The Stata Journal publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go “beyond the Stata manual” in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those teaching, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

For more information on the Stata Journal, including information for authors, see the web page

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates:

Subscriptions mailed to US and Canadian addresses:

3-year subscription (includes printed and electronic copy) $153
2-year subscription (includes printed and electronic copy) $110
1-year subscription (includes printed and electronic copy) $ 59
1-year student subscription (includes printed and electronic copy) $ 35

Subscriptions mailed to other countries:

3-year subscription (includes printed and electronic copy) $225
2-year subscription (includes printed and electronic copy) $158
1-year subscription (includes printed and electronic copy) $ 83
1-year student subscription (includes printed and electronic copy) $ 59
3-year subscription (electronic only) $153

Back issues of the Stata Journal may be ordered online at

http://www.stata.com/bookstore/sj.html

The Stata Journal is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the Stata Journal, StataCorp, 4905 Lakeway Drive, College Station TX 77845, USA, or email sj@stata.com.
The fillin command (see \texttt{[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:

\begin{verbatim}
id  choice
 1    1
 2    3
 3    1
 4    2
\end{verbatim}

Now suppose that you wish to run a nested logit model using \texttt{nlogit} (see \texttt{[R] nlogit}). This command requires all choices, those made and those not made, to be explicit. With even 4 values of \texttt{id} and 3 values of \texttt{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

\begin{verbatim}
.fillin id choice
\end{verbatim}

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

\begin{verbatim}
.rename \_fillin chosen
.replace chosen = 1 - chosen
\end{verbatim}

If you do not rename or drop \_fillin, it will get in the way of a subsequent \texttt{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 \texttt{amount} is zero for each pair of \texttt{donor} and \texttt{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:

\begin{verbatim}
.fillin donor year
.replace amount = 0 if \_fillin
\end{verbatim}
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

```
. clear
. set obs 20
. gen y = _n
. gen x = y
. fillin y x
```

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.