mi extract — Extract original or imputed data from mi data

Description

mi extract # replaces the data in memory with the data for \( m = \# \). The data are not \texttt{mi} set.

Menu

Statistics > Multiple imputation

Syntax

\texttt{mi extract \# [, options]}

where \( 0 \leq \# \leq M \)

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear</td>
<td>okay to replace unsaved data in memory</td>
</tr>
<tr>
<td>\texttt{esample(...) }</td>
<td>rarely specified option</td>
</tr>
<tr>
<td>\texttt{esample(varname) }</td>
<td>... syntax when ( # &gt; 0 )</td>
</tr>
<tr>
<td>\texttt{esample(varname #_e)}</td>
<td>... syntax when ( # = 0; 1 \leq #_e \leq M )</td>
</tr>
</tbody>
</table>

Options

\texttt{clear} specifies that it is okay to replace the data in memory even if the current data have not been saved to disk.

\texttt{esample(varname \[#_e\])} is rarely specified. It is for use after \texttt{mi estimate} (see [MI] \texttt{mi estimate}) when the \texttt{esample(newvar)} option was specified to store in \texttt{newvar} the \texttt{e(sample)} for \( m = 1, m = 2, \ldots, m = M \). It is now desired to extract the data for one \( m \) and for \texttt{e(sample)} set correspondingly.

\texttt{mi extract \#, esample(varname \#_e), \# > 0}, is the usual case in this unlikely event. One extracts one of the imputation datasets and redefines \texttt{e(sample)} based on the \texttt{e(sample)} previously stored for \( m = \# \).

The odd case is \texttt{mi extract 0, esample(varname \#_e), \#_e > 0}. One extracts the original data but defines \texttt{e(sample)} based on the \texttt{e(sample)} previously stored for \( m = \#_e \).

Specifying the \texttt{esample()} option changes the sort order of the data.
Remarks and examples

If you wanted to give up on `mi` and just get your original data back, you could type

```
  . mi extract 0
```

You might do this if you wanted to send your original data to a coworker or you wanted to try a different approach to dealing with the missing values in these data. Whatever the reason, the result is that the original data replace the data in memory. The data are not `mi set`. Your original `mi` data remain unchanged.

If you suspected there was something odd about the imputations in `m = 3`, you could type

```
  . mi extract 3
```

You would then have a dataset in memory that looked just like your original, except the missing values of the imputed and passive variables would be replaced with the imputed and passive values from `m = 3`. The data are not `mi set`. Your original data remain unchanged.

Also see

- [MI] Intro — Introduction to `mi`
- [MI] `mi replace0` — Replace original data