Title

stata.com

mi rename — Rename variable

Description

`mi rename` renames variables.

Menu

Statistics > Multiple imputation

Syntax

`mi rename` *oldname newname* [`, noupdate`]

Option

`noupdate` in some cases suppresses the automatic `mi update` this command might perform; see [MI] **noupdate option**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Specifying the noupdate option

If you are renaming more than one variable, you can speed execution with no loss of safety by specifying the `noupdate` option after the first `mi rename`:

```
. mi rename ageyears age
. mi rename timeinstudy studytime, noupdate
. mi rename personid id, noupdate
```

The above is generally good advice. When giving one `mi` command after another, you may specify `noupdate` after the first command to speed execution.

What to do if you accidentally use rename

Assume that you just typed

```
. rename ageyears age
```

rather than typing

```
. mi rename ageyears age
```

as you should have. No damage has been done yet, but if you give another `mi` command and it runs `mi update` (see [MI] **mi update**), real damage will be done. We will discuss that and what to do about it in the sections that follow, but first, if you have given no additional `mi` commands, use `rename` (not `mi rename`) to rename the variable back to how it was:

```
. rename age ageyears
```

Then use `mi rename` as you should have in the first place:

```
. mi rename ageyears age
```

The sections below handle the case where `mi update` has run. You will know that `mi update` has run because since the rename, you gave some `mi` command—perhaps even `mi update` itself—and you saw a message like one of these:

(variable `ageyears` dropped in $m > 0$)

(imputed variable `ageyears` unregistered because not in $m = 0$)

(passive variable `ageyears` unregistered because not in $m = 0$)

(regular variable `ageyears` unregistered because not in $m = 0$)

What to do if you accidentally use rename on wide data

If `ageyears` was unregistered, no damage was done, and no additional action needs to be taken.

If `ageyears` was registered as regular, no damage was done. However, your renamed variable is no longer registered. Reregister the variable under its new name by typing `mi register regular age`; see [MI] **mi set**.

If `ageyears` was registered as imputed or passive, you just lost all values for $m > 0$. Passive variables are usually not too difficult to re-create; see [MI] **mi passive**. If the variable was imputed, well, hope that you will have saved your data recently when you make this error and, before that, learn good computing habits.

What to do if you accidentally use rename on mlong data

If `ageyears` was unregistered, no damage was done, and no additional action needs to be taken.

If `ageyears` was registered as regular, no damage was done. However, your renamed variable is no longer registered. Reregister the variable under its new name by typing `mi register regular age`; see [MI] **mi set**.

If `ageyears` was registered as imputed or passive, you just lost all values for $m > 0$. We offer the same advice as we offered when the data were wide: Passive variables are usually not too difficult to re-create—see [MI] **mi passive**—and otherwise hope that you will have saved your data recently when you make this error. It is always a good idea to save your data periodically.

What to do if you accidentally use rename on flong data

The news is better in this case; no matter how your variables were registered, you have not lost data.

If `ageyears` was unregistered, no further action is required.

If `ageyears` was registered as regular, you need to reregister the variable under its new name by typing `mi register regular age`; see [MI] **mi set**.

If `ageyears` was registered as passive or imputed, you need to reregister the variable under its new name by typing `mi register passive age` or `mi register imputed age`.

What to do if you accidentally use rename on flongsep data

The news is not as good in this case.

If `ageyears` was unregistered, no damage was done. When `mi update` ran, it noticed that old variable `ageyears` no longer appeared in $m > 0$ and that new variable `age` now appeared in $m = 0$, so `mi update` dropped the first and added the second to $m > 0$, thus undoing any damage. There is nothing more that needs to be done.

If `ageyears` was registered as regular, no damage was done, but you need to reregister the variable by typing `mi register regular age`; see [MI] **mi set**.

If `ageyears` was registered as passive or imputed, you have lost the values in $m > 0$. Now would probably be a good time for us to mention how you should work with a copy of your flongsep data; see [MI] **mi copy**.

Also see

[MI] **Intro** — Introduction to mi