Title

cluster kmeans and kmedians — Kmeans and kmedians cluster analysis

Syntax
Menu
Description
Options
Remarks and examples
Methods and formulas
Reference
Also see

Syntax

Kmeans cluster analysis

```
cluster kmeans [varlist] [if] [in] , k(#) [options]
```

Kmedians cluster analysis

```
cluster kmedians [varlist] [if] [in] , k(#) [options]
```

option	Description
Main	
*k(#)	perform cluster analysis resulting in # groups
measure(*measure*)	similarity or dissimilarity measure; default is L2 (Euclidean)
name(*clname*)	name of resulting cluster analysis
Options	
start(*start_option*)	obtain *k* initial group centers by using *start_option*; see *Options* for details
keepcenters	append the *k* final group means or medians to the data
Advanced	
generate(*groupvar*)	name of grouping variable
iterate(#)	maximum number of iterations; default is iterate(10000)

*k(#) is required.

Menu

cluster kmeans

Statistics > Multivariate analysis > Cluster analysis > Cluster data > Kmeans

cluster kmedians

Statistics > Multivariate analysis > Cluster analysis > Cluster data > Kmedians

Description

cluster kmeans and cluster kmedians perform kmeans and kmedians partition cluster analysis, respectively. See [MV] **cluster** for a general discussion of cluster analysis and a description of the other cluster commands.

Options

Main

`k(`*#*`)` is required and indicates that # groups are to be formed by the cluster analysis.

`measure(`*measure*`)` specifies the similarity or dissimilarity measure. The default is `measure(L2)`, Euclidean distance. This option is not case sensitive. See [MV] ***measure_option*** for detailed descriptions of the supported measures.

`name(`*clname*`)` specifies the name to attach to the resulting cluster analysis. If `name()` is not specified, Stata finds an available cluster name, displays it for your reference, and attaches the name to your cluster analysis.

Options

`start(`*start_option*`)` indicates how the k initial group centers are to be obtained. The available *start_options* are

`krandom`[`(`*seed#*`)`], the default, specifies that k unique observations be chosen at random, from among those to be clustered, as starting centers for the k groups. Optionally, a random-number seed may be specified to cause the command `set seed` *seed#* (see [R] **set seed**) to be applied before the k random observations are chosen.

`firstk`[`, exclude`] specifies that the first k observations from among those to be clustered be used as the starting centers for the k groups. With the `exclude` option, these first k observations are not included among the observations to be clustered.

`lastk`[`, exclude`] specifies that the last k observations from among those to be clustered be used as the starting centers for the k groups. With the `exclude` option, these last k observations are not included among the observations to be clustered.

`random`[`(`*seed#*`)`] specifies that k random initial group centers be generated. The values are randomly chosen from a uniform distribution over the range of the data. Optionally, a random-number seed may be specified to cause the command `set seed` *seed#* (see [R] **set seed**) to be applied before the k group centers are generated.

`prandom`[`(`*seed#*`)`] specifies that k partitions be formed randomly among the observations to be clustered. The group means or medians from the k groups defined by this partitioning are to be used as the starting group centers. Optionally, a random-number seed may be specified to cause the command `set seed` *seed#* (see [R] **set seed**) to be applied before the k partitions are chosen.

`everykth` specifies that k partitions be formed by assigning observations 1, $1+k$, $1+2k$, ... to the first group; assigning observations 2, $2+k$, $2+2k$, ... to the second group; and so on, to form k groups. The group means or medians from these k groups are to be used as the starting group centers.

`segments` specifies that k nearly equal partitions be formed from the data. Approximately the first N/k observations are assigned to the first group, the second N/k observations are assigned to the second group, and so on. The group means or medians from these k groups are to be used as the starting group centers.

`group(`*varname*`)` provides an initial grouping variable, *varname*, that defines k groups among the observations to be clustered. The group means or medians from these k groups are to be used as the starting group centers.

`keepcenters` specifies that the group means or medians from the k groups that are produced be appended to the data.

Advanced

generate(*groupvar*) provides the name of the grouping variable to be created by cluster kmeans or cluster kmedians. By default, this will be the name specified in name().

iterate(#) specifies the maximum number of iterations to allow in the kmeans or kmedians clustering algorithm. The default is iterate(10000).

Remarks and examples

stata.com

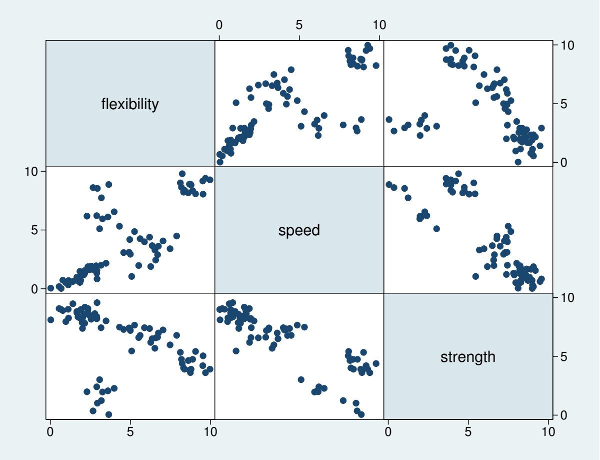
Two examples are presented, one using cluster kmeans with continuous data and the other using cluster kmeans and cluster kmedians with binary data. Both commands work similarly with the different types of data.

▷ Example 1

You have measured the flexibility, speed, and strength of the 80 students in your physical education class. You want to split the class into four groups, based on their physical attributes, so that they can receive the mix of flexibility, strength, and speed training that will best help them improve.

Here is a summary of the data and a matrix graph showing the data:

```
. use http://www.stata-press.com/data/r13/physed
. summarize flex speed strength
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
 flexibility |        80    4.402625    2.788541        .03       9.97
       speed |        80    3.875875    3.121665        .03       9.79
    strength |        80    6.439875    2.449293        .05       9.57
. graph matrix flex speed strength
```



As you expected, based on what you saw the first day of class, the data indicate a wide range of levels of performance for the students. The graph seems to indicate that there are some distinct groups, which leads you to believe that your plan will work well.

You decide to perform a cluster analysis to create four groups, one for each of your class assistants. You have had good experience with kmeans clustering in the past and generally like the behavior of the absolute-value distance.

You do not really care what starting values are used in the cluster analysis, but you do want to be able to reproduce the same results if you ever decide to rerun your analysis. You decide to use the `krandom()` option to pick k of the observations at random as the initial group centers. You supply a random-number seed for reproducibility. You also add the `keepcenters` option so that the means of the four groups will be added to the bottom of your dataset.

```
. cluster k flex speed strength, k(4) name(g4abs) s(kr(385617)) mea(abs) keepcen

. cluster list g4abs
g4abs  (type: partition,  method: kmeans,  dissimilarity: L1)
      vars: g4abs (group variable)
     other: cmd: cluster kmeans flex speed strength, k(4) name(g4abs)
                  s(kr(385617)) mea(abs) keepcen
            varlist: flexibility speed strength
            k: 4
            start: krandom(385617)
            range: 0 .

. table g4abs

----------------------
    g4abs |      Freq.
----------+-----------
        1 |         15
        2 |         20
        3 |         35
        4 |         10
----------------------

. list flex speed strength in 81/L, abbrev(12)

     +--------------------------------------+
     | flexibility      speed    strength   |
     |--------------------------------------|
 81. |       8.852   8.743333       4.358   |
 82. |      5.9465     3.4485      6.8325   |
 83. |    1.969429   1.144857    8.478857   |
 84. |       3.157      6.988       1.641   |
     +--------------------------------------+

. drop in 81/L
(4 observations deleted)
```

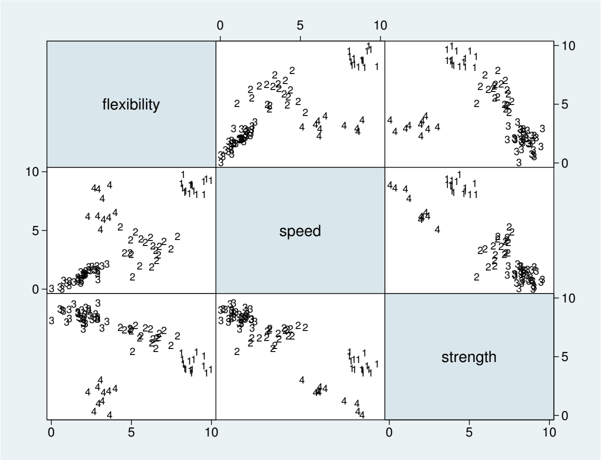
```
. tabstat flex speed strength, by(g4abs) stat(min mean max)
Summary statistics: min, mean, max
  by categories of: g4abs

   g4abs |  flexib~y     speed  strength
---------+------------------------------
       1 |      8.12      8.05      3.61
         |     8.852  8.743333     4.358
         |      9.97      9.79      5.42
---------+------------------------------
       2 |      4.32      1.05      5.46
         |    5.9465    3.4485    6.8325
         |      7.89      5.32      7.66
---------+------------------------------
       3 |       .03       .03      7.38
         |  1.969429  1.144857  8.478857
         |      3.48      2.17      9.57
---------+------------------------------
       4 |      2.29      5.11       .05
         |     3.157     6.988     1.641
         |      3.99      8.87      3.02
---------+------------------------------
   Total |       .03       .03       .05
         |  4.402625  3.875875  6.439875
         |      9.97      9.79      9.57
----------------------------------------
```

After looking at the last 4 observations (which are the group means because you specified `keepcenters`), you decide that what you really wanted to see was the minimum and maximum values and the mean for the four groups. You remove the last 4 observations and then use the `tabstat` command to view the desired statistics.

Group 1, with 15 students, is already doing well in flexibility and speed but will need extra strength training. Group 2, with 20 students, needs to emphasize speed training but could use some improvement in the other categories as well. Group 3, the largest, with 35 students, has serious problems with both flexibility and speed, though they did well in the strength category. Group 4, the smallest, with 10 students, needs help with flexibility and strength.

Because you like looking at graphs, you decide to view the matrix graph again but with group numbers used as plotting symbols.

```
. graph matrix flex speed strength, m(i) mlabel(g4abs) mlabpos(0)
```



The groups, as shown in the graph, do appear reasonably distinct. However, you had hoped to have groups that were about the same size. You are curious what clustering to three or five groups would produce. For no good reason, you decide to use the first k observations as initial group centers for clustering to three groups and random numbers within the range of the data for clustering to five groups.

```
. cluster k flex speed strength, k(3) name(g3abs) start(firstk) measure(abs)
. cluster k flex speed strength, k(5) name(g5abs) start(random(33576))
> measure(abs)
. table g3abs g4abs, col
```

```
─────────────────────────────────────────────
          │                g4abs
    g3abs │      1       2       3       4  Total
──────────┼──────────────────────────────────
        1 │                             10     10
        2 │             18      35             53
        3 │     15       2                     17
─────────────────────────────────────────────
```

```
. table g5abs g4abs, col

---------------------------------------------
          |              g4abs
    g5abs |     1      2      3      4  Total
----------+----------------------------------
        1 |           20                  20
        2 |    15                         15
        3 |                         6      6
        4 |                         4      4
        5 |                  35           35
---------------------------------------------
```

With three groups, the unequal-group-size problem gets worse. With five groups, the smallest group gets split. Four groups seem like the best option for this class. You will try to help the assistant assigned to group 3 in dealing with the larger group.

You might want to investigate the results of using different random seeds in the command used to generate the 4 groups earlier in this example. Because these data do not have clearly defined, well-separated clusters, there is a good chance that clusters based on different starting values will be different.

◁

▷ Example 2

You have just started a women's club. Thirty women from throughout the community have sent in their requests to join. You have them fill out a questionnaire with 35 yes–no questions relating to sports, music, reading, and hobbies. A description of the 35 variables is found in example 3 of [MV] **clustermat**.

In planning the first meeting of the club, you want to assign seats at the five lunch tables on the basis of shared interests among the women. You really want people placed together who share the same positive interests, not who share dislikes. From all the available binary similarity measures, you decide to use the Jaccard coefficient as the binary similarity measure because it does not include jointly zero comparisons in its formula; see [MV] ***measure_option***. The Jaccard coefficient is also easy to understand.

You decide to examine the groupings produced by kmeans and kmedians clustering.

```
. use http://www.stata-press.com/data/r13/wclub, clear

. cluster kmeans bike-fish, k(5) measure(Jaccard) st(firstk) name(gr5)

. cluster kmed bike-fish, k(5) measure(Jaccard) st(firstk) name(kmedian5)

. cluster list kmedian5
kmedian5  (type: partition,  method: kmedians,  similarity: Jaccard)
      vars: kmedian5 (group variable)
     other: cmd: cluster kmedians bike-fish, k(5) measure(Jaccard) st(firstk)
                 name(kmedian5)
            varlist: bike bowl swim jog hock foot base bask arob fshg dart clas
                 cntr jazz rock west romc scif biog fict hist cook shop soap
                 sew crft auto pokr brdg kids hors cat dog bird fish
            k: 5
            start: firstk
            range: 1 0
```

You used the first *k* observations as starting centers for both kmeans and kmedians—the `st(firstk)` option.

What size groups did each method produce, and how closely did the results agree?

```
. table gr5 kmedian5, row col

----------+-----------------------------------------
          |                 kmedian5
      gr5 |     1      2      3      4      5  Total
----------+-----------------------------------------
        1 |     7                                  7
        2 |     1      6                           7
        3 |                   5                    5
        4 |                          5             5
        5 |     1             1             4      6
          |
    Total |     9      6      6      5      4     30
----------+-----------------------------------------
```

There is reasonably strong agreement between the results from `cluster kmeans` and `cluster kmedians`. Because the tables can seat only eight comfortably, the grouping produced by `cluster kmeans` will be used because the group sizes range from five to seven, whereas the groups from `cluster kmedians` range from four to nine.

◁

Methods and formulas

Kmeans cluster analysis and its variant, kmedians cluster analysis, are discussed in most cluster-analysis books; see *References* in [MV] **cluster**. [MV] **cluster** also provides a general discussion of cluster analysis, including kmeans and kmedians clustering, and discusses the available `cluster` subcommands.

Kmeans and kmedians clustering are iterative procedures that partition the data into k groups or clusters. The procedure begins with k initial group centers. Observations are assigned to the group with the closest center. The mean or median of the observations assigned to each of the groups is computed, and the process is repeated. These steps continue until all observations remain in the same group from the previous iteration.

To avoid endless loops, an observation will be reassigned to a different group only if it is closer to the other group center. For a tied distance between an observation and two or more group centers, the observation is assigned to its current group if that is one of the closest and to the lowest numbered group otherwise.

The `start()` option provides many ways to specify the beginning group centers. These include methods that specify the actual starting centers, as well as methods that specify initial partitions of the data from which the beginning centers are computed.

Some kmeans clustering algorithms recompute the group centers after each reassignment of an observation. Other algorithms, including Stata's `cluster kmeans` and `cluster kmedians` commands, recompute the group centers only after a complete pass through the data. A disadvantage of this method is that orphaned group centers—one that has no observations that are closest to it—can occur. The advantage of recomputing means only at the end of each pass through the data is that the sort order of the data does not potentially change your result.

Stata deals with orphaned centers by finding the observations that are farthest from the centers and using them as new group centers. The observations are then reassigned to the closest groups, including these new centers.

Continuous or binary data are allowed with `cluster kmeans` and `cluster kmedians`. The mean of a group of binary observations for a variable is the proportion of ones for that group of observations and variable. The median of a group of binary observations for a variable is almost always either zero or one. However, if there are an equal number of zeros and ones for a group, the median is

0.5. The binary similarity measures can accommodate the comparison of a binary observation to a proportion. See [MV] ***measure_option*** for details on this subject and for the formulas for all the available (dis)similarity measures.

Reference

Makles, A. 2012. Stata tip 110: How to get the optimal k-means cluster solution. *Stata Journal* 12: 347–351.

Also see

[MV] **cluster notes** — Place notes in cluster analysis

[MV] **cluster stop** — Cluster-analysis stopping rules

[MV] **cluster utility** — List, rename, use, and drop cluster analyses

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

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