mi select — Programmer's alternative to mi extract

Description Syntax Option Remarks and examples
Stored results Also see

Description

`mi select` is a programmer's command. It is a faster, more dangerous version of `mi extract`; see [MI] **mi extract**. Before using `mi select`, the `mi` data must be preserved; see [P] **preserve**.

`mi select init` initializes `mi select #` and must be used before the first call to `mi select #`.

`mi select #` replaces the data in memory with a copy of the data for $m = \#$. The data are not `mi set`.

Syntax

```
mi select init [ , fast ]

mi select #
```

where $0 \leq \# \leq M$, and where typical usage is

```
quietly mi query
local M = r(M)

preserve
mi select init
local priorcmd "`r(priorcmd)'"

forvalues m=1(1)`M' {
    mi select `m'
    ...
    `priorcmd'
}

restore
```

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Option

`fast`, specified with `mi select init`, specifies that the data delivered by `mi select #` commands not be changed except for sort order. Then `mi select` can operate more quickly. `fast` is allowed with all styles but currently affects the performance with the wide style only.

If `fast` is not specified, the data delivered by `mi select #` may be modified freely before the next `mi select #` call. However, the data may not be dropped. `mi select` uses characteristics (see [P] **char**) stored in `_dta[]` to know its state.

Remarks and examples

The two `mi select` commands work in tandem. `mi select init` initializes `mi select #`.

`mi select init` returns macro `r(priorcmd)`, which you are to issue as a command between each `mi select #` call. `r(priorcmd)` is not required to be issued before the first call to `mi select #`, although you may issue it if that is convenient. `mi select #` calls can be made in any order, and the same m may be selected repeatedly.

The data delivered by `mi select #` differ from those delivered by `mi extract` in that there may be extra variables in the dataset. One of the extra variables, `_mi_id`, is a unique observation identifier.

If you want to post changes made in the selected data back to the `mi` data, you can write a file containing `mi_id` and the updated variables and then use `_mi_id` to match that to the `mi` data after your final `restore`. By default, changes to the selected data will not be posted back to the underlying `mi` data.

In the case of wide data, the `mi` data have no `_mi_id` variable. `_mi_id` in the selected data is reflected in the current order of the `mi` data.

Stored results

`mi select init` returns the following in `r()`:

Macros

`r(priorcmd)` command to be issued prior to calling `mi select #`; this command will be either `restore`, `preserve` or nothing

Also see

[MI] **Intro** — Introduction to mi

[MI] **mi extract** — Extract original or imputed data from mi data

[MI] **Technical** — Details for programmers

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.

For suggested citations, see the FAQ on citing Stata documentation.