

Package ‘kamila’

September 20, 2026

Type Package

Version 0.2.0

Date 2026-09-18

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

Encoding UTF-8

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

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

Imports stats, parallel, KernSmooth, Rcpp

LinkingTo Rcpp

Suggests testthat, withr, xml2, shiny, cluster, clustMixType, clustrd,
VarSelLCM, flexmix, mvtnorm, mclust, mixtools, MASS, pkgload,
covr

Config/roxygen2/version 8.1.0

Config/testthat/edition 3
NeedsCompilation yes
Repository CRAN
Date/Publication 2026-09-20 00:20:02 UTC

Contents

kamila-package	2
classifyKamila	3
dptmCpp	4
dummyCodeFactorDf	5
genMixedData	6
gmsClust	7
kamila	9
wkmeans	12
Index	14

kamila-package	<i>Clustering for mixed continuous and categorical data sets</i>
----------------	--

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

Author(s)

Alex Foss and Marianthi Markatou
Maintainer: Alex Foss <alexanderhfoss@gmail.com>

References

AH Foss, M Markatou, B Ray, and A Heching (in press). A semiparametric method for clustering mixed data. Machine Learning, DOI: 10.1007/s10994-016-5575-7.

DS Modha and S Spangler (2003). Feature weighting in k-means clustering. Machine Learning 52(3), 217-237.

Examples

```
## Not run:  
# Generate a synthetic mixed-type dataset  
set.seed(123)  
dat <- genMixedData(100, 2, 2,  
  nCatLevels = 4, nConWithErr = 1, nCatWithErr = 1,  
  popProportions = c(0.3, 0.7), conErrLev = 0.3, catErrLev = 0.2  
)
```

```

conVars <- data.frame(scale(dat$conVars))
catVarsFac <- data.frame(lapply(as.data.frame(dat$catVars), factor))
catVarsDum <- dummyCodeFactorDf(catVarsFac)

# Modha-Spangler clustering with default Hartigan-Wong algorithm
gmsResHw <- gmsClust(conVars, catVarsDum, nclust = 2)

# Modha-Spangler clustering with Lloyd algorithm
# NOTE searchDensity should be >= 10 for optimal performance:
# this is just a syntax demo
gmsResLloyd <- gmsClust(conVars, catVarsDum, nclust = 2,
  algorithm = "Lloyd", searchDensity = 3)

# KAMILA clustering with fixed number of clusters
kamRes <- kamila(conVars, catVarsFac, numClust = 2, numInit = 10)

# KAMILA clustering with prediction strength estimation for number of clusters
kamPsRes <- kamila(conVars, catVarsFac, numClust = 2:3, numInit = 10,
  calcNumClust = "ps", numPredStrCvRun = 5, predStrThresh = 0.5)
kamPsRes$nClust$bestNClust

# Plot results using base R graphics
plot(conVars[, 1], conVars[, 2], col = kamRes$finalMemb, pch = 19,
  main = "KAMILA Clusters", xlab = "Continuous Variable 1",
  ylab = "Continuous Variable 2")

## End(Not run)

```

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	For mixed-type data, a list of length 2, with first element a data frame of continuous variables, and second element a data frame of categorical factors. For continuous-only or categorical-only models, either a single data frame or a list containing the single data frame.

Details

A function that takes `obj`, the output from the `kamila` function, and `newData`, which contains the new observations to classify. For mixed-type models, `newData` must be a list of length 2 (continuous data frame and categorical factor data frame). For single-modality models (continuous-only or categorical-only), `newData` can be supplied either as a data frame or as a list containing the single data frame.

Value

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

References

Foss A, Markatou M; kamila: Clustering Mixed-Type Data in R and Hadoop. Journal of Statistical Software, 83(13). 2018. doi: 10.18637/jss.v083.i13

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 $N \times M$ matrix of distances between a $N \times P$ set of points and a $M \times P$ 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	<i>Dummy coding of a data frame of factor variables</i>
-------------------	---

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
-----	----------------------------------

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

```
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	<i>A general implementation of Modha-Spangler clustering for mixed-type data.</i>
----------	---

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

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

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:

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

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

Foss A, Markatou M; kamila: Clustering Mixed-Type Data in R and Hadoop. Journal of Statistical Software, 83(13). 2018. doi: 10.18637/jss.v083.i13

Modha DS, Spangler WS; Feature Weighting in k-Means Clustering. Machine Learning, 52(3). 2003. doi: 10.1023/a:1024016609528

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	<i>KAMILA clustering of mixed-type, continuous-only, or categorical-only data.</i>
--------	--

Description

KAMILA is an iterative clustering method that equitably balances the contribution of continuous and categorical variables. It also natively supports continuous-only and categorical-only data.

Usage

```

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

```

Arguments

conVar	An optional data frame of continuous variables. At least one of conVar or catFactor must be specified.
catFactor	An optional data frame of factors. At least one of conVar or catFactor must be specified.
numClust	The number of clusters returned by the algorithm.
numInit	The number of initializations used.
conWeights	A vector of continuous weights for the continuous variables.
catWeights	A vector of weights for the categorical variables.
maxIter	The maximum number of iterations in each run.
conInitMethod	Character: The method used to initialize each run.
catBw	The bandwidth used for the categorical kernel.
verbose	Logical: Whether detailed results should be printed and returned.
calcNumClust	Character: Method for selecting the number of clusters.
numPredStrCvRun	Numeric: Number of CV runs for prediction strength method. Ignored unless calcNumClust == 'ps'
predStrThresh	Numeric: Threshold for prediction strength method. Ignored unless calcNumClust == 'ps'
numCores	Numeric or cluster: Number of CPU cores to use for parallel execution, or a cluster object created by <code>parallel::makeCluster</code> . Defaults to 1 (sequential execution). Ignored unless calcNumClust == 'ps'.

Details

KAMILA (KAY-means for MIXed LARge 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.

In addition to mixed-type data, KAMILA natively supports single-modality data:

- **Continuous-only data:** When `catFactor = NULL`, KAMILA clusters continuous data using semi-parametric radial kernel density estimation (RKDE), selecting the best initialization via continuous pseudo log-likelihood maximization.
- **Categorical-only data:** When `conVar = NULL`, KAMILA clusters categorical data using fast C++ smoothed multinomial estimation, selecting the best initialization via categorical log-likelihood maximization.

Weighting scheme: If no weights are desired, set all weights to 1 (the default setting). Let $a_1, \dots, a_p$ denote the weights for p continuous variables. Let $b_1, \dots, b_q$ 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. Cluster pair co-membership evaluation is accelerated via an Rcpp routine (`calcPsCpp`) that calculates exact agreement proportions via contingency table partitioning in $O(N + K^2)$ time and $O(K^2)$ memory, which is mathematically and numerically identical to the pairwise formulation without creating an $O(N^2)$ distance matrix.

Prediction strength scores range from 0 to 1, reflecting cluster stability and reproducibility across cross-validation folds. Higher scores indicate greater cluster reproducibility. Relative differences in prediction strength scores across candidate values of `numClust` provide insight into cluster structure: a sharp drop in prediction strength when transitioning from k to $k+1$ clusters suggests that $k+1$ creates unstable or arbitrary partitions. Following Tibshirani & Walther (2005), the largest candidate k whose score (mean plus standard error) meets or exceeds `predStrThresh` is selected.

Value

A list with the following results objects:

<code>finalMemb</code>	A numeric vector with cluster assignment indicated by integer.
<code>numIter</code>	
<code>finalLogLik</code>	The pseudo log-likelihood of the returned clustering.
<code>finalObj</code>	

finalCenters	
finalProbs	
input	Object with the given input parameter values.
nClust	An object describing the results of selecting the number of clusters, empty if <code>calcNumClust == 'none'</code> .
verbose	An optionally returned object with more detailed information.

References

Foss A, Markatou M; kamila: Clustering Mixed-Type Data in R and Hadoop. Journal of Statistical Software, 83(13). 2018. doi: 10.18637/jss.v083.i13

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)

# Standard KAMILA clustering with a specified number of clusters
kamRes <- kamila(conDf, catDf, numClust = 2, numInit = 10)
table(kamRes$finalMemb, dat$trueID)

## Not run:
# KAMILA clustering with prediction strength estimation for number of clusters
kamPsRes <- kamila(
  conDf, catDf,
  numClust = 2:4, numInit = 10,
  calcNumClust = "ps", numPredStrCvRun = 10, predStrThresh = 0.8
)
kamPsRes$nClust$bestNClust
table(kamPsRes$finalMemb, dat$trueID)

## End(Not run)
```

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

```
# 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`, [3](#)
`dptmCpp`, [4](#)
`dummyCodeFactorDf`, [5](#), [13](#)
`genMixedData`, [6](#)
`gmsClust`, [7](#)
`kamila`, [9](#)
`kamila-package`, [2](#)
`kmeans`, [13](#)
`wkmeans`, [12](#)