Package ‘IBclust’

August 28, 2025

Type Package

Title Information Bottleneck Methods for Clustering Mixed-Type Data

Version 1.2.1

Description Implements multiple variants of the Information Bottleneck ('IB') method for clustering datasets containing continuous, categorical (nominal/ordinal) and mixed-type variables.
The package provides deterministic, agglomerative, generalized, and standard 'IB' clustering algorithms that preserve relevant information while forming interpretable clusters. The Deterministic Information Bottleneck is described in Costa et al. (2024) <doi:10.48550/arXiv.2407.03389>. The standard 'IB' method originates from Tishby et al. (2000) <doi:10.48550/arXiv.physics/0004057>, the agglomerative variant from Slonim and Tishby (1999) <https://papers.nips.cc/paper/1651-agglomerative-information-bottleneck>, and the generalized 'IB' from Strouse and Schwab (2017) <doi:10.1162/NECO_a_00961>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports Rcpp, stats, utils, np, rje, Rdpack, RcppEigen

Suggests mclust

LinkingTo Rcpp, RcppArmadillo, RcppEigen

RoxygenNote 7.3.2

RdMacros Rdpack

NeedsCompilation yes

Author Efthymios Costa [aut],
Ioanna Papatsouma [aut],
Angelos Markos [aut, cre]

Maintainer Angelos Markos <amarkos@gmail.com>

Archs x64

Contents

AIBmix — *Agglomerative Information Bottleneck Clustering for Mixed-Type Data*

Description

The `AIBmix` function implements the Agglomerative Information Bottleneck (AIB) algorithm for hierarchical clustering of datasets containing mixed-type variables, including categorical (nominal and ordinal) and continuous variables. This method merges clusters so that information retention is maximised at each step to create meaningful clusters, leveraging bandwidth parameters to handle different categorical data types (nominal and ordinal) effectively (Slonim and Tishby 1999).

Usage

```
AIBmix(X, s = -1, lambda = -1,
       scale = TRUE, contkernel = "gaussian",
       nomkernel = "aitchisonaitken", ordkernel = "liracine",
       cat_first = FALSE)
```

Arguments

`X` A data frame containing the data to be clustered. Variables should be of type `numeric` (for continuous variables), `factor` (for nominal variables) or `ordered` (for ordinal variables).

`s` A numeric value or vector specifying the bandwidth parameter(s) for continuous variables. The values must be greater than 0. The default value is -1, which enables the automatic selection of optimal bandwidth(s). Argument is ignored when no variables are continuous.

`lambda` A numeric value or vector specifying the bandwidth parameter for categorical variables. The default value is -1, which enables automatic determination of the optimal bandwidth. For nominal variables and `nomkernel = 'aitchisonaitken'`, the maximum allowable value of `lambda` is $(l-1)/l$, where l represents the number of categories, whereas for `nomkernel = 'liracine'` the maximum allowable value is 1. For ordinal variables, the maximum allowable value of `lambda` is 1, regardless of what `ordkernel` is being used. Argument is ignored when all variables are continuous.

`scale` A logical value indicating whether the continuous variables should be scaled to have unit variance before clustering. Defaults to `TRUE`. Argument is ignored when all variables are categorical.

`contkernel` Kernel used for continuous variables. Can be one of `gaussian` (default) or `epanechnikov`. Argument is ignored when no variables are continuous.

`nomkernel` Kernel used for nominal (unordered categorical) variables. Can be one of `aitchisonaitken` (default) or `liracine`. Argument is ignored when no variables are nominal.

`ordkernel` Kernel used for ordinal (ordered categorical) variables. Can be one of `liracine` (default) or `wangvanryzin`. Argument is ignored when no variables are ordinal.

`cat_first` A logical value indicating whether bandwidth selection is prioritised for the categorical variables, instead of the continuous. Defaults to `FALSE`. Set to `TRUE` if you suspect that the continuous variables are not informative of the cluster structure. Can only be `TRUE` when all bandwidths are selected automatically (i.e. `s = -1`, `lambda = -1`).

Details

The `AIBmix` function produces a hierarchical agglomerative clustering of the data while retaining maximal information about the original variable distributions. The Agglomerative Information Bottleneck algorithm uses an information-theoretic criterion to merge clusters so that information retention is maximised at each step, hence creating meaningful clusters with maximal information about the original distribution. Bandwidth parameters for categorical (nominal, ordinal) and continuous variables are adaptively determined if not provided. This process identifies stable and interpretable cluster assignments by maximizing mutual information while controlling complexity. The method is well-suited for datasets with mixed-type variables and integrates information from all variable types effectively.

The following kernel functions can be used to estimate densities for the clustering procedure. For continuous variables:

- *Gaussian (RBF) kernel (Silverman 1998):*

$$K_c\left(\frac{x-x'}{s}\right) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(x-x')^2}{2s^2}\right\}, \quad s > 0.$$

- *Epanechnikov kernel (Epanechnikov 1969):*

$$K_c(x-x';s) = \begin{cases} \frac{3}{4\sqrt{5}}\left(1-\frac{(x-x')^2}{5s^2}\right), & \text{if } \frac{(x-x')^2}{s^2} < 5 \\ 0, & \text{otherwise} \end{cases}, \quad s > 0.$$

For nominal (unordered categorical variables):

- *Aitchison & Aitken kernel (Aitchison and Aitken 1976):*

$$K_u(x=x';\lambda) = \begin{cases} 1-\lambda, & \text{if } x = x' \\ \frac{\lambda}{\ell-1}, & \text{otherwise} \end{cases}, \quad 0 \le \lambda \le \frac{\ell-1}{\ell}.$$

- *Li & Racine kernel (Ouyang et al. 2006):*

$$K_u(x=x';\lambda) = \begin{cases} 1, & \text{if } x = x' \\ \lambda, & \text{otherwise} \end{cases}, \quad 0 \le \lambda \le 1.$$

For ordinal (ordered categorical) variables:

- *Li & Racine kernel (Li and Racine 2003):*

$$K_o(x=x';\nu) = \begin{cases} 1, & \text{if } x = x' \\ \nu^{|x-x'|}, & \text{otherwise} \end{cases}, \quad 0 \le \nu \le 1.$$

- *Wang & van Ryzin kernel (Wang and Van Ryzin 1981):*

$$K_o(x=x';\nu) = \begin{cases} 1-\nu, & \text{if } x = x' \\ \frac{1-\nu}{2}\nu^{|x-x'|}, & \text{otherwise} \end{cases}, \quad 0 \le \nu \le 1.$$

The bandwidth parameters s, λ, and ν control the smoothness of the density estimate and are automatically determined by the algorithm if not provided by the user using the approach in Costa et al. (2025). ℓ is the number of levels of the categorical variable. For ordinal variables, the lambda parameter of the function is used to define ν.

Value

A list containing the following elements:

`merges`	A data frame with 2 columns and n rows, showing which observations are merged at each step.
`merge_costs`	A numeric vector tracking the cost incurred by each merge $I(Z_m; Y) - I(Z_{m-1}; Y)$.
`partitions`	A list containing n sub-lists. Each sub-list includes the cluster partition at each step.
`I_Z_Y`	A numeric vector including the mutual information $I(Z_m; Y)$ as the number of clusters m increases.
`I_X_Y`	A numeric value of the mutual information $I(X; Y)$ between observation indices and location.
`info_ret`	A numeric vector of length n including the fraction of the original information retained after each merge.
`dendrogram`	A dendrogram visualising the cluster hierarchy. The height is determined by the cost of cluster merges.
`s`	A numeric vector of bandwidth parameters used for the continuous variables. A value of -1 is returned if all variables are categorical.
`lambda`	A numeric vector of bandwidth parameters used for the categorical variables. A value of -1 is returned if all variables are continuous.

Author(s)

Efthymios Costa, Ioanna Papatsouma, Angelos Markos

References

Slonim N, Tishby N (1999). "Agglomerative Information Bottleneck." *Advances in Neural Information Processing Systems*, **12**.

Aitchison J, Aitken CG (1976). "Multivariate binary discrimination by the kernel method." *Biometrika*, **63**(3), 413–420.

Li Q, Racine J (2003). "Nonparametric estimation of distributions with categorical and continuous data." *Journal of Multivariate Analysis*, **86**(2), 266–292.

Silverman BW (1998). *Density Estimation for Statistics and Data Analysis (1st Ed.)*. Routledge.

Ouyang D, Li Q, Racine J (2006). "Cross-validation and the estimation of probability distributions with categorical data." *Journal of Nonparametric Statistics*, **18**(1), 69–100.

Wang M, Van Ryzin J (1981). "A class of smooth estimators for discrete distributions." *Biometrika*, **68**(1), 301–309.

Epanechnikov VA (1969). "Non-parametric estimation of a multivariate probability density." *Theory of Probability & Its Applications*, **14**(1), 153–158.

Costa E, Papatsouma I, Markos A (2025). "A Deterministic Information Bottleneck Method for Clustering Mixed-Type Data." doi:10.48550/arXiv.2407.03389, arXiv:2407.03389, `https://arxiv.org/abs/2407.03389`.

Examples

```
# Example dataset with categorical, ordinal, and continuous variables
set.seed(123)
data_mix <- data.frame(
 cat_var = factor(sample(letters[1:3], 100, replace = TRUE)),    # Nominal categorical variable
  ord_var = factor(sample(c("low", "medium", "high"), 100, replace = TRUE),
                   levels = c("low", "medium", "high"),
                   ordered = TRUE),                                # Ordinal variable
  cont_var1 = rnorm(100),                                        # Continuous variable 1
  cont_var2 = runif(100)                                         # Continuous variable 2
)

# Perform Mixed-Type Hierarchical Clustering with Agglomerative IB
result_mix <- AIBmix(X = data_mix, lambda = -1, s = -1, scale = TRUE)

# Print clustering results
plot(result_mix$dendrogram, xlab = "", sub = "")  # Plot dendrogram

# Simulated categorical data example
set.seed(123)
data_cat <- data.frame(
  Var1 = as.factor(sample(letters[1:3], 200, replace = TRUE)),  # Nominal variable
  Var2 = as.factor(sample(letters[4:6], 200, replace = TRUE)),  # Nominal variable
  Var3 = factor(sample(c("low", "medium", "high"), 200, replace = TRUE),
                levels = c("low", "medium", "high"), ordered = TRUE)  # Ordinal variable
)

# Run AIBmix with automatic lambda selection
result_cat <- AIBmix(X = data_cat, lambda = -1)

# Print clustering results
plot(result_cat$dendrogram, xlab = "", sub = "")  # Plot dendrogram

# Simulated continuous data example
set.seed(123)
# Continuous data with 200 observations, 5 features
data_cont <- as.data.frame(matrix(rnorm(1000), ncol = 5))

# Run AIBmix with automatic bandwidth selection
result_cont <- AIBmix(X = data_cont, s = -1, scale = TRUE)

# Print clustering results
plot(result_cont$dendrogram, xlab = "", sub = "")  # Plot dendrogram
```

`DIBmix` *Deterministic Information Bottleneck Clustering for Mixed-Type Data*

Description

The `DIBmix` function implements the Deterministic Information Bottleneck (DIB) algorithm for clustering datasets containing continuous, categorical (nominal and ordinal), and mixed-type variables. This method optimizes an information-theoretic objective to preserve relevant information in the cluster assignments while achieving effective data compression (Costa et al. 2025).

Usage

```
DIBmix(X, ncl, randinit = NULL,
       s = -1, lambda = -1, scale = TRUE,
       maxiter = 100, nstart = 100,
       contkernel = "gaussian",
       nomkernel = "aitchisonaitken", ordkernel = "liracine",
       cat_first = FALSE, verbose = FALSE)
```

Arguments

`X`	A data frame containing the input data to be clustered. It should include categorical variables (`factor` for nominal and `ordered` for ordinal) and continuous variables (`numeric`).
`ncl`	An integer specifying the number of clusters.
`randinit`	An optional vector specifying the initial cluster assignments. If `NULL`, cluster assignments are initialized randomly.
`s`	A numeric value or vector specifying the bandwidth parameter(s) for continuous variables. The values must be greater than 0. The default value is -1, which enables the automatic selection of optimal bandwidth(s). Argument is ignored when no variables are continuous.
`lambda`	A numeric value or vector specifying the bandwidth parameter for categorical variables. The default value is -1, which enables automatic determination of the optimal bandwidth. For nominal variables and `nomkernel = 'aitchisonaitken'`, the maximum allowable value of `lambda` is $(l-1)/l$, where l represents the number of categories, whereas for `nomkernel = 'liracine'` the maximum allowable value is 1. For ordinal variables, the maximum allowable value of `lambda` is 1, regardless of what `ordkernel` is being used. Argument is ignored when all variables are continuous.
`scale`	A logical value indicating whether the continuous variables should be scaled to have unit variance before clustering. Defaults to `TRUE`. Argument is ignored when all variables are categorical.
`maxiter`	The maximum number of iterations allowed for the clustering algorithm. Defaults to 100.
`nstart`	The number of random initializations to run. The best clustering solution is returned. Defaults to 100.
`contkernel`	Kernel used for continuous variables. Can be one of `gaussian` (default) or `epanechnikov`. Argument is ignored when no variables are continuous.
`nomkernel`	Kernel used for nominal (unordered categorical) variables. Can be one of `aitchisonaitken` (default) or `liracine`. Argument is ignored when no variables are nominal.
`ordkernel`	Kernel used for ordinal (ordered categorical) variables. Can be one of `liracine` (default) or `wangvanryzin`. Argument is ignored when no variables are ordinal.
`cat_first`	A logical value indicating whether bandwidth selection is prioritised for the categorical variables, instead of the continuous. Defaults to `FALSE`. Set to `TRUE` if you suspect that the continuous variables are not informative of the cluster structure. Can only be `TRUE` when data is of mixed-type and all bandwidths are selected automatically (i.e. `s = -1`, `lambda = -1`).
`verbose`	Logical. Defaults to `FALSE` to suppress progress messages. Change to `TRUE` to print.

Details

The `DIBmix` function clusters data while retaining maximal information about the original variable distributions. The Deterministic Information Bottleneck algorithm optimizes an information-theoretic objective that balances information preservation and compression. Bandwidth parameters for categorical (nominal, ordinal) and continuous variables are adaptively determined if not provided. This iterative process identifies stable and interpretable cluster assignments by maximizing mutual information while controlling complexity. The method is well-suited for datasets with mixed-type variables and integrates information from all variable types effectively.

The following kernel functions can be used to estimate densities for the clustering procedure. For continuous variables:

- *Gaussian (RBF) kernel (Silverman 1998):*

$$K_c\left(\frac{x-x'}{s}\right) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(x-x')^2}{2s^2}\right\}, \quad s > 0.$$

- *Epanechnikov kernel (Epanechnikov 1969):*

$$K_c(x-x';s) = \begin{cases} \frac{3}{4\sqrt{5}}\left(1-\frac{(x-x')^2}{5s^2}\right), & \text{if } \frac{(x-x')^2}{s^2} < 5 \\ 0, & \text{otherwise} \end{cases}, \quad s > 0.$$

For nominal (unordered categorical variables):

- *Aitchison & Aitken kernel (Aitchison and Aitken 1976):*

$$K_u(x=x';\lambda) = \begin{cases} 1-\lambda, & \text{if } x = x' \\ \frac{\lambda}{\ell-1}, & \text{otherwise} \end{cases}, \quad 0 \le \lambda \le \frac{\ell-1}{\ell}.$$

- *Li & Racine kernel (Ouyang et al. 2006):*

$$K_u(x=x';\lambda) = \begin{cases} 1, & \text{if } x = x' \\ \lambda, & \text{otherwise} \end{cases}, \quad 0 \le \lambda \le 1.$$

For ordinal (ordered categorical) variables:

- *Li & Racine kernel (Li and Racine 2003):*

$$K_o(x=x';\nu) = \begin{cases} 1, & \text{if } x = x' \\ \nu^{|x-x'|}, & \text{otherwise} \end{cases}, \quad 0 \le \nu \le 1.$$

- *Wang & van Ryzin kernel (Wang and Van Ryzin 1981):*

$$K_o(x=x';\nu) = \begin{cases} 1-\nu, & \text{if } x = x' \\ \frac{1-\nu}{2}\nu^{|x-x'|}, & \text{otherwise} \end{cases}, \quad 0 \le \nu \le 1.$$

The bandwidth parameters s, λ, and ν control the smoothness of the density estimate and are automatically determined by the algorithm if not provided by the user using the approach in Costa et al. (2025). ℓ is the number of levels of the categorical variable. For ordinal variables, the lambda parameter of the function is used to define ν.

Value

An object of class `"gibclust"` representing the final clustering result. The returned object is a list with the following components:

- `Cluster` — An integer vector giving the cluster assignments for each data point.
- `Entropy` — A numeric value representing the entropy of the cluster assignments at convergence.
- `CondEntropy` — A numeric value representing the conditional entropy of cluster assignment, given the observation weights $H(T \mid X)$.
- `MutualInfo` — A numeric value representing the mutual information, $I(Y;T)$, between the original labels (Y) and the cluster assignments (T).
- `InfoXT` — A numeric value representing the mutual information, $I(X;T)$, between the original observations weights (X) and the cluster assignments (T).
- `beta` — A numeric vector of the final beta values used in the iterative procedure.
- `alpha` — A numeric value of the strength of conditional entropy used, controlling fuzziness of the solution. This is by default equal to 0 for `DIBmix`.
- `s` — A numeric vector of bandwidth parameters used for the continuous variables. A value of -1 is returned if all variables are categorical.
- `lambda` — A numeric vector of bandwidth parameters used for the categorical variables. A value of -1 is returned if all variables are continuous.
- `call` — The matched call.
- `ncl` — Number of clusters.
- `n` — Number of observations.
- `iters` — Number of iterations used to obtain the returned solution.
- `converged` — Logical indicating whether convergence was reached before `maxiter`.
- `conv_tol` — Numeric convergence tolerance; by default 0 for `DIBmix`.
- `contcols` — Indices of continuous columns in `X`.
- `catcols` — Indices of categorical columns in `X`.
- `kernels` — List with names of kernels used for continuous, nominal, and ordinal features.

Objects of class `"gibclust"` support the following methods:

- `print.gibclust`: Display a concise description of the fitted clustering.
- `summary.gibclust`: Show detailed information including cluster sizes, information-theoretic metrics, hyperparameters, and convergence details.
- `plot.gibclust`: Produce diagnostic plots:
  - `type = "sizes"`: barplot of cluster sizes or hardened sizes (IB/GIB).
  - `type = "info"`: barplot of entropy, conditional entropy, and mutual information.
  - `type = "beta"`: trajectory of $\log \beta$ over iterations (only available for hard clustering outputs obtained using `DIBmix`).

Author(s)

Efthymios Costa, Ioanna Papatsouma, Angelos Markos

References

Costa E, Papatsouma I, Markos A (2025). "A Deterministic Information Bottleneck Method for Clustering Mixed-Type Data." doi:10.48550/arXiv.2407.03389, arXiv:2407.03389, https://arxiv.org/abs/2407.03389.

Aitchison J, Aitken CG (1976). "Multivariate binary discrimination by the kernel method." *Biometrika*, **63**(3), 413–420.

Li Q, Racine J (2003). "Nonparametric estimation of distributions with categorical and continuous data." *Journal of Multivariate Analysis*, **86**(2), 266–292.

Silverman BW (1998). *Density Estimation for Statistics and Data Analysis (1st Ed.)*. Routledge.

Ouyang D, Li Q, Racine J (2006). "Cross-validation and the estimation of probability distributions with categorical data." *Journal of Nonparametric Statistics*, **18**(1), 69–100.

Wang M, Van Ryzin J (1981). "A class of smooth estimators for discrete distributions." *Biometrika*, **68**(1), 301–309.

Epanechnikov VA (1969). "Non-parametric estimation of a multivariate probability density." *Theory of Probability & Its Applications*, **14**(1), 153–158.

Examples

```
# Example 1: Basic Mixed-Type Clustering
set.seed(123)

# Create a more realistic dataset with mixed variable types
data_mix <- data.frame(
  # Categorical variables
  education = factor(sample(c("High School", "Bachelor", "Master", "PhD"), 150,
                            replace = TRUE, prob = c(0.4, 0.3, 0.2, 0.1))),
  employment = factor(sample(c("Full-time", "Part-time", "Unemployed"), 150,
                             replace = TRUE, prob = c(0.6, 0.25, 0.15))),

  # Ordinal variable
  satisfaction = factor(sample(c("Low", "Medium", "High"), 150, replace = TRUE),
                        levels = c("Low", "Medium", "High"), ordered = TRUE),

  # Continuous variables
  income = rlnorm(150, meanlog = 10, sdlog = 0.5),  # Log-normal income
  age = rnorm(150, mean = 35, sd = 10),             # Normally distributed age
  experience = rpois(150, lambda = 8)               # Years of experience
)

# Perform Mixed-Type Clustering
result_mix <- DIBmix(X = data_mix, ncl = 3, nstart = 50)

# View results
print(paste("Number of clusters found:", length(unique(result_mix$Cluster))))
print(paste("Mutual Information:", round(result_mix$MutualInfo, 3)))
table(result_mix$Cluster)

# Example 2: Comparing cat_first parameter
# When categorical variables are more informative
result_cat_first <- DIBmix(X = data_mix, ncl = 3,
                           cat_first = TRUE,  # Prioritize categorical variables
                           nstart = 50)
```

```
# When continuous variables are more informative (default)
result_cont_first <- DIBmix(X = data_mix, ncl = 3,
                            cat_first = FALSE,
                            nstart = 50)

# Compare clustering performance
if (requireNamespace("mclust", quietly = TRUE)){  # For adjustedRandIndex
  print(paste("Agreement between approaches:",
              round(mclust::adjustedRandIndex(result_cat_first$Cluster,
                    result_cont_first$Cluster), 3)))
  }

plot(result_cat_first, type = "sizes") # Bar plot of cluster sizes
plot(result_cat_first, type = "info")  # Information-theoretic quantities plot
plot(result_cat_first, type = "beta")  # Plot of evolution of beta

# Simulated categorical data example
data_cat <- data.frame(
  Var1 = as.factor(sample(letters[1:3], 200, replace = TRUE)),  # Nominal variable
  Var2 = as.factor(sample(letters[4:6], 200, replace = TRUE)),  # Nominal variable
  Var3 = factor(sample(c("low", "medium", "high"), 200, replace = TRUE),
                levels = c("low", "medium", "high"), ordered = TRUE)  # Ordinal variable
)

# Perform hard clustering on categorical data with Deterministic IB
result_cat <- DIBmix(X = data_cat, ncl = 3, lambda = -1, nstart = 10)

# Print clustering results
print(result_cat$Cluster)       # Cluster assignments
print(result_cat$Entropy)       # Final entropy
print(result_cat$MutualInfo)    # Mutual information

# Simulated continuous data example
set.seed(123)
# Continuous data with 200 observations, 5 features
data_cont <- as.data.frame(matrix(rnorm(1000), ncol = 5))

# Perform hard clustering on continuous data with Deterministic IB
result_cont <- DIBmix(X = data_cont, ncl = 3, s = -1, nstart = 10)

# Print clustering results
print(result_cont$Cluster)       # Cluster assignments
print(result_cont$Entropy)       # Final entropy
print(result_cont$MutualInfo)    # Mutual information

# Summary of output
print(result_cont)
summary(result_cont)
```

GIBmix *Generalised Information Bottleneck Clustering for Mixed-Type Data*

Description

The GIBmix function implements the Generalised Information Bottleneck (GIB) algorithm for clustering datasets containing continuous, categorical (nominal and ordinal), and mixed-type variables.

This method optimizes an information-theoretic objective to preserve relevant information in the cluster assignments while achieving effective data compression (Strouse and Schwab 2017).

Usage

```
GIBmix(X, ncl, beta, alpha, randinit = NULL,
       s = -1, lambda = -1, scale = TRUE,
       maxiter = 100, nstart = 100,
       conv_tol = 1e-5, contkernel = "gaussian",
       nomkernel = "aitchisonaitken", ordkernel = "liracine",
       cat_first = FALSE, verbose = FALSE)
```

Arguments

`X`	A data frame containing the input data to be clustered. It should include categorical variables (`factor` for nominal and `ordered` for ordinal) and continuous variables (`numeric`).
`ncl`	An integer specifying the number of clusters.
`beta`	Regularisation strength characterizing the tradeoff between compression and relevance. Must be non-negative.
`alpha`	Strength of conditional entropy term. Must be in the range $[0, 1]$. Setting `alpha = 0` calls the `DIBmix` function and ignores the value of `beta`, while `alpha = 1` calls `IBmix` instead.
`randinit`	An optional vector specifying the initial cluster assignments. If `NULL`, cluster assignments are initialized randomly.
`s`	A numeric value or vector specifying the bandwidth parameter(s) for continuous variables. The values must be greater than 0. The default value is -1, which enables the automatic selection of optimal bandwidth(s). Argument is ignored when no variables are continuous.
`lambda`	A numeric value or vector specifying the bandwidth parameter for categorical variables. The default value is -1, which enables automatic determination of the optimal bandwidth. For nominal variables and `nomkernel = 'aitchisonaitken'`, the maximum allowable value of `lambda` is $(l-1)/l$, where l represents the number of categories, whereas for `nomkernel = 'liracine'` the maximum allowable value is 1. For ordinal variables, the maximum allowable value of `lambda` is 1, regardless of what `ordkernel` is being used. Argument is ignored when all variables are continuous.
`scale`	A logical value indicating whether the continuous variables should be scaled to have unit variance before clustering. Defaults to `TRUE`. Argument is ignored when all variables are categorical.
`maxiter`	The maximum number of iterations allowed for the clustering algorithm. Defaults to 100.
`nstart`	The number of random initializations to run. The best clustering solution is returned. Defaults to 100.
`conv_tol`	Convergence tolerance level; for a cluster membership matrix $U^{(m)}$ at iteration m, convergence is achieved if $\sum_{i,j} \lvert U_{i,j}^{m+1} - U_{i,j}^{m} \rvert \leq$ `conv_tol`. Must be in range $[0, 1]$. Defaults to `1e-5`.
`contkernel`	Kernel used for continuous variables. Can be one of `gaussian` (default) or `epanechnikov`. Argument is ignored when no variables are continuous.

`nomkernel`	Kernel used for nominal (unordered categorical) variables. Can be one of `aitchisonaitken` (default) or `liracine`. Argument is ignored when no variables are nominal.
`ordkernel`	Kernel used for ordinal (ordered categorical) variables. Can be one of `liracine` (default) or `wangvanryzin`. Argument is ignored when no variables are ordinal.
`cat_first`	A logical value indicating whether bandwidth selection is prioritised for the categorical variables, instead of the continuous. Defaults to `FALSE`. Set to `TRUE` if you suspect that the continuous variables are not informative of the cluster structure. Can only be `TRUE` when data is of mixed-type and all bandwidths are selected automatically (i.e. `s = -1`, `lambda = -1`).
`verbose`	Logical. Defaults to `FALSE` to suppress progress messages. Change to `TRUE` to print.

Details

The `GIBmix` function produces a fuzzy clustering of the data while retaining maximal information about the original variable distributions. The Generalised Information Bottleneck algorithm optimizes an information-theoretic objective that balances information preservation and compression. Bandwidth parameters for categorical (nominal, ordinal) and continuous variables are adaptively determined if not provided. This iterative process identifies stable and interpretable cluster assignments by maximizing mutual information while controlling complexity. The method is well-suited for datasets with mixed-type variables and integrates information from all variable types effectively. Set $\alpha = 1$ and $\alpha = 0$ to recover the Information Bottleneck and its Deterministic variant, respectively. If $\alpha = 0$, the algorithm ignores the value of the regularisation parameter β.

The following kernel functions can be used to estimate densities for the clustering procedure. For continuous variables:

- *Gaussian (RBF) kernel (Silverman 1998):*

$$K_c\left(\frac{x-x'}{s}\right) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(x-x')^2}{2s^2}\right\}, \quad s > 0.$$

- *Epanechnikov kernel (Epanechnikov 1969):*

$$K_c(x-x';s) = \begin{cases} \frac{3}{4\sqrt{5}}\left(1-\frac{(x-x')^2}{5s^2}\right), & \text{if } \frac{(x-x')^2}{s^2} < 5 \\ 0, & \text{otherwise} \end{cases}, \quad s > 0.$$

For nominal (unordered categorical variables):

- *Aitchison & Aitken kernel (Aitchison and Aitken 1976):*

$$K_u(x = x';\lambda) = \begin{cases} 1-\lambda, & \text{if } x = x' \\ \frac{\lambda}{\ell-1}, & \text{otherwise} \end{cases}, \quad 0 \le \lambda \le \frac{\ell-1}{\ell}.$$

- *Li & Racine kernel (Ouyang et al. 2006):*

$$K_u(x = x';\lambda) = \begin{cases} 1, & \text{if } x = x' \\ \lambda, & \text{otherwise} \end{cases}, \quad 0 \le \lambda \le 1.$$

For ordinal (ordered categorical) variables:

- *Li & Racine kernel (Li and Racine 2003):*

$$K_o(x = x'; \nu) = \begin{cases} 1, & \text{if } x = x' \\ \nu^{|x-x'|}, & \text{otherwise} \end{cases}, \quad 0 \leq \nu \leq 1.$$

- *Wang & van Ryzin kernel (Wang and Van Ryzin 1981):*

$$K_o(x = x'; \nu) = \begin{cases} 1 - \nu, & \text{if } x = x' \\ \frac{1-\nu}{2}\nu^{|x-x'|}, & \text{otherwise} \end{cases}, \quad 0 \leq \nu \leq 1.$$

The bandwidth parameters s, λ, and ν control the smoothness of the density estimate and are automatically determined by the algorithm if not provided by the user using the approach in Costa et al. (2025). ℓ is the number of levels of the categorical variable. For ordinal variables, the lambda parameter of the function is used to define ν.

Value

An object of class `"gibclust"` representing the final clustering result. The returned object is a list with the following components:

`Cluster` An integer vector giving the cluster assignments for each data point.

`Entropy` A numeric value representing the entropy of the cluster assignments at convergence.

`CondEntropy` A numeric value representing the conditional entropy of cluster assignment, given the observation weights $H(T \mid X)$.

`MutualInfo` A numeric value representing the mutual information, $I(Y;T)$, between the original labels (Y) and the cluster assignments (T).

`InfoXT` A numeric value representing the mutual information, $I(X;T)$, between the original observations weights (X) and the cluster assignments (T).

`beta` A numeric vector of the final beta values used in the iterative procedure.

`alpha` A numeric value of the strength of conditional entropy used, controlling fuzziness of the solution. This is by default equal to 0 for `DIBmix`.

`s` A numeric vector of bandwidth parameters used for the continuous variables. A value of -1 is returned if all variables are categorical.

`lambda` A numeric vector of bandwidth parameters used for the categorical variables. A value of -1 is returned if all variables are continuous.

`call` The matched call.

`ncl` Number of clusters.

`n` Number of observations.

`iters` Number of iterations used to obtain the returned solution.

`converged` Logical indicating whether convergence was reached before `maxiter`.

`conv_tol` Numeric convergence tolerance.

`contcols` Indices of continuous columns in `X`.

`catcols` Indices of categorical columns in `X`.

`kernels` List with names of kernels used for continuous, nominal, and ordinal features.

Objects of class `"gibclust"` support the following methods:

- `print.gibclust`: Display a concise description of the fitted clustering.
- `summary.gibclust`: Show detailed information including cluster sizes, information-theoretic metrics, hyperparameters, and convergence details.
- `plot.gibclust`: Produce diagnostic plots:
  - `type = "sizes"`: barplot of cluster sizes or hardened sizes (IB/GIB).
  - `type = "info"`: barplot of entropy, conditional entropy, and mutual information.
  - `type = "beta"`: trajectory of $\log \beta$ over iterations (only available for hard clustering outputs obtained using `DIBmix`).

Author(s)

Efthymios Costa, Ioanna Papatsouma, Angelos Markos

References

Strouse DJ, Schwab DJ (2017). "The Deterministic Information Bottleneck." *Neural Computation*, **29**(6), 1611–1630.

Aitchison J, Aitken CG (1976). "Multivariate binary discrimination by the kernel method." *Biometrika*, **63**(3), 413–420.

Li Q, Racine J (2003). "Nonparametric estimation of distributions with categorical and continuous data." *Journal of Multivariate Analysis*, **86**(2), 266–292.

Silverman BW (1998). *Density Estimation for Statistics and Data Analysis (1st Ed.)*. Routledge.

Ouyang D, Li Q, Racine J (2006). "Cross-validation and the estimation of probability distributions with categorical data." *Journal of Nonparametric Statistics*, **18**(1), 69–100.

Wang M, Van Ryzin J (1981). "A class of smooth estimators for discrete distributions." *Biometrika*, **68**(1), 301–309.

Epanechnikov VA (1969). "Non-parametric estimation of a multivariate probability density." *Theory of Probability & Its Applications*, **14**(1), 153–158.

Examples

```
# Example dataset with categorical, ordinal, and continuous variables
set.seed(123)
data_mix <- data.frame(
  cat_var = factor(sample(letters[1:3], 100, replace = TRUE)),    # Nominal categorical variable
  ord_var = factor(sample(c("low", "medium", "high"), 100, replace = TRUE),
                   levels = c("low", "medium", "high"),
                   ordered = TRUE),                                # Ordinal variable
  cont_var1 = rnorm(100),                                        # Continuous variable 1
  cont_var2 = runif(100)                                         # Continuous variable 2
)

# Perform Mixed-Type Fuzzy Clustering with Generalised IB
result_mix <- GIBmix(X = data_mix, ncl = 3, beta = 2, alpha = 0.5, nstart = 20)

# Print clustering results
print(result_mix$Cluster)       # Cluster membership matrix
print(result_mix$Entropy)       # Entropy of final clustering
print(result_mix$CondEntropy)   # Conditional entropy of final clustering
print(result_mix$MutualInfo)    # Mutual information between Y and T

# Summary of output
```

```
summary(result_mix)

# Simulated categorical data example
set.seed(123)
data_cat <- data.frame(
  Var1 = as.factor(sample(letters[1:3], 200, replace = TRUE)),  # Nominal variable
  Var2 = as.factor(sample(letters[4:6], 200, replace = TRUE)),  # Nominal variable
  Var3 = factor(sample(c("low", "medium", "high"), 200, replace = TRUE),
                levels = c("low", "medium", "high"), ordered = TRUE)  # Ordinal variable
)

# Perform Fuzzy Clustering on categorical data with Generalised IB
result_cat <- GIBmix(X = data_cat, ncl = 2, beta = 25, alpha = 0.75, lambda = -1, nstart = 10)

# Print clustering results
print(result_cat$Cluster)       # Cluster membership matrix
print(result_cat$Entropy)       # Entropy of final clustering
print(result_cat$CondEntropy)   # Conditional entropy of final clustering
print(result_cat$MutualInfo)    # Mutual information between Y and T

# Simulated continuous data example
set.seed(123)
# Continuous data with 200 observations, 5 features
data_cont <- as.data.frame(matrix(rnorm(1000), ncol = 5))

# Perform Fuzzy Clustering on continuous data with Generalised IB
result_cont <- GIBmix(X = data_cont, ncl = 2, beta = 50, alpha = 0.75, s = -1, nstart = 10)

# Print clustering results
print(result_cont$Cluster)       # Cluster membership matrix
print(result_cont$Entropy)       # Entropy of final clustering
print(result_cont$CondEntropy)   # Conditional entropy of final clustering
print(result_cont$MutualInfo)    # Mutual information between Y and T

plot(result_cont, type = "sizes") # Bar plot of cluster sizes (hardened assignments)
plot(result_cont, type = "info")  # Information-theoretic quantities plot
```

`IBmix` *Information Bottleneck Clustering for Mixed-Type Data*

Description

The `IBmix` function implements the Information Bottleneck (IB) algorithm for clustering datasets containing continuous, categorical (nominal and ordinal), and mixed-type variables. This method optimizes an information-theoretic objective to preserve relevant information in the cluster assignments while achieving effective data compression (Strouse and Schwab 2019).

Usage

```
IBmix(X, ncl, beta, randinit = NULL,
       s = -1, lambda = -1, scale = TRUE,
       maxiter = 100, nstart = 100,
       conv_tol = 1e-5, contkernel = "gaussian",
       nomkernel = "aitchisonaitken", ordkernel = "liracine",
       cat_first = FALSE, verbose = FALSE)
```

Arguments

`X`	A data frame containing the input data to be clustered. It should include categorical variables (`factor` for nominal and `Ord.factor` for ordinal) and continuous variables (`numeric`).
`ncl`	An integer specifying the number of clusters.
`beta`	Regularisation strength characterizing the tradeoff between compression and relevance. Must be non-negative.
`randinit`	An optional vector specifying the initial cluster assignments. If `NULL`, cluster assignments are initialized randomly.
`s`	A numeric value or vector specifying the bandwidth parameter(s) for continuous variables. The values must be greater than 0. The default value is -1, which enables the automatic selection of optimal bandwidth(s). Argument is ignored when no variables are continuous.
`lambda`	A numeric value or vector specifying the bandwidth parameter for categorical variables. The default value is -1, which enables automatic determination of the optimal bandwidth. For nominal variables and `nomkernel = 'aitchisonaitken'`, the maximum allowable value of `lambda` is $(l-1)/l$, where l represents the number of categories, whereas for `nomkernel = 'liracine'` the maximum allowable value is 1. For ordinal variables, the maximum allowable value of `lambda` is 1, regardless of what `ordkernel` is being used. Argument is ignored when all variables are continuous.
`scale`	A logical value indicating whether the continuous variables should be scaled to have unit variance before clustering. Defaults to `TRUE`. Argument is ignored when all variables are categorical.
`maxiter`	The maximum number of iterations allowed for the clustering algorithm. Defaults to 100.
`nstart`	The number of random initializations to run. The best clustering solution is returned. Defaults to 100.
`conv_tol`	Convergence tolerance level; for a cluster membership matrix $U^{(m)}$ at iteration m, convergence is achieved if $\sum_{i,j}\lvert U_{i,j}^{m+1} - U_{i,j}^{m}\rvert \leq$ `conv_tol`. Must be in range $[0, 1]$. Defaults to `1e-5`.
`contkernel`	Kernel used for continuous variables. Can be one of `gaussian` (default) or `epanechnikov`. Argument is ignored when no variables are continuous.
`nomkernel`	Kernel used for nominal (unordered categorical) variables. Can be one of `aitchisonaitken` (default) or `liracine`. Argument is ignored when no variables are nominal.
`ordkernel`	Kernel used for ordinal (ordered categorical) variables. Can be one of `liracine` (default) or `wangvanryzin`. Argument is ignored when no variables are ordinal.
`cat_first`	A logical value indicating whether bandwidth selection is prioritised for the categorical variables, instead of the continuous. Defaults to `FALSE`. Set to `TRUE` if you suspect that the continuous variables are not informative of the cluster structure. Can only be `TRUE` when data is of mixed-type and all bandwidths are selected automatically (i.e. `s = -1`, `lambda = -1`).
`verbose`	Logical. Defaults to `FALSE` to suppress progress messages. Change to `TRUE` to print.

Details

The `IBmix` function produces a fuzzy clustering of the data while retaining maximal information about the original variable distributions. The Information Bottleneck algorithm optimizes an information-theoretic objective that balances information preservation and compression. Bandwidth parameters for categorical (nominal, ordinal) and continuous variables are adaptively determined if not provided. This iterative process identifies stable and interpretable cluster assignments by maximizing mutual information while controlling complexity. The method is well-suited for datasets with mixed-type variables and integrates information from all variable types effectively.

The following kernel functions can be used to estimate densities for the clustering procedure. For continuous variables:

- *Gaussian (RBF) kernel (Silverman 1998):*

$$K_c\left(\frac{x-x'}{s}\right) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(x-x')^2}{2s^2}\right\}, \quad s > 0.$$

- *Epanechnikov kernel (Epanechnikov 1969):*

$$K_c(x-x';s) = \begin{cases} \frac{3}{4\sqrt{5}}\left(1-\frac{(x-x')^2}{5s^2}\right), & \text{if } \frac{(x-x')^2}{s^2} < 5 \\ 0, & \text{otherwise} \end{cases}, \quad s > 0.$$

For nominal (unordered categorical variables):

- *Aitchison & Aitken kernel (Aitchison and Aitken 1976):*

$$K_u(x=x';\lambda) = \begin{cases} 1-\lambda, & \text{if } x = x' \\ \frac{\lambda}{\ell-1}, & \text{otherwise} \end{cases}, \quad 0 \le \lambda \le \frac{\ell-1}{\ell}.$$

- *Li & Racine kernel (Ouyang et al. 2006):*

$$K_u(x=x';\lambda) = \begin{cases} 1, & \text{if } x = x' \\ \lambda, & \text{otherwise} \end{cases}, \quad 0 \le \lambda \le 1.$$

For ordinal (ordered categorical) variables:

- *Li & Racine kernel (Li and Racine 2003):*

$$K_o(x=x';\nu) = \begin{cases} 1, & \text{if } x = x' \\ \nu^{|x-x'|}, & \text{otherwise} \end{cases}, \quad 0 \le \nu \le 1.$$

- *Wang & van Ryzin kernel (Wang and Van Ryzin 1981):*

$$K_o(x=x';\nu) = \begin{cases} 1-\nu, & \text{if } x = x' \\ \frac{1-\nu}{2}\nu^{|x-x'|}, & \text{otherwise} \end{cases}, \quad 0 \le \nu \le 1.$$

The bandwidth parameters s, λ, and ν control the smoothness of the density estimate and are automatically determined by the algorithm if not provided by the user using the approach in Costa et al. (2025). ℓ is the number of levels of the categorical variable. For ordinal variables, the lambda parameter of the function is used to define ν.

Value

An object of class "gibclust" representing the final clustering result. The returned object is a list with the following components:

Cluster	An integer vector giving the cluster assignments for each data point.
Entropy	A numeric value representing the entropy of the cluster assignments at convergence.
CondEntropy	A numeric value representing the conditional entropy of cluster assignment, given the observation weights $H(T \mid X)$.
MutualInfo	A numeric value representing the mutual information, $I(Y;T)$, between the original labels (Y) and the cluster assignments (T).
InfoXT	A numeric value representing the mutual information, $I(X;T)$, between the original observations weights (X) and the cluster assignments (T).
beta	A numeric vector of the final beta values used in the iterative procedure.
alpha	A numeric value of the strength of conditional entropy used, controlling fuzziness of the solution. This is by default equal to 0 for DIBmix.
s	A numeric vector of bandwidth parameters used for the continuous variables. A value of -1 is returned if all variables are categorical.
lambda	A numeric vector of bandwidth parameters used for the categorical variables. A value of -1 is returned if all variables are continuous.
call	The matched call.
ncl	Number of clusters.
n	Number of observations.
iters	Number of iterations used to obtain the returned solution.
converged	Logical indicating whether convergence was reached before maxiter.
conv_tol	Numeric convergence tolerance.
contcols	Indices of continuous columns in X.
catcols	Indices of categorical columns in X.
kernels	List with names of kernels used for continuous, nominal, and ordinal features.

Objects of class "gibclust" support the following methods:

- print.gibclust: Display a concise description of the fitted clustering.
- summary.gibclust: Show detailed information including cluster sizes, information-theoretic metrics, hyperparameters, and convergence details.
- plot.gibclust: Produce diagnostic plots:
  - type = "sizes": barplot of cluster sizes or hardened sizes (IB/GIB).
  - type = "info": barplot of entropy, conditional entropy, and mutual information.
  - type = "beta": trajectory of $\log \beta$ over iterations (only available for hard clustering outputs obtained using DIBmix).

Author(s)

Efthymios Costa, Ioanna Papatsouma, Angelos Markos

References

Strouse DJ, Schwab DJ (2019). "The information bottleneck and geometric clustering." *Neural Computation*, **31**(3), 596–612.

Aitchison J, Aitken CG (1976). "Multivariate binary discrimination by the kernel method." *Biometrika*, **63**(3), 413–420.

Li Q, Racine J (2003). "Nonparametric estimation of distributions with categorical and continuous data." *Journal of Multivariate Analysis*, **86**(2), 266–292.

Silverman BW (1998). *Density Estimation for Statistics and Data Analysis (1st Ed.)*. Routledge.

Ouyang D, Li Q, Racine J (2006). "Cross-validation and the estimation of probability distributions with categorical data." *Journal of Nonparametric Statistics*, **18**(1), 69–100.

Wang M, Van Ryzin J (1981). "A class of smooth estimators for discrete distributions." *Biometrika*, **68**(1), 301–309.

Epanechnikov VA (1969). "Non-parametric estimation of a multivariate probability density." *Theory of Probability & Its Applications*, **14**(1), 153–158.

Examples

```
# Example dataset with categorical, ordinal, and continuous variables
set.seed(123)
data_mix <- data.frame(
  cat_var = factor(sample(letters[1:3], 100, replace = TRUE)),    # Nominal categorical variable
  ord_var = factor(sample(c("low", "medium", "high"), 100, replace = TRUE),
                   levels = c("low", "medium", "high"),
                   ordered = TRUE),                                # Ordinal variable
  cont_var1 = rnorm(100),                                         # Continuous variable 1
  cont_var2 = runif(100)                                          # Continuous variable 2
)

# Perform Mixed-Type Fuzzy Clustering
result_mix <- IBmix(X = data_mix, ncl = 3, beta = 2, nstart = 10)

# Print clustering results
print(result_mix$Cluster)       # Cluster membership matrix
print(result_mix$InfoXT)        # Mutual information between X and T
print(result_mix$MutualInfo)    # Mutual information between Y and T

# Summary of output
summary(result_mix)

# Simulated categorical data example
set.seed(123)
data_cat <- data.frame(
  Var1 = as.factor(sample(letters[1:3], 200, replace = TRUE)),  # Nominal variable
  Var2 = as.factor(sample(letters[4:6], 200, replace = TRUE)),  # Nominal variable
  Var3 = factor(sample(c("low", "medium", "high"), 200, replace = TRUE),
                levels = c("low", "medium", "high"), ordered = TRUE)  # Ordinal variable
)

# Perform fuzzy clustering on categorical data with standard IB
result_cat <- IBmix(X = data_cat, ncl = 3, beta = 15, lambda = -1, nstart = 10, maxiter = 200)

# Print clustering results
print(result_cat$Cluster)       # Cluster membership matrix
```

```
print(result_cat$InfoXT)        # Mutual information between X and T
print(result_cat$MutualInfo)    # Mutual information between Y and T

plot(result_cat, type = "sizes") # Bar plot of cluster sizes (hardened assignments)
plot(result_cat, type = "info")  # Information-theoretic quantities plot

# Simulated continuous data example
set.seed(123)
# Continuous data with 200 observations, 5 features
data_cont <- as.data.frame(matrix(rnorm(1000), ncol = 5))

# Perform fuzzy clustering on continuous data with standard IB
result_cont <- IBmix(X = data_cont, ncl = 3, beta = 50, s = -1, nstart = 10)

# Print clustering results
print(result_cont$Cluster)       # Cluster membership matrix
print(result_cont$InfoXT)        # Mutual information between X and T
print(result_cont$MutualInfo)    # Mutual information between Y and T
```

Index