

mi ptrace — Load parameter-trace file into Stata

Description Stored results	Syntax Also see	Options	Remarks and examples
-------------------------------	--------------------	---------	----------------------

Description

Parameter-trace files, files with suffix `.stptrace`, are created by the `saveptrace()` option of `mi impute mvn`; see [MI] **mi impute mvn**. These are not Stata datasets, but they can be loaded as if they were by using `mi ptrace use`. Their contents can be described without loading them by using `mi ptrace describe`.

Syntax

```
mi ptrace describe [using] filename
```

```
mi ptrace use filename [, use_options]
```

<i>use_options</i>	Description
<code>clear</code>	okay to replace existing data in memory
<code>double</code>	load variables as doubles (default is floats)
<code><u>select</u>(<i>selections</i>)</code>	what to load (default is all)

where *selections* is a space-separated list of individual selections. Individual selections are of the form

```
b[yname, xname]  
v[yname, yname]
```

where *ynames* and *xnames* are displayed by `mi ptrace describe`. You may also specify

```
b[#_y, #_x]  
v[#_y, #_y]
```

where `#_y` and `#_x` are the variable numbers associated with *yname* and *xname*, and those too are shown by `mi ptrace describe`.

For `b`, you may also specify `*` to mean all possible index elements. For instance,

```
b[*,*]      all elements of b  
b[yname,*] row corresponding to yname  
b[*,xname] column corresponding to xname
```

Similarly, `b[#_y,*]` and `b[*,#_x]` are allowed. The same is allowed for `v`, and also, the second element can be specified as `<`, `<=`, `=`, `>=`, or `>`. For instance,

<code>v[yname,=]</code>	variance of <i>yname</i>
<code>v[*,=]</code>	all variances (diagonal elements)
<code>v[*,<]</code>	lower triangle
<code>v[*,<=]</code>	lower triangle and diagonal
<code>v[*,>=]</code>	upper triangle and diagonal
<code>v[*,>]</code>	upper triangle

In `mi ptrace describe` and in `mi ptrace use`, *filename* must be specified in quotes if it contains special characters or blanks. *filename* is assumed to be *filename.stptrace* if the suffix is not specified.

Options

`clear` specifies that it is okay to clear the dataset in memory, even if it has not been saved to disk since it was last changed.

`double` specifies that elements of `b` and `v` are to be loaded as doubles; they are loaded as floats by default.

`select(selections)` allows you to load subsets of `b` and `v`. If the option is not specified, all of `b` and `v` are loaded. That result is equivalent to specifying `select(b[*,*] v[*,<=])`. The `<=` specifies that just the diagonal and lower triangle of symmetric matrix `v` be loaded.

Specifying `select(b[*,*])` would load just `b`.

Specifying `select(v[*,<=])` would load just `v`.

Specifying `select(b[*,*] v[*,<=])` would load `b` and the diagonal elements of `v`.

Remarks and examples

[stata.com](https://www.stata.com)

Say that we impute the values of y_1 and y_2 assuming that they are multivariate normal distributed, with their means determined by a linear combination of x_1 , x_2 , and x_3 , and their variance constant. Writing this more concisely, $\mathbf{y} = (y_1, y_2)'$ is distributed $\text{MVN}(\mathbf{XB}, \mathbf{V})$, where \mathbf{B} : 2×3 and \mathbf{V} : 2×2 . If we use MCMC or EM procedures to produce values of \mathbf{B} and \mathbf{V} to be used to generate values for \mathbf{y} , we must ensure that we use sufficient iterations so that the iterative procedure stabilizes. `mi impute mvn` (see [\[MI\] mi impute mvn](#)) provides the worst linear combination (WLC) of the elements of \mathbf{B} and \mathbf{V} . If we want to perform other checks, we can specify `mi impute mvn's saveptrace(filename)` option. `mi impute` then produces a file containing `m` (imputation number), `iter` (overall iteration number), and the corresponding \mathbf{B} and \mathbf{V} . The last `iter` for each `m` is the \mathbf{B} and \mathbf{V} that `mi impute mvn` used to impute the missing values.

When we used `mi impute mvn`, we specified burn-in and burn-between numbers, say, `burnin(300)` and `burnbetween(100)`. If we also specified `saveptrace()`, the file produced is organized as follows:

record #	m	iter	B	V	
1	1	-299	
2	1	-298	
.	
.	
299	1	-1	
300	1	0	<- used to impute m=1
301	2	1	.	.	
302	2	2	.	.	
.	
.	
399.	1	99	
400.	1	100	<- used to impute m=2
401.	2	101	
.	
.	

The file is not a Stata dataset, but `mi ptrace use` can load the file and convert it into Stata format, and then it will look just like the above except for the following:

- The record number will become the Stata observation number.
- B will become variables `b_y1x1`, `b_y1x2`, and `b_y1x3`; and `b_y2x1`, `b_y2x2`, and `b_y2x3`. (Remember, we had 2 *y* variables and 3 *x* variables.)
- V will become variables `v_y1y1`, `v_y2y1`, and `v_y2y2`. (This is the diagonal and lower triangle of **V**; variable `v_y1y2` is not created because it would be equal to `v_y2y1`.)
- Variable labels will be filled in with the underlying names of the variables. For instance, the variable label for `b_y1x1` might be “experience, age”, and that would remind us that `b_y1x1` contains the coefficient on age used to predict experience. `v_y2y1` might be “education, experience”, and that would remind us that `v_y2y1` contains the covariance between education and experience.

Stored results

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

```
Scalars
  r(tc)      %tc date-and-time file created
  r(nx)      number of x variables (columns of B)
  r(ny)      number of y variables (rows of B)
Macros
  r(x)       space-separated [op.]varname of x
  r(y)       space-separated [op.]varname of y
  r(id)      name of file creator
```

Also see

[MI] [intro](#) — Introduction to mi

[MI] [mi impute mvn](#) — Impute using multivariate normal regression