Package: clustMixType (via r-universe)

October 29, 2024

Version 0.4-2

Date 2024-06-27

Title k-Prototypes Clustering for Mixed Variable-Type Data

Author Gero Szepannek [aut, cre], Rabea Aschenbruck [aut]

Maintainer Gero Szepannek <gero.szepannek@web.de>

Imports RColorBrewer, tibble, combinat, dplyr, rlang

Suggests testthat

Description Functions to perform k-prototypes partitioning clustering for mixed variable-type data according to Z.Huang (1998): Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Variables, Data Mining and Knowledge Discovery 2, 283-304.

License GPL (>=2)

RoxygenNote 7.2.3

NeedsCompilation no

Encoding UTF-8

Repository https://g-rho.r-universe.dev

RemoteUrl https://github.com/g-rho/clustmixtype

RemoteRef HEAD

RemoteSha c12ce7dce59f3a217cf5aae36d957690e4f5cf30

Contents

---

clprofiles *Profiling k-Prototypes Clustering*

Description

Visualization of a k-prototypes clustering result for cluster interpretation.

Usage

```
clprofiles(object, x, vars = NULL, col = NULL)
```

Arguments

object	Object resulting from a call of resulting kproto. Also other kmeans like objects with object$cluster and object$size are possible.
x	Original data.
vars	Optional vector of either column indices or variable names.
col	Palette of cluster colours to be used for the plots. As a default RColorBrewer's brewer.pal(max(unique(object$cluster)), "Set3") is used for k > 2 clusters and lightblue and orange else.

Details

For numerical variables boxplots and for factor variables barplots of each cluster are generated.

Author(s)

<gero.szepannek@web.de>

Examples

```
# generate toy data with factors and numerics

n   <- 100
prb <- 0.9
muk <- 1.5
clusid <- rep(1:4, each = n)

x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x1 <- as.factor(x1)

x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x2 <- as.factor(x2)

x3 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
```

```
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))

x <- data.frame(x1,x2,x3,x4)

# apply k-prototyps
kpres <- kproto(x, 4)
clprofiles(kpres, x)

# in real world clusters are often not as clear cut
# by variation of lambda the emphasize is shifted towards factor / numeric variables
kpres <- kproto(x, 2)
clprofiles(kpres, x)

kpres <- kproto(x, 2, lambda = 0.1)
clprofiles(kpres, x)

kpres <- kproto(x, 2, lambda = 25)
clprofiles(kpres, x)
```

---

kproto *k-Prototypes Clustering*

Description

Computes k-prototypes clustering for mixed-type data.

Usage

```
kproto(x, ...)

## Default S3 method:
kproto(
  x,
  k,
  lambda = NULL,
  type = "huang",
  iter.max = 100,
  nstart = 1,
  na.rm = "yes",
  keep.data = TRUE,
  verbose = TRUE,
  init = NULL,
  p_nstart.m = 0.9,
  ...
)
```

Arguments

x	Data frame with both numerics and factors (also ordered factors are possible).
...	Currently not used.
k	Either the number of clusters, a vector specifying indices of initial prototypes, or a data frame of prototypes of the same columns as x.
lambda	Parameter > 0 to trade off between Euclidean distance of numeric variables and simple matching coefficient between categorical variables (if type = "huang"). Also a vector of variable specific factors is possible where the order must correspond to the order of the variables in the data. In this case all variables' distances will be multiplied by their corresponding lambda value.
type	Character, to specify the distance for clustering. Either "huang" or "gower" (cf. details below).
iter.max	Numeric; maximum number of iterations if no convergence before.
nstart	Numeric; If > 1 repetitive computations with random initializations are computed and the result with minimum tot.dist is returned.
na.rm	Character, either "yes" to strip NA values for complete case analysis, "no" to keep and ignore NA values, "imp.internal" to impute the NAs within the algorithm or "imp.onestep" to apply the algorithm ignoring the NAs and impute them after the partition is determined.
keep.data	Logical, whether original should be included in the returned object.
verbose	Logical, whether additional information about process should be printed. Caution: For verbose=FALSE, if the number of clusters is reduced during the iterations it will not mentioned.
init	Character, to specify the initialization strategy. Either "nbh.dens", "sel.cen" or "nstart.m". Default is "NULL", which results in nstart repetitive algorithm computations with random starting prototypes. Otherwise, nstart is not used. Argument k must be a number if a specific initialization strategy is choosen!
p_nstart.m	Numeric, probability(=0.9 is default) for init="nstart.m", where the strategy assures that with a probability of p_nstart.m at least one of the m sets of initial prototypes contains objects of every cluster group (cf. Aschenbruck et al. (2023): Random-based Initialization for clustering mixed-type data with the k-Prototypes algorithm. In: *Cladag 2023 Book of abstracts and short spapers*, isbn: 9788891935632.).

Details

Like k-means, the k-prototypes algorithm iteratively recomputes cluster prototypes and reassigns clusters, whereby with type = "huang" clusters are assigned using the distance $d(x, y) = d_{euclid}(x, y) + \lambda d_{simple\,matching}(x, y)$. Cluster prototypes are computed as cluster means for numeric variables and modes for factors (cf. Huang, 1998). Ordered factors variables are treated as categorical variables.

For type = "gower" range-normalized absolute distances from the cluster median are computed for the numeric variables (and for the ranks of the ordered factors respectively). For factors simple matching distance is used as in the original k prototypes algorithm. The prototypes are given by the median for numeric variables, the mode for factors and the level with the closest rank to the median

rank of the corresponding cluster (cf. Szepannek et al., 2024).
In case of `na.rm = FALSE`: for each observation variables with missings are ignored (i.e. only the remaining variables are considered for distance computation). In consequence for observations with missings this might result in a change of variable's weighting compared to the one specified by `lambda`. For these observations distances to the prototypes will typically be smaller as they are based on fewer variables.
The `type` argument also accepts input `"standard"`, but this naming convention is deprecated and has been renamed to `"huang"`. Please use `"huang"` instead.

Value

`kmeans` like object of class `kproto`:

`cluster`	Vector of cluster memberships.
`centers`	Data frame of cluster prototypes.
`lambda`	Distance parameter lambda.
`size`	Vector of cluster sizes.
`withinss`	Vector of within cluster distances for each cluster, i.e. summed distances of all observations belonging to a cluster to their respective prototype.
`tot.withinss`	Target function: sum of all observations' distances to their corresponding cluster prototype.
`dists`	Matrix with distances of observations to all cluster prototypes.
`iter`	Prespecified maximum number of iterations.
`trace`	List with two elements (vectors) tracing the iteration process: `tot.dists` and `moved` number of observations over all iterations.
`inits`	Initial prototypes determined by specified initialization strategy, if init is either 'nbh.dens' or 'sel.cen'.
`nstart.m`	only for 'init = nstart_m': determined number of randomly choosen sets.
`data`	if 'keep.data = TRUE' than the original data will be added to the output list.
`type`	Type argument of the function call.
`stdization`	Only returned for `type = "gower"`: List of standardized ranks for ordinal variables and an additional element `num_ranges` with ranges of all numeric variables. Used by `predict.kproto`.

Author(s)

<gero.szepannek@web.de>

References

- Szepannek, G. (2018): clustMixType: User-Friendly Clustering of Mixed-Type Data in R, *The R Journal 10/2*, 200-208, doi:10.32614/RJ2018048.
- Aschenbruck, R., Szepannek, G., Wilhelm, A. (2022): Imputation Strategies for Clustering Mixed-Type Data with Missing Values, *Journal of Classification*, doi:10.1007/s00357022-09422y.

- Szepannek, G., Aschenbruck, R., Wilhelm, A. (2024): Clustering Large Mixed-Type Data with Ordinal Variables, *Advances in Data Analysis and Classification*, doi:10.1007/s11634-024005955.
- Z.Huang (1998): Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Variables, *Data Mining and Knowledge Discovery 2*, 283-304.

Examples

```
# generate toy data with factors and numerics

n   <- 100
prb <- 0.9
muk <- 1.5
clusid <- rep(1:4, each = n)

x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x1 <- as.factor(x1)

x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x2 <- as.factor(x2)

x3 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))

x <- data.frame(x1,x2,x3,x4)

# apply k-prototypes
kpres <- kproto(x, 4)
clprofiles(kpres, x)

# in real world clusters are often not as clear cut
# by variation of lambda the emphasize is shifted towards factor / numeric variables
kpres <- kproto(x, 2)
clprofiles(kpres, x)

kpres <- kproto(x, 2, lambda = 0.1)
clprofiles(kpres, x)

kpres <- kproto(x, 2, lambda = 25)
clprofiles(kpres, x)
```

`lambdaest` *Compares Variability of Variables*

Description

Investigation of the variables' variances/concentrations to support specification of lambda for k-prototypes clustering.

Usage

```
lambdaest(
  x,
  num.method = 1,
  fac.method = 1,
  outtype = "numeric",
  verbose = TRUE
)
```

Arguments

x	Data.frame with both numerics and factors.
num.method	Integer 1 or 2. Specifies the heuristic used for numeric variables.
fac.method	Integer 1 or 2. Specifies the heuristic used for factor variables.
outtype	Specifies the desired output: either 'numeric', 'vector' or 'variation'.
verbose	Logical whether additional information about process should be printed.

Details

Variance (num.method = 1) or standard deviation (num.method = 2) of numeric variables and $1 - \sum_i p_i^2$ (fac.method = 1) or $1 - \max_i p_i$ (fac.method = 2) for factors is computed.

Value

lambda	Ratio of averages over all numeric/factor variables is returned. In case of outtype = "vector" the separate lambda for all variables is returned as the inverse of the single variables' variation as specified by the num.method and fac.method argument. outtype = "variation" directly returns these quantities and is not meant to be passed directly to kproto().

Author(s)

<gero.szepannek@web.de>

Examples

```
# generate toy data with factors and numerics

n   <- 100
prb <- 0.9
muk <- 1.5
clusid <- rep(1:4, each = n)

x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x1 <- as.factor(x1)

x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
```

```
x2 <- as.factor(x2)

x3 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))

x <- data.frame(x1,x2,x3,x4)

lambdaest(x)
res <- kproto(x, 4, lambda = lambdaest(x))
```

`plot.kproto` *Assign k-Prototypes Clusters*

Description

Plot distributions of the clusters across the variables.

Usage

```
## S3 method for class 'kproto'
plot(x, ...)
```

Arguments

`x`	Object resulting from a call of `kproto`.
`...`	Additional arguments to be passet to `clprofiles` such as e.g. `vars`.

Details

Wrapper around `clprofiles`. Only works for `kproto` object created with `keep.data = TRUE`.

Author(s)

<gero.szepannek@web.de>

Examples

```
# generate toy data with factors and numerics

n   <- 100
prb <- 0.9
muk <- 1.5
clusid <- rep(1:4, each = n)

x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x1 <- as.factor(x1)
```

```
x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x2 <- as.factor(x2)

x3 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))

x <- data.frame(x1,x2,x3,x4)

# apply k-prototyps
kpres <- kproto(x, 4)
plot(kpres, vars = c("x1","x3"))
```

`predict.kproto` *Assign k-Prototypes Clusters*

Description

Predicts k-prototypes cluster memberships and distances for new data.

Usage

```
## S3 method for class 'kproto'
predict(object, newdata, ...)
```

Arguments

`object`	Object resulting from a call of `kproto`.
`newdata`	New data frame (of same structure) where cluster memberships are to be predicted.
`...`	Currently not used.

Value

`kmeans` like object of class `kproto`:

`cluster`	Vector of cluster memberships.
`dists`	Matrix with distances of observations to all cluster prototypes.

Author(s)

<gero.szepannek@web.de>

Examples

```
# generate toy data with factors and numerics

n   <- 100
prb <- 0.9
muk <- 1.5
clusid <- rep(1:4, each = n)

x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x1 <- as.factor(x1)

x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x2 <- as.factor(x2)

x3 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))

x <- data.frame(x1,x2,x3,x4)

# apply k-prototyps
kpres <- kproto(x, 4)
predicted.clusters <- predict(kpres, x)
```

`stability_kproto` *Determination the stability of k Prototypes Clustering*

Description

Calculating the stability for a k-Prototypes clustering with k clusters or computing the stability-based optimal number of clusters for k-Prototype clustering. Possible stability indices are: `Jaccard`, `Rand`, `Fowlkes \& Mallows` and `Luxburg`.

Usage

```
stability_kproto(
  object,
  method = c("rand", "jaccard", "luxburg", "fowlkesmallows"),
  B = 100,
  verbose = FALSE,
  ...
)
```

Arguments

object	Object of class kproto resulting from a call with kproto(..., keep.data=TRUE)
method	character specifying the stability, either one or more of luxburg, fowlkesmallows, rand or/and jaccard.
B	numeric, number of bootstrap samples
verbose	Logical whether information about the bootstrap procedure should be given.
...	Further arguments passed to kproto, like: • nstart: If > 1 repetitive computations of kproto with random initial prototypes are computed. • lambda: Factor to trade off between Euclidean distance of numeric variables and simple matching coefficient between categorical variables.

Value

The output contains the stability for a given k-Prototype clustering in a list with two elements:

kp_stab	stability values for the given clustering
kp_bts_stab	stability values for each bootstrap samples

Author(s)

Rabea Aschenbruck

References

- Aschenbruck, R., Szepannek, G., Wilhelm, A.F.X (2023): Stability of mixed-type cluster partitions for determination of the number of clusters. *Submitted*.
- von Luxburg, U. (2010): Clustering stability: an overview. *Foundations and Trends in Machine Learning, Vol 2, Issue 3*. doi:10.1561/2200000008.
- Ben-Hur, A., Elisseeff, A., Guyon, I. (2002): A stability based method for discovering structure in clustered data. *Pacific Symposium on Biocomputing*. doi:10/bhfxmf.

Examples

```
## Not run:
# generate toy data with factors and numerics
n   <- 10
prb <- 0.99
muk <- 2.5

x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x1 <- as.factor(x1)
x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x2 <- as.factor(x2)
x3 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
```

```
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x <- data.frame(x1,x2,x3,x4)

#' # apply k-prototypes
kpres <- kproto(x, 4, keep.data = TRUE)

# calculate cluster stability
stab <- stability_kproto(method = c("luxburg","fowlkesmallows"), object = kpres)


## End(Not run)
```

---

summary.kproto *Summary Method for kproto Cluster Result*

Description

Investigation of variances to specify lambda for k-prototypes clustering.

Usage

```
## S3 method for class 'kproto'
summary(object, data = NULL, pct.dig = 3, ...)
```

Arguments

object	Object of class kproto.
data	Optional data set to be analyzed. If !(is.null(data)) clusters for data are assigned by predict(object, data). If not specified the clusters of the original data ara analyzed which is only possible if kproto has been called using keep.data = TRUE.
pct.dig	Number of digits for rounding percentages of factor variables.
...	Further arguments to be passed to internal call of summary() for numeric variables.

Details

For numeric variables statistics are computed for each clusters using summary(). For categorical variables distribution percentages are computed.

Value

List where each element corresponds to one variable. Each row of any element corresponds to one cluster.

Author(s)

<gero.szepannek@web.de>

Examples

```
# generate toy data with factors and numerics

n   <- 100
prb <- 0.9
muk <- 1.5
clusid <- rep(1:4, each = n)

x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x1 <- as.factor(x1)

x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x2 <- as.factor(x2)

x3 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))

x <- data.frame(x1,x2,x3,x4)

res <- kproto(x, 4)
summary(res)
```

`validation_kproto` *Validating k Prototypes Clustering*

Description

Calculating the preferred validation index for a k-Prototypes clustering with k clusters or computing the optimal number of clusters based on the choosen index for k-Prototype clustering. Possible validation indices are: `cindex`, `dunn`, `gamma`, `gplus`, `mcclain`, `ptbiserial`, `silhouette` and `tau`.

Usage

```
validation_kproto(
  method = "silhouette",
  object = NULL,
  data = NULL,
  type = "huang",
  k = NULL,
  lambda = NULL,
  kp_obj = "optimal",
  verbose = FALSE,
  ...
)
```

Arguments

method	Character specifying the validation index: cindex, dunn, gamma, gplus, mcclain, ptbiserial, silhouette (default) or tau.
object	Object of class kproto resulting from a call with kproto(..., keep.data=TRUE).
data	Original data; only required if object == NULL and neglected if object != NULL.
type	Character, to specify the distance for clustering; either "huang" or "gower".
k	Vector specifying the search range for optimum number of clusters; if NULL the range will set as 2:sqrt(n). Only required if object == NULL and neglected if object != NULL.
lambda	Factor to trade off between Euclidean distance of numeric variables and simple matching coefficient between categorical variables.
kp_obj	character either "optimal" or "all": Output of the index-optimal clustering (kp_obj == "optimal") or all computed cluster partitions (kp_obj == "all"); only required if object != NULL.
verbose	Logical, whether additional information about process should be printed.
...	Further arguments passed to kproto, like: • nstart: If > 1 repetitive computations of kproto with random initializations are computed. • na.rm: Character, either "yes" to strip NA values for complete case analysis, "no" to keep and ignore NA values, "imp.internal" to impute the NAs within the algorithm or "imp.onestep" to apply the algorithm ignoring the NAs and impute them after the partition is determined.

Details

More information about the implemented validation indices:

- cindex

$$Cindex = \frac{S_w - S_{min}}{S_{max} - S_{min}}$$

For S_{min} and S_{max} it is necessary to calculate the distances between all pairs of points in the entire data set ($\frac{n(n-1)}{2}$). S_{min} is the sum of the "total number of pairs of objects belonging to the same cluster" smallest distances and S_{max} is the sum of the "total number of pairs of objects belonging to the same cluster" largest distances. S_w is the sum of the within-cluster distances.
The minimum value of the index is used to indicate the optimal number of clusters.

- dunn

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

The following applies: The dissimilarity between the two clusters C_i and C_j is defined as $d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$ and the diameter of a cluster is defined as $diam(C_k) = \max_{x,y \in C} d(x, y)$.
The maximum value of the index is used to indicate the optimal number of clusters.

- `gamma`

$$Gamma = \frac{s(+) - s(-)}{s(+) + s(-)}$$

Comparisons are made between all within-cluster dissimilarities and all between-cluster dissimilarities. $s(+)$ is the number of concordant comparisons and $s(-)$ is the number of discordant comparisons. A comparison is named concordant (resp. discordant) if a within-cluster dissimilarity is strictly less (resp. strictly greater) than a between-cluster dissimilarity.
The maximum value of the index is used to indicate the optimal number of clusters.

- `gplus`

$$Gplus = \frac{2 \cdot s(-)}{\frac{n(n-1)}{2} \cdot (\frac{n(n-1)}{2} - 1)}$$

Comparisons are made between all within-cluster dissimilarities and all between-cluster dissimilarities. $s(-)$ is the number of discordant comparisons and a comparison is named discordant if a within-cluster dissimilarity is strictly greater than a between-cluster dissimilarity.
The minimum value of the index is used to indicate the optimal number of clusters.

- `mcclain`

$$McClain = \frac{\bar{S}_w}{\bar{S}_b}$$

$\bar{S}_w$ is the sum of within-cluster distances divided by the number of within-cluster distances and $\bar{S}_b$ is the sum of between-cluster distances divided by the number of between-cluster distances.
The minimum value of the index is used to indicate the optimal number of clusters.

- `ptbiserial`

$$Ptbiserial = \frac{(\bar{S}_b - \bar{S}_w) \cdot (\frac{N_w \cdot N_b}{N_t^2})^{0.5}}{s_d}$$

$\bar{S}_w$ is the sum of within-cluster distances divided by the number of within-cluster distances and $\bar{S}_b$ is the sum of between-cluster distances divided by the number of between-cluster distances.
N_t is the total number of pairs of objects in the data, N_w is the total number of pairs of objects belonging to the same cluster and N_b is the total number of pairs of objects belonging to different clusters. s_d is the standard deviation of all distances.
The maximum value of the index is used to indicate the optimal number of clusters.

- `silhouette`

$$Silhouette = \frac{1}{n} \sum_{i=1}^{n} \frac{b(i) - a(i)}{max(a(i), b(i))}$$

$a(i)$ is the average dissimilarity of the i*th* object to all other objects of the same/own cluster. $b(i) = min(d(i, C))$, where $d(i, C)$ is the average dissimilarity of the i*th* object to all the other clusters except the own/same cluster.
The maximum value of the index is used to indicate the optimal number of clusters.

- `tau`

$$Tau = \frac{s(+) - s(-)}{((\frac{N_t(N_t-1)}{2} - t)\frac{N_t(N_t-1)}{2})^{0.5}}$$

Comparisons are made between all within-cluster dissimilarities and all between-cluster dissimilarities. $s(+)$ is the number of concordant comparisons and $s(-)$ is the number of discordant comparisons. A comparison is named concordant (resp. discordant) if a within-cluster dissimilarity is strictly less (resp. strictly greater) than a between-cluster dissimilarity.
N_t is the total number of distances $\frac{n(n-1)}{2}$ and t is the number of comparisons of two pairs of objects where both pairs represent within-cluster comparisons or both pairs are between-cluster comparisons.
The maximum value of the index is used to indicate the optimal number of clusters.

Value

For computing the optimal number of clusters based on the choosen validation index for k-Prototype clustering the output contains:

`k_opt`	optimal number of clusters (sampled in case of ambiguity)
`index_opt`	index value of the index optimal clustering
`indices`	calculated indices for $k = 2, ..., k_{max}$
`kp_obj`	if(kp_obj == "optimal") the kproto object of the index optimal clustering and if(kp_obj == "all") all kproto which were calculated

For computing the index-value for a given k-Prototype clustering the output contains:

`index`	calculated index-value

Author(s)

Rabea Aschenbruck

References

- Aschenbruck, R., Szepannek, G. (2020): Cluster Validation for Mixed-Type Data. *Archives of Data Science, Series A, Vol 6, Issue 1*. doi:10.5445/KSP/1000098011/02.
- 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, Vol 61, Issue 6*. doi:10.18637/jss.v061.i06.

Examples

```
## Not run:
# generate toy data with factors and numerics
n   <- 10
prb <- 0.99
muk <- 2.5

x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x1 <- as.factor(x1)
x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
```

```
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb)))
x2 <- as.factor(x2)
x3 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x <- data.frame(x1,x2,x3,x4)


# calculate optimal number of cluster, index values and clusterpartition with Silhouette-index
val <- validation_kproto(method = "silhouette", data = x, k = 3:5, nstart = 5)


# apply k-prototypes
kpres <- kproto(x, 4, keep.data = TRUE)

# calculate cindex-value for the given clusterpartition
cindex_value <- validation_kproto(method = "cindex", object = kpres)

## End(Not run)
```

Index