mi describe — Describe mi data

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`mi query` reports whether the data in memory are `mi` data and, if they are, reports the style in which they are set.

`mi describe` provides a more detailed report on `mi` data.

Menu

Statistics > Multiple imputation

Syntax

`mi query`

`mi describe` [, *describe_options*]

describe_options	Description
`detail`	show missing-value counts for $m = 1$, $m = 2$, …
`noupdate`	see [MI] **noupdate option**

`collect` is allowed with `mi query` and `mi describe`; see [U] **11.1.10 Prefix commands**.

Options

`detail` reports the number of missing values in $m = 1$, $m = 2$, …, $m = M$ in the imputed and passive variables, along with the number of missing values in $m = 0$.

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

Remarks and examples

Remarks are presented under the following headings:

mi query
mi describe

mi query

`mi query` without `mi` data in memory reports

```
. mi query
(data not mi set)
```

With `mi` data in memory, you see something like

```
. mi query
data mi set wide, M = 15
last mi update 20jan2025 15:30:20, approximately 5 minutes ago
```

`mi query` does not burden you with unnecessary information. It mentions when `mi update` was last run because you should run it periodically; see [MI] **mi update**.

mi describe

`mi describe` more fully describes `mi` data:

```
. mi describe
Style: mlong
       last mi update 22dec2024 15:30:20, approximately 2 minutes ago
Observations:
   Complete            90
   Incomplete          10  (M = 20 imputations)
   ---------------------
   Total               100
Variables:
   Imputed: 2; smokes(10) age(5)
   Passive: 1; agesq(5)
   Regular: 0
   System:  3; _mi_m _mi_id _mi_miss
   (there are 3 unregistered variables; gender race chd)
```

`mi describe` lists the style of the data, the number of complete and incomplete observations, M (the number of imputations), the registered variables, and the number of missing values in $m = 0$ of the imputed and passive variables. In the output, the lines

```
Variables:
   Imputed: 2; smokes(10) age(5)
```

means that the `smokes` variable contains 10 missing values in $m = 0$ and that `age` contains 5. Those values are soft missings and thus eligible to be imputed. If one of `smokes`' missing values in $m = 0$ were hard, the lines would read

```
Variables:
   Imputed: 2; smokes(9+1) age(5)
```

`mi describe` reports information about $m = 0$. To obtain information about all m's, use `mi describe, detail`:

```
. mi describe, detail
Style: mlong
       last mi update 22dec2024 15:30:20, approximately 3 minutes ago
Observations:
   Complete            90
   Incomplete          10  (M = 20 imputations)
   ----------------------
   Total              100
Variables:
   Imputed: 2; smokes(10; 20*0) age(5; 20*0)
   Passive: 1; agesq(5; 20*0)
   Regular: 0
   System:  3; _mi_m _mi_id _mi_miss
   (there are 3 unregistered variables; gender race chd)
```

In this example, all imputed values are nonmissing. We can see that from

```
Variables:
   Imputed: 2; smokes(10; 20*0) age(5; 20*0)
```

Note the `20*0` after the semicolons. That is the number of missing values in $m = 1$, $m = 2$, …, $m = 20$. In the `smokes` variable, there are 10 missing values in $m = 0$, then 0 in $m = 1$, then 0 in $m = 2$, and so on. If $m = 17$ had two missing imputed values, the lines would read

```
Variables:
   Imputed: 2; smokes(10; 16*0, 2, 3*0) age(5; 20*0)
```

`16*0, 2, 3*0` means that for $m = 1$, $m = 2$, …, $m = 20$, the first 16 have 0 missing values, the next has 2, and the last 3 have 0.

If `smokes` had $9 + 1$ missing values rather than 10—that is, 9 soft missing values plus 1 hard missing rather than all 10 being soft missing—and all 9 soft missings were filled in, the line would read

```
Variables:
   Imputed: 2; smokes(9+1; 20*0) age(5; 20*0)
```

The 20 imputations are shown as having no soft missing values. It goes without saying that they have 1 hard missing. Think of `20*0` as meaning `20*(0+1)`.

If `smokes` had $9 + 1$ missing values and two of the soft missings in $m = 18$ were still missing, the line would read

```
Variables:
   Imputed: 2; smokes(9+1; 16*0, 2, 3*0) age(5; 20*0)
```

Stored results

`mi query` stores the following in `r()`:

Scalars

`r(update)`	seconds since last `mi update`
`r(m)`	m if `r(style)=="flongsep"`
`r(M)`	M if `r(style)!="flongsep"`

Macros

`r(style)`	*style*
`r(name)`	*name* if `r(style)=="flongsep"`

Note that `mi query` issues a return code of 0 even if the data are not `mi`. In that case, `r(style)` is " ".

`mi describe` stores the following in `r()`:

Scalars

`r(update)`	seconds since last `mi update`
`r(N)`	number of observations in $m = 0$
`r(N_incomplete)`	number of incomplete observations in $m = 0$
`r(N_complete)`	number of complete observations in $m = 0$
`r(M)`	M

Macros

`r(style)`	*style*
`r(ivars)`	names of imputed variables
`r(_0_miss_ivars)`	$\#=$. in each `r(ivars)` in $m = 0$
`r(_0_hard_ivars)`	$\#>$. in each `r(ivars)` in $m = 0$
`r(pvars)`	names of passive variables
`r(_0_miss_pvars)`	$\#\geq$. in each `r(pvars)` in $m = 0$
`r(rvars)`	names of regular variables

If the `detail` option is specified, for each m, $m = 1, 2, \ldots, M$, also stored are

Macros

`r(_`m`_miss_ivars)`	$\#=$. in each `r(ivars)` in m
`r(_`m`_miss_pvars)`	$\#\geq$. in each `r(pvars)` in m

Also see

[MI] **Intro** — Introduction to mi

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.