Package 'L1centrality'

July 14, 2026

Title Graph/Network Analysis Based on L1 Centrality

Version 0.5.2

Description Analyze graph/network data using L1 centrality and prestige. Functions for deriving global, local, and group L1 centrality/prestige are provided. Routines for visual inspection of a graph/network are also provided. Details are in Kang and Oh (2026a) <doi:10.1080/01621459.2025.2520467>, Kang and Oh (2026b) <doi:10.1080/00031305.2

URL https://github.com/seungwoo-stat/L1centrality

BugReports https://github.com/seungwoo-stat/L1centrality/issues

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 2.10)

LazyData true

Imports doParallel, foreach, graphics, igraph, methods, parallel, Rcpp, stats, utils, withr

LinkingTo Rcpp

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Seungwoo Kang [aut, cre] (ORCID: <https://orcid.org/0000-0001-8082-0794>), Hee-Seok Oh [aut]

Maintainer Seungwoo Kang <kangsw@skku.edu>

Repository CRAN

Date/Publication 2026-07-14 02:40:12 UTC

Contents

group_reduce 2
Heterogeneity 4
L1cent 5
L1centEDGE 9

L1centGROUP 11
L1centLOC 13
L1centMDS 16
L1centNB 19
MCUmovie 21
rokassembly21 22

Index **24**

`group_reduce` *Group Reduced Graph*

Description

Computes the vertex multiplicities (vertex weights) and the distance matrix of the group reduced graph. The group reduced graph is constructed by replacing a group of vertices in the original graph by a single 'pseudo-vertex'.

Usage

```
group_reduce(g, nodes, vertex_weight, method, edge_weight_transform)

## S3 method for class 'igraph'
group_reduce(
  g,
  nodes,
  vertex_weight = NULL,
  method = c("minimum", "maximum", "average"),
  edge_weight_transform = NULL
)

## S3 method for class 'matrix'
group_reduce(
  g,
  nodes,
  vertex_weight = NULL,
  method = "minimum",
  edge_weight_transform = NULL
)
```

Arguments

`g`	An `igraph` graph object or a distance matrix. When `g` is a distance matrix, the (i, j) component of the distance matrix is the geodesic distance from the ith vertex to the jth vertex.
`nodes`	A vector of integers. Each integer indicates the index of the vertex.

`vertex_weight` An optional nonnegative multiplicity (weight) vector for (vertex) weighted networks. The sum of its components must be positive. If set to `NULL` (the default), all vertices will have the same positive weight (multiplicity) of 1, i.e., `g` is treated as a vertex unweighted graph. The length of the `vertex_weight` must be equivalent to the number of vertices.

`method` A character string. It specifies the method of setting the edge weight between the pseudo-vertex and the other vertices. Note that the S3 method for the `matrix` class only supports the `minimum` option. This is because it is not possible to derive the group reduced graph's distance matrix from the original distance matrix when using the maximum or average method. On the other hand, the group reduced graph's distance matrix can be derived from the original distance matrix when the minimum method is used. See the discussion in Kang (2025).

- `minimum` (the default): the minimum method is used in setting the edge weights.
- `maximum`: the maximum method is used in setting the edge weights.
- `average`: the average method is used in setting the edge weights.

`edge_weight_transform` An optional function to transform the edge weights when `g` is an `igraph` object and an edge weight attribute exists. This argument is ignored when `g` is a distance matrix.

Details

The group reduced graph is constructed by replacing the vertices indicated in the argument `nodes` with a single 'pseudo-vertex'. The multiplicity (vertex weight) of this new vertex is set to the sum of the multiplicities of the vertices within `nodes`. An edge from the pseudo-vertex to any vertex that is not in `nodes`, say v, is created in the group reduced graph if there is at least one edge from the vertices in `nodes` to v in the original graph. The weight of this newly added edge is determined using one of the following methods:

- Minimum method: The edge weight from the pseudo-vertex to v is set to the minimum of the edge weights of the edges between the vertices in `nodes` to v in the original graph.
- Maximum method: The edge weight from the pseudo-vertex to v is set to the maximum of the edge weights of the edges between the vertices in `nodes` to v in the original graph.
- Average method: The edge weight from the pseudo-vertex to v is set to the average of the edge weights of the edges between the vertices in `nodes` to v in the original graph.

An edge from v to the pseudo-vertex is set in a similar manner. For details, refer to Kang (2025).

Value

A list consisting of three objects:

- 'distmat': A matrix representing the group reduced graph's distance matrix, where the first row and column correspond to the pseudo-vertex.
- 'vertex_weight': A vector of the group reduced graph's vertex multiplicity. The first element corresponds to the pseudo-vertex.
- 'label': A vector of the vertex names specified by `nodes`.

Note

Multiple edges (edges with the same head and tail vertices) are not allowed, because they make the edge weight setting procedure confusing.

References

S. Kang. *Topics in Non-Euclidean Dimension Reduction*. PhD thesis, Seoul National University, 2025.

See Also

`L1cent()` for L_1 centrality/prestige. `L1centGROUP()` internally uses `group_reduce()`.

Examples

```
# Group reduced graph of the 'Iron Man' series using the minimum method
vertex_weight <- igraph::V(MCUmovie)$worldwidegross
ironman_series <- c(1,3,7)
reduced_graph <- group_reduce(MCUmovie, nodes = ironman_series, vertex_weight = vertex_weight)
reduced_graph$distmat[1:3,1:3]
reduced_graph$label

# Multiplicity of the pseudo-vertex equals the sums of the replaced vertices' multiplicities
reduced_graph$vertex_weight[1] == sum(vertex_weight[ironman_series])
```

Heterogeneity *Lorenz Curve and the Gini Coefficient*

Description

Draws a Lorenz curve (the group heterogeneity plot) and computes the Gini coefficient (the group heterogeneity index).

Usage

```
Lorenz_plot(x, add = FALSE, ...)

Gini(x)
```

Arguments

x	A numeric vector.
add	A logical value. • `TRUE`: add the Lorenz curve to an already existing plot. • `FALSE` (the default): draw the Lorenz curve to a new graphic device.
...	Further graphical parameters supplied to the internal `base::plot()` (when add = `FALSE`) or `graphics::lines()` (when add = `TRUE`) function. See `graphics::par()` document.

Value

`Lorenz_plot()` draws a Lorenz curve (the group heterogeneity plot) and returns an invisible copy of a Gini coefficient (the group heterogeneity index).

`Gini()` returns a Gini coefficient.

References

S. Kang and H.-S. Oh. On a notion of graph centrality based on L_1 data depth. *Journal of the American Statistical Association*, 121(553): 400–412, 2026.

M. O. Lorenz. Methods of measuring the concentration of wealth. *Publications of the American Statistical Association*, 9(70): 209–219, 1905.

See Also

`plot()` methods for objects of class L1cent and L1centLOC support plotting a Lorenz curve. `summary()` methods for objects of class L1cent, L1centLOC, and L1centNB provide the Gini coefficient as one of the summary statistics.

Examples

```
vertex_weight <- igraph::V(MCUmovie)$worldwidegross
cent <- L1cent(MCUmovie, vertex_weight = vertex_weight)
gini <- Lorenz_plot(cent, asp = 1) # one can use "plot(cent, asp = 1)"
graphics::abline(0, 1, lty = 2)
# group heterogeneity index
gini
gini == Gini(cent)
```

`L1cent` *L1 Centrality/Prestige*

Description

Computes L_1 centrality or L_1 prestige for each vertex. The L_1 centrality/prestige is a graph centrality/prestige measure defined for the vertices of a graph. It is (roughly) defined by (1 − minimum multiplicity required for a selected vertex to become the median of the graph). For directed graphs, L_1 centrality quantifies the prominence of a vertex in *making* a choice and L_1 prestige quantifies the prominence of a vertex in *receiving* a choice. For undirected graphs, the two measures are identical.

Usage

```
L1cent(g, vertex_weight, mode, edge_weight_transform)

## S3 method for class 'igraph'
L1cent(
  g,
  vertex_weight = NULL,
```

```
  mode = c("centrality", "prestige"),
  edge_weight_transform = NULL
)

## S3 method for class 'matrix'
L1cent(
  g,
  vertex_weight = NULL,
  mode = c("centrality", "prestige"),
  edge_weight_transform = NULL
)

## S3 method for class 'L1cent'
print(x, ...)

## S3 method for class 'L1cent'
plot(x, y = NULL, add = FALSE, ...)

## S3 method for class 'L1cent'
summary(object, ...)
```

Arguments

g	An igraph graph object or a distance matrix. • When g is an igraph object, the graph must be connected. For a directed graph, it must be strongly connected. • When g is a matrix, it should represent a distance matrix. All entries of the distance matrix must be finite. Here, the (i, j) component of the distance matrix is the geodesic distance from the ith vertex to the jth vertex.
vertex_weight	An optional nonnegative multiplicity (weight) vector for (vertex) weighted networks. The sum of its components must be positive. If set to NULL (the default), all vertices will have the same positive weight (multiplicity) of 1, i.e., g is treated as a vertex unweighted graph. The length of the vertex_weight must be equivalent to the number of vertices.
mode	A character string. For an undirected graph, either choice gives the same result. • centrality (the default): L_1 centrality (prominence of each vertex in terms of *making* a choice) is used for analysis. • prestige: L_1 prestige (prominence of each vertex in terms of *receiving* a choice) is used for analysis.
edge_weight_transform	An optional function to transform the edge weights when g is an igraph object and an edge weight attribute exists. This argument is ignored when g is a distance matrix.
x	An L1cent object, obtained as a result of the function L1cent().
...	Further arguments passed to or from other methods.
y	An optional argument providing the coordinates for a scatter plot. It could be an object of class L1cent or L1centLOC, or a numeric vector.

add — A logical value. This argument is considered only when drawing a Lorenz curve.
- TRUE: add the Lorenz curve to an already existing plot.
- FALSE (the default): draw the Lorenz curve to a new graphic device.

object — An L1cent object, obtained as a result of the function L1cent().

Details

Suppose that g is a (strongly) connected graph consisting of n vertices $v_1, \ldots, v_n$ whose multiplicities (vertex weights) are $\eta_1, \ldots, \eta_n \geq 0$, respectively, and $\eta_\cdot = \sum_{k=1}^n \eta_k > 0$.

The centrality median vertex of this graph is the node minimizing the weighted sum of distances. That is, v_i is the centrality median vertex if

$$\sum_{k=1}^{n} \eta_k d(v_i, v_k)$$

is minimized, where $d(v_i, v_k)$ denotes the geodesic (shortest path) distance from v_i to v_k. See igraph::distances() for algorithms for computing geodesic distances between vertices. When the indices are swapped to $d(v_k, v_i)$ in the display above, we call the node minimizing the weighted sum as the prestige median vertex. When the graph is undirected, the prestige median vertex and the centrality median vertex coincide, and we call it the graph median, following Hakimi (1964).

The L_1 centrality for an arbitrary node v_i is defined as 'one minus the minimum weight that is required to make it a centrality median vertex.' This concept of centrality is closely related to the data depth for ranking multivariate data, as defined in Vardi and Zhang (2000). It turns out that the following formula computes the L_1 centrality for the vertex v_i:

$$1 - \mathcal{S}(\mathsf{g}) \max_{j \neq i} \left\{ \frac{\sum_{k=1}^n \eta_k (d(v_i, v_k) - d(v_j, v_k))}{\eta_\cdot d(v_j, v_i)} \right\}^+,$$

where $\{\cdot\}^+ = \max(\cdot, 0)$ and $\mathcal{S}(\mathsf{g}) = \min_{i \neq j} d(v_i, v_j)/d(v_j, v_i)$. The L_1 centrality of a vertex is in $[0, 1]$ by the triangle inequality, and the centrality median vertex has centrality 1. The L_1 prestige is defined analogously, with the indices inside the distance function swapped.

For an undirected graph, $\mathcal{S}(\mathsf{g}) = 1$ since the distance function is symmetric. Moreover, L_1 centrality and L_1 prestige measures concide.

For details, refer to Kang and Oh (2026a) for undirected graphs, and Kang and Oh (2026b) for directed graphs.

Value

L1cent() returns an object of class L1cent. It is a numeric vector whose length is equivalent to the number of vertices in the graph g. Each component of the vector is the L_1 centrality (if mode = "centrality") or the L_1 prestige (if mode = "prestige") of each vertex in the given graph.

print.L1cent() prints L_1 centrality or L_1 prestige values and returns them invisibly.

plot.L1cent() draws a Lorenz curve (the group heterogeneity plot) and returns an invisible copy of a Gini coefficient (the group heterogeneity index) if argument y is not supplied. Otherwise, it draws a scatter plot.

summary.L1cent() returns an object of class table. It is a summary of the prominence values with the five-number summary, mean, and the Gini coefficient.

Note

The function is valid only for connected graphs. If the graph is directed, it must be strongly connected.

References

S. L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3): 450–459, 1964.

S. Kang and H.-S. Oh. On a notion of graph centrality based on L_1 data depth. *Journal of the American Statistical Association*, 121(553): 400–412, 2026a.

S. Kang and H.-S. Oh. L_1 prominence measures for directed graphs. *The American Statistician*, 80(2): 301–309, 2026b.

Y. Vardi and C.-H. Zhang. The multivariate L_1-median and associated data depth. *Proceedings of the National Academy of Sciences*, 97(4): 1423–1426, 2000.

See Also

`L1centLOC()`, `L1centNB()`, `L1centMDS()`, `L1centEDGE()`, `L1centGROUP()` for L_1 centrality- or prestige-based analysis. See L1centrality-package for each function's support range.

`igraph::betweenness()`, `igraph::closeness()`, `igraph::degree()`, `igraph::eigen_centrality()` for centrality measures.

Heterogeneity for drawing the Lorenz curve and computing the Gini coefficient.

Examples

```
# igraph object and distance matrix as an input lead to the same result
vertex_weight <- igraph::V(MCUmovie)$worldwidegross
cent_igraph <- L1cent(MCUmovie, vertex_weight = vertex_weight)
cent_matrix <- L1cent(igraph::distances(MCUmovie), vertex_weight = vertex_weight)
all(cent_igraph == cent_matrix)

# Top 6 vertices with the highest L1 centrality
utils::head(sort(cent_igraph, decreasing = TRUE))

# Lorenz curve
plot(cent_igraph)

# Summary statistics
summary(cent_igraph)
```

L1centEDGE *Multiscale Edge Representation*

Description

Derives a multiscale edge representation. Each vertex is connected to its local median, which is found in its L_1 centrality-based neighborhood.

Usage

```
L1centEDGE(g, vertex_weight, alpha, edge_weight_transform)

## S3 method for class 'igraph'
L1centEDGE(g, vertex_weight = NULL, alpha, edge_weight_transform = NULL)

## S3 method for class 'matrix'
L1centEDGE(g, vertex_weight = NULL, alpha, edge_weight_transform = NULL)

## S3 method for class 'L1centEDGE'
print(x, ...)

## S3 method for class 'L1centEDGE'
plot(x, ...)

## S3 method for class 'L1centEDGE'
summary(object, ...)
```

Arguments

g	An igraph graph object or a distance matrix. • When g is an igraph object, the graph must be undirected and connected. • When g is a matrix, it should be symmetric and represent a distance matrix. All entries of the distance matrix must be finite.
vertex_weight	An optional nonnegative multiplicity (weight) vector for (vertex) weighted networks. The sum of its components must be positive. If set to NULL (the default), all vertices will have the same positive weight (multiplicity) of 1, i.e., g is treated as a vertex unweighted graph. The length of the vertex_weight must be equivalent to the number of vertices.
alpha	A number or a numeric vector of locality levels. Values must be between 0 and 1.
edge_weight_transform	An optional function to transform the edge weights when g is an igraph object and an edge weight attribute exists. This argument is ignored when g is a distance matrix.
x	An L1centEDGE object, obtained as a result of the function L1centEDGE().

`...` Further arguments passed to or from other methods.

- `plot()` method: Further graphical parameters supplied to the internal `igraph::plot.igraph()` function. See igraph::plot.common document.
- `print()` and `summary()` methods: Further arguments passed to the `base::print()` and `base::summary()` functions, respectively.

`object` An `L1centEDGE` object, obtained as a result of the function `L1centEDGE()`.

Details

In a global perspective, any given undirected graph can be represented as a star-shaped directed graph, with each vertex making a connection to the median vertex. Based on this idea, an undirected graph can be represented as a directed graph, with each vertex making a connection to the *local* median vertex. The local median vertex of, say, v_i, is defined as a median vertex among the L_1 centrality-based neighborhood of v_i. By varying the level of locality, the given graph can be visually inspected at multiple scales. Refer to Kang and Oh (2026) for details.

Value

`L1centEDGE()` returns an object of class `L1centEDGE`. It is a list of 'edge lists'. The length of the list is equivalent to the length of `alpha`, and the names of the list are the values of `alpha`. The `i`th component of the list is a 2-column matrix, and each row defines one directed edge, i.e., it is an edge list. The second column is the local (level `alpha[i]`) median of the vertex at the first column. There may be more than one edge from each vertex, since there may be more than one local median.

`print.L1centEDGE()` prints the edge lists and returns them invisibly.

`plot.L1centEDGE()` draws directed graphs corresponding to each value of `alpha`. In each display, each vertex gives a directed edge to its local median vertex. The local median vertices are shown with larger circles.

`summary.L1centEDGE()` returns an object of class `table` with the number of local medians at each locality level `alpha`.

Note

The function is valid only for undirected and connected graphs.

References

S. Kang and H.-S. Oh. On a notion of graph centrality based on L_1 data depth. *Journal of the American Statistical Association*, 121(553): 400–412, 2026.

See Also

`L1cent()`, `L1centNB()`, `L1centLOC()`.

Examples

```
library(igraph)
MCU_edge <- L1centEDGE(MCUmovie, vertex_weight = V(MCUmovie)$worldwidegross, alpha = 5/32)
plot(MCU_edge)
```

L1centGROUP *Group L1 Centrality/Prestige*

Description

Computes group L_1 centrality or group L_1 prestige for the specified groups of vertices. For undirected graphs, the two measures are identical.

Usage

```
L1centGROUP(g, nodes, vertex_weight, mode, method, edge_weight_transform)

## S3 method for class 'igraph'
L1centGROUP(
  g,
  nodes,
  vertex_weight = NULL,
  mode = c("centrality", "prestige"),
  method = c("minimum", "maximum", "average"),
  edge_weight_transform = NULL
)

## S3 method for class 'matrix'
L1centGROUP(
  g,
  nodes,
  vertex_weight = NULL,
  mode = c("centrality", "prestige"),
  method = "minimum",
  edge_weight_transform = NULL
)

## S3 method for class 'L1centGROUP'
print(x, ...)
```

Arguments

g	An igraph graph object or a distance matrix. • When g is an igraph object, the graph must be connected. For a directed graph, it must be strongly connected. • When g is a matrix, it should represent a distance matrix. All entries of the distance matrix must be finite. Here, the (i, j) component of the distance matrix is the geodesic distance from the ith vertex to the jth vertex.
nodes	A vector of integers or a list of integer vectors. Each integer indicates the index of the vertex. When the argument is provided as a list, the function computes the group L_1 prominence for each set of vertices specified by the individual list

elements. When it is given as an integer vector, the function computes the group L_1 prominence for the single group defined by that vector.

`vertex_weight` An optional nonnegative multiplicity (weight) vector for (vertex) weighted networks. The sum of its components must be positive. If set to `NULL` (the default), all vertices will have the same positive weight (multiplicity) of 1, i.e., g is treated as a vertex unweighted graph. The length of the `vertex_weight` must be equivalent to the number of vertices.

`mode` A character string. For an undirected graph, either choice gives the same result.

- `centrality` (the default): L_1 centrality (prominence of each vertex in terms of *making* a choice) is used for analysis.
- `prestige`: L_1 prestige (prominence of each vertex in terms of *receiving* a choice) is used for analysis.

`method` A character string. It specifies the method of setting the edge weight between the pseudo-vertex and the other vertices. Note that the S3 method for the `matrix` class only supports the `minimum` option. This is because it is not possible to derive the group reduced graph's distance matrix from the original distance matrix when using the maximum or average method. On the other hand, the group reduced graph's distance matrix can be derived from the original distance matrix when the minimum method is used. See the discussion in Kang (2025).

- `minimum` (the default): the minimum method is used in setting the edge weights.
- `maximum`: the maximum method is used in setting the edge weights.
- `average`: the average method is used in setting the edge weights.

`edge_weight_transform`
An optional function to transform the edge weights when g is an `igraph` object and an edge weight attribute exists. This argument is ignored when g is a distance matrix.

`x` An `L1centGROUP` object, obtained as a result of the function `L1centGROUP()`.

`...` Further arguments passed to or from other methods. This argument is ignored here.

Details

Given a group of vertices on a graph, we first construct a group reduced graph by replacing the group of vertices by a single 'pseudo-vertex' (see `group_reduce()` for the method of setting vertex multiplicities and edge weights in the group reduced graph). The group L_1 centrality (prestige) of this group is defined as the L_1 centrality (prestige) of the pseudo-vertex in the group reduced graph.

Value

`L1centGROUP()` returns an object of class `L1centGROUP`. It is a numeric value of the group L_1 centrality (if `mode = "centrality"`) or the group L_1 prestige (if `mode = "prestige"`) of the specified group of vertices.

`print.L1centGROUP()` prints group L_1 centrality or group L_1 prestige value and returns it invisibly.

Note

The function is valid only for connected graphs. If the graph is directed, it must be strongly connected. Multiple edges (edges with the same head and tail vertices) are not allowed, because they make the edge weight setting procedure confusing.

References

S. Kang. *Topics in Non-Euclidean Dimension Reduction*. PhD thesis, Seoul National University, 2025.

See Also

`L1cent()` for L_1 centrality/prestige, `group_reduce()` for details on the minimum, maximum, and average methods.

Examples

```
# Group L1 centrality of the 'Spider-Man' series
vertex_weight <- igraph::V(MCUmovie)$worldwidegross
L1centGROUP(MCUmovie, nodes = list(16, c(16,23,27)), vertex_weight = vertex_weight)
```

L1centLOC *Local L1 Centrality/Prestige*

Description

Computes local L_1 centrality or local L_1 prestige at each `alpha` level for every vertex. For undirected graphs, the two measures are identical.

Usage

```
L1centLOC(g, vertex_weight, alpha, mode, edge_weight_transform, parallel)

## S3 method for class 'igraph'
L1centLOC(
  g,
  vertex_weight = NULL,
  alpha,
  mode = c("centrality", "prestige"),
  edge_weight_transform = NULL,
  parallel = FALSE
)

## S3 method for class 'matrix'
L1centLOC(
  g,
  vertex_weight = NULL,
```

```
  alpha,
  mode = c("centrality", "prestige"),
  edge_weight_transform = NULL,
  parallel = FALSE
)

## S3 method for class 'L1centLOC'
print(x, ...)

## S3 method for class 'L1centLOC'
plot(x, y = NULL, add = FALSE, threshold = NULL, ...)

## S3 method for class 'L1centLOC'
summary(object, ...)
```

Arguments

g	An igraph graph object or a distance matrix. • When g is an igraph object, the graph must be connected. For a directed graph, it must be strongly connected. • When g is a matrix, it should represent a distance matrix. All entries of the distance matrix must be finite. Here, the (i, j) component of the distance matrix is the geodesic distance from the ith vertex to the jth vertex.
vertex_weight	An optional nonnegative multiplicity (weight) vector for (vertex) weighted networks. The sum of its components must be positive. If set to NULL (the default), all vertices will have the same positive weight (multiplicity) of 1, i.e., g is treated as a vertex unweighted graph. The length of the vertex_weight must be equivalent to the number of vertices.
alpha	A number or a numeric vector of locality levels. Values must be between 0 and 1.
mode	A character string. For an undirected graph, either choice gives the same result. • centrality (the default): L_1 centrality (prominence of each vertex in terms of *making* a choice) is used for analysis. • prestige: L_1 prestige (prominence of each vertex in terms of *receiving* a choice) is used for analysis.
edge_weight_transform	An optional function to transform the edge weights when g is an igraph object and an edge weight attribute exists. This argument is ignored when g is a distance matrix.
parallel	A boolean indicating whether to run the algorithm parallel across multiple cores. By default set to FALSE.
x	An L1centLOC object, obtained as a result of the function L1centLOC().
...	Further arguments passed to or from other methods.
y	An optional argument providing the coordinates for a scatter plot. It could be an object of class L1cent or L1centLOC, or a numeric vector.

add	A logical value. This argument is considered only when drawing a Lorenz curve. • TRUE: add the Lorenz curve to an already existing plot. • FALSE (the default): draw the Lorenz curve to a new graphic device.
threshold	A number between 0 and 1. Vertices that have their maximum and minimum local L_1 prominence value difference above the threshold are indicated in colored lines.
object	An L1centLOC object, obtained as a result of the function L1centLOC().

Details

Suppose that the given graph has n vertices. We choose about $n\alpha$ vertices (L_1 centrality- or L_1 prestige-based neighborhood) for each vertex (see L1centNB()), and compute the L_1 centrality or L_1 prestige of the vertex conditioned on these vertices, i.e., derive the L_1 centrality or L_1 prestige locally. For details, refer to Kang and Oh (2026a) for undirected graphs, and Kang and Oh (2026b) for directed graphs.

Value

L1centLOC() returns an object of class L1centLOC. It is a list of numeric vectors. The length of the list is equivalent to the length of alpha, and the names of the list are the values of alpha. Each component of the list is a numeric vector whose length is equivalent to the number of vertices in the graph g. Specifically, the ith component of the list is a vector of local L_1 centrality at level alpha[i] for each vertex (if mode = "centrality") or local L_1 prestige at level alpha[i] for each vertex (if mode = "prestige").

print.L1centLOC() prints local L_1 centrality or local L_1 prestige values at each locality level alpha and returns them invisibly.

plot.L1centLOC() draws a following plot.

- y is not supplied and alpha is of length one: A Lorenz curve (the group heterogeneity plot) and returns an invisible copy of a Gini coefficient (the group heterogeneity index). threshold is ignored.
- y is supplied and alpha is of length one: A scatter plot of x versus y. threshold is ignored.
- alpha's length is larger than one: A plot of alpha versus local L_1 prominence values (in a uniform margin) for each vertex. If threshold is set, vertices that have their maximum and minimum local L_1 prominence value difference above the threshold are indicated in colored lines. y is ignored.

summary.L1centLOC() returns an object of class table. It is a summary of the prominence values with the five-number summary, mean, and the Gini coefficient, at each level of alpha.

Note

The function is valid only for connected graphs. If the graph is directed, it must be strongly connected.

References

S. Kang and H.-S. Oh. On a notion of graph centrality based on L_1 data depth. *Journal of the American Statistical Association*, 121(553): 400–412, 2026a.

S. Kang and H.-S. Oh. L_1 prominence measures for directed graphs. *The American Statistician*, 80(2): 301–309, 2026b.

See Also

`L1cent()` for L_1 centrality/prestige, `L1centNB()` for L_1 centrality/prestige-based neighborhood.

Examples

```
weight <- igraph::V(MCUmovie)$worldwidegross
MCUmovie_cent <- L1cent(MCUmovie, vertex_weight = weight)
MCUmovie_loc_cent <- L1centLOC(MCUmovie, vertex_weight = weight, alpha = 5/32)
plot(MCUmovie_cent, MCUmovie_loc_cent,
     main = "MCU movie network: global vs. local centrality")
```

L1centMDS *Fitting a Target Plot*

Description

`L1centMDS()` and `plot.L1centMDS()` are used together to draw a target plot, which is a target-shaped 2D plot that aids in the visual inspection of an undirected graph using the L_1 centrality. See Kang and Oh (2026) for a formal definition of a target plot.

Usage

```
L1centMDS(g, tol, maxiter, verbose, edge_weight_transform)

## S3 method for class 'igraph'
L1centMDS(
  g,
  tol = 1e-05,
  maxiter = 1000,
  verbose = TRUE,
  edge_weight_transform = NULL
)

## S3 method for class 'matrix'
L1centMDS(
  g,
  tol = 1e-05,
  maxiter = 1000,
  verbose = TRUE,
  edge_weight_transform = NULL
```

```
)

## S3 method for class 'L1centMDS'
plot(x, zoom = 1, main = NULL, show_edge = FALSE, ...)

## S3 method for class 'L1centMDS'
print(x, ...)
```

Arguments

g	An igraph graph object or a distance matrix. • When g is an igraph object, the graph must be undirected and connected. • When g is a matrix, it should be symmetric and represent a distance matrix. All entries of the distance matrix must be finite.
tol	A numerical tolerance. The gradient descent method terminates if the relative magnitude of the gradient falls below tol as in Kruskal (1964b). By default set to 10^{-5}.
maxiter	A number of maximum iteration allowances for the gradient descent algorithm. By default set to 1000.
verbose	A boolean. • TRUE (the default): for each iteration, prints (1) current number of iterations, (2) current stress, and (3) relative magnitude of the gradient to the console. At the end, the final message is printed to the console; total number of iterations and final stress. • FALSE: suppress message to the console.
edge_weight_transform	An optional function to transform the edge weights when g is an igraph object and an edge weight attribute exists. This argument is ignored when g is a distance matrix.
x	An L1centMDS object, obtained as a result of the function L1centMDS().
zoom	A numerical value on how much to zoom-in the plot. By default set to 1 (no zoom).
main	Title of the plot. If set to NULL (the default), the title prints "Target plot / Stress = X".
show_edge	A boolean. • TRUE: edges are shown in the target plot. When x, the L1centMDS object, is generated from a matrix object g, this argument is ignored since g does not carry information about the edge connections. That is, edges can only be drawn if the L1centMDS object is generated from an igraph object. • FALSE (the default): edges are not shown in the target plot.
...	Further arguments passed to or from other methods. • plot() method: Further graphical parameters supplied to the internal base::plot() (for points), graphics::text() (for labels), and graphics::segments() (for edges) functions. See graphics::par() document. To supply an argument to the first one, use the prefix 'plot.', for the second one, 'text.',

and for the last one, 'edge.'. For instance, `plot.cex = 1` sets the size of the point, whereas `text.cex = 1` sets the size of the label.

- `print()` method: This argument is ignored.

Details

Denoting the L_1 centrality of vertex i as $c_i \in (0, 1]$, a point representing that vertex is placed on a concentric circle with radius $r_i = -\log(c_i)$. Representing each vertex as (r_i, θ_i) (in circular coordinates), the values of θ_i are derived using nonmetric multidimensional scaling proposed in Kruskal (1964a,b). The initial configuration is derived using classical multidimensional scaling (`stats::cmdscale()`). A gradient descent algorithm is employed in deriving optimal θ_is.

Value

`L1centMDS()` returns an object of class `L1centMDS`. It is a list consisting of three vectors:

- 'radius': Radius of a point representing each vertex in the target plot's circular coordinate system, i.e., $-\log(L_1$ centrality$)$ for each vertex.
- 'theta': Angle (in radians) of a point representing each vertex in the target plot's circular coordinate system.
- 'stress': Stress measure defined in Kruskal (1964a).

`plot.L1centMDS()` draws a target plot. Four concentric circles denote the 1st to 4th quartiles of the radius, and the values of the L_1 centrality quartiles are shown in red text. Note that red texts denote the L_1 centrality quartiles, *not* radius quartiles.

`print.L1centMDS()` prints number of iterations it took to fit a target plot.

Note

The function `L1centMDS()` is valid only for undirected and connected graphs. Also, `L1centMDS()` only considers graphs with equal vertex multiplicities.

References

S. Kang and H.-S. Oh. On a notion of graph centrality based on L_1 data depth. *Journal of the American Statistical Association*, 121(553): 400–412, 2026.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1): 1–27, 1964a.

J. B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2): 115–129, 1964b.

See Also

`L1cent()` for L_1 centrality/prestige, `MASS::isoMDS()` and `stats::cmdscale()` for multidimensional scaling methods.

Examples

```
parameters <- L1centMDS(MCUmovie, verbose = FALSE)
plot(parameters)
```

L1centNB *L1 Centrality/Prestige-Based Neighborhood*

Description

Derives L_1 centrality- or L_1 prestige-based neighborhood of each vertex. For undirected graphs, the two neighborhood are identical.

Usage

```
L1centNB(g, vertex_weight, mode, edge_weight_transform)

## S3 method for class 'igraph'
L1centNB(
  g,
  vertex_weight = NULL,
  mode = c("centrality", "prestige"),
  edge_weight_transform = NULL
)

## S3 method for class 'matrix'
L1centNB(
  g,
  vertex_weight = NULL,
  mode = c("centrality", "prestige"),
  edge_weight_transform = NULL
)

## S3 method for class 'L1centNB'
print(x, ...)

## S3 method for class 'L1centNB'
summary(object, ...)
```

Arguments

g	An igraph graph object or a distance matrix. • When g is an igraph object, the graph must be connected. For a directed graph, it must be strongly connected. • When g is a matrix, it should represent a distance matrix. All entries of the distance matrix must be finite. Here, the (i, j) component of the distance matrix is the geodesic distance from the ith vertex to the jth vertex.
vertex_weight	An optional nonnegative multiplicity (weight) vector for (vertex) weighted networks. The sum of its components must be positive. If set to NULL (the default), all vertices will have the same positive weight (multiplicity) of 1, i.e., g is treated as a vertex unweighted graph. The length of the vertex_weight must be equivalent to the number of vertices.

mode — A character string. For an undirected graph, either choice gives the same result.

- `centrality` (the default): L_1 centrality (prominence of each vertex in terms of *making* a choice) is used for analysis.
- `prestige`: L_1 prestige (prominence of each vertex in terms of *receiving* a choice) is used for analysis.

edge_weight_transform — An optional function to transform the edge weights when g is an igraph object and an edge weight attribute exists. This argument is ignored when g is a distance matrix.

x — An L1centNB object, obtained as a result of the function L1cent().

... — Further arguments passed to or from other methods. This argument is ignored here.

object — An L1centNB object, obtained as a result of the function L1centNB().

Details

For an undirected graph, if the graph is symmetrized (in a way defined in Kang and Oh 2026a) w.r.t. a vertex v, vertex v becomes the graph median (Kang and Oh 2026a), i.e., v has L_1 centrality 1. Based on this property, we define the L_1 centrality-based neighborhood of vertex v as vertices that have large L_1 centrality on the symmetrized graph w.r.t. vertex v.

For a directed graph, a vertex of interest, say v, is made to a centrality and prestige median vertex by the procedure described in Kang and Oh (2026b). We call the resulting graph as the modified graph w.r.t. v. L_1 centrality (prestige) -based neighborhood of vertex v is a set of vertices that have large L_1 centrality (prestige) on the modified graph w.r.t. vertex v.

Value

L1centNB() returns an object of class L1centNB. It is a list of numeric vectors. The length of the list is equivalent to the number of vertices in the graph g, and the names of the list are vertex names. Each component of the list is a numeric vector whose length is equivalent to the number of vertices in the graph g. Specifically, the ith component of the list is a vector of the L_1 centrality of each vertex, for the modified graph g w.r.t. the ith vertex (if mode = "centrality") or the L_1 prestige of each vertex, for the modified graph g w.r.t. the ith vertex (if mode = "prestige").

print.L1centNB() prints L_1 centrality or L_1 prestige values at the modified graph w.r.t. each vertex and returns them invisibly.

summary.L1centNB() returns an object of class table. It is a summary of the prominence values with the five-number summary, mean, and the Gini coefficient, at each modified graph.

Note

The function is valid only for connected graphs. If the graph is directed, it must be strongly connected.

References

S. Kang and H.-S. Oh. On a notion of graph centrality based on L_1 data depth. *Journal of the American Statistical Association*, 121(553): 400–412, 2026a.

S. Kang and H.-S. Oh. L_1 prominence measures for directed graphs. *The American Statistician*, 80(2): 301–309, 2026b.

See Also

`L1cent()` for L_1 centrality/prestige, `L1centLOC()` and `L1centEDGE()` internally uses `L1centNB()`.

Examples

```
NB <- L1centNB(MCUmovie, vertex_weight = igraph::V(MCUmovie)$worldwidegross)
# Top 6 L1 centrality-based neighbors of "Iron Man"
utils::head(sort(NB$"Iron Man", decreasing = TRUE))
```

MCUmovie *Marvel Cinematic Universe Movie Network*

Description

Network between 32 movies from the Marvel Cinematic Universe (MCU) that were released between 2008 and 2023. Each movie represents one vertex.

An edge between movies i and j is formed if there is at least one cast in common. Denoting the set of casts of movie i as A_i, the weight of this edge is given as $(|A_i \cap A_j|/|A_i \cup A_j|)^{-1}$, where $|\cdot|$ denotes the cardinality of a set.

Usage

```
data(MCUmovie)
```

Format

An undirected, connected, and (edge) weighted `igraph` graph object with 32 vertices and 278 edges.

Vertex attributes:

- 'name': name of the movie. e.g., *Guardians of the Galaxy Vol. 3*.
- 'worldwidegross': worldwide gross in USD. Archived from IMDb on Nov. 3rd, 2023.
- 'year': release year of the movie.

Edge attribute: 'weight'. Given as a dissimilarity between two vertices. See the description above.

Source

IMDb: https://www.imdb.com

References

G. Choi and H.-S. Oh. Heavy-snow transform: A new method for graph signals. Manuscript, 2021.

S. Kang and H.-S. Oh. On a notion of graph centrality based on L_1 data depth. *Journal of the American Statistical Association*, 121(553): 400–412, 2026.

`rokassembly21` *Republic of Korea's 21st National Assembly Bill Cosponsorship Network*

Description

Network between 317 members of the Republic of Korea's 21st National Assembly (May 30th, 2020–May 29th, 2024). Each member of the assembly represents one vertex.

An edge between two members is formed if there is at least one cosponsored bill. The weight of this edge is given as 1/(number of cosponsored bills between two members) during the first 40 months of the 21st assembly (Jun. 2020–Sep. 2023).

Usage

```
data(rokassembly21)
```

Format

An undirected, connected, and (edge) weighted `igraph` graph object with 317 vertices and 47,657 edges.

Vertex attributes:

- 'name': Pseudonyms of each member. They are in the format of the party's initial character, followed by a random number (e.g., D4). Each party's initial character is:
  - 'D': Democratic Party of Korea.
  - 'P': People Power Party.
  - 'J': Justice Party.
  - 'O': Others (Basic Income Party, Hope of Korea, The Progressive Party, Transition Korea).
- 'party': Factor with 7 levels. Denotes the political party of each member as of Sep. 2023. Note that independent members are assigned to their original party.
- 'gender': Factor with 2 levels. 'M' (male) or 'F' (female).
- 'nelect': Number of legislative terms in the assembly for each member. Ranges from 1 to 6.
- 'district': Indicates if each member is a district representative (`TRUE`) or a proportional representative (`FALSE`).
- 'full': Indicates if each member was in the assembly for the first 40 months. `TRUE` for the members in the office for all 40 months. Members who started their term via by-election, resigned, or lost their seat for any reason during the 40 months are coded as `FALSE`.
- 'nbill': Number of bills cosponsored by each member.

Edge attribute: 'weight'. Given as a dissimilarity between two vertices. See the description above.

Source

The National Assembly of the Republic of Korea

- Bill information: https://likms.assembly.go.kr/bill/main.do
- Member information: https://open.assembly.go.kr/portal/assm/search/memberSchPage.do

References

S. Kang and H.-S. Oh. On a notion of graph centrality based on L_1 data depth. *Journal of the American Statistical Association*, 121(553): 400–412, 2026.

Index

∗ **datasets**
MCUmovie, 21
rokassembly21, 22

base::plot(), *4*, *17*
base::print(), *10*
base::summary(), *10*

Gini *(*Heterogeneity*)*, 4
graphics::lines(), *4*
graphics::par(), *4*, *17*
graphics::segments(), *17*
graphics::text(), *17*
group_reduce, 2
group_reduce(), *12, 13*

Heterogeneity, 4, *8*

igraph::betweenness(), *8*
igraph::closeness(), *8*
igraph::degree(), *8*
igraph::distances(), *7*
igraph::eigen_centrality(), *8*
igraph::plot.common, *10*
igraph::plot.igraph(), *10*

L1cent, *5*, 5
L1cent(), *4*, *10*, *13*, *16*, *18*, *21*
L1centEDGE, 9
L1centEDGE(), *8*, *21*
L1centGROUP, 11
L1centGROUP(), *4*, *8*
L1centLOC, *5*, 13
L1centLOC(), *8*, *10*, *21*
L1centMDS, 16
L1centMDS(), *8*
L1centNB, *5*, 19
L1centNB(), *8*, *10*, *15, 16*
L1centrality-package, *8*
L1pres *(*L1cent*)*, 5
L1presGROUP *(*L1centGROUP*)*, 11
L1presLOC *(*L1centLOC*)*, 13
L1presNB *(*L1centNB*)*, 19
L1prestige *(*L1cent*)*, 5
Lorenz_plot *(*Heterogeneity*)*, 4

MASS::isoMDS(), *18*
MCUmovie, 21

plot.L1cent *(*L1cent*)*, 5
plot.L1centEDGE *(*L1centEDGE*)*, 9
plot.L1centLOC *(*L1centLOC*)*, 13
plot.L1centMDS *(*L1centMDS*)*, 16
print.L1cent *(*L1cent*)*, 5
print.L1centEDGE *(*L1centEDGE*)*, 9
print.L1centGROUP *(*L1centGROUP*)*, 11
print.L1centLOC *(*L1centLOC*)*, 13
print.L1centMDS *(*L1centMDS*)*, 16
print.L1centNB *(*L1centNB*)*, 19

rokassembly21, 22

stats::cmdscale(), *18*
summary.L1cent *(*L1cent*)*, 5
summary.L1centEDGE *(*L1centEDGE*)*, 9
summary.L1centLOC *(*L1centLOC*)*, 13
summary.L1centNB *(*L1centNB*)*, 19