000</td>
<td>2200</td>
<td>3300</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3000</td>
<td>2000</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

To convert these data to the long form, we type

```
. reshape long inc ue, i(id) j(year)
```

<table>
<thead>
<tr>
<th>Data</th>
<th>wide</th>
<th>-&gt;</th>
<th>long</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of obs.</td>
<td>3</td>
<td>-&gt;</td>
<td>9</td>
</tr>
<tr>
<td>Number of variables</td>
<td>8</td>
<td>-&gt;</td>
<td>5</td>
</tr>
<tr>
<td>j variable (3 values)</td>
<td>-&gt;</td>
<td>year</td>
<td></td>
</tr>
<tr>
<td>xij variables:</td>
<td>inc80 inc81 inc82 -&gt; inc</td>
<td>ue80 ue81 ue82 -&gt; ue</td>
<td></td>
</tr>
</tbody>
</table>

There is no variable named `year` in our original, wide-form dataset. `year` will be a new variable in our long dataset. After this conversion, we have

```
.list, sep(3)
```

<table>
<thead>
<tr>
<th>id</th>
<th>year</th>
<th>sex</th>
<th>inc</th>
<th>ue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>0</td>
<td>5000</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>81</td>
<td>0</td>
<td>5500</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>82</td>
<td>0</td>
<td>6000</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>1</td>
<td>2000</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>81</td>
<td>1</td>
<td>2200</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>82</td>
<td>1</td>
<td>3300</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>0</td>
<td>3000</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>81</td>
<td>0</td>
<td>2000</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>82</td>
<td>0</td>
<td>1000</td>
<td>1</td>
</tr>
</tbody>
</table>
We can return to our original, wide-form dataset by using `reshape wide`.

```
. reshape wide inc ue, i(id) j(year)
(note: j = 80 81 82)
```

<table>
<thead>
<tr>
<th>Data</th>
<th>long</th>
<th>wide</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of obs.</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Number of variables</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>j variable (3 values)</td>
<td>year</td>
<td>(dropped)</td>
</tr>
<tr>
<td>xij variables:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inc -&gt; inc80 inc81 inc82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ue -&gt; ue80 ue81 ue82</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
. list
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>ic80</th>
<th>ue80</th>
<th>ic81</th>
<th>ue81</th>
<th>ic82</th>
<th>ue82</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5000</td>
<td>0</td>
<td>5500</td>
<td>1</td>
<td>6000</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2000</td>
<td>1</td>
<td>2200</td>
<td>0</td>
<td>3300</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3000</td>
<td>0</td>
<td>2000</td>
<td>0</td>
<td>1000</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Converting from wide to long creates the `j` (`year`) variable. Converting back from long to wide drops the `j` (`year`) variable.

⚠️ Technical note

If your data are in wide form and you do not have a group identifier variable (the `i(varlist)` required option), you can create one easily by using `generate`; see [D] `generate`. For instance, in the last example, if we did not have the `id` variable in our dataset, we could have created it by typing

```
. generate id = _n
```

Avoiding and correcting mistakes

`reshape` often detects when the data are not suitable for reshaping; an error is issued, and the data remain unchanged.

➤ Example 2

The following wide data contain a mistake:

```
. use http://www.stata-press.com/data/r13/reshape2, clear
. list
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>sex</th>
<th>inc80</th>
<th>inc81</th>
<th>inc82</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5000</td>
<td>5500</td>
<td>6000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2000</td>
<td>2200</td>
<td>3300</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3000</td>
<td>2000</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2400</td>
<td>2500</td>
<td>2400</td>
</tr>
</tbody>
</table>
```
. reshape long inc, i(id) j(year)
(note:  j = 80 81 82)
variable id does not uniquely identify the observations.
Your data are currently wide. You are performing a reshape long. You
specified i(id) and j(year). In the current wide form, variable id should
uniquely identify the observations. Remember this picture:

<table>
<thead>
<tr>
<th>long</th>
<th>wide</th>
</tr>
</thead>
<tbody>
<tr>
<td>i j a b</td>
<td>i a1 a2 b1 b2</td>
</tr>
<tr>
<td>1 1 1 2</td>
<td>1 1 3 2 4</td>
</tr>
<tr>
<td>1 2 3 4</td>
<td>2 5 7 6 8</td>
</tr>
<tr>
<td>2 1 5 6</td>
<td></td>
</tr>
<tr>
<td>2 2 7 8</td>
<td></td>
</tr>
</tbody>
</table>

Type reshape error for a list of the problem observations.
r(9);

The i variable must be unique when the data are in the wide form; we typed i(id), yet we have 2
observations for which id is 2. (Is person 2 a male or female?)

Example 3

It is not a mistake when the i variable is repeated when the data are in long form, but the following
data have a similar mistake:

. use http://www.stata-press.com/data/r13/reshapexp1
. list

<table>
<thead>
<tr>
<th>id</th>
<th>year</th>
<th>sex</th>
<th>inc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1</td>
<td>80</td>
<td>5000</td>
</tr>
<tr>
<td>2.</td>
<td>1</td>
<td>81</td>
<td>5500</td>
</tr>
<tr>
<td>3.</td>
<td>1</td>
<td>81</td>
<td>5400</td>
</tr>
<tr>
<td>4.</td>
<td>1</td>
<td>82</td>
<td>6000</td>
</tr>
</tbody>
</table>

. reshape wide inc, i(id) j(year)
(note:  j = 80 81 82)
values of variable year not unique within id
Your data are currently long. You are performing a reshape wide. You
specified i(id) and j(year). There are observations within i(id) with the
same value of j(year). In the long data, variables i() and j() together
must uniquely identify the observations.

<table>
<thead>
<tr>
<th>long</th>
<th>wide</th>
</tr>
</thead>
<tbody>
<tr>
<td>i j a b</td>
<td>i a1 a2 b1 b2</td>
</tr>
<tr>
<td>1 1 1 2</td>
<td>1 1 3 2 4</td>
</tr>
<tr>
<td>1 2 3 4</td>
<td>2 5 7 6 8</td>
</tr>
<tr>
<td>2 1 5 6</td>
<td></td>
</tr>
<tr>
<td>2 2 7 8</td>
<td></td>
</tr>
</tbody>
</table>

Type reshape error for a list of the problem variables.
r(9);

In the long form, i(id) does not have to be unique, but j(year) must be unique within i; otherwise,
what is the value of inc in 1981 for which id==1?
**Example 4**

Consider some long-form data that have no mistakes. We list the first 4 observations.

```plaintext
. use http://www.stata-press.com/data/r13/reshape6
. list in 1/4
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>year</th>
<th>sex</th>
<th>inc</th>
<th>ue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>0</td>
<td>5000</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>81</td>
<td>0</td>
<td>5500</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>82</td>
<td>0</td>
<td>6000</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>1</td>
<td>2000</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Say that when converting the data to wide form, however, we forget to mention the `ue` variable (which varies within person).

```plaintext
. reshape wide inc, i(id) j(year)
```

```
variable ue not constant within id
Your data are currently long. You are performing a reshape wide. You typed something like
```

```plaintext
. reshape wide a b, i(id) j(year)
```

There are variables other than `a`, `b`, `id`, `year` in your data. They must be constant within `id` because that is the only way they can fit into wide data without loss of information.

The variable or variables listed above are not constant within `id`. Perhaps the values are in error. Type reshape error for a list of the problem observations.

Either that, or the values vary because they should vary, in which case you must either add the variables to the list of xij variables to be reshaped, or drop them.

```
r(9);```

Here `reshape` observed that `ue` was not constant within `id` and so could not restructure the data so that there were single observations on `id`. We should have typed

```plaintext
. reshape wide inc ue, i(id) j(year)
```
In summary, there are three cases in which `reshape` will refuse to convert the data:
1. The data are in wide form and \( i \) is not unique.
2. The data are in long form and \( j \) is not unique within \( i \).
3. The data are in long form and an unmentioned variable is not constant within \( i \).

**Example 5**

With some mistakes, `reshape` will probably convert the data and produce a surprising result. Suppose that we forget to mention that the `ue` variable varies within `id` in the following wide data:

```stata
use http://www.stata-press.com/data/r13/reshape1
list
id  sex  inc80  inc81  inc82  ue80  ue81  ue82
1. 1  0   5000   5500   6000   0   1   0
2. 2  1   2000   2200   3300   1   0   0
3. 3  0   3000   2000   1000   0   0   1
reshape long inc, i(id) j(year)
```

```
Data wide -> long
Number of obs. 3 -> 9
Number of variables 8 -> 7
j variable (3 values) -> year
xij variables:
inc80 inc81 inc82 -> inc
```

```stata
list, sep(3)
```

```
id  year  sex  inc  ue80  ue81  ue82
1.  80  0 5000   0   1   0
2.  81  0 5500   0   1   0
3.  82  0 6000   0   1   0
4.  80  1 2000   1   0   0
5.  81  1 2200   1   0   0
6.  82  1 3300   1   0   0
7.  80  0 3000   0   0   1
8.  81  0 2000   0   0   1
9.  82  0 1000   0   0   1
```

We did not state that `ue` varied within `i`, so the variables `ue80`, `ue81`, and `ue82` were left as is. `reshape` did not complain. There is no real problem here because no information has been lost. In fact, this may actually be the result we wanted. Probably, however, we simply forgot to include `ue` among the \( X_{ij} \) variables.

If you obtain an unexpected result, here is how to undo it:
1. If you typed `reshape long` ... to produce the result, type `reshape wide` (without arguments) to undo it.
2. If you typed `reshape wide` ... to produce the result, type `reshape long` (without arguments) to undo it.
So, we can type

```. reshape wide```

to get back to our original, wide-form data and then type the `reshape long` command that we intended:

```. reshape long inc ue, i(id) j(year)```

**reshape long and reshape wide without arguments**

Whenever you type a `reshape long` or `reshape wide` command with arguments, `reshape` remembers it. Thus you might type

```. reshape long inc ue, i(id) j(year)```

and work with the data like that. You could then type

```. reshape wide```

to convert the data back to the wide form. Then later you could type

```. reshape long```

to convert them back to the long form. If you save the data, you can even continue using `reshape wide` and `reshape long` without arguments during a future Stata session.

Be careful. If you create new $X_{ij}$ variables, you must tell `reshape` about them by typing the full `reshape` command, although no real damage will be done if you forget. If you are converting from long to wide form, `reshape` will catch your error and refuse to make the conversion. If you are converting from wide to long, `reshape` will convert the data, but the result will be surprising: remember what happened when we forgot to mention the `ue` variable and ended up with `ue80`, `ue81`, and `ue82` in our long data; see example 5. You can `reshape long` to undo the unwanted change and then try again.

**Missing variables**

When converting data from wide form to long form, `reshape` does not demand that all the variables exist. Missing variables are treated as variables with missing observations.

➤ **Example 6**

Let’s drop `ue81` from the wide form of the data:

```
. use http://www.stata-press.com/data/r13/reshape1, clear
. drop ue81
. list
```

<table>
<thead>
<tr>
<th></th>
<th>id</th>
<th>sex</th>
<th>inc80</th>
<th>inc81</th>
<th>inc82</th>
<th>ue80</th>
<th>ue82</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1</td>
<td>0</td>
<td>5000</td>
<td>5500</td>
<td>6000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.</td>
<td>2</td>
<td>1</td>
<td>2000</td>
<td>2200</td>
<td>3300</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3.</td>
<td>3</td>
<td>0</td>
<td>3000</td>
<td>2000</td>
<td>1000</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
. reshape long inc ue, i(id) j(year)
(note: j = 80 81 82)
(note: ue81 not found)

<table>
<thead>
<tr>
<th></th>
<th>wide -&gt; long</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of obs.</td>
<td>3 -&gt; 9</td>
</tr>
<tr>
<td>Number of variables</td>
<td>7 -&gt; 5</td>
</tr>
<tr>
<td>j variable (3 values)</td>
<td>-&gt; year</td>
</tr>
<tr>
<td>xij variables:</td>
<td></td>
</tr>
<tr>
<td>inc80 inc81 inc82 -&gt; inc</td>
<td></td>
</tr>
<tr>
<td>ue80 ue81 ue82 -&gt; ue</td>
<td></td>
</tr>
</tbody>
</table>

Advanced issues with basic syntax: i()

The i() option can indicate one i variable (as our past examples have illustrated) or multiple variables. An example of multiple i variables would be hospital ID and patient ID within each hospital.

```
. reshape ... , i(hid pid)
```

Unique pairs of values for hid and pid in the data define the grouping variable for reshape.

Advanced issues with basic syntax: j()

The j() option takes a variable name (as our past examples have illustrated) or a variable name and a list of values. When the values are not provided, reshape deduces them from the data. Specifying the values with the j() option is rarely needed.

```
reshape never makes a mistake when the data are in long form and you type reshape wide. The values are easily obtained by tabulating the j variable.
```
reshape can make a mistake when the data are in wide form and you type `reshape long` if your variables are poorly named. Say that you have the `inc80`, `inc81`, and `inc82` variables, recording income in each of the indicated years, and you have a variable named `inc2`, which is not income but indicates when the area was reincorporated. You type

```
   . reshape long inc, i(id) j(year)
```

`reshape` sees the `inc2`, `inc80`, `inc81`, and `inc82` variables and decides that there are four groups in which \( j = 2, 80, 81, \) and \( 82. \)

The easiest way to solve the problem is to rename the `inc2` variable to something other than “inc” followed by a number; see [D] `rename`.

You can also keep the name and specify the \( j \) values. To perform the reshape, you can type

```
   . reshape long inc, i(id) j(year 80-82)
```

or

```
   . reshape long inc, i(id) j(year 80 81 82)
```

You can mix the dash notation for value ranges with individual numbers. `reshape` would understand \( 80 \ 82-87 \ 89 \ 91-95 \) as a valid values specification.

At the other extreme, you can omit the `j()` option altogether with `reshape long`. If you do, the \( j \) variable will be named `_j`.

### Advanced issues with basic syntax: \( xij \)

When specifying variable names, you may include @ characters to indicate where the numbers go.

#### Example 7

Let’s reshape the following data from wide to long form:

```
   . use http://www.stata-press.com/data/r13/reshape3, clear
   . list
   +-----+--------+--------+--------+--------+--------+--------+--------+--------+        
   | id  | sex   | inc80r | inc81r | inc82r | ue80   | ue81   | ue82   |
   +-----+--------+--------+--------+--------+--------+--------+--------+        
   | 1.  | 0  | 5000   | 5500   | 6000   | 0      | 1      | 1      |
   | 2.  | 1  | 2000   | 2200   | 3300   | 1      | 0      | 0      |
   | 3.  | 0  | 3000   | 2000   | 1000   | 0      | 0      | 1      |
   +-----+--------+--------+--------+--------+--------+--------+--------+        
```

```
   . reshape long inc@r ue, i(id) j(year)
   (note: j = 80 81 82)
```

Data wide -> long

<table>
<thead>
<tr>
<th>Data</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of obs.</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>Number of variables</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>( j ) variable (3 values)</td>
<td></td>
<td>year</td>
</tr>
<tr>
<td>( xij ) variables:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inc80r inc81r inc82r</td>
<td>-&gt;</td>
<td>incr</td>
</tr>
<tr>
<td>ue80 ue81 ue82</td>
<td>-&gt;</td>
<td>ue</td>
</tr>
</tbody>
</table>
. list, sep(3)

<table>
<thead>
<tr>
<th>id</th>
<th>year</th>
<th>sex</th>
<th>incr</th>
<th>ue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>0</td>
<td>5000</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>81</td>
<td>0</td>
<td>5500</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>82</td>
<td>0</td>
<td>6000</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>1</td>
<td>2000</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>81</td>
<td>1</td>
<td>2200</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>82</td>
<td>1</td>
<td>3300</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>0</td>
<td>3000</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>81</td>
<td>0</td>
<td>2000</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>82</td>
<td>0</td>
<td>1000</td>
<td>1</td>
</tr>
</tbody>
</table>

At most one @ character may appear in each name. If no @ character appears, results are as if the @ character appeared at the end of the name. So, the equivalent reshape command to the one above is

. reshape long inc@r ue@, i(id) j(year)

inc@r specifies variables named inc#r in the wide form and incr in the long form. The @ notation may similarly be used for converting data from long to wide format:

. reshape wide inc@r ue, i(id) j(year)

Advanced issues with basic syntax: String identifiers for j()

The string option allows j to take on string values.

> Example 8

Consider the following wide data on husbands and wives. In these data, incm is the income of the man and incf is the income of the woman.

. use http://www.stata-press.com/data/r13/reshape4, clear
. list

<table>
<thead>
<tr>
<th>id</th>
<th>kids</th>
<th>incm</th>
<th>incf</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5000</td>
<td>5500</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2000</td>
<td>2200</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3000</td>
<td>2000</td>
</tr>
</tbody>
</table>

These data can be reshaped into separate observations for males and females by typing

. reshape long inc, i(id) j(sex) string
(note: j = f m)

<table>
<thead>
<tr>
<th>Data</th>
<th>wide -&gt; long</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of obs.</td>
<td>3 -&gt; 6</td>
</tr>
<tr>
<td>Number of variables</td>
<td>4 -&gt; 4</td>
</tr>
<tr>
<td>j variable (2 values)</td>
<td>-&gt; sex</td>
</tr>
<tr>
<td>xij variables:</td>
<td>incf incm -&gt; inc</td>
</tr>
</tbody>
</table>
The string option specifies that \( j \) take on nonnumeric values. The result is

```
. list, sep(2)
```

<table>
<thead>
<tr>
<th>id</th>
<th>sex</th>
<th>kids</th>
<th>inc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>f</td>
<td>0</td>
<td>5500</td>
</tr>
<tr>
<td>1</td>
<td>m</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>2</td>
<td>f</td>
<td>1</td>
<td>2200</td>
</tr>
<tr>
<td>2</td>
<td>m</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>2</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>m</td>
<td>2</td>
<td>3000</td>
</tr>
</tbody>
</table>

\( \text{sex} \) will be a string variable. Similarly, these data can be converted from long to wide form by typing

```
. reshape wide inc, i(id) j(sex) string
```

Strings are not limited to being single characters or even having the same length. You can specify the location of the string identifier in the variable name by using the \( @ \) notation.

**Example 9**

Suppose that our variables are named \( \text{id}, \text{kids}, \text{incmale}, \) and \( \text{incfem} \).

```
. use http://www.stata-press.com/data/r13/reshapexp2, clear
. list
```

<table>
<thead>
<tr>
<th>id</th>
<th>kids</th>
<th>incmale</th>
<th>incfem</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5000</td>
<td>5500</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2000</td>
<td>2200</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3000</td>
<td>2000</td>
</tr>
</tbody>
</table>

```
. reshape long inc, i(id) j(sex) string
(note: j = fem male)
```

Data

<table>
<thead>
<tr>
<th></th>
<th>wide</th>
<th>long</th>
</tr>
</thead>
<tbody>
<tr>
<td>obs.</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>vars</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>j var</td>
<td>2</td>
<td>sex</td>
</tr>
<tr>
<td>xij</td>
<td>incfem incmale</td>
<td>inc</td>
</tr>
</tbody>
</table>

```
. list, sep(2)
```

<table>
<thead>
<tr>
<th>id</th>
<th>sex</th>
<th>kids</th>
<th>inc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fem</td>
<td>0</td>
<td>5500</td>
</tr>
<tr>
<td>1</td>
<td>male</td>
<td>0</td>
<td>5000</td>
</tr>
<tr>
<td>2</td>
<td>fem</td>
<td>1</td>
<td>2200</td>
</tr>
<tr>
<td>2</td>
<td>male</td>
<td>1</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>fem</td>
<td>2</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>male</td>
<td>2</td>
<td>3000</td>
</tr>
</tbody>
</table>
If the wide data had variables named `minc` and `finc`, the appropriate `reshape` command would have been

```
. reshape long @inc, i(id) j(sex) string
```

The resulting variable in the long form would be named `inc`.

We can also place strings in the middle of the variable names. If the variables were named `incMome` and `incFome`, the `reshape` command would be

```
. reshape long inc@ome, i(id) j(sex) string
```

Be careful with string identifiers because it is easy to be surprised by the result. Say that we have wide data having variables named `incm`, `incf`, `uem`, `uef`, `agem`, and `agef`. To make the data long, we might type

```
. reshape long inc ue age, i(id) j(sex) string
```

Along with these variables, we also have the variable `agenda`. `reshape` will decide that the sexes are `m`, `f`, and `nda`. This would not happen without the `string` option if the variables were named `inc0`, `inc1`, `ue0`, `ue1`, `age0`, and `age1`, even with the `agenda` variable present in the data.

Advanced issues with basic syntax: Second-level nesting

Sometimes the data may have more than one possible `j` variable for reshaping. Suppose that your data have both a year variable and a sex variable. One logical observation in the data might be represented in any of the following four forms:

- **list in 1/4** // The long-long form
  
<table>
<thead>
<tr>
<th>hid</th>
<th>sex</th>
<th>year</th>
<th>inc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>f</td>
<td>90</td>
<td>3200</td>
</tr>
<tr>
<td>2</td>
<td>f</td>
<td>91</td>
<td>4700</td>
</tr>
<tr>
<td>3</td>
<td>m</td>
<td>90</td>
<td>4500</td>
</tr>
<tr>
<td>4</td>
<td>m</td>
<td>91</td>
<td>4600</td>
</tr>
</tbody>
</table>

- **list in 1/2** // The long-year wide-sex form
  
<table>
<thead>
<tr>
<th>hid</th>
<th>year</th>
<th>minc</th>
<th>finc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90</td>
<td>4500</td>
<td>3200</td>
</tr>
<tr>
<td>2</td>
<td>91</td>
<td>4600</td>
<td>4700</td>
</tr>
</tbody>
</table>

- **list in 1/2** // The wide-year long-sex form
  
<table>
<thead>
<tr>
<th>hid</th>
<th>sex</th>
<th>inc90</th>
<th>inc91</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>f</td>
<td>3200</td>
<td>4700</td>
</tr>
<tr>
<td>2</td>
<td>m</td>
<td>4500</td>
<td>4600</td>
</tr>
</tbody>
</table>

- **list in 1** // The wide-wide form
  
<table>
<thead>
<tr>
<th>hid</th>
<th>minc90</th>
<th>minc91</th>
<th>finc90</th>
<th>finc91</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4500</td>
<td>4600</td>
<td>3200</td>
<td>4700</td>
</tr>
</tbody>
</table>
reshape can convert any of these forms to any other. Converting data from the long–long form to the wide–wide form (or any of the other forms) takes two reshape commands. Here is how we would do it:

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>sex</td>
<td>year</td>
</tr>
<tr>
<td>long</td>
<td>long</td>
<td>wide</td>
</tr>
<tr>
<td>long</td>
<td>long</td>
<td>wide</td>
</tr>
<tr>
<td>wide</td>
<td>long</td>
<td>long</td>
</tr>
<tr>
<td>long</td>
<td>wide</td>
<td>wide</td>
</tr>
<tr>
<td>wide</td>
<td>wide</td>
<td>long</td>
</tr>
<tr>
<td>wide</td>
<td>long</td>
<td>wide</td>
</tr>
<tr>
<td>wide</td>
<td>wide</td>
<td>long</td>
</tr>
<tr>
<td>wide</td>
<td>wide</td>
<td>wide</td>
</tr>
</tbody>
</table>

### Description of advanced syntax

The advanced syntax is simply a different way of specifying the reshape command, and it has one seldom-used feature that provides extra control. Rather than typing one reshape command to describe the data and perform the conversion, such as

```
. reshape long inc, i(id) j(year)
```
you type a sequence of reshape commands. The initial commands describe the data, and the last command performs the conversion:

```
. reshape i id
. reshape j year
. reshape xij inc
. reshape long
```

**reshape i** corresponds to **i()** in the basic syntax.

**reshape j** corresponds to **j()** in the basic syntax.

**reshape xij** corresponds to the variables specified in the basic syntax. **reshape xij** also accepts the **atwl()** option for use when @ characters are specified in the **fvarnames**. **atwl** stands for at-when-long. When you specify names such as **inc@r** or **ue@**, in the long form the names become **incr** and **ue**, and the @ character is ignored. **atwl()** allows you to change @ into whatever you specify. For example, if you specify **atwl(X)**, the long-form names become **incXr** and **ueX**.

There is also one more specification, which has no counterpart in the basic syntax:

```
. reshape xi varlist
```

In the basic syntax, Stata assumes that all unspecified variables are constant within i. The advanced syntax works the same way, unless you specify the **reshape xi** command, which names the constant-within-i variables. If you specify **reshape xi**, any variables that you do not explicitly specify are dropped from the data during the conversion.

As a practical matter, you should explicitly drop the unwanted variables before conversion. For instance, suppose that the data have variables **inc80**, **inc81**, **inc82**, **sex**, **age**, and **age2** and that you no longer want the **age2** variable. You could specify

```
. reshape xi sex age
```

or

```
. drop age2
```

and leave **reshape xi** unspecified.
reshape xi does have one minor advantage. It saves reshape the work of determining which variables are unspecified. This saves a relatively small amount of computer time.

Another advanced-syntax feature is reshape query, which is equivalent to typing reshape by itself. reshape query reports which reshape parameters have been defined. reshape i, reshape j, reshape xij, and reshape xi specifications may be given in any order and may be repeated to change or correct what has been specified.

Finally, reshape clear clears the definitions. reshape definitions are stored with the dataset when you save it. reshape clear allows you to erase these definitions.

The basic syntax of reshape is implemented in terms of the advanced syntax, so you can mix basic and advanced syntaxes.

Stored results

reshape stores the following characteristics with the data (see [P] char):

- \_dta[ReS\_i] \_i variable names
- \_dta[ReS\_j] \_j variable name
- \_dta[ReS\_jv] \_j values, if specified
- \_dta[ReS\_Xij] \_X_{ij} variable names
- \_dta[ReS\_Xi] \_X_i variable names, if specified
- \_dta[ReS\_atwl] \_atwl() value, if specified
- \_dta[ReS\_str] \_1 if option string specified; 0 otherwise

Acknowledgment

This version of reshape was based in part on the work of Jeroen Weesie (1997) of the Department of Sociology at Utrecht University, The Netherlands.

References


Also see

[D] save — Save Stata dataset
[D] stack — Stack data
[D] xpose — Interchange observations and variables
[P] char — Characteristics