Title

stata.com

vl set — Set system-defined variable lists

Description Quick start Syntax Options Remarks and examples
Stored results Also see

Description

`vl set` is designed to identify variables that are to be treated as factor variables in Stata's estimation commands.

`vl set` creates the system-defined variable lists `vlcategorical`, `vlcontinuous`, `vluncertain`, and `vlother`. Variables are placed in them based on their values (integer or noninteger, all nonnegative, etc.) and default or user-specified cutoffs for the number of levels in a variable.

`vl move` moves variables from one classification to another.

Variable lists are actually global macros, and they are saved with the dataset. See [D] **vl rebuild**.

For an introduction to the `vl` commands, see [D] **vl**.

Quick start

Classify all numeric variables in the dataset

```
vl set
```

Same as above, and include a `vldummy` classification for 0/1 variables

```
vl set, dummy
```

Classify all numeric variables in the dataset, and list each variable as it is classified

```
vl set, list
```

Put nonnegative integer variables with 6 or fewer categories into `vlcategorical`; put nonnegative integer variables with 7–20 categories into `vluncertain`; put nonnegative integer variables with more than 20 categories into `vlcontinuous`

```
vl set, categorical(6) uncertain(20)
```

Classify only the variables `x1-x100`

```
vl set x1-x100
```

Discard the existing classifications, and classify all numeric variables again

```
vl set, clear
```

Redo the classification of the variable `age`

```
vl set age, redo
```

Update the stored statistics for the variable `age`, but do not change its classification

```
vl set age, update
```

Move the variables `x8` and `x20` out of their current classification and into `vlcategorical`

```
vl move (x8 x20) vlcategorical
```

Move all the variables in vluncertain into vlcontinuous

```
vl move vluncertain vlcontinuous
```

Syntax

Create system-defined variable lists

```
vl set [ varlist ] [ , options ]
```

Move variables from their current system-defined variable list to another

```
vl move (varlist) vlsysname
```

Move all variables in one system-defined variable list to another

```
vl move vlsysname1 vlsysname2
```

varlist contains only numeric variables. If not specified, then all numeric variables in the dataset are classified.

options	Description
categorical(#)	upper limit for the number of categories in vlcategorical
uncertain(#)	upper limit for the number of categories in vluncertain
dummy	create variable list vldummy containing 0/1 variables
list[(*list_options*)]	list variables as they are classified
clear	discard all existing classifications and make new classifications
redo	redo classifications for variables in *varlist*
update	update stored statistics for variables in *varlist*, but do not change their classification
nonotes	suppress the notes below the summary table

collect is allowed with vl set; see [U] 11.1.10 Prefix commands.

Options

categorical(#) specifies that variables containing nonnegative integers be put into the vlcategorical variable list when the number of levels is between 2 and # inclusive. Variables with only one level (that is, constants) are put into the vlother variable list. The default is categorical(10).

categorical(.) can be specified to set the upper limit effectively to infinity. That is, all variables containing nonnegative integers (whose values are less than $2^{31} = 2{,}147{,}483{,}648$) are put into vlcategorical. Setting # to . or a large value can slow computation time considerably when the number of observations is extremely large.

uncertain(#) specifies that variables containing nonnegative integers be put into the vluncertain variable list when the number of levels are between categorical(#) + 1 and # inclusive. The default is uncertain(100).

must be $\geq$ categorical(#). To omit the vluncertain classification, set # = categorical(#) or specify uncertain(0).

`uncertain(.)` can be specified to set the upper limit effectively to infinity. That is, all variables containing nonnegative integers (whose values are less than $2^{31} = 2{,}147{,}483{,}648$) with more than `categorical(#)` levels are put into `vluncertain`. Setting `#` to `.` or a large value can slow computation time considerably when the number of observations is extremely large.

`dummy` specifies that a `vldummy` variable list be created containing 0/1 variables. By default, 0/1 variables are put into `vlcategorical`.

`list`[`(`*list_options*`)`] lists variables as they are classified. The classification is shown as well as the number of levels for variables in `vlcategorical` and `vluncertain`. *list_options* are as follows:

`minimum` shows the minimum value of each variable;

`maximum` shows the maximum value of each variable; and

`observations` shows the number of nonmissing values of each variable.

The same listing can be obtained using `vl list` after running `vl set`.

`clear` specifies that all the system-defined variable lists (if any) be dropped and the classifications redone. It is equivalent to running `vl clear, system` and then running `vl set`.

`redo` specifies that the classifications be redone for the variables in *varlist*. It is equivalent to running `vl drop (`*varlist*`), system` and then running `vl set` *varlist*.

`update` specifies that all statistics (number of levels, minimum value, maximum value, and number of nonmissing observations) that are saved for the variables in *varlist* be updated but the classifications of the variables not be changed. `update` is intended for use when observations are added to or dropped from the data and you want the classifications to remain unchanged.

`nonotes` specifies that the notes at the bottom of the summary table not be displayed. By default, the notes are shown.

Remarks and examples

stata.com

`vl set` creates the system-defined variable lists `vlcategorical`, `vlcontinuous`, `vluncertain`, and `vlother`.

The `vlcategorical` variable list is intended for variables that will be used as factor variables in estimation commands.

The `vlcontinuous` variable list is intended for variables that will be used as continuous variables in estimation commands.

The `vluncertain` variable list is intended for variables that we may want to treat as factors or as continuous, and we will decide which on a case-by-case basis. As we decide, we use `vl move` to move them out of `vluncertain` and into `vlcategorical` or `vlcontinuous`. For example, we decide we want variable `q31`, currently in `vluncertain`, to be a factor variable. We type

```
. vl move (q31) vlcategorical
```

In the above, note that `q31` is enclosed in parentheses. *varlist*s must always be enclosed in parentheses in `vl move`.

When `q31` is moved into `vlcategorical`, it is automatically moved out of `vluncertain`. The system-defined variable lists are always kept as disjoint sets. That is, a variable can only appear in one system-defined variable list. User-defined variable lists can be made to be overlapping. See [D] **vl create** and [D] **vl**.

Suppose we look at the remaining variables in `vluncertain`, and we decide that they all should be treated as continuous. We type

```
. vl move vluncertain vlcategorical
```

Suppose we look at the remaining variables in `vluncertain`, and we decide we do not want any of them in any of the estimation commands we wish to run. We could move them to `vlother`.

```
. vl move vluncertain vlother
```

`vlother` is intended to be a garbage classification for variables you do not want to use in estimation commands. `vl set` puts variables that are constant and variables that are missing for all observations into `vlother`.

Suppose, however, we simply want some variables gone from the system-defined variable lists. We do not want them shown when we do a `vl list`. To make them gone, gone, gone, use `vl drop`.

```
. vl drop (varlist), system
```

This removes the variables in *varlist* from the system-defined variable lists.

We can also

```
. vl drop vluncertain
```

This removes all the variables in `vluncertain`. `vluncertain` still exists, but it is empty. We can still move other variables into it if we want. System-defined variable lists always exist although they may be empty. They cannot be renamed. If you do not like this behavior, you can create your own variable lists using `vl create`. For example,

```
. vl create mycat = vlcategorical
. vl create mycont = vlcontinuous
```

If you are done using the system-defined variable lists and do not want them around, you can remove them by typing

```
. vl clear, system
```

The system-defined variable lists will be gone, but user-defined variable lists will remain. When you clear the system-defined variable lists, you also erase the statistics that are stored with each variable in the system.

When `vl set` runs, it calculates the minimum, maximum, and number of nonmissing observations for each variable. It also computes the number of levels for the variables in `vlcategorical` and `vluncertain`. It does not compute the number of levels for other variables. That is why `vl set` is so fast even when there are millions of observations.

Computing the exact number of levels when there are thousands of levels can be time consuming. You can have `vl set` compute the number of levels for more variables by specifying the option `uncertain(#)` and setting `#` to a large number or missing (`.`). But expect it to be much slower when there are lots of observations.

To use variable lists with other Stata commands, type `$` in front of the variable-list name. Remember: With the `vl` commands, do not use `$`. With other Stata commands, use `$`.

```
. display "$vlcategorical"
. summarize $vlcontinuous
. regress y i.($vlcategorical) $vlcontinuous
```

If you know Stata, you will have already sensed that variable lists are global macros.

In this example, we used `i.($vlcategorical)` to turn the variables in `vlcategorical` into factor variables. More likely, however, you will want to create your own variable lists based on the system-defined variable lists, and then apply factor-variable operators. The `vl create`, `vl modify`, and `vl substitute` commands were designed for this purpose. See [D] **vl create**.

Variable lists are saved with the dataset. Not only are variable lists saved but also all the `vl` system information and variable statistics are saved. To make the `vl` system come back to life in the state we last had it, after we `use` a dataset, we type

```
. vl rebuild
```

See [D] **vl rebuild**.

For examples of using `vl set` and its options, see [D] **vl**.

Stored results

`vl set` stores the following in `r()`:

Scalars

`r(k_system)`	number of variables in system-defined variable lists
`r(k_vlcategorical)`	number of variables in `vlcategorical`
`r(k_vlcontinuous)`	number of variables in `vlcontinuous`
`r(k_vluncertain)`	number of variables in `vluncertain`
`r(k_vlother)`	number of variables in `vlother`
`r(k_vldummy)`	number of variables in `vldummy` when defined

Macros

`r(vlsysnames)`	names of system-defined variable lists

Also see

[D] **vl** — Manage variable lists

[D] **vl create** — Create and modify user-defined variable lists

[D] **vl drop** — Drop variable lists or variables from variable lists

[D] **vl list** — List contents of variable lists

[D] **vl rebuild** — Rebuild variable lists