Title

stata.com

vl create — Create and modify user-defined variable lists

Description Quick start Syntax Remarks and examples Also see

Description

`vl create` creates user-defined variable lists.

`vl modify` modifies existing user-defined variable lists.

`vl substitute` creates a variable list using factor-variable operators operating on variable lists.

After creating a variable list called *vlusername*, the expression $*vlusername* can be used in Stata anywhere a *varlist* is allowed. Variable lists are actually global macros, and the `vl` commands are a convenient way to create and manipulate them. They are saved with the dataset. See [D] **vl rebuild**.

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

Quick start

Create a variable list

```
vl create demographics = (age_cat gender)
```

Add variables to a variable list

```
vl modify demographics = demographics + (educ_cat income_cat)
```

Add the variables in the variable list named `othervars` to the existing variable list called `myxvars`

```
vl modify myxvars = myxvars + othervars
```

Remove the variable `x8` from the variable list

```
vl modify myxvars = myxvars - (x8)
```

Apply factor-variable operator `i.` to all the variables in a variable list

```
vl substitute idemographics = i.demographics
```

Create interactions between the levels of the variables in the variable list `demographics` and the continuous variables in the variable list `vlcontinuous`

```
vl substitute myinteractions = i.demographics#c.vlcontinuous
```

Run a regression specifying the independent variables using variable lists

```
regress y $idemographics $myxvars $myinteractions
```

Syntax

Create user-defined variable lists

```
vl create vlusername = (varlist)
vl create vlusername = vlname +|- (varlist)
vl create vlusername = vlname1 [+|- vlname2]
```

Modify user-defined variable lists

```
vl modify vlusername = (varlist)
vl modify vlusername = vlname +|- (varlist)
vl modify vlusername = vlname1 [+|- vlname2]
```

Apply factor-variable operators to variable-list names

```
vl substitute vlusername = i.vlname
vl substitute vlusername = i.vlname1#i.vlname2
vl substitute vlusername = i.vlname1##c.vlname2
```

Label a user-defined variable-list name

```
vl label vlusername ["label"]
```

vlname is an existing user-defined variable-list name or a system-defined variable-list name. When specifying *varlist*, it is always enclosed in parentheses: (*varlist*). See [D] **vl**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

vl create

`vl create` creates a new variable list. It can be created from a list of variables:

```
. vl create myxvars = (x1-x100)
```

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

When we are discussing the `vl` commands and say "variable list," we mean a named variable list created by `vl create` or `vl set`. In this case, we created the variable list `myxvars`. A traditional Stata list of variables, that is, a *varlist*, we will call a *varlist*.

A new variable list also can be created from an existing variable list:

```
. vl create indepvars = myxvars
```

vl modify

`vl modify` is the same as `vl create`, except that `vl modify` cannot create new variables lists, and `vl create` cannot modify existing lists.

The operator `+` can be used to take the union of two variable lists with duplicates removed.

```
. vl modify indepvars = myxvars + othervars
```

The operator `-` can be used to obtain the difference of two variable lists.

```
. vl modify indepvars = myxvars - othervars
```

Now `indepvars` contains the variables that are in `myxvars` excluding any that are in `othervars`. If there are variables in `othervars` that are not in `myxvars`, it is not an error. These variables are simply ignored.

The `+` and `-` operators can be used with *varlist*s as well.

```
. vl modify indepvars = myxvars + (w1 w2 w3)
```

`(`*varlist*`)` must be specified after `+` or `-`, never before.

To list the variables in a variable list, use `vl list`. To see a directory of variable lists that have been created, type `vl dir`. See [D] **vl list** for details on these two commands.

`vl label` attaches a label to the variable list that is displayed by `vl dir`.

```
. vl label indepvars "My brilliant choice of variables"
```

To delete `indepvars`, type

```
. vl drop indepvars
```

`vl drop` has other uses too; see [D] **vl drop**.

Using variable lists with other Stata commands

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 "$indepvars"
. summarize $indepvars
. regress y $indepvars
```

If you know Stata, you will have already figured out that variable lists are global macros. But the `vl` system is more than another way to create global macros. For instance, variable lists are saved with the dataset. Global macros are not. Both variable lists and other `vl` system information 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**.

vl substitute

Factor-variable operators can be used with variable lists. There are two ways to do this.

The first is to use factor-variable operators on the global macro form of the variable list like so:

```
. regress y i.($myfactors)##c.($mycontinuous)
```

Here myfactors is a user-defined variable list containing variables you want treated as factors. mycontinuous are variables you want treated as continuous. Specifying i.(...)##c.(...) means you want main effects of the factors plus interactions of all their levels with the continuous variables. Note that the parentheses, (), are required.

A second way to use factor-variable operators with variable lists is with the command vl substitute. For example,

```
. vl substitute myinteractions = i.myfactors##c.mycontinuous
. regress y $myinteractions
```

would produce the same result as the previous command. However, using vl substitute has the advantage that the variable lists it creates will be saved with your dataset, just like any other variable list.

See [U] **11.4.3 Factor variables**.

You can mix variable names with names of variable lists:

```
. vl substitute myinteractions = i.gender##c.(mycontinuous x100)
```

Here gender and x100 are variable names and mycontinuous is a variable list.

Be careful when mixing variable names and names of variable lists. vl substitute first assumes names are names of variable lists. Then it looks for variable names. For example, if you have both a variable named x and a variable list named x, and you specify

```
. vl substitute myinteractions = i.gender##c.(mycontinuous x)
```

then vl substitute will assume x is the variable list.

Using vl substitute to create a user-defined variable list is a one-shot deal. These variable lists cannot be modified after they are created. If you want to change them, first drop them,

```
. vl drop myinteractions
```

and then define them again:

```
. vl substitute myinteractions = i.myfactors##c.mycontinuous
```

For examples using vl create, vl modify, and vl substitute, see [D] **vl**.

Also see

[D] **vl** — Manage 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

[D] **vl set** — Set system-defined variable lists