Package ‘kamila’

October 13, 2022

Type Package

Version 0.1.2

Date 2020-03-10

Author Alexander Foss [aut, cre],
Marianthi Markatou [aut]

Maintainer Alexander Foss <alexanderhfoss@gmail.com>

Title Methods for Clustering Mixed-Type Data

Description Implements methods for clustering mixed-type data, specifically combinations of continuous and nominal data. Special attention is paid to the often-overlooked problem of equitably balancing the contribution of the continuous and categorical variables. This package implements KAMILA clustering, a novel method for clustering mixed-type data in the spirit of k-means clustering. It does not require dummy coding of variables, and is efficient enough to scale to rather large data sets. Also implemented is Modha-Spangler clustering, which uses a brute-force strategy to maximize the cluster separation simultaneously in the continuous and categorical variables. For more information, see Foss, Markatou, Ray, & Heching (2016) <doi:10.1007/s10994-016-5575-7> and Foss & Markatou (2018) <doi:10.18637/jss.v083.i13>.

Depends R (>= 3.0.0)

License GPL-3 | file LICENSE

URL https://github.com/ahfoss/kamila

BugReports https://github.com/ahfoss/kamila/issues

Imports stats, abind, KernSmooth, gtools, Rcpp, plyr

LinkingTo Rcpp

Suggests testthat, clustMD, ggplot2, Hmisc

RoxygenNote 7.1.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-03-13 07:20:02 UTC
kamila-package

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>kamila-package</td>
<td>2</td>
</tr>
<tr>
<td>classifyKamila</td>
<td>4</td>
</tr>
<tr>
<td>dptmCpp</td>
<td>5</td>
</tr>
<tr>
<td>dummyCodeFactorDf</td>
<td>5</td>
</tr>
<tr>
<td>genMixedData</td>
<td>6</td>
</tr>
<tr>
<td>gmsClust</td>
<td>7</td>
</tr>
<tr>
<td>kamila</td>
<td>9</td>
</tr>
<tr>
<td>wkmeans</td>
<td>12</td>
</tr>
</tbody>
</table>

Index 14

| kamila-package                | Clustering for mixed continuous and categorical data sets |

Description

A collection of methods for clustering mixed type data, including KAMILA (KAy-means for MIxed LArge data) and a flexible implementation of Modha-Spangler clustering.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>kamila</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>0.1.0</td>
</tr>
<tr>
<td>Date:</td>
<td>2015-10-06</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-3</td>
</tr>
</tbody>
</table>

Author(s)

Alex Foss and Marianthi Markatou

Maintainer: Alex Foss <alexanderhfoss@gmail.com>

References


Examples

```r
## Not run:
```
# import and format a mixed-type data set
data(Byar, package='clustMD')
Byar$logSpap <- log(Byar$Serum.prostatic.acid.phosphatase)
conInd <- c(5,6,8:10,16)
conVars <- Byar[,conInd]
conVars <- data.frame(scale(conVars))
catVarsFac <- Byar[,-c(1:2,conInd,11,14,15)]
catVarsFac[] <- lapply(catVarsFac, factor)
catVarsDum <- dummyCodeFactorDf(catVarsFac)

# Modha-Spangler clustering with kmeans default Hartigan-Wong algorithm
gmsResHw <- gmsClust(conVars, catVarsDum, nclust = 3)

# Modha-Spangler clustering with kmeans Forgy-Lloyd algorithm
# NOTE searchDensity should be >= 10 for optimal performance:
# this is just a syntax demo
gmsResLloyd <- gmsClust(conVars, catVarsDum, nclust = 3,
algorithm = "Lloyd", searchDensity = 3)

# KAMILA clustering
kamRes <- kamila(conVars, catVarsFac, numClust=3, numInit=10)

# Plot results
ternarySurvival <- factor(Byar$SurvStat)
levels(ternarySurvival) <- c('Alive','DeadProst','DeadOther')[c(1,2,rep(3,8))]
plottingData <- cbind(
  conVars,
  catVarsFac,
  KamilaCluster = factor(kamRes$finalMemb),
  MSCluster = factor(gmsResHw$results$cluster))
plottingData$Bone.metastases <- ifelse(
  plottingData$Bone.metastases == 1,
  yes='Yes',
  no='No')

# Plot Modha-Spangler/Hartigan-Wong results
msPlot <- ggplot(plottingData,
  aes(
    x=logSpap,
    y=Index.of.tumour.stage.and.histologic.grade,
    color=ternarySurvival,
    shape=MSCluster))
plotOpts <- function(pl) (pl + geom_point() +
  scale_shape_manual(values=c(2,3,7)) + geom_jitter())
plotOpts(msPlot)

# Plot KAMILA results
kamPlot <- ggplot(
  plottingData,
  aes(
    x=logSpap,
    y=Index.of.tumour.stage.and.histologic.grade,
    color=ternarySurvival,
    shape=MSCluster))
plotOpts(kamPlot)
classifyKamila

Classify new data into existing KAMILA clusters

Description

A function that classifies a new data set into existing KAMILA clusters using the output object from the kamila function.

Usage

classifyKamila(obj, newData)

Arguments

obj

An output object from the kamila function.

newData

A list of length 2, with first element a data frame of continuous variables, and second element a data frame of categorical factors.

Details

A function that takes obj, the output from the kamila function, and newData, a list of length 2, where the first element is a data frame of continuous variables, and the second element is a data frame of categorical factors. Both data frames must have the same format as the original data used to construct the kamila clustering.

Value

An integer vector denoting cluster assignments of the new data points.

References


Examples

# Generate toy data set
set.seed(1234)
dat1 <- genMixedData(400, nConVar = 2, nCatVar = 2, nCatLevels = 4,
nConWithErr = 2, nCatWithErr = 2, popProportions = c(.5,.5),
conErrLev = 0.2, catErrLev = 0.2)
# Partition the data into training/test set
trainingIds <- sample(nrow(dat1$conVars), size = 300, replace = FALSE)
catTrain <- data.frame(apply(dat1$catVars[trainingIds,], 2, factor), stringsAsFactors = TRUE)
conTrain <- data.frame(scale(dat1$conVars)[trainingIds,, stringsAsFactors = TRUE)
catTest <- data.frame(apply(dat1$catVars[-trainingIds,,], 2, factor), stringsAsFactors = TRUE)
conTest <- data.frame(scale(dat1$conVars)[-trainingIds,, stringsAsFactors = TRUE)
# Run the kamila clustering procedure on the training set
kamilaObj <- kamila(conTrain, catTrain, numClust = 2, numInit = 10)
table(dat1$trueID[trainingIds], kamilaObj$finalMemb)
# Predict membership in the test data set
kamilaPred <- classifyKamila(kamilaObj, list(conTest, catTest))
table(dat1$trueID[-trainingIds], kamilaPred)

dptmCpp

Calculate distances from a set of points to a set of centroids

Description
A function that calculates a NxM matrix of distances between a NxP set of points and a MxP set of points.

Usage
dptmCpp(pts, myMeans, wgts)

Arguments
pts             A matrix of points
myMeans         A matrix of centroids, must have same ncol as pts
wgts            A Px1 vector of variable weights

Value
A MxP matrix of distances

dummyCodeFactorDf

Dummy coding of a data frame of factor variables

Description
Given a data frame of factor variables, this function returns a numeric matrix of 0–1 dummy-coded variables.

Usage
dummyCodeFactorDf(dat)

Arguments
dat            A data frame of factor variables
GenMixedData

Value
A numeric matrix of 0–1 dummy coded variables

Examples
dd <- data.frame(a=factor(1:8), b=factor(letters[1:8]), stringsAsFactors = TRUE)
dummyCodeFactorDf(dd)

GenMixedData Generate simulated mixed-type data with cluster structure.

Description
This function simulates mixed-type data sets with a latent cluster structure, with continuous and nominal variables.

Usage
genMixedData(
sampSize, nConVar, nCatVar, nCatLevels, nConWithErr, nCatWithErr, popProportions, conErrLev, catErrLev
)

Arguments
sampSize Integer: Size of the simulated data set.
nConVar The number of continuous variables.
nCatVar The number of categorical variables.
nCatLevels Integer: The number of categories per categorical variables. Currently must be a multiple of the number of populations specified in popProportions.
nConWithErr Integer: The number of continuous variables with error.
nCatWithErr Integer: The number of categorical variables with error.
popProportions A vector of scalars that sums to one. The length gives the number of populations (clusters), with values denoting the prior probability of observing a member of the corresponding population. NOTE: currently only two populations are supported.
conErrLev A scalar between 0.01 and 1 denoting the univariate overlap between clusters on the continuous variables specified to have error.
catErrLev Univariate overlap level for the categorical variables with error.
Details

This function simulates mixed-type data sets with a latent cluster structure. Continuous variables follow a normal mixture model, and categorical variables follow a multinomial mixture model. Overlap of the continuous and categorical variables (i.e. how clear the cluster structure is) can be manipulated by the user. Overlap between two clusters is the area of the overlapping region defined by their densities (or, for categorical variables, the summed height of overlapping segments defined by their point masses). The default overlap level is 0.01 (i.e. almost perfect separation). A user-specified number of continuous and categorical variables can be specified to be "error variables" with arbitrary overlap within 0.01 and 1.00 (where 1.00 corresponds to complete overlap). NOTE: Currently, only two populations (clusters) are supported. While exact control of overlap between two clusters is straightforward, controlling the overlap between the K choose 2 pairwise combinations of clusters is a more difficult task.

Value

A list with the following elements:

- **trueID**: Integer vector giving population (cluster) membership of each observation
- **trueMus**: Mean parameters used for population (cluster) centers in the continuous variables
- **conVars**: The continuous variables
- **errVariance**: Variance parameter used for continuous error distribution
- **popProbsNoErr**: Multinomial probability vectors for categorical variables without measurement error
- **popProbsWithErr**: Multinomial probability vectors for categorical variables with measurement error
- **catVars**: The categorical variables

Examples

```r
dat <- genMixedData(100, 2, 2, nCatLevels=4, nConWithErr=1, nCatWithErr=1, popProportions=c(0.3,0.7), conErrLev=0.3, catErrLev=0.2)
with(dat,plot(conVars,col=trueID))
with(dat,table(data.frame(catVars[,1:2],trueID, stringsAsFactors = TRUE)))
```

---

**gmsClust**

A general implementation of Modha-Spangler clustering for mixed-type data.

Description

Modha-Spangler clustering estimates the optimal weighting for continuous vs categorical variables using a brute-force search strategy.
Usage

gmsClust(
  conData,  
catData,  
nclust,  
searchDensity = 10,  
clustFun = wkmeans,  
conDist = squaredEuc,  
catDist = squaredEuc,
...  
)

Arguments

conData A data frame of continuous variables.
catData A data frame of categorical variables; the allowable variable types depend on 
the specific clustering function used.
nclust An integer specifying the number of clusters.
searchDensity An integer determining the number of distinct cluster weightings evaluated in 
the brute-force search.
clustFun The clustering function to be applied.
conDist The continuous distance function used to construct the objective function.
catDist The categorical distance function used to construct the objective function.
... Arguments to be passed to the clustFun.

Details

Modha-Spangler clustering uses a brute-force search strategy to estimate the optimal weighting for 
continuous vs categorical variables. This implementation admits an arbitrary clustering function 
and arbitrary objective functions for continuous and categorical variables.

The input parameter clustFun must be a function accepting inputs (conData, catData, conWeight, 
nclust, ...) and returning a list containing (at least) the elements cluster, conCenters, and catCenters. 
The list element "cluster" contains cluster memberships denoted by the integers 1:nclust. The list 
elements "conCenters" and "catCenters" must be data frames whose rows denote cluster centroids. 
The function clustFun must allow nclust = 1, in which case $centers returns a data frame with a 
single row. Input parameters conDist and catDist are functions that must each take two data frame 
rows as input and return a scalar distance measure.

Value

A list containing the following results objects:

results A results object corresponding to the base clustering algorithm
objFun A numeric vector of length searchDensity containing the values of the objective function for each weight used
kamila

Qcon  A numeric vector of length searchDensity containing the values of the continuous component of the objective function

Qcat  A numeric vector of length searchDensity containing the values of the categorical component of the objective function

bestInd  The index of the most successful run

weights  A numeric vector of length searchDensity containing the continuous weights used

References


Examples

## Not run:
# Generate toy data set with poor quality categorical variables and good
# quality continuous variables.
set.seed(1)
dat <- genMixedData(200, nConVar=2, nCatVar=2, nCatLevels=4, nConWithErr=2,
nCatWithErr=2, popProportions=c(.5,.5), conErrLev=0.3, catErrLev=0.8)
catDf <- dummyCodeFactorDf(data.frame(apply(dat$catVars, 2, factor), stringsAsFactors = TRUE))
conDf <- data.frame(scale(dat$conVars), stringsAsFactors = TRUE)
msRes <- gmsClust(conDf, catDf, nclust=2)
table(msRes$results$cluster, dat$trueID)
## End(Not run)

kamila  KAMILA clustering of mixed-type data.

Description

KAMILA is an iterative clustering method that equitably balances the contribution of continuous and categorical variables.

Usage

kamila(
  conVar,
  catFactor,
  numClust,
  numInit,
conWeights = rep(1, ncol(conVar)),
catWeights = rep(1, ncol(catFactor)),
maxIter = 25,
conInitMethod = "runif",
catBw = 0.025,
verbose = FALSE,
calcNumClust = "none",
numPredStrCvRun = 10,
predStrThresh = 0.8
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conVar</td>
<td>A data frame of continuous variables.</td>
</tr>
<tr>
<td>catFactor</td>
<td>A data frame of factors.</td>
</tr>
<tr>
<td>numClust</td>
<td>The number of clusters returned by the algorithm.</td>
</tr>
<tr>
<td>numInit</td>
<td>The number of initializations used.</td>
</tr>
<tr>
<td>conWeights</td>
<td>A vector of continuous weights for the continuous variables.</td>
</tr>
<tr>
<td>catWeights</td>
<td>A vector of continuous weights for the categorical variables.</td>
</tr>
<tr>
<td>maxIter</td>
<td>The maximum number of iterations in each run.</td>
</tr>
<tr>
<td>conInitMethod</td>
<td>Character: The method used to initialize each run.</td>
</tr>
<tr>
<td>catBw</td>
<td>The bandwidth used for the categorical kernel.</td>
</tr>
<tr>
<td>verbose</td>
<td>Logical: Whether detailed results should be printed and returned.</td>
</tr>
<tr>
<td>calcNumClust</td>
<td>Character: Method for selecting the number of clusters.</td>
</tr>
<tr>
<td>numPredStrCvRun</td>
<td>Numeric: Number of CV runs for prediction strength method. Ignored unless calcNumClust == 'ps'</td>
</tr>
<tr>
<td>predStrThresh</td>
<td>Numeric: Threshold for prediction strength method. Ignored unless calcNumClust == 'ps'</td>
</tr>
</tbody>
</table>

Details

KAMILA (KAY-means for MIXed LAarge data sets) is an iterative clustering method that equitably balances the contribution of the continuous and categorical variables. It uses a kernel density estimation technique to flexibly model spherical clusters in the continuous domain, and uses a multinomial model in the categorical domain.

Weighting scheme: If no weights are desired, set all weights to 1 (the default setting). Let \( a_1, \ldots, a_p \) denote the weights for \( p \) continuous variables. Let \( b_1, \ldots, b_q \) denote the weights for \( q \) categorical variables. Currently, continuous weights are applied during the calculation of Euclidean distance, as: Categorical weights are applied to the log-likelihoods obtained by the level probabilities given cluster membership as: Total log likelihood for the \( k \)th cluster is obtained by weighting the single continuous log-likelihood by the mean of all continuous weights plus \( \log\text{LikCat}_k \): Note that weights between 0 and 1 are admissible; weights equal to zero completely remove a variable's influence on the clustering; weights equal to 1 leave a variable's contribution unchanged. Weights between 0 and 1 may not be comparable across continuous and categorical variables. Estimating
the number of clusters: Default is no estimation method. Setting calcNumClust to ‘ps’ uses the prediction strength method of Tibshirani & Walther (J. of Comp. and Graphical Stats. 14(3), 2005). There is no perfect method for estimating the number of clusters; PS tends to give a smaller number than, say, BIC based methods for large sample sizes. The user must specify the number of cross-validation runs and the threshold for determining the number of clusters. The smaller the threshold, the larger the number of clusters selected.

Value

A list with the following results objects:

- finalMemb: A numeric vector with cluster assignment indicated by integer.
- numIter: The number of iterations of the algorithm.
- finalLogLik: The pseudo log-likelihood of the returned clustering.
- finalObj: An object describing the results of selecting the number of clusters, empty if calcNumClust == ’none’.
- finalCenters: Final cluster centers.
- finalProbs: Final cluster probabilities.
- input: Object with the given input parameter values.
- nClust: An object describing the results of selecting the number of clusters, empty if calcNumClust == ’none’.
- verbose: An optionally returned object with more detailed information.

References


Examples

# Generate toy data set with poor quality categorical variables and good # quality continuous variables.
set.seed(1)
dat <- genMixedData(200, nConVar = 2, nCatVar = 2, nCatLevels = 4,
                   nConWithErr = 2, nCatWithErr = 2, popProportions = c(.5, .5),
                   conErrLev = 0.3, catErrLev = 0.8)
catDf <- data.frame(apply(dat$catVars, 2, factor), stringsAsFactors = TRUE)
conDf <- data.frame(scale(dat$conVars), stringsAsFactors = TRUE)
kamRes <- kamila(conDf, catDf, numClust = 2, numInit = 10)
table(kamRes$finalMemb, dat$trueID)
**wkmeans**  
*Weighted k-means for mixed-type data*

**Description**

Weighted k-means for mixed continuous and categorical variables. A user-specified weight `conWeight` controls the relative contribution of the variable types to the cluster solution.

**Usage**

`wkmeans(conData, catData, conWeight, nclust, ...)`

**Arguments**

- `conData` The continuous variables. Must be coercible to a data frame.
- `catData` The categorical variables, either as factors or dummy-coded variables. Must be coercible to a data frame.
- `conWeight` The continuous weight; must be between 0 and 1. The categorical weight is `1-conWeight`.
- `nclust` The number of clusters.
- `...` Optional arguments passed to `kmeans`.

**Details**

A simple adaptation of `stats::kmeans` to mixed-type data. Continuous variables are multiplied by the input parameter `conWeight`, and categorical variables are multiplied by `1-conWeight`. If factor variables are input to `catData`, they are transformed to 0-1 dummy coded variables with the function `dummyCodeFactorDf`.

**Value**

A `stats::kmeans` results object, with additional slots `conCenters` and `catCenters` giving the actual centers adjusted for the weighting process.

**See Also**

- `dummyCodeFactorDf`
- `kmeans`

**Examples**

```r
# Generate toy data set with poor quality categorical variables and good quality continuous variables.
set.seed(1)
dat <- genMixedData(200, nConVar=2, nCatVar=2, nCatLevels=4, nConWithErr=2, nCatWithErr=2, popProportions=c(.5,.5), conErrLev=0.3, catErrLev=0.8)
catDf <- data.frame(apply(dat$catVars, 2, factor), stringsAsFactors = TRUE)
```
conDf <- data.frame(scale(dat$conVars), stringsAsFactors = TRUE)

# A clustering that emphasizes the continuous variables
r1 <- with(dat, wkmeans(conDf, catDf, 0.9, 2))
table(r1$cluster, dat$trueID)

# A clustering that emphasizes the categorical variables; note argument
# passed to the underlying stats::kmeans function
r2 <- with(dat, wkmeans(conDf, catDf, 0.1, 2, nstart=4))
table(r2$cluster, dat$trueID)
Index

classifyKamila, 4

dptmCpp, 5
dummyCodeFactorDf, 5, 12

genMixedData, 6
gmsClust, 7

kamila, 9
kamila-package, 2
kmeans, 12

wkmeans, 12