Cluster Validation for Mixed-Type Data

Rabea Aschenbruck and Gero Szepannek

Abstract For cluster analysis based on mixed-type data (i.e. data consisting of numerical and categorical variables), comparatively few clustering methods are available. One popular approach to deal with this kind of problems is an extension of the *k-means* algorithm (Huang, 1998), the so-called *k-prototype* algorithm, which is implemented in the R package `clustMixType` (Szepannek and Aschenbruck, 2019).

It is further known that the selection of a suitable number of clusters *k* is particularly crucial in partitioning cluster procedures. Many implementations of cluster validation indices in R are not suitable for mixed-type data. This paper examines the transferability of validation indices, such as the Gamma index, Average Silhouette Width or Dunn index to mixed-type data. Furthermore, the R package `clustMixType` is extended by these indices and their application is demonstrated. Finally, the behaviour of the adapted indices is tested by a short simulation study using different data scenarios.

Rabea Aschenbruck · Gero Szepannek
University of Applied Sciences Stralsund, Zur Schwedenschanze 15, D-18435 Stralsund, Germany
🖂 `rabea.aschenbruck@hochschule-stralsund.de`
🖂 `gero.szepannek@hochschule-stralsund.de`

ARCHIVES OF DATA SCIENCE, SERIES A
(ONLINE FIRST) DOI: 10.5445/KSP/1000098011/02
KIT SCIENTIFIC PUBLISHING ISSN 2363-9881
Vol. 6, No. 1, 2020

CC BY-SA

1 Introduction

In practice, users are often confronted with mixed-type data (i.e. data consisting of numerical and categorical variables), while in theoretical development this data situation is neglected in many cases. In cluster analysis based on mixed-type data only few cluster methods are available (a comprehensive overview is given in Ahmad and Khan, 2019). In this paper mixed-type data are objects with l numerical and $m-l$ categorical features. A popular distance measure for categorical and numerical features is based on the similarity measure of Gower (Gower, 1971). Another intuitive distance between two mixed-type objects is given by

$$d_{MT}(X,Y) = \sum_{j=1}^{l}(x_j - y_j)^2 + \lambda \sum_{j=l+1}^{m} \delta(x_j, y_j) \tag{1}$$

with factor $\lambda > 0$ and *Simple Matching*

$$\delta(x_j, y_j) = \begin{cases} 0 & if\ x_j = y_j, \\ 1 & if\ x_j \neq y_j \end{cases} \tag{2}$$

which is considered in this paper. The choice of λ strongly influences the distances but is not the scope of this paper, where the defaults were used (Szepannek, 2018; Huang, 1997).

A popular approach to cluster mixed-type data using the above-mentioned distance is the *k-prototype* algorithm, an extension to the *k-means* algorithm proposed by Huang (1998). This algorithm minimises

$$\sum_{k=1}^{q}\sum_{i=1}^{n} w_{i,k}\, d_{MT}(X_i, c_k), \tag{3}$$

where q is the number of clusters and n is the number of objects. For the entries in matrix $[w]_{i=1,\dots,n;\ k=1,\dots,q}$ we assume $\sum_{k=1}^{q} w_{i,k} = 1$ and $w_{i,k} \in \{0, 1\}$. The prototype of cluster k is noted with c_k, where the prototype is the mean for the numerical features and the mode for the categorical features. The cluster algorithm is implemented in the R package `clustMixType` (Szepannek and Aschenbruck, 2019). For an enumeration of other algorithms for clustering mixed-type data see Szepannek (2018).

As for the *k-means* algorithm the number of clusters must be prespecified in advance. Validation methods have been identified for the *k-prototype* algorithm which enable the rating of clusters and the determination of the optimal number of clusters. This paper examines the transformation of validation indices for clustering mixed-type data. The transformed indices will extend the R package `clustMixType`.

First, the cluster indices to be examined are presented in Section 2. The extension of the R package `clustMixType` with the transferred indices is shown in Section 3. Then a simulation study is conducted in order to compare the performance of the implemented indices (Section 4). This paper ends with a summary and outlook in Section 5.

2 Cluster Indices to Be Examined

An overview of existing validation indices is given e.g. in Halkidi et al. (2016) or Desgraupes (2018). The selection of the validation indices under consideration in this paper is mainly based on the publication on the `NbClust` R package (Charrad et al., 2014) providing a huge list of well-known cluster validation indices for numeric data. The functionality of this R package can't be used for mixed-type data, e.g. for the *k-prototype* algorithm. In the following the straightforward transformation of validation indices based on distances is presented.

Cindex, McClain Index and Ptbiserial Index

The three indices *Cindex*, *McClain index* and *Ptbiserial index* are all based on distances between the objects to be clustered. These indices may readily be generalised for mixed-type data using the distance for mixed-type objects in equation (1). If N_w (resp. N_b) denotes the number of pairs in the same (resp. different) cluster, then S_w (resp. S_b) is the sum of the N_w within-cluster distances (resp. N_b between-cluster distances).

The standard deviation of all distances is denoted by s_d and the total number of pairs of objects by $N_t = \frac{n(n-1)}{2}$. In order to determine S_{min} and S_{max}, all N_t distances between all pairs of objects have to be calculated first: S_{min} is the sum of the N_w smallest distances and S_{max} the sum of the N_w largest distances.

Using the notation explained above, the indices are defined as:

$$v_{Cindex} = \frac{S_w - S_{min}}{S_{max} - S_{min}}, \tag{4}$$

$$v_{McClain} = \frac{\bar{S}_w}{\bar{S}_b} = \frac{S_w / N_w}{S_b / N_b}, \tag{5}$$

$$v_{Ptbiserial} = \frac{(\bar{S}_b - \bar{S}_w) \cdot \sqrt{(N_w \cdot N_b)/N_t^2}}{s_d}. \tag{6}$$

Gamma Index, Gplus Index and Tau Index

Calculation of the *Gamma index*, *Gplus index* and *Tau index* is based on concordant and discordant comparisons: Every within-cluster distance has to be compared with every between-cluster distance, which results in high computational costs. The number of comparisons is given by the total number of pairs of pairs of objects and is denoted by $N_D = \frac{N_t(N_t-1)}{2}$. If a within-cluster distance is strictly smaller than a between-cluster distance, the comparison is called concordant. The number of concordant comparisons is called $s(+)$. The number of discordant comparisons is denoted by $s(-)$, where a comparison is called discordant if a within-cluster distance is stricly greater than a between-cluster distance. Calculation of the *Tau index* further requires the number of comparisons of two pairs of objects where both pairs are within- or between cluster comparisons (denoted by t). The definitions of the above mentioned indices is given by:

$$v_{Gamma} = \frac{s(+) - s(-)}{s(+) + s(-)}, \tag{7}$$

$$v_{Gplus} = \frac{s(-)}{N_D}, \tag{8}$$

$$v_{Tau} = \frac{s(+) - s(-)}{\sqrt{(N_D - t)N_D}}. \tag{9}$$

Dunn Index

Calculation of the *Dunn index* requires the determination of the distance between two clusters C_i and C_j, which is given by the distance between their closest points $d(C_i, C_j) = \min_{X \in C_i, Y \in C_j} d_{MT}(X, Y)$, with distance d_{MT} for mixed-type objects as mentioned in equation (1). The diameter of a cluster C_K ist defined as $diam(C_k) = \max_{X,Y \in C_k} d_{MT}(X, Y)$. The *Dunn index* is defined by:

$$v_{Dunn} = \frac{\min_{1 \le i < j \le q} d(C_i, C_j)}{\max_{1 \le k \le q} diam(C_k)}. \tag{10}$$

Silhouette Index

The *Silhouette index* (also known as *Average Silhouette Width*) relates the average within-cluster distance for the cluster C_A assigned to the object X_i $a(X_i) = \frac{1}{n_A - 1} \sum_{j \in \{C_A \setminus X_i\}} d(X_i, X_j)$ to the best alternative. The number of objects in cluster C_A is denoted by n_A. In addition, all alternative cluster assignments are considered and their average within-cluster distance is determined. The best alternative is denoted by $b(X_i) = \min_{S \neq A} \frac{1}{n_S} \sum_{j \in C_S} d(X_i, X_j)$, where n_S is the number of objects in cluster C_S. The definition of the *Silhouette index* then looks as follows:

$$v_{Silhouette} = \frac{1}{n} \sum_{i=1}^{n} \frac{b(X_i) - a(X_i)}{max(a(X_i), b(X_i))}. \tag{11}$$

Overview of the Transformed Indices and Their Optimality Criteria

The transformation of some other well-known indices is not straightforward, e. g., the indices based on within-group dispersion matrix (*Calinski-Harabasz index*, *Duda index*, *Friedman index* or *Pseudot2 index*), *CCC index* (needs calculation of $X^T X$), *Frey index* (only for hierarchical cluster methods) or *SDindex* and *SDbw index* (both based on the variance of the variables) (formulas and definitions in Charrad et al., 2014). Extensions of these indices to mixed-type data could be a subject of further research. In Table 1 the validation indices, which were transferred for rating the *k-prototype* solution for mixed-type data are shown along with their range as well as the optimality criterion.

Table 1: Overview of the transformed indices and their optimality criteria.

Index Name	**Index Range**	**Criterion of Optimality**
Cindex	[0, 1]	minimum
Dunn index	$[0, +\infty)$	maximum
Gamma index	$[-1, +1]$	maximum
Gplus index	$[0, \frac{N_w \cdot N_b}{N_D}]$	minimum
McClain index	$[0, +\infty)$	minimum
Ptbiserial index	$(-\infty, +\infty)$	maximum
Silhouette index	$[-1, +1]$	maximum
Tau index	$(-1, +1)$	maximum

3 Extension of R Package `clustMixType`

As already mentioned, the `clustMixType` R package has been extended to include the transferred indices for mixed-type data in version 0.2-1. The output of the R function `kproto()` is an object of class `kproto`, containing a cluster partition generated by the *k-prototype* algorithm. There are two applications for each index implemented in the R package:

1. Call the function of your preferred validation index on a `kproto` object and you will get the index value **rating your cluster partition**. Note that the function `kproto()` must be called with the input parameter `keep.data = TRUE`.
2. You can pass your data and the search range for the optimal number of clusters. Then you will **get the optimum number of clusters** and also a named vector with the index values for the search range.

Additional information on the implemented validation indices can be found in the help file of the R package `clustMixType`.

In Listing 1 the application of the implemented R function for the *McClain index* of both cases is shown exemplarily: Rating a cluster partition (with prespecified number of clusters) based on an existing `kproto` object and receiving an index-proposed number of clusters. In the latter case the function must be called with a specified search range as an input parameter.

Listing 1: Sample Application of the R function for McClain Index.

```r
> # data generation with two categorical and two numerical variables
> n   <- 10
> prb <- 0.99
> muk <- 2.5
> x1 <- sample(c("A", "B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
> x1 <- as.factor(c(x1, sample(c("A", "B"), 2*n, replace = TRUE, prob = c(1-prb, prb))))
> x2 <- sample(c("A", "B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
> x2 <- as.factor(c(x2, sample(c("A", "B"), 2*n, replace = TRUE, prob = c(1-prb, prb))))
> x3 <- c(rnorm(n, -muk), rnorm(n, muk), rnorm(n, -muk), rnorm(n, muk))
> x4 <- c(rnorm(n, -muk), rnorm(n, muk), rnorm(n, -muk), rnorm(n, muk))
> x <- data.frame(x1,x2,x3,x4)
>
> # apply k prototyps on sampled data x
> kpres <- kproto(x, 4, keep.data = TRUE)
>
> # application 1: calculate the index-value of one kproto object/cluster partition
> # function output is the index-value for the given cluster partition
> mcclain_kproto(object = kpres)
[1] 0.06946939
>
> # application 2: calculate optimal number of clusters
> # function output is a list with the index-optimal number of clusters ($k_opt)
> #   and the index-values of the considered cluster-sizes ($indices)
> mcclain_kproto(data = x, k = 3:5, nstart = 5, verbose = FALSE)
$k_opt
[1] 5

$indices
         3          4          5
0.33311849 0.06946939 0.06919922
```

4 Simulation Study

In order to compare the performance of the aforementioned indices a short simulation study on different data situations was conducted.

Procedure of the Simulation Study

For the simulation study we have chosen the simulation structure shown in Listing 2. The data to be clustered were also simulated: Different metacharacteristics were varied in order to mimic potential data situations (shown in Table 2).

Datasets were created for every combination of the specifications mentioned above. For these datasets the *k-prototype* cluster partitions with number of clusters between 2 and $k_{mx} = 10$ were determined. For every cluster partition all eight indices were calculated to suggest the optimal number of clusters. The whole procedure was repeated 10 times.

Listing 2: Pseudo Code: Procedure of the Simulation Study.

```
 1 > # function to generate data to the according parameters (cf. table 2)
 2 > create_data <- function(nB, nC, nV, CAe){
 3 >   n  <- nB/nC
 4 >   mu <- 0.125
 5 >
 6 >   if(nC == 2){
 7 >
 8 >     x1 <- c(sample(c("A", "B"), n, replace = TRUE, prob = c(CAe, 1-CAe)),
 9 >             sample(c("A", "B"), n, replace = TRUE, prob = c(1-CAe, CAe)))
10 >     y1 <- c(rnorm(n, mean = mu, sd = 1-CAe), rnorm(n, mean = -mu, sd = 1-CAe))
11 >     data <- data.frame(x1, y1)
12 >
13 >     if(nV > 1){
14 >       for(ci in 2:nV){
15 >         xi <- c(sample(c("A", "B"), n, replace = TRUE, prob = c(CAe, 1-CAe)),
16 >                 sample(c("A", "B"), n, replace = TRUE, prob = c(1-CAe, CAe)))
17 >         yi <- c(rnorm(n, mean = mu, sd = 1-CAe), rnorm(n, mean = -mu, sd = 1-CAe))
18 >         data <- data.frame(data, xi, yi)
19 >       }
20 >     }
21 >
22 >   }else{  #nC == 4
23 >
24 >     x1 <- c(sample(c("A", "B"), 2*n, replace = TRUE, prob = c(CAe, 1-CAe)),
25 >             sample(c("A" ,"B"), 2*n, replace = TRUE, prob = c(1-CAe, CAe)))
26 >     y1 <- c(rnorm(n, mean = mu, sd = 1-CAe), rnorm(n, mean = -mu, sd = 1-CAe),
27 >             rnorm(n, mean = mu, sd = 1-CAe), rnorm(n, mean = -mu, sd = 1-CAe))
28 >     data <- data.frame(x1, y1)
29 >
30 >     if(nV > 1){
31 >       for(ci in 2:nV){
32 >         xi <- c(sample(c("A", "B"), 2*n, replace = TRUE, prob = c(CAe, 1-CAe)),
33 >                 sample(c("A", "B"), 2*n, replace = TRUE, prob = c(1-CAe, CAe)))
34 >         yi <- c(rnorm(n, mean = mu, sd = 1-CAe), rnorm(n, mean = -mu, sd = 1-CAe),
35 >                 rnorm(n, mean = mu, sd = 1-CAe), rnorm(n, mean = -mu, sd = 1-CAe))
36 >         data <- data.frame(data, xi, yi)
37 >       }
38 >     }
39 >
40 >   }
41 >
42 >   return(data)
43 > }
44 >
45 > # simulation study
46 > k_mx <- 10          # cluster partitions with a maximum number of clusters of 10
47 >
48 > # repeating the cluster validation procedure for the 8 indices on all data
49 > #   situations (cf. table 2) 10 times
50 > for(i in 1:10){
51 >
52 >   # generating data with function as defined in line 2
53 >   data <- create_data(nB, nC, nV, CAe)
54 >
55 >   # for all 8 indices
56 >   INDEXNAME_val <- rep(NA, k_mx)
57 >
58 >   for(k in 2:k_mx){
59 >
60 >     # applying the k prototype algorithm to cluster the given data
61 >     kpres <- kproto(x = data, k = k, nstart = 20, keep.data = TRUE)
62 >
63 >     # calculating the index-values for the eight cluster indices
64 >     INDEXNAME_val[k] <- INDEXNAME_kproto(object = kpres)
65 >   }
66 >
67 >   # determining the index-optimal k for every index depending on the optimality criterion
68 >   INDEXNAME <- max(INDEXNAME_val) | min(INDEXNAME_val)
69 > }
```

Table 2: Variables of the generated data sets and their specifications.

Variable	**Explanation**	**Specification**
`nB`	number of objects	200, 400
`nC`	number of clusters (of equal size)	2, 4
`nV`	number of numerical/categorical features	1, 2, 4
`CAe`	similarity of objects in different clusters	1, 0.98

Table 3: Results of the Simulation Study.

	`nB`	`nV`	`CAe`	Cindex	Dunn	Gamma	Gplus	McClain	Ptbiserial	Silhouette	Tau
				Mode of the 10 index-based k_{opt}							
number of clusters nC = 2	200	1	1	2	NA	2	2	2	2	2	2
	200	2	1	2	NA	2	2	2	2	2	2
	200	4	1	2	NA	2	2	2	2	2	2
	200	1	0.98	2 $\frac{9}{10}$	2 $\frac{9}{10}$	2 $\frac{9}{10}$	2 $\frac{9}{10}$	10 $\frac{8}{10}$	2	2 $\frac{7}{10}$	2
	200	2	0.98	2	2	2	2	7/9 $\frac{3}{10}$	2	2	2
	200	4	0.98	2	2	2	2	3 $\frac{6}{10}$	2	2	2
	400	1	1	2	NA	2	2	2	2	2	2
	400	2	1	2	NA	2	2	2	2	2	2
	400	4	1	2	NA	2	2	2	2	2	2
	400	1	0.98	2 $\frac{9}{10}$	2/3 $\frac{5}{10}$	2 $\frac{9}{10}$	2 $\frac{9}{10}$	9 $\frac{4}{10}$	2	2/3 $\frac{5}{10}$	2
	400	2	0.98	2	2	2	2	10 $\frac{3}{10}$	2	2	2
	400	4	0.98	2	2	2	2	3 $\frac{6}{10}$	2	2	2
number of clusters nC = 4	200	1	1	3	3	4	4	4	3	4	3
	200	2	1	3	3	4	4	4	3	4	3
	200	4	1	3	3	4	4	4	3	4	3
	200	1	0.98	4	4	4	4	10 $\frac{9}{10}$	2 $\frac{4}{10}$	4	2 $\frac{5}{10}$
	200	2	0.98	2 $\frac{9}{10}$	2 $\frac{9}{10}$	2 $\frac{9}{10}$	2 $\frac{9}{10}$	8 $\frac{4}{10}$	2 $\frac{9}{10}$	5 $\frac{7}{10}$	2 $\frac{9}{10}$
	200	4	0.98	2	2	4 $\frac{6}{10}$	4 $\frac{8}{10}$	9/10 $\frac{3}{10}$	2	4	2
	400	1	1	3	3	4	4	4	3	4	3
	400	2	1	3	3	4	4	4	3	4	3
	400	4	1	3	3	4	4	4	3	4	3
	400	1	0.98	4	4	4	4	10	4 $\frac{9}{10}$	4	4 $\frac{9}{10}$
	400	2	0.98	2	2	2	2	9/10 $\frac{4}{10}$	2	4 $\frac{6}{10}$	2
	400	4	0.98	2	2	2 $\frac{6}{10}$	2/4 $\frac{5}{10}$	8 $\frac{7}{10}$	2	4	2

Results of the Index-Performance

Table 3 (top) summarizes the cluster results for two clusters in terms of the mode of the ten repetitions. If there was not the same number of clusters proposed in the 10 iterations the frequency of the mode is printed in gray. Obviously, the *Dunn index* had problems in data situations with only identical objects in the cluster. Even if this represents a more or less theoretical situation, it becomes more likely to occur for categorical features. In conclusion, most performance measures identified the correct number of cluster, the *Ptbiserial index* and the *Tau index* even always.

Table 3 (bottom) also shows the results of the simulation study of datasets with 4 clusters. In this simulation the *Ptbiserial index* and the *Tau index* performed worse compared to the results for clustered datasets with 2 clusters: Both indices almost never determine the correct number of clusters. The same result holds for the *Cindex* and the *Dunn index*.

Finally, in Figure 1 there is an overview of the index performance of the simulation study, splitted by the results for datasets with 2 and 4 clusters and the third graph shows the results of the whole simulation study. Overall, the *Silhouette index* as well as the two concordant and discordant comparisons based indices, namely the *Gamma index* and the *Gplus index*, performed best for the analysed data situations.

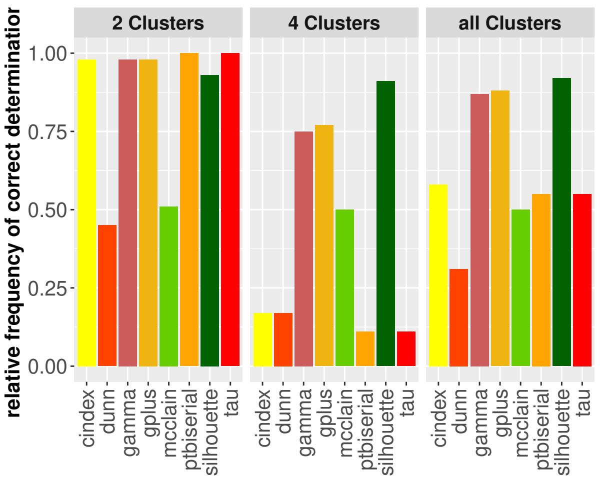

Figure 1: An overview of the relative frequency of determining the correct number of clusters, splitted in 2 or 4 cluster cases and all datasets.

Runtime Analysis of the Considered Indices

In addition, the runtimes of the different validation indices were compared for different data situations. The results are given in Figure 2. As expected, the runtime increases in the number of variables and objects in the datasets. The *McClain index* and the *Silhouette index* are by far faster than the others. Their runtime differs in orders of magnitude. Calculation of the *Tau index* takes longest.

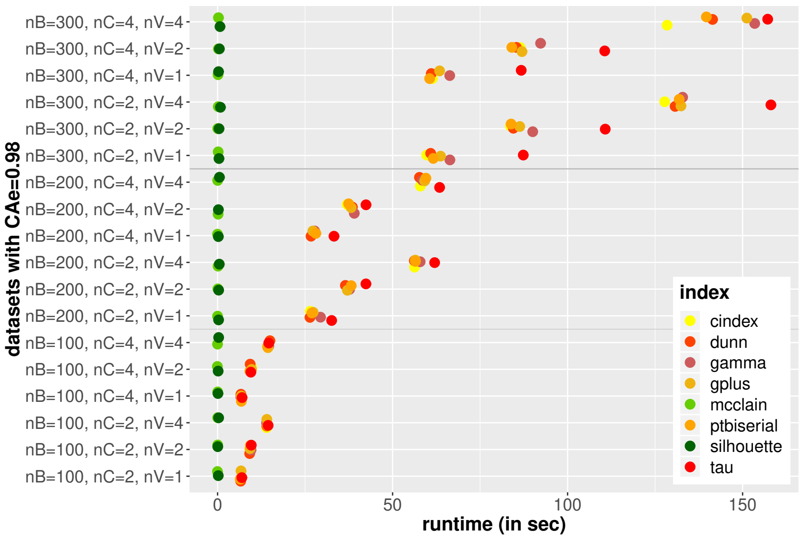

Figure 2: Results of the runtime analysis of the different validation indices.

5 Conclusion

As a summary, we can see that cluster validation indices based on distances can be transferred to mixed-type data. The *Dindex* and the *Hubertus statistic* could also be transferred, which has not been done so far. The eight indices presented and transformed in this paper are already implemented in the R package `clustMixType`.

In a simulation study it turned out that the *Dunn index* performed worse compared to the other indices. The *Tau index* is the validation index with the longest computation time. The shortest runtime has been observed for the

McClain index, but the suggested number of clusters was often misleading. In our study the *Average Silhouette Width* is most suitable with respect to both, runtime and determination of the correct number of clusters.

Future work may consider a comparison of the presented indices to cluster validation using the concept of cluster stability for mixed-type data (Hennig, 2007).

References

Ahmad A, Khan S (2019) Survey of State-of-the-Art Mixed Data Clustering Algorithms. IEEE Access 7:31883–31902, Han G, Yu FR, Chen HC, Shen B, Esposito C, Su X, et al. (eds). DOI: 10.1109/ACCESS.2019.2903568.

Charrad M, Ghazzali N, Boiteau V, Niknafs A (2014) NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set. Journal of Statistical Software 61(6). DOI: 10.18637/jss.v061.i06.

Desgraupes B (2018) clusterCrit: Clustering Indices (Vignette of R package). URL: `https://CRAN.R-project.org/package=clusterCrit`. R package version 1.2.8.

Gower JC (1971) A General Coeffcient of Similarity and Some of Its Properties. Biometrics 27(4):857–871. DOI: 10.2307/2528823.

Halkidi M, Vazirgiannia M, Hennig C (2016) Method-Independent Indices for Cluster Validation and Estimating the number of Clusters. In: Handbook of Cluster Analysis, Hennig C, Meila M, Murtagh F, Rocci R (eds). Hennig, C. and Meila, M. and Murtagh, F. and Rocci, R., Boca Raton (USA), pp. 595–618. DOI: 10.1201/b19706.

Hennig C (2007) Cluster-wise Assessment of Cluster Stability. Computational Statistics and Data Analysis 25(1):258–271. DOI: 10.1023/A:1009769707641.

Huang Z (1997) Clustering Large Data Sets With Mixed Numeric and Categorical Values. Proceedings of the First Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 21–34.

Huang Z (1998) Extension to the k-Means algorithmus for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery 2(6):283–304. DOI: 10.1023/A:1009769707641.

Szepannek G (2018) clustMixType: User-Friendly Clustering of Mixed-Type Data in R. The R Journal 10(2):200–208. DOI: 10.32614/RJ-2018-048.

Szepannek G, Aschenbruck R (2019) clustMixType: k-Prototypes Clustering for Mixed Variable-Type Data. URL: `https://CRAN.R-project.org/package=clusterMixType`. R package version 0.2-1.