Package 'flexord'

March 27, 2025

Title Flexible Clustering of Ordinal and Mixed-with-Ordinal Data

Version 1.0.0

Description Extends the capabilities for flexible partitioning and model-based clustering available in the packages 'flexclust' and 'flexmix' to handle ordinal and mixed-with-ordinal data types via new distance, centroid and driver functions that make various assumptions regarding ordinality. Using them within the flex-scheme allows for easy comparisons across methods.

Imports flexclust (>= 1.5.0), methods, mvtnorm, flexmix

Suggests rmarkdown, knitr

License GPL-2

Encoding UTF-8

URL https://zettlchen.github.io/flexord/

BugReports https://github.com/dernst/flexord/issues

RoxygenNote 7.3.2

VignetteBuilder rmarkdown, knitr

Depends R (>= 4.3)

NeedsCompilation no

Author Lena Ortega Menjivar [aut, cre] (<https://orcid.org/0000-0001-5785-4021>), Dominik Ernst [aut] (<https://orcid.org/0009-0005-5579-2636>), Theresa Scharl [ctb] (<https://orcid.org/0000-0001-8850-3312>), Bettina Gruen [ctb] (<https://orcid.org/0000-0001-7265-4773>), Ivan Kondofersky [ctb]

Maintainer Lena Ortega Menjivar <lena.ortega-menjivar@boku.ac.at>

Repository CRAN

Date/Publication 2025-03-27 17:50:02 UTC

Contents

`centMode` *Centroid Functions for K-Centroids Clustering of (Ordinal) Categorical/Mixed Data*

Description

Functions to calculate cluster centroids for K-centroids clustering that extend the options available in package **flexclust**.

`centMode` calculates centroids based on the mode of each variable. `centMin` determines centroids within a specified range which minimize the supplied distance metric. `centOptimNA` replicates the behaviour of `flexclust::centOptim()` but removes missing values.

These functions are designed for use with `flexclust::kcca()` or functions that are built upon it. Their use is easiest via the wrapper `kccaExtendedFamily()`.

Usage

```
centMode(x)

centMin(x, dist, xrange = NULL)

centOptimNA(x, dist)
```

Arguments

`x`	A numeric matrix or data frame.
`dist`	The distance measure function used in `centMin` and `centOptimNA`.
`xrange`	The range of the data in `x`. Currently only used for `centMin`. Options are: • `NULL` (default): defaults to `"all"`. • `"all"`: uses the same minimum and maximum value for each column of `x` by determining the whole range of values in the data object `x`.

- `"columnwise"`: uses different minimum and maximum values for each column of `x` by determining the columnwise ranges of values in the data object `x`.
- A vector of `c(min, max)`: specifies the same minimum and maximum value for each column of `x`.
- A list of vectors `list(c(min1, max1), c(min2, max2),...)` with length `ncol(x)`: specifies different minimum and maximum values for each column of `x`.

Details

- `centMode`: Column-wise modes are used as centroids, and ties are broken randomly. In combination with Simple Matching Distance (`distSimMatch`), this results in the `kmodes` algorithm.
- `centMin`: Column-wise centroids are calculated by minimizing the specified distance measure between the values in `x` and all possible levels of `x`.
- `centOptimNA`: Column-wise centroids are calculated by minimizing the specified distance measure via a general purpose optimizer. Unlike in `flexclust::centOptim()`, NAs are removed from the starting search values and disregarded in the distance calculation.

Value

A named numeric vector containing the centroid values for each column of `x`.

See Also

`kccaExtendedFamily()`, `flexclust::kcca()`

Examples

```
# Example: Mode as centroid
dat <- data.frame(A = rep(2:5, 2),
                  B = rep(1:4, 2),
                  C = rep(c(1, 2, 4, 5), 2))
centMode(dat)
## within kcca
flexclust::kcca(dat, 3, family=kccaExtendedFamily('kModes')) #default centroid

# Example: Centroid is level for which distance is minimal
centMin(dat, flexclust::distManhattan, xrange = 'all')
## within kcca
flexclust::kcca(dat, 3,
                family=flexclust::kccaFamily(dist=flexclust::distManhattan,
                                             cent=\(y) centMin(y, flexclust::distManhattan,
                                                               xrange='all')))

# Example: Centroid calculated by general purpose optimizer with NA removal
nas <- sample(c(TRUE, FALSE), prod(dim(dat)),
              replace=TRUE, prob=c(0.1,0.9)) |>
       matrix(nrow=nrow(dat))
dat[nas] <- NA
```

```
centOptimNA(dat, flexclust::distManhattan)
## within kcca
flexclust::kcca(dat, 3, family=kccaExtendedFamily('kGower')) #default centroid
```

`distGDM2` *Distance Functions for K-Centroids Clustering of (Ordinal) Categorical/Mixed Data*

Description

Functions to calculate the distance between a matrix `x` and a matrix `c`, which can be used for K-centroids clustering via `flexclust::kcca()`.

`distSimMatch` implements Simple Matching Distance (most frequently used for categorical, or symmetric binary data) for K-centroids clustering.

`distGower` implements Gower's Distance after Gower (1971) and Kaufman & Rousseeuw (1990) for mixed-type data with missings for K-centroids clustering.

`distGDM2` implements GDM2 distance for ordinal data introduced by Walesiak et al. (1993) and adapted to K-centroids clustering by Ernst et al. (2025).

These functions are designed for use with `flexclust::kcca()` or functions that are built upon it. Their use is easiest via the wrapper `kccaExtendedFamily()`. However, they can also easily be used to obtain a distance matrix of `x`, see Examples.

Usage

```
distGDM2(x, centers, genDist, xrange = NULL)

distGower(x, centers, genDist)

distSimMatch(x, centers)
```

Arguments

`x`	A numeric matrix or data frame.
`centers`	A numeric matrix with `ncol(centers)` equal to `ncol(x)` and `nrow(centers)` smaller or equal to `row(x)`.
`genDist`	Additional information on `x` required for distance calculation. Filled automatically if used within `flexclust::kcca()`.

- For `distGower`: A character vector of variable specific distances to be used with length equal to `ncol(x)`. The following options are possible:
  - `distEuclidean`: Euclidean distance between the scaled variables.
  - `distManhattan`: absolute distance between the scaled variables.
  - `distJaccard`: counts of zero if both binary variables are equal to 1, and 1 otherwise.
  - `distSimMatch`: Simple Matching Distance, i.e. the number of agreements between variables.

- For `distGDM2`: Function creating a distance function that will be primed on `x`.
- For `distSimMatch`: not used.

`xrange` Range specification for the variables. Currently only used for `distGDM2` (as `distGower` expects `x` to be already scaled). Possible values are:

- `NULL` (default): defaults to `"all"`.
- `"all"`: uses the same minimum and maximum value for each column of `x` by determining the whole range of values in the data object `x`.
- `"columnwise"`: uses different minimum and maximum values for each column of `x` by determining the columnwise ranges of values in the data object `x`.
- A vector of `c(min, max)`: specifies the same minimum and maximum value for each column of `x`.
- A list of vectors `list(c(min1, max1), c(min2, max2),...)` with length `ncol(x)`: specifies different minimum and maximum values for each column of `x`.

Details

- `distSimMatch`: Simple Matching Distance between two observations is calculated as the proportion of disagreements acros all variables. Described, e.g., in Kaufman & Rousseeuw (1990), p. 24. If this is used in K-centroids analysis in combination with mode centroids (as implemented in `centMode`), this results in the **kModes** algorithm. A wrapper for this algorithm is obtained with `kccaExtendedFamily(which='kModes')`.
- `distGower`: Distances are calculated for each column (Euclidean distance, `distEuclidean`, is recommended for numeric, Manhattan distance, `distManhattan` for ordinal, Simple Matching Distance, `distSimMatch` for categorical, and Jaccard distance, `distJaccard` for asymmetric binary variables), and they are summed up as:

$$d(x_i, x_k) = \frac{\sum_{j=1}^{p} \delta_{ikj} d(x_{ij}, x_{kj})}{\sum_{j=1}^{p} \delta_{ikj}}$$

  where p is the number of variables and with the weight δ_{ikj} being 1 if both values x_{ij} and x_{kj} are not missing, and in the case of asymmetric binary variables, at least one of them is not 0. Please note that for calculating Gower's distance, scaling of numeric/ordered variables is required (as f.i. by `.ScaleVarSpecific`). A wrapper for K-centroids analysis using Gower's distance in combination with a numerically optimized centroid is found in `kccaExtendedFamily(which='kGower')`.
- `distGDM2`: GDM2 distance for ordinal variables conducts only relational operations on the variables, such as $\leq$, $\geq$ and $=$. By translating x to its relative frequencies and empirical cumulative distributions, we are able to extend this principle to compare two arbitrary values, and thus use it within K-centroids clustering. For more details, see Ernst et al. (2025). A wrapper for this algorithm in combination with a numerically optimized centroid is found in `kccaExtendedFamily(which='kGDM2')`.

The distances functions presented here can also be used in clustering algorithms that rely on distance matrices (such as hierarchical clustering and PAM), if applied accordingly, see Examples.

Value

A matrix of dimensions `c(nrow(x), nrow(centers))` that contains the distance between each row of `x` and each row of `centers`.

References

- Ernst, D, Ortega Menjivar, L, Scharl, T, Grün, B (2025). *Ordinal Clustering with the flexScheme.* Austrian Journal of Statistics. *Submitted manuscript*.
- Gower, JC (1971). *A General Coefficient for Similarity and Some of Its Properties.* Biometrics, 27(4), 857-871. doi:10.2307/2528823
- Kaufman, L, Rousseeuw, P (1990). *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley Series in Probability and Statistics. doi:10.1002/9780470316801
- Leisch, F (2006). *A Toolbox for K-Centroids Cluster Analysis.* Computational Statistics and Data Analysis, 17(3), 526-544. doi:10.1016/j.csda.2005.10.006
- Kaufman, L, Rousseeuw, P (1990.) *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley Series in Probability and Statistics, New York: John Wiley & Sons. doi:10.1002/9780470316801
- Walesiak, M (1993). *Statystyczna Analiza Wielowymiarowa w Badaniach Marketingowych.* Wydawnictwo Akademii Ekonomicznej, 44-46.
- Weihs, C, Ligges, U, Luebke, K, Raabe, N (2005). *klaR Analyzing German Business Cycles*. In Baier D, Decker, R, Schmidt-Thieme, L (eds.). Data Analysis and Decision Support, 335-343. Berlin: Springer-Verlag. doi:10.1007/3540283978_36

See Also

`flexclust::kcca()`, `klaR::kmodes()`, `cluster::daisy()`, `clusterSim::dist.GDM()`

Examples

```
# Example 1: Simple Matching Distance
set.seed(123)
dat <- data.frame(question1 = factor(sample(LETTERS[1:4], 10, replace=TRUE)),
                  question2 = factor(sample(LETTERS[1:6], 10, replace=TRUE)),
                  question3 = factor(sample(LETTERS[1:4], 10, replace=TRUE)),
                  question4 = factor(sample(LETTERS[1:5], 10, replace=TRUE)),
                  state = factor(sample(state.name[1:10], 10, replace=TRUE)),
                  gender = factor(sample(c('M', 'F', 'N'), 10, replace=TRUE,
                                         prob=c(0.45, 0.45, 0.1))))
datmat <- data.matrix(dat)
initcenters <- datmat[sample(1:10, 3),]
distSimMatch(datmat, initcenters)
## within kcca
flexclust::kcca(dat, k=3, family=kccaExtendedFamily('kModes'))
## as a distance matrix
as.dist(distSimMatch(datmat, datmat))

# Example 2: GDM2 distance
flexclust::kcca(dat, k=3, family=kccaExtendedFamily('kGDM2'))
```

```
# Example 3: Gower's distance
# Ex. 3.1: single variable type case with no missings:
flexclust::kcca(datmat, 3, kccaExtendedFamily('kGower'))

# Ex. 3.2: single variable type case with missing values:
nas <- sample(c(TRUE, FALSE), prod(dim(dat)), replace = TRUE,
   prob=c(0.1, 0.9)) |>
   matrix(nrow = nrow(dat))
dat[nas] <- NA
flexclust::kcca(dat, 3, kccaExtendedFamily('kGower', cent=centMode))

#Ex. 3.3: mixed variable types (with or without missings):
dat <- data.frame(cont = sample(1:100, 10, replace=TRUE)/10,
                  bin_sym = as.logical(sample(0:1, 10, replace=TRUE)),
                  bin_asym = as.logical(sample(0:1, 10, replace=TRUE)),
                  ord_levmis = factor(sample(1:5, 10, replace=TRUE),
                                      levels=1:6, ordered=TRUE),
                  ord_levfull = factor(sample(1:4, 10, replace=TRUE),
                                       levels=1:4, ordered=TRUE),
                  nom = factor(sample(letters[1:4], 10, replace=TRUE),
                               levels=letters[1:4]))
dat[nas] <- NA
flexclust::kcca(dat, 3, kccaExtendedFamily('kGower'))
```

`FLXMCregbetabinom` *FlexMix Driver for Regularized Beta-Binomial Mixtures*

Description

This model driver can be used to cluster data using the beta-binomial distribution.

Usage

```
FLXMCregbetabinom(
  formula = . ~ .,
  size = NULL,
  alpha = 0,
  eps = sqrt(.Machine$double.eps)
)
```

Arguments

`formula`	A formula which is interpreted relative to the formula specified in the call to `flexmix::flexmix()` using `stats::update.formula()`. Only the left-hand side (response) of the formula is used. Default is to use the original model formula specified in `flexmix::flexmix()`.

size	Number of trials (one or more). Default NULL implies that the number of trials is inferred columnwise by the maximum value observed.
alpha	A non-negative scalar acting as regularization parameter. Can be regarded as adding alpha observations equal to the population mean to each component.
eps	Lower threshold for the shape parameters a and b.

Details

Using a regularization parameter alpha greater than zero can be viewed as adding alpha observations equal to the population mean to each component. This can be used to avoid degenerate solutions (i.e., probabilites of 0 or 1). It also has the effect that clusters become more similar to each other the larger alpha is chosen. For small values this effect is, however, mostly negligible.

Value

An object of class "FLXC".

References

Ernst, D, Ortega Menjivar, L, Scharl, T, Grün, B (2025). *Ordinal Clustering with the flex-Scheme.* Austrian Journal of Statistics. *Submitted manuscript*.

Kondofersky, I (2008). *Modellbasiertes Clustern mit der Beta-Binomialverteilung.* Bachelor's thesis, Ludwig-Maximilians-Universität München.

Examples

```
library("flexmix")
library("flexord")
library("flexclust")

# Sample data
k <- 4     # nr of clusters
size <- 4  # nr of trials
N <- 100   # obs. per cluster

set.seed(0xdeaf)

# random probabilities per component
probs <- lapply(seq_len(k), \(ki) runif(10, 0.01, 0.99))

# sample data
dat <- lapply(probs, \(p) {
    lapply(p, \(p_i) {
        rbinom(N, size, p_i)
    }) |> do.call(cbind, args=_)
}) |> do.call(rbind, args=_)

true_clusters <- rep(1:4, rep(N, k))

# Sample data is drawn from a binomial distribution but we fit
```

```
# beta-binomial which is a slight mis-specification but the
# beta-binomial can be seen as a generalized binomial.
m <- flexmix(dat~1, model=FLXMCregbetabinom(size=size, alpha=0),
             cluster = true_clusters)

# Cluster without regularization
m1 <- stepFlexmix(dat~1, model=FLXMCregbetabinom(size=size, alpha=0), k=k)

# Cluster with regularization
m2 <- flexmix(dat~1, model=FLXMCregbetabinom(size=size, alpha=1), k=k,
              cluster = posterior(m1))

# Both models are mostly able to reconstruct the true clusters (ARI ~ 0.95)
# (it's a very easy clustering problem)
# Small values for the regularization don't seem to affect the ARI (much)
randIndex(clusters(m1), true_clusters)
randIndex(clusters(m2), true_clusters)
```

FLXMCregbinom *FlexMix Driver for Regularized Binomial Mixtures*

Description

This model driver can be used to cluster data using the binomial distribution.

Usage

```
FLXMCregbinom(formula = . ~ ., size = NULL, hasNA = FALSE, alpha = 0, eps = 0)
```

Arguments

formula	A formula which is interpreted relative to the formula specified in the call to flexmix::flexmix() using stats::update.formula(). Only the left-hand side (response) of the formula is used. Default is to use the original model formula specified in flexmix::flexmix().
size	Number of trials (one or more). Default NULL implies that the number of trials is inferred columnwise by the maximum value observed.
hasNA	Boolean whether the data set may contain NA values. Default is FALSE. For data sets without NAs, the same results are obtained but it runs slightly faster when the absence of NAs can be assumed.
alpha	A non-negative scalar acting as regularization parameter. Can be regarded as adding alpha observations equal to the population mean to each component.
eps	A numeric value in [0, 1). When greater than zero, probabilities are truncated to be within in [eps, 1-eps].

Details

Using a regularization parameter `alpha` greater than zero can be viewed as adding `alpha` observations equal to the population mean to each component. This can be used to avoid degenerate solutions (i.e., probabilites of 0 or 1). It also has the effect that clusters become more similar to each other the larger `alpha` is chosen. For small values this effect is, however, mostly negligible.

Parameter estimation is achieved using the MAP estimator for each component and variable using a Beta prior.

Value

An object of class `"FLXC"`.

References

- Ernst, D, Ortega Menjivar, L, Scharl, T, Grün, B (2025). *Ordinal Clustering with the flexScheme.* Austrian Journal of Statistics. *Submitted manuscript*.

Examples

```
library("flexmix")
library("flexord")
library("flexclust")

# Sample data
k <- 4     # nr of clusters
size <- 4  # nr of trials
N <- 100   # obs. per cluster

set.seed(0xdeaf)

# random probabilities per component
probs <- lapply(seq_len(k), \(ki) runif(10, 0.01, 0.99))

# sample data
dat <- lapply(probs, \(p) {
    lapply(p, \(p_i) {
        rbinom(N, size, p_i)
    }) |> do.call(cbind, args=_)
}) |> do.call(rbind, args=_)

true_clusters <- rep(1:4, rep(N, k))

# Cluster without regularization
m1 <- stepFlexmix(dat~1, model=FLXMCregbinom(size=size, alpha=0), k=k)

# Cluster with regularization
m2 <- stepFlexmix(dat~1, model=FLXMCregbinom(size=size, alpha=1), k=k)

# Both models are mostly able to reconstruct the true clusters (ARI ~ 0.96)
# (it's a very easy clustering problem)
# Small values for the regularization don't seem to affect the ARI (much)
```

```
randIndex(clusters(m1), true_clusters)
randIndex(clusters(m2), true_clusters)
```

FLXMCregmultinom *FlexMix Driver for Regularized Multinomial Mixtures*

Description

This model driver can be used to cluster data using a multinomial distribution.

Usage

```
FLXMCregmultinom(formula = . ~ ., r = NULL, alpha = 0)
```

Arguments

formula	A formula which is interpreted relative to the formula specified in the call to flexmix::flexmix() using stats::update.formula(). Only the left-hand side (response) of the formula is used. Default is to use the original model formula specified in flexmix::flexmix().
r	Number of different categories. Values are assumed to be integers in 1:r. Default NULL implies that the number of different categories is inferred columnwise by the maximum value observed.
alpha	A non-negative scalar acting as regularization parameter. Can be regarded as adding alpha observations equal to the population mean to each component.

Details

Using a regularization parameter alpha greater than zero acts as adding alpha observations conforming to the population mean to each component. This can be used to avoid degenerate solutions. It also has the effect that clusters become more similar to each other the larger alpha is chosen. For small values it is mostly negligible however.

For regularization we compute the MAP estimates for the multinomial distribution using the Dirichlet distribution as prior, which is the conjugate prior. The parameters of this prior are selected to correspond to the marginal distribution of the variable across all observations.

Value

An object of class "FLXC".

References

- Galindo Garre, F, Vermunt, JK (2006). *Avoiding Boundary Estimates in Latent Class Analysis by Bayesian Posterior Mode Estimation* Behaviormetrika, 33, 43-59. - Ernst, D, Ortega Menjivar, L, Scharl, T, Grün, B (2025). *Ordinal Clustering with the flex-Scheme.* Austrian Journal of Statistics. *Submitted manuscript*.

Examples

```
library("flexmix")
library("flexord")
library("flexclust")


set.seed(0xdeaf)

# Sample data
k <- 4     # nr of clusters
nvar <- 10  # nr of variables
r <- sample(2:7, size=nvar, replace=TRUE)  # nr of categories
N <- 100   # obs. per cluster


# random probabilities per component
probs <- lapply(seq_len(k), \(ki) runif(nvar, 0.01, 0.99))

# sample data by drawing from a binomial distribution with size = r - 1
# values are expect values to lie inside 1:r hence we add +1.
dat <- lapply(probs, \(p) {
    mapply(\(p_i, r_i) {
        rbinom(N, r_i, p_i) + 1
    }, p, r-1, SIMPLIFY=FALSE) |> do.call(cbind, args=_)
}) |> do.call(rbind, args=_)

true_clusters <- rep(1:4, rep(N, k))

# Cluster without regularization
m1 <- stepFlexmix(dat~1, model=FLXMCregmultinom(r=r, alpha=0), k=k)

# Cluster with regularization
m2 <- stepFlexmix(dat~1, model=FLXMCregmultinom(r=r, alpha=1), k=k)

# Both models are mostly able to reconstruct the true clusters (ARI ~ 0.95)
# (it's a very easy clustering problem)
# Small values for the regularization don't seem to affect the ARI (much)
randIndex(clusters(m1), true_clusters)
randIndex(clusters(m2), true_clusters)
```

FLXMCregnorm *FlexMix Driver for Regularized Multivariate Normal Mixtures*

Description

This model driver implements the regularization method as introduced by Fraley and Raftery (2007) for univariate normal mixtures. Default parameters for the regularization according to that paper may be obtained using `FLXMCregnorm_defaults()`. We extend this to the multivariate case assuming independence between variables within components, i.e., we only implement the special

case where the covariance matrix is diagonal. For more general applications of normal mixtures see package **mclust**.

Usage

```
FLXMCregnorm(formula = . ~ ., params)
```

Arguments

formula	A formula which is interpreted relative to the formula specified in the call to `flexmix::flexmix()` using `stats::update.formula()`. Only the left-hand side (response) of the formula is used. Default is to use the original model formula specified in `flexmix::flexmix()`.
params	Prior parameters for normal mixtures. You may obtain default values according to Fraley and Raftery (2007) using `FLXMCregnorm_defaults()`. As the prior depends on the number of components it is probably not advisable to run `stepFlexmix` with more than one value of `k` at a time.

Details

For the regularization the conjugate prior distributions for the normal distribution are used, which are:

- Normal prior with parameter `mu_p` and `sigma^2/kappa_p` for the mean.
- Inverse Gamma prior with parameters `nu_p/2` and `zeta_p^2/2` for the variance.

Value

An object of class `"FLXC"`.

References

- Ernst, D, Ortega Menjivar, L, Scharl, T, Grün, B (2025). *Ordinal Clustering with the flexScheme.* Austrian Journal of Statistics. *Submitted manuscript*.
- Fraley, C, Raftery, AE (2007) *Bayesian Regularization for Normal Mixture Estimation and Model-Based Clustering.* Journal of Classification, 24(2), 155-181

See Also

FLXMCregnorm_defaults

Examples

```
library("flexmix")
library("flexord")
library("flexclust")

# example data
data("iris", package = "datasets")
my_iris <- subset(iris, select=setdiff(colnames(iris), "Species")) |>
```

```
    as.matrix()

# cluster one model with a scale parameter similar to the default for 3 components
params <- FLXMCregnorm_defaults(my_iris, zeta_p = c(0.23, 0.06, 1.04, 0.19))
m1 <- stepFlexmix(my_iris ~ 1, k = 3,
    model=FLXMCregnorm(params = params))
summary(m1)

# rand index of clusters vs species
randIndex(clusters(m1), iris$Species)

# cluster one model with default scale parameter
params <- FLXMCregnorm_defaults(my_iris, k = 3)
m2 <- stepFlexmix(my_iris ~ 1, k = 3,
    model = FLXMCregnorm(params = params))
summary(m2)

# rand index of clusters vs species
randIndex(clusters(m2), iris$Species)

# rand index between both models (should be >= 0.8)
randIndex(clusters(m1), clusters(m2))
```

`FLXMCregnorm_defaults` *Data-Driven Default Parameters for Regularized Normal Mixtures*

Description

Determines the default values for regularized univariate normal mixtures as proposed by Fraley and Raftery (2007) based on the data set to be clustered and the number of components in the mixture model.

Usage

```
FLXMCregnorm_defaults(x, zeta_p = NULL, kappa_p = 0.01, nu_p = 3, k = NULL)
```

Arguments

`x`	The data set to be clustered. Should be the same data set as is used in `flexmix::flexmix()`'s model formula.
`zeta_p`	Scale (hyperparameter for IG prior). If not given the empirical variance divided by the square of the number of components is used as per Fraley and Raftery (2007). `mu_p` is computed from the data as the overall means across all components. A value for the scale hyperparameter `zeta_p` may be specified directly. Otherwise the empirical variance divided by the square of the number of components is used as per Fraley and Raftery (2007). In which case the number of components (parameter `k`) needs to be specified.

kappa_p	Shrinkage parameter. Corresponds to adding kappa_p observations according to the population mean to each component (hyperparameter for IG prior)
nu_p	Degress of freedom (hyperparameter for IG prior)
k	Number of components assumed for the mixture model (not used if zeta_p is given).

Value

A named list with values for mu_p, kappa_p, nu_p and zeta_p.

---

kccaExtendedFamily *Extending K-Centroids Clustering to (Mixed-with-)Ordinal Data*

Description

This wrapper creates objects of class "kccaFamily", which can be used with flexclust::kcca() to conduct K-centroids clustering using the following methods:

- **kModes** (after Weihs et al., 2005)
- **kGower** (Gower's distance after Kaufman & Rousseeuw, 1990, and a user specified centroid)
- **kGDM2** (GDM2 distance after Walesiak et al., 1993, and a user specified centroid)

Usage

```
kccaExtendedFamily(which = c('kModes', 'kGDM2', 'kGower'),
                   cent = NULL,
                   preproc = NULL,
                   xrange = NULL,
                   xmethods = NULL,
                   trim = 0, groupFun = 'minSumClusters')
```

Arguments

which	One of either 'kModes', 'kGDM2' or 'kGower', the three predefined methods for K-centroids clustering. For more information on each of them, see the Details section.
cent	Function for determining cluster centroids. This argument is ignored for `which='kModes'`, and `centMode` is used. For `'kGDM2'` and `'kGower'`, `cent=NULL` defaults to a general purpose optimizer.
preproc	Preprocessing function applied to the data before clustering. This argument is ignored for `which='kGower'`. In this case, the default preprocessing proposed by Gower (1971) and Kaufman & Rousseeuw (1990) is conducted. For `'kGDM2'` and `'kModes'`, users can specify preprocessing steps here, though this is not recommended.

`xrange` The range of the data in `x`. Options are:

- `"all"`: uses the same minimum and maximum value for each column of `x` by determining the whole range of values in the data object `x`.
- `"columnwise"`: uses different minimum and maximum values for each column of `x` by determining the columnwise ranges of values in the data object `x`.
- A vector of `c(min, max)`: specifies the same minimum and maximum value for each column of `x`.
- A list of vectors `list(c(min1, max1), c(min2, max2),...)` with length `ncol(x)`: specifies different minimum and maximum values for each column of `x`.

This argument is ignored for `which='kModes'`. `xrange=NULL` defaults to `"all"` for `'kGDM2'`, and to `"columnwise"` for `'kGower'`.

`xmethods` An optional character vector of length `ncol(x)` that specifies the distance measure for each column of `x`. Currently only used for `'kGower'`. For `'kGower'`, `xmethods=NULL` results in the use of default methods for each column of `x`. For more information on allowed input values, and default measures, see the Details section.

`trim` Proportion of points trimmed in robust clustering, wee `flexclust::kccaFamily()`.

`groupFun` A character string specifying the function for clustering.

Default is `'minSumClusters'`, see [flexclust::kccaFamily()].

Details

Wrappers for defining families are obtained by specifying `which` using:

- `which='kModes'` creates an object for **kModes** clustering, i.e., K-centroids clustering using Simple Matching Distance (counts of disagreements) and modes as centroids. Argument `cent` is ignored for this method.
- `which='kGower'` creates an object for performing clustering using Gower's method as described in Kaufman & Rousseeuw (1990):
  - Numeric and/or ordinal variables are scaled by $\frac{\mathbf{x}-\min \mathbf{x}}{\max \mathbf{x}-\min \mathbf{x}}$. Note that for ordinal variables the internal coding with values from 1 up to their maximum level is used.
  - Distances are calculated for each column (Euclidean distance, `distEuclidean`, is recommended for numeric, Manhattan distance, `distManhattan` for ordinal, Simple Matching Distance, `distSimMatch` for categorical, and Jaccard distance, `distJaccard` for asymmetric binary variables), and they are summed up as:

$$d(x_i, x_k) = \frac{\sum_{j=1}^{p} \delta_{ikj} d(x_{ij}, x_{kj})}{\sum_{j=1}^{p} \delta_{ikj}}$$

where p is the number of variables and with the weight δ_{ikj} being 1 if both values x_{ij} and x_{kj} are not missing, and in the case of asymmetric binary variables, at least one of them is not 0.
The columnwise distances used can be influenced in two ways: By passing a character vector of length p to `xmethods` that specifies the distance for each column. Options are:

distEuclidean, distManhattan, distJaccard, and distSimMatch. Another option is to not specify any methods within kccaExtendedFamily, but rather pass a "data.frame" as argument x in kcca, where the class of the column is used to infer the distance measure. distEuclidean is used on numeric and integer columns, distManhattan on columns that are coded as ordered factors, distSimMatch is the default for categorically coded columns, and distJaccard is the default for binary coded columns.
For this method, if cent=NULL, a general purpose optimizer with NA omission is applied for centroid calculation.

- which='kGDM2' creates an obejct for clustering using the GDM2 distance for ordinal variables. The GMD2 distance was first introduced by Walesiak et al. (1993), and adapted in Ernst et al. (2025), as the distance measure within flexclust::kcca().

  This distance respects the ordinal nature of a variable by conducting only relational operations to compare values, such as $\leq$, $\geq$ and $=$. By obtaining the relative frequencies and empirical cumulative distributions of x, we allow for comparison of two arbitrary values, and thus are able to conduct K-centroids clustering. For more details, see Ernst et al. (2025).

Also for this method, if cent=NULL, a general purpose optimizer with NA omission will be applied for centroid calculation.

Scale handling. In 'kModes', all variables are treated as unordered factors. In 'kGDM2', all variables are treated as ordered factors, with strict assumptions regarding their ordinality. 'kGower' is currently the only method designed to handle mixed-type data. For ordinal variables, the assumptions are more lax than with GDM2 distance.

NA handling. NA handling via omission and upweighting non-missing variables is currently only implemented for 'kGower'. Within 'kModes', the omission of NA responses can be avoided by coding missings as valid factor levels. For 'kGDM2', currently the only option is to omit missing values completely.

Value

An object of class "kccaFamily".

References

- Ernst, D, Ortega Menjivar, L, Scharl, T, Grün, B (2025). *Ordinal Clustering with the flexScheme.* Austrian Journal of Statistics. *Submitted manuscript*.
- Gower, JC (1971). *A General Coefficient for Similarity and Some of Its Properties.* Biometrics, 27(4), 857-871. doi:10.2307/2528823
- Kaufman, L, Rousseeuw, P (1990). *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley Series in Probability and Statistics. doi:10.1002/9780470316801
- Leisch, F (2006). *A Toolbox for K-Centroids Cluster Analysis.* Computational Statistics and Data Analysis, 17(3), 526-544. doi:10.1016/j.csda.2005.10.006
- Walesiak, M (1993). *Statystyczna Analiza Wielowymiarowa w Badaniach Marketingowych.* Wydawnictwo Akademii Ekonomicznej, 44-46.
- Weihs, C, Ligges, U, Luebke, K, Raabe, N (2005). *klaR Analyzing German Business Cycles.* In: Data Analysis and Decision Support, Springer: Berlin. 335-343. doi:10.1007/354028397-8_36

See Also

flexclust::kcca(), flexclust::stepFlexclust(), flexclust::bootFlexclust()

Examples

```
# Example 1: kModes
set.seed(123)
dat <- data.frame(cont = sample(1:100, 10, replace=TRUE)/10,
                  bin_sym = as.logical(sample(0:1, 10, replace=TRUE)),
                  bin_asym = as.logical(sample(0:1, 10, replace=TRUE)),
                  ord_levmis = factor(sample(1:5, 10, replace=TRUE),
                                      levels=1:6, ordered=TRUE),
                  ord_levfull = factor(sample(1:4, 10, replace=TRUE),
                                       levels=1:4, ordered=TRUE),
                  nom = factor(sample(letters[1:4], 10, replace=TRUE),
                               levels=letters[1:4]))
flexclust::kcca(dat, k=3, family=kccaExtendedFamily('kModes'))

# Example 2: kGDM2
flexclust::kcca(dat, k=3, family=kccaExtendedFamily('kGDM2',
                                                    xrange='columnwise'))
# Example 3: kGower
flexclust::kcca(dat, 3, kccaExtendedFamily('kGower'))
nas <- sample(c(TRUE,FALSE), prod(dim(dat)), replace=TRUE, prob=c(0.1,0.9)) |>
   matrix(nrow=nrow(dat))
dat[nas] <- NA
flexclust::kcca(dat, 3, kccaExtendedFamily('kGower',
                                           xrange='all'))
flexclust::kcca(dat, 3, kccaExtendedFamily('kGower',
                                           xmethods=c('distEuclidean',
                                                      'distEuclidean',
                                                      'distJaccard',
                                                      'distManhattan',
                                                      'distManhattan',
                                                      'distSimMatch')))
#the case where column 2 is a binary variable, but is symmetric
```

---

lowbackpain *Low Back Pain Diagnoses and Diagnosis Criteria*

Description

In a 2011 study, Smart et al. (2011) collected information on 464 Irish and British patients suffering from low back pain regarding:

- the type of low back pain (classified into "nociceptive", "peripheral neuropathic", and "central neuropathic")
- the presence/absence of 38 clinical criteria and symptoms relating to low back pain.

Fop et al. (2017) conducted Latent Class Analysis on this data set to retrieve the experts' classifications; and by comparing nested models they were able to select 11 out of 38 criteria which contain the most of the relevant grouping information while avoiding redundancy.

Usage

```
data('lowbackpain')
```

Format

A list containing:

data: A 464x11 binary matrix indicating the presence/absence of the 11 selected criteria for each of the 464 patients.

group: A factor of length 464 indicating the diagnosis each patient received, numerically coded (order has no meaning).

index: The index for the criteria explaining which symptom they refer to.

Source

Supplemental Content for Fop et al. (2017): doi:10.1214/17AOAS1061SUPP

References

- Fop, M, Smart, K, Murphy, TB (2017). *Variable Selection for Latent Class Analysis with Application to Low Back Pain Diagnosis*. The Annals of Applied Statitics. 11(4), 2080-2110. doi:10.1214/17aoas1061
- Smart, K, Blake, C, Staines, A, Doody, C (2011). *The Discriminative Validity of "Nociceptive", "Peripheral Neuropathic", and "Central Sensitization" as Mechanisms-Based Classifications of Musculoskeletal Pain*. The Clinical Journal of Pain. 27, 655-663. doi:10.1097/AJP.0b013e318215f16a

---

`risk` *Risk Aversion*

Description

Survey data from 563 respondents on frequency of risk taking on six different types. Taken from the companion package to *Market Segmentation Analysis: Understanding It, Doing It, and Making It Useful* (Dolnicar et al., 2018, doi:10.1007/9789811088186).

Usage

```
data('risk')
```

Format

A matrix with 563 respondents (rows) and 6 variables (columns) named `Recreational`, `Health`, `Career`, `Financial`, `Safety` and `Social`.

Details

The data was collected by academic researchers using a permission based online panel.

The sample was taken from adult Australian residents who have undertaken at least one holiday in the last year which involved staying away from home for at least four nights.

The respondents were asked: "Which risks have you taken in the past?" and answered on a 5-point scale with options:

- Never (1)
- Rarely (2)
- Quite often (3)
- Often (4)
- Very often (5)

The six types of risk were:

- Recreational: e.g. rock-climbing, scuba diving
- Health: e.g., smoking, poor diet, high alcohol consumption
- Career: e.g., quitting a job without another to go to
- Financial: e.g., gambling, risky investments
- Safety: e.g., speeding
- Social: e.g., standing for election, publicly challenging a rule or decision

Source

Sara Dolnicar.

Data and help page are taken from the companion package to *Market Segmentation Analysis: Understanding It, Doing It, and Making It Useful* (Dolnicar et al., 2018, doi:10.1007/9789811088186).

URL: https://statistik.boku.ac.at/nachlass_leisch/MSA/

References

- Hajibaba H, Dolnicar S (2017). *Helping When Disaster Hits*. In: Dolnicar S (ed) Peer-to-Peer Accomodation Networks: Pushing the Boundaries, Goodfellow Publishers, Oxford, chap.21, 235-243. doi:10.23912/97819113965123619
- Hajibaba H, Karlsson L, Dolnicar S (2017) *Residents Open Their Homes to Tourists When Disaster Strikes.* Journal of Travel Research. 58(8), 1065-1078. doi:10.1177/0047287516677167

Index