Package ‘SIMPLICA’

September 3, 2025

Title Biclustering via Simplivariate Component Analysis

Version 1.0.0

Description Identifies constant, additive, multiplicative, and user-defined simplivariate components in numeric data matrices using a genetic algorithm. Supports flexible pattern definitions and provides visualization for general biclustering applications across diverse domains. The method builds on simplivariate models as introduced in Hageman et al. (2008) <doi:10.1371/journal.pone.0003259> and is related to biclustering frameworks as reviewed by Madeira and Oliveira (2004) <doi:10.1109/TCBB.2004.2>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 2.10)

LazyData true

LazyDataCompression xz

Imports GA, ggplot2, ggsci

URL `https://github.com/joshageman/SIMPLICA`

BugReports `https://github.com/joshageman/SIMPLICA/issues`

NeedsCompilation no

Author Jos Hageman [aut, cre]

Maintainer Jos Hageman <`jos.hageman@wur.nl`>

Repository CRAN

Date/Publication 2025-09-03 08:00:18 UTC

Contents

`additiveMatrix` *Generate an additive approximation of a data matrix*

Description

Constructs a matrix with an additive structure based on the row means, column means, and the grand mean of the original matrix.

Usage

```
additiveMatrix(mat)
```

Arguments

`mat`	A numeric matrix or data frame with numeric entries.

Details

The result is an approximation A of the original matrix, where:

$$A[i,j] = rowMean[i] + colMean[j] - grandMean$$

This model captures main additive effects of rows and columns, commonly used in exploratory data analysis or baseline modeling.

Value

A numeric matrix of the same dimension as `mat`, approximating it using an additive model.

Examples

```
m <- matrix(c(1, 2, 3, 4), nrow = 2)
additiveMatrix(m)
```

`additiveMatrixFitter` *Additive Pattern Fitter*

Description

Fits an additive pattern to matrix data using training data specified by a mask. The pattern is based on row and column main effects, creating an additive model where each cell value is the sum of a row effect and column effect minus the overall mean.

Usage

```
additiveMatrixFitter(mat, trainMask)
```

Arguments

`mat`	Numeric matrix containing the data to fit
`trainMask`	Logical matrix of same dimensions as mat, indicating which cells to use for training

Details

The function creates an additive pattern by:

- Masking non-training cells as NA to compute statistics only on training data
- Computing overall mean of training data
- Computing row means and column means from training data
- Replacing non-finite means with overall mean as fallback
- Creating additive pattern as outer sum of row and column effects minus overall mean

Value

Numeric matrix of same dimensions as mat containing the fitted additive pattern

componentCVPatterns *Cross-Validation of Simplivariate Component Patterns*

Description

Performs pure cross-validation over specified patterns with mandatory fitters. This function evaluates different pattern fitting models using cross-validation to determine the best model for a given data subset.

Usage

```
componentCVPatterns(
  df,
  rows,
  cols,
  patternFunctions,
  patternFitters,
  preferenceOrder = names(patternFunctions),
  nRepeats = 40,
  testFraction = 0.2,
  minCellsForModels = 25,
  parsimonyMargin = 0.05,
  requireFitters = TRUE,
  verbose = FALSE
)
```

Arguments

`df`	A matrix or data frame containing the data
`rows`	Row indices to subset from df
`cols`	Column indices to subset from df
`patternFunctions`	A named list of pattern functions to evaluate
`patternFitters`	A named list of fitter functions corresponding to each pattern
`preferenceOrder`	Character vector specifying the preference order of patterns (default: names of patternFunctions)
`nRepeats`	Integer, number of cross-validation repeats (default: 40)
`testFraction`	Numeric, fraction of data to use for testing in each CV fold (default: 0.2)
`minCellsForModels`	Integer, minimum number of cells required for reliable CV (default: 25)
`parsimonyMargin`	Numeric, margin for parsimony selection as fraction (default: 0.05)
`requireFitters`	Logical, whether to require fitters for all patterns (default: TRUE)
`verbose`	Logical, whether to print progress messages (default: FALSE)

Value

A list containing:

`decision`	Character, the selected best pattern name
`reason`	Character, explanation of the selection reasoning
`cv`	Data frame with CV summary statistics for each model
`repeats`	Data frame with detailed results from each CV repeat
`meta`	List with metadata about the CV procedure

`defaultPatternFitters` *Default Pattern Fitters for SIMPLICA*

Description

Returns a list of default pattern fitting functions used in SIMPLICA. These fitters estimate different types of patterns (constant, additive, multiplicative) from matrix data using specified training cells.

Usage

```
defaultPatternFitters()
```

Details

Each fitter function takes two arguments:

- mat: Numeric matrix containing the data to fit
- trainMask: Logical matrix indicating which cells to use for training

All fitters return a fitted matrix of the same dimensions as the input.

Value

A named list containing pattern fitting functions:

- constant: Fits a constant value (mean of training data)
- additive: Fits an additive pattern using additiveMatrixFitter
- multiplicative: Fits a multiplicative pattern using multiplicativeMatrixFitter

Examples

```
# Retrieve default pattern fitters
fitters <- defaultPatternFitters()

# Add a custom diagonal pattern fitter
diagonalFitter <- function(mat, trainMask) {
  # Extract diagonal values from training data
  minDim <- min(nrow(mat), ncol(mat))
  diagIndices <- cbind(1:minDim, 1:minDim)
  # Only use diagonal elements that are in the training mask
  validDiag <- trainMask[diagIndices]
  if (any(validDiag)) {
    diagVal <- mean(mat[diagIndices][validDiag])
  } else {
    diagVal <- mean(mat[trainMask])  # fallback to overall mean
  }
  matrix(diagVal, nrow = nrow(mat), ncol = ncol(mat))
}

# Extend the list with your own pattern
fitters$diagonal <- diagonalFitter
```

`defaultPatternFunctions` *Default pattern generators for SIMPLICA*

Description

Returns a named list of default matrix approximation functions used to score component patterns. Each function must take a numeric matrix (i.e., a component: subset of rows and columns) and return a matrix of the same dimensions that approximates the original matrix according to a specific structural pattern (e.g., constant, additive, etc.).

Usage

```
defaultPatternFunctions()
```

Details

This list can be passed to `fitness2()` via the `patternFunctions` argument. Users can extend or override the default patterns by modifying the returned list.

Requirements for pattern functions:

Custom pattern functions must:

- Take a numeric matrix as input.
- Return a numeric matrix of the same dimensions.
- Be compatible with `sum(abs(...))` and `sum((...)^2)` operations for fitness scoring.

Value

A named list of functions, each representing a matrix approximation method.

Examples

```
# Retrieve default pattern functions
patterns <- defaultPatternFunctions()

# Add a custom pattern based on diagonal structure
diagonalPattern <- function(m) {
  diagVal <- mean(diag(as.matrix(m)))
  matrix(diagVal, nrow = nrow(m), ncol = ncol(m))
}

# Extend the list with your own pattern
patterns$diagonal <- diagonalPattern
```

`fitness` *Fitness function with automatic pattern selection per Simplivariate Component*

Description

Fitness function with automatic pattern selection per Simplivariate Component

Usage

```
fitness(
  string,
  df,
  dfMean,
  penalty,
  patternFunctions = defaultPatternFunctions(),
  returnPatterns = FALSE,
  ...
)
```

Arguments

`string` Vector with length nrow(df) + ncol(df): component labels for rows and columns (in this order).

`df` Numeric matrix: full data.

`dfMean` Scalar: global mean of df.

`penalty` Named vector with penalty weights per pattern.

`patternFunctions` Named list of functions returning pattern-based approximations.

`returnPatterns` Logical: if TRUE, also returns chosen pattern per component.

`...` Additional arguments passed to the GA functions

Value

Either total fitness (numeric), or list(fitness, componentPatterns) if returnPatterns = TRUE.

`fitnessForOneComponent` *Compute best pattern-based fitness for a single Simplivariate Component*

Description

Compute best pattern-based fitness for a single Simplivariate Component

Usage

```
fitnessForOneComponent(mat, dfMean, patternFunctions, penalty)
```

Arguments

`mat` A numeric matrix (the component)

`dfMean` Overall mean of the full data matrix

`patternFunctions` Named list of functions for structure types

`penalty` Named numeric vector of penalties per pattern type

Value

Numeric fitness value (higher is better)

Examples

```
m <- matrix(rnorm(100, mean = 10), nrow = 10)
f <- fitnessForOneComponent(m, mean(m), defaultPatternFunctions(),
              c(constant = 0, additive = 1.0, multiplicative = 0))
```

gaintegerMutation *Integer mutation operator for genetic algorithms*

Description

Applies mutation to a selected parent vector by replacing each gene with a random value (within bounds) with a given mutation probability. Used in integer-encoded GAs.

Usage

```
gaintegerMutation(object, parent, ...)
```

Arguments

`object`	A GA object containing at least the slots `@population`, `@upper`, `@lower`, and `@pmutation`.
`parent`	An integer index indicating which individual in the population to mutate.
`...`	Further arguments (unused, included for compatibility).

Value

A numeric vector representing the mutated individual.

gaintegerOnePointCrossover *One-point crossover operator for integer-encoded genetic algorithms*

Description

Performs one-point crossover on two parent individuals. A single crossover point is selected, and all genes before (and including) that point are exchanged between the parents.

Usage

```
gaintegerOnePointCrossover(object, parents, ...)
```

Arguments

`object`	A GA object with a `@population` slot (a matrix).
`parents`	A 2-row matrix of values indexing the parents from the current population.
`...`	Further arguments (unused, included for compatibility).

Value

A list with two elements:

children A 2-row matrix of the resulting offspring.

fitness A numeric vector of `NA` values to be replaced by fitness evaluation.

`gaintegerPopulationFactory` *GA Integer Population Factory*

Description

Creates a factory function for generating initial populations for genetic algorithms with integer chromosomes, where a specified fraction of variables are set to zero.

Usage

```
gaintegerPopulationFactory(zeroFraction, verbose = FALSE)
```

Arguments

`zeroFraction`	Numeric value between 0 and 1 specifying the fraction of variables to set to zero in each individual
`verbose`	Logical indicating whether to print information about zero fraction

Value

A function that takes a GA object and returns an initial population matrix

---

`globals` *Global Variables Declaration*

Description

This file declares global variables used in the SIMPLICA package to avoid R CMD check notes about "no visible binding for global variable". These variables are typically used within ggplot2 aesthetics and data manipulation functions where they refer to column names in data frames created during execution.

`multiplicativeMatrix` *Generate an multiplicative approximation of a data matrix*

Description

Approximates a data matrix using a low-rank multiplicative model based on a fixed outer product of centered row and column effects.

Usage

```
multiplicativeMatrix(mat)
```

Arguments

`mat`	A numeric matrix with values to approximate.

Details

The model assumes:

$$M[i,j] = mu + a * rowEffect[i] * colEffect[j]$$

where `mu` is the overall mean of the input matrix, and `rowEffect` and `colEffect` are centered integer sequences.

Value

A numeric matrix of the same size as `mat`, containing the fitted values.

`multiplicativeMatrixFitter` *Multiplicative Pattern Fitter*

Description

Fits a multiplicative pattern to matrix data using training data specified by a mask. The pattern is based on the outer product of centered row and column effects, creating a bilinear surface that can capture multiplicative interactions between rows and columns.

Usage

```
multiplicativeMatrixFitter(mat, trainMask)
```

Arguments

`mat`	Numeric matrix containing the data to fit
`trainMask`	Logical matrix of same dimensions as mat, indicating which cells to use for training

Details

The function creates a multiplicative pattern by:

- Computing the mean of training data as baseline
- Creating centered row effects (0 to nRows-1, mean-centered)
- Creating centered column effects (0 to nCols-1, mean-centered)
- Taking outer product to form bilinear pattern
- Fitting scaling coefficient using least squares on training data
- Returning baseline plus scaled pattern

Value

Numeric matrix of same dimensions as mat containing the fitted multiplicative pattern

---

`plotComponentResult` *Plot non-contiguous simplivariate components by pattern type, with optional reordering*

Description

Visualizes GA-detected simplivariate components on the original matrix as outlined cells, colored by pattern type.

Usage

```
plotComponentResult(
  df,
  string,
  componentPatterns,
  componentScores,
  scoreCutoff = 0,
  showAxisLabels = TRUE,
  showComponentLabels = TRUE,
  title = "Detected Components",
  rearrange = FALSE,
  grayscale = TRUE
)
```

Arguments

`df`	Original data matrix
`string`	Best GA string (vector of length nrow(df) + ncol(df); rows first, then cols)
`componentPatterns`	Vector of component types (from fitness(..., returnPatterns = TRUE))

`componentScores`
: Vector of fitness scores per component

`scoreCutoff` Minimum score a component must have to be shown (default: 0 = show all)

`showAxisLabels` Logical: show axis tick labels (default: TRUE)

`showComponentLabels`
: Logical: show component labels inside clusters (default: TRUE)

`title` Title for the plot (default: "Detected Components")

`rearrange` Logical: reorder rows and columns to group components (default: FALSE)

`grayscale` Logical: use grayscale for heatmap background (default: TRUE)

Value

ggplot object

`print.summaryComponents` *Print method for summaryComponents*

Description

Print method for summaryComponents

Usage

```
## S3 method for class 'summaryComponents'
print(x, showDetails = FALSE, maxLinesDetails = 200L, ...)
```

Arguments

`x` An object produced by `summaryComponents()`.

`showDetails` Logical. If TRUE, also print row/column indices per component.

`maxLinesDetails`
: Integer. Max number of indices to print per dimension (rows/cols) before truncation (default 200).

`...` Further arguments passed to or from other methods (not used here).

Value

The input object `x`, invisibly. Called for its side effect of printing a formatted component summary to the console.

`SCAMonitorFactory` *Simplivariate Component Analysis monitoring function factory for GA progress*

Description

Creates a monitoring function that prints the current generation and the best fitness score to the console at specified intervals. Intended for use as a `monitor` function in GA runs.

Usage

```
SCAMonitorFactory(interval = 100)
```

Arguments

`interval`	An integer specifying the interval for printing progress updates. Default is 100 (prints every 100 generations).

Value

A monitoring function that can be used with GA. The returned function takes a GA object and prints progress information at the specified interval.

Examples

```
# Create monitor that prints every 100 generations (default)
monitor <- SCAMonitorFactory()
# ga(..., monitor = monitor)

# Create monitor that prints every 50 generations
monitor <- SCAMonitorFactory(50)
# ga(..., monitor = monitor)
```

`simplica` *SIMPLICA: Simultaneous Identification of Simplivariate Components*

Description

Implements the SIMPLICA algorithm to identify Simplivariate Components in data matrices using a genetic algorithm. These components are related to clusters or biclusters, but defined here in terms of specific structural patterns (constant, additive, multiplicative, or user-defined).

Usage

```
simplica(
  df,
  maxIter = 2000,
  popSize = 300,
  pCrossover = 0.6,
  pMutation = 0.03,
  zeroFraction = 0.9,
  elitism = 100,
  numSimComp = 5,
  verbose = FALSE,
  mySeeds = 1:5,
  interval = 100,
  penalty = c(constant = 0, additive = 1, multiplicative = 0),
  patternFunctions = defaultPatternFunctions(),
  doSimplicaCV = TRUE,
  cvControl = NULL
)
```

Arguments

`df` A numeric data matrix to analyze

`maxIter` Maximum number of generations for the genetic algorithm (default: 2000)

`popSize` Population size for the genetic algorithm (default: 300)

`pCrossover` Crossover probability for genetic algorithm (default: 0.6)

`pMutation` Mutation probability for genetic algorithm (default: 0.03)

`zeroFraction` Fraction of population initialized with zeros (default: 0.9)

`elitism` Number of best individuals preserved between generations (default: 100)

`numSimComp` Number of Simplivariate Components simultaneously optimized (default: 5)

`verbose` Logical, whether to print SIMPLICA progress information (default: FALSE)

`mySeeds` Vector of random seeds for replicate runs (default: 1:5)

`interval` Interval for monitoring GA progress (default: 100)

`penalty` Named vector of penalty values for each pattern type (default: c(constant = 0, additive = 1, multiplicative = 0))

`patternFunctions` List of pattern functions used for fitness evaluation (default: defaultPatternFunctions())

`doSimplicaCV` Logical, run cross-validated relabeling with simplicaCV() after GA (default: TRUE)

`cvControl` Optional list to tune simplicaCV; fields passed to simplicaCV via do.call. Defaults if omitted:

- patternFitters = defaultPatternFitters()
- preferenceOrder = names(patternFunctions)

- nRepeats = 40
- testFraction = 0.2
- minCellsForModels = 25
- parsimonyMargin = 0.05
- requireFitters = TRUE
- updateObject = TRUE
- verbose = verbose

Value

A list with:

- best: simplica object (includes original GA result; if doSimplicaCV=TRUE, also componentPatternsUpdated and componentAudit)
- raw: list of "ga" objects (one per seed, from the GA package)

References

Hageman, J. A., Wehrens, R., & Buydens, L. M. C. (2008). "Simplivariate Models: Ideas and First Examples." PLoS ONE, 3(9), e3259. doi:10.1371/journal.pone.0003259

Madeira, S. C., & Oliveira, A. L. (2004). "Biclustering Algorithms for Biological Data Analysis: A Survey." IEEE/ACM Transactions on Computational Biology and Bioinformatics, 1(1), 24–45. doi:10.1109/TCBB.2004.2

Examples

```
data("simplicaToy")
# Minimal run just to demonstrate function usage, run with default GA parameters
fit <- simplica(df = simplicaToy$data,
                maxIter = 200,
                popSize = 50,
                mySeeds = 1,
                elitism = 1,
                verbose = TRUE)
plotComponentResult(df = simplicaToy$data,
                    string            = fit$best$string,
                    componentPatterns = fit$best$componentPatternsUpdated,
                    componentScores   = fit$best$componentScores,
                    showAxisLabels    = FALSE,
                    title             = "SIMPLICA on simplicaToy",
                    scoreCutoff       = 25000)
```

simplicaCV *Test Simplivariate Components with Cross-Validation Pattern Selection*

Description

This function performs cross-validation-based pattern testing for Simplivariate Components in a SIMPLICA object. It evaluates different pattern functions using cross-validation and selects the best performing pattern for each component. Fitters are required for all patterns with no fallback options.

Usage

```
simplicaCV(
  foundObject,
  df,
  patternFunctions = defaultPatternFunctions(),
  patternFitters = defaultPatternFitters(),
  preferenceOrder = names(patternFunctions),
  nRepeats = 40,
  testFraction = 0.2,
  minCellsForModels = 25,
  parsimonyMargin = 0.05,
  requireFitters = TRUE,
  updateObject = TRUE,
  verbose = FALSE,
  ignoreNaComponents = TRUE
)
```

Arguments

`foundObject` A simplica object containing Simplivariate Components

`df` Data frame or matrix with the original data

`patternFunctions`
List of pattern functions to evaluate (default: defaultPatternFunctions())

`patternFitters` List of pattern fitting functions (default: defaultPatternFitters())

`preferenceOrder`
Character vector specifying preference order for pattern selection (default: names(patternFunctions))

`nRepeats` Integer, number of cross-validation repeats (default: 40)

`testFraction` Numeric, fraction of data to use for testing (default: 0.2)

`minCellsForModels`
Integer, minimum number of cells required for model fitting (default: 25)

`parsimonyMargin`
Numeric, margin for parsimony-based model selection (default: 0.05)

`requireFitters` Logical, whether fitters are required for all patterns (default: TRUE) `updateObject` Logical, whether to update and return the input object (default: TRUE)

`verbose` Logical, whether to print progress messages (default: FALSE)

`ignoreNaComponents`
Logical, whether to skip components with NA patterns (default: TRUE)

Details

The function performs the following steps:

- Validates the input simplica object and data dimensions
- Checks that all pattern functions have corresponding fitters
- For each simplivariate component, performs cross-validation pattern evaluation
- Selects the best performing pattern based on RMSE and parsimony
- Updates component patterns and provides detailed test information

Value

If `updateObject = TRUE`, returns the input `simplica` object with two new fields:

`componentPatternsUpdated` Character vector with the selected pattern per component after cross-validation. If a component is skipped or empty, the entry is `NA`.

`componentAudit` Data frame containing detailed cross-validation results for each component, with the following columns:

`componentId` Numeric ID of the component.

`originalPattern` Pattern label originally assigned.

`selectedPattern` Pattern chosen after CV-based evaluation.

`reason` Explanation of why a pattern was selected or skipped.

`nRows`, `nCols`, `nCells` Dimensions of the component.

`nRepeats`, `testFraction`, `parsimonyMargin` CV settings used.

`cvMean_<pattern>` Mean RMSE over CV folds for each tested pattern.

`cvSd_<pattern>` Standard deviation of RMSE across CV folds.

`winFrac_<pattern>` Fraction of CV repeats where the pattern was the best performer.

If `updateObject = FALSE`, returns a list with the same two elements (`componentPatternsUpdated`, `componentAudit`).

`simplicaToy` *Toy matrix with one multiplicative and one additive bicluster*

Description

A small 30×60 matrix to demonstrate SIMPLICA in a controlled setting. Contains one multiplicative and one additive simplivariate component (non-overlapping).

Usage

```
data(simplicaToy)
```

Format

A list with three elements:

data numeric matrix of dimension 30×60

trueComponents list of length 2 with `type`, `rows`, `cols`

description character string

Examples

```
data("simplicaToy")
str(simplicaToy)
image(t(simplicaToy$data))
```

`summaryComponents` *Summarize GA-found Simplivariate Components*

Description

Compute a tidy summary of simplivariate Components found by SIMPLICA. Returns a data.frame with class `"summaryComponents"` and an attribute holding row/column index lists for printing details.

Usage

```
summaryComponents(results, scoreCutoff = 0)
```

Arguments

`results`	A list or 'simplica' object with fields: `nRows`, `nCols`, `string`, `componentScores`, and either `componentPatternsUpdated` or `componentPatterns`.
`scoreCutoff`	Numeric. Minimum score to include a component (default 0).

Value

A data.frame with columns: `componentId`, `pattern`, `score`, `rows`, `cols`, `size`; class is `c("summaryComponents","data.fr` The attribute `"indices"` stores a list with per-component `rowIdx` and `colIdx`.

Index