Package: giniCI (via r-universe)

June 3, 2026

Type Package

Title Gini-Based Composite Indicators

Version 0.1.3

Maintainer Viet Duong Nguyen <viet-duong.nguyen@outlook.com>

Description An implementation of Gini-based weighting approaches in constructing composite indicators, providing functionalities for normalization, aggregation, and ranking comparison.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports DescTools (>= 0.99.57), ggplot2 (>= 3.5.1), ggrepel (>= 0.9.6), ggpubr (>= 0.6.0)

Suggests R.rsp

VignetteBuilder R.rsp

URL https://github.com/novidu/giniCI

BugReports https://github.com/novidu/giniCI/issues

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev libx11-dev zlib1g-dev

Repository https://novidu.r-universe.dev

Date/Publication 2026-01-04 08:09:13 UTC

RemoteUrl https://github.com/novidu/ginici

RemoteRef HEAD

RemoteSha 02a45099182c421e815c4ea989b08478ca311b47

Contents

`bli` *OECD Better Life Index Indicators*

Description

This dataset includes 11 well-being indicators for 36 countries spanning the years 2014 to 2017. The indicators are derived from the OECD Better Life Index, which measures 11 topics deemed essential by the OECD for assessing material living conditions and quality of life.

Usage

```
bli
```

Format

A data frame with 144 rows and 13 variables:

COUNTRY Country.

YEAR Year.

HO_HISH Housing expenditure: percentage of housing costs in households gross adjusted disposable income.

IW_HADI Household income: average amount of money that a household earns per year, after taxes.

JE_EMPL Employment rate: percentage of people, aged 15 to 64, currently in a paid job.

SC_SNTWS Quality of support network: percentage of people who believe they can rely on their friends in case of need.

ES_EDUA Educational attainment: percentage of people, aged 25 to 64, having at least an upper-secondary (high school) degree.

EQ_AIRP Air pollution: average concentration of particulate matter (PM2.5) in the air.

CG_VOTO Voter turnout: percentage of registered voters who voted during recent elections.

HS_LEB Life expectancy: average number of years a person can expect to live.

SW_LIFS Life satisfaction: average self-evaluation of life satisfaction, on a scale from 0 to 10.

PS_REPH Homicide rate: average number of reported homicides per 100,000 people.

WL_TNOW Leisure and personal care: average number of hours per day spent on leisure and personal care, including sleeping and eating.

Source

OECD (2024). Better Life Index (Editions 2014, 2015, 2016, and 2017), *OECD Social and Welfare Statistics (database)*. Accessed on December 06, 2024.

---

giniCI *Computing Gini-based Composite Indicators*

Description

Compute a composite index with weighting schemes based on the Gini coefficient of constituent indicators, with options for aggregation methods and horizontal variability adjustment.

Usage

```
giniCI(inds, method = c("equal", "gini", "reci"),
       agg = c("ari", "geo"), hv = TRUE, ci.pol = c("pos", "neg"),
       time = NULL, ref.time = NULL, only.ci = FALSE)
```

Arguments

inds	a matrix or data frame of indicators to be aggregated.
method	weighting method to be used. See 'Details'.
agg	aggregation function to be used, with "ari" denoting the weighted arithmetic mean and "geo" denoting the weighted geometric mean.
hv	a logical value indicating whether horizontal variability adjustment should be applied.
ci.pol	a character value indicating the polarity of composite index. Use "pos" if increasing values of the composite index correspond to positive variations of the phenomenon (e.g., socio-economic developments). Otherwise, use "neg" if increasing values of the composite index correspond to to negative variations of the phenomenon (e.g., vulnerability and poverty).
time	a vector of temporal factors for indicators. The length of time must equal the number of rows in inds. If NULL, the input indicators are treated as cross-sectional data.
ref.time	a value denoting the reference time for weighting. If provided, weights will be derived using only observations at the reference time.
only.ci	a logical value indicating whether only the composite index should be returned.

Details

The defaut method is "equal" that produces equal weights $1/n$ where n is the number of indicators. For methods "gini" (Gini-based weighting) and "reci" (reciprocal Gini-based weighting), weights are defined based on the Gini coefficient of indicators. Let G_i be the Gini coefficient of the i-th indicator, the weights by methods "gini" and "reci" are respectively computed as $w_i^{\text{gini}} = \frac{G_i}{\sum_{i=1}^{n} G_i}$ and $w_i^{\text{reci}} = \frac{1/G_i}{\sum_{i=1}^{n} 1/G_i}$.

Temporal factors can be applied to methods "gini" and "reci". If either time or ref.time is NULL, the weighting process is run on all observations. If both time and ref.time are not NULL, only observations at the reference time are used for weight computation.

When aggregating the indicators, the aggregate score for the j-th unit is computed by applying the chosen aggregation function with the obtained weights to values in the j-th row. If hv = TRUE, horizontal variability adjustment is executed by introducing a penalty for units with unbalanced values among dimensions. The penalty for the j-th unit is defined as the the index of dispersion (variance-to-mean ratio) of values in the j-th row. If ci.pol = "pos", the penalties is subtracted from the aggregate scores to form the composite index. If ci.pol = "neg" the penalties is added to the aggregate scores to form the composite index.

Value

A list containing the following components:

ci	the composite index.
w	the weights assigned.
pen	the horizontal variability penalties (if hv = TRUE).

If only.ci = TRUE, the function will return only the composite index.

Note

Methods "gini" and "reci" require non-negative indicators for the calculation of Gini coefficients. In addition, option hv = TRUE cannot be used if any row contains negative values. Therefore, it may be necessary to use normalize to scale the indicators to non-negative ranges before computing the composite index.

A Gini coefficient of zero occurs when the indicators are constant or do not change over the reference time. If a zero Gini coefficient is obtained for for one or more indicators, method "gini" returns the corresponding weights as zero while method "reci" cannot be applied.

Author(s)

Viet Duong Nguyen, Chiara Gigliarano, Mariateresa Ciommi

References

Gini, C. (1914). Sulla misura della concentrazione e della variabilita dei caratteri. *Atti del Reale Istituto Veneto di Scienze, Lettere ed Arti*, 62(5), 1203–1248.

Mazziotta, M., & Pareto, A. (2016). On a Generalized Non-compensatory Composite Index for Measuring Socio-economic Phenomena. *Social Indicators Research*, 127, 983–1003.

Ciommi, M., Gigliarano, C., Emili, A., Taralli, S., & Chelli, F. M. (2017). A new class of composite indicators for measuring well-being at the local level: An application to the Equitable and Sustainable Well-being (BES) of the Italian Provinces. *Ecological Indicators*, 76, 281–296.

See Also

normalize, rankComp.

Examples

```
data(bli)

# Indicator polarity
bli.pol = c("neg", "pos", "pos", "pos", "pos", "neg",
            "pos", "pos", "pos", "neg", "pos")

# Goalpost normalization without using time factors
bli.norm <- normalize(inds = bli[, 3:13], method = "goalpost",
                      ind.pol = bli.pol)

# Goalpost normalization using time factors and a reference time
bli.norm.2014 <- normalize(inds = bli[, 3:13], method = "goalpost",
                           ind.pol = bli.pol, time = bli$YEAR,
                           ref.time = 2014)

# Adjusted Mazziotta-Pareto index
bli.ampi <- giniCI(bli.norm, ci.pol = "pos")
bli.ampi$ci

# Gini-based weighted arithmetic mean with reference time
bli.gini <- giniCI(bli.norm.2014, method = "gini", ci.pol = "pos",
                   time = bli$YEAR, ref.time = 2014)
bli.gini$ci
bli.gini$w

# Reciprocal Gini-based weighted geometric mean with reference time
bli.reci <- giniCI(bli.norm.2014, method = "reci", agg = "geo",
                   ci.pol = "pos", time = bli$YEAR, ref.time = 2014)
bli.reci$ci
bli.reci$w
```

`normalize` *Indicator Normalization*

Description

Perform normalization based on indicators' polarity.

Usage

```
normalize(inds, method = c("min-max", "goalpost"), ind.pol,
          gp.range = c(70, 130), time = NULL, ref.time = NULL,
          ref.value = NULL)
```

Arguments

`inds`	a numeric vector, matrix, or data frame which provides indicators to be normalized.

method	normalization method to be used. See 'Details'.
ind.pol	a character vector whose elements can be "pos" (positive) or "neg" (negative), indicating the polarity of indicators. An indicator's polarity is the sign of the relation between the indicator and the phenomenon to be measured.
gp.range	a vector of the form c(a,b) giving the normalization range for method "goalpost". The default value is c(70,130).
time	a vector of temporal factors for input indicators. The length of time must equal the number of rows in inds. If NULL, the input data will be treated as cross-sectional.
ref.time	a value denoting the reference time for normalization. See 'Details'.
ref.value	a vector containing reference values for indicators to facilitate the interpretation of results, required by method "goalpost". When normalizing each indicator, their reference values will be mapped to the midpoint of gp.range. See 'Details'.

Details

By default, each indicator x is normalized by method "min-max" with the formulas

$$\tilde{x}_i^+ = \frac{x_i - \inf_x}{\sup_x - \inf_x},$$

or

$$\tilde{x}_i^- = 1 - \tilde{x}_i^+,$$

where $\sup_x$ and $\inf_x$ are respectively the superior and inferior values of the indicator. The former formula is applied to indicators with positive polarity while the latter one is used for those with negative polarity.

If either time or ref.time is NULL, the superior and inferior values are respectively the maximum and minimum values of x. If both time and ref.time are not NULL, the superior and inferior values are respectively the maximum and minimum values of x observed at the reference time. In other words, if time is not provided or provided without specifying a value for ref.time, the input data will be treated as cross-sectional.

For method "goalpost", a vector of reference values for indicators is required. If not specified by users (ref.value = NULL), these values are automatically set #' to the indicator means for cross-sectional data or to the indicator means at the reference time for longitudinal data.

Method "goalpost" computes two goalposts for normalization as $\text{gp\_min}_x = \text{ref}_x - \Delta$ and $\text{gp\_max}_x = \text{ref}_x + \Delta$, where ref_x is the reference value of x and $\Delta = (\sup_x - \inf_x)/2$. Indicators with positive polarity are rescaled using the formula

$$\tilde{x}_i^+ = \frac{x_i - \text{gp\_min}_x}{\text{gp\_max}_x - \text{gp\_min}_x}(b - a) + a,$$

while indicators with negative polarity are rescaled using the formula

$$\tilde{x}_i^- = a + b - \tilde{x}_i^+.$$

If an indicator follows a symmetric probability distribution and its reference value is set to the mean, the normalized values will theoretically remain in the range $[a, b]$. In other cases, the normalized values may extend beyond gp.range.

Value

An object of class "data.frame" containing normalized indicators.

Author(s)

Viet Duong Nguyen, Chiara Gigliarano, Mariateresa Ciommi

References

Mazziotta, M., & Pareto, A. (2016). On a Generalized Non-compensatory Composite Index for Measuring Socio-economic Phenomena. *Social Indicators Research*, 127, 983–1003.

See Also

giniCI.

Examples

```
# Generate data samples
set.seed(1)
df1 <- data.frame(X1 = rnorm(100, 0, 5),
                  X2 = runif(100, 1, 10),
                  X3 = rpois(100, 10))
set.seed(1)
df2 <- data.frame(X1 = rnorm(300, 0, 5),
                  X2 = runif(300, 1, 10),
                  X3 = rpois(300, 10),
                  time = rep(c(2020:2022), rep(100,3)))

# Min-max normalization
df1.mm <- normalize(inds = df1,
                    ind.pol = c("pos", "neg", "pos"))
summary(df1.mm)
df2.mm <- normalize(inds = df2[, 1:3],
                    ind.pol = c("pos", "neg", "pos"),
                    time = df2[, 4], ref.time = 2020)
summary(df2.mm)

# Goalpost normalization
df1.gp <- normalize(inds = df1, method = "goalpost",
                    ind.pol = c("pos", "neg", "pos"))
summary(df1.gp)
df2.gp <- normalize(inds = df2[, 1:3], method = "goalpost",
                    ind.pol = c("pos", "neg", "pos"),
                    time = df2[, 4], ref.time = 2020)
summary(df2.gp)
```

rankComp *Ranking Comparison*

Description

Perform a ranking comparison between two indices.

Usage

```
rankComp(ref, alt, highest.first = TRUE, id = NULL, time = NULL)
```

Arguments

ref	a numeric vector of reference index values.
alt	a numeric vector of alternative index values.
highest.first	a logical value indicating whether the highest value gets ranking #1. If FALSE, the lowest value gets ranking #1.
id	a vector of unit identifiers.
time	a vector of temporal factors.

Value

An object of classes "rankComp" and "data.frame" containing the following columns:

id	the unit identifiers (if provided).
time	the temporal factors (if provided).
ref.rank	the ranking based on the reference index.
alt.rank	the ranking based on the alternative index.
shift	the ranking shifts between two indices.

Author(s)

Viet Duong Nguyen, Chiara Gigliarano, Mariateresa Ciommi

See Also

summary.rankComp, rankScatterPlot, rankShiftPlot, rankRankPlot.

Examples

```
data(bli)

# Goalpost normalization
bli.pol = c("neg", "pos", "pos", "pos", "pos", "neg",
            "pos", "pos", "pos", "neg", "pos")
bli.norm.2014 <- normalize(inds = bli[, 3:13], method = "goalpost",
                           ind.pol = bli.pol, time = bli$YEAR,
                           ref.time = 2014)

# Composite indices
ci.gini <- giniCI(bli.norm.2014, method = "gini",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)
ci.reci <- giniCI(bli.norm.2014, method = "reci", agg = "geo",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)

# Ranking comparison
ci.comp <- rankComp(ci.gini, ci.reci, id = bli$COUNTRY, time = bli$YEAR)
print(ci.comp)
summary(ci.comp)
```

`rankRankPlot` *Rank-rank Plot*

Description

Generate rank-rank plots for ranking comparison.

Usage

```
rankRankPlot(object, id.col = FALSE, p.size = 1.5, p.nudge = 0.05,
             s.width = 0.5,  lab.size = 3.88,
             max.overlaps = 10, max.tick = 50,
             ref.lab = "Reference ranking", alt.lab = "Alternative ranking",
             y.lab = NULL, combine = FALSE, nr = NULL, nc = NULL)
```

Arguments

`object`	an object of class `"rankComp"`, usually, an output of a call to `rankComp`.
`id.col`	a logical value indicating whether the rank-rank segment should be colored by unit identifiers. It is not recommended if having more than 20 units.
`p.size`	size of segment endpoints.
`p.nudge`	horizontal adjustment value to nudge the starting position of labels.
`s.width`	line width for rank-rank segments.
`lab.size`	label size value.

max.overlaps	a value to exclude the label if it has too many overlaps. The default value is 10. Set max.overlaps = Inf to always show all labels.
max.tick	a positive integer to control the maximum number of axis ticks. The default value is 50. Set max.tick equal to or greater than the number of rankings to display all rankings on the axis.
ref.lab	name of the reference index.
alt.lab	name of the alternative index.
y.lab	label of the y-axis.
combine	a logical value indicating whether to generate a grid that combines plots from different time factors (If object$time is not NULL).
nr	(optional) number of rows in the plot grid.
nc	(optional) number of columns in the plot grid.

Value

A plot comparing two rankings connected by segments. In case object$time is not NULL, a list of plots for different time factors and the combined grid (if combine = TRUE) will be returned. The function does not print the return if it is assigned to an object. Use print with the storing object to generate the plot.

Author(s)

Viet Duong Nguyen, Chiara Gigliarano, Mariateresa Ciommi

See Also

rankComp, rankScatterPlot, rankShiftPlot.

Examples

```
data(bli)

# Goalpost normalization
bli.pol = c("neg", "pos", "pos", "pos", "pos", "neg",
            "pos", "pos", "pos", "neg", "pos")
bli.norm.2014 <- normalize(inds = bli[, 3:13], method = "goalpost",
                           ind.pol = bli.pol, time = bli$YEAR,
                           ref.time = 2014)

# Composite indices
ci.gini <- giniCI(bli.norm.2014, method = "gini",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)
ci.reci <- giniCI(bli.norm.2014, method = "reci", agg = "geo",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)

# Ranking comparison plots
ci.comp <- rankComp(ci.gini, ci.reci, id = bli$COUNTRY, time = bli$YEAR)
```

```
rankScatterPlot(ci.comp)$'2014'
rankShiftPlot(ci.comp)$'2015'
rankRankPlot(ci.comp)$'2016'

# Storing and printing
p.scatter <- rankScatterPlot(ci.comp, combine = TRUE, max.overlaps = 20)
print(p.scatter$'2017') # or: print(p.scatter[[4]])
print(p.scatter$'comb') # or: print(p.scatter[[5]])
```

`rankScatterPlot` *Rank Scatter Plot*

Description

Generate rank scatter plots for ranking comparison.

Usage

```
rankScatterPlot(object, p.col = "black", p.size = 1.5, p.shape = 19,
                lab = TRUE, lab.col = "red", lab.size = 3.88,
                ref.line = TRUE, max.overlaps = 10, max.tick = 50,
                ref.lab = "Reference ranking", alt.lab = "Alternative ranking",
                combine = FALSE, nr = NULL, nc = NULL)
```

Arguments

`object`	an object of class `"rankComp"`, usually, an output of a call to `rankComp`.
`p.col`	point color code. See 'Color Specification' in `par`.
`p.size`	point size value.
`p.shape`	point shape value. See 'pch values' in `points`.
`lab`	a logical value indicating whether identifier labels should be assigned to scatter points. If `object$id` is `NULL`, the points will be labeled by the row index.
`lab.col`	color code for labels. See 'Color Specification' in `par`.
`lab.size`	label size value.
`ref.line`	a logical value indicating whether a 45-degree reference line should be added to the plot.
`max.overlaps`	a value to exclude the label if it has too many overlaps. The default value is `10`. Set `max.overlaps = Inf` to always show all labels.
`max.tick`	a positive integer to control the maximum number of axis ticks. The default value is `50`. Set `max.tick` equal to or greater than the number of rankings to display all rankings on the axis.
`ref.lab`	name of the reference index.
`alt.lab`	name of the alternative index.
`combine`	a logical value indicating whether to generate a grid that combines plots from different time factors (If `object$time` is not `NULL`).
`nr`	(optional) number of rows in the plot grid.
`nc`	(optional) number of columns in the plot grid.

Value

A scatter plot displaying unit rankings for two indices. In case `object$time` is not `NULL`, a list of plots for different time factors and the combined grid (if `combine = TRUE`) will be returned. The function does not print the return if it is assigned to an object. Use `print` with the storing object to generate the plot.

Author(s)

Viet Duong Nguyen, Chiara Gigliarano, Mariateresa Ciommi

See Also

`rankComp`, `rankShiftPlot`, `rankRankPlot`.

Examples

```
data(bli)

# Goalpost normalization
bli.pol = c("neg", "pos", "pos", "pos", "pos", "neg",
            "pos", "pos", "pos", "neg", "pos")
bli.norm.2014 <- normalize(inds = bli[, 3:13], method = "goalpost",
                           ind.pol = bli.pol, time = bli$YEAR,
                           ref.time = 2014)

# Composite indices
ci.gini <- giniCI(bli.norm.2014, method = "gini",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)
ci.reci <- giniCI(bli.norm.2014, method = "reci", agg = "geo",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)

# Ranking comparison plots
ci.comp <- rankComp(ci.gini, ci.reci, id = bli$COUNTRY, time = bli$YEAR)
rankScatterPlot(ci.comp)$'2014'
rankShiftPlot(ci.comp)$'2015'
rankRankPlot(ci.comp)$'2016'

# Storing and printing
p.scatter <- rankScatterPlot(ci.comp, combine = TRUE, max.overlaps = 20)
print(p.scatter$'2017') # or: print(p.scatter[[4]])
print(p.scatter$'comb') # or: print(p.scatter[[5]])
```

`rankShiftPlot` *Rank Shift Plot*

Description

Generate rank shift plots for ranking comparison.

Usage

```
rankShiftPlot(object, p.cols = c("black", "red"), p.shapes = c(1, 8),
              p.sizes = c(1.5, 1.5), s.col = "black",
              s.type = 1, s.width = 0.5, max.tick = 50,
              ref.lab = "Reference ranking", alt.lab = "Alternative ranking",
              y.lab = "Ranking", combine = FALSE, nr = NULL, nc = NULL)
```

Arguments

object	an object of class "rankComp", usually, an output of a call to rankComp.
p.cols	a vector with two elements denoting the color codes for reference and alternative positions. See 'Color Specification' in par.
p.shapes	a vector with two elements denoting the shapes for reference and alternative positions. See 'pch values' in points.
p.sizes	a vector with two elements denoting the sizes for reference and alternative positions.
s.col	color code for rank shift segments. See 'Color Specification' in par.
s.type	line type for rank shift segments. See 'Line Type Specification' in par.
s.width	line width for rank shift segments.
max.tick	a positive integer to control the maximum number of axis ticks. The default value is 50. Set max.tick equal to or greater than the number of rankings to display all rankings on the axis.
ref.lab	name of the reference index.
alt.lab	name of the alternative index.
y.lab	label of the y-axis.
combine	a logical value indicating whether to generate a grid that combines plots from different time factors (If object$time is not NULL).
nr	(optional) number of rows in the plot grid.
nc	(optional) number of columns in the plot grid.

Value

A plot displaying shifts in ranking between two indices. In case object$time is not NULL, a list of plots for different time factors and the combined grid (if combine = TRUE) will be returned. The function does not print the return value if it is assigned to an object. Use print with the storing object to produce the plot.

Author(s)

Viet Duong Nguyen, Chiara Gigliarano, Mariateresa Ciommi

See Also

rankComp, rankScatterPlot, rankRankPlot.

Examples

```
data(bli)

# Goalpost normalization
bli.pol = c("neg", "pos", "pos", "pos", "pos", "neg",
            "pos", "pos", "pos", "neg", "pos")
bli.norm.2014 <- normalize(inds = bli[, 3:13], method = "goalpost",
                           ind.pol = bli.pol, time = bli$YEAR,
                           ref.time = 2014)

# Composite indices
ci.gini <- giniCI(bli.norm.2014, method = "gini",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)
ci.reci <- giniCI(bli.norm.2014, method = "reci", agg = "geo",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)

# Ranking comparison plots
ci.comp <- rankComp(ci.gini, ci.reci, id = bli$COUNTRY, time = bli$YEAR)
rankScatterPlot(ci.comp)$'2014'
rankShiftPlot(ci.comp)$'2015'
rankRankPlot(ci.comp)$'2016'

# Storing and printing
p.scatter <- rankScatterPlot(ci.comp, combine = TRUE, max.overlaps = 20)
print(p.scatter$'2017') # or: print(p.scatter[[4]])
print(p.scatter$'comb') # or: print(p.scatter[[5]])
```

---

summary.rankComp *Summarizing Ranking Comparison*

Description

Summary method for class "rankComp" and print method for class "summary.rankComp".

Usage

```
## S3 method for class 'rankComp'
summary(object, n.pick = 10L, n.q = 10L, ...)

## S3 method for class 'summary.rankComp'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

object	an object of class "rankComp", usually, an output of a call to rankComp.
n.pick	a positive integer specifying the number of units considered to form the top/bottom based on the alternative index. The default value is 10.

n.q	a positive integer specifying the n-quantiles considered to compute quantile rankings. The default value is 10 (deciles).
...	further arguments passed to or from other methods.
x	an object of class "summary.rankComp", usually, a output of a call to summary.rankComp.
digits	number of significant digits to use when printing.

Details

summary.rankComp provides details on the ranking comparison between the reference and the alternative indices stored in the object of class "rankComp". print.summary.rankComp prints summary information using a smart digit format for the components.

Value

An object of class "summary.rankComp" which is a list of components:

par	a vector storing the values of n.pick and n.q.
n.unit	the number of ranked units (by temporal factors if available).
shift.stats	a data frame with rows presenting the summary statistics of ranking shifts: minimum, first quartile, median, mean, third quartile, and maximum.
asr	a data frame giving the average shift in ranking (ASR) $$\mathrm{ASR} = \frac{1}{m}\sum_{i=1}^{m}|\mathrm{rank}_i^{\mathrm{alt}} - \mathrm{rank}_i^{\mathrm{ref}}|,$$ where m is the number of units considered. The rows present the ASR for all units, and for the top and the bottom units based on the alternative index ranking.
per	a data frame giving the percentage of equal rankings (PER) $$\mathrm{PER} = 100 \times \frac{1}{m}\sum_{i=1}^{m}\{\mathrm{rank}_i^{\mathrm{alt}} = \mathrm{rank}_i^{\mathrm{ref}}\},$$ where m is the number of units considered. The rows present the PER for all units, and for the top and the bottom units based on the alternative index ranking.
asq	the average shift in quantile ranking (by temporal factors if available). This value is similar to the ASR for all units, but using the quantile ranking of two indices.

For shift.stats, asr, and per, multiple columns will be generated according to temporal factors if object$time is not NULL.

Author(s)

Viet Duong Nguyen, Chiara Gigliarano, Mariateresa Ciommi

References

Mariani, F., Ciommi, M., & Recchioni, M. C. (2024). Two in One: A New Tool to Combine Two Rankings Based on the Voronoi Diagram. *Social Indicators Research*, 175, 989–1005.

See Also

rankComp.

Examples

```
data(bli)

# Goalpost normalization
bli.pol = c("neg", "pos", "pos", "pos", "pos", "neg",
            "pos", "pos", "pos", "neg", "pos")
bli.norm.2014 <- normalize(inds = bli[, 3:13], method = "goalpost",
                           ind.pol = bli.pol, time = bli$YEAR,
                           ref.time = 2014)

# Composite indices
ci.gini <- giniCI(bli.norm.2014, method = "gini",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)
ci.reci <- giniCI(bli.norm.2014, method = "reci", agg = "geo",
                  ci.pol = "pos", time = bli$YEAR, ref.time = 2014,
                  only.ci = TRUE)

# Ranking comparison
ci.comp <- rankComp(ci.gini, ci.reci, id = bli$COUNTRY, time = bli$YEAR)
print(ci.comp)
summary(ci.comp)
```

Index