noupdate option — The noupdate option

Description Syntax Option Remarks and examples Also see

Description

Many `mi` commands allow the `noupdate` option. This entry describes the purpose of that option.

Syntax

```
mi ... [ , ... noupdate ... ]
```

Option

`noupdate` specifies that the `mi` command in question need not perform an `mi update` because you are certain that there are no inconsistencies that need fixing; see [MI] **mi update**. `noupdate` is taken as a suggestion; `mi update` will still be performed if the command sees evidence that it needs to be. Not specifying the option does not mean that an `mi update` will be performed.

Remarks and examples

Some `mi` commands perform modifications to the data, and those modifications will go very poorly—even to the point of corrupting your data—if certain assumptions about your data are not true. Usually, those assumptions are true, but to be safe, the commands check the assumptions. They do this by calling `mi update`; see [MI] **mi update**. `mi update` checks the assumptions and, if they are not true, corrects the data so that the assumptions are true. `mi update` always reports the data corrections it makes.

All of this reflects an abundance of caution, with the result that some commands spend more time running `mi update` than they spend performing their intended task.

Commands that use `mi update` to verify assumptions have a `noupdate` option. When you specify that option, the command skips checking the assumptions, which is to say it skips calling `mi update`. More correctly, the command skips calling `mi update` if the command sees no obvious evidence that `mi update` needs to be called.

You can make commands run faster by specifying `noupdate`. Should you? Unless you are noticing poor performance, we would say no. It is, however, absolutely safe to specify `noupdate` if the only commands executed since the last `mi update` are `mi` commands. The following would be perfectly safe:

```
. mi update
. mi passive, noupdate: generate agesq = age*age
. mi rename age age_at_admission, noupdate
. mi ...
```

The following would be safe, too:

```
. mi update
. mi passive, noupdate: generate agesq = age*age
. summarize agesq
. mi rename age age_at_admission, noupdate
. mi ...
```

It would be safe because `summarize` is a reporting command that does not change the data; see [R] **summarize**.

The problem `mi` has is that it is not in control of your session and data. Between `mi` commands, `mi` does not know what you have done to the data. The following would not be recommended and has the potential to go very poorly:

```
. mi update
. mi passive, noupdate: generate agesq = age*age
. drop if female
. drop agesq
. mi ..., noupdate          // do not do this
```

By the rules for using `mi`, you should perform an `mi update` yourself after a `drop` command, or any other command that changes the data, but it usually does not matter whether you follow that rule because `mi` will check eventually, when it matters. That is, `mi` will check if you do not specify the `noupdate` option.

The `noupdate` option is recommended for use by programmers in programs that code a sequence of `mi` commands.

Also see

[MI] **Intro** — Introduction to mi

[MI] **mi update** — Ensure that mi data are consistent