Title

stata.com

mi import nhanes1 — Import NHANES-format data into mi

Description

`mi import nhanes1` imports data recorded in the format used by the National Health and Nutrition Examination Survey (NHANES) produced by the National Center for Health Statistics (NCHS) of the U.S. Centers for Disease Control and Prevention (CDC); see https://www.cdc.gov/nchs/data/nhanes/nhanes3/dna_secondary_data_analysis_guidelines.pdf.

Menu

Statistics > Multiple imputation

Syntax

`mi import nhanes1` *name*`,` *required_options* [*true_options* *odd_options*]

where *name* is the name of the flongsep data to be created.

required_options	Description
`using(`*filenamelist*`)`	input filenames for $m = 1$, $m = 2$, ...
`id(`*varlist*`)`	identifying variable(s)

Note: `use` the input file for m=0 before issuing `mi import nhanes1`.

true_options	Description
`uppercase`	prefix and suffix in uppercase
`clear`	okay to replace unsaved data in memory

odd_options	Description
`nacode(#)`	not applicable code; default is `0`
`obscode(#)`	observed code; default is `1`
`impcode(#)`	imputed code; default is `2`
`impprefix("`*string*`" "`*string*`")`	variable prefix; default is `"" ""`
`impsuffix("`*string*`" "`*string*`")`	variable suffix; default is `"if" "mi"`

Note: The *odd_options* are not specified unless you need to import data that are nhanes1-like but not really nhanes1 format.

Options

`using(`*filenamelist*`)` is required; it specifies the names of the `.dta` datasets containing $m = 1$, $m = 2$, ..., $m = M$. The dataset corresponding to $m = 0$ is not specified; it is to be in memory at the time the `mi import nhanes1` command is given.

The filenames might be specified as

```
using(nh1 nh2 nh3 nh4 nh5)
```

which states that $m = 1$ is in file `nh1.dta`, $m = 2$ is in file `nh2.dta`, ..., and $m = 5$ is in file `nh5.dta`. Also, `{#-#}` is understood, so the files could just as well be specified as

```
using(nh{1-5})
```

The braced numeric range may appear anywhere in the name, and thus

```
using(nh{1-5}imp)
```

would mean that `nh1imp.dta`, `nh2imp.dta`, ..., `nh5imp.dta` contain $m = 1$, $m = 2$, ..., $m = 5$.

Alternatively, a comma-separated list can appear inside the braces. Filenames `nhfirstm.dta`, `nhsecondm.dta`, ..., `nhfifthm.dta` can be specified as

```
using(nh{first,second,third,fourth,fifth}m)
```

Filenames can be specified with or without the `.dta` suffix and must be enclosed in quotes if they contain special characters.

`id(`*varlist*`)` is required and is usually specified as `id(seqn)` or `id(SEQN)` depending on whether your variable names are in lowercase or uppercase. `id()` specifies the variable or variables that uniquely identify the observations in each dataset. Per the nhanes1 standard, the variable should be named `seqn` or `SEQN`.

`uppercase` is optional; it specifies that the variable suffixes `IF` and `MI` of the nhanes1 standard are in uppercase. The default is lowercase. (More correctly, when generalizing beyond nhanes1 format, the `uppercase` option specifies that all prefixes and suffixes are in uppercase.)

`nacode(#)`, `obscode(#)`, and `impcode(#)` are optional and are never specified when reading true nhanes1 data. The defaults `nacode(0)`, `obscode(1)`, and `impcode(2)` correspond to the nhanes1 definition. These options allow changing the codes for not applicable, observed, and imputed.

`impprefix("`*string*`" "`*string*`")` and `impsuffix("`*string*`" "`*string*`")` are optional and are never specified when reading true nhanes1 data. The defaults `impprefix("" "")` and `impsuffix("if" "mi")` correspond to the nhanes1 definition. These options allow setting different prefixes and suffixes.

`clear` specifies that it is okay to replace the data in memory even if they have changed since they were saved to disk. Remember, `mi import nhanes1` starts with the first of the NHANES data in memory and ends with `mi` data in memory.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Description of the nhanes1 format

Nhanes1 is not really an official format; it is the format used for a particular dataset distributed by NCHS. Because there currently are no official or even informal standards for multiple-imputation data, perhaps the method used by the NCHS for NHANES will catch on, so we named it nhanes1. We included the 1 on the end of the name in case the format is modified.

Data in nhanes1 format consist of a collection of $M + 1$ separate files. The first file contains the original data. The remaining M files contain the imputed values for $m = 1$, $m = 2$, ..., $m = M$.

The first file contains a variable named `seqn` containing a sequence number. The file also contains other variables that comprise the nonimputed variables. Imputed variables, however, have their names suffixed with `IF`, standing for imputation flag, and those variables contain 1s, 2s, and 0s. 1 means that the value of the variable in that observation was observed, 2 means that the value was missing, and 0 means not applicable. Think of 0 as being equivalent to hard missing. The value is not observed for good reason and therefore was not imputed.

The remaining M files contain `seqn` and the imputed variables themselves. In these files, unobserved values are imputed. This time, imputed variable names are suffixed with `MI`.

Here is an example:

```
. use https://www.stata-press.com/data/r17/nhorig
. list
```

	seqn	a	bIF	cIF
1.	1	11	1	1
2.	2	14	2	2

The above is the first of the $M + 1$ datasets. The `seqn` variable is the sequence number. The `a` variable is a regular variable; we know that because the name does not end in `IF`. The `b` and `c` variables are imputed, and this dataset contains their imputation flags. Both variables are observed in the first observation and unobserved in the second.

Here is the corresponding dataset for $m = 1$:

```
. use https://www.stata-press.com/data/r17/nh1
. list
```

	seqn	bMI	cMI
1.	1	2	3
2.	2	4.5	8.5

```
. save nh1
file nh1.dta saved
```

This dataset states that in $m = 1$, `b` is equal to 2 and 4.5 and `c` is equal to 3 and 8.5.

We are about to show you the dataset for $m = 2$. Even before looking at it, however, we know that 1) it will have two observations; 2) it will have the `seqn` variable containing 1 and 2; 3) it will have two more variables named `bMI` and `cMI`; and 4) `bMI` will be equal to 2 and `cMI` will be equal to 3 in observations corresponding to `seqn` $= 1$. We know the last because in the first dataset, we learned that `b` and `c` were observed in `seqn` $= 1$.

```
. use https://www.stata-press.com/data/r17/nh2
. list
```

	seqn	a	bMI	cMI
1.	1	11	2	3
2.	2	14	5.5	9.5

```
. save nh2
file nh2.dta saved
```

Importing nhanes1 data

The procedure to import nhanes1 data is this:

1. `use` the dataset corresponding to $m = 0$; see [D] **use**.
2. Issue `mi import nhanes1` *name* ..., where *name* is the name of the `mi` flongsep dataset to be created.
3. Perform the checks outlined in *Using mi import nhanes1, ice, flong, and flongsep* of [MI] **mi import**.
4. Use `mi convert` (see [MI] **mi convert**) to convert the data to a more convenient style such as wide, mlong, or flong.

To import the `nhorig.dta`, `nh1.dta`, and `nh2.dta` datasets described in the section above, we will specify `mi import nhanes1`'s `uppercase` option because the suffixes were in uppercase. We type

```
. use https://www.stata-press.com/data/r17/nhorig
. mi import nhanes1 mymi, using(nh1 nh2) id(seqn) uppercase
```

The lack of any error message means that we have successfully converted nhanes1-format files `nhorig.dta`, `nh1.dta`, and `nh2.dta` to `mi` flongsep files `mymi.dta`, `_1_mymi.dta`, and `_2_mymi.dta`.

We will now perform the checks outlined in *Using mi import nhanes1, ice, flong, and flongsep* of [MI] **mi import**, which are to run `mi describe` and `mi varying` (see [MI] **mi describe** and [MI] **mi varying**) to verify that variables are registered correctly:

```
. mi describe
Style: flongsep mymi
       last mi update 30apr2021 22:49:45, 0 seconds ago
Observations:
   Complete            1
   Incomplete          1  (M = 2 imputations)
   ---------------------
   Total               2
Variables:
   Imputed: 2; b(1) c(1)
   Passive: 0
   Regular: 0
   System:  2; _mi_id _mi_miss
   (there are 2 unregistered variables; seqn a)
```

```
. mi varying
            Possible problem   Variable names
-------------------------------------------------------------------------------
         imputed nonvarying:  (none)
         passive nonvarying:  (none)
       unregistered varying:  (none)
*unregistered super/varying:  (none)
 unregistered super varying:  (none)
-------------------------------------------------------------------------------
* super/varying means super varying but would be varying if registered as
  imputed; variables vary only where equal to soft missing in m=0.
```

`mi varying` reported no problems.

We finally convert to style flong, although in real life we would choose styles mlong or wide. We are choosing flong because it is more readable:

```
. mi convert flong, clear
. list, separator(2)
```

	seqn	a	b	c	_mi_id	_mi_miss	_mi_m
1.	1	11	2	3	1	0	0
2.	2	14	.	.	2	1	0
3.	1	11	2	3	1	.	1
4.	2	14	4.5	8.5	2	.	1
5.	1	11	2	3	1	.	2
6.	2	14	5.5	9.5	2	.	2

The flong data are in memory. We are done with the converted data in flongsep format, so we erase the files:

```
. mi erase mymi
(files mymi.dta _1_mymi.dta _2_mymi.dta erased)
```

Also see

[MI] **Intro** — Introduction to mi

[MI] **mi import** — Import data into mi