vl create — Create and modify user-defined variable lists

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 \texttt{vlusername}, the expression \$\texttt{vlusername} can be used in Stata anywhere a \texttt{varlist} is allowed. Variable lists are actually \texttt{global macros}, and the \texttt{vl} commands are a convenient way to create and manipulate them. They are saved with the dataset. See \cite{D} \texttt{vl rebuild}.

For an introduction to the \texttt{vl} commands, see \cite{D} \texttt{vl}.

Quick start

Create a variable list

\begin{verbatim}
 vl create demographics = (age_cat gender)
\end{verbatim}

Add variables to a variable list

\begin{verbatim}
 vl modify demographics = demographics + (educ_cat income_cat)
\end{verbatim}

Add the variables in the variable list named \texttt{othervars} to the existing variable list called \texttt{myxvars}

\begin{verbatim}
 vl modify myxvars = myxvars + othervars
\end{verbatim}

Remove the variable \texttt{x8} from the variable list

\begin{verbatim}
 vl modify myxvars = myxvars - (x8)
\end{verbatim}

Apply factor-variable operator \texttt{i.} to all the variables in a variable list

\begin{verbatim}
 vl substitute idemographics = i.demographics
\end{verbatim}

Create interactions between the levels of the variables in the variable list \texttt{demographics} and the continuous variables in the variable list \texttt{vlcontinuous}

\begin{verbatim}
 vl substitute myinteractions = i.demographics#c.vlcontinuous
\end{verbatim}

Run a regression specifying the independent variables using variable lists

\begin{verbatim}
 regress y $idemographics $myxvars $myinteractions
\end{verbatim}
Syntax

Create user-defined variable lists

\[
\begin{align*}
\text{vl create } & \text{ vUsername} = (\text{varlist}) \\
\text{vl create } & \text{ vUsername} = \text{vName} + | - (\text{varlist}) \\
\text{vl create } & \text{ vUsername} = \text{vName}1 [ + | - \text{vName2}] \\
\end{align*}
\]

Modify user-defined variable lists

\[
\begin{align*}
\text{vl modify } & \text{ vUsername} = (\text{varlist}) \\
\text{vl modify } & \text{ vUsername} = \text{vName} + | - (\text{varlist}) \\
\text{vl modify } & \text{ vUsername} = \text{vName}1 [ + | - \text{vName2}] \\
\end{align*}
\]

Apply factor-variable operators to variable-list names

\[
\begin{align*}
\text{vl substitute } & \text{ vUsername} = i.\text{vName} \\
\text{vl substitute } & \text{ vUsername} = i.\text{vName1}/#i.\text{vName2} \\
\text{vl substitute } & \text{ vUsername} = i.\text{vName1}/##c.\text{vName2} \\
\end{align*}
\]

Label a user-defined variable-list name

\[
\text{vl label vUsername} ["label"]
\]

\textbf{vl create}

\textbf{vl create} creates a new variable list. It can be created from a list of variables:

\[
. \text{ vl create myxvars} = (x1-x100)
\]

In the above, note that the \textit{varlist} is enclosed in parentheses. \textit{varlists} must always be enclosed in parentheses.

When we are discussing the \textbf{vl} commands and say “variable list,” we mean a named variable list created by \textbf{vl create} or \textbf{vl set}. In this case, we created the variable list \textit{myxvars}. A traditional Stata list of variables, that is, a \textit{varlist}, we will call a \textit{varlist}.

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

\[
. \text{ vl create indepvars} = \text{myxvars}
\]
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 `varlists` 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 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