Description

`expand` replaces each observation in the dataset with \( n \) copies of the observation, where \( n \) is equal to the required expression rounded to the nearest integer. If the expression is less than 1 or equal to `missing`, it is interpreted as if it were 1, and the observation is retained but not duplicated.

Quick start

Duplicate each observation 3 times, resulting in the original and 2 copies

```stata
expand 3
```

Duplicate each observation the number of times stored in \( v \)

```stata
expand v
```

As above, but flag duplicated observations using generated `newv`

```stata
expand v, generate(newv)
```

As above, but only duplicate observations where `catvar` equals 4

```stata
expand v if catvar==4, generate(newv)
```

Menu

Data > Create or change data > Other variable-transformation commands > Duplicate observations
Syntax

```
expand [ = ] exp [ if ] [ in ] [ , generate( newvar ) ]
```

Option

generate( newvar ) creates new variable newvar containing 0 if the observation originally appeared in the dataset and 1 if the observation is a duplicate. For instance, after an expand, you could revert to the original observations by typing `keep if newvar==0`.

Remarks and examples

Example 1

`expand` is, admittedly, a strange command. It can, however, be useful in tricky programs or for reformatting data for survival analysis (see examples in `[R] Epitab`). Here is a silly use of `expand`:

```
. use https://www.stata-press.com/data/r16/expandxmpl
. list
```

```
+---+---+
<table>
<thead>
<tr>
<th>n</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
+---+---+
```

. expand n

```
(1 negative count ignored; observation not deleted)
(1 zero count ignored; observation not deleted)
(3 observations created)
```

. list

```
+---+---+
<table>
<thead>
<tr>
<th>n</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>
+---+---+
```

The new observations are added to the end of the dataset. `expand` informed us that it created 3 observations. The first 3 observations were not replicated because `n` was less than or equal to 1. `n` is 2 in the fourth observation, so `expand` created one replication of this observation, bringing the total number of observations of this type to 2. `expand` created two replications of observation 5 because `n` is 3.

Because there were 5 observations in the original dataset and because `expand` adds new observations onto the end of the dataset, we could now undo the expansion by typing `drop in 6/1`.


References


Also see

[D] *contract* — Make dataset of frequencies and percentages
[D] *expandcl* — Duplicate clustered observations
[D] *fillin* — Rectangularize dataset