Package 'GAPR'

May 27, 2025

Title Generalized Association Plots

Version 0.1.0

Description Provides a comprehensive framework for visualizing associations and interaction structures in matrix-formatted data using Generalized Association Plots (GAP). The package implements multiple proximity computation methods (e.g., correlation, distance metrics), ordering techniques including hierarchical clustering (HCT) and Rank-2-Ellipse (R2E) seriation, and optional flipping strategies to enhance visual symmetry. It supports a variety of covariate-based color annotations, allows flexible customization of layout and output, and is suitable for analyzing multivariate data across domains such as social sciences, genomics, and medical research. The method is based on Generalized Association Plots introduced by Chen (2002) <https://www3.stat.sinica.edu.tw/statistica/J12N1/J12N11/J12N11.html> and further extended by Wu, Tien, and Chen (2010) <doi:10.1016/j.csda.2008.09.029>.

License MIT + file LICENSE

Encoding UTF-8

Imports Rcpp, ComplexHeatmap, RColorBrewer, gridExtra, grid, dendextend, circlize, seriation, magick,

Suggests MASS

LinkingTo Rcpp

RoxygenNote 7.3.2

NeedsCompilation yes

Author Shu-Yu Lin [aut, cre]

Maintainer Shu-Yu Lin <shuyuuu89@gmail.com>

Repository CRAN

Date/Publication 2025-05-27 08:00:10 UTC

Contents

`AR` *Compute the Anti-Robinson (AR) score*

Description

Calculates the total number of Anti-Robinson violations over all triplets in the matrix using the specified ordering. This is equivalent to GAR with a full window.

Usage

```
AR(mat, order)
```

Arguments

`mat`	A symmetric numeric distance matrix.
`order`	A permutation vector specifying the new row/column order.

Value

The AR score (the total number of structural violations).

Please refer to `GAP` for complete usage examples.

`computeProximity` *Compute Proximity Matrix*

Description

This function takes a numeric matrix and computes a square proximity matrix (similarity or distance) based on a specified method.

Usage

```
computeProximity(data, proxType, side, isContainMissingValue)
```

Arguments

`data`	A numeric matrix with n rows and p columns. Each row typically represents an observation.
`proxType`	An integer specifying the type of proximity measure to use.
`side`	An integer indicating the direction for computing proximity.
`isContainMissingValue`	An integer indicating whether the input data contains missing values.

Details

proxType

Available proxType options include:

- `0`: Euclidean
- `1`: Pearson correlation
- `2`: Kendall correlation
- `3`: Spearman correlation
- `4`: Adjusted tangent correlation (atancorr)
- `5`: City-block (Manhattan) distance
- `6`: Absolute Pearson correlation
- `7`: Uncentered correlation
- `8`: Absolute uncentered correlation
- `20`: Hamman similarity (binary)
- `21`: Jaccard index (binary)
- `22`: Phi coefficient (binary)
- `23`: Rao coefficient (binary)
- `24`: Rogers-Tanimoto similarity (binary)
- `25`: Simple matching coefficient (binary)
- `26`: Sneath coefficient (binary)
- `27`: Yule's Q (binary)

Ensure the data type matches the selected method. For example, binary methods should only be used on binary (0/1) data.

side

Use `0` for row-wise proximity and `1` for column-wise proximity.

isContainMissingValue

Set to `1` if the input data includes missing values; otherwise, use `0`.

Value

A square matrix representing the proximity between rows or columns, depending on the selected side.

Examples

```
# =======================
# Example 1: Crabs dataset with distance method (Euclidean distance)
# =======================
# Step 1: Compute proximity matrix
if (requireNamespace("MASS", quietly = TRUE)) {
  df_crabs <- as.matrix(MASS::crabs[, -c(1:3)])  # Use continuous variables only
  row_prox_crabs <- computeProximity(
    data = df_crabs,
    proxType = 0,               # 0 = Euclidean distance
    side = 0,                   # 0 = row-wise proximity
    isContainMissingValue = 0
  )

  # Step 2: Obtain R2E ordering
  r2e_order_crabs <- ellipse_sort(row_prox_crabs)  # R2E ordering

  # Step 3: Apply AVG-R2E ordering
  hctree_result_crabs <- hctree_sort(
    row_prox_crabs,                  # use distance matrix directly
    externalOrder = r2e_order_crabs,  # apply r2e order
    orderType = 2,                   # 2 = Average-linkage
    flipType = 1                     # 1 = Flip based on externalOrder
  )

  avg_r2e_order_crabs <- hctree_result_crabs$order + 1

  # Inspect results
  avg_r2e_order_crabs
}

# =======================
# Example 2: Crabs dataset with distance method (Pearson correlation)
# =======================
if (requireNamespace("MASS", quietly = TRUE)) {
  df_crabs <- as.matrix(MASS::crabs[, -c(1:3)])  # Use continuous variables only
  row_prox_pearson <- computeProximity(
    data = df_crabs,
    proxType = 1,               # 1 = Pearson correlation (internally 1 - cor)
    side = 0,                   # 0 = row-wise proximity
    isContainMissingValue = 0
  )

  # Step 2: Obtain R2E ordering
  r2e_order_pearson <- ellipse_sort(row_prox_pearson)  # R2E ordering

  # Step 3: Inspect results
 dist_pearson <- as.dist(1 - row_prox_pearson) # convert correlation matrix to distance matrix
  dist_pearson_MT <- as.matrix(dist_pearson)

  hctree_result_pearson <- hctree_sort(
    dist_pearson_MT,                   # use distance matrix directly
```

```
  externalOrder = r2e_order_pearson,  # apply r2e order
  orderType = 2,                      # 2 = Average-linkage
  flipType = 1                        # 1 = Flip based on externalOrder
)

avg_r2e_order_pearson <- hctree_result_pearson$order + 1

# Inspect results
avg_r2e_order_pearson
}
```

`ellipse_sort` *Ellipse Sort*

Description

This function applies an Rank-2-Ellipse seriation method to reorder a proximity matrix.

Usage

```
ellipse_sort(data)
```

Arguments

`data`	A square numeric proximity matrix (either n × n or p × p), representing pairwise distances or similarities between items.

Value

An integer vector representing the reordered indices of the matrix rows.

Please refer to `computeProximity` for complete usage examples.

`GAP` *Generalized Association Plots (GAP)*

Description

Generates a generalized association plot for the given matrix or data frame, with optional proximity computation, ordering, flipping, coloring, and export options.

Usage

```
GAP(
  data,
  isProximityMatrix = FALSE,
  XdNum = NULL,
  XcNum = NULL,
  YdNum = NULL,
  YcNum = NULL,
  row.name = NULL,
  Xd.name = NULL,
  Xc.name = NULL,
  row.prox = NULL,
  col.prox = NULL,
  show.row.prox = TRUE,
  show.col.prox = TRUE,
  row.order = NULL,
  col.order = NULL,
  row.flip = NULL,
  col.flip = NULL,
  row.externalOrder = NULL,
  col.externalOrder = NULL,
  original.color = NULL,
  row.color = NULL,
  col.color = NULL,
  Xd.color = NULL,
  Xc.color = NULL,
  Yd.color = NULL,
  Yc.color = NULL,
  row.label.size = NULL,
  col.label.size = NULL,
  Xd.label.size = NULL,
  Xc.label.size = NULL,
  Yd.label.size = NULL,
  Yc.label.size = NULL,
  colorbar.margin = 1.5,
  border = FALSE,
  border.width = 1,
  isContainMissingValue = 0,
  MissingValue.color = "gray",
  exp.row_order = FALSE,
  exp.column_order = FALSE,
  exp.row_names = FALSE,
  exp.column_names = FALSE,
  exp.Xc = FALSE,
  exp.Yc = FALSE,
  exp.Xd = FALSE,
  exp.Yd = FALSE,
  exp.Xd_codebook = FALSE,
```

```
  exp.Yd_codebook = FALSE,
  exp.originalmatrix = FALSE,
  exp.row_prox = FALSE,
  exp.col_prox = FALSE,
  PNGfilename = NULL,
  PNGwidth = 1800,
  PNGheight = 1200,
  PNGres = 150,
  show.plot = FALSE
)
```

Arguments

`data`	A data frame to be visualized.
`isProximityMatrix`	Logical. Whether the input data is already a proximity matrix.
`XdNum`, `XcNum`, `YdNum`, `YcNum`	Integer vectors specifying discrete/continuous covariates on X and Y axes.
`row.name`	Either a character vector, or an integer vector to be used as row names.
`Xd.name`, `Xc.name`	Either A string, or a character vector to be used as Xc.name/Xd.name.
`row.prox`, `col.prox`	A string indicating the method used to compute row/column proximity.
`show.row.prox`, `show.col.prox`	Logical. Whether to show row/column proximity matrices.
`row.order`, `col.order`	A string specifying the method used to order rows/columns.
`row.flip`, `col.flip`	A string specifying the row/column flipping method.
`row.externalOrder`, `col.externalOrder`	Integer vectors used as external references for flipping.
`original.color`	Color palette for the original data matrix.
`row.color`, `col.color`	Color palettes for the row/column proximity matrices.
`Xd.color`, `Xc.color`, `Yd.color`, `Yc.color`	Color palettes for covariate matrices.
`row.label.size`, `col.label.size`	Numeric values controlling the font size of row and column labels.
`Xd.label.size`, `Xc.label.size`, `Yd.label.size`, `Yc.label.size`	Numeric values controlling the font size of covariate labels for X and Y axes.
`colorbar.margin`	Numeric. The margin space between the colorbar and the main plot area.
`border`	Logical. Whether to draw borders around each matrix.
`border.width`	Numeric value specifying border width.

`isContainMissingValue`
: Integer. Set to `1` if the input data contains missing values; otherwise, use `0`.

`MissingValue.color`
: Color to represent missing values in the matrix. Default is `"gray"`.

`exp.row_order`, `exp.column_order`
: Logical. Whether to export row/column order.

`exp.row_names`, `exp.column_names`
: Logical. Whether to export sorted row/column names.

`exp.Xc`, `exp.Yc`, `exp.Xd`, `exp.Yd`
: Logical. Whether to export sorted covariate matrices.

`exp.Xd_codebook`, `exp.Yd_codebook`
: Logical. Whether to export codebooks for discrete covariates.

`exp.originalmatrix`
: Logical. Whether to export the reordered original matrix.

`exp.row_prox`, `exp.col_prox`
: Logical. Whether to export computed proximity matrices (after ordering).

`PNGfilename` A string specifying the output filename for the PNG image.

`PNGwidth`, `PNGheight`
: Width/height of the PNG image in pixels.

`PNGres` Resolution of the PNG image in DPI.

`show.plot` Logical. Whether to display the plot in the R graphics window after generation.

Details

isProximityMatrix

If `isProximityMatrix = TRUE`, you may directly provide a proximity matrix as the input `data`. In this case, only row-based settings will be applied, such as `row.order`, `row.flip`, and `row.externalOrder`. Note that correlation matrices (e.g., `"pearson"`) must be converted to distance matrices before being used, and the selected color scheme must also be one of the supported diverging palettes (e.g., `"GAP_Blue_White_Red"`, `"BrBG"`, `"PiYG"`, `"PRGn"`, `"PuOr"`, `"RdBu"`, `"RdGy"`).

XdNum, XcNum, YdNum, YcNum

These parameters are used to specify which columns in `data` should be treated as covariates on the X or Y axes. Provide the column indices (e.g., `XdNum = c(3, 5)`) of discrete or continuous variables.

Xd.name, Xc.name

If not provided, the default labels will be a sequence of numbers based on the number of selected variables (e.g., "1", "2", ..., up to the length of XdNum or XcNum).

row.name

This parameter can be:

- A character vector providing custom row names.
- An integer (column index) indicating a column in `data` to be used as row names.
- If `row.name = NULL`, the row names will be automatically generated as `1:nrow(data)`.

row.prox, col.prox

Available proximity methods for `row.prox` and `col.prox` include:

- "euclidean"
- "pearson"
- "kendall"
- "spearman"
- "atancorr" (adjusted tangent correlation)
- "city-block" (Manhattan distance)
- "abs_pearson"
- "uncenteredcorr"
- "abs_uncenteredcorr"
- "maximum"
- "canberra"

For binary data, the following methods are supported:

- "hamman"
- "jaccard"
- "phi"
- "rao"
- "rogers"
- "simple"
- "sneath"
- "yule"

show.row.prox, show.col.prox

If set to TRUE, the corresponding proximity matrix will be visualized. If set to FALSE, the proximity matrix will not be shown, but the associated proximity and ordering methods will still be applied. In such cases, the dendrogram (tree structure) will appear alongside the original plot, reflecting the proximity-based ordering.

row.order, col.order

The ordering method determines how the rows or columns are reordered. Supported options include:

- "original" — Use the original data order.
- "random" — Randomly permute the order.
- "reverse" — Reverse the original order.
- "r2e" — Rank-two ellipse ordering.
- "single" — Single-linkage hierarchical clustering.
- "complete" — Complete-linkage hierarchical clustering.
- "average" — Average-linkage hierarchical clustering (UPGMA).
- *any method name from the* seriation *package* — such as "TSP", "Spectral", "ARSA", etc.

If the ordering method is "original", "random", or "reverse", then proximity matrices are not required, and the parameters row.prox or col.prox may be left unset.

For all other ordering methods, a proximity matrix must be computed first. Therefore, row.prox or col.prox must be specified accordingly.

Note: it is necessary to explicitly specify one of the valid ordering options; the function does not assume a default.

row.flip, col.flip

Supported flipping methods include:

- "r2e" — Flip using the rank-two ellipse (R2E) method.
- "uncle" — Apply uncle-flipping based on tree structure.
- "grandpa" — Apply grandpa-flipping based on tree structure.

Usage restrictions:

1. Flipping is only applicable when a hierarchical clustering tree is generated. Therefore, if row.order or col.order is set to "original", "random", "reverse", "r2e", or a seriation method, tree structures are not built and flipping cannot be applied.
2. When using "r2e" as the ordering method, only "r2e" flipping is allowed. "uncle" or "grandpa" flipping will be ignored.
3. **Do not specify both** externalOrder **and** flip **at the same time.** These options are mutually exclusive. If both are provided, the function will throw an error.

row.externalOrder, col.externalOrder

External orders are used as references when flipping the hierarchical clustering tree. If a tree is available, the external order guides the flipping of the dendrogram's leaf nodes to better match a predefined sequence.

Important: Do not use externalOrder together with flip; they are mutually exclusive.

Color settings

The function supports a variety of color palette options for visualizing the original matrix, proximity matrices, and covariate matrices.

Supported built-in palettes include:

- "GAP_Rainbow"
- "GAP_Blue_White_Red"
- "GAP_d"
- "grayscale_palette"

You may also specify any palette name from the RColorBrewer package. However, note that some palettes—such as those under the "Qualitative" category—are not suitable for visualizing continuous data like proximity matrices.

All palette names must be passed as character strings (e.g., "GAP_Rainbow", "Set1").

original.color: The system will automatically determine the appropriate default color palette based on data type. If the input data is binary, the default is a grayscale palette; otherwise, it defaults to "GAP_Rainbow".

row.color, col.color: The system chooses a default palette based on the proximity method used. For distance-based methods (e.g., `"euclidean"`, `"city-block"`), the default is `"GAP_Rainbow"`. For correlation-based methods (e.g., `"pearson"`, `"spearman"`), the default is `"GAP_Blue_White_Red"`.

Xd.color, Yd.color (discrete covariates): The default color palette is `"GAP_d"`, which supports up to 16 distinct categories. If there are more than 16 unique levels, a custom palette should be provided by the user.

Label size settings

Font sizes for axis labels and covariate matrices can be customized individually. Default values are:

- `row.label.size`: 2
- `col.label.size`: 8
- `Xd.label.size`, `Xc.label.size`, `Yd.label.size`, `Yc.label.size`, `Xc.label.size`: 8

You may increase or decrease these values to improve readability depending on figure size and resolution.

Export-related options (`exp.*`)

When any of the `exp.*` parameters are set to `TRUE`, the corresponding data will be stored in a list and returned by the function. This allows users to programmatically retrieve the order, reordered matrix, proximity matrices, covariate data, or codebooks after plotting.

PNG output settings

The following parameters control the export of the PNG image:

- `PNGfilename`: The name of the PNG file to be saved.
  - The file extension `.png` must be included manually (e.g., `"myplot.png"`).
  - If no file path is specified, the image will be saved in a system-generated temporary directory (via `tempdir()`) using the default filename `"output_plot.png"`.
  - To save the image to a specific location, provide the full path (e.g., `"C:/.../myplot.png"`).
- `PNGwidth`: Width of the output image in pixels. Default = 1800.
- `PNGheight`: Height of the output image in pixels. Default = 1200.
- `PNGres`: Resolution (dots per inch, DPI). Default = 150.

Value

A composite plot (e.g., heatmap with annotations) is saved or displayed. Additional information may be exported based on the settings.

If one or more export-related options (`exp.*`) are set to `TRUE`, the function returns a list containing the requested components. Each element in the list corresponds to an exportable data object,

Examples

```
# Example using the crabs dataset from the MASS package
if (requireNamespace("MASS", quietly = TRUE)) {
  df_crabs <- MASS::crabs
  CRAB_result <- GAP(
    data = df_crabs,
    YdNum = c(1,2),        # First two columns as Y discrete covariates
```

```
  YcNum = 3,             # Third column as Y continuous covariate
  row.name = c(1,2,3),   # Use First three columns as row names
  row.prox = "euclidean",
  col.prox = "euclidean",
  row.order = "average",
  col.order = "average",
  row.flip = "r2e",
  col.flip = "r2e",
  border = TRUE,
  border.width = 1,
  exp.row_order = TRUE,
  exp.column_order = TRUE,
  exp.row_names = TRUE,
  exp.column_names = TRUE,
  exp.Yd_codebook = TRUE,
  exp.Yd = TRUE,
  exp.Yc = TRUE,
  exp.originalmatrix = TRUE,
  exp.row_prox = TRUE,
  exp.col_prox = TRUE,
  PNGfilename = file.path(tempdir(), "output_plot.png"),
  show.plot = TRUE
)

# Access exported results:
CRAB_result$row_order       # Row order after ordering
CRAB_result$column_order    # Column order after ordering
CRAB_result$row_names       # Row names after ordering
CRAB_result$column_names    # Column names after ordering
CRAB_result$Yd_codebook     # Codebook for Y discrete covariates
CRAB_result$Yd              # Y discrete covariates after ordering
CRAB_result$Yc              # Y continuous covariates after ordering
CRAB_result$originalmatrix  # Original matrix (after ordering)
CRAB_result$row_prox        # Row proximity matrix (after ordering)
CRAB_result$col_prox        # Column proximity matrix (after ordering)

# Evaluate row ordering quality
AR(CRAB_result$row_prox, CRAB_result$row_order)
GAR(CRAB_result$row_prox, CRAB_result$row_order, w = 10)
RGAR(CRAB_result$row_prox, CRAB_result$row_order, w = 10)

# Evaluate column ordering quality
AR(CRAB_result$col_prox, CRAB_result$column_order)
GAR(CRAB_result$col_prox, CRAB_result$column_order)
RGAR(CRAB_result$col_prox, CRAB_result$column_order)
}
```

GAP_Blue_White_Red *Color palette: GAP_Blue_White_Red*

Description

A diverging color palette from blue to white to red, suitable for visualizing correlations.

Usage

```
GAP_Blue_White_Red
```

Format

An object of class `character` of length 20.

`GAP_d` *Color palette: GAP_d*

Description

A 16-category discrete color palette.

Usage

```
GAP_d
```

Format

An object of class `character` of length 16.

`GAP_Rainbow` *Color palette: GAP_Rainbow*

Description

A color palette with 13 colors in rainbow used for continuous data.

Usage

```
GAP_Rainbow
```

Format

An object of class `character` of length 13.

GAR *Compute the Generalized Anti-Robinson (GAR) score*

Description

Calculates the number of Anti-Robinson violations within a specified window `w`, allowing evaluation of local structural consistency in the reordered matrix.

Usage

```
GAR(mat, order, w = NULL)
```

Arguments

`mat`	A symmetric numeric distance matrix.
`order`	A permutation vector specifying the new row/column order.
`w`	Window size (integer). If NULL, uses global comparisons (equivalent to AR).

Value

The GAR score (the total number of violations).

Please refer to `GAP` for complete usage examples.

`grayscale_palette` *Color palette: grayscale_palette*

Description

A simple two-color grayscale palette from white to black, often used for binary data visualization.

Usage

```
grayscale_palette
```

Format

An object of class `character` of length 2.

`hctree_sort` *HCTree Sort*

Description

This function applies a hierarchical clustering tree (HCT) sorting algorithm to reorder rows of a proximity matrix. It supports external ordering constraints, different linkage-based order types, and optional flipping for optimal layout.

Usage

```
hctree_sort(distance_matrix, externalOrder = NULL, orderType, flipType)
```

Arguments

`distance_matrix`	A square numeric proximity matrix (either n × n or p × p) representing pairwise distances between items.
`externalOrder`	An integer vector specifying an initial or external ordering of the items (can be empty or NULL if not used).
`orderType`	An integer indicating the type of hierarchical clustering order to apply.
`flipType`	An integer indicating the flipping methods.

Details

distance_matrix

The input matrix must represent pairwise distances between items. If you start with a similarity matrix (e.g., a correlation matrix), you must convert it to a dissimilarity matrix before use. For example, for correlation-based similarities, use `as.matrix(as.dist(1 - cor_matrix))` or other appropriate transformations to convert it to a proper distance matrix. The matrix should also be symmetric and non-negative.

orderType

Specifies the linkage method used for hierarchical clustering:

- `0`: Single-linkage
- `1`: Complete-linkage
- `2`: Average-linkage (UPGMA)

flipType

Controls how the branches of the clustering tree are flipped:

- `1`: Flip based on `externalOrder` This option should be used only when `externalOrder` is provided.
- `2`: Uncle-flipping
- `3`: Grandpa-flipping

Important: Do not specify both `flipType = 1` and a `NULL` or missing `externalOrder`. When using `flipType = 1`, `externalOrder` must be a valid integer vector.

Please refer to `computeProximity` for complete usage examples.

Value

A list representing a dendrogram tree structure, containing: `left`, `right`, and `height` for tree construction, and `order` for the optimal leaf order.

RGAR *Compute the Relative Generalized Anti-Robinson (RGAR) score*

Description

This function returns the relative GAR score, representing the proportion of Anti-Robinson violations over the total number of evaluated triplets.

Usage

```
RGAR(mat, order, w = NULL)
```

Arguments

`mat`	A symmetric numeric distance matrix.
`order`	A permutation vector specifying the new row/column order.
`w`	Window size (integer). If NULL, uses global comparisons (equivalent to AR normalized).

Value

The RGAR score (between 0 and 1).

Please refer to `GAP` for complete usage examples.

Index