giniCI: Gini-based Composite Indicators

Version: 0.1.0

V. D. Nguyen, C. Gigliarano, and M. Ciommi

February 2025

1 Introduction

The `giniCI` package is designed for constructing composite indicators using Gini-based weighting approaches (Ciommi et al., 2017). These methods benchmark a sample of units in a multidimensional context by assuming that the relative importance of each individual dimension depends solely on its distribution across the units. From this point of view, weights are assigned to individual indicators based on their distributional dispersion, captured by Gini coefficients. Consequently, the obtained weighting scheme emphasizes the differences arising from indicators that exhibit greater heterogeneity (or greater homogeneity, in the case of reciprocal weighting) across the sample, favoring units with the highest values in those indicators during aggregation.

The package offers a suite of functions to facilitate the development of composite indicators to support data-driven decision-making. It includes tools for (1) variable normalization, (2) composite index computation, and (3) ranking comparison. The workflow begins with data standardization, followed by indicator aggregation using weighted arithmetic or geometric means, and finishes with rank-shift analysis to compare composite indices. By automating these processes and integrating visualization tools, `giniCI` simplifies composite index construction while enhancing interpretability and usability in socio-economic studies. The package is particularly handy for researchers and decision makers seeking a systematic, transparent, and statistically grounded method for multidimensional measurement, enabling robust cross-unit comparisons and actionable policy insights.

2 Data Normalization

Individual indicators are often measured in different units and may have different polarities (relationships) with the phenomenon being measured. The `normalize()` function can be applied for ensuring that all variables are transformed into a comparable scale and aligned in the same direction of contribution to the composite index.

2.1 Method `"min-max"`

This is the default method that rescales each indicator x to a $[0, 1]$ range based on their "inferior" and "superior" values using two formulas

$$\tilde{x}_i^+ = \frac{x_i - \inf_x}{\sup_x - \inf_x}, \text{ or} \tag{1}$$

$$\tilde{x}_i^- = 1 - \tilde{x}_i^+. \tag{2}$$

Formula (1) is applied to indicators with positive polarity, while Formula (2) is used for those with negative polarity. For cross-sectional data, the inferior and superior values are respectively the minimum and maximum values of the indicators.

If the data includes a temporal dimension (e.g., panel datasets) and a reference time is specified by users, the superior and inferior values are respectively defined as the maximum and minimum values observed at that reference time. The purpose of selecting the reference time is to normalize observations from other periods (including future data) against a chosen baseline, enabling meaningful comparisons over time. An example of data normalization with the method `"min-max"` is provided below. Let us generate two 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)))
```

Here, `df1` is cross-sectional data and `df2` is longitudinal data with a temporal variable `time` including three factors `2020`, `2021`, and `2022`. If the reference

time for `df2` is set to `2020`, the `"min-max"` scaling results for two datasets can be obtained as follows:

```
df1.mm <- normalize(inds = df1,
                    ind.pol = c("pos", "neg", "pos"))
df2.mm <- normalize(inds = df2[, 1:3],
                    ind.pol = c("pos", "neg", "pos"),
                    time = df2[, 4], ref.time = 2020)
##  Warning message:
##  12 negative value(s) generated in columns(s): 1, 2, 3
```

As can be seen in the case of `df2.mm`, normalization using a fixed reference time may result in negative scaled values. This occurs when observations in other periods fall outside the interval formed by the reference time's minimum and maximum values. When these values are generated, a warning message will be displayed to alert users, as they can affect downstream calculations (e.g., Gini-based weighting approaches that require non-negative inputs). If the temporal variable is present but no reference time is specified, the `normalize()` function will treat the data as cross-sectional.

2.2 Method `goalpost`

The other method for transformation is `"goalpost"` (Mazziotta and Pareto, 2016), which requires a reference value ref_x for each indicator x to facilitate the interpretation of results. If the reference values are not provided by users, they are automatically assigned as follows: for cross-sectional data, ref_x defaults to the indicator's mean; and for longitudinal data, ref_x defaults to the indicator's mean at the reference time (if specified).

For each indicator, two goalposts are established as

$$\text{gp\_min}_x = \text{ref}_x - \Delta, \text{ and} \tag{3}$$

$$\text{gp\_max}_x = \text{ref}_x + \Delta, \tag{4}$$

where $\Delta = (\sup_x - \inf_x)/2$. Using a normalization range $[a, b]$ (default: $[70, 130]$), indicators with positive polarity are normalized using the formula

$$\tilde{x}_i^+ = \frac{x_i - \text{gp\_min}_x}{\text{gp\_max}_x - \text{gp\_min}_x}(b - a) + a, \tag{5}$$

whereas those with negative polarity are normalized using the formula

$$\tilde{x}_i^- = a + b - \tilde{x}_i^+. \tag{6}$$

As a result, the reference value ref_x is mapped to the midpoint $(a+b)/2$ of the normalization range. Below is the code snippet of applying `"goalpost"` scaling to the datasets `df1` and `df2`:

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

If an indicator follows a symmetric probability distribution and its reference value is set to the mean, the normalized values will theoretically remain in $[a, b]$. Otherwise, in cases of asymmetric distributions or reference value deviation from the mean, the normalized values may extend beyond the normalization range.

3 Composite Index Computation

`giniCI()` is the core function of the package that constructs composite indicators by aggregating multiple dimensions into a single index, using weighting schemes derived from Gini coefficients to reflect the relative importance of indicators. Unlike traditional aggregation approaches, the function incorporates an option for horizontal variability adjustment, ensuring the synthetic index reflects both inter-unit inequality and intra-unit imbalance.

3.1 Weighting Methods

The function `giniCI()` supports three weighting methods for computing composite indices. The default method is `"equal"`, where all indicators are assigned an equal weight $w_i^{\text{equal}} = 1/n$ with n as the number of dimensions. Two other methods compute weights for each indicator i based on its Gini coefficient G_i. The Gini-based weighting method (`"gini"`) assigns weights proportionally to the Gini coefficients using the formula

$$w_i^{\text{gini}} = \frac{G_i}{\sum_{k=1}^{n} G_k}. \tag{7}$$

This approach assigns greater weights to indicators with more imbalanced distributions. Conversely, the reciprocal Gini-based weighting method (`"reci"`) defines weights using the inverse of Gini coefficients, computed as

$$w_i^{\text{reci}} = \frac{1/G_i}{\sum_{k=1}^{n} 1/G_k}. \tag{8}$$

By this way, the method ensures that indicators with more homogeneous distributions have a greater influence on the composite index.

Temporal factors can be applied to the methods `"gini"` and `"reci"`. If the data includes a temporal variable and a reference time is specified by users, only observations at the reference time are used to compute the weights. If time factors are not provided or provided without any reference time, the weighting process is run on all observations.

Note: Both methods `"gini"` and `"reci"` require indicators with non-negative values to calculate Gini coefficients. In cases negative values are present, it is recommended to consider scaling the data with the `normalize()` function before using these methods. If the Gini coefficient of an indicator is zero (i.e., the indicator is constant), the method `"gini"` assigns a weight of zero to that indicator, while the method `"reci"` is non-applicable.

3.2 Aggregation and Horizontal Variability Adjustment

For each observation in the dataset, the function `giniCI()` computes the composite index score C_j for the j-th row r_j as

$$C_j = f(r_j; w) \pm D_{r_j}, \tag{9}$$

where f is a function that aggregate r_j using a weighting scheme w, and D_{r_j} is the index of dispersion (variance-to-mean ratio) of r_j. The weight components in w are derived from one of three methods introduced in the previous section (`"equal"`, `"gini"`, or `"reci"`). Whereas, the aggregation function can be selected as either the weighted arithmetic mean (`agg = "ari"`), which works for all numeric values, or the weighted geometric mean (`agg = "geo"`), which requires strictly positive indicators.

The index of dispersion D_{r_j} is added or subtracted from the composite score if the option for horizontal variability adjustment is chosen (`hv = TRUE`). This mechanism introduces a penalty for units based on the imbalance among indicator values (Muro et al., 2011). The sign of the adjustment depends on the relationship between the composite index and the phenomenon being measured. The penalty is subtracted if increasing composite scores correspond to positive variations of the phenomenon (e.g., socio-economic developments). Otherwise, the penalty is added if increasing composite scores correspond to negative variations of the phenomenon (e.g., vulnerability or poverty). Since the index of dispersion should be computed only for data measured on a ratio scale, the horizontal variability adjustment option cannot be applied to units containing negative indicator values.

3.3 Examples

Below is an example code demonstrating the usage of `giniCI()` to construct composite indicators from the dataset `bli`. This dataset includes 11 well-being indicators for 36 countries spanning the years 2014 to 2017. To compute the adjusted Mazziotta-Pareto index (Mazziotta and Pareto, 2016), the normalization and weighting processes need to be performed without incorporating temporal factors:

```
data(bli)
bli.pol <- c("neg", "pos", "pos", "pos", "pos", "neg",
             "pos", "pos", "pos", "neg", "pos")
bli.norm <- normalize(bli[, 3:13], method = "goalpost",
                      ind.pol = bli.pol)
bli.ampi <- giniCI(bli.norm, ci.pol = "pos")
```

Note that longitudinal data can be treated as cross-sectional when constructing composite indices, as in the case of the adjusted Mazziotta-Pareto index. However, for longitudinal data, it is recommended to use a specific reference time during normalization and weighting. This ensures that normalization goalposts and weights remain consistent (even when future data is added), enabling robust comparisons over time. Below is the code snippet for computing composite indicators using the Gini-based weighted arithmetic aggregation and the reciprocal Gini-based weighted geometric aggregation, with 2014 as the reference year:

```
bli.norm.2014 <- normalize(inds = bli[, 3:13],
                           method = "goalpost",
                           ind.pol = bli.pol,
                           time = bli$YEAR, ref.time = 2014)
bli.gini <- giniCI(bli.norm.2014,
                   method = "gini", ci.pol = "pos",
                   time = bli$YEAR, ref.time = 2014)
bli.reci <- giniCI(bli.norm.2014,
                   method = "reci", agg = "geo", ci.pol = "pos",
                   time = bli$YEAR, ref.time = 2014)
```

4 Ranking Comparison

4.1 Ranking Shift Analysis

The ranking comparison functionality in the `giniCI` package is designed to evaluate shifts between a reference composite index and an alternative composite index. The function `rankComp()` calculates the rankings for each unit based on both indices, computes the shift by subtracting the alternative rank from the reference rank, and outputs a data frame with these details. A positive shift indicates an improvement in unit performance, while a negative shift suggests the opposite. The comparison result between two Gini-based composite indices in Section 3.3 can be obtained by:

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

The function `summary.rankComp()` provides the summary method for the output from a call to `rankComp()`. Besides six summary statistics for the ranking shifts, it provides three key measures (by temporal factors if provided):

```
summary(ci.comp)
##  Number of ranked units:
##  2014 2015 2016 2017
##    36   36   36   36
##
##  Ranking shift summary statistics:
##         2014 2015 2016 2017
##  Min.     -3   -4   -6   -5
##  1st Q.   -1   -1   -1   -2
##  Median    0    0    0    0
##  Mean      0    0    0    0
##  3rd Q.    1    1    1    1
##  Max.      5    4    5    4
##
```

```
##  Average shift in ranking:
##             2014  2015  2016  2017
##  All units 1.333 1.278 1.111 1.611
##  Top 10    1.100 1.100 1.100 1.300
##  Bottom 10 1.600 1.200 0.400 1.000
##
##  Percentage of equal rankings:
##             2014  2015  2016  2017
##  All units 22.22 36.11 41.67 22.22
##  Top 10    30.00 40.00 40.00 30.00
##  Bottom 10 10.00 30.00 70.00 40.00
##
##  Average shift in 10-quantile ranking:
##     2014   2015   2016   2017
##   0.2222 0.3889 0.3889 0.3889
```

The average shift in ranking (ASR) quantifies the mean absolute difference between the rankings assigned by the reference and alternative indices. In parallel, the percentage of equal rankings (PER) indicates the proportion of units that receive identical rankings from both indices. The ASR and PER are measured for all units and for top/bottom-ranked units based on the alternative index (default: 10 units). Additionally, the average shift in quantile rankings (ASQ) provides a quantile-based assessment of ranking differences (default: 10-quantile), serving as an alternative to the ASR. Collectively, these metrics offer a comprehensive overview of the discrepancies between the two ranking systems (see Mariani et al. 2024).

4.2 Visualization Tools

The `giniCI` package offers three functions to generate ranking comparison graphs from the output of `rankComp`. These functions return a plot object (or a list of plot objects if temporal factors are present), which can be stored and printed. The plots are customizable, allowing users to adjust colors, sizes, shapes, and label displays to meet the desired results.

The function `rankScatterplot()` visualizes the relationship between two ranking systems using a two-dimensional scatter plot. Each dot's position on the horizontal and vertical axes corresponds to a unit's ranking according to the reference and alternative indices. A 45-degree reference line can be added for classifying ranking changes. Units located in the lower half-plane indicate an improvement in performance, while those in the upper half-plane

indicate a decline in performance. Units positioned on the reference line have identical rankings in both indices.

```
rankScatterPlot(ci.comp)$'2014'
```

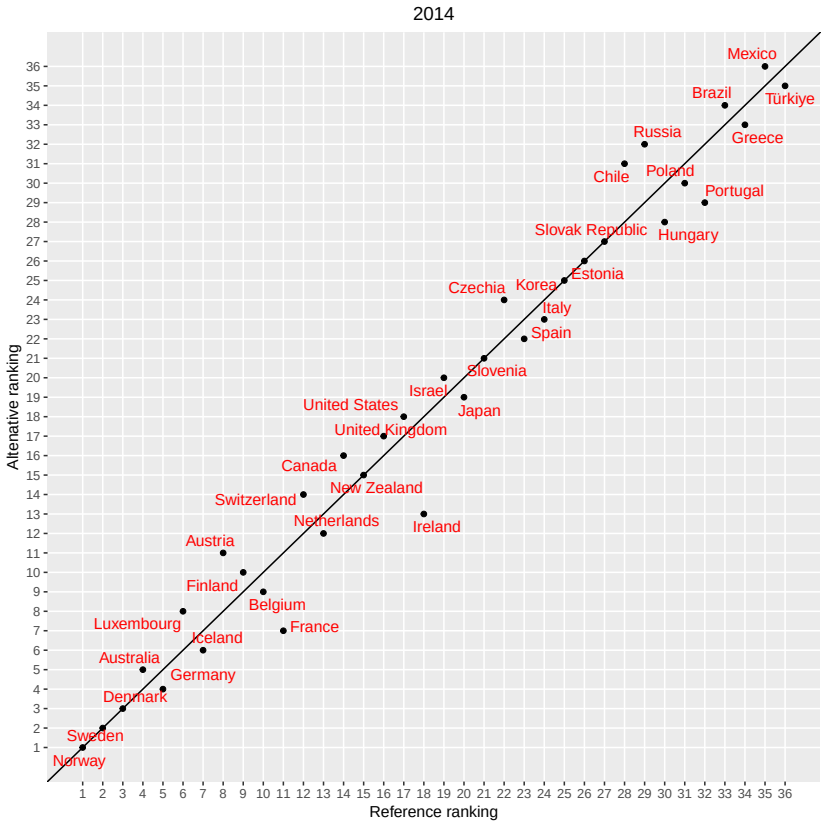

Figure 1: Rank scatter plot for two ranking systems in 2014

The function `rankShiftplot()` represents shifts in ranking by displaying each unit as a pair of vertically aligned points: the first point (default: black-bordered circle) corresponds to the unit's position in the reference ranking, and the second point (default: solid red circle) corresponds to its position in the alternative ranking. These points are connected by a line segment, enabling users to easily identify the direction and magnitude of ranking shifts. If the reference and alternative points for a unit overlap, it signifies that the unit's ranking remains unchanged.

```
rankShiftPlot(ci.comp)$'2015'
```

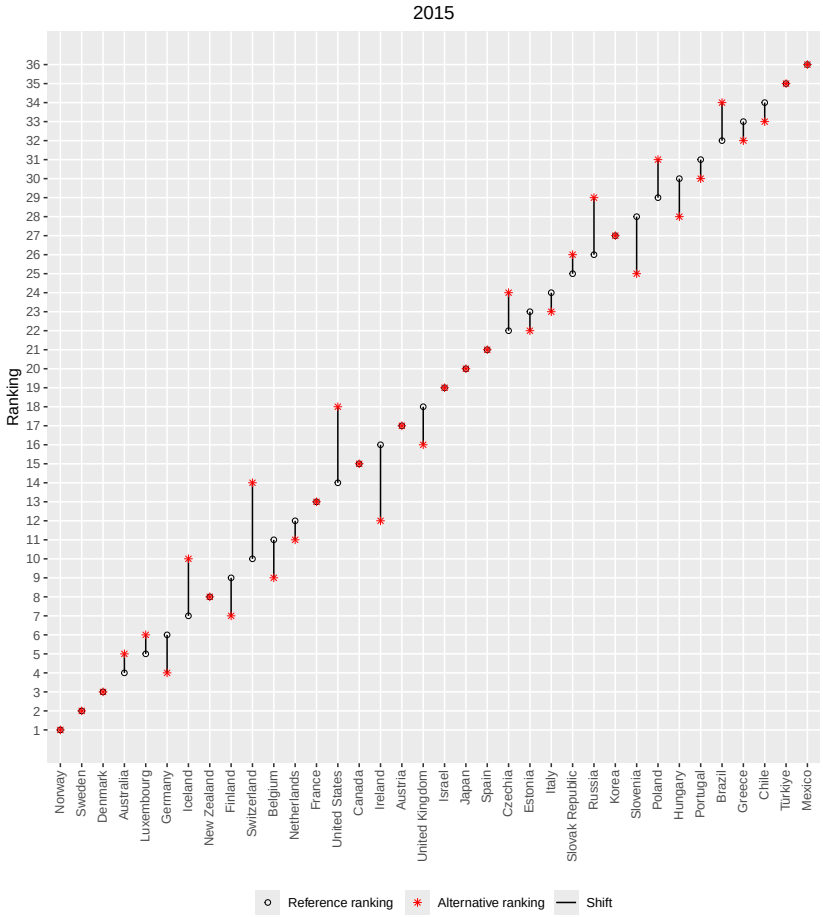

Figure 2: Rank shift plot for two ranking systems in 2015

The final function, `rankRankplot()`, arranges two ranking systems side by side and uses connecting lines to visualize how the position of each unit changes between them. Upward-sloping segments indicate an improvement in the alternative ranking compared to the reference ranking, while downward-sloping segments indicate a decline. The length of non-horizontal segments represent the magnitude of ranking shifts, with longer segments highlighting more substantial changes in position.

```
rankRankPlot(ci.comp)$'2016'
```

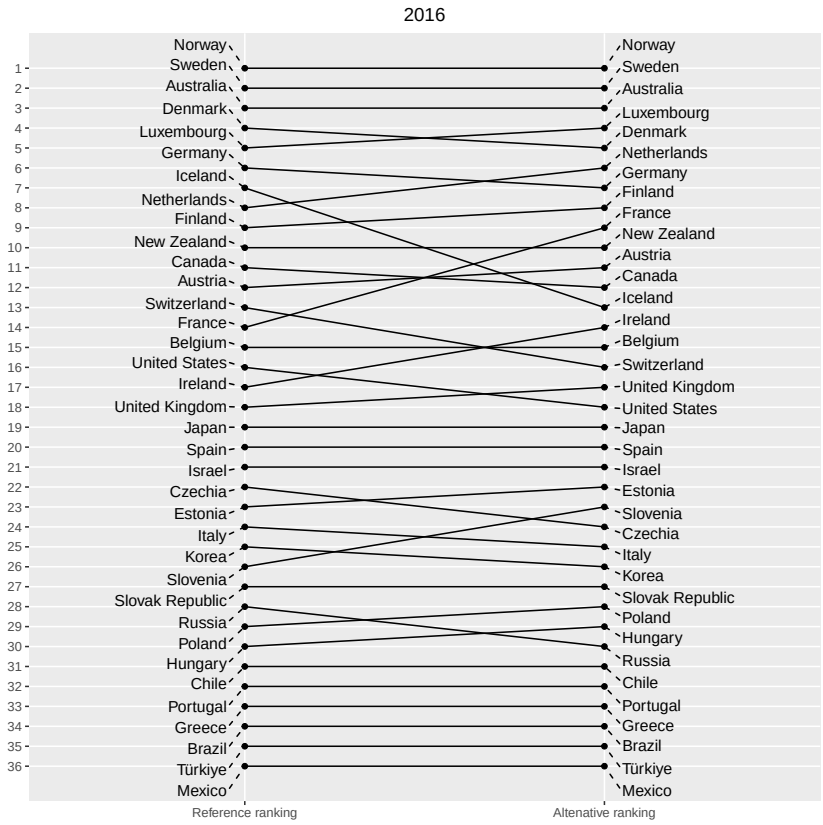

Figure 3: Rank-rank plot for two ranking systems in 2016

References

M. Ciommi, C. Gigliarano, A. Emili, S. Taralli, and F. M. Chelli. 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, 2017. doi: 10.1016/j.ecolind.2016.12.050.

F. Mariani, M. Ciommi, and M. C. Recchioni. Two in One: A New Tool to Combine Two Rankings Based on the Voronoi Diagram. *Social Indicators Research*, 175:989–1005, 2024. doi: 10.1007/s11205-023-03192-9.

M. Mazziotta and A. Pareto. On a Generalized Non-compensatory Com-

posite Index for Measuring Socio-economic Phenomena. *Social Indicators Research*, 127:983–1003, 2016. doi: 10.1007/s11205-015-0998-2.

P. D. Muro, M. Mazziotta, and A. Pareto. Composite Indices of Development and Poverty: An Application to MDGs. *Social Indicators Research*, 104: 1–18, 2011. doi: 10.1007/s11205-010-9727-z.