Package ‘clustMixType’

April 23, 2020

Version 0.2-5
Date 2020-04-22
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
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, <DOI:10.1023/A:1009769707641>.
License GPL (>= 2)
RoxygenNote 7.1.0
NeedsCompilation no
Encoding UTF-8
Repository CRAN
Date/Publication 2020-04-23 11:20:06 UTC

R topics documented:

clprofiles ................................................................. 2
kproto ................................................................. 3
lambdaest .............................................................. 6
predict.kproto ......................................................... 7
summary.kproto ....................................................... 8
validation_kproto ..................................................... 9

Index 14
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

```r
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))
```
```r
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)
```

---

**kproto**

### k-Prototypes Clustering

**Description**

Computes k-prototypes clustering for mixed-type data.

**Usage**

```r
kproto(x, ...)```

```r
## Default S3 method:
kproto(
  x,
  k,
  lambda = NULL,
  iter.max = 100,
  nstart = 1,
  na.rm = TRUE,
  keep.data = TRUE,
  verbose = TRUE,
  ...
)
```

**Arguments**

- `x` Data frame with both numerics and factors.
- `...` Currently not used.
kproto

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. 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.

\( \text{iter.max} \)
Maximum number of iterations if no convergence before.

\( \text{nstart} \)
If > 1 repetitive computations with random initializations are computed and the result with minimum \( \text{tot.dist} \) is returned.

\( \text{na.rm} \)
A logical value indicating whether NA values should be stripped before the computation proceeds.

\( \text{keep.data} \)
Logical whether original should be included in the returned object.

\( \text{verbose} \)
Logical whether information about the cluster procedure should be given. Caution: If \( \text{verbose} = \text{FALSE} \), the reduction of the number of clusters is not mentioned.

Details

The algorithm like k-means iteratively recomputes cluster prototypes and reassigns clusters. Clusters are assigned using \( d(x, y) = d_{\text{euclid}}(x, y) + \lambda d_{\text{simple matching}}(x, y) \). Cluster prototypes are computed as cluster means for numeric variables and modes for factors (cf. Huang, 1998). In case of \( \text{na.rm} = \text{FALSE} \): 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 \( \lambda \). Further note: For these observations distances to the prototypes will typically be smaller as they are based on fewer variables.

Value

\( \text{kmeans} \) like object of class kproto:

- \( \text{cluster} \) Vector of cluster memberships.
- \( \text{centers} \) Data frame of cluster prototypes.
- \( \lambda \) Distance parameter \( \lambda \).
- \( \text{size} \) Vector of cluster sizes.
- \( \text{withinss} \) Vector of within cluster distances for each cluster, i.e. summed distances of all observations belonging to a cluster to their respective prototype.
- \( \text{tot.withinss} \) Target function: sum of all observations’ distances to their corresponding cluster prototype.
- \( \text{dists} \) Matrix with distances of observations to all cluster prototypes.
- \( \text{iter} \) Prespecified maximum number of iterations.
- \( \text{trace} \) List with two elements (vectors) tracing the iteration process: \( \text{tot.dists} \) and \( \text{moved} \) number of observations over all iterations.
kproto

Author(s)

<gero.szepannek@web.de>

References

- 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

```r
# 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")`

**Arguments**

- `x`: Original data.
- `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'.

**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**

```r
# 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))

---

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

**Description**

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

**Usage**

```r
## 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-prototypes
kpres <- kproto(x, 4)
predicted.clusters <- predict(kpres, x)

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

```r
# generate toy data with factors and numerics
n <- 100
tau <- 0.9
muk <- 1.5
clusid <- rep(1:4, each = n)
x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(tau, 1-tau))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-tau, tau)))
x1 <- as.factor(x1)
x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(tau, 1-tau))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-tau, tau)))
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)
```

Description

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

validation_kproto(
  method = NULL,
  object = NULL,
  data = NULL,
  k = NULL,
  kp_obj = "optimal",
  ...
)

Arguments

method character specifying the validation index: cindex, dunn, gamma, gplus, mcclain, ptbiserial, silhouette 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.

k Vector specifying the search range for optimum number of clusters; if NULL the range will set as 2:floor(sqrt(n)). Only required if object == NULL and neglected if object != NULL.

kp_obj character either "optimal" or "all": Output of the index-optimal clustering (kp_obj == "optimal") or all computed clusterpartitions (kp_obj == "all"); only required if object != NULL.

Further arguments passed to kproto, like:

• nstart: If > 1 repetetive computations of kproto with random initializations are computed.

• lambda: Factor to trade off between Euclidean distance of numeric variables and simple matching coefficient between categorical variables.

• verbose: Logical whether information about the cluster procedure should be given. Caution: If verbose=FALSE, the reduction of the number of clusters is not mentioned.

Details

More information about the implemented validation indices:

• cindex

\[ C_{index} = \frac{S_w - S_{min}}{S_{max} - S_{min}} \]

For \( S_{min} \) and \( S_{max} \) it is nessesary to calculate the distances between all pairs of points in the entire data set \( \binom{n}{2} \). \( S_{min} \) 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.
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.

**dunn**

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

**gamma**

$$\Gamma = s(+) - s(-) \over s(+) + s(-)$$

Comparisons are made between all within-cluster dissimilarities and all between-cluster dissimilarities. $s(\cdot)$ is the number of concordant comparisons and $s(\cdot)$ 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**

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

Comparisons are made between all within-cluster dissimilarities and all between-cluster dissimilarities. $s(\cdot)$ 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**

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

$\bar{S}_w$ 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. $N_i$ 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**

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

$a(i)$ 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. $s(\cdot)$ is the number of concordant comparisons and $s(\cdot)$ 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 chosen 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


Examples

```r
# 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)
Index

∗Topic **classif**
  kproto, 3
∗Topic **cluster**
  kproto, 3
∗Topic **multivariate**
  kproto, 3
clprofiles, 2
kmeans, 4, 7
kproto, 3, 10
lambdaest, 6
predict.kproto, 7
summary.kproto, 8
validation_kproto, 9