Package 'clustMixType'

September 10, 2018

Version 0.1-36

Date 2018-09-05

Title k-Prototypes Clustering for Mixed Variable-Type Data

Author Gero Szepannek

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 6.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2018-09-10 11:10:11 UTC

R topics documented:

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, iter.max = 100,
  nstart = 1, na.rm = TRUE, keep.data = TRUE, verbose = FALSE, ...)
```

Arguments

`x`	Data frame with both numerics and factors.
`...`	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. 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.
`iter.max`	Maximum number of iterations if no convergence before.

`nstart`	If > 1 repetetive computations with random initializations are computed and the result with minimum tot.dist is returned.
`na.rm`	A logical value indicating whether NA values should be stripped before the computation proceeds.
`keep.data`	Logical whether original should be included in the returned object.
`verbose`	Logical whether report of the number of NAs for each variable should be skipped.

Details

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

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.

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

```
# 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 ment 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))
```

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)
```

---

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)
```

Index