Title

stata.com

cluster generate — Generate grouping variables from a cluster analysis

Description

`cluster generate` creates summary or grouping variables from a hierarchical cluster analysis; the result depends on the function. A single variable may be created containing a group number based on the requested number of groups or cutting the dendrogram at a specified (dis)similarity value. A set of new variables may be created if a range of group sizes is specified.

Users can add more `cluster generate` functions; see [MV] **cluster programming subroutines**.

Quick start

Generate grouping variable `g5` with 5 groups from the most recent cluster analysis

```
cluster generate g5 = groups(5)
```

Same as above, 4 grouping variables (`g4`, `g5`, `g6`, and `g7`) with 4, 5, 6, and 7 groups

```
cluster generate g = groups(4/7)
```

Same as above, but use the cluster analysis named `myclus`

```
cluster generate g = groups(4/7), name(myclus)
```

Generate grouping variable `mygroups` from the most recent cluster analysis by cutting the dendrogram at dissimilarity value 38

```
cluster generate mygroups = cut(38)
```

Menu

Statistics > Multivariate analysis > Cluster analysis > Postclustering > Summary variables from cluster analysis

Syntax

Generate grouping variables for specified numbers of clusters

```
cluster generate { newvar | stub } = groups(numlist) [ , options ]
```

Generate grouping variable by cutting the dendrogram

```
cluster generate newvar = cut(#) [ , name(clname) ]
```

options	Description
name(*clname*)	name of cluster analysis to use in producing new variables
ties(error)	produce error message for ties; default
ties(skip)	ignore requests that result in ties
ties(fewer)	produce results for largest number of groups smaller than your request
ties(more)	produce results for smallest number of groups larger than your request

Options

name(*clname*) specifies the name of the cluster analysis to use in producing the new variables. The default is the name of the cluster analysis last performed, which can be reset by using the cluster use command; see [MV] **cluster utility**.

ties(error | skip | fewer | more) indicates what to do with the groups() function for ties. A hierarchical cluster analysis has ties when multiple groups are generated at a particular (dis)similarity value. For example, you might have the case where you can uniquely create two, three, and four groups, but the next possible grouping produces eight groups because of ties.

ties(error), the default, produces an error message and does not generate the requested variables.

ties(skip) specifies that the offending requests be ignored. No error message is produced, and only the requests that produce unique groupings will be honored. With multiple values specified in the groups() function, ties(skip) allows the processing of those that produce unique groupings and ignores the rest.

ties(fewer) produces the results for the largest number of groups less than or equal to your request. In the example above with groups(6) and using ties(fewer), you would get the same result that you would by using groups(4).

ties(more) produces the results for the smallest number of groups greater than or equal to your request. In the example above with groups(6) and using ties(more), you would get the same result that you would by using groups(8).

Remarks and examples

stata.com

`cluster generate` provides two built-in functions: `groups()` and `cut()`. Examples of how to use the `groups()` function can be found in [MV] **cluster dendrogram**, [MV] **cluster linkage**, and [MV] **cluster stop**. More examples of the `groups()` and `cut()` functions of `cluster generate` are provided here.

The `groups(`*numlist*`)` function generates grouping variables, giving the grouping for the specified numbers of clusters from a hierarchical cluster analysis. If one number is given, *newvar* is produced with group numbers going from 1 to the number of clusters requested. If more than one number is specified, a new variable is generated for each number by using the provided *stub* name appended with the number.

The `cut(#)` function generates a grouping variable corresponding to cutting the dendrogram (see [MV] **cluster dendrogram**) of a hierarchical cluster analysis at the specified (dis)similarity value.

You may find it easier to understand these functions by looking at a dendrogram from a hierarchical cluster analysis. The `cluster dendrogram` command produces dendrograms (cluster trees) from a hierarchical cluster analysis; see [MV] **cluster dendrogram**.

▷ Example 1

Example 1 of [MV] **cluster linkage** examines a dataset with 50 observations with four variables. Here we use complete-linkage clustering and use the `groups()` function of `cluster generate` to produce a grouping variable, splitting the data into two groups.

```
. use https://www.stata-press.com/data/r18/labtech
. cluster completelinkage x1 x2 x3 x4, name(L2clnk)
. cluster dendrogram L2clnk, xlabel(, angle(90) labsize(*.75))
  (graph omitted)
. cluster generate g2 = group(2), name(L2clnk)
. codebook g2
```

```
-------------------------------------------------------------------------------
g2                                                              (unlabeled)
-------------------------------------------------------------------------------

                  Type: Numeric (byte)

                 Range: [1,2]                         Units: 1
         Unique values: 2                         Missing .: 0/50

            Tabulation: Freq.  Value
                        26  1
                        24  2
```

```
. by g2, sort: summarize x*
```

```
-> g2 = 1
```

Variable	Obs	Mean	Std. dev.	Min	Max
x1	26	91.5	37.29432	17.4	143
x2	26	74.58077	41.19319	4.8	142.1
x3	26	101.0077	36.95704	16.3	147.9
x4	26	71.77308	43.04107	6.6	146.1

```
-> g2 = 2
```

Variable	Obs	Mean	Std. dev.	Min	Max
x1	24	18.8	23.21742	0	77
x2	24	30.05833	37.66979	0	143.6
x3	24	18.54583	21.68215	.2	69.7
x4	24	41.89167	43.62025	.1	130.9

The `group()` function of `cluster generate` created a grouping variable named `g2`, with ones indicating the 26 observations that belong to the left main branch of the dendrogram and twos indicating the 24 observations that belong to the right main branch of the dendrogram. The summary of the `x` variables used in the cluster analysis for each group shows that the second group is characterized by lower values.

We could have obtained the same grouping variable by using the `cut()` function of `cluster generate`.

```
. cluster generate g2cut = cut(200)
. table g2 g2cut, nototals
```

	g2cut	
	1	2
g2		
1	26	
2		24

We did not need to specify the `name()` option because this was the latest cluster analysis performed, which is the default. The `table` output shows that we obtained the same result with `cut(200)` as with `group(2)` for this example.

How many groups are produced if we cut the tree at the value 105.2?

```
. cluster generate z = cut(105.2)
. codebook z, tabulate(20)
```

```
z                                                                (unlabeled)
-------------------------------------------------------------------------------

                  Type: Numeric (byte)
                 Range: [1,11]                        Units: 1
         Unique values: 11                            Missing .: 0/50

            Tabulation: Freq.  Value
                          3  1
                          3  2
                          5  3
                          1  4
                          2  5
                          2  6
                         10  7
                         10  8
                          8  9
                          5  10
                          1  11
```

The `codebook` command (see [D] **codebook**) shows that the result of cutting the dendrogram at the value 105.2 produced 11 groups ranging in size from 1 to 10 observations.

The `group()` function of `cluster generate` may be used to create multiple grouping variables with one call. Here we create the grouping variables for groups of size 3–12:

```
. cluster generate gp = gr(3/12)
. summarize gp*
```

Variable	Obs	Mean	Std. dev.	Min	Max
gp3	50	2.26	.8033095	1	3
gp4	50	3.14	1.030356	1	4
gp5	50	3.82	1.438395	1	5
gp6	50	3.84	1.461897	1	6
gp7	50	3.96	1.603058	1	7
gp8	50	4.24	1.911939	1	8
gp9	50	5.18	2.027263	1	9
gp10	50	5.94	2.385415	1	10
gp11	50	6.66	2.781939	1	11
gp12	50	7.24	3.197959	1	12

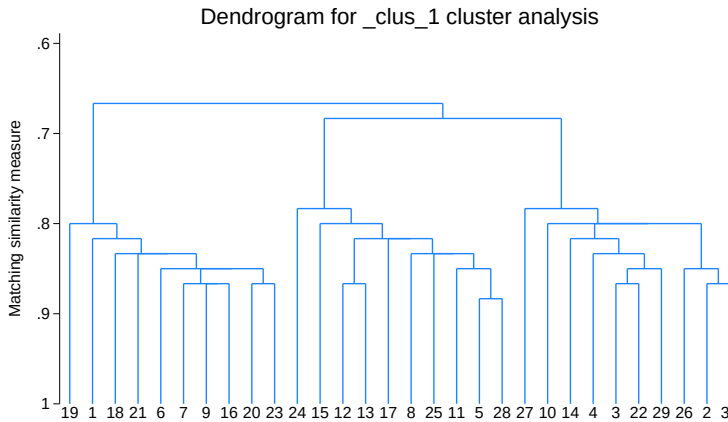
Here we used abbreviations for `generate` and `group()`. The `group()` function takes a numlist; see **[U] 11.1.8 numlist**. We specified `3/12`, indicating the numbers 3–12. `gp`, the stub name we provide, is appended with the number as the variable name for each group variable produced.

◁

▷ Example 2

Example 2 of [MV] **cluster linkage** shows the following dendrogram from the single-linkage clustering of 30 observations on 60 variables. In that example, we used the `group()` function of `cluster generate` to produce a grouping variable for three groups. What happens when we try to obtain four groups from this clustering?

```
. use https://www.stata-press.com/data/r18/homework, clear
. cluster singlelinkage a1-a60, measure(matching)
cluster name: _clus_1
. cluster tree
```




```
. cluster generate g4 = group(4)
cannot create 4 groups because of ties
r(198);
```

Stata complains that it cannot create four groups from this cluster analysis.

The `ties()` option gives us control over this situation. We just need to decide whether we want more groups or fewer groups than we asked for when faced with ties. We demonstrate both ways.

```
. cluster generate more4 = gr(4), ties(more)
. cluster generate less4 = gr(4), ties(fewer)
. summarize more4 less4
```

Variable	Obs	Mean	Std. dev.	Min	Max
more4	30	2.933333	1.638614	1	5
less4	30	2	.8304548	1	3

For this cluster analysis, `ties(more)` with `group(4)` produces five groups, whereas `ties(fewer)` with `group(4)` produces three groups.

The `ties(skip)` option is convenient when we want to produce a range of grouping variables.

```
. cluster generate group = gr(4/20), ties(skip)
. summarize group*
```

Variable	Obs	Mean	Std. dev.	Min	Max
group5	30	2.933333	1.638614	1	5
group9	30	4.866667	2.622625	1	9
group13	30	7.066667	3.92106	1	13
group18	30	9.933333	5.419844	1	18

With this cluster analysis, the only unique groupings available are 5, 9, 13, and 18 within the range 4–20.

◁

Also see

[MV] **cluster** — Introduction to cluster-analysis commands

[MV] **clustermat** — Introduction to clustermat commands

[D] **egen** — Extensions to generate

[D] **generate** — Create or change contents of variable