Package 'UnitMix'

March 16, 2026

Title Detecting Measurement-Unit Errors via Gaussian Mixture Models

Version 0.0.1

Description Tools to detect and correct measurement-unit errors in multivariate numeric data using model-based clustering. Gaussian mixture models with user-defined translation vectors identify clusters of records that differ in scale or unit. Core functionality includes cluster assignment via the EM algorithm, error correction based on posterior probabilities and pairwise scatterplot visualizations. For more details see Di Zio, Guarnera and Luzi (2005) <https://www150.statcan.gc.ca/n1/en/pub/12-001-x/2005001/article/8087-eng.pdf>.

License GPL-3

Encoding UTF-8

Imports mvtnorm

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0)

NeedsCompilation no

Author Cristina Faricelli [aut, cre],
Renato Magistro [aut]

Maintainer Cristina Faricelli <cristina.faricelli@istat.it>

Repository CRAN

Date/Publication 2026-03-16 16:30:14 UTC

Contents

assign.cluster *Assign Clusters Using Custom Translations and EM Algorithm*

Description

Performs clustering on a numeric dataset by shifting the mean vector according to user-defined translation vectors (shifts), and fitting a Gaussian Mixture Model (GMM) using the Expectation-Maximization (EM) algorithm.

Usage

```
assign.cluster(
  db,
  var,
  errorPatterns,
  prob_thrsh = 0,
  entropy_thrsh = 1,
  max_iter = 500,
  tol = 1e-06,
  ID = NULL
)
```

Arguments

`db`	A numeric data frame or matrix containing the variables to be clustered. All columns must be numeric and positive.
`var`	A character vector indicating the names of the numeric variables in ‘db‘ to be used for clustering. All specified variables must be numeric and positive.
`errorPatterns`	A list of positive numeric vectors, each representing a shift from the global mean vector to define a cluster centroid. Each vector must have the same length as the number of columns in `data` and each element of the vector represent the erroneous scale factor for each examined variable (see example). Avoid patterns where all the examinated variables are considered erroneus.
`prob_thrsh`	Numeric. Minimum required posterior probability for an observation to be assigned to a cluster (default is 0). If no cluster exceeds this value, the observation is left unassigned.
`entropy_thrsh`	Numeric. Maximum normalized Shannon entropy allowed for a cluster assignment (default is 1). Entropy is calculated on each observation's posterior distribution and normalized by `log(k)` where `k` is the number of clusters. Lower values indicate more confident assignments.
`max_iter`	Integer. Maximum number of iterations for the EM algorithm. Default is 500.
`tol`	Numeric. Convergence threshold for the log-likelihood. EM stops when the change in log-likelihood between iterations is below this value.

`ID`	(Optional) A character string specifying the name of the identifier column in ‘db‘. If provided and the column exists in ‘db‘, its values will be included in the output as the first column, labeled ‘ID‘. This allows traceability of each corrected record. If not provided or if the specified column does not exist, a default sequence from 1 to the number of rows will be used as identifiers.

Details

Each observation is softly assigned to all clusters through posterior probabilities. Final cluster assignment is made only if the maximum posterior probability exceeds a user-defined threshold (‘prob_thrsh‘) and the normalized Shannon entropy of the probability vector is below another threshold (‘entropy_thrsh‘). This allows uncertainty-aware clustering. Entropy is low, close to zero, when the posterior probability is concentrated on a single cluster. The maximum entropy value is 1. Values below 0.1 are generally considered acceptable. The default value for ‘prob_thrsh‘ is 0, while for ‘entropy_thrsh‘ it is 1, which means that clusters are returned exactly as produced by the algorithm, without any filtering.

This method assumes multivariate log-normality and equal covariance across clusters. It implements robust assignment logic by rejecting uncertain observations based on entropy and minimum probability thresholds. Observations with no dominant cluster (high entropy or low posterior probability) are marked with cluster 0. It is the **user's responsibility** to ensure that input variables are:

- Numeric
- Strictly positive (since log transformation is applied)
- Appropriately scaled (e.g., unit conversion issues should be captured by ‘errorPatterns‘)

If these conditions are not met, the function will return informative error messages and halt execution.

Input data is not preprocessed automatically: **preprocessing must be handled by the user**.

Value

A list containing:

data A data frame including:

- The observation ID (from the ‘ID‘ column if provided, otherwise a sequence number),
- The original variables used for clustering,
- The final cluster assignment for each observation (with unassigned cases labeled as `0`),
- The normalized Shannon entropy of the posterior distribution,
- The posterior probabilities for each cluster.

postprob A data frame containing posterior probabilities for each cluster. One row per observation, one column per cluster.

sigma The estimated shared covariance matrix used in the Gaussian mixture model.

mu A matrix of cluster centroids, obtained by translating the overall mean vector with each specified error pattern.

iterations The number of EM iterations executed before convergence.

References

Di Zio, M., Guarnera, U., & Luzi, O. (2005). Editing systematic unity measure errors through mixture modelling. *Survey Methodology*, 31(1), 53–63. https://www.istat.it/it/files/2014/05/Survey-Methodology-311-53-63.pdf

Examples

```
# Simulated dataset with unit-of-measurement error on 3 values
data <- data.frame(
  x1 = c(rlnorm(50, meanlog = 4, sdlog = 0.3), rlnorm(3, meanlog = 4, sdlog = 0.3) * 1000),
  x2 = rlnorm(53, meanlog = 6, sdlog = 0.3)
)

 var <- names(data)

# Define two errorPatterns: no shift, and a ×1000 shift in x1 only
errorPatterns <- list(
  c(0, 0),           # Cluster 1: correct units
  c(1000, 0)         # Cluster 2: scale error ×1000
)

# Run the clustering algorithm
ac <- assign.cluster(
  db = data,
  var= var,
  errorPatterns = errorPatterns
)

# Examine the results
table(ac$data$cluster)
```

`cluster.plot` *Visualize Cluster Assignments with Pairwise Scatterplots*

Description

This function dynamically generates a set of scatter plots for all pairs of variables, log-transformed, in the provided dataset, enabling detailed visualisation of cluster assignments. The points are coloured according to the cluster to which each observation is assigned, based on posterior probabilities. If the number of variables is large, the number of plots is automatically adjusted to maintain readability, creating multiple graphs. Observations are only assigned to a cluster if the a posteriori probability for that cluster is above a specified threshold (default = 0.6).

Usage

```
cluster.plot(
  db,
  var,
  postprob = 1,
  errorPatterns = ncol(data),
  threshold = 0.6
)
```

Arguments

`db`	A numeric data frame or matrix containing the variables to plot.
`var`	A character vector indicating the names of the ckusters numeric variables in ‘db. All specified variables must be numeric and positive.
`postprob`	A matrix or dataframe of posterior probabilities where each column represents a cluster. It is typically obtained from the function `assign.cluster()`. If set to 1 (default), it is internally set to a matrix assigning all observations to a single cluster. They must have the same row number of ’db’
`errorPatterns`	A list of positive numeric vectors that have been identified as being associated with potential unit errors which will be highlighted in the graphs. If not provided, no special errorPatterns are highlighted (default = ncol(data), used internally)
`threshold`	Minimum probability threshold for cluster assignment (default = 0.6).

Details

It is the **user’s responsibility** to ensure that input variables are:

- Numeric
- Strictly positive (since log transformation is applied)
- Appropriately scaled (e.g., unit conversion issues should be captured by ‘errorPatterns‘)

If these conditions are not met, the function will return informative error messages and halt execution.

Input data is not preprocessed automatically: **preprocessing must be handled by the user**.

Value

No value returned. The function produces side-effect graphs, which are useful for visually inspecting the separation between clusters and the reliability of assignments.

See Also

`assign.cluster`

Examples

```
data <- data.frame(
 x1 = c(rlnorm(50, meanlog = 4, sdlog = 0.3), rlnorm(10, meanlog = 4, sdlog = 0.3) * 1000),
 x2 = rlnorm(60, meanlog = 6, sdlog = 0.3)
)

var <- names(data)
```

```
# Define two errorPatterns: no shift, and a ×1000 shift in x1 only
errorPatterns <- list(
  c(0, 0),           # Cluster 1: correct units
  c(1000, 0)         # Cluster 2: scale error ×1000
)

results <- assign.cluster(
  db = data,
  var= var,
  errorPatterns = errorPatterns
)

cluster.plot(data, var, postprob = results$postprob, errorPatterns, threshold = 0.6)
```

`refine.cluster` *Refine Cluster Assignments Using Mahalanobis Distance Compatibility*

Description

Post-processing function for the output of `assign.cluster`. It evaluates the compatibility of each record with its assigned cluster using Mahalanobis distance on log-transformed data. Records incompatible with their cluster (distance > chi-square cutoff) or belonging to unreliable clusters (low compatibility proportion or small size) are reassigned to cluster 0 (unassigned). This enhances assignment reliability by discarding outliers and unstable groups.

Usage

```
refine.cluster(
  ac,
  vars,
  compat_level = 0.99,
  min_good_prop = 0.9,
  min_cluster_size = 5
)
```

Arguments

`ac`	List. Output object returned by `assign.cluster`.
`vars`	Character vector. Names of the variables used in `assign.cluster`. Must match exactly those in `ac`.
`compat_level`	Numeric. Confidence level for the chi-square cutoff used to assess individual compatibility (default: 0.99).
`min_good_prop`	Numeric. Minimum proportion of compatible records required to consider a cluster reliable (default: 0.90).
`min_cluster_size`	Integer. Minimum cluster size required to consider a cluster valid (default: 5).

Details

This method assumes multivariate log-normality and uses the shared covariance matrix from `assign.cluster`. It is the user's responsibility to ensure input variables are numeric and strictly positive (log transformation applied). Cluster-level diagnostics exclude cluster 0. Good clusters must meet both size and compatibility thresholds. For visualization pipe to `cluster.plot`.

Value

A list containing:

`data`	A data frame with the original results and additional columns: `mahal_d2` (Mahalanobis distance squared), `compatible_Mah` (logical indicator of compatibility), and `cluster_refined` (refined cluster assignments).
`postprob`	Posterior probability matrix with rows set to zero for discarded records.
`cluster_summary`	Diagnostic table reporting, for each cluster: `n_total`, `n_compatible`, `n_incompatible`, and `prop_compatible`.
`params`	List of parameters used in the refinement process.

See Also

`assign.cluster`, `cluster.plot`

Examples

```
# Simulated dataset with:
# - 50 correct observations
# - 3 unit-of-measurement errors (x1 ×1000)
# - 2 structural outliers

set.seed(123)

data <- data.frame(
  x1 = c(
    rlnorm(50, meanlog = 4, sdlog = 0.3),        # correct data
    rlnorm(3,  meanlog = 4, sdlog = 0.3) * 1000, # scale error
    rlnorm(2,  meanlog = 8, sdlog = 0.1)         # outliers
  ),
  x2 = c(
    rlnorm(50, meanlog = 6, sdlog = 0.3),
    rlnorm(3,  meanlog = 6, sdlog = 0.3),
    rlnorm(2,  meanlog = 9, sdlog = 0.1)         # outliers
  )
)

var <- names(data)

# Define two errorPatterns: no shift, and ×1000 shift in x1 only
errorPatterns <- list(
  c(0, 0),       # Cluster 1: correct units
  c(1000, 0)     # Cluster 2: scale error ×1000
)

# Run the clustering algorithm
ac <- assign.cluster(
  db = data,
  var = var,
  errorPatterns = errorPatterns
)

# Examine initial cluster assignments
table(ac$data$cluster)

# Run the refine cluster algorithm
rc <- refine.cluster(ac, var, min_cluster_size=1)

# Re-examine cluster assignments
table(rc$data$cluster_refined)
```

Index