***Clustering***

SPSS macros by Kirill Orlov

kior@akado.ru, ttnphns@gmail.com

<https://www.spsstools.net/en/KO-spssmacros>

All rights reserved

*Clustering*. Macros for hierarchical cluster analysis (with options of constraint for a preexistent structure, precocious stop of agglomeration, and other), for computation of distances between already available groups/clusters and for assigning new objects to them. Macro for initializing cluster centres in K-means method clustering.

*Read “*[*About SPSS macros*](https://www.spsstools.net/en/KO-aboutmacros)*” what are they and how to run them.*

*The “Protected directory” error.* Some of the macros described in the current document write temporary files to hard disc. If you don't have full Administrator rights of your computer, it may cause error saying, among things: *“SPSS Statistics cannot access a file... specifies a protected directory...”*, meaning that the default directory the macro wants to use is protected on your PC. To solve the problem, in Syntax window issue command: CD 'myfolder'., where 'myfolder' is the path/name of some folder where you are allowed to save files to.

* [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK) performs agglomerative hierarchical cluster analysis (11 linkage methods to choose). Matrix of proximities (distances or similarities) between objects or their groups is required as input. The procedure offers several options: there is a possibility to make the algorithm first assemble clusters specified by the user, prior than merge them, and also a possibility to interrupt the agglomeration at any step and to save the remaining proximity matrix.
* [!KO\_HIECLUEX](#_MACRO_!KO_HIECLUEX:_HIERARCHICAL) performs agglomerative hierarchical cluster analysis with three exotic linkage methods: Hausdorff distance, Modified Hausdorff distance, and Point-centroid cross-distance.
* [!KO\_ASSCLU](#_МАКРОС_!ASSCLU:_СБОРКА) assembles the user-given clusters and computes distances between them, it also assigns new objects to these existent clusters. As the input there is required a matrix of distances or similarities between objects, an existing cluster structure (some grouping), and new objects, if any.
* [!KO\_POINTCLUD](#_MACRO_!KO_POINTCLUD:_DISTANCES) computes distances between objects and clusters (groups). As the input there is required a matrix of distances or similarities between objects and an existing cluster structure (some grouping).
* [!KO\_DENDRO](#_MACRO_!KO_DENDRO:_DENDROGRAM) builds dendrogram according to an agglomeration schedule of clustering.
* [!KO\_KMINI](#_MACRO_!KO_KMINI:_INITIAL) creates or selects initial centres for k-means cluster analysis; there is several methods to choose.

# MACRO !KO\_HIECLU: HIERARCHICAL CLUSTER ANALYSIS

Version 3, Sep 2019 (Version 1, Feb 2014). Tested on SPSS Statistics 20, 22, 25.

!KO\_hieclu matrix= *var1 to var150* /\*Columns constituting the matrix body, may use “to”

/seq= /\*Sequence of rows/columns must be as of the taken columns (COL, default) or

/\*as of the taken rows (ROW)

/id= *id* /\*Optionally: numeric identifier variable of rows/columns

/n= /\*If being clustered items are already clusters: variable with within-cluster frequencies

/square= /\*For distance matrix: at input, square its elements (YES) or don’t do (NO, default)

/method= BAVERAGE /\*Linkage method: SINGLE, COMPLETE, BAVERAGE, EQBAVER, WAVERAGE, CENTROID,

/\*MEDIAN, WARD, MNSSQ, MIVAR, MNVAR

/preclu= *clu8* /\*Optionally: cluster membership variable setting preexistent structure

/stop= /\*Optionally: stop agglomeration prematurely: STEP number, NCLU number, PROP proportion

/\*IPROP proportion, COEF value, MAXN number, BOTHN number, ASS1, ASS2, ASS3 (see)

/save= *2 20* /\*Optionally: save cluster membership variables as new dataset:

/\*one cluster solution (number) or range of solutions (two numbers) or all (ALL),

/\*or at the moment of stop (ATSTOP)

/msave= /\*Optionally, if STOP specified: save remnant matrix (filename)

/sched= *'d:\exercise\history.sav'* /\*Optionally: save agglomeration schedule

/\*(file or declared dataset name)

/root= /\*For some methods (see): take root of agglomeration schedule (YES) or

/\*not (NO, default)

/cumul= /\*Cumulate values in agglomeration schedule (YES) or no (NO, default)

/print= /\*Printout: full (FULL, default), w/o info about ID of rows/columns in matrix (NOID),

/\*w/o agglomeration schedule (NOSCHED), only summary (SUMM).

Minimal specification MATRIX, METHOD.

The macro does hierarchical clustering based on a pairwise proximity matrix (dissimilarities/distances or similarities). Offered are 11 methods of agglomeration (linkage), including “flexible” average linkage methods. Note that methods – centroid, median, Ward, sum-of-squares, variance increment, variance – require, for the sake of geometric correctness, the matrix of squared euclidean distances, at worst case – other metric distances (squared), but in no case similarities.

The macro is somewhat slower than command CLUSTER of SPSS, but considerably excels it by wealth of possibilities in-built in the algorithm. Specifically, besides usual clusterization of objects the macro allows to:

* cluster not just single objects but their groups (in some cases it is equivalent to frequency weighting of objects) – see subcommand N;
* impose preexistent arbitrary cluster structure which the algorithm will have to gather stepwisely before it will go on agglomerating usual way – see subcommand PRECLU;
* break agglomeration at any step, arbitrary or by a criterion, and save the remaining proximity matrix – see subcommand STOP;
* assign new objects to existent clusters and not do any agglomeration among the latter;
* there is more linkage methods in the macro than in CLUSTER command.

The macro does not build dendrogram or other graphs. To obtain dendrogram, use separate macro !KO\_DENDRO working with agglomeration schedule that you can save. You can also save cluster solutions – cluster membership variables. In the Viewer, the macro outputs agglomeration schedule and information about the analysis.

EXAMPLE 1.

proximities v1 to v10 /view= case /measure= seuclid /matrix= out(\*) /print= none.

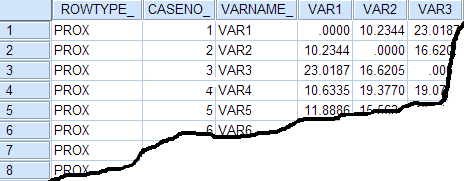
!KO\_hieclu matrix= var1 to var80 /method= COMPLETE /sched= 'd:\exercise\schedule.sav'.

!KO\_dendro sched= 'd:\exercise\schedule.sav'.

* Being clustered are 80 cases (by10 variables). Matrix of squared euclidean distances between cases is computed by PROXIMITIES command and output as new working dataset.
* Macro !KO\_HIECLU performs clustering by complete linkage method and saves agglomeration history as external file.
* Macro !KO\_DENDRO takes that file and builds dendrogram.

***Matrix structure***

The dataset must be a matrix of pairwise distances or similarities. Variable names – matrix columns – up to 8 bytes. Required is the presence of variable VARNAME\_ naming rows in correspondence to columns. Names, which are the values of the variable, should be written in the same case of letters as the identical to them names of the columns. The macro does not require that the rows and the columns go in the same order or their number and list be fully identical: the macro will by itself select from the input matrix rows and columns with the same names and will co-order them so that the compiled matrix be square and diagonalized structure. Variable ROWTYPE\_ and other auxiliary are not necessary in the input matrix.



***Subcommands***

**MATRIX**

Specify variables of the working dataset which are proper the columns of the matrix of distances or similarities. You may list all or just needed columns and in arbitrary sequence. May use “to” to specify by range.

*Specification of open range with the help of “?”*. In some instances there may arise need to specify a range between a pair of variables which themselves are not included in the range. Use “?” at the sides, for that. For example, *?VARNAME\_ to ENDVAR?* means all variables found in the dataset between variables *VARNAME\_* and *ENDVAR*, excluding these two. To specify a range which is open from one side, use “?” only from that side. For example: *?VARNAME\_ to VAR100* or *VAR1 to ENDVAR?*.

If your data are **distances** (dissimilarities) then “diagonal” values – i.e., data in cells on the intersection of rows and columns of the same name – must be *zeros*, and all other (“offdiagonal”) values must be nonnegative; greater value corresponds to greater dissimilarity. While if your data are **similarities** then the “diagonal” values must be *positive*, “offdiagonal” values may have any sign; greater value (greeting the sign) corresponds to greater similarity. The macro does not use diagonal values in clustering, but it knows if the data are distances or similarities by looking at those values.

**SEQ**

Upon taking the input matrix according to MATRIX subcommand the macro compiles, out of its present rows and columns with the same name, the square symmetric matrix for the analysis. Subcommand SEQ dictates in what sequence rows/columns will go in it:

COL - (default) rows and columns will go in the order of column names list as it is in MATRIX subcommand.

ROW - rows and columns will go in the order of cases (values of VARNAME\_) in the input matrix.

If the matrix, as taken by MATRIX subcommand, appears already square and with co-ordered rows and columns then subcommand SEQ plays no role.

EXAMPLE 2. Change objects order in matrix for another, random.

compute sortvar= uniform(1).

sort cases by sortvar.

!KO\_hieclu matrix= var1 to var80 /seq= ROW /id= objid /method= BAVERAGE.

* Rows of the input matrix are being sorted by values of random variable SORTVAR.
* SEQ=ROW of the macro compiles the matrix for the analysis, where order of rows/columns will be as of the rows of the input matrix.
* Object identifier variable OBJID (attached to the matrix) retains the original identification of rows/columns.
* Changing of order in matrix usually does not affect results; it may affect if there are equal (tied) values of proximities in the matrix.

**ID**

You may specify name (up to 8 bytes) of a numeric variable identifying rows/columns. Its values will be used in agglomeration schedule – in its columns *Cluster1* and *Cluster2*, to designate the merging clusters. Normally that variable should have no duplicating values and missing values; their presence won’t affect clusterization, however the agglomeration schedule will become useless for subsequent correct building of the dendrogram. Value: *-999* is not allowed in the variable. By default/nonspecifying of this subcommand, the sequential numbers of rows/columns in the matrix compiled for the analysis will become their (and hence cluster) identifiers.

*Designating clusters in agglomeration schedule*. When two clusters are merged the cluster with lesser identifying code appears in column Cluster1, and the cluster with greater code – in column Cluster2; the emergent cluster inherits the lesser code of the two, i.e. code of Cluster1, and it appears further in the agglomeration schedule under that code until it merges with a cluster having yet a smaller code.

There is also column *CluCode* in agglomeration schedule, showing the code of the emergent cluster: it is its code in the cluster membership variable (see s/c SAVE), not the identificatory code that was spoken about above and that appears in columns Cluster1 and Cluster2.

**N**

If rows/columns of the input matrix represent not single objects but clusters or groups of objects then it is necessary to tell the macro what are the frequencies in them. Specify the attached variable (name up to 8 bytes) showing the within-cluster number of objects (values – positive integers). If all values =1 then it is equivalent to clustering of singleton objects, that is, to not using the subcommand.

It is assumed that the proximities between the clusters (groups) contained in the matrix should correspond to the nature of method of clustering you are going to use (see s/c METHOD and also s/c MSAVE). For example, if you are about to use centroid method, the input distance between two groups in the matrix is expected to be the squared distance between centroids of the groups. Or, for a more complex example, if you are about to use Ward’s method, then the input distance between two groups in the matrix is expected to be the double increase of sum of squared deviations emerging from the union of those two groups.

This subcommand is incompatible with methods WAVERAGE, MNSSQ, and MNVAR. Clustering objects by SINGLE, COMPLETE, EQBAVER, MEDIAN methods is insensitive to this subcommand.

Whether s/c N can be used for the purpose of frequency *weighting* of objects – see paragraph “Some questions” below.

**SQUARE**

This subcommand is only for distances (dissimilarities). Some methods in the macro (CENTROID, MEDIAN, WARD, MNSSQ, MIVAR, MNVAR) demand the input distances to be squared. If your matrix contains nonsquared distances you may request to square them at input: SQUARE=YES.

**METHOD**

Specify linkage method in agglomeration. The methods differ in respect to how they define proximity between any two clusters at every step. Colligation coefficient (shown in agglomeration schedule) is the proximity between the two clusters merged at a given step.

SINGLE - method of **single linkage** or **nearest neighbour**. Proximity between two clusters is the proximity between their two closest objects. This value is one of values of the input matrix.

COMPLETE - method of **complete linkage** or **farthest neighbour**. Proximity between two clusters is the proximity between their two most distant objects. This value is one of values of the input matrix.

BAVERAGE [*value*] - method of **between-group average linkage** (UPGMA). Proximity between two clusters is the arithmetic mean of all the proximities between the objects of one, on one side, and the objects of the other, on the other side. About optional parameter *value* after the keyword – see below.

EQBAVER [*value*] - **simple average**, or method of **equilibrious** **between-group average linkage** (WPGMA). Proximity between two clusters is the arithmetic mean of all the proximities between the objects of one, on one side, and the objects of the other, on the other side; while the subclusters of which each of these two clusters were merged recently have equalized influence on that proximity – even if the subclusters differed in the number of objects. About optional parameter *value* after the keyword – see below.

WAVERAGE - method of **within-group average linkage** (MNDIS). Proximity between two clusters is the arithmetic mean of all the proximities in their joint cluster.

CENTROID - **centroid** method (UPGMC). Proximity between two clusters is the proximity between their geometric centroids: squared euclidean distance between those.

MEDIAN - **median**, or **equilibrious** **centroid** method (WPGMC). Proximity between two clusters is the proximity between their geometric centroids (squared euclidean distance between those); while the centroids are defined so that the subclusters of which each of these two clusters were merged recently have equalized influence on its centroid – even if the subclusters differed in the number of objects.

WARD - **Ward**’s method, or minimal **increase of sum-of-squares** (MISSQ). Proximity between two clusters is the magnitude by which the summed square in their joint cluster will be greater than the combined summed square in these two clusters: SS12-(SS1+SS2). (Between two singleton objects this quantity = squared euclidean distance / 2.)

MNSSQ - method of minimal **sum-of-squares**. Proximity between two clusters is the summed square in their joint cluster: SS12. (Between two singleton objects this quantity = squared euclidean distance / 2.)

MIVAR - method of minimal **increase of variance**. Proximity between two clusters is the magnitude by which the mean square in their joint cluster will be greater than the weightedly (by the number of objects) averaged mean square in these two clusters: MS12-(n1MS1+n2MS2)/(n1+n2) = [SS12-(SS1+SS2)]/(n1+n2). (Between two singleton objects this quantity = squared euclidean distance / 4.)

MNVAR - method of minimal **variance**. Proximity between two clusters is the mean square in their joint cluster: MS12 = SS12/(n1+n2). (Between two singleton objects this quantity = squared euclidean distance / 4.).

Methods EQBAVER, MNSSQ, MIVAR, MNVAR are absent in SPSS; see about them e.g. Podany J. New combinatorial clustering methods // Vegetatio, 1989, 81: 61-77.

First 5 methods permit any proximity measures (any similarities or distances). Last 6 methods require distances; and fully correct will be to use only squared euclidean distances, because these methods compute centroids in euclidean space. You can square distances by SQUARE s/c.

Methods centroid, median, minimal increase of variance – may give sometimes the so-called *reverses*: a phenomenon when the two clusters being merged at some step appear closer to each other than pairs of clusters merged earlier. That is because these methods do not belong to so the called ultrametric.

Methods of single linkage and centroid belong to so called *space contracting*, or “chaining”. That means that they tend to attach objects one by one to clusters, and so they demonstrate relatively smooth growth of curve “% of clustered objects”. On the contrary, methods of complete linkage, Ward’s, sum-of-squares, increase of variance, and variance commonly get considerable share of objects clustered even on early steps, and then proceed merging yet those – therefore their curve “% of clustered objects” is steep from the first steps. These methods are called *space* *dilating*. Other methods fall in-between.

**Flexible average linkage methods UPGMA and WPGMA** (Belbin, L. et al. A Comparison of Two Approaches to Beta-Flexible Clustering // Multivariate Behavioral Research, 1992, 27, 417–433.) are specified by using the parameter in the range -1≤ *value*<1 after word BAVERAGE (flexible UPGMA) or EQBAVER (flexible WPGMA). For example: METHOD=BAVERAGE -0.1. Parameter 0 is equivalent to standard UPGMA or WPGMA. The parameter *brings in correction* for the being computed between-cluster proximity, which depends on the size (amount of de-compactness) of the clusters. The meaning of the parameter is that it makes the method of agglomeration more space dilating as the parameter gets more negative, and more space contracting as it gets more positive. It is recommended to use values close to 0, otherwise quality decreases. Based on studying of UPGMA and WPGMA ability to restore simulated clusters in euclidean space Belbin and colleagues concluded that the parameter a bit below 0 (about -0.1 for UPGMA and -0.2 for WPGMA) is optimal in the majority of cases, and at presence of outliers/noise it is useful to shift it somewhat lower.

**PRECLU**

By this subcommand you can specify preexistent cluster structure. Specify numeric cluster membership variable (name up to 8 bytes) attached to the matrix dataset. Values of the variable are the codes of some preexistent clusters or groups. Then the macro will first do agglomeration separately within these clusters (as if they were different datasets) to assemble each, and next will continue agglomeration now among these clusters. Thus, the macro in any case starts agglomerating from separate objects (or groups thereof – in the presence of N subcommand), but in this case it does it in the regime of constraint for merge among objects.

The preexistent cluster structure variable may have *missings*. Missings are treated by the macro as objects that are prohibited to merge with *each other* prior they are merged with clusters or nonmising objects. The variable should have no value: *-999*.

EXAMPLE 3.

!KO\_hieclu matrix= var1 to var80 /method= BAVERAGE /preclu= agegroup /save= 2 10.

dataset name clusol.

* Being clustered are 80 objects already grouped (“preclustered”) according to AGEGROUP. The macro will initially cluster to attain the classification AGEGROUP; when it assembles it will continue agglomerating now merging those groups.
* Requested is to save cluster solutions 2 through 10 clusters in new dataset. That dataset is named CLUSOL.

**STOP**

Optional subcommand by which you can stop (exit) agglomeration prior its end. Specify stopping criterion:

STEP *integer* - stop after this, by number, step of agglomeration.

NCLU *integer* - stop upon reaching this number of clusters (single object is also count as a cluster).

PROP *proportion* - stop upon reaching a specific share of object clusteredness, i.e. proportion of objects that ceased be singles and found at least pair for them: specify decimal fraction. This option is sensitive to specifying N s/c, because those are frequencies of objects considered already clustered together.

IPROP *proportion* - stop upon reaching a specific share of matrix rows/columns clusteredness: specify decimal fraction. This option is allowed only if s/c N is specified.

COEF *value* - stop upon reaching the given value of the colligation coefficient. This option is sensitive to how subcommand CUMUL is specified, and also to ROOT.

MAXN *integer* - stop as soon as merging formed a cluster with this (not less) number of objects.

BOTHN *integer* - stop as soon as there merged two clusters each of which had no less than this number of objects.

The following three stop options require subcommand PRECLU be specified and they mean to do only *assemble* of the preexistent cluster structure and *assigning* of the new objects to those existent clusters. For “new objects” these options take missings in PRECLU variable. Having done assemble/assignment the macro will stop: no agglomeration between the clusters will follow.

ASS1 - only assemble preexistent clusters; do not assign to them new objects (the latter will stay separate objects).

ASS2 - do assemble of preexistent clusters and then independent assignment of new objects to them; in this version a new object (a missing) after getting enrolled to the closest to it cluster does not influence assignments of other new objects, because characteristics of clusters are hold till the end of assignment process.

ASS3 - do assemble of preexistent clusters and then progressive assignment of new objects to them; in this version a new object (a missing) after getting enrolled to the closest to it cluster can influence subsequent assignments of other new objects, because assignments runningly change characteristics of clusters.

If there were no missings in PRECLU variable, ASS2 and ASS3 are equivalent to ASS1: only assemble. Options ASS1 and ASS2 are possible also to accomplish by macro [!KO\_ASSCLU](#_МАКРОС_!ASSCLU:_СБОРКА).

EXAMPLE 4. Assign new objects to old clusters.

!KO\_hieclu matrix= var1 to var320 /method= CENTROID /preclu= clu /stop= ASS3 /id= clu /save= ATSTOP.

* Variable CLU is old clusters – identified in past wave of segmenting respondents, plus – in the form of missings – new respondents. Distance matrix between all – old plus new – respondents enters the analysis.
* STOP=ASS3 will first pull old respondents together in those old clusters (this is the preexistent structure), after that will assign (by progressive variant) new respondents to their closest clusters (by centroid method approach). There will be no subsequent (between-cluster) agglomeration.
* The researcher decided here to enter variable CLU itself as the cluster identifier ID – in order to make sure, by contemplating agglomeration schedule, that at first old clusters will be gathered and then new objects will be assigned to them.
* SAVE=ATSTOP will save cluster solution at the moment of the process stop, i.e., after assembly and assignment: the output cluster membership variable *CLUSTOP* in this example will be equivalent to the ID variable saved beside it, albeit it will contain different codes.

**SAVE**

Subcommand for saving cluster solutions – cluster membership variables into new unnamed working dataset. Specify one number (the solution with this number of clusters will be saved) or two numbers, 1st less than the 2nd (range of solutions from… to … clusters will be saved), or ALL (all cluster solutions will be saved). Cluster codes in these saved variables are always sequential numbers 1, 2, 3,…, and not codes of the identifier (see s/c ID) which is also saved in concert (if ID wasn’t specified then sequential numbers of rows/columns are used, *SEQNO\_*). Also, variable *VARNAME\_* of the input matrix is saved.

With s/c STOP specified, you may as well put SAVE=ATSTOP. In this case the macro saves two cluster membership variables: clusters left after the step of break of the agglomeration (*CLUSTOP*) and clusters at one step before it (*CLUSTOP#*).

**MSAVE**

This subcommand is allowed if s/c STOP is specified. It saves the matrix as it is at the moment of agglomeration break. Specify external SAV file for saving, in quotes or apostrophes (may not specify a declared dataset). You can later use the matrix again as the input matrix to !KO\_HIECLU, which will amount to resuming the interrupted clusterization. You can also use the matrix in other kinds of analysis, for example multidimensional scaling.

Values in the being saved matrix are the proximities between the clusters as they were computed, by the agglomeration method you used, by the moment of exit from the agglomeration. Definition, what is the proximity between clusters with the given agglomeration method – see it in the description of s/c METHOD. Please note that with methods WARD and MNSSQ their output proximities (see s/c METHOD) are multiplied by 2, and with methods MIVAR and MNVAR are multiplied by 4: so is required for the sake of compatibility with the input proximities (normally squared euclidean distances, with these methods).

There will be column *METHOD\_* with the matrix informing about the method used, a memo (you may delete it). Also, there will be column *N\_* with the within-cluster frequencies of objects. During resuming the clustering with !KO\_HIECLU macro you will have to name that column in N subcommand. Next, at the matrix there will be identifier column for rows/columns – these are the codes having been designating clusters in the agglomeration schedule: either the codes from your specified variable (s/c ID) or sequential numbers in the analyzed matrix (the column then will be named *SEQNO\_*). Column *VARNAME\_* in the saved matrix – is *not* that column with the same name from the input matrix but is newly created column consisting of names *var1 var2 var3*, etc.

Attention. In order to correctly use the saved matrix in native SPSS Statistics procedures such as CLUSTER command, you will need to delete columns *METHOD\_* and *N\_*, and also to prescribe a label to the value ‘PROX’ of column *ROWTYPE\_*. The label must begin with the word DISSIMILARITY if the proximities in the matrix are dissimilarities, and with the word SIMILARITY, if the proximities in the matrix are similarities.

EXAMPLE 5. Resuming an interrupted clusterization.

!KO\_hieclu matrix= var1 to var80 /method= CENTROID /stop= NCLU 20 /msave= 'd:\exercise\mat.sav'.

get file 'd:\exercise\mat.sav'.

!KO\_hieclu matrix= var1 to var20 /n= n\_ /method= CENTROID.

* The first command clusters 80 objects and stops when there remain 20 clusters. The remnant matrix is being saved.
* The matrix gets opened and the second clustering command clusters to the end. Note that subcommand /N is applied. The results will be the same as if to cluster everything till the end at one time.
* Normally one should use in the second clustering the former agglomeration method (in this case, CENTROID) because with each method the proximities are computed in its own way. Departure from that rule is possible, but the consequences should be thoroughly thought over. Method mismatch will make cluster analysis not halt-resume but rather heuristic two-stage one.

**SCHED**

Subcommand saving agglomeration schedule (dendrogram can be built on it later). Indicate SAV file to save, in quotes or apostrophes, or the name of an earlier declared yet not existing dataset. May also specify SCHED=\* to save to the new unnamed dataset, but then don’t use subcommand SAVE.

**ROOT**

This subcommand acts only for methods with linkage in the form of squared distance (CENTROID, MEDIAN) or sum of squares of deviations (WARD, MNSSQ, MIVAR, MNVAR). If ROOT=YES, the macro returns agglomeration schedule with square root taken of colligation coefficients. By default, ROOT=NO, which corresponds to the implementation in SPSS command CLUSTER. The difference is only in the being displayed and saved coefficients of colligation in the agglomeration schedule; the results of clustering are the same. Remember that for the 6 named methods the macro expects the input distances to be square.

**CUMUL**

This subcommand manages the mode of presentation of colligation coefficients in the agglomeration schedule. It does not influence clustering results but is important for building a dendrogram. Also, CUMUL=YES is incompatible with an analysis of similarities. By default and with CUMUL=NO, the proximity between two merging clusters is used as the colligation coefficient. With CUMUL=YES, the *cumulated* proximity is used: the colligation coefficient at each step is the coefficient on the previous step plus the proximity on the given step.

Traditionally, in methods based on *increment* of nondensity, such as Ward’s method or MIVAR, there they usually show (including on the dendrogram) cumulative coefficient (CUMUL=YES), it is sooner for convenience reasons than theoretical ones. Thus, with CUMUL=YES the returned colligation coefficient in Ward’s method represents the overall within-cluster sum-of-squares observed at the moment of a given step. In the majority of other methods – not Ward or MIVAR – they usually show the proximity in the pair being currently merged (CUMUL=NO). If ROOT=YES, the root is taken before cumulating.

EXAMPLE 6. Different versions of return in Ward’s method.

!KO\_hieclu matrix= var1 to var80 /method= WARD /sched= 'd:\exercise\schedule.sav' /cumul= YES.

!KO\_hieclu matrix= var1 to var80 /method= WARD /sched= 'd:\exercise\schedule.sav' /root= YES.

* The first command returns agglomeration schedule (and hence dendrogram) as in SPSS procedure Cluster.
* The second command returns agglomeration schedule (and hence dendrogram) similar to R command hclust where the so called “Ward-2” algorithm version is realized (see. Murtagh F., Legendre P. Ward's hierarchical clustering method // Journal of Classification, 31, 2014).
* The clustering results are the same, while the general looks of the dendrogram may somewhat differ.

**PRINT**

Printout to Viewer window:

FULL - (default) detailed report: summary, information on identification/order of rows/columns in the analyzed matrix, agglomeration schedule.

NOID - don’t show information on identification/order of rows/columns in the analyzed matrix.

NOSCHED - don’t show agglomeration schedule.

SUMM - show only summary.

***Special regimes***

The macro does not obey weighting (however, it doesn’t take in the procedure cases with missing and nonpositive weights). It is not suited for the split state of the dataset (SPLIT FILE). The macro obeys commands selecting cases (SELECT IF, FILTER, USE), including those standing under TEMPORARY command.

# 

***Some questions***

*What is the difference of N subcommand and PRECLU subcommand?* If each row/column of the input matrix is itself already a cluster/group then you have to specify the number of objects in it; it is being done via the variable named in s/c N. But if it is requested to make agglomeration to gather several rows/columns in one cluster in the first place then you have to link them by the code of such cluster in the variable named in s/c PRECLU. Both subcommands may exist together.

*I want to frequency-weight objects: so that each row of the proximity matrix would mean several identical objects. May I use s/c N for that?* Yes, if you are using methods SINGLE, COMPLETE, BAVERAGE, EQBAVER, CENTROID, MEDIAN. When these methods merge two clusters, they pay no attention to the density within the clusters. Therefore s/c N will be equivalent to the explicit propagation of the rows/columns of the matrix, i.e., to frequency weighting of the objects. In general, there exist the following groups of methods among the methods, for which the macro permits using s/c N:

* SINGLE, COMPLETE, EQBAVER, MEDIAN. These methods pay no attention to the within-cluster frequencies when they merge clusters: they are insensitive to frequency weighting of objects. So, using or not using of s/c N does not affect clustering of objects into clusters.
* BAVERAGE, CENTROID. These methods are sensitive to the within-cluster frequencies, but are insensitive to the within-cluster densities of the being merged clusters. You may use s/c N for frequency weighting, and its using affects the results of clustering; the results are the same as with explicit propagation of rows/columns.
* WARD, MIVAR. These methods are sensitive to the within-cluster densities of the being merged clusters. Specifying s/c N with frequencies is not equivalent in result to the explicit propagation of rows/columns of the matrix. Therefore, if you need frequency weighting of objects, do propagation of rows/columns of the matrix.

# MACRO !KO\_HIECLUEX: HIERARCHICAL CLUSTER ANALYSIS WITH EXOTIC LINKAGE METHODS

Version 1, Sep 2019. Tested on SPSS Statistics 20, 22, 25.

!KO\_hiecluex matrix= *var1 to var150* /\*Columns constituting the matrix body, may use “to”

/seq= /\*Sequence of rows/columns must be as of the taken columns (COL, default) or

/\*as of the taken rows (ROW)

/id= *id* /\*Optionally: numeric identifier variable of rows/columns

/fw= /\*Optionally: variable with frequency weights

/method= CROSSPC /\*Linkage method: HAUSDORFF, MHAUSDORFF, CROSSPC

/stop= /\*Optionally: stop agglomeration prematurely: STEP number, NCLU number,

/\*PROP proportion, IPROP proportion, COEF value, MAXN number, BOTHN number

/save= *2 20* /\*Optionally: save cluster membership variables as new dataset:

/\*one cluster solution (number) or range of solutions (two numbers) or all (ALL),

/\*or at the moment of stop (ATSTOP)

/msave= /\*Optionally, if STOP specified: save remnant matrix (filename)

/sched= *'d:\exercise\history.sav'* /\*Optionally: save agglomeration schedule

/\*(file or declared dataset name)

/print= /\*Printout: full (FULL, default), w/o info about ID of rows/columns in matrix (NOID),

/\*w/o agglomeration schedule (NOSCHED), only summary (SUMM).

Minimal specification MATRIX, METHOD.

The macro does hierarchical clustering based on a pairwise matrix of dissimilarities (distances). Offered are 3 methods of agglomeration (linkage): Hausdorff distance, Modified Hausdorff distance, Point-centroid cross-distance. These distances are of a “set distances” kind. Hierarchical linkage methods based on these distances we conditionally named exotic, because the standard implementation founded on Lance–Williams recurrent formula (which is at the core of !KO\_HIECLU) is not suitable for them. !KO\_HIECLUEX therefore is programmed as another macro. In other respect functionality and options of !KO\_HIECLUEX are similar to those of [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK).

Note that method Point-centroid cross-distance requires, for the sake of geometric correctness, the matrix of euclidean distances (non-squared); other distances/dissimilarities will suit only to the extent how much they are close to euclidean distances in the sense of being able to form Euclidean space.

EXAMPLE 1.

proximities v1 to v10 /view= case /measure= jaccard /matrix= out(\*) /print= none.

do repeat var= var1 to var80.

compute var= 1-var.

end repeat.

!KO\_hiecluex matrix= var1 to var80 /method= HAUSDORFF /sched= 'd:\exercise\schedule.sav'.

!KO\_dendro sched= 'd:\exercise\schedule.sav'.

* Being clustered are 80 cases (by10 binary variables). Matrix of Jaccard similarities between cases is computed by PROXIMITIES command and output as new working dataset.
* Jaccard similarities are converted to dissimilarities by subtracting from 1.
* Macro !KO\_HIECLUEX performs clustering by Hausdorff linkage method and saves agglomeration history as external file.
* Macro !KO\_DENDRO takes that file and builds dendrogram.

***Matrix structure***

See description in macro [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK). In !KO\_HIECLUEX, the requirements towards the matrix are the same, but the matrix can only be distances (dissimilarities): !KO\_HIECLUEX is not intended for similarities.

***Subcommands***

**MATRIX**

Specify variables of the working dataset that are proper the columns of the matrix of distances. You may list all or just needed columns and in arbitrary sequence. May use “to” to specify by range.

*Specification of open range with the help of “?”*. Same as in !KO\_HIECLU.

The matrix must be a distance (dissimilarity) matrix. Its “diagonal” values – i.e., data in cells on the intersection of rows and columns of the same name – must be *zeros*, and all other (“offdiagonal”) values must be nonnegative; greater value corresponds to greater dissimilarity.

**SEQ**

This subcommand is the same as in macro [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK).

**ID**

You may specify name (up to 8 bytes) of a numeric variable identifying rows/columns. Its values will be used in agglomeration schedule – in its columns *Cluster1* and *Cluster2*, to designate the merging clusters. This subcommand is the same as in macro !KO\_HIECLU (read there) except that it does not permit missings in the identifier variable.

*Designating clusters in agglomeration schedule*. Same way as in !KO\_HIECLU (read there).

**FW**

Optional subcommand for frequency weighting. You can request the macro to count, at the clusterization, this or that matrix row/column not a single object but multiple objects – a group of *identical* objects. For that, specify the attached variable (name up to 8 bytes) showing the number of object copies (values – positive integers). If all values =1 then it is equivalent to clustering of singleton objects, that is, to not using the subcommand.

Colligation coefficient with METHOD=HAUSDORFF is insensitive to this subcommand.

How this s/c differs from similar N s/c of macro !KO\_HIECLU – see paragraph “Some questions” below.

**METHOD**

Specify linkage method in agglomeration. The methods differ in respect to how they define proximity between any two clusters at every step. Colligation coefficient (shown in agglomeration schedule) is the proximity between the two clusters merged at a given step.

HAUSDORFF - proximity between two clusters is **Hausdorff distance** between them. It is defined as follows. Each object of the first cluster finds for itself the nearest neighbour in the second cluster, and from these (minimal) distances one maximal is taken. And oppositely: each object of the second cluster finds its nearest neighbour in the first cluster, and from these (minimal) distances one maximal is taken. Of the two thus found distances the greater is chosen.

MHAUSDORFF - proximity between two clusters is **Modified Hausdorff distance** between them. It is defined as follows. Each object of the first cluster finds for itself the nearest neighbour in the second cluster, and of these (minimal) distances the arithmetic average is taken. And oppositely: each object of the second cluster finds its nearest neighbour in the first cluster, and of these (minimal) distances the arithmetic average is taken. Of the two thus obtained values the greater is chosen.

CROSSPC - proximity between two clusters is **Point-centroid cross-distance** between them. It is the sum of distances from objects of one cluster to the centroid of the second cluster, plus the sum of distances from objects of the second one to the centroid of the first, and divide by the combined number of objects in the two clusters.

CROSSPC expects that the input distances are euclidean (nonsquared). If your distances are non-euclidean the computed result can be informatively invalid; and there also can happen the error of taking square root of a negative number. HAUSDORFF/ MHAUSDORFF are compatible with any dissimilarities.

All three methods may give sometimes the so-called *reverses*: a phenomenon when the two clusters being merged at some step appear closer to each other than pairs of clusters merged earlier. That is because these methods do not belong to so the called ultrametric.

**STOP**

Optional subcommand by which you can stop (exit) agglomeration prior its end. Specify stopping criterion:

STEP *integer* - stop after this, by number, step of agglomeration.

NCLU *integer* - stop upon reaching this number of clusters (single object is also count as a cluster).

PROP *proportion* - stop upon reaching a specific share of object clusteredness, i.e., proportion of objects which ceased be singles and found at least pair for them: specify decimal fraction. This option is sensitive to specifying FW s/c, because object “duplicates” are considered objects already clustered together.

IPROP *proportion* - stop upon reaching a specific share of matrix rows/columns clusteredness: specify decimal fraction. This option is allowed only if s/c FW is specified.

COEF *value* - stop upon reaching the given value of the colligation coefficient.

MAXN *integer* - stop as soon as merging formed a cluster with this (not less) number of objects.

BOTHN *integer* - stop as soon as there merged two clusters each of which had no less than this number of objects.

**SAVE**

Subcommand for saving cluster solutions – cluster membership variables into new unnamed working dataset. The s/c is the same as that in [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK).

**MSAVE**

This subcommand is allowed if s/c STOP is specified. It saves the distance matrix as it is at the moment of agglomeration break. Specify external SAV file for saving, in quotes or apostrophes. This subcommand is completely analogous to that in [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK).

**SCHED**

Subcommand saving agglomeration schedule (dendrogram can be built on it later). This subcommand is identical to that in [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK).

**PRINT**

Printout to Viewer window. This subcommand is identical to that in [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK).

***Special regimes***

The macro does not obey weighting (however, it doesn’t take in the procedure cases with missing and nonpositive weights). It is not suited for the split state of the dataset (SPLIT FILE). The macro obeys commands selecting cases (SELECT IF, FILTER, USE), including those standing under TEMPORARY command.

***Some questions***

*How does subcommand FW differ from subcommand N of macro !KO\_HIECLU?* Both subcommands are akin in that they specify a variable with frequencies *n* of objects in groups (each row/column of the matrix is seen then as a group of objects). S/c N does not demand *n* objects to be identical to each other. S/c FW considers *n* objects are identical, each other copies. Thus, FW is a subcommand for frequency weighting; N is wider functionally, but can be used for frequency weighting as a particular case (see details in !KO\_HIECLU).

*May I run in !KO\_HIECLUEX a stop-resume agglomeration like in EXAMPLE 5 of macro !KO\_HIECLU?* Not quite so. This is because of the difference between FW and N. In !KO\_HIECLUEX, you too can break agglomeration and save the matrix, then agglomerate that matrix further, specifying /FW=*N\_*. However, FW will see all the *n* objects (*n* – a value from *N\_* variable) *identical* objects, whereas in fact these are just objects collected in one cluster and not having to be same, in general case.

# MACRO !KO\_ASSCLU: CLUSTER ASSEMBLY / OBJECT ASSIGNMENT TO CLUSTERS

Version 2, Jan 2020 (Version 1, June 2015). Tested on SPSS Statistics 20, 22, 25.

!KO\_assclu matrix= *var1 to var150* /\*Columns constituting the matrix body, may use “to”

/seq= /\*Sequence of rows/columns must be as of the taken columns (COL, default) or

/\*as of the taken rows (ROW)

/id= *id* /\*Numeric identifier variable of objects

/cluvar= *clu5* /\*Cluster membership variable of preexistent structure

/square= /\*For distance matrix: at input, square its elements (YES) or don’t do (NO, default)

/method= BAVERAGE /\*Linkage method: SINGLE, COMPLETE, BAVERAGE, WAVERAGE, CENTROID,

/\*WARD, MNSSQ, MIVAR, MNVAR, HAUSDORFF, MHAUSDORFF, CROSSPC

/save= YES /\*Optional: save updated cluster membership variable as new dataset –

/\*YES or NO (default)

/msave1= *'d:\exercise\mx1.sav'* /\*Optional: save intermediate assembled matrix

/\*(filename)

/msave2= *'d:\exercise\mx2.sav'* /\*Optional: save final assembled matrix

/\*(filename)

/print= /\*Printout: full (FULL, default), w/o info about ID of rows/columns in matrix (NOID),

/\*only summary (SUMM).

Minimal specification MATRIX, ID, CLUVAR, METHOD.

This macro is intended to establish proximities between known clusters (or generally, between any groups of objects defined by a nominal variable), and to enlist objects whose belonging to the clusters/groups is unknown (“new objects”) to their nearest clusters/groups.

The macro takes a proximity matrix between objects, and a variable of some preexistent cluster (group) structure. Without doing actually hierarchical cluster analysis (stepwise agglomeration) it **assembles** those preexistent clusters by a requested linkage method, the result of which is the proximity matrix between the clusters. Then, if there are “new” objects that are required to **assign** to those preexistent clusters (each object – to its closest cluster), the macro does it.

Everything this macro does can be done also in [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK) (see options ASS1 – assembly, and ASS2 – assembly/assignment there in subcommand STOP). But !KO\_ASSCLU is faster than !KO\_HIECLU, and it is its use. !KO\_ASSCLU has somewhat different and more handy form of result output than the equivalent, in result, run of !KO\_HIECLU.

Assignment executed by macro !KO\_ASSCLU is equivalent in result to assignment STOP=ASS2 of macro !KO\_HIECLU, i.e., the point is about the independent assignment. New objects are assigned by !KO\_ASSCLU to clusters all at once and independently of each other; whereby the only influence on the appointment of each new object to this or that preexistent cluster is the effect of the proximity between this single object and the preexistent clusters in their initial state in which they entered the analysis.

Unlike !KO\_HIECLU macro, this macro accepts proximity matrix between individual objects only, not between groups of objects (i.e., matrix row/column cannot represent several objects, be they even identical).

EXAMPLE 1. Enlist new objects to old clusters or groups.

!KO\_assclu matrix= var1 to var630 /cluvar= oldclus /id= objid /method= SINGLE.

* VAR1 through VAR630 is the matrix of distances between objects, some objects old (already partitioned into groups or clusters) and some new (to be distributed among those groups/clusters – assign to the closest). Variable OLDCLUS specifies that cluster structure; its codes are the cluster codes, while new objects have no codes yet, they are missing.
* The assignment will be nearest neighbour method. The macro will report which cluster every new object will have been assigned to. The object identifier variable is necessary.

***Matrix structure***

Same as in macro [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK). Proximities may be, as are there, distances or similarities. Those must be proximities between individual objects.

***Subcommands***

**MATRIX, SEQ**

These subcommands are the same as in macro [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK).

**ID**

Required subcommand naming (up to 8 bytes) the numeric identifier variable for objects, i.e. rows/columns of the input matrix. All the variable values must be valid (cf. !KO\_HIECLU where it is not necessary). The variable must have no value: *-999*. Duplicating values are allowed (but barely may ever be needed).

**CLUVAR**

Required subcommand to specify the preexistent cluster (group) structure of the objects – numeric nonconstant variable (name up to 8 bytes). Its values are cluster/group codes. Any cluster may consist even of one object. *Missings* are allowed – they are treated as new objects that need to be assigned to the preexistent clusters/groups. The variable must have no value: *-999*.

**SQUARE**

This subcommand is only for distances (dissimilarities). Some methods in the macro (CENTROID, MEDIAN, WARD, MNSSQ, MIVAR, MNVAR) demand the input distances to be squared. If your matrix contains nonsquared distances you may request to square them at input: SQUARE=YES. This option is ignored with METHOD=CROSSPC.

**METHOD**

Specify method defining proximity between clusters (at assembly) as well as between clusters and new objects (at assignment). These are same methods existing in macros !KO\_HIECLU and !KO\_HIECLUEX, only here they are applied not stepwisely but in lump, which in no way change their meaning.

SINGLE - method of **single linkage** or **nearest neighbour**. Proximity between two clusters is the proximity between their two closest objects. This value is one of values of the input matrix.

COMPLETE - method of **complete linkage** or **farthest neighbour**. Proximity between two clusters is the proximity between their two most distant objects. This value is one of values of the input matrix.

BAVERAGE - method of **between-group average linkage** (UPGMA). Proximity between two clusters is the arithmetic mean of all the proximities between the objects of one, on one side, and the objects of the other, on the other side.

WAVERAGE - method of **within-group average linkage** (MNDIS). Proximity between two clusters is the arithmetic mean of all the proximities in their joint cluster.

CENTROID - **centroid** method (UPGMC). Proximity between two clusters is the proximity between their geometric centroids: squared euclidean distance between those.

WARD - **Ward**’s method, or minimal **increase of sum-of-squares** (MISSQ). Proximity between two clusters is the magnitude by which the summed square in their joint cluster will be greater than the combined summed square in these two clusters: SS12-(SS1+SS2). (Between two singleton objects this quantity = squared euclidean distance / 2.)

MNSSQ - method of minimal **sum-of-squares**. Proximity between two clusters is the summed square in their joint cluster: SS12. (Between two singleton objects this quantity = squared euclidean distance / 2.)

MIVAR - method of minimal **increase of variance**. Proximity between two clusters is the magnitude by which the mean square in their joint cluster will be greater than the weightedly (by the number of objects) averaged mean square in these two clusters: MS12-(n1MS1+n2MS2)/(n1+n2) = [SS12-(SS1+SS2)]/(n1+n2). (Between two singleton objects this quantity = squared euclidean distance / 4.)

MNVAR - method of minimal **variance**. Proximity between two clusters is the mean square in their joint cluster: MS12 = SS12/(n1+n2). (Between two singleton objects this quantity = squared euclidean distance / 4.).

HAUSDORFF - proximity between two clusters is **Hausdorff distance** between them. It is defined as follows. Each object of the first cluster finds for itself the nearest neighbour in the second cluster, and from these (minimal) distances one maximal is taken. And oppositely: each object of the second cluster finds its nearest neighbour in the first cluster, and from these (minimal) distances one maximal is taken. Of the two thus found distances the greater is chosen.

MHAUSDORFF - proximity between two clusters is **Modified Hausdorff distance** between them. It is defined as follows. Each object of the first cluster finds for itself the nearest neighbour in the second cluster, and of these (minimal) distances the arithmetic average is taken. And oppositely: each object of the second cluster finds its nearest neighbour in the first cluster, and of these (minimal) distances the arithmetic average is taken. Of the two thus obtained values the greater is chosen.

CROSSPC - proximity between two clusters is **Point-centroid cross-distance** between them. It is the sum of distances from objects of one cluster to the centroid of the second cluster, plus the sum of distances from objects of the second one to the centroid of the first, and divide by the combined number of objects in the two clusters.

Methods SINGLE, COMPLETE, BAVERAGE, WAVERAGE, HAUSDORFF, MHAUSDORFF permit any proximity measures (any similarities or distances). Methods CENTROID, WARD, MNSSQ, MIVAR, MNVAR, CROSSPC require distances; and fully correct will be to use only euclidean distances, because these methods compute centroids in euclidean space. Of these 6 methods, method CROSSPC expects nonsquared euclidean distances, and the other 5 expect squared euclidean distances (you can square distances by s/c SQUARE).

Note that methods EQBAVER and MEDIAN which are present in !KO\_HIECLU, are absent here. Because they are possible only as stepwise. You can assemble clusters by these methods using macro !KO\_HIECLU.

**SAVE**

Subcommand saving, as new unnamed dataset, the updated cluster membership variable CLUVAR, in which “new objects” (former missings, if any) were assigned to the clusters. Specify YES (save) or NO (don’t save, default). Assigned objects will be placed in the tail of the variable. For the assigned objects, there will be added columns *PROX* (the proximity to the cluster of assignment), *OBOB* (the proximity to the nearest other new object), *CLUCLU* (the proximity of the cluster of assignment to its nearest other cluster – it’s the proximity being *prior* acts of assignment). By comparing the three values you might decide that it is better not to enlist the new object to the old cluster but to form a new cluster of the pair of new objects, or, maybe, to merge the two old clusters.

Identifier variable (see s/c ID) is also saved to the dataset, as well as variable *VARNAME\_* borrowed from the input matrix.

EXAMPLE 2.

!KO\_assclu matrix= var1 to var630 /cluvar= oldclus /id= objid /method= SINGLE /save= YES.

!KO\_hieclu matrix= var1 to var630 /preclu= oldclus /id= objid /method= SINGLE /stop= ASS2 /save= ATSTOP.

* These are two equivalent commands. They differ only by the format of the data being saved to the new dataset. !KO\_ASSCLU saved cluster membership variable *OLDCLUS* where new objects are assigned to the old clusters, and the new objects are moved to the tail of the variable. !KO\_HIECLU saved cluster membership variables *CLUSTOP#* and *CLUSTOP*. *CLUSTOP* is equivalent to *OLDCLUS*, but it uses different codes (sequential positive integers) and the new objects are not moved to the tail.

**MSAVE1, MSAVE2**

Subcommands by which you can (if need) save the assembled proximity matrix between the clusters (groups) as external file (specify, for each subcommand, own path/name in quotes or apostrophes). MSAVE1 saves the matrix after the assembly of the preexistent clusters and prior assigning new objects to them (*intermediate* matrix). New objects in this matrix are represented by their rows/columns and go last there. MSAVE2 saves the matrix after the new objects are included in the clusters (*final* matrix). So, there is no rows/columns corresponding to new objects in it. You may specify both subcommands, one any, or none of the two. If there were no new objects (missings in the variable specified in CLUVAR) the intermediate matrix will be identical to the final.

MSAVE1 corresponds to /STOP=ASS1 /MSAVE= … of macro !KO\_HIECLU, with the difference that new objects will go last in the matrix. MSAVE2 is equivalent to /STOP=ASS2 /MSAVE=… of macro !KO\_HIECLU.

You can later use the matrix in other kinds of analysis, for example multidimensional scaling. And also to input it in macro !KO\_HIECLU for subsequent clustering of the assembled clusters. You may not, however, use the assembled matrix as input to !KO\_ASSCLU because !KO\_ASSCLU accepts only matrix between *objects*, not groups of objects.

Values in the saved matrix are the proximities between clusters as they were computed by the method you used (s/c METHOD). With methods WARD and MNSSQ output proximities are multiplied by 2, and with methods MIVAR and MNVAR multiplied by 4: so is required for compatibility with the input proximities (squared euclidean distances).

There will be column *METHOD\_* with the matrix informing about the method used, a memo (you may delete it). Also, there will be column *N\_* with the within-cluster frequencies of objects. The identifier of rows/columns is the clustering variable named in s/c CLUVAR. If in the input CLUVAR variable there were missings (“new objects”) then in the intermediate matrix (MSAVE1) there will be also the variable specified in s/c ID and identifying the new objects. Column *VARNAME\_* in the saved matrix – is *not* that column with the same name from the input matrix but is newly created column consisting of names *var1 var2 var3*, etc.

Attention. In order to correctly use the saved matrix in native SPSS Statistics procedures such as CLUSTER command, you will need to delete columns *METHOD\_* and *N\_*, and also to prescribe a label to the value ‘PROX’ of column *ROWTYPE\_*. The label must begin with the word DISSIMILARITY if the proximities in the matrix are dissimilarities, and with the word SIMILARITY, if the proximities in the matrix are similarities.

EXAMPLE 3. Assembly of matrix with new objects assignment.

!KO\_assclu matrix= var1 to var150 /cluvar= pre /id= objid /method= BAVERAGE

/msave1= 'd:\exercise\ass\_mx1.sav' /msave2= 'd:\exercise\ass\_mx2.sav'.

* The macro assembles preexistent clusters (defined by variable PRE) and, if there are new objects (missings in PRE), attaches them to the clusters (to the closest ones). Method is average linkage. The proximity matrix upon the assembly but prior the assignment is saved by s/c MSAVE1, and the final matrix – by s/c MSAVE2.

!KO\_hieclu matrix= var1 to var150 /preclu= pre /id= objid /method= BAVERAGE /stop= ASS2

/msave= 'd:\exercise\hie\_mx.sav'.

* Shown how the same task is done by !KO\_HIECLU. STOP=ASS2 performs the assembly and then the same assignment as macro !KO\_ASSCLU does. The saved matrix is identical to the one saved by s/c MSAVE2. The only difference is about the column attached to the matrix as the identifier. There it was *PRE*, and here it is *OBJID*. But you could specify ID=PRE in macro !KO\_HIECLU.

!KO\_hieclu matrix= var1 to var150 /preclu= pre /id= objid /method= BAVERAGE /stop= ASS1

/msave= 'd:\exercise\hie\_mx.sav'.

* This run of !KO\_HIECLU assemblies the matrix without assigning new objects to it. The saved matrix is identical to the one saved by s/c MSAVE1. However, one difference is about the column attached as the identifier. There it was *PRE*, and here it is *OBJID*. But you could specify ID=PRE in macro !KO\_HIECLU. The other difference is about the order of rows/columns: MSAVE1 of !KO\_ASSCLU moved new objects to the tail of the matrix.

**PRINT**

Printout to Viewer window:

FULL - (default/unspecifying) detailed report: summary, information on identification/order of rows/columns in the analyzed matrix, report on assignment of new objects.

NOID - don’t show information on identification/order of rows/columns in the matrix entered the analysis.

SUMM - show only summary.

# 

***Special regimes***

The macro does not obey weighting (however, it doesn’t take in the procedure cases with missing and nonpositive weights). It is not suited for the split state of the dataset (SPLIT FILE). The macro obeys commands selecting cases (SELECT IF, FILTER, USE), including those standing under TEMPORARY command.

# MACRO !KO\_POINTCLUD: DISTANCES BETWEEN OBJECTS AND CLUSTERS

Version 2, Jan 2022 (Version 1, May 2016). Tested on SPSS Statistics 20, 22, 26.

!KO\_pointclud matrix= *var1 to var150* /\*Columns constituting the matrix body, may use “to”

/seq= /\*Sequence of rows/columns must be as of the taken columns (COL, default) or

/\*as of the taken rows (ROW)

/id= *id* /\*Numeric identifier variable of objects

/groups= *clu5* /\*Groups: either one categorical variable or a set of binary variables;

/\*may use “to”

/square= /\*At input, square the distances (YES) or don’t do (NO, default)

/method= AVER /\*Method (distance from an object to what): NEAR, FARTH, AVER, CENTROID,

/\*MEDOID

/self= /\*For AVER, CENTROID: account for the object in its group: IN (default)

/\*or don’t (OUT)

/saveid= /\*For NEAR, FARTH, MEDOID: return id of these terminal objects:

/\*'path/file' or declared 'dataset'

/print= /\*Printout: full (FULL, default) or w/o info about ID of rows/columns in

/\*matrix (NOID).

Minimal specification MATRIX, ID, GROUPS, METHOD.

Computes distances between objects and existing (user defined) clusters (or groups) of objects. Output is a new unnamed working dataset showing the distance from each object to each cluster/group of objects.

The macro takes a distance (dissimilarity) matrix between individual objects, and variables defining some preexistent cluster (grouping) structure among them. The groups may intersect or be disjoint by the composition of objects.

EXAMPLE 1.

!KO\_pointclud matrix= var1 to var630 /groups= oldclus /id= objid /method= NEAR.

* VAR1 through VAR630 is the matrix of distances between objects. Variable OLDCLUS specifies some grouping among the objects, for instance, it is a partition of objects into clusters. OBJID is the identifier variable of objects.
* The macro will return distances between the objects of the input dataset and the clusters. The distance computed is to a nearest neighbour (METHOD=NEAR). It is the distance between the object and the closest to it object of the cluster being considered.

filter by selvar.

!KO\_pointclud matrix= var1 to var630 /groups= oldclus /id= objid /method= NEAR.

* In this case, the same as above is being done, but the analysis is comprised only of the dataset cases filtered in.

***Matrix structure***

Same as in macro [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK). But these can only be distances/dissimilarities, not similarities. Those must be distances between individual objects.

***Subcommands***

**MATRIX, SEQ**

These subcommands are the same as in macro [!KO\_HIECLU](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK).

**ID**

Required subcommand naming (up to 8 bytes) the numeric identifier variable for objects, i.e., rows/columns of the input matrix. All the variable values must be valid. Duplicating values are allowed (but barely may ever be needed).

**GROUPS**

Subcommand to specify the preexistent cluster (group) structure of the objects. You can specify it by one of two modes:

* One numeric categorical variable. Its values are group codes. As many there are values (must be two minimum) – so many there are groups. A group may consist even of one object.
* Two or more numeric binary (with codes 1 and 0) variables, name-by-name and/or via “to”. Each such variable corresponds to a group, code 1 signifying belonging to it. An object may belong to a single group, to multiple groups, or to none of the groups. In each variable, there must be units (i.e., a group cannot be empty). As a specific case, a variable may have no zeros (then this group presents by itself all the analysis objects taken together).

Variable names specified in GROUPS – up to 8 bytes. Missings are prohibited (at least in the dataset cases taken into the analysis).

EXAMPLE 2.

!KO\_pointclud matrix= var1 to var120 var161 /groups= total gr1 to gr4 /id= objid /method= AVER

/self= OUT.

* VAR1 through VAR120 and var161 are user-selected columns of the distance matrix between objects. Only these selected objects will enter the analysis. Binary (1 vs 0) variables TOTAL and GR1 through GR4 set the groupings. Variable TOTAL contains no zeros, therefore corresponds to the entire sample of cases (in this instance all the objects taken to the analysis as one group).
* METHOD=AVER orders to calculate the average distance from each object to the objects belonging to each of the groups. If the object itself is a member of the group it confronts with, then it does not participate in the average (its zero distance to itself is withdrawn from averaging), because SELF=OUT.

**SQUARE**

This subcommand is to square the distances at input, if you need so: SQUARE=YES. Computation of distances to centroid (METHOD=CENTROID), for example, requires to input squared distances.

**METHOD**

Indicate which kind of distance between an object and a group you want. That distance will be obtained between each individual object *i* (dataset case taken in the analysis) and each group of objects (also taken in the analysis).

NEAR - distance to **nearest neighbour**. The distance from object *i* to the closest to it object belonging to the group. This value is one of values of the input matrix.

FARTH - distance to **farthest neighbour**. The distance from object *i* to the most distant from it object belonging to the group. This value is one of values of the input matrix.

AVER - compute **average** distance from object *i* to the objects of the group.

CENTROID - compute distance from object *i* to the **centroid** (arithmetic mean location) of the group.

MEDOID - distance from object *i* to the **medoid** of the group. This value is one of values of the input matrix. Medoid is an object in a group whose sum of distances from it to all the other objects therein is minimal. (If there appeared more than one medoid in a group, the macro will choose among them the object with greater ordinal index.)

All the methods except CENTROID allow any dissimilarities/distances at input. With CENTROID method, only euclidean distances would be correct to use; other metric distances could be input, maybe, with heuristic purpose. For technical reason, CENTROID method needs the input distances be square. You can square distances at input by SQUARE subcommand.

*Self-participation*. When object *i* belongs to the group to which its distance must be found out, its participation in determining the distance is as follows. With methods NEAR and FARTH, object *i* does not participate, i.e., it cannot be its own neighbour, except in the case when it is the only member of the group (and then the distance will be 0). With method MEDOID, object *i* participates in the determining of medoid and can itself occur the medoid. With methods AVER and CENTROID, you can choose participation or non-participation – see s/c SELF.

**SELF**

This subcommand acts only under METHOD= AVER or CENTROID. When the object whose distance to a group is to compute belongs itself to the group, you can select whether it will participate (SELF=IN, also default/unspecifying) or won’t participate (SELF=OUT) in the calculation of that distance. The second choice implies temporary withdrawal of the object from its group (leave-one-out action) at the computation of the distance to the group. Say, if METHOD=AVER and SELF=OUT, the zero distance of the object to itself does not enter the averaging of distances, and with SELF=IN, it enters.

If the object is the sole member of the group, it is not withdrawn, so the distance of it to the group = 0.

**SAVEID**

This subcommand is ignored with METHOD= AVER or CENTROID. Under METHOD= NEAR, FARTH, or MEDOID, obtained distances are distances to specific, real objects within groups. If you need to identify these objects – to know their codes by the ID variable, use SAVEID to obtain a dataset structurally similar to the main output dataset, and containing identification codes of objects in place of the corresponding distances. Say, with method NEAR, you’ll get the dataset showing which namely objects have appeared to be those nearest neighbours the distances to which the macro has established.

With methods NEAR and FARTH, if more than one object of the group appear to be the nearest (or farthest, respectively) neighbour of the object *i*, the subcommand will select among them, to show, the object with greater ordinal index.

In SAVEID, specify path/name of .SAV file in quotes or apostrophes – to save the file on disc, or indicate name of the dataset previously declared by you (optionally may take the name in quotes/apostrophes) – to output to a new dataset.

EXAMPLE 3.

dataset declare medoids.

!KO\_pointclud matrix= var1 to var630 /groups= oldclus /id= objid /method= MEDOID /saveid= medoids.

* The distances to group medoids will be in the new working, unnamed dataset, while the medoids themselves (their identification by OBJID variable values) will appear in the dataset MEDOIDS, previously declared. Since a medoid is always a fixed object in a group, variables in MEDOIDS dataset showing medoids will be constants.

**PRINT**

Printout to Viewer window:

FULL - (default/ unspecifying) standard report.

NOID - don’t show information on identification/order of rows/columns in the in the matrix entered the analysis.

***Special regimes***

The macro does not obey weighting (however, it doesn’t take in the procedure cases with missing and nonpositive weights). It is not suited for the split state of the dataset (SPLIT FILE). The macro obeys commands selecting cases (SELECT IF, FILTER, USE), including those standing under TEMPORARY command.

# MACRO !KO\_DENDRO: DENDROGRAM

Version 3, Sep 2019 (Version 1, Mar 2014). Tested on SPSS Statistics 20, 22, 25.

!KO\_dendro sched= \* /\*Agglomeration schedule: external sav file (quoted/apostrophed) or star

/\*(if the file is open as the working dataset)

/revers= /\*Allow reverses (YES, default) or not (NO)

/label= *'d:\exercise\distmx.sav' strid* /\*Optional: use these labels for objects

/\*(file/dataset name and the variable name that sets labels

/bycolor= *'d:\exercise\objects.sav' group* /\*Optional: give colour to objects

/\*(file/dataset name and the variable name that sets colours)

/aspect= /\*Appearance of the tree: FENCE (default) or GRAPE

/size= /\*Optional, graph size: two numbers in pixels, width and height

/format= /\*Format for object labels.

Minimal specification SCHED.

The macro builds, in output Viewer, dendrogram according to an agglomeration schedule. Agglomeration schedule is returned by macro !KO\_HIECLU or !KO\_HIECLUEX and it is same to what SPSS command CLUSTER produces. !KO\_DENDRO was made a separate macro because a user possibly might want to compose their own agglomeration schedule.

If subcommands LABEL and BYCOLOR were not specified, the macro also outputs, as a new unnamed dataset, two variables: object code (*ID*) and its level of merge on the dendrogram (*MERGELEV*). Objects go top to down in the sequence of the dendrogram. This dataset is of potential value for a user. If LABEL or BYCOLOR were specified, an empty dataset is output.

EXAMPLE 1.

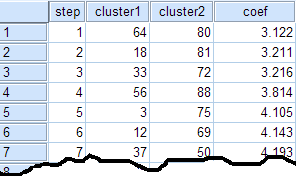
!KO\_hieclu matrix= var1 to var80 /method= COMPLETE /sched= \*.

!KO\_dendro sched= \*.

* Hierarchical clustering saves agglomeration schedule to an unnamed dataset.
* !KO\_DENDRO builds dendrogram.

***Agglomeration schedule structure***

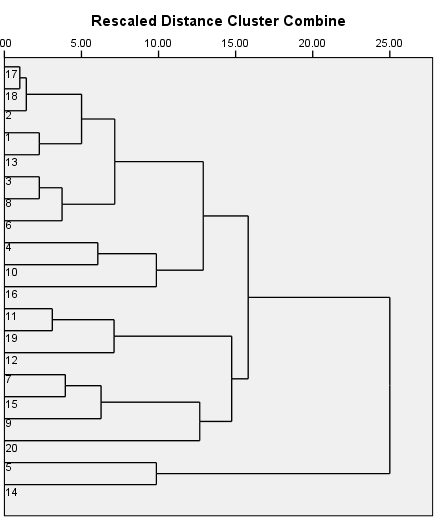
This SAV file must have numeric variables *CLUSTER1, CLUSTER2, COEF*. CLUSTER1 and CLUSTER2 contain cluster identifying codes; they will represent object codes (labels) on the dendrogram. COEF is the colligation coefficient – similarity or distance between the merging clusters cluster1 and cluster2 or the corresponding cumulative coefficient (if the coefficient is cumulated then it must be distances, not similarities; see s/c CUMUL in macro !KO\_HIECLU for more info).



Rows of the schedule correspond to the steps of the agglomeration. There is the *rule*: when two clusters, cluster1=X and cluster2=Y merge, the combined cluster receives code X, i.e. code of cluster1, with which it participates hereafter. (That means, consequently, that in variable *CLUSTER2* codes cannot repeat, while in *CLUSTER1* they can.)

***Dendrogram appearance***

On the dendrogram, the colligation coefficients ­– the distances – are rescaled to range 1-25, as SPSS command CLUSTER does it; similarities get converted to distances by simply reverting their sign before the rescaling. Dendrogram look:



Object labels on a dendrogram are cluster codes – the values of *CLUSTER1* and *CLUSTER2* columns of the agglomeration schedule. You can request for them other labels, including string labels, by subcommand LABEL. Note that visually nice layout of the labels may need manual editing of the built chart, for what you must open it to edit, by double-clicking it.

***Subcommands***

**SCHED**

Specify path to the agglomeration schedule. Specify asterisk if it is the working dataset. Or specify path/name in quotes/apostrophes if it is an external file. Don’t specify dataset name in SPSS Statistics vers. earlier than 26.

**REVERS**

Reverse is the phenomenon when a cluster unites with another one on a lower level (magnitude of distance) than the level it itself emerged on. Some methods of hierarchical clustering (centroid, median, minimal increase of variance) can sometimes give reverses on their dendrogram. If you want to suppress displaying reverses on the dendrogram specify REVERS=NO. There will be no reverses then, but the dendrogram at the areas of reverses will reproduce the agglomeration schedule not exactly.

**LABEL**

Optional subcommand by which you can subscribe objects with other labels than the default numeric codes borrowed from the agglomeration schedule. Specify external SAV file (in quotes/apostrophes) or the name of the open dataset, and, after it – the name of a variable in it containing labels for objects. The variable can be numeric or string. The labels are arbitrary; including so that you can sign different objects with the same labels.

In the mentioned file/dataset there *must be* numeric variable *ID*, which values should be *nonduplicating*, *ascendingly sorted* object codes that were used in variables *CLUSTER1* and *CLUSTER2* of the agglomeration schedule. (Complete concurrence isn’t necessary, generally: for example, in variable *ID* there may be more different codes than there are in the agglomeration schedule.)

**BYCOLOR**

Optional subcommand by which you can show objects in different colour – for example, colour differently objects belonging to different preset groups. Specify external SAV file (in quotes/apostrophes) or the name of the open dataset, and, after it – the name of a grouping variable in it. Objects belonging to one group will receive same colour on the dendrogram.

In the mentioned file/dataset there *must be* numeric variable *ID*, which values should be *nonduplicating*, *ascendingly sorted* object codes that were used in variables *CLUSTER1* and *CLUSTER2* of the agglomeration schedule. (Complete concurrence isn’t necessary, generally: for example, in variable *ID* there may be more different codes than there are in the agglomeration schedule.)

EXAMPLE 2.

!KO\_dendro sched= \* /label= 'd:\exercise\matrix.sav' caselab

/bycolor= 'd:\exercise\respondents.sav' agegroup.

* There being built a dendrogram for the agglomeration schedule open as the working dataset. Requested is to show respondents belonging to different age groups by different colour: the file is used where there exists the sorted list of respondents *ID* (same codes as in agglomeration schedule) and variable *AGEGROUP*. To sign respondents with labels from variable *CASELAB* there used another file in this example, but there also sorted variable *ID* is found.

**ASPECT**

Dendrogram look. By default and by ASPECT=FENCE, the tree looks like a fence, objects are “ground” on one line. With ASPECT=GRAPE objects “pend” and the tree looks like a grape.

**SIZE**

You can modify the default size of the graph as you like. Specify two numbers – width and height, in pixels.

**FORMAT**

If labels to subscribe objects are numeric codes, these are typically integers, and then the subcommand is not needed. If your numeric codes are not integers, specify the format with the needed amount of decimal places in order to display them all on the dendrogram. For example: FORMAT = f8.3.

***Special regimes***

The macro is not meant for special regimes (weighting, splitting etc.).

# MACRO !KO\_KMINI: INITIAL CENTRES FOR K-MEANS CLUSTERING

Version 1, Feb 2017. Tested on SPSS Statistics 17, 20, 22.

!KO\_kmini vars= *v1 to v6* /\*Variables (name-by-name and/or via to)

/id= /\*Optional: case identifier variable

/k= *5* /\*Produce this number of cluster centres

/method= GREP /\*Method: RGC, RP, RUNFP, SIMFP, KMPP, GREP, WARD (see)

/outfile= \* /\*File path/name to save the centres, or asterisk

/print= YES /\*Printout the centres: YES (default) or NO.

Minimal specification VARS, K, METHOD.

The macro creates/selects, by various methods, initial centres for subsequent cluster analysis by k-means method.

EXAMPLE 1.

!KO\_kmini vars= v1 to v6 /k= 5 /method= RUNFP /outfile= 'd:\exercise\centers.sav'.

quick cluster v1 to v6 /criteria= cluster(5) mxiter(10) converge(0) /method= kmeans(noupdate)

/file= 'd:\exercise\centers.sav'.

* !KO\_KMINI collects 5 cases by RUNFP method to be the starting cluster centres, and saves them as file.
* QUICK CLUSTER takes that file and performs k-means clustering (5 clusters).

***Subcommands***

**VARS**

Data which cases you will be clustering by k-means method. Specify name-by-name and/or via “to” scale variables (names up to 8 bytes) constituting these data. The macro excludes missings listwise: if at least in one VARS variables or in ID (if specified) the case is missing, that case is omitted from the procedure.

**ID**

Optionally specify one case identifying variable, if you want s/c PRINT to show which cases were selected to be the centres (in methods that make real cases of the dataset the initial centres).

**K**

How many centres you need (i.e., how many clusters you are going to extract in a cluster analysis). Specify positive integer greater than 1 and lesser than the number of valid cases in data VARS.

**METHOD**

Method to create or select initial cluster centres. Choose:

RGC - *centroids of random subsamples*. The data are partitioned randomly by *k* nonoverlapping, by membership, groups, and centroids of these groups are appointed to be the initial centres. Thus, centres are calculated, not selected from the existent dataset cases. This method yields centres that lie close to each other and to the general centroid of the data.

RP - *randomly selected points*. *k* distinct cases of the data are randomly selected to be the initial centres.

RUNFP - *farthest points (running selection)*. First *k* cases are taken as centres and then during the run through the rest of the cases of the dataset there progressively replacements among the centres are done; the aim of the replacements is to obtain in the end *k* points most distant from each other in the variable space. These points (cases) occupying peripheral positions in the data cloud are the produced initial centres. (The method is used as the default in SPSS command QUICK CLUSTER. See details in SPSS Algorithms.)

SIMFP - *farthest points (simple selection)*. The first centre is selected as a random case from the dataset. The 2nd centre is selected as the case maximally distant from that centre. The 3rd centre is selected as the case maximally distant from those two (from the nearest of the two), - and so on.

KMPP - *random* *farthest points, or k-means++*. The first centre is selected as a random case from the dataset. The 2nd centre is selected also randomly, but the probability of selection of a case is proportional to the distance (square euclidean) of it to that (1st) centre. The 3rd centre is selected also randomly with the probability of selection proportional to the distance of a case to the nearest of those two centres, - and so on. (Arthur, D., Vassilvitskii, S. K-means++: the advantages of careful seeding. // Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms. 2007., 1027–1035.)

GREP - *group representative points*. The method idea – to collect as centres *k* most representative, “deputy” cases. The 1st centre is taken as the case closest to the general data cenroid. Then the rest of the centres are selected from the data points in such a way that each point is considered as to whether it is closer (and how much, in terms of squared euclidean distance) to a set of points than each one of the latter is to any of the already existing centres. I.e., each point is examined as a candidate to represent some group of points not yet well enough represented by the centres already collected. Point most representative in this respect is selected as the next centre. (Kaufman, L. Rousseeuw, P.J. Finding groups in data: an introduction to cluster analysis., 1990. See also: Pena, J.M. et al. An empirical comparison of four initialization methods for the K-means algorithm // Pattern Recognition Lett. 20 (10), 1999, 1027-1040.)

WARD - *hierarchical clustering by Ward method*. Cluster analysis by this method is performed and *k* clusters are extracted; their centroids are considered the *k* initial centres. Thus, centres are calculated, not selected from the existent dataset cases.

Methods RGC, RP, SIMFP, KMPP depend on random numbers and may change their result from run to run. Use menu Transform – Random Number Generators or corresponding syntax to manage random number seed.

Method RUNFP may be sensitive to case order in the dataset; but method GREP is not (apart from occasions when there are many identical cases, ties, in the data). Method GREP may fail to collect all k centres if k is large relative the number of cases in the data (n), especially when k>n/2. The macro will inform if the data do not allow that method to collect k centres. Method GREP is the slowest one, it computes matrix of distances between all cases, therefore it won’t suit if there are many tens of thousands or millions of cases.

Method WARD may be sensitive to case order in the dataset. The method won’t suit if there are many tens of thousands or millions of cases because it computes matrix of distances between all data cases.

If you have too many cases and you want to use methods such as WARD or GREP, just use a random subsample of cases (with commands SAMPLE or USE, for instance).

**OUTFILE**

Specify in quotes/apostrophes path/name of a SAV file or name of a previously declared dataset, to save the initial centres. You may also specify asterisk, then the centres will be output in a new unnamed dataset. Don’t specify name of a declared dataset when METHOD=WARD.

If you don’t specify the subcommand, the centres won’t be saved, but can be displayed (s/c PRINT).

**PRINT**

Print out initial centres into output Viewer: PRINT=YES (also default). By PRINT=NO, no printing out.

***Special regimes***

The macro does not obey weighting (however, it doesn’t take in the procedure cases with missing and nonpositive weights). It is not suited for the split state of the dataset (SPLIT FILE). The macro obeys commands selecting cases (SELECT IF, FILTER, USE), including those standing under TEMPORARY command.