vl — Manage variable lists

Description Remarks and examples Also see

Description

`vl` stands for variable list. It is a suite of commands for creating and managing named variable lists. Lists are intended especially to be used as arguments to estimation commands.

In particular, the suite is designed to help divide variables into two groups: one group that will be treated as factor variables and another group that will be treated as continuous or interval variables.

`vl` creates two types of named variable lists: system-defined variable lists, created automatically by `vl set`, and user-defined variable lists, created by `vl create`. You will usually use `vl set` to create system-defined variable lists first, and then create your own variable lists from them with `vl create`.

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.

Variable lists are saved with the dataset.

Remarks and examples

Remarks are presented under the following headings:

Introduction

The vl commands are the following:

System only

vl set	initializes the system-defined variable lists based on the number of levels and other characteristics of a variable
vl move	moves variables from one system-defined variable list to another

User only

vl create	creates user-defined variable lists
vl modify	adds or removes variables from user-defined variable lists
vl label	adds a label to a user-defined variable list
vl substitute	creates a user-defined variable list using factor-variable operators

System or user

vl list	lists the contents of variable lists, either system or user
vl dir	displays the defined variable lists, either system or user
vl drop	deletes variable lists or removes variables from multiple variable lists
vl clear	deletes all variable lists
vl rebuild	restores variable lists

The first thing to note is that some vl commands only work with system-defined variable lists, some only work with user-defined variable lists, and others work with both.

vl set is typically used first. It initializes the system-defined variable lists. By default, it classifies all the numeric variables in your dataset. Or you can specify *varlist* and have it classify only those variables.

When we are discussing the vl commands and say "variable list", we mean a named variable list created by vl set or vl create. A traditional Stata list of variables, that is, *varlist*, we will call *varlist*. Variable lists contain *varlist*s.

vl create allows you to create your own variable lists, either starting with system-defined variable lists or with *varlist*s you specify. There is no need to run vl set and create system-defined variable lists. You can create your own from scratch. If you are familiar with the variables in your dataset and know which ones you want treated as factor variables and which as continuous variables, you may want to create only user-defined variable lists.

vl rebuild restores all the vl-generated variable lists after loading a dataset that previously had variable lists. Stata saves variable lists when you save your data, but when you use the saved data file, they are not automatically restored.

We will explain how to use vl with a series of examples.

vl set and system-defined variable lists

We will first show examples using Stata's automobile dataset because it only has a small number of variables and the output will not be too lengthy. We will do that even though you are unlikely to want to use vl with this small dataset. vl is intended for use with dozens or even thousands of variables.

```
. sysuse auto
(1978 automobile data)
```

Typing `vl set` without *varlist* classifies all the numeric variables in the data.

```
. vl set
```

	Macro's contents	
Macro	# Vars	Description
System		
$vlcategorical	2	categorical variables
$vlcontinuous	2	continuous variables
$vluncertain	7	perhaps continuous, perhaps categorical variables
$vlother	0	all missing or constant variables

Notes

1. Review contents of **vlcategorical** and **vlcontinuous** to ensure they are correct. Type **vl list vlcategorical** and type **vl list vlcontinuous**.
2. If there are any variables in **vluncertain**, you can reallocate them to **vlcategorical**, **vlcontinuous**, or **vlother**. Type **vl list vluncertain**.
3. Use **vl move** to move variables among classifications. For example, type **vl move (x50 x80) vlcontinuous** to move variables **x50** and **x80** to the continuous classification.
4. *vlnames* are global macros. Type the *vlname* without the leading dollar sign (**$**) when using **vl** commands. Example: **vlcategorical** *not* **$vlcategorical**. Type the dollar sign with other Stata commands to get a *varlist*.

By default, all numeric variables are put into one of four system-defined variable lists: `vlcategorical`, `vlcontinuous`, `vluncertain`, or `vlother`.

`vlcategorical` is intended for variables that are to be used as factor variables. `vlcontinuous` is intended for variables that are to be treated as continuous. `vluncertain` is intended for variables that may be categorical or may be continuous. `vlother` is a garbage classification intended for variables you want to ignore. `vl set` only puts constants and variables that are always missing into `vlother`, but you can move other variables there—more on that later.

Classification criteria for system-defined variable lists

Division into `vlcategorical`, `vlcontinuous`, or `vluncertain` is determined by several criteria.

First, if the variable contains any noninteger values, it goes in `vlcontinuous`.

Second, if the variable has negative values, it goes in `vlcontinuous` because factor variables in Stata must be nonnegative. If you have a variable that has values -1 and 1, you must recode it as 0 and 1 (or 1 and 2 or any other two distinct nonnegative integers) before you can use it as a factor variable.

Third, values of factor variables must be smaller than $2^{31} = 2{,}147{,}483{,}648$, so a variable with any values $\geq 2^{31}$ goes in `vlcontinuous`.

Fourth, constants, even when nonnegative integers, go in `vlother`.

For the remaining variables containing nonnegative integers, where they are placed is determined by two cutoffs, which can be specified by the options `categorical(#)` and `uncertain(#)`.

When the number of levels (distinct values), L, is

$$2 \leq L \leq \texttt{categorical(\#)}$$

the variable goes in `vlcategorical`. When

$$\texttt{categorical(\#)} < L \leq \texttt{uncertain(\#)}$$

the variable goes in `vluncertain`. When

$$L > \texttt{uncertain(\#)}$$

the variable goes in `vlcontinuous`.

The defaults are `categorical(10)` and `uncertain(100)`, which are admittedly arbitrary. They were chosen because they are easy-to-remember round numbers. In many cases, you will want to use different cutoffs. See the next section, where we reset `categorical(#)` and `uncertain(#)`.

Moving variables into another classification

`vl list` will show how each variable was classified and why.

```
. vl list, minimum maximum observations
```

Variable	Macro	Values	Levels	Min	Max	Obs
rep78	$vlcategorical	integers >=0	5	1	5	69
foreign	$vlcategorical	0 and 1	2	0	1	74
headroom	$vlcontinuous	noninteger		1.5	5	74
gear_ratio	$vlcontinuous	noninteger		2.19	3.89	74
price	$vluncertain	integers >=0	74	3291	15906	74
mpg	$vluncertain	integers >=0	21	12	41	74
trunk	$vluncertain	integers >=0	18	5	23	74
weight	$vluncertain	integers >=0	64	1760	4840	74
length	$vluncertain	integers >=0	47	142	233	74
turn	$vluncertain	integers >=0	18	31	51	74
displacement	$vluncertain	integers >=0	31	79	425	74

We specified options `minimum`, `maximum`, and `observations` to display the minimum and maximum values of each variable and the number of nonmissing observations.

`vl set` does not use the minimum and maximum to determine whether the variable goes in `vlcategorical`, `vlcontinuous`, or `vluncertain`. If the variable is a nonnegative integer, only the number of levels matters to `vl set`. A variable with levels 1,000,000 and 2,000,000 is classified the same as a variable with levels 0 and 1. The minimum and maximum can be displayed because you might want to use them to reclassify the variables.

In our example, we look at the number of levels and the minimum and maximum of the variables in `vluncertain`, and we decide we want to treat them all as continuous. We use `vl move` to move them into `vlcontinuous`.

```
. vl move vluncertain vlcontinuous
note: 7 variables specified and 7 variables moved.
```

Macro	# Added/Removed
$vlcategorical	0
$vlcontinuous	7
$vluncertain	-7
$vlother	0

When variables are moved into a different system-defined variable list, they are moved out of their current list.

Moving on, variable rep78, which gives the vehicle repair record, is worth some thought.

```
. tabulate rep78
```

Repair record 1978	Freq.	Percent	Cum.
1	2	2.90	2.90
2	8	11.59	14.49
3	30	43.48	57.97
4	18	26.09	84.06
5	11	15.94	100.00
Total	69	100.00	

rep78 could be considered categorical and used as a factor variable or could be considered as an interval variable and treated as a continuous variable.

Let's say we want to move it into vlcontinuous. To specify variable names directly, you specify them in parentheses. We move rep78.

```
. vl move (rep78) vlcontinuous
note: 1 variable specified and 1 variable moved.
```

Macro	# Added/Removed
$vlcategorical	-1
$vlcontinuous	1
$vluncertain	0
$vlother	0

vl create and user-defined variable lists

vl set and vl move are a first-pass classification of your variables. Next you will likely want to create specialized variable lists for use as independent variables for an estimation command.

You can create variable lists based on a specific set of variables. Use vl create and specify a *varlist* enclosed in parentheses, ().

```
. vl create power = (gear_ratio displacement weight)
note: $power initialized with 3 variables.
. vl create nonpower = (turn length rep78)
note: $nonpower initialized with 3 variables.
```

We want to model mpg. We created the variable list power, containing variables we think are related to power, and another variable list nonpower, containing variables that are not related to power but might be predictive of mpg.

After creating these variable lists, we decide the variable length belongs in power instead of nonpower. So we add it to power by using the vl modify command.

```
. vl modify power = power + (length)
note: 1 variable added to $power.
```

vl create and vl modify are like generate and replace in Stata. vl create creates new variable lists. vl modify modifies existing variable lists.

vl list

We can use vl list to see the variable lists to which the variable length belongs.

```
. vl list (length), user
```

Variable	Macro	Values	Levels
length	$nonpower	integers >=0	47
length	$power	integers >=0	47

We used vl list with *varlist* enclosed in parentheses. We specified option user to list only the user-defined variable lists.

If we do not want length in nonpower, we must explicitly move it out.

```
. vl modify nonpower = nonpower - (length)
note: 1 variable removed from $nonpower.
```

In this way, vl modify differs from vl move. vl move moves a variable out of its current system-defined variable list when the variable is moved into a new one. vl modify only modifies the specified variable list.

We can create new user-defined variable lists from existing variable lists, whether user or system defined.

```
. vl create xvars = power + nonpower
note: $xvars initialized with 6 variables.
```

Using (*) to specify the *varlist* for vl list gives a listing ordered by variable name first and then variable-list name.

```
. vl list (*)
```

Variable	Macro	Values	Levels
price	$vlcontinuous	integers >=0	74
price	not in vluser		74
mpg	$vlcontinuous	integers >=0	21
mpg	not in vluser		21
rep78	$vlcontinuous	integers >=0	5
rep78	$nonpower	integers >=0	5
rep78	$xvars	integers >=0	5
headroom	$vlcontinuous	noninteger	
headroom	not in vluser		
trunk	$vlcontinuous	integers >=0	18
trunk	not in vluser		18
weight	$vlcontinuous	integers >=0	64
weight	$power	integers >=0	64
weight	$xvars	integers >=0	64
length	$vlcontinuous	integers >=0	47
length	$power	integers >=0	47
length	$xvars	integers >=0	47
turn	$vlcontinuous	integers >=0	18
turn	$nonpower	integers >=0	18
turn	$xvars	integers >=0	18
displacement	$vlcontinuous	integers >=0	31
displacement	$power	integers >=0	31
displacement	$xvars	integers >=0	31
gear_ratio	$vlcontinuous	noninteger	
gear_ratio	$power	noninteger	
gear_ratio	$xvars	noninteger	
foreign	$vlcategorical	0 and 1	2
foreign	not in vluser		2

See [D] **vl list** for all the different ways it can list variable lists and variables.

vl substitute and factor-variable operators

Factor-variable operators can be used with variable lists using vl substitute. Here is an example:

```
. vl substitute indepvars = i.vlcategorical##c.xvars
```

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

To see what is in indepvars, we use the global macro syntax with a $ in front of its name and use display to view its contents.

```
. display "$indepvars"
i.foreign gear_ratio displacement weight length turn rep78 i.foreign#c.gear_ratio i
> .foreign#c.displacement i.foreign#c.weight i.foreign#c.length i.foreign#c.turn i.
> foreign#c.rep78
```

To use variable lists with other Stata commands, we do the same thing. We treat the list name like the global macro it is and put a $ in front of it.

```
. regress mpg $indepvars
```

Source	SS	df	MS	Number of obs	=	69
				F(13, 55)	=	20.86
Model	1945.54632	13	149.657409	Prob > F	=	0.0000
Residual	394.656577	55	7.17557413	R-squared	=	0.8314
				Adj R-squared	=	0.7915
Total	2340.2029	68	34.4147485	Root MSE	=	2.6787

mpg	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
foreign						
Foreign	-32.65519	24.36955	-1.34	0.186	-81.49286	16.18248
gear_ratio	-.0847818	1.959716	-0.04	0.966	-4.012141	3.842577
(*output omitted*)						
foreign#c.rep78						
Foreign	4.480624	1.10794	4.04	0.000	2.260263	6.700985
_cons	50.52293	8.553643	5.91	0.000	33.38104	67.66481

Just like all the other user-defined variable lists, variable lists created by `vl substitute` are saved with the data. See [D] **vl rebuild**.

Exploring data with vl set

Consider a bigger dataset. It is fictitious data, designed to mimic real questionnaire data.

```
. use https://www.stata-press.com/data/r19/questionnaire, clear
(Fictitious Questionnaire Data)
```

`vl` can be used to explore your data. It is a bit like `codebook` except that `codebook` provides more information. `vl set`, however, is much faster. `vl set` is even speedy with datasets containing millions of observations and thousands of variables.

We run `vl set` with the `list()` option, which is equivalent to using the `vl list` command. We also specify the option `nonotes` to suppress the notes at the end of the table.

```
. vl set, list(min max obs) nonotes
```

Variable	Macro	Values	Levels	Min	Max	Obs
gender	$vlcategorical	0 and 1	2	0	1	1,058
age	$vluncertain	integers >=0	47	2	64	1,058
q1	$vluncertain	integers >=0	40	1	47	1,048
q2	$vlcategorical	integers >=0	3	1	3	1,046
q3	$vlcategorical	0 and 1	2	0	1	1,049
q4	$vlcategorical	0 and 1	2	0	1	1,042
q5	$vlcategorical	0 and 1	2	0	1	1,048
q6	$vlcategorical	integers >=0	3	1	3	1,046
q7	$vlcategorical	0 and 1	2	0	1	1,047
q8	$vlcategorical	0 and 1	2	0	1	1,046
q9	$vlcategorical	0 and 1	2	0	1	1,051
q10	$vlcategorical	0 and 1	2	0	1	1,047
q11	$vlcategorical	0 and 1	2	0	1	1,042
q12	$vlcategorical	integers >=0	5	1	5	1,052
q13	$vlcategorical	0 and 1	2	0	1	1,045
q14	$vlcategorical	0 and 1	2	0	1	1,047
q15	$vluncertain	integers >=0	36	0	37	1,040
q16	$vlcategorical	integers >=0	3	1	3	1,046
q17	$vlcategorical	0 and 1	2	0	1	1,054
q18	$vlcategorical	integers >=0	7	1	7	1,048
q19	$vlcategorical	0 and 1	2	0	1	1,043
q20	$vluncertain	integers >=0	30	1	30	1,048
check1	$vlother	constant	1	1	1	1,058
q21	$vluncertain	integers >=0	39	2	40	1,048
q22	$vluncertain	integers >=0	32	3	36	1,050
q23	$vlcategorical	integers >=0	10	1	10	1,050
q24	$vlcontinuous	negative		-1	1	1,050
(*output omitted*)						
q45	$vlcontinuous	noninteger		8.7	69.9	1,045
(*output omitted*)						
q60	$vlother	all missing		.	.	0
(*output omitted*)						
q76	$vlcontinuous	integers >=0	>100	84	287	1,051
(*output omitted*)						
q161	$vlcategorical	0 and 1	2	0	1	1,047
check8	$vlother	constant	1	1	1	1,058

Summary

	Macro's contents	
Macro	# Vars	Description
System		
$vlcategorical	138	categorical variables
$vlcontinuous	3	continuous variables
$vluncertain	21	perhaps continuous, perhaps categorical variables
$vlother	9	all missing or constant variables

From the summary table, we see that most of the variables were put in vlcategorical. The default cutoff for the number of levels for vlcategorical is 10, so these 138 variables all have 10 levels or less.

Three variables were put in vlcontinuous. One, q24, has negative values. Its values are actually only −1 and 1. So it is integer with only two levels, yet it is classified as continuous. Factor variables must be nonnegative, so any variable with negative values is put into vlcontinuous. We need to recode q24 as 0/1 (or 1/2, etc.) to use it as a factor variable.

The variable q45 was put in vlcontinuous because it contains noninteger values.

The variable q76 was put in vlcontinuous because, although it is a nonnegative integer, it has over 100 levels. The default cutoff is 100 for determining whether variables are put in vlcontinuous or vluncertain. Note that the output does not say exactly how many levels, just that the number is greater than 100.

The variable list vluncertain contains 21 variables. These are nonnegative integers with the number of levels > 10 and ≤ 100.

The variable list vlother contains nine variables. These variables are either constants or all missing—variables not suitable for any statistical analyses.

Changing the cutoffs for classification

The default classification produced by vl set was not very useful in this case. vl set put too many variables in vlcategorical, and it put too many in vluncertain. Most of the variables in vluncertain are integer-valued scales, and we want those in vlcontinuous.

We will fix this. We run vl set again to re-create the classifications, and this time, we specify categorical(4) and uncertain(19), meaning that variables in vlcategorical can have up to 4 levels and variables with 5 to 19 levels are placed in vluncertain. We also specify the option dummy to tell vl set to smarten up and put all the 0/1 variables in their own classification. Finally, we specify option clear to clear the old classifications. See [D] **vl set**.

```
. vl set, categorical(4) uncertain(19) dummy clear nonotes
```

	Macro's contents	
Macro	# Vars	Description
System		
$vldummy	99	0/1 variables
$vlcategorical	16	categorical variables
$vlcontinuous	21	continuous variables
$vluncertain	26	perhaps continuous, perhaps categorical variables
$vlother	9	all missing or constant variables

We did not really need to create the vldummy variable list. Had we wanted to treat the dummy variables as factor variables, we could have let vl set put them in vlcategorical, as it would by default. Note that vldummy contains only 0/1 variables. A 1/2 variable is still put in vlcategorical.

Moving variables from one classification to another

At this point, we are happy with the variables that are in `vlcategorical` and `vlcontinuous`. We are unhappy with having variables in `vluncertain`, and we have 26 of them! Those variables have between 5 and 19 levels. Let's list the variables and categorize them by hand.

```
. vl list vluncertain
```

Variable	Macro	Values	Levels
q12	$vluncertain	integers >=0	5
q18	$vluncertain	integers >=0	7
q23	$vluncertain	integers >=0	10
q27	$vluncertain	integers >=0	8
q28	$vluncertain	integers >=0	15
q35	$vluncertain	integers >=0	7
q39	$vluncertain	integers >=0	5
q54	$vluncertain	integers >=0	10
q63	$vluncertain	integers >=0	7
q66	$vluncertain	integers >=0	5
q80	$vluncertain	integers >=0	5
q81	$vluncertain	integers >=0	5
q92	$vluncertain	integers >=0	5
q93	$vluncertain	integers >=0	7
q99	$vluncertain	integers >=0	5
q103	$vluncertain	integers >=0	7
q111	$vluncertain	integers >=0	7
q112	$vluncertain	integers >=0	7
q119	$vluncertain	integers >=0	8
q120	$vluncertain	integers >=0	7
q124	$vluncertain	integers >=0	14
q127	$vluncertain	integers >=0	5
q132	$vluncertain	integers >=0	7
q135	$vluncertain	integers >=0	10
q141	$vluncertain	integers >=0	12
q157	$vluncertain	integers >=0	7

Many of the variables have seven levels. Let's tabulate one of them.

```
. tabulate q18
```

Question 18	Freq.	Percent	Cum.
very strongly disagree	136	12.98	12.98
strongly disagree	148	14.12	27.10
disagree	144	13.74	40.84
neither agree nor disagree	146	13.93	54.77
agree	173	16.51	71.28
strongly agree	146	13.93	85.21
very strongly agree	155	14.79	100.00
Total	1,048	100.00	

This variable contains a Likert scale and, because of that, we want to treat the variable as continuous. In fact, all the variables with seven levels are Likert scales. We move them all into `vlcontinuous`.

```
. vl move (q18 q35 q63 q93 q103 q111 q112 q120 q132 q157) vlcontinuous
note: 10 variables specified and 10 variables moved.
```

Macro	# Added/Removed
$vldummy	0
$vlcategorical	0
$vlcontinuous	10
$vluncertain	-10
$vlother	0

Now we can list the remaining `vluncertain` variables.

```
. vl list vluncertain
```

Variable	Macro	Values	Levels
q12	$vluncertain	integers >=0	5
q23	$vluncertain	integers >=0	10
q27	$vluncertain	integers >=0	8
q28	$vluncertain	integers >=0	15
q39	$vluncertain	integers >=0	5
q54	$vluncertain	integers >=0	10
q66	$vluncertain	integers >=0	5
q80	$vluncertain	integers >=0	5
q81	$vluncertain	integers >=0	5
q92	$vluncertain	integers >=0	5
q99	$vluncertain	integers >=0	5
q119	$vluncertain	integers >=0	8
q124	$vluncertain	integers >=0	14
q127	$vluncertain	integers >=0	5
q135	$vluncertain	integers >=0	10
q141	$vluncertain	integers >=0	12

You can decide for yourself where they go and use `vl move` to place them.

Dropping variables and rebuilding variable lists

We have variables in `vlother`.

```
. vl list vlother
```

Variable	Macro	Values	Levels
check1	$vlother	constant	1
check2	$vlother	constant	1
q60	$vlother	all missing	
check3	$vlother	constant	1
check4	$vlother	constant	1
check5	$vlother	constant	1
check6	$vlother	constant	1
check7	$vlother	constant	1
check8	$vlother	constant	1

We could use vl drop to remove them from the vl system classification. But we do not want them in our dataset, so we drop them.

```
. drop $vlother
```

Now if we run

```
. vl list
variable check1 not found
    Run vl rebuild to rebuild vl macros.
r(111);
```

we get an error! vl keeps track of all the variables put into variable lists, and whenever a vl command is run, it first checks that everything is okay. It discovered missing variables and needs confirmation that this is intentional. If it is, we vl rebuild the system.

```
. vl rebuild
Rebuilding vl macros ...
```

	Macro's contents	
Macro	# Vars	Description
System		
$vldummy	99	0/1 variables
$vlcategorical	16	categorical variables
$vlcontinuous	31	continuous variables
$vluncertain	16	perhaps continuous, perhaps categorical variables
$vlother	0	all missing or constant variables

Changing variables and updating variable lists

If you change the values of a variable, you need to vl set the variable again to update its statistics. You can update its statistics leaving its classification unchanged or tell vl set to redo the classification as well.

We noticed that age had a suspiciously low minimum.

```
. vl list (age), min max obs
```

Variable	Macro	Values	Levels	Min	Max	Obs
age	$vlcontinuous	integers >=0	>19	2	64	1,058

We do not believe a two-year-old took our questionnaire. Let's find the ID of this subject.

```
. list id age if age == 2
```

	id	age
543.	05034558	2

We check our original data source and discover that the subject was 20 years old. We correct the value of age.

```
. replace age = 20 if id == "05034558" & age == 2
(1 real change made)
```

Now the minimum of age stored by vl is wrong. We could ignore it, or we could fix it by using the update option of vl set. The option update does not change the classification of a variable; it only updates the stored statistics.

```
. vl set age, update list(min max obs) nonotes
```

Variable	Macro	Values	Levels	Min	Max	Obs
age	$vlcontinuous	integers >=0	47	18	64	1,058

Summary

	Macro's contents	
Macro	# Vars	Description
System		
$vldummy	99	0/1 variables
$vlcategorical	16	categorical variables
$vlcontinuous	31	continuous variables
$vluncertain	16	perhaps continuous, perhaps categorical variables
$vlother	0	all missing or constant variables

If we wanted to redo the classification of age and update its statistics, we would type

```
. vl set age, redo
```

(*output omitted*)

Saving and using datasets with variable lists

When we save our data, the vl system is saved.

```
. save quest_with_vl
file quest_with_vl.dta saved
```

However, when we use our data, the vl system is not automatically restored.

```
. use quest_with_vl
(Fictitious Questionnaire Data)
```

Type vl rebuild to bring the system back to life.

```
. vl rebuild
Rebuilding vl macros ...
```

	Macro's contents	
Macro	# Vars	Description
System		
$vldummy	99	0/1 variables
$vlcategorical	16	categorical variables
$vlcontinuous	31	continuous variables
$vluncertain	16	perhaps continuous, perhaps categorical variables
$vlother	0	all missing or constant variables

See [D] **vl rebuild** for other instances when you need to run vl rebuild.

User-defined variable lists and factor-variable operators

We continue with our previous example using fictitious questionnaire data.

The system-defined variable lists are good for organizing variables. We would not use them, however, to specify *varlist*s for estimation commands if for no other reason than we do not want to use all the variables in the dataset. For this purpose, we need to create user-defined variable lists.

Here is a variable list containing demographic variables we want to use for model fitting.

```
. vl create demographics = (gender q3 q4 q5)
note: $demographics initialized with 4 variables.
```

We are going to create two more variable lists: `factors`, containing variables we want to treat as factor variables, and `control_scales`, containing variables we want to treat as continuous.

```
. vl create factors = vldummy + vlcategorical
note: $factors initialized with 115 variables.
. vl create control_scales = (q15 q20 q21 q22)
note: $control_scales initialized with 4 variables.
```

This is the real power of `vl`. We created `factors` from `vldummy` plus `vlcategorical`. But `factors` contains variables in `demographics`, and we want to handle the `demographics` variables differently. So we remove them from `factors`. We also remove some other variables we do not want in our model.

```
. vl modify factors = factors - demographics
note: 4 variables removed from $factors.
. vl modify factors = factors - (q155 q156 q158)
note: 3 variables removed from $factors.
```

We are going to fit a `poregress` model, and our variables of interest (ones for which we want to do inference) are the categorical variables `q7`, `q13`, and `q16`, and the continuous variable `q35`.

We create a variable list with the categorical ones, and remove them from `factors`.

```
. vl create fvofinterest = (q7 q13 q16)
note: $fvofinterest initialized with 3 variables.
. vl modify factors = factors - fvofinterest
note: 3 variables removed from $factors.
```

Now we use `vl substitute` to create a variable list that contains factor variables.

```
. vl substitute interest = i.fvofinterest q35
```

Notice that we tucked the continuous variable `q35` in at the end. `vl substitute` lets you specify variable lists and variables by using factor-variable operators—or not—in a natural way.

If you want to see the contents of a variable list created using `vl substitute`, you can `display` it:

```
. display "$interest"
i.q7 i.q13 i.q16 q35
```

The one thing to remember about `vl substitute` is that it is a one-shot deal. Once the variable list is created, you cannot modify it. If you want to change it, you must delete it using `vl drop` and then re-create it using `vl substitute`.

We are going to go nuts and create a variable list consisting of bushels of interactions.

```
. vl substitute controlvars = i.demographics i.factors##c.control_scales
```

The interest variable list contains our variables of interest for poregress. The controlvars variable list contains control variables for the model.

```
. poregress q1 $interest, controls($controlvars)

Estimating lasso for q1 using plugin
Estimating lasso for 1bn.q7 using plugin
Estimating lasso for 1bn.q13 using plugin
Estimating lasso for 2bn.q16 using plugin
Estimating lasso for 3bn.q16 using plugin
Estimating lasso for q35 using plugin

Partialing-out linear model          Number of obs                =        339
                                     Number of controls           =      1,137
                                     Number of selected controls  =         12
                                     Wald chi2(5)                 =      12.89
                                     Prob > chi2                  =     0.0244
```

q1	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
q7						
yes	-1.333003	.7441531	-1.79	0.073	-2.791516	.1255107
q13						
yes	.4321797	.684376	0.63	0.528	-.9091725	1.773532
q16						
2	.6905278	.8355682	0.83	0.409	-.9471559	2.328211
3	2.497944	.8572828	2.91	0.004	.8177008	4.178188
q35	-.1238627	.1833827	-0.68	0.499	-.4832861	.2355608

```
Note: Chi-squared test is a Wald test of the coefficients of the variables
      of interest jointly equal to zero. Lassos select controls for model
      estimation. Type lassoinfo to see number of selected variables in each
      lasso.
```

Using vl, we can specify huge *varlist*s in a succinct notation. If we were to list the expanded estimation command, it would take half a page!

Updating variable lists created by vl substitute

What is especially convenient about variable lists is how easy they are to modify. Suppose we decide we do not want q13 in our model. We cannot explicitly change interest because it was created by vl substitute, but we can change fvofinterest.

```
. vl modify fvofinterest = fvofinterest - (q13)
note: 1 variable removed from $fvofinterest.
```

We now update `interest` using `vl rebuild`.

```
. vl rebuild
Rebuilding vl macros ...
```

	Macro's contents	
Macro	# Vars	Description
System		
$vldummy	99	0/1 variables
$vlcategorical	16	categorical variables
$vlcontinuous	31	continuous variables
$vluncertain	16	perhaps continuous, perhaps categorical variables
$vlother	0	all missing or constant variables
User		
$demographics	4	variables
$factors	105	variables
$control_scales	4	variables
$fvofinterest	2	variables
$interest		factor-variable list
$controlvars		factor-variable list

And we see that `q13` is gone from our variable list.

```
. display "$interest"
i.q7 i.q16 q35
```

Also see

[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

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