The fillin command (see [R] fillin) does precisely one thing: it fills in the gaps in a rectangular data structure. That is very well explained in the manual entry, but people who do not yet know the command often miss it, so here is one more plug. Suppose that you have a dataset of people and the choices they make, something like this:

<table>
<thead>
<tr>
<th>id</th>
<th>choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Now suppose that you wish to run a nested logit model using nlogit (see [R] nlogit). This command requires all choices, those made and those not made, to be explicit. With even 4 values of id and 3 values of choice, we need 12 observations so that each combination of variables exists once in the dataset; hence, 8 more are needed in this case. The solution is just

```
.fillin id choice
```

and a new variable, _fillin, is added to the dataset with values 1 if the observation was “filled in” and 0 otherwise. Thus count if _fillin tells you how many observations were added. You will often want to replace or rename _fillin to something appropriate:

```
.rename _fillin chosen
.replace chosen = 1 - chosen
```

If you do not rename or drop _fillin, it will get in the way of a subsequent fillin. Usually, the decision is clear-cut: Either _fillin has a natural interpretation, so you want to keep it, or a relative, under a different name; or _fillin was just a by-product, and you can get rid of it without distress.

Another common variant is to show zero counts or amounts explicitly. With a dataset of political donations for several years, we might want an observation showing that amount is zero for each pair of donor and year not matched by a donation. This typically leads to neater tables and graphs and may be needed for modeling: in particular, for panel models, the zeros must be present as observations. The main idea is the same, but the aftermath is different:

```
.fillin donor year
.replace amount = 0 if _fillin
```
Naturally if we have more than one donation from various donors in various years, we might also want to collapse (or just possibly contract) the data, but that is the opposite kind of problem.

Yet another common variant is the creation of a grid for some purpose, perhaps before data entry, or before you draw a graph. You can be very lazy by typing

\begin{verbatim}
. clear
. set obs 20
. gen y = _n
. gen x = y
. fillin y x
\end{verbatim}

which creates a $20 \times 20$ grid. This is good, but sometimes you want something different; see functions fill() and seq() in [R] egen.

The messiest fillin problems are when some of the categories you want are not present in the dataset at all. If you know a person is not one of the values of donor, no amount of filling in will add a set of zeros for that person. One strategy here is to add pseudo-observations so that every category occurs at least once and then to fillin in terms of that larger dataset. This is just a variation on the technique for creating a grid out of nothing.

As far as you can see from what is here, fillin just does things in place, so you need not worry about file manipulation. This is an illusion, as underneath the surface, fillin is firing up cross (see [R] cross), which does the work using files. Thus cross is more fundamental. A forthcoming tip will say more.