***Internal clustering criteria***

SPSS macros by Kirill Orlov

kior@akado.ru, ttnphns@gmail.com

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

All rights reserved

*Internal clustering criteria.* Computation of indices, such as Calinski–Harabasz, Davies–Bouldin, Cubic clustering criterion, Ratkowsky–Lance, C-Index, correlation, Gamma statistic, Dunn (several types), Silhouette statistic (several types), AIC, BIC, and other indices helpful in choosing the better clustering partition, in particular, to decide how many clusters one should extract in a cluster analysis.

*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.

The following internal clustering criteria are offered: Calinski–Harabasz, SSw “elbow”, Davies–Bouldin, Cubic Clustering Criterion, Log SS Ratio, Log Det Ratio, PBM, Ratkowsky–Lance, AIC, BIC, Point-biserial correlation, Goodman-Kruskal Gamma, C-Index, Dunn, Generalized Dunn, McClain–Rao, Silhouette index, Simplified (deviation) Silhouette index.

* Clustering criteria based on ideology of analysis-of-variance in euclidean space. Based on ratios of sums-of-squares of deviations within and between clusters: B/W, B/T or W/T.
  + - **Calinski–Harabasz** is a multivariate analogue of Fisher’s F statistic. It recognizes well any convex clusters. [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS) takes variables as input, and [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ) takes matrix of distances. These macros also output **SSw “elbow” criterion**.
    - **Davies–Bouldin** is similar to the former but without its tendency towards approximately same-sized, by the number of objects inside, clusters; Davies–Bouldinrather prefers clusters equally-distanced from each other. [!KO\_DAVBOULV](#_MACRO_!DAVBOULV:_DAVIES-BOULDIN_1) takes variables as input, and [!KO\_DAVBOULM](#_MACRO_!KO_DAVBOULM:_DAVIES-BOULDIN) takes matrix of distances.
    - **Cubic clustering criterion** is like Calinski–Harabasz. It was (questionably) standardized for comparing results obtained on different data. Prefers spherical clusters. [!KO\_CCCRITV](#_MACRO_!KO_CCCRITV:_CUBIC) takes variables as input, and [!KO\_CCCRITM](#_MACRO_!KO_CCCRITM:_CUBIC) takes matrix of distances.
    - **Log SS Ratio** is akin to Calinski–Harabasz, but instead of normalizing *B/W* it uses logarithm. This criterion is output by macros [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS) and [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ).
    - **Log Det Ratio** – logarithm of the inverted Wilks’ lambda; it is a MANOVA criterion considering volumetric property of the data cloud. Computed by [!KO\_CCCRITV](#_MACRO_!RATLAN:_RATKOWSKY–LANCE).
  + Clustering criteria professing univariate approach: analysis goes by each variable. These are fixed attributes: the data are not considered as lying in space where they might be arbitrarily rotated.
    - **Ratkowsky–Lance** is designed for scale features (where it is based on the analysis-of-variance idea) as well as for categorical features (based on the chi-square statistic idea). [!KO\_RATLAN](#_MACRO_!KO_RATLAN:_RATKOWSKY–LANCE) takes variables as input.
    - **AIC** and **BIC** clustering criteria also allow for both scale and categorical attributes. These indices are linked to the idea of variational entropy. They put penalty for excess of clusters and thus make it possible to prefer groundedly a parsimonious (few clusters) solution. [!KO\_AICBIC](#_MACRO_!KO_AICBIC:_INFORMATION) takes variables as input.
  + Clustering criteria based on ideology of “cophenetic” correlation (correlation between likeness of objects and their falling into same cluster).
* **Point-biserial correlation** is usual Pearson r. [!KO\_RPBCLU](#_MACRO_!KO_RPBCLU:_POINT-BISERIAL) takes proximity matrix as input.
* Goodman–Kruskal **Gamma** is nonparametric, monotonic correlation. [!KO\_GAMMACLU](#_MACRO_!GAMMACLU:_GAMMA_1) takes proximity matrix as input.
* **C-Index** assesses how much close the cluster partition is to (unreachable) ideal one in the current setting. This criterion is equivalent to the rescaled Pearson r. [!KO\_CINDEX](#_MACRO_!KO_CINDEX:_C-INDEX) takes proximity matrix as input.
* Other criteria:
* **Dunn** seeks for cluster solution with maximally demarcated, separated clusters – if possible, of approximately same physical size (diameter). The macro computes different versions of the criterion. [!KO\_DUNN](#_MACRO_!KO_DUNN:_DUNN) takes proximity matrix as input.
* **McClain–Rao** is the ratio of the average same-cluster distance to the average between-cluster distance among objects. [!KO\_RPBCLU](#_MACRO_!RPBCLU:_POINT-BISERIAL) takes proximity matrix as input.
* **PBM** is eclectic criterion taking into account sums of deviations (not their squares) from centroids and separation between centroids. [!KO\_DAVBOULV](#_MACRO_!DAVBOULV:_DAVIES-BOULDIN_1) takes variables as input, and [!KO\_DAVBOULM](#_МАКРОС_!COPHEN:_КОФЕНЕТИЧЕСКАЯ) – proximity matrix.
* **Silhouette statistic** (the macro computes several versions of) is able to assess quality of clusterization of each separate object, not just of the entire cluster solution. The criterion measures justifiedness of putting objects in their clusters. [!KO\_SILHOU](#_MACRO_!KO_SILHOU:_SILHOUETTE) takes proximity matrix as input. [!KO\_SILDEV](#_MACRO_!KO_SILDEV:_SILHOUETTE) takes variables as input and computes one of the criterion versions.

***Internal clustering criteria***

**Table 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| criterion | basic idea | variable type  (if input is variables) | proximity type (if input is proximities) | to changing of number of objects N | to same linear transform of (scale) variables | to different linear transform of (scale) variables | to multiplying of proximities by number | to adding number to proximities | spatial biases (under euclidean distances or scale variables) | | | | solution is better when the criterion value is | comment |
| to distributional shape in (roundish) clusters: prefers bell-shaped or flat\*\* | to cluster shape: prefers spherical or ellipsoidal\*\*\* | to rotation of whole cloud about data centre: prefers… | to rotations of individual ellipsoid clusters about their centres: prefers configuration…\*\*\*\* |
| SSw | ANOVA in space\* | scale | eucl dist | sensitive | only multiplied by constant | sensitive | only multiplied by constant | sensitive | insensit | insensit | insensit | insensit | bottom elbow |  |
| Calinski–Harabasz | ANOVA in space | scale | eucl dist | sensitive | insensit | sensitive | insensit | sensitive | insensit | insensit | insensit | insensit | higher |  |
| Davies–Bouldin | ANOVA in space | scale | eucl dist | insensit | insensit | sensitive | insensit | sensitive | insensit | insensit | insensit | insensit | lower | !KO\_DAVBOULV has option of L1 deviations |
| PBM | Eclectic: deviations sum, between-centroid distances | scale | eucl dist | barely sensitive | only multiplied by constant | sensitive | sensitive | sensitive | bell | ellips | insensit | pile, ring | higher | penalty for “excess” of clusters  !KO\_DAVBOULV has option of L1 deviations |
| CCC | ANOVA in space | scale | eucl dist | sensitive | insensit | sensitive | insensit | sensitive | insensit | spheric | insensit | pile | higher |  |
| Log SS Ratio | ANOVA in space | scale | eucl dist | barely sensitive | insensit | sensitive | insensit | sensitive | insensit | insensit | insensit | insensit | top elbow |  |
| Log Det Ratio | MANOVA | scale | - | barely sensitive | insensit | insensit | - | - | insensit | ellips | insensit | pile | top elbow |  |
| Ratkowsky–Lance | univariate\* ANOVA; univariate chi-square | scale; nominal | - | insensit | insensit | insensit | - | - | insensit | ellips | equal variances along axes | chain | higher |  |
| AIC, BIC | information entropy (univariate) | scale; nominal | - | sensitive | only shifted by constant | only shifted by constant | - | - | insensit | ellips | equal variances along axes | chain; don’t like star or ring | lower | penalty for “excess” of clusters |
| Dunn | cluster closeness relative their “diameters” | - | any | (orig. version)  sensitive | - | - | insensit | sensitive | (orig. version) flat | (orig. version) insensit | (orig. version) insensit | (orig. version) pile, ring | higher | !KO\_DUNN computes several versions of the criterion |
| Point-biserial r | cophenetic correlation | - | any | insensit | - | - | insensit | insensit | flat | spheric | insensit | pile, ring | higher |  |
| Gamma | cophenetic correlation | - | any | insensit | - | - | insensit | insensit | not tested | not tested | not tested | not tested | higher |  |
| C-index | internal density relative it ideal | - | any | insensit | - | - | insensit | insensit | flat | spheric | insensit | pile, ring | lower |  |
| McClain–Rao | internal density relative between-cluster one | - | any | insensit | - | - | insensit | sensitive | bell | ellips | insensit | pile, ring  (barely) | lower, bottom elbow |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Silhouette | justifiedness of enlisting to the cluster rather than to the neighbour one | scale (DEVIAT) | any | insensit (orig. version; DEVIAT);  sensitive (other versions) | insensit (DEVIAT) | sensitive (DEVIAT) | insensit | sensitive | bell (orig. version; DEVIAT);  flat (NEAR; FARTH) | ellips (orig. version; DEVIAT; NEAR); spheric (FARTH) | insensit | pile (orig. version; DEVIAT);  pile, ring (NEAR); chain, star (FARTH) | higher | computed for each object  !KO\_SILHOU computes several versions of the criterion |

\*”in space” means accounting for the multivariate nature of data yet ignoring covariational associations. “univariate” means ignoring multivariate/spatial nature of data: the index is summed or averaged over terms computed for individual variables.

\*\*under same within-cluster variances and not changing between-centroid distances.

\*\*\*under same within-cluster variances; ellipsoid clusters are all parallel to each other.

\*\*\*\*closely located clusters: pile vs chain; ring vs star



**Internal clustering criteria** or indices exist to assess internal validity of a partition of objects into groups (clusters or other classes).

Internal validity: general idea

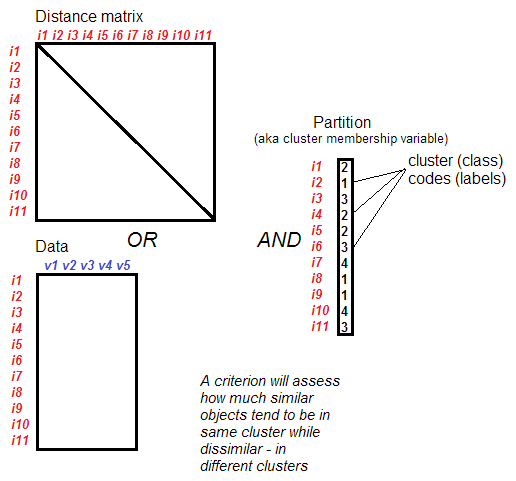
Internal validity of a partition of a set of objects is its justifiedness from the perspective of that information about the set of objects which was used by the procedure having done the partition. Consequently, internal validity answers the question, was that characteristic of the objects information accounted of “successfully” or “fully” in the act of partition. (And contrary, external validity of a partition is how well the partition corresponds to that information about the set of objects which was not used in the act of partition.)

Internal validity: operationally

Internal validity of a grouping is greater when there greater is the extent of similar objects falling in the same group while dissimilar – in different groups. In other words, same-group points must, in majority, be more similar to each other than different-group points are. Or, to formulate in terms of density: the more dense are groups inside and the less is density outside them (or the more distant groups draw apart) the higher is internal validity. Different clustering criteria, depending on their formula, *differently realize and accentuate* that intuitive principle when testing internal validity.

What input

Partition (grouping) of objects, and set – data (cases X variables) or matrix of proximities between objects. The set provides information about similarity between the objects.



Partition/grouping: what

Internal clustering criteria are applicable not only to results of clustering. Any partition in classes of *any* origin (cluster analysis, machine or manual classification), if these groups do not intersect by *membership* of elements (while spatially, the classes might intersect), can be checked for internal validity by those indices. Criteria presented in this document are meant for nonhierarchical classification, That is, groups don’t divide in subgroups in the being assessed partition.

Usage: comparing different k

Most often, internal clustering criteria are used for comparing cluster partitions with *different number of clusters k* obtained via the same method of clustering (or other method of grouping) basing on the same input set (same proximity matrix or same data). The purpose of such comparison is to choose the best k, i.e. the partition with the most valid number of clusters. In that context internal clustering criteria are also called sometimes stopping rules of clusterization. See details further.

Usage: comparing different methods

You may also compare partitions (with the same or different number k of clusters/classes) given by *different procedures/modes* (for example, different methods of cluster analysis) basing on the same input set. Generally, it’s all the same for a criterion, which way – same or not – the being compared groupings were obtained, you may even don’t know which way it were. If you are comparing different methods under the same value of k you then are selecting the “better” method (at that k).

Usage: comparing not identical sets of objects

This is possible. One should understand that for a clustering criterion objects “i” in the set – are just anonymous rows. Therefore it will be correct to compare, by the criterion value, partitions P1 and P2 which partly or completely are comprised of not the same objects. At so doing k may be one or different in the partitions. However, if P1 and P2 consist of different number of objects one may use criterion only if it is insensitive to the number of objects N.

Usage: with different variants of input (not identical features or not identical proximity matrices)

This is possible, but it is a subtle and problematic point (because it starts to go away from the idea of internal validity). Speaking now here of the direct comparison of a criterion’s values val1 and val2, where val1 was obtained from input dataset (variables or proximities) X1 and partition P1, but val2 was obtained from dataset X2 and partition P2. Specifically:

(1) Might compare partitions with the same k and gotten from the same method but differing on the used *proximity measure* between the objects. For example, one partition could be the result of clustering of matrix of euclidean distances (L2 norm), another – of matrix of Manhattan distances (L1 norm), third – of matrix of Minkovski distances with L3 norm. There is nothing formally illegitimate in such comparison – if you are ready to assume that different types of distances computed on the same data are *immediately comparable* in your case. But if they, the measures, have systematic difference for you – the difference one wants to level out (for instance, different lifting or range among values) – then do the corresponding “standardization” of the matrices before computing the clustering criterion. Considering the question of distance matrix transform it is also useful to inquire about how this or that clustering criterion reacts to transforming of matrix elements. “Universal” criteria like point-biserial correlation or C-index don’t react to addition of a constant to proximities, so the overall level of distance magnitudes in matrix is not important to them.

(2) Might also compare partitions with the same k and after the same method, but differing on the set of attributes, *variables in data*. Here one have to repeat all those same warnings about *comparability* for you values of those different sets of variables: if they are incomparable (e.g. by level or range) – take care to bring them to comparability. Also, as a rule, clustering criteria aren’t indifferent to the number of variables: it would be incorrect, in general case, to compare directly criterion value obtained on data with 2 variables with value obtained on data with 5 variables.

(3) Let’s say it separately about linear transformation of variables such as z-standardization. May one compare with a clustering criterion partitions (of the same k) of which one was received from raw data and the other was received from these same variables, only standardized? The answer to this question depends on a concrete criterion. If the criterion is insensitive to *different* linear transform of the variables, then you may.

Comparing different k: two types of criteria

Most often internal clustering criteria are used to select the optimal number of clusters *k*. (All those cluster partitions with different k must be obtained by you and be present in the dataset as the cluster membership variables; that is, a criterion assesses already existing, done partitions.) One looks at a plot where by X axis there go solutions with different number of clusters in ascending or descending order, – for example, k from 2 to 20, and by Y axis there deposits the index magnitude.

There are *extremum* criteria and *elbow* criteria. With extremum criteria, the higher is value (or inversely, lower – depending on the concrete criterion), the better is partition; consequently, absolutely best k corresponds to the maximal (or the minimal) criterion value when k runs consecutive values. With elbow criteria, their value monotonically increases (or inversely, decreases – depending on the concrete criterion) as k grows, and absolutely best k corresponds to the locus of edge of this tendency where subsequent increase of k no longer is accompanied by steep growth (decline) of the criterion. The advantage of extremum criteria over elbow criteria is that for any two k one can judge which is better k; therefore extremum criteria are applicable for comparisons not only of a series of *consecutive* values of k. Elbow criteria do not allow comparing nonadjacent k and generally pairs of k, because it is unclear by which “side” of these two k or maybe between them the elbow is located. This is an essential drawback of elbow indices.

Comparing different k: priority of sharpness over extremum

Need to say that in practice the sharpness on the bend – of a peak or an elbow – has major importance for extremum-type criteria too. On a plot of value profile of such a criterion by different consecutive k one should pay attention not only to *max* (or *min*, depending on the specific criterion) value in the profile, but to sharp bends tendency, not necessarily coinciding with max. If a partition with the given k is much better than partitions with k-1 and with k+1, i.e. there is a peak, then it is a strong argument for that k, even if there on the plot exist regions of k where the criterion is generally “better”. Even a one sided bend (elbow) may occur preferable to absolute max for extremum criteria. The reason for these advices is as follows.

The point is that various clustering indices have their peculiar *small and having background, inherent* character biases in respect to the number of clusters: some “prefer” many clusters while other – few clusters. And manifestation of these tendencies depends on peculiarities of the data: there is almost impossible to invent datasets with different k that would be equi-valid to each other, simultaneously for all possible criteria[[1]](#footnote-1). Simulation experiments generating specified k clusters show that all criteria “are wrong” from time to time when clusters are mutually enough tight: they err in the sense that the overall max value does not match with the man-claimed number of the generated clusters. If to pay attention to peaks and elbows, rather than to max, then criteria are “wrong” less frequent in such experiments. (One should, however, realize the limitation of such simulation experiments in appraisal of the bias of clustering criteria: because a clustering criterion is not on the mission to discover the intended, at the generating, cluster structure, it simply assesses the sharpness of the structure as it turned out, while it might have turned out not at all such as it had been conceived at random generating.) By the idea, clustering criteria assisting to select a better k should have zero level base favouritism to k. Unfortunalely, this ideal is hardly likely achievable.

Some criteria (for example BIC or PBM) consciously prefer solutions with small number of clusters, then it is said they “penalize for the excess of clusters”. C-Index, contrary, openly tends to reward solutions with a greater number of clusters.

Criterion vs eye

If data are interval, clusters are not infrequently discernible visually on scatterplots in space of the variables or their principal components. But eye has its own prejudices (apophenia) and it is just one of, and is not the best, clustering criteria. Often this or that clustering criterion based on a statistical formula will “uncover” clusters not noticeable to eye, which interpretation afterwards will confirm their validity by content.

Choosing criterion: data nature

Some criteria (1) require as input a *set of data* (cases x variables), and it is cases that are objects partitioned in clusters/classes. Some such criteria demand scale, quantitative variables; while other – categorical variables or mix of scale and categorical. Some criteria may be optimal for binary variables. Criteria of other type (2) are based on analysis of *proximity matrix* between objects. Often such criteria don’t care what – cases/respondents or variables/attributes – constitute the items broken in clusters, because a proximity matrix might exist for items of any nature. Some of the criteria of type (2) demand specific proximity measures, for example, euclidean distances. While to other criteria the kind of proximities is indifferent; those latter are called universal criteria. (But the “universality” question is more delicate than it seems, since these criteria do, for example, summation of proximities, and there rises theoretical question whether any or not any kind of proximities may be summed.) Some criteria (3) can be calculated equivalently from variables (scale) as well as from matrix (of euclidean distances).

Note: if you clustered cases of a data set by K-means method but you are assessing the quality of solution by a criterion that demands proximity matrix (and not data set), compute matrix of (squared) euclidean distances (different proximity matrices can be obtained in output in SPSS by commands PROXIMITIES or CLUSTER) and input it.

Number of objects, or hilliness

There are criteria reacting to (proportionally equal) increase or decrease of frequency in clusters. That seems natural because adding objects into clusters amplifies relief of the distributional shape in the data, when clusters don’t coincide any much, and so the criterion value will expectedly enhance. But there are criteria that don’t react to such alteration of N: albeit it is important to such criteria that the density inside clusters be higher than outside clusters they do not reward strengthening of density through the increase of the number of objects in clusters.

Spatial shape

If a criterion requires scale data or euclidean distances, clusters might be of this or that configuration in the space. Here different clustering criteria have own preferences, i.e. they may reward, moderately, clusters exhibiting a specific spatial shape or relative position in the cluster solution. This quite intricate question can be split in three sub-questions: is the criterion sensitive, and how, (1) to the shape of cluster contours (round or oblong or curved); (2) to the rotation of oblong clusters relative one another, i.e. about their centroids; (3) to the rotation of the whole data cloud about its general centre (in the space of scale variables).

Remark for (1): there happens impression of *false* preference of round clusters. Not an existing clustering criterion demands clusters to nonoverlap by their margins in space, but the majority of cluster analysis methods output clusters exactly not overlapping in space. In these conditions (clusters are not allowed to superimpose physically) round clusters could get reseating closer to one another in space than oblong clusters with uncontrolled rotatedness, due to which the latter just have fewer chances to be encountered or be formed by clusterization in real investigation data – where, as we know, clusters are usually next to one another. Due to that phenomenon clustering criteria which are *in*sensitive to cluster’s outline, such as Calinski-Harabasz, more often *run into* “good” solutions with round, rather than elongated, clusters. This doesn’t mean that these criteria themselves prefer round clusters.

Distributional shape in clusters

There are criteria giving preference to clusters with uniform, flat distribution inside (for example, hyperball), and there are criteria giving preference to clusters with bell-shape distribution inside (like normal distribution); while other criteria don’t take the shape of density distribution in a cluster as important.

Space dimension

One more not easy question – reaction of different clustering criteria to the increase of dimensionality of the space, which is “spanned” by the data split into clusters. That question is connected, among other things, to curse of dimensionality that “hang over” euclidean distances on which many clustering criteria are based.

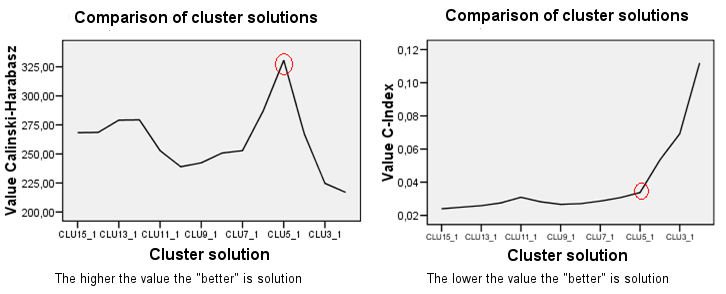
Statistical significance

Internal clustering criteria are not accompanied by probabilistic p-value, since they don’t infer about population and they are busy just with the dataset at hand. Of course, a good cluster solution in the form of high value of the criterion may be the consequence of contingent peculiarities of the concrete sample, overfitting. Cross-validation by equivalent dataset (in the form of stability check and generalizability check) will always be helpful.

**EXAMPLE**. Application of two clustering criteria for decision about the number of clusters in cluster analysis. Here is five pretty contacting clusters; eye does not recognize them at once.



With this data cloud hierarchical cluster by average linkage analysis was done based on euclidean distances, and all cluster partitions from 15-cluster to 2-cluster were saved. Then used were 2 clustering criteria, Calinski–Harabasz and C-Index, in attempt to choose which solution is the best.



As seen on the left plot, Calinski–Harabasz quite easily (in the given example) managed the task, indicating at 5-cluster solution as absolutely the best. C-Index, however, recommends 15- or 9-cluster solutions (C-Index is “better” when lower). Nevertheless this needs to be ignored and needs to pay attention to the bend which C-Index gives at 5 cluster: 5-cluster solution is still good but 4-cluster is much worse. Therefore, the best solution to select is 5-cluster one even on the right plot.

Of course, one should understand that if cluster structure in your data is almost entirely absent, then neither of all criteria will help choose the “correct” cluster solution, for there is no one. In such instances there’ll be no peaks or bends, but will be relatively smooth line trends, ascending, descending or horizontal – depending on the data and the criterion.

# MACRO !KO\_CALHARV: CALINSKI–HARABASZ, SSW, and LOG SS RATIO CRITERIA (INPUT – VARIABLES)

Version 3, Jul 2018 (Version 1, May 2001). Tested on SPSS Statistics 17, 20, 22.

!KO\_calharv vars= *v1 v3 to v10* /\*Quantitative variables by which to compare clustering solutions

/\*(name-by-name and/or via “to”)

/missing= VARIABLE /\*Delete cases with missing data in VARS listwise (LISTWISE, default)

/\*or sparing (VARIABLE)

/clusol= *clu10\_1 clu9\_1 clu8\_1 clu7\_1 clu6\_1 clu5\_1 clu4\_1 clu3\_1 clu2\_1*

/\*Variables representing cluster solutions (name-by-name list).

Minimal specification VARS, CLUSOL.

**Calinski–Harabasz criterion**

This index is Fisher’s F applied to multivariate data (because it does not account for covariational relations it is called Fisher’s pseudo-F). It is the ratio of the between-cluster nondensity, understood as the variance of clusters’ centres about the centre of the whole configuration of points, to the within-cluster nondensity, understood as the pooled variance of cluster members about their centres. The greater is the criterion’s value the better is the cluster partition. Formula (Calinski, R. B., Harabasz, J. A dendrite method for cluster analysis // Communications in Statistics, 1974, 3, 1–27.):

where *W* is the sum of squares of deviations inside clusters in the given *k*-cluster solution (that sum is equal to the trace of the pooled within-cluster scatter matrix **W**); *B* is the sum of squares of deviations of cluster centroids from the overall mean (that sum is equal to the trace of the between-cluster scatter matrix **B**); *n* – number of objects in the data.

Being based on the analysis-of-variance idea, the criterion is primarily for convex clusters, but it does not care whether they are round or oblong (ellipsoid). Also, it is insensitive to rotations of data in space: be it rotation of the whole cloud or rotation of individual clusters about their centres. It is insensitive to the distributional shape in clusters (bell vs uniform): thought the criterion embodies ANOVA idea, it does not do significance testing and therefore does not insist on normal distribution.

The criterion is sensitive to the number of objects in data, so the compared clustering partitions must be comprised of the same total number of objects *n*. Although this criterion’s dependency on *n* is linear, it is difficult to remove because the slope depends on *k*.

Keeping everything other equal, the criterion tends to give preference to cluster partitions with approximately equal-populated, by the number of objects, clusters (because on the background of many-object clusters few-object clusters contribute little in *B*, but they enter in *k*, and that diminishes the formula numerator, centroids’ variance).

**SSW criterion**

This simply is the pooled within-cluster sum of squares of deviations, *W*. The criterion is suited only to compare partitions of the same data with different number of clusters *k*. On the plot where consecutive k are laid on X-axis ascendingly or descendingly, the “best” partition corresponds to a “elbow” after which the curve is relatively flat for subsequent k. This is a popular but primitive index.

**Log SS Ratio criterion**

logarithm of the ration between the between- and the within-cluster sums of squares of deviations (Hartigan, J.A. Clustering algorithms. New York: Wiley, 1975).

The profile plot of this criterion is usually like SSW, only upturned. SSW marks the best solution by “bottom elbow”, while Log SS Ratio by “top elbow”. In else respect the criterion is similar with Calinski–Harabasz (besides that it is elbow criterion, not extremum criterion). Its asset over Calinski–Harabasz is that it almost does not depend on the total number of objects *n*.

***Subcommands***

**VARS**

Variables (quantitative) by which you need to assess cluster partitions. They must not have value: *-999.*

**MISSING**

By default and by MISSING=LISTWISE, cases which are missing at least in one variable VARS are omitted from the analysis. MISSING=VARIABLE deletes missings from each variable separately and uses in calculations all valid values of the variables. The index is not computable (will come in result as sysmis) if after exclusion of all missings it appear that at least by one variable there is less than 2 clusters.

**CLUSOL**

Name-by-name list of numeric variables (minimum one) which group the cases – cluster partitions, or “cluster membership variables”; variable names should be *no longer than 8 bytes*. The variables are recommended to write in the sequence of increase or decrease of the number of clusters in them. Minimal number of clusters in a variable – 2 (or the result will come system missing).

Grouping variables’ values (cluster codes) may be any numbers: each unique value designates a cluster. The macro excludes from the analysis, separately for each grouping variable, cases which are missing or have value 0. Researcher may use this fact if they want to ignore some entire clusters or individual objects in the analysis: just make these clusters/objects codes in that grouping variable user- or system-missing or replace them with 0; the number of clusters left for the analysis must be no less than 2. Thus, the being compared partitions can, in principle, consist of different number of objects (while how much such comparison is acceptable – this depends on the criterion).

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). To select cases for the analysis you may use filtering (FILTER or USE) and SELECT IF. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_CALHARM: CALINSKI–HARABASZ, SSW, and LOG SS RATIO CRITERIA (INPUT – MATRIX)

Version 3, Jul 2018 (Version 1, Jul 2000). Tested on SPSS Statistics 17, 20, 22.

!KO\_calharm matrix= *var1 to var100* /\*Variables constituting matrix of distances

/\*(name-by-name and/or via “to”)

/square= NO /\*Whether to square the distances (YES) or not (NO, default)

/\*because they’re already squared

/clusol= *clu10\_1 clu9\_1 clu8\_1 clu7\_1 clu6\_1 clu5\_1 clu4\_1 clu3\_1 clu2\_1*

/\*Variables representing cluster solutions (name-by-name list).

Minimal specification MATRIX, CLUSOL.

This macro computes Calinski-Harabasz, SSW, Log SS Ratio criteria, like !KO\_CALHARV macro, but it accepts not variables (features) as input but a square symmetric matrix of distances between objects. For the sake of geometrical correctness it needs that those be squared euclidean distances or euclidean distances (the macro can square them). If it’s other distances/dissimilarities than euclidean, the criteria are correct exactly to the extent to which the distances are close to be euclidean. In any case, they ought to be metric distances. In any case, the matrix must be distances, not similarities.

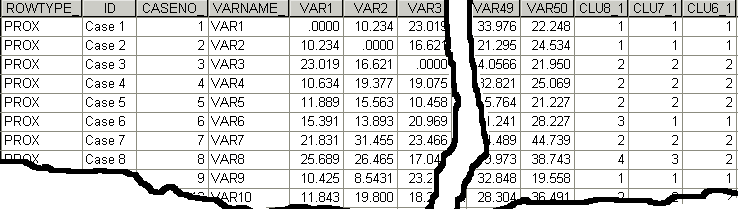
Calinski-Harabasz and Log SS Ratio are insensitive to proportional alteration of distances (of squared or nonsquared input distances).

***Matrix structure***

The matrix must be square symmetric. The macro does not require the matrix to include columns ROWTYPE\_ and VARNAME\_ or other auxiliary. The cluster membership variables must be filed in the same dataset as the matrix. There must be no variable named *CASENUM#* in the dataset.

EXAMPLE 1.

!KO\_calharm matrix= var1 to var50 /clusol= clu8\_1 clu7\_1 clu6\_1.



* The body of the square symmetric distance matrix between 50 objects is defined by columns VAR1 to VAR50 (and correspondingly by the same number of rows). Cluster solution (grouping) variables – CLU8\_1 CLU7\_1 CLU6\_1, their values are cluster codes.
* The macro will compute 3 values of the index – one for each cluster solution, and output it as a new unnamed dataset, and will produce a plot.

***Subcommands***

**MATRIX**

Specify variables which are the columns of the distance matrix itself. The matrix must be square symmetric and must not contain missings. You may specify the list name by name and/or by range via “to”. The matrix body found in the working dataset should be square and symmetric, and you must input all the body columns by MATRIX subcommand.

*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?*.

**SQUARE**

The macro treats input distances as squared euclidean. If your matrix is nonsquared euclidean distances (or measures which you treat as nonsquared euclidean distances), you should indicate the macro that it should square them: SQUARE=YES.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). Do not use SELECT IF before the macro because you will ruin the square shape of the input matrix. You may, however, use filtering (FILTER or USE) for case selection. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_DAVBOULV: DAVIES-BOULDIN and PBM CRITERIA (INPUT – VARIABLES)

Version 3, Jun 2018 (Version 1, Sep 2012). Tested on SPSS Statistics 17, 20, 22.

!KO\_davboulv vars= *v1 v3 to v10* /\*Quantitative variables by which to compare clustering solutions

/\*(name-by-name and/or via “to”)

/clusol= *clu10\_1 clu9\_1 clu8\_1 clu7\_1 clu6\_1 clu5\_1 clu4\_1 clu3\_1 clu2\_1*

/\*Variables representing cluster solutions (name-by-name list)

/norm= /\*Use euclidean (L2, default) or manhattan (L1) norm.

Minimal specification VARS, CLUSOL.

**Davies–Bouldin criterion**

This criterion (Davies, D. L., Bouldin, D. W. A cluster separation measure // IEEE Transactions on Pattern

Analysis and Machine Intelligence, 1979, 1(2), 224-227) is a ratio of within-cluster non-density to the distance between cluster centres. Formula:



where *s* is the root mean square deviation in a cluster from its centroid, the measure of its dispersion; *d* is the distance between centroids *i* and *j*. Each cluster *i* (there is *k* clusters) is paired with each of the rest, and the maximal ratio of the two *s* sum to the *d* is recorded. This quantity characterizes the degree of nonisolatedness of cluster *i*. The magnitude of the criterion characterizing the cluster partition in whole is the average of *k* such values. The lower is the criterion value the better is cluster partition.

Based, like Calinski-Harabasz, on the idea of ANOVA in space, this criterion is like it by the majority of properties regarding spatial configuration and within-cluster distribution. The criterion is insensitive to the number of objects *n*, so it may be used to compare groupings with different number of objects.

As opposed to Calinski-Harabasz, Davies-Bouldin index isn’t inclined towards similar-sized, in respect to the number of objects, clusters; instead, it tends to prefer solutions with equally-distanced from each other clusters (which is associated with that it measures cluster division not from the overall centre of configuration but from the neighbour cluster).

**PBM criterion**

This criterion is named as the acronym of its authors’ names (Pakhira, M.K., Bandyopadhyay, S., Maulik, U. Validity index for crisp and fuzzy clusters // Pattern Recognition, 2004, 37, 487-501) is the product of three factors:

where is the sum of deviations (not sum of squared deviations) of objects from the total centroid of the sample (*n* objects in all); is the sum of deviations of the *n* objects from cluster centroids – each object from its cluster centroid; is the distance between a pair of cluster centroids, – thus, the third factor is the maximal distance between cluster centroids. The idea of the criterion is as follows. The second factor expresses the intra-cluster compactness, and the third one – inter-cluster divorce; as simultaneously both are greater a cluster partition is better. But with the increase of fractionness (number of clusters, *k*) there is the natural tendency for both factors to increase, and because there is a wish to take off the tendency, the curbing multiplier 1/*k* is introduced. In so doing, the criterion “penalizes” for the excess of the number of clusters. If , the macro won’t compute the criterion’s value. The higher is the criterion’s value the better is partition.

The criterion is sensitive to a linear transform of variables; if it is same to all the variables – then reacts linearly. It holds some preference towards ellipsoid clusters over round ones. The criterion is weakly sensitive to the number of objects *n*.

***Subcommands***

**VARS**

Variables (quantitative) by which you need to assess cluster partitions of cases. The macro excludes cases with missing values listwise.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**NORM**

This subcommand allows to request the criteria on the base on Manhattan instead of Euclidean distances. By default and with NORM=L2 (see criteria’ formulas), *ew* and *et* are euclidean deviations of an object, *s* is standard, i.e. root mean square deviation (computed on “df=n”) in a cluster from its centroid, and *d* is euclidean distance between cluster centroids. With NORM=L1, *ew* and *et* are manhattan deviations of an object, *s* is mean absolute deviation in a cluster from its centroid, and *d* is manhattan distance between cluster centroids.

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). To select cases for the analysis, you may use filtering (FILTER or USE) and SELECT IF. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_DAVBOULM: DAVIES-BOULDIN and PBM CRITERIA (INPUT – MATRIX)

Version 2, Jun 2018 (Version 1, Sep 2012). Tested on SPSS Statistics 17, 20, 22.

!KO\_davboulm matrix= *var1 to var100* /\*Variables constituting matrix of distances

/\*(name-by-name and/or via “to”)

/square= NO /\*Whether to square the distances (YES) or not (NO, default)

/\*because they’re already squared

/clusol= *clu10\_1 clu9\_1 clu8\_1 clu7\_1 clu6\_1 clu5\_1 clu4\_1 clu3\_1 clu2\_1*

/\*Variables representing cluster solutions (name-by-name list).

Minimal specification MATRIX, CLUSOL.

This macro computes Davies-Bouldin and PBM criteria, like !KO\_DAVBOULV macro (with NORM=L2), but it accepts not variables (features) as input but a square symmetric matrix of distances between objects. For the sake of geometrical correctness it needs that those be squared euclidean distances or euclidean distances (the macro can square them). If it’s other distances/dissimilarities than euclidean, Davies-Bouldin and PBM are correct exactly to the extent to which the distances are close to be euclidean. In any case, they ought to be metric distances. In any case, the matrix must be distances, not similarities. The situation and input are just like with macro [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ).

Davies-Bouldin is insensitive and PBM is sensitive to proportional alteration of distances (of squared or nonsquared input distances).

***Subcommands***

**MATRIX**

Specify variables which are the columns of the distance matrix itself. The matrix must be square symmetric and must not contain missings. You may specify the list name by name and/or by range via “to”. The matrix body found in the working dataset should be square and symmetric, and you must input all the body columns by MATRIX subcommand.

*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?*.

**SQUARE**

The macro treats input distances as squared euclidean. If your matrix is nonsquared euclidean distances (or measures which you treat as nonsquared euclidean distances) you should indicate the macro that it should square them: SQUARE=YES.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). Do not use SELECT IF before the macro because you will ruin the square shape of the input matrix. You may, however, use filtering (FILTER or USE) for case selection. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_CCCRITV: CUBIC CLUSTERING CRITERION, LOG DET RATIO CRITERION (INPUT – VARIABLES)

Version 2, Jul 2018 (Version 1, Jul 2016). Tested on SPSS Statistics 17, 20, 22.

!KO\_cccritv vars= *v1 v3 to v10* /\*Quantitative variables by which to compare clustering solutions

/\*(name-by-name and/or via “to”)

/clusol= *clu10\_1 clu9\_1 clu8\_1 clu7\_1 clu6\_1 clu5\_1 clu4\_1 clu3\_1 clu2\_1*

/\*Variables representing cluster solutions (name-by-name list)

/fast= YES /\*Fast regime: YES or NO (default).

Minimal specification VARS, CLUSOL.

**Cubic Clustering criterion**

This index is based on the ratio of within-cluster sum of squares of deviations to the total sum of squares of deviations, *W/T*, quantity known as Wilks lambda. In multivariate data, lambda takes account of covariational relations too, but in the case of cubic clustering criterion (CCC) they are ignored, so one may speak of multivariate pseudo-lambda of Wilks (like in the case of Calinski-Harabasz criterion we speak of pseudo-F of Fisher). 1-W/T is R-squared = Eta-squared. CCC has formula (Sarle, W.S. Cubic clustering criterion // SAS Technical Report A-108, 1983, SAS Institute):

where

(**W** is the pooled within-cluster scatter matrix; **T** is the scatter matrix of the total sample)

*RsqE* is the expected value of *Rsq* under conditions where our data would be cluster-free homogeneous and would represent by themselves a rectangular briquette of adjoining cubes. Thus, CCC is the logarithm of the ratio of the observed *W/T* to the expected *W/T* in the absence of clusters. *Stab* is the empirically selected stabilizer (see below).

where *n* is the number of objects in data; *k* is the number of clusters; *p* is the dimensionality of data which equals the number of nonzero singular values of the data covariance matrix **T***/(n-1)*. Quantities *u* are those singular values normalized, i.e., divided by constant *c* which is defined this way:

where *P* is the product of the first *p\** singular values. Sorting out *p\*= p, p-1, p-2…,* and computing repeatedly *c* and normalizing by it the singular values, set the choice on such *p\** and hence such *u* for the formula, so that *p\** be less than *k*, and *up\** be not less than 1.

is a correction allowing to compare CCC values in data with different dimensionality (number of variables) and different number of objects. So, CCC is one of few clustering criteria which at the development were attempted to standardize (how much successfully – is a question) for different data sets. Further, it departs from concrete “null” hypothesis: cluster-free data come from uniform distribution (“hypebox”). The complexity of the formulae is because of that standardization of the value, not because of the basic idea (*W/T*) of the criterion. CCC can be positive or negative. The higher is the value the better is cluster solution, that is, the more pronounced are clusters.

I would note that my examination of the criterion output by the macro makes one doubt in sufficient standardization quality of CCC. In particular, the criterion appears rather sensitive to the number of objects in data; therefore, in my opinion, cluster partitions must consist of equal number of objects *n*.

As founded on the ANOVA-in-space idea, CCC is mostly suited to detect convex clusters. It resembles Calinski-Harabasz in a number of properties, but, in contrast with it, prefers round clusters to elongated (at dimensionality *p* greater than 2 or 3), and of the latter – prefers parallel in pile. Why CCC has these tendencies – becomes understandable if to analyze the formula of *RsqE*, how this quantity respond to spatial mutations of the cloud.

**Log Det Ratio criterion**

where **T** and **W** are, respectively, the total scatter matrix and the pooled within scatter matrix. Originally (Scott, A.J., Symons, M.J. Clustering methods based on likelihood ratio criteria // Biometrics, 1971, 27, 387-397), the criterion is also multiplied by *n*, but that is unnecessary. If either or both determinants are zero, the criterion is not computed. This index expresses the MANOVA ideology, i.e., it is frankly sensitive to the shape data cloud and the shape of the pooled (centroid-superposed) cluster. It is clear since *det* is connected with the volume of a multidimensional body. The criterion’s value is increased if clusters are oblong and parallel and together occupy a big area (i.e., they aren’t “chain” of clusters). However, the criterion is of elbow-type criteria, nor extremum-type criteria. The best number of clusters is that k, further growth of which is accompanied only by gently slopping of the criterion’s value.

***Subcommands***

**VARS**

Variables (quantitative) by which you need to assess cluster partitions of cases. The macro excludes cases with missing values listwise.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**FAST**

This subcommand acts only when there is more than one variable specified in CLUSOL. FAST=YES speeds up the macro, what can be apparent in case of large data (many variables VARS and many cluster solutions CLUSOL). Do not apply FAST=YES if any CLUSOL variables contain missing values or zeroes. By default/unspecification, FAST=NO.

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). To select cases for the analysis you may use filtering (FILTER or USE) and SELECT IF. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_CCCRITM: CUBIC CLUSTERING CRITERION (INPUT – MATRIX)

Version 1, Jul 2016. Tested on SPSS Statistics 17, 20, 22.

!KO\_cccritm matrix= *var1 to var100* /\* Variables constituting matrix of distances

/\*(name-by-name and/or via “to”)

/square= NO /\*Whether to square the distances (YES) or not (NO, default)

/\*because they’re already squared

/clusol= *clu10\_1 clu9\_1 clu8\_1 clu7\_1 clu6\_1 clu5\_1 clu4\_1 clu3\_1 clu2\_1*

/\*Variables representing cluster solutions (name-by-name list)

/fast= YES /\*Fast regime: YES or NO (default).

Minimal specification MATRIX, CLUSOL.

This macro computes cubic clustering criterion (CCC), like !KO\_CCCRITV macro, but it accepts not variables (features) as input but a square symmetric matrix of distances between objects. For the sake of geometrical correctness it needs that those be squared euclidean distances or euclidean distances (the macro can square them). If it’s other distances/dissimilarities than euclidean, criterion CCC is correct exactly to the extent to which the distances are close to be euclidean. In any case, they ought to be metric distances. In any case, the matrix must be distances, not similarities. The situation and input are just like with macro [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ).

The index is insensitive to proportional alteration of distances (of squared or nonsquared input distances).

Unlike !KO\_CCCRITV, this macro does not compute Log Det Ratio criterion. If you need this criterion, create, out of the distance matrix, data “objects × variables”, i.e. coordinates, and use !KO\_CCCRITV. The coordinates are created from a distance matrix by means of metric multidimensional scaling, for example, Torgerson method aka PCoA; wherein you should request a larger as possible number of variables, in order to exhaust all or almost all of the variance. Simple PCoA macro you can find among my matrix functions “MATRIX – END MATRIX functions”.

***Subcommands***

**MATRIX**

Specify variables which are the columns of the distance matrix itself. The matrix must be square symmetric and must not contain missings. You may specify the list name by name and/or by range via “to”. The matrix body found in the working dataset should be square and symmetric, and you must input all the body columns by MATRIX subcommand.

*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?*.

**SQUARE**

The macro treats input distances as squared euclidean. If your matrix is nonsquared euclidean distances (or measures which you treat as nonsquared euclidean distances) you should indicate the macro that it should square them: SQUARE=YES.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**FAST**

This subcommand acts only when there is more than one variable specified in CLUSOL. FAST=YES speeds up the macro what can be apparent in case of large data (big matrix MATRIX and many cluster solutions CLUSOL). Do not apply FAST=YES if any CLUSOL variables contain missing values or zeroes. By default/unspecification, FAST=NO.

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). Do not use SELECT IF before the macro because you will ruin the square shape of the input matrix. You may, however, use filtering (FILTER or USE) for case selection. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_RATLAN: RATKOWSKY–LANCE CRITERION

Version 2, Jul 2016 (Version 1, Jul 2012). Tested on SPSS Statistics 17, 20, 22.

!KO\_ratlan scavars= *V1 V3 to V10* /\*Quantitative variables by which to compare clustering solutions (may be /\*via “to”)

/catvars= /\*Categorical variables by which to compare clustering solutions (may be /\*via “to”)

/clusol= *CLU10\_2 CLU9\_2 CLU8\_2 CLU7\_2 CLU6\_2 CLU5\_2 CLU4\_2 CLU3\_2 CLU2\_2*

/\*Variables representing cluster solutions (name-by-name list)

/univar= NO /\*Save also criterion value for each variable: YES or NO (default).

Minimal specification CLUSOL and at least one of SCAVARS, CATVARS.

This index (Ratkowsky, D.A., Lance, G.N. A criterion for determining the number of groups in a classification // Australian Computer J, 1978, 10, 115-117.) assesses the degree of scatter of clusters separately by each variable, and the index values from all the variables are averaged into single value. The criterion is fit for both scale as well as nominal variables.

For each scale variable *q* the index is (normalized by the number of clusters *k*) correlation ratio Eta:

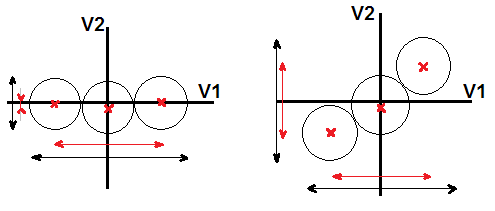
(*B* and *T* are the between-cluster and the total sums of squares of deviations, respectively)

For each nominal variable *r* the index is (normalized by the number of clusters *k*) association coefficient Cramer’s V:

(*χ2* is the association chi-square measure between the variable (having number of categories *l*) and the clustering variable; *n* is the sample size)

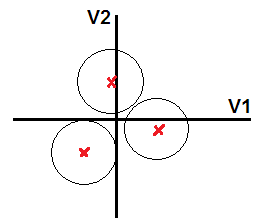
The overall index is simply the arithmetic average of these quantities from all variables, *p1* scale and *p2* nominal; else speaking, Ratkowsky-Lance is

The higher is value the better is partition. Because the criterion is averaging univariate, it is sensitive to rotation of a data cloud in the space of scale variables. On the left picture, three clusters differ in centroids along V1 and almost coincide in centroids along V2. By variable V2, the spread of centroids relative the data variation is very small, so criterion is close to 0 by that variable. Since overall Ratkowsky-Lance is the simple unweighted mean between the criterion value by V1 and the criterion value by V2, that overall value will be small for the left picture, - less than that for the right picture where the data cloud is rotated so that centroids appear to be spread sufficiently strongly by both variables. Ratkowsky-Lance rewards data rotated so that variances along the axes differ little.

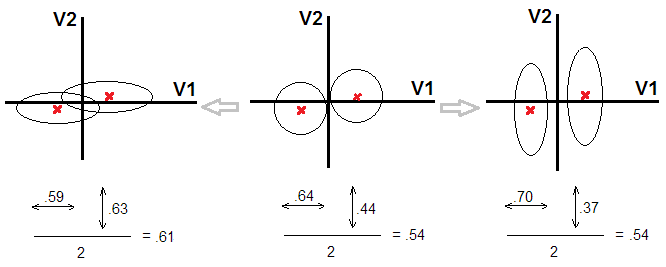


Thus, overall Ratkowsky-Lance won’t suit you if for your data rotated in space – are still those “same” data as before rotation. You might want to replace scale variables by their principal components prior application of the criterion; rotation into principal components will transfer configurations like the one on the right, where principal axes are diagonal, into such like the one on the left.

If clusters are scattered about space of scale variables evenly, which is typically associated with variable uncorrelatedness, then overall Ratkowsky-Lance changes weakly at rotations, as for example with such data:



Ratkowsky-Lance criterion is also liable to influences of cluster shape and their rotation about their centroids. On the picture below, two round clusters in the space of scale variables V1 and V2 diverge stronger by V1 (criterion = .64) than by V2 (criterion = .44); overall criterion = .54.



On the left, these clusters are stretched into ellipses (preserving the initial bivariate within-cluster variance) along V1, forming a “chain”, and on the right they are stretched along V2, forming a “pile”. The criterion magnitude did not change on the right (.54), but on the left it increased up to .61. That is somewhat unexpected for eye, which would count the best of all cluster solution to be of the right – where cluster don’t contact at all. But for Ratkowsky-Lance criterion the result is in the order of things and that has its meaning. This criterion accentuates (encourages) shrinking (compaction) of a cluster along that variable by which clusters are close (by their centres); while if clusters are far apart their shrinking inside them along the variable separating them is almost not encouraged (because, in a sense, it is already superfluous).

As space dimensionality grows, the criterion begins regularly preferring ellipsoid clusters to spherical ones. It is insensitive to the distributional shape in clusters (bell vs uniform). The criterion is insensitive to the number of objects *n*, so it may be used to compare groupings with different number of objects.

If a variable is dichotomous, there is no difference whether you treat it as scale or as categorical (because Eta and Cramer’s V are equal in that case). It was shown that Ratkowsky-Lance is very good at detecting the number of clusters in dichotomous data (Dimitriadou, E., Dolnicar, S., Weingassel, A. An examination of indexes for determining the number of clusters in binary data sets // Psychometrika, 2002, 67(1), 137-160.).

***Subcommands***

**SCAVARS, CATVARS**

Specify scale variables in SCAVARS. Specify categorical variables in CATVARS. You may write the lists name by name and/or via “to” . You must specify at least one list of the two. The macro excludes cases with missing values listwise: if a case is missing in at least one variable, that case is excluded from calculations by all the variables.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**UNIVAR**

Optional subcommand saving (UNIVAR=YES) into the output dataset the criterion’s value for each variable, not only the averaged value. Note that criterion is computed only if it is computable for every variable, else system missing is output.

Thanks to this option you may see/compare contribution of individual variables to the divergence between clusters.

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). To select cases for the analysis you may use filtering (FILTER or USE) and SELECT IF. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_RPBCLU: POINT-BISERIAL CORRELATION and McCLAIN-RAO

Version 2, Jul 2018 (Version 1, Sep 2001). Tested on SPSS 17, 20, 22.

!KO\_rpbclu matrix= *var1 to var200* /\*Variables constituting matrix of similarities or distances

/\*(name-by-name and/or via “to”)

/clusol= *clu6 clu5 clu4 clu3 clu2*

/\*Variables representing cluster solutions (name-by-name list).

Minimal specification MATRIX, CLUSOL.

**Point-biserial correlation criterion**

The idea to correlate the dissimilarity of objects with their classification is evident and simple, it is called cophenetic correlation. When the classification is a cluster partition, the speech goes about measuring agreement between dissimilarity of objects one from another, on one side, and the fact whether these objects belong to different clusters or to one same cluster, on the other side. If more similar objects fall in same cluster and more dissimilar ones fall in different clusters, the correlation will be high, thence the given cluster solution is good. Attractiveness of correlation-based indices over such measures like variance-based Calinski–Harabasz index lies in their universality: they can be used both to distances or similarities as proximity measures between objects, – and to any measures, irrespective of whether those are in accord or not with the euclidean (and more generally, metric) space geometry.

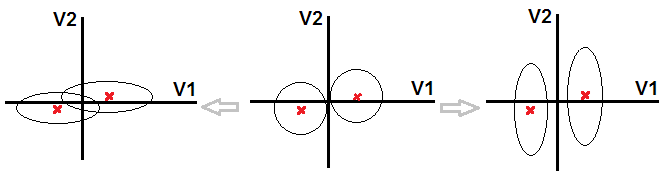
Point-biserial coefficient of correlation *rpb* is the usual Pearson *r* between the magnitude of proximity (likeness) and the fact “in the same cluster / in different clusters”. It shows relative difference between the average size of within-cluster pairwise proximities and the average size of between-cluster pairwise proximities:

where *Mb* is the mean distance between different-cluster objects (such distances are *fb*), and *Mw* is the mean distance between same-cluster objects (such distances are *fw*); *f* distances in all. *s* is the standard deviation in distribution of all the distances. Point-biserial r varies between -1 and 1. The higher the value the better is cluster partition. *Rpb* has moderately pronounced background preference towards middle number of clusters (about 4-10).

This index is insensitive to adding a constant to proximities (say wider, to linear transform of proximities). For example, on the next picture there shown are two clusters, and on the right – they after addition of 10 mm to all the pairwise distances. Many other clustering criteria will consider the clusteredness on the right to be less convincing than the one on the left, which also concurs with the visual impression. But point-biserial correlation will judge both solutions to be of equal quality. Such “geometric” counter-intuitiveness of the criterion may sometimes be seen as its drawback.



In euclidean space using euclidean distances, *rpb* value will be the higher the more to the right is the configuration on the following picture (distances between centroids and within-cluster variances are hold constant here).



I.e. the criterion encourages ellipsoids if they form pile, not chain. But with the rise of space dimensionality cluster partitions formed of ellipsoid clusters start to lose stably to partitions formed of round clusters. Thus, *rpb* by and large sympathizes to spherical clusters.

Point-biserial correlation criterion is apt for both distances or similarities (similarities are internally converted by the macro linearly into distances).

The criterion is insensitive to the number of objects in data, so the being compared cluster partitions may consist of not equal total number of objects *n*.

**McClain-Rao criterion**

Defined as the ratio of the average distance between different-cluster objects to the average distance between same-cluster objects (McClain, J.O., Rao,V.R. Clustisz: A program to test for the quality of clustering of a set of objects // Journal of Marketing Research, 1975, 12(4), 456-460):

where *Sb* and *Sw* are summed distances between different-cluster objects and between same-cluster objects, respetively; other symbols see above. If *Sb*=0, the index is not computed. Formally, the lower the value the better is cluster partition, but the criterion inherently strongly favours greater k, so there is the main reason to look at the bend upwards (bottom “elbow”).

McClain–Rao criterion is apt for both distances or similarities (similarities are internally converted by the macro linearly into distances).

The criterion has a small tendency to reward ellipsoid clusters organized in pile or ring. Adding a constant to proximities changes the criterion linearly. The criterion is insensitive to the number of objects in data, so the being compared cluster partitions may consist of not equal total number of objects *n*.

***Subcommands***

**MATRIX**

Specify variables which are the columns of the distance matrix (zeros must stand on the diagonal, off-diagonal elements must be nonnegative, greater value corresponds to greater dissimilarity) or of the similarity matrix (positive numbers must stand on the diagonal, off-diagonal elements may have any sign, greater value corresponds to greater similarity). The matrix must be square symmetric and must not contain missings. Be aware that the macro does not check the matrix correctness: so take care that it meets the just above described requirements. The structure of a matrix and requrements thereof are shown above in macro [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ). The matrix body found in the working dataset should be square and symmetric, and you must input all the body columns by MATRIX subcommand.

*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?*.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). Do not use SELECT IF before the macro because you will ruin the square shape of the input matrix. You may, however, use filtering (FILTER or USE) for case selection. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_GAMMACLU: GAMMA STATISTIC

Version 2, Jan 2022 (Version 1, Jul 2012). Tested on SPSS Statistics 20, 22, 26.

!KO\_gammaclu matrix= *var1 to var200* /\*Variables constituting matrix of similarities or distances

/\*(name-by-name and/or via “to”)

/rescrnd= *3* /\*Optional: rescale/round continuous proximities, for speed

/clusol= *clu6 clu5 clu4 clu3 clu2*

/\*Variables representing cluster solutions (name-by-name list).

Minimal specification MATRIX, CLUSOL.

This index is Goodman-Kruskal Gamma, a statistic expressing nonparametric rank correlation. Unlike point-biserial correlation, gamma does not operate with the magnitude of pairwise proximities themselves, but rather assesses the probability that two randomly taken same-cluster objects will have greater similarity than two randomly taken different-cluster objects:



where *w+* is the number of instances (combinations) in the data when two same-cluster objects are closer to each other than two different-cluster objects; *w-* is the number of instances in the data when two same-cluster objects are farther from each other than two different-cluster objects. Gamma varies between -1 and 1. The higher the value the better is cluster partition.

Because gamma is a measure of monotonic correlation, it is insensitive to any monotonic alteration of proximities. Therefore, in particular, it is unimportant whether you enter square or nonsquare distances. Such insensitivity to magnitude of proximities in the specific geometric sense is counter-intuitive and may sometimes be seen as a drawback.

In euclidean space using euclidean distances, gamma’s attitude towards ellipsoid clusters in comparison to spherical clusters is like that of point-biserial correlation (see description in !KO\_RPBCLU): with the increase of space dimensionality there grows preference to spherical clusters.

Gamma criterion is apt for both distances or similarities.

The criterion is insensitive to the number of objects in data, so the being compared cluster partitions may consist of not equal total number of objects *n*.

Macro !KO\_GAMMACLU is quite slow and therefore is not recommended for data with thousands of objects. But subcommand RESCRND can speed up its job.

***Subcommands***

**MATRIX**

Specify variables which are the columns of the distance matrix (zeros must stand on the diagonal, off-diagonal elements must be nonnegative, greater value corresponds to greater dissimilarity) or of the similarity matrix (positive numbers must stand on the diagonal, off-diagonal elements may have any sign, greater value corresponds to greater similarity). The matrix must be square symmetric and must not contain missings. Be aware that the macro does not check the matrix correctness: so take care that it meets the just above described requirements. The structure of a matrix and requrements thereof are shown above in macro [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ). The matrix body found in the working dataset should be square and symmetric, and you must input all the body columns by MATRIX subcommand.

*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?*.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**RESCRND**

Calculation of Gamma, when data (in our case they are values of a proximity matrix) are continuous, may take much time even with a matrix of moderate size. Reducing the number of distinct values in the matrix will considerably accelerate the process. S/c RESCRND reduces the number of distinct values in the input matrix, discretizing them by the rescale-rounding method. Specify digit: 1, 2, 3, 4, or 5. It is the number of decimal positions the macro will retain for the values of the matrix after linearly rescaling them into the range 0–1. The quickest work of the macro will be under RESCRND=1, because with such discretization there can remain not above 11 distinct values. Under RESCRND=2 there will remain not above 101 of such. Under RESCRND=5 there will remain not above 100001 distinct values, and so the run time of the macro can be long. The maximal work time will be (with continuous input) in case you unspecify/omit the subcommand, for no discretizing will take place at all then.

There exists obvious trade-off between scrupulousness and speed. When the number of distinct values among proximities has been made too small, gamma loses sensitivity to that how small changes in distances/similarities between objects correlate with their clusteredness. Imagine the marginal case when there only two different values in a proximity matrix. Then the index will express association most “rough” one: between variable “small distance vs big distance” and variable “in one cluster vs in different clusters”. But, from the other side, too small differences in distances/similarities cannot bear practical significance for clusterization. Imagine that the original distances in the matrix span the range 0.2 to 9.7. On this background, a pair of distances, say, 3.56146 and 3.56148, from the matrix – are two practically indistinguishable distances; it is hardly a reason lending them to be different and be able to determine differently the cluster outcome of the objects. Therefore, a moderate discretization will be justified: sooner, it almost will not affect Gamma’s value, while will accelerate the job. RESCRND=3 variant is the acceptable trade-off in most cases.

The macro shows the number of distinct values left in the proximity matrix after discretization – in variable *N#$.*

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). Do not use SELECT IF before the macro because you will ruin the square shape of the input matrix. You may, however, use filtering (FILTER or USE) for case selection. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_CINDEX: C-INDEX

Version 1, Sep 2001. Tested on SPSS 11.5, 13, 15.

!KO\_cindex matrix= *var1 to var200* /\*Variables constituting matrix of similarities or distances

/\*(name-by-name and/or via “to”)

/clusol= *clu6 clu5 clu4 clu3 clu2*

/\*Variables representing cluster solutions (name-by-name list)

/fast= YES /\*Fast regime: YES or NO (default).

Minimal specification MATRIX, CLUSOL.

C-Index (Hubert, L.J., Levin, J.R. A general statistical framework for assessing categorical clustering in free recall // Psychol. Bull., 1976, 83, 1072-1080) is a universal criterion like point-biserial correlation. It shows how much the observed within-cluster density – the sum of within-cluster pairwise proximities – is relatively close to such sum utmost maximal (mostly actually unattainable) under the given number of within-cluster proximities.

where *Sw* is the summed distance between same-cluster objects (there are *fw* such distances); *Smin* is the sum of *fw*least distances between objects (no matter within-cluster or between-cluster a distance is), *Smax* is, analogously, the sum of *fw*greatest distances between objects. C-Index varies between 0 and 1. The lower is the value the better is the cluster partition.

C-Index is linearly identical (but with a negative slope coefficient) to point-biserial correlation (see !KO\_RPBCLU) rescaled relative its empirical maximum. I.e., if *rpb*=.57, say, but in the data with the given partition being considered *rpb* cannot exceed, say, .95, then the rescaled *rpb*=.57/.95. C-Index criterion is the magnitude which is equivalent to this one; thereby, rescaled correlation and C-Index are the same thing, in logical contents. C-Index tends to prefer more clusters than *rpb*. For – during clusterization – while clusters are small (and hence they are many) they have more chances to approach closely that “ideal” which this criterion purports.

Like point-biserial correlation, C-Index is insensitive to linear transformation of proximities. Such indifference to proximity size is counter-intuitive in a specific geometrical sense and may sometimes be regarded a drawback.

In euclidean space using euclidean distances, the index’s attitude to ellipsoid clusters in comparison with spherical ones is similar to that of point-biserial correlation (see !KO\_RPBCLU description): with the increase of space dimensionality there strengthens the preference for spherical clusters. By my tentative impressions in simulation trials, C-Index punishes chains stronger (see picture in !KO\_RPBCLU description) than it rewards piles, while point-biserial r, contrary, stronger rewards the second than it punishes the first. And in regard of round clusters, both criteria often give very similar results.

C-index criterion is apt for both distances or similarities (similarities are internally converted by the macro linearly into distances).

The criterion is insensitive to the number of objects in data, so the being compared cluster partitions may consist of not equal total number of objects *n*.

Macro !KO\_CINDEX is comparatively slow and is not recommended for large (thousands) amount of objects. S/c FAST substantially speeds up the job.

***Subcommands***

**MATRIX**

Specify variables which are the columns of the distance matrix (zeros must stand on the diagonal, off-diagonal elements must be nonnegative, greater value corresponds to greater dissimilarity) or of the similarity matrix (positive numbers must stand on the diagonal, off-diagonal elements may have any sign, greater value corresponds to greater similarity). The matrix must be square symmetric and must not contain missings. Be aware that the macro does not check the matrix correctness: so take care that it meets the just above described requirements. The structure of a matrix and requrements thereof are shown above in macro [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ). The matrix body found in the working dataset should be square and symmetric, and you must input all the body columns by MATRIX subcommand.

*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?*.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**FAST**

This subcommand acts only when there is more than one variable specified in CLUSOL. FAST=YES speeds up the macro, what can be apparent in case of large data (big matrix MATRIX and many cluster solutions CLUSOL). Do not apply FAST=YES if any CLUSOL variables contain missing values or zeroes. By default/unspecification, FAST=NO.

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). Do not use SELECT IF before the macro because you will ruin the square shape of the input matrix. You may, however, use filtering (FILTER or USE) for case selection. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_DUNN: DUNN CRITERION (SEVERAL TYPES)

Version 2, Jul 2016 (Version 1, Sep 2012). Tested on SPSS Statistics 17, 20, 22.

!KO\_dunn matrix= *VAR1 TO VAR200* /\*Variables constituting matrix of similarities or distances

/\*(name-by-name and/or via “to”)

/clusol= *CLU5\_1 CLU4\_1 CLU3\_1 CLU2\_1*

/\*Variables representing cluster solutions (name-by-name list)

/distb= /\*Formulation of between-cluster distance (see): SINGLE (default), COMPLETE, BAVERAGE,

/\*CENTROID, CROSSPC, HAUSDORFF

/diamw= /\*Formulation of within-cluster "diameter" (see): MXDIST (default), AVDIST, AVDEV.

Minimal specification MATRIX, CLUSOL.

This criterion (Dunn, J.C. Well separated clusters and optimal fuzzy partitions // J Cybernetics, 1974, 95-104) is the ratio, in a given cluster solution of *k* clusters, of the minimal between-cluster distance between clusters considered in pairs to the maximal within-cluster “diameter” among all the clusters.

The greater is this proportion (which can be from zero upwards) the more separated, drawn apart are clusters constituting the cluster partition (so the better it is).

In the original formulation, for between-cluster distance *dij* there the smallest distance between objects of cluster *i* and objects of cluster *j* is taken, i.e., it is the “nearest neighbour” distance between *i* and *j*. And for cluster diameter *diami* there is taken the maximal distance inside a cluster *i*, i.e., the distance between its utmost points. Such index prefers exactly solutions with spatially more separated clusters, to the detriment of solutions having clusters though dense inside but with between-cluster “noise” (in particular, if clusters touch or superimpose even by a single point that spoils the cluster solution at once).

Besides the original version of the index, there were suggested later (Bezdek, C., Pal, N.R. Some new indexes of cluster validity // IEEE Trans. Systems, Man and Cybernetics − B, 28 (1998), 301–315. See also: Vendramin, L., Campello, R.J.G.B., Hruschka, E.R. On the comparison of relative clustering validity criteria // Proceedings of the 2009 SIAM International Conference on Data Mining, 2009, 733-744.) yet other variants, with other definitions for *dij* and *diami*, but with the same formula. Macro !KO\_DUNN implements all those variants of the criterion considered in Vendramin et al. – see subcommands DISTB and DIAMW. These alternative formulations of the criterion (combinedly called sometimes *generalized Dunn index,* GDI) have their own inclinations, and their attitude towards between-cluster noise is more tolerant. It is important to note that values of Dunn criterion of the different types cannot be compared with each other. You may only compare classifications within the criterion of the same type.

Whichever formulation of *dij* and *diami*, the basic formula of the criterion, in which of all *dij* there is taken one minimal and of all *diami* – one maximal, logically presupposes an influence of the magnitude of discrepancy among *dij* and among *diami*, on the criterion. The presence of one disproportionally large (by physical size) cluster in a cluster solution or the tight togetherness of two clusters on the background of well drawn apart rest clusters – spoils the picture (reduces the index). It follows, that Dunn “prefers” partitions with approximately equally separated equally same-sized clusters.

The index is insensitive to proportional transform of proximities, but it reacts to adding of a constant.

Dunn criterion is apt for both distances or similarities (similarities are internally converted by the macro linearly into distances).

The criterion, at least in its original version, is sensitive to the number of objects in data, so the being compared cluster partitions must consist of equal total number of objects *n*.

***Subcommands***

**MATRIX**

Specify variables which are the columns of the distance matrix (zeros must stand on the diagonal, off-diagonal elements must be nonnegative, greater value corresponds to greater dissimilarity) or of the similarity matrix (positive numbers must stand on the diagonal, off-diagonal elements may have any sign, greater value corresponds to greater similarity). The matrix must be square symmetric and must not contain missings. Be aware that the macro does not check the matrix correctness: so take care that it meets the just above described requirements. The structure of a matrix and requrements thereof are shown above in macro [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ). The matrix body found in the working dataset should be square and symmetric, and you must input all the body columns by MATRIX subcommand.

*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?*.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**DISTB**

Select definition of between-cluster distance *dij*:

SINGLE - (also default/unspecification) single linkage or nearest neighbour distance. It is the minimal distance between points (objects) of one cluster and points (objects) of the other cluster. Corresponds to the original version of the criterion.

COMPLETE - complete linkage or farthest neighbour distance. It is the maximal distance between points of one cluster and points of the other cluster.

BAVERAGE - average linkage distance. It is the averaged distance between points of one cluster, from one side, and points of the other cluster, from the other side.

CENTROID - between-centroid distance. It is straight distance between the centroids of two clusters in euclidean space.

CROSSPC - point-centroid crossdistance. It is the sum of distances from points of one cluster to the centroid of the other, plus sum of distances from points of the second cluster to the centroid of the first, and divide by the total number of points in the two clusters.

HAUSDORFF - Hausdorff distance, which is defined as follows. Each point of the first cluster finds for itself the nearest neighbour in the second cluster, and from these (minimal) distances one maximal is taken. The same is done oppositely: each point of the second cluster finds its nearest neighbour in the first cluster, and the largest of these distances is taken. Of the two thus found distances the greater is chosen.

Variants CENTROID and CROSSPC expect that the input proximities are euclidean distances (nonsquared). If your distances are non-euclidean or if the proximities are similarities, the computed result can be informatively invalid; and there also can happen the error of taking of square root of a negative number.

**DIAMW**

Select definition of within-cluster diameter *diami*:

MXDIST - (also default/unspecification) maximal distance inside a cluster, i.e. the distance between most apart points (objects) in it. Corresponds to the original version of the criterion.

AVDIST - mean distance in a cluster, i.e. averaged distance between all points in a cluster taken pairwise.

AVDEV - doubled mean deviation from a cluster centroid, i.e. averaged distance between cluster points and its centroid, multiplied by 2. This option expects that the input proximities are euclidean distances (nonsquared). If your distances are non-euclidean or if the proximities are similarities, the computed result can be informatively invalid; and there also can happen the error of taking of square root of a negative number.

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro switches off dataset weights in the end of its work, so if weighting is important to you, switch it on again. The macro does not obey temporary operations (standing under TEMPORARY). Do not use SELECT IF before the macro because you will ruin the square shape of the input matrix. You may, however, use filtering (FILTER or USE) for case selection. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

# MACRO !KO\_SILHOU: SILHOUETTE STATISTIC (SEVERAL TYPES)

Version 3, Jul 2017 (Version 1, Oct 2010). Tested on SPSS Statistics 17, 20, 22.

*This macro needs SPSS Statistics 17 or higher*

!KO\_silhou matrix= *VAR1 to VAR250* /\*Variables constituting matrix of similarities or distances

/\*(name-by-name and/or via “to”)

/clusol= *CLU6\_2 CLU5\_2 CLU4\_2 CLU3\_2 CLU2\_2*

/\*Variables representing cluster solutions (name-by-name list)

/type= /\*Type of silhouette statistic: AVER (default), NEAR, FARTH, FARNEAR, DEVIAT (see)

/caps= *'nei#' 'sil#'* /\*Two prefixes into variable names: 1st into variable of closest cluster,

/\*2nd into silhouette variable

/single= *0* /\*Silhouette value for a single object: value or SYSMIS (missing) or

/\*IFMERGE or ASSEEDS (see)

/widorig= *-1* /\*Optional, for silhouette width plot: scale origin (default is 0)

/widcoll= /\*Optional, for silhouette width plot: combine clusters with percent less than (value, default is 5).

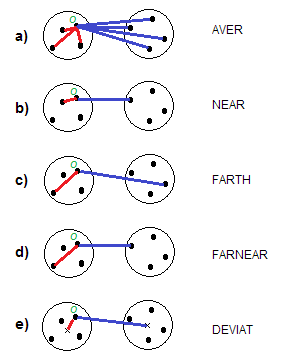
Minimal specification MATRIX, CLUSOL, CAPS, SINGLE.

Silhouette Statistic is a universal clustering criterion which measures the degree of justification of the fact that objects were assigned to the given clusters rather than to the closest neighbour clusters. The great worth of Silhouette statistic is that it is computed even on the level of each object, and therefore it serves as an index that allows to judge how well a given specific object is classified. As for the quality of a cluster solution in whole, usually simply the arithmetic mean of silhouette values of all objects in the given cluster solution serves as a measure of it. For object *o* Silhouette statistic (Kaufman, L., Rousseeuw, P. Finding groups in data: an introduction to cluster analysis. New York, 1990):

where *a(o)* is the remoteness of object *o* from other objects of that same cluster; *b(o)* is the remoteness of object *o* from objects of another cluster, the one closest to object *o*. It is meant that the remoteness *b(o)* is computed between our object and objects of every foreign cluster (there’s *k* clusters in all). The smallest of these computed *k-*1 remotenesses will be *b(o)* taken in the formula. Silhouette statistic can fluctuate from -1 (theoretically the worst clusterization of an object) to 1 (theoretically the best clusterization of an object), and it meaning is: in what degree we lose (or win) in the goodness of the object classifiedness if we transfer it from its actual cluster to another one, the closest to it.

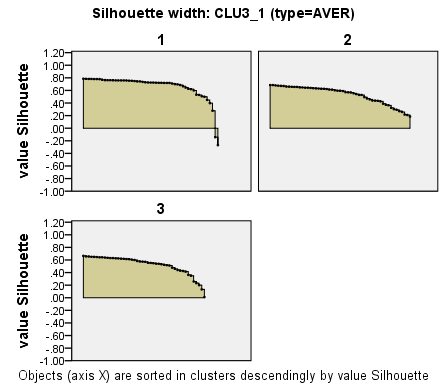
Depending on what to regard as “remoteness”, several types of Silhouette statistic might be obtained:

1. Remoteness as *average* distance: *a(o)* is the average distance from object *o* to other objects of its cluster; *b(o)* is the average distance from object *o* to objects of another cluster, closest to *o*. It is Silhouette statistic of average linkage; it is it what was introduced by Kaufman and Rousseeuw and is usually referred to by speaking of Silhouette Statistic. The macro computes namely this type by default.
2. Remoteness as *minimal* distance: *a(o)* is the distance from object *o* to the nearest object in its cluster; *b(o)* is the distance from object *o* to the nearest object of another cluster, closest to *o*. It is Silhouette statistic of nearest neighbour.
3. Remoteness as *maximal* distance: *a(o)* is the distance from object *o* to the farthest object in its cluster; *b(o)* is the distance from object *o* to the farthest object of another cluster, closest to *o*. It is Silhouette statistic of farthest neighbour.
4. Remoteness *a(o)* is the distance from object *o* to the farthest object in its cluster; *b(o)* is the distance from object *o* to the nearest object of another cluster, closest to *o*. It is mixed variant.
5. Remoteness as distance to geometric *centre*: *a(o)* is the distance from object *o* to the centroid of its cluster; *b(o)* is the distance from object *o* to the centroid of another cluster, closest to *o*. It is Silhouette statistic of deviation.



Should be stressed that values of Silhouette statistics of *different types cannot* be compared with each other: they have different characteristic level of magnitude. For example, the nearest neighbour type has values usually higher than other types have, and the mixed type usually has lower of all values, often quite negative. It is allowable to compare values of different partitions only within one same type of Silhouette statistic. Within any type there preserves the principle: the higher the value, the better is the clusteredness.

The macro computes Silhouette statistic values for objects in given cluster partitions and saves these variables in the working dataset. It also saves variables showing code of the closest cluster. The plot of the averaged Silhouette statistic over a cluster partition is built if there were specified more than one partition in CLUSOL. If there is single cluster partition in CLUSOL, the macro will draw *silhouette width plot* for it: Silhouette statistic values in each cluster separately will line up descending, forming the cluster’s “silhouette”. The higher and the thicker is the cluster silhouette, the better are clustered the objects constituting the cluster. Having opened the plot for editing, you may add the horizontal reference line to it, showing the mean value of the Silhouette statistic, characterizing the cluster solution in total (set Reference line to Mean).



The index is insensitive to proportional transform of proximities, but it reacts to adding of a constant. The criterion (in its original version, TYPE=AVER) is insensitive to the number of objects in data, so the being compared cluster partitions may consist of non-equal total number of objects *n*. Preferences of the criterion to the shape of distribution and to spatial configuration of clusters – see **Table 1**.

Silhouette statistic criterion is apt for both distances or similarities (similarities are internally converted by the macro linearly into distances).

*Using silhouette value for improvement of a clustering*. That for each object the closest to it other cluster is known, opens an opportunity for improving a clustering. Object(s) having Silhouette statistic too low comparatively with other objects of its cluster, can be reassigned by the user to the cluster closest to that object (its code is output by the macro), then rerun the macro to see if that transfer enhanced the object’s Silhouette statistic and the average Silhouette statistic of the entire cluster solution. One might do that transfer several times with the same or different objects (there comes out a kind of a relocating cluster algorithm aiming to improve classification without altering the number of clusters). Average Silhouette statistic ceases to improve soon, but for some while its variance can still be decreasing, i.e. thickness of some cluster silhouettes be increasing.

***Subcommands***

**MATRIX**

Specify variables which are the columns of the distance matrix (zeros must stand on the diagonal, off-diagonal elements must be nonnegative, greater value corresponds to greater dissimilarity) or of the similarity matrix (positive numbers must stand on the diagonal, off-diagonal elements may have any sign, greater value corresponds to greater similarity). The matrix must be square symmetric and must not contain missings. Be aware that the macro does not check the matrix correctness: so take care that it meets the just above described requirements. The structure of a matrix and requrements thereof are shown above in macro [!KO\_CALHARM](#_МАКРОС_!CALHARM:_КРИТЕРИЙ). The matrix body found in the working dataset should be square and symmetric, and you must input all the body columns by MATRIX subcommand.

*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?*.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**TYPE**

Type of Silhouette statistic:

AVER - (also default/unspecification) average linkage. This is the classic version of Silhouette statistic as introduced by its inventors; it can be considered most universal.

NEAR - nearest neighbour. Because this version positively reacts to increase of *n* in clusters it can reward solutions with less number of clusters approximately equally populated with objects.

FARTH - farthest neighbour. Because this version negatively reacts to increase of *n* in clusters it can reward solutions with greater number of clusters.

FARNEAR - mixed type: farthest neighbour in own cluster, nearest – in the neighbourhood cluster. This variant can reward solutions with greater number of clusters.

DEVIAT - deviations from centroid. This version (also known as “Simplified Silhouette index”) expects that the input proximities are euclidean distances (nonsquared). If your distances are non-euclidean, the computed result can be informatively invalid; and there also can happen the error of taking of square root of a negative number (in the latter case results won’t be computed at all). If the proximities are similarities this version is inapplicable.

**CAPS**

Names of the variables being saved by the macro in the working dataset are concatenated of names in CLUSOL list and a prefix. Specify two prefixes – the first one for the variables showing the code of the closest cluster, and the second one for the variables with Silhouette statistic value. The prefixes may be put in quotes or apostrophes (it is desirable when you use period, as a separator, in the end of a prefix). Be aware that at a repeated run of the macro the variables saved earlier into the dataset and having the same names will be replaced, if you don’t change prefixes for new ones.

**SINGLE**

This is a required subcommand which influences if there are singleton objects (i.e. clusters consisting of one object) in a cluster solution. It sets, what value of Silhouette statistic to assign for such objects. For they have undefined *a(o)* value because there are no other objects in the cluster. There emerges the question what value of *sil* statistic to assign to singletons. Choose:

*value* - specify arbitrary value of *sil* between -1 and 1, it will be given to all single objects *s*. Authors of the original criterion recommended value 0, and it is used in many packages by default. But that is specifically for the original version TYPE=AVER. For other versions, 0 does not necessarily appear to be optimal suggestion[[2]](#footnote-2).

SYSMIS - do not set a value, the value *sil(s)* will be system missing. Consequently, singleton objects will not be taken into account when computing the average value of *sil* for the whole cluster solution, and the value will be equal to the average *sil* only of nonsingleton objects. So use this option if it doesn’t bother you – the magnitude of the proportion of objects united in clusters in a cluster solution.

IFMERGE - each singleton object *s* will be calculated its own value of the criterion, by the following logistic function:

where *b(s)* is *b* of object *s* (see criterion formula); and are the mean and st. deviation of values *a* among nonsingleton objects, and is the average silhouette value among them. Coefficient *w* is assumed to be (you may change *w* for another value in the first command of macro body DEFINE… /COEF… !DEFAULT). The meaning of the formula is this. A singleton object will receive *sil* close to -1 (worst possible value) if the object is located near to any clusters or singleton objects: since *b(s)* is small; and it will receive *sil* close to the averaged *sil* of nonsingleton objects (that is, indifferent value) if it is located far away from all clusters/objects: since *b(s)* is big. That is, a singleton is punished by low *sil* when it could be merged with somebody (but it is not merged). While if it is an outcast (so could not be merged with anybody) then it may be ignored and we may suppose it does not spoil the cluster solution, and so it is assigned the value of *sil* close to the average of the clustered (merged) objects. At *b(s)=* a singleton object will receive *sil* value in the middle between -1 and .

ASSEEDS - (this option is only for TYPE=AVER or DEVIAT) all singleton objects *s* are prescribed the same value *a(s)*, after what *sil(s)* of an object is computed by the criterion formula. The value *a(s)* is determined so: in each nonsingleton cluster the average pairwise distance between its objects is computed; weighted mean of these averages with weight = 1/(*n* in cluster) is the scalar *a(s)* (with TYPE=DEVIAT the obtained *a(s)* is divided by 2 because in this case it is the distance to centroid). The meaning of the approach is this. Singleton object *s* is considered as the initiator of a new, its own cluster, and therefore it is now as if in a state of expecting for itself an object to pair with, so that the new cluster can emerge. The needed *a(s)* is the expected distance between *s* and its pair, not yet existing object, and it is assumed to be the average distance in clusters already existing. The weight inverse to the frequency in a cluster means that in calculation of *a(s)* we take as more important “opinion” of clusters low-populated with objects, because it is permissible to decide that such clusters are more “young by time of formation” and, therefore, clusters not formed yet – which founders-to-be are singleton objects, as we see it – are more similar to those ones than to “old” (much populated with objects) clusters.

**WIDORIG**

This subcommand acts only when there is single variable in CLUSOL. The subcommand sets the level of origin of the silhouette value on silhouette width plot. Specify number from -1 to less than 1. By default, 0 is assumed. Points (objects) with value lower than the origin, if there are any, will be found below the horizontal line corresponding to the level of origin. This subcommand does not affect the result, but only the look of the mentioned plot. You can always change the origin level in a plot already drawn, by opening it to edit and setting origin for Y axis.

**WIDCOLL**

This subcommand acts only when there is single variable in CLUSOL. The subcommand sets the frequency threshold in clusters, below which silhouettes of such small-populated clusters will be combined in one category “Other (Silhouette mix)” on silhouette width plot. Specify number from 0 to 100. This number is the percent of frequency in a cluster of the total count of objects. The default is set to 5. I.e., clusters each having less than 5% of all the sample will be merged on the plot in one frame “Other (Silhouette mix)”. If you want to see separately silhouettes of all clusters, even singleton objects, specify WIDCOLL=0.

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). Do not use SELECT IF before the macro because you will ruin the square shape of the input matrix. You may, however, use filtering (FILTER or USE) for case selection. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

***Some questions***

*Silhouette width plot appears empty*. The format of the cluster membership variable indicated in CLUSOL does not correspond to the number of decimal digits of its cluster codes. Usually codes are integers, however, when you have fractional numbers, the integer format will not allow the plot to display. Set the variable fractional format to properly display digits after the decimal point.

# MACRO !KO\_SILDEV: SILHOUETTE STATISTIC (TYPE DEVIAT, INPUT – VARIABLES)

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

*This macro needs SPSS Statistics 17 or higher*

!KO\_sildev vars= *V1 to V10* /\*Quantitative variables by which to compare clustering solutions

/\*(name-by-name and/or via “to”)

/clusol= *CLU6\_2 CLU5\_2 CLU4\_2 CLU3\_2 CLU2\_2*

/\*Variables representing cluster solutions (name-by-name list)

/caps= *'nei#' 'sil#'* /\*Two prefixes into variable names: 1st into variable of closest cluster,

/\*2nd into silhouette variable

/single= SYSMIS /\*Silhouette value for a single object: value or SYSMIS (missing) or

/\*IFMERGE (see)

/widorig= /\*Optional, for silhouette width plot: scale origin (default is 0)

/widcoll= /\*Optional, for silhouette width plot: combine clusters with percent less than (value, default is 5).

Minimal specification VARS, CLUSOL, CAPS, SINGLE.

This macro computes Silhouette statistic of deviation-from-centroid type (TYPE=DEVIAT, see macro !KO\_SILHOU), also known as “Simplified Silhouette index”. It takes as input not distance matrix but variables. Result is the same as !KO\_SILHOU outputs.

The macro creates temporary variable *CASENUM#* in the input dataset, so make sure you have no such variable, to avoid overwriting.

***Subcommands***

**VARS**

Variables (quantitative) by which you need to assess cluster partitions of cases. The macro excludes cases with missing values listwise.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

**CAPS**

This subcommand is the same as in [!KO\_SILHOU](#_MACRO_!SILHOU:_SILHOUETTE_1).

**SINGLE**

This is a required subcommand which influences if there are singleton objects (i.e. clusters consisting of one object) in a cluster solution. It sets what value of Silhouette statistic to assign for such objects. For they have undefined *a(o)* value because there are no other objects in the cluster. There emerges the question what value of *sil* statistic to assign to singletons. Choose:

*value* - specify arbitrary value of *sil* between -1 and 1, it will be given to all single objects *s*. Authors of the original criterion recommended value 0 for it, but it is not necessarily the optimal suggestion for type DEVIAT. (More details – see s/c SINGLE in macro !KO\_SILHOU.)

SYSMIS - do not set a value, the value *sil(s)* will be system missing. Consequently, singleton objects will not be taken into account when computing the average value of *sil* for the whole cluster solution, and the value will be equal to the average *sil* only of nonsingleton objects. So use this option if it doesn’t bother you – the magnitude of the proportion of objects united in clusters in a cluster solution.

IFMERGE - each singleton object *s* will be calculated its own value of the criterion, by the following logistic function:

where *b(s)* is *b* of object *s* (see criterion formula); and are the mean and st. deviation of values *a* among nonsingleton objects, and is the average silhouette value among them. Coefficient *w* is assumed to be (you may change *w* for another value in the first command of macro body DEFINE… /COEF… !DEFAULT). The meaning of the formula is this. A singleton object will receive *sil* close to -1 (worst possible value) if the object is located near to any clusters or singleton objects: since *b(s)* is small; and it will receive *sil* close to the averaged *sil* of nonsingleton objects (that is, indifferent value) if it is located far away from all clusters/objects: since *b(s)* is big. That is, a singleton is punished by low *sil* when it could be merged with somebody (but it is not merged). While if it is an outcast (so could not be merged with anybody) then it may be ignored and we may suppose it does not spoil the cluster solution, and so it is assigned the value of *sil* close to the average of the clustered (merged) objects. At *b(s)=* a singleton object will receive *sil* value in the middle between -1 and .

This macro does not support option SINGLE=ASSEEDS.

**WIDORIG**, **WIDCOLL**

These subcommands are the same as in [!KO\_SILHOU](#_MACRO_!SILHOU:_SILHOUETTE_1).

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). To select cases for the analysis you may use filtering (FILTER or USE) and SELECT IF. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

***Some questions***

*Silhouette width plot appears empty*. See same question in [!KO\_SILHOU](#_MACRO_!SILHOU:_SILHOUETTE_1).

# MACRO !KO\_AICBIC: INFORMATION CRITERIA AIC and BIC

Version 1, March 2007. Tested on SPSS 11.5, 13, 15

!KO\_aicbic scavars= *zq8\_1 to zq8\_8* /\*Scale variables (standardized recommended),

/\*name-by-name and/or via “to”

/catvars= *q3 q12* /\*Categorical variables, name-by-name and/or via “to”

/clusol= *clu6 clu5 clu4 clu3 clu2*

/\*Variables representing cluster solutions (name-by-name list).

Minimal specification CLUSOL and at least one of SCAVARS, CATVARS.

Akaike Information Criterion (AIC) and Schwarz Bayesian Information Criterion (BIC) are widely used for assessment of informativeness of statistical models. The informativeness is higher as the fewer are there parameters and the more accurately they predict. SPSS procedure of TwoStep Cluster Analysis computes synthetic – allowing to account both continuous (scale) and categorical variables – indices AIC and BIC to determine initial (reference for the algorithm) number of clusters during clustering in “auto” regime. The macro uses AIC and BIC formulas borrowed from the description of TwoStep Cluster Analysis[[3]](#footnote-3) (IBM SPSS Statistics 20 Algorithms, IBM Corporation, 2011. See also: Fraley, C., Raftery, A.E. How many clusters? Which clustering method? Answers via model-based cluster analysis // Computer Journal, 1998, 4, 578–588).

where *m* is the number of parameters in a being considered *k*-cluster solution and *Li* is the log-likelihood of cluster *i* in it.

where *ni* is the frequency in cluster *i*, *E1q* is the entropy of cluster *i* by scale variable *q* (*p1* scale variables in all), *E2r* is the entropy of cluster *i* by nominal variable *r* (*p2* nominal variables in all); *lr* is the number of categories in variable *r*. Entropy of a cluster by a variable:

(*sq2* is computed on “df=n” variance in variable *q* in cluster *i*; while *c* is constant added to escape logarithming of zero and took as the variance of variable *q* in the whole sample, so if input variables are standardized then *c*=1)

(*ntr* is the number of objects in category *t* of variable *r*, in cluster *i*)

As it is seen, the likelihood function is combined of variation (entropy) of continuous variables and equidistribution (entropy) of categorical variables; when summed up of all clusters of the given partition it expresses by itself the within-cluster nondensity of that cluster solution.

AIC and BIC are kin of each other and often give close results in comparisons of partitions quality. Both criteria (since they penalize for the “excess” of the number of parameters) give preference, under all else conditions equal, to small number of clusters (and that tendency is stronger in BIC), therefore they are recommended in cases where there is a request “less clusters desirably”. The lower the index value the better is cluster solution.

Both criteria are insensitive to inequality of variances of scale variables; while they depend by their level (i.e. linearly with slope 1) on the general level of magnitude of the variances in scale variables.

Like Ratkowsky-Lance criterion, the given indices may be called univariate-summarizing and not multivariate, because they compute their principal measure (entropy in this case) by each variable separately prior combining it in the single value. In attitudes to spatial characteristics of clusters and to the shape of distribution in clusters these criteria are similar to Ratkowsky-Lance. However, one should mind that Ratkowsky-Lance is sensitive to inequality of variances of scale variables, because it processes them “as is”. AIC/BIC, as already said, implicitly equalize variances of these variables (squeeze/stretch the data cloud along the variable axes up to attainment of variance equality). Still, if one speaks of spatial features of clusters in the *input data*, the tendencies in that respect are similar in AIC/BIC to those in Ratkowsky-Lance. AIC/BIC are somewhat more robust to rotation of the whole cloud than Ratkowsky-Lance, but they would prefer, too, the variances to be same along variable axes.

The criteria are sensitive to the number of objects in data, so the being compared cluster partitions must consist of equal total number of objects *n*.

***Subcommands***

**SCAVARS, CATVARS**

In SCAVARS specify scale variables. In CATVARS specify categorical nominal variables (numeric). Both lists can be written name-by-name and/or via “to”. You must specify at least one of the two lists. The macro excludes missing values listwise: if at least in one variable a case is missing value, that case is excluded from calculations by all the variables.

It doesn’t matter whether variances in scale variables are equal, since the criterion formula equalizes the variances. However, the overall level of variance of scale variables influences the criterion’s value. Therefore, it is recommended to enter the variables as *standardized* – all variances brought to 1 (you may standardize variables with the help of Descriptive Statistics – Descriptives). In this case, contribution of scale and of categorical variables may be considered by and large balanced between the two types of variables.

**CLUSOL**

Name-by-name list of case-grouping variables – see subcommand description above in macro [!KO\_CALHARV](#_MACRO_!KO_CALHARV:_CALINSKI–HARABAS).

***Special regimes***

The macro does not obey weighting of the dataset (however, it doesn’t take in the procedure cases with missing and nonpositive weights) and is not suited for the split state of the dataset (SPLIT FILE). The macro does not obey temporary operations (standing under TEMPORARY). To select cases for the analysis you may use filtering (FILTER or USE) and SELECT IF. Besides that, you can exclude from the analysis different cases in different cluster solutions by assigning them missing or zero codes (see CLUSOL s/c about it).

1. For example: let there be 2 round clusters in a 2-variable space (distance between them is 1), or 3 such clusters (triangle of them, distance between them is 1), or 4 such clusters (square of them, distance between the neighbours is 1). Specifics in disposition of the clusters are not the same in these three configurations (in the 2-cluster the data cloud is oblong; in the 4-cluster there exist between-cluster distances greater than 1) which complicates it to regard the three configurations equally internally valid by some “universal” internal validity. The very *universal* internal validity is what doesn’t exist. Some clustering criteria will respond to the above not sameness in the configurations by giving preference to one or another of them (and this is what enters the concept of the bias of a criterion towards k), while others – will not. [↑](#footnote-ref-1)
2. If the implicit meaning of setting *sil(s)=*0 for every singleton object *s* is in defining of *a(s)* and *b(s)* to be the remoteness of *s* from two *imaginary* closest to it clusters *A* and *B*, which are equi-distanced from it: *a(s)=b(s)*, and *that’s why* *s* has remained single (for justifiedness to assign it to *A* is exactly as justifiedness to assign it to *B*), – then, apparently, value *sil(s)=*0 is logically correct – as neutral, i.e. expressing such “undecided” status of *s* – for any type of Silhouette statistic in which *a(.)* and *b(.)* are samely defined distance functions (i.e., for instance, both *a(.)* and *b(.)* are distances to a nearest neighbour, or both are distances to a centroid, etc.). Of five types TYPE only TYPE=FARNEAR contains *a(.)* and *b(.)* that are not samely defined functions. Thus, only for TYPE=FARNEAR *sil(s)=*0 is obviously an incorrect setting. [↑](#footnote-ref-2)
3. This doesn’t mean the indices’ values output by the TwoStep cluster analysis procedure will always coincide with such computed by the macro. The thing is that SPSS TwoStep Cluster algorithm computes AIC or BIC within the flow of clustering (so those are “tentative” values) and not after clustering (i.e. from the ready cluster partition). [↑](#footnote-ref-3)