Package 'inequantiles'

May 29, 2026

Title Quantile-Based Inequality Indicators for Complex Survey Data

Version 0.1.0

Description Estimates quantile-based inequality indicators from complex survey data, including the quantile ratio index (QRI), quintile share Ratio (QSR), Palma ratio, and percentile ratios, together with the Gini coefficient. Influence functions are provided for linearization and variance estimation, along with a rescaled bootstrap for complex sampling designs. Estimation from grouped data is also supported. See Scarpa et al. (2025) <doi:10.1093/jssam/smaf024> for details.

License MIT + file LICENSE

URL https://silviascarpa.github.io/inequantiles/, https://github.com/silviascarpa/inequantiles/

BugReports https://github.com/silviascarpa/inequantiles/issues/

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Depends R (>= 3.5)

LazyData true

Imports Rdpack

RdMacros Rdpack

Suggests knitr, rmarkdown, ggplot2, scales, kableExtra, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Silvia Scarpa [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-4003-1621>), Stefan Sperlich [aut]

Maintainer Silvia Scarpa <silvia.scarpa@unimore.it>

Repository CRAN

Date/Publication 2026-05-29 12:00:02 UTC

Contents

csquantile . . . 2
gini_grouped . . . 4
if_gini . . . 5
if_qri . . . 7
if_quantile . . . 9
if_ratio_quantiles . . . 10
if_share_ratio . . . 12
inequantiles . . . 14
plot_inequality_curve . . . 16
qri . . . 19
qri_grouped . . . 21
quantile_grouped . . . 23
ratio_quantiles . . . 26
rescaled_bootstrap . . . 27
share_ratio . . . 32
superpop_qri . . . 33
synthouse . . . 35

Index **39**

`csquantile` *Quantile estimation with complex sampling data*

Description

Computes quantiles for weighted or unweighted data, allowing for sampling weights and several interpolation types. The method extends the standard quantile definitions of Hyndman and Fan (1996) and Harrell and Davis (1982) estimator to the case of complex survey data by incorporating sampling weights into the cumulative distribution function and interpolation points, as proposed in Scarpa et al. (2025).

Usage

```
csquantile(y, weights = NULL, probs = seq(0, 1, 0.1), type = 4, na.rm = FALSE)
```

Arguments

`y`	Numeric vector of observations.
`weights`	Optional numeric vector of sampling weights; if `NULL` (default), all observations are equally weighted.
`probs`	Numeric vector of probabilities (default: `seq(0, 1, 0.1)`).
`type`	Quantile estimation type: integer `4`–`9` or `"HD"` for Harrell–Davis (default: `4`).
`na.rm`	Logical; if `TRUE`, missing values are removed before computing. Default: `FALSE`. Higher-level functions in this package handle `NA` removal before calling `csquantile()`, so the default is kept `FALSE` to avoid redundant filtering.

Details

Consider a random sample s of size n. Let $y_1, \ldots, y_n$ be the sample observations from a finite population, with order statistics $y_{(1)} \leq \ldots \leq y_{(n)}$ and corresponding sampling weights $w_1, \ldots, w_n$. Define the cumulative weights $W_j = \sum_{i \leq j} w_i$ and the total weight $W_n = \sum_{i=1}^{n} w_i$. The weighted quantile estimator is computed as a linear interpolation between adjacent order statistics:

$$\widehat{Q}(p) = y_{(k-1)} + (y_{(k)} - y_{(k-1)}) \frac{p - \widehat{r}_{k-1}}{\widehat{r}_k - \widehat{r}_{k-1}},$$

where $\widehat{r}_k$ denotes the estimated cumulative distribution function (the "plotting position"), and the order k is such that $W_{k-1} - m_{k-1} < W_n p < W_k - m_k$, with m_k determined by the interpolation method.

The table below summarizes the six interpolation types (4–9) extended from Hyndman and Fan (1996) to incorporate sampling weights, as described in Scarpa et al. (2025).

Type	**Estimator** $\widehat{r}_k$	**Interpolation** $\widehat{m}_k$	**Selection rule for** k
4	W_k / W_n	0	$W_{k-1} \leq W_n p < W_k$
5	$(W_k - \frac{1}{2} w_k) / W_n$	$w_k / 2$	$W_{k-1} - \frac{w_{k-1}}{2} \leq W_n p < W_k - \frac{w_k}{2}$
6	$W_k / (W_n + w_n)$	$w_n p$	$W_{k-1} \leq (W_n + w_n) p < W_k$
7	W_{k-1} / W_{n-1}	$w_k - w_n p$	$W_{k-2} \leq W_{n-1} p < W_{k-1}$
8	$(W_k - \frac{1}{3} w_k) / (W_n + \frac{w_n}{3})$	$\frac{w_k}{3} + \frac{w_n}{3} p$	$W_{k-1} - \frac{w_{k-1}}{3} \leq (W_n - \frac{w_n}{3}) p < W_k - \frac{w_k}{3}$
9	$(W_k - \frac{3}{8} w_k) / (W_n + \frac{1}{4} w_n)$	$\frac{3}{8} w_k + \frac{w_n}{4} p$	$W_{k-1} - \frac{3 w_{k-1}}{8} \leq (W_n + \frac{w_n}{4}) p < W_k - \frac{3 w_k}{8}$

The function supports several interpolation rules (types 4–9) and extends the quantile definitions in Hyndman and Fan (1996) to incorporate sampling weights. For unweighted data, the function returns the standard R quantiles.

The Harrell–Davis estimator ("HD") is extended to the weighted case as proposed in Kreutzmann (2018), by redefining the weighting coefficients $\widehat{\mathcal{W}}_j(p)$ for order statistics as:

$$\widehat{\mathcal{W}}_j(p) = b_{(W_j/W_n)}\{(W_n+w_n)p, W_n-(W_n+w_n)p+w_n\} - b_{(W_{j-1}/W_n)}\{(W_n+w_n)p, W_n-(W_n+w_n)p+w_n\},$$

where $b_x(a, b)$ denotes the incomplete beta function.

The resulting quantile estimator is $\widehat{Q}_{HD}(p) = \sum_{j \in s} \widehat{\mathcal{W}}_j(p) y_{(j)}$. For unweighted data, the function returns the Harrell-Davis quantile estimator.

Value

A named numeric vector of estimated quantiles corresponding to `probs`.

References

Harrell FE, Davis CE (1982). "A new distribution-free quantile estimator." *Biometrika*, **69**, 635–640.

Hyndman RJ, Fan Y (1996). "Sample quantiles in statistical packages." *The American Statistician*, **50**, 361–365.

Kreutzmann AK (2018). "Estimation of sample quantiles: challenges and issues in the context of income and wealth distributions." *AStA Wirtschafts-und Sozialstatistisches Archiv*, **12**, 245–270.

Scarpa S, Ferrante MR, Sperlich S (2025). "Inference for the quantile ratio inequality index in the context of survey data." *Journal of Survey Statistics and Methodology*. doi:10.1093/jssam/smaf024.

Examples

```
data(synthouse)
y <- synthouse$eq_income
w <- synthouse$weight

# Unweighted quantiles
csquantile(y, probs = c(0.25, 0.5, 0.75), type = 6)

# Weighted quantiles
csquantile(y, weights = w, probs = c(0.25, 0.5, 0.75), type = 6)

# Harrell-Davis estimator
csquantile(y, weights = w, probs = c(0.25, 0.5, 0.75), type = "HD")
```

`gini_grouped` *Gini Coefficient for Grouped Data*

Description

Computes the Gini coefficient from grouped income data based on linear interpolation of income shares.

Usage

```
gini_grouped(Y, freq)
```

Arguments

`Y`	Numeric vector of total amounts per group (e.g., total income per income class)
`freq`	Numeric vector of frequencies per group or class.

Details

Consider grouped data divided into J classes with known boundaries, observed frequencies $f_1, \ldots, f_J$ and total amounts $Y_1, \ldots, Y_J$. The Gini coefficient is approximated by linear interpolation of cumulative shares, as:

$$G \approx 1 - \sum_{j=1}^{J} (s_j + s_{j-1})(u_j - u_{j-1})$$

where:

- $p_j = f_j / \sum_{i=1}^{J} f_i$ is the population share of group j;
- $c_j = Y_j / \sum_{i=1}^{J} Y_i$ is the share of the variable of interest in group j;
- $s_j = \sum_{k=1}^{j} c_k$ is the cumulative share of the variable up to group j;
- $u_j = \sum_{k=1}^{j} p_k$ is the cumulative population share up to group j;
- $s_0 = u_0 = 0$ by convention.

This formula computes twice the area between the egalitarian line (perfect equality) and the Lorenz curve obtained by linearly interpolating the points (u_j, s_j). Since it assumes all observations within a group have identical values, it provides a *lower-bound* estimate of the true Gini coefficient, actual inequality may be larger (Jorda et al. 2021). The bias magnitude depends on the number of groups and how they are defined.

Value

A numeric value representing the estimated Gini coefficient on grouped data. The Gini coefficient ranges from 0 (perfect equality) to 1 (complete inequality). Note that it assumes equality within groups.

References

Jorda V, Sarabia JM, Jäntti M (2021). "Inequality measurement with grouped data: Parametric and non-parametric methods." *Journal of the Royal Statistical Society Series A: Statistics in Society*, **184**(3), 964–984.

See Also

`qri_grouped` for computing the quantile ratio index from grouped data.

Other grouped data functions: `qri_grouped()`, `quantile_grouped()`

Examples

```
income_freq <- c(1200, 1800, 1500, 800, 400, 20, 10)
income_tot <- c(18800, 16300, 44700, 33900, 21500, 22100, 98300)

gini_grouped(Y = income_tot, freq = income_freq)
```

---

if_gini *Influence Function for the Gini Coefficient*

Description

Computes the influence function for the Gini coefficient, useful for variance estimation and linearization in complex survey designs Langel and Tillé (2013).

Usage

```
if_gini(y, weights = NULL, na.rm = TRUE)
```

Arguments

y	Numeric vector of income or variable of interest.
weights	Numeric vector of sampling weights. If NULL (default), equal weights are assumed (simple random sampling).
na.rm	Logical. Should missing values be removed? Default is TRUE.

Details

The influence function for the Gini coefficient is computed using the linearization method, following Deville (1999) framework and as defined by Langel and Tillé (2013). The influence function for Gini is:

$$I(\widehat{G})_k = \frac{2W_k(y_k - \bar{Y}_k) + \hat{Y} - \hat{N}y_k - G(\hat{Y} + y_k\hat{N})}{\hat{N}\hat{Y}}$$

where:

- $W_k = \sum_{i=1}^{k} w_i$ is the cumulative sum of weights up to rank k
- $\bar{Y}_k = \frac{\sum_{l \in S} w_l y_l 1(W_l \leqslant W_k)}{W_k}$ is the weighted mean of values up to rank k
- $\hat{N} = \sum_i w_i$ is the total sum of weights
- $\hat{Y} = \sum_i w_i y_i$ is the weighted total of the variable
- G is the Gini coefficient estimate

Value

A numeric vector of the same length as y containing the influence function values for each observation, returned in the same order as the input y.

References

Deville J (1999). "Variance estimation for complex statistics and estimators: linearization and residual techniques." *Survey methodology*, **25**, 193–204.

Langel M, Tillé Y (2013). "Variance estimation of the Gini index: revisiting a result several times published." *Journal of the Royal Statistical Society Series A*, **176**, 521–540.

See Also

Other influence functions: if_qri(), if_quantile(), if_ratio_quantiles(), if_share_ratio()

Examples

```
data(synthouse)

eq <- synthouse$eq_income # Equivalized disposable income

# Simple example
z <- if_gini(eq)

# With weights
w <- synthouse$weight
z_weighted <- if_gini(y = eq, weights = w)
```

`if_qri` *Influence Function for the Quantile Ratio Index*

Description

Computes the influence function of the quantile ratio index (QRI) in the context of finite population for all observations, as defined in Scarpa et al. (2025), under simple and complex sampling. See Deville (1999) for an introduction to the definition of influence function in finite population theory.

Usage

```
if_qri(y, weights = NULL, type = 6, na.rm = TRUE)
```

Arguments

`y`	A numeric vector of data values
`weights`	A numeric vector of sampling weights (optional). If `NULL`, equal weights are assumed.
`type`	Quantile estimation type: integer `4`–`9` or `"HD"` for Harrell–Davis (default: `6`). See `csquantile`.
`na.rm`	Logical. Should missing values be removed? Default is `TRUE`.

Details

The influence function for the QRI is computed on each observation as

$$I(\widehat{QRI})_k = -\int_0^1 \frac{\left(\frac{\frac{p}{2}-\mathbf{1}(y_k\leq \widehat{Q}(p/2))}{\widehat{f}(\widehat{Q}(p/2))\,\widehat{N}}\right)\widehat{Q}(1-p/2) - \left(\frac{(1-\frac{p}{2})-\mathbf{1}(y_k\leq \widehat{Q}(1-p/2))}{\widehat{f}(\widehat{Q}(1-p/2))\,\widehat{N}}\right)\widehat{Q}(p/2)}{\widehat{Q}(1-p/2)^2}\,dp$$

where:

- $\widehat{Q}(p)$ is the weighted sample quantile of order p, computed using the internal function `csquantile()`,

- $\widehat{f}(\cdot)$ denotes the estimated income density function,
- $\widehat{N} = \sum_i w_i$ is the estimated population size, where w_i is the sampling weight associated to the i-th individual.

The density function $\widehat{f}(y)$ is estimated via a Gaussian kernel smoother:

$$\widehat{f}(y) = \frac{1}{\widehat{N}} \sum_{j \in s} w_j K\left(\frac{y - y_j}{h}\right) = \frac{1}{\widehat{N}\, h\sqrt{2\pi}} \sum_{j \in s} w_j \exp\left\{-\frac{(y - y_j)^2}{2h^2}\right\},$$

where $K(\cdot)$ is the Gaussian kernel.

The bandwidth is chosen as:

$$h = 0.79 \cdot \mathrm{IQR} \cdot \widehat{N}^{-1/5},$$

where IQR is the interquartile range of the weighted sample.

Value

A numeric vector of influence function values (one per observation), returned in the same order as the input `y`.

References

Deville J (1999). "Variance estimation for complex statistics and estimators: linearization and residual techniques." *Survey methodology*, **25**, 193–204.

Scarpa S, Ferrante MR, Sperlich S (2025). "Inference for the quantile ratio inequality index in the context of survey data." *Journal of Survey Statistics and Methodology*. doi:10.1093/jssam/smaf024.

See Also

`qri` for the QRI inequality indicator estimator, `csquantile` for weighted quantile estimation.

Other influence functions: `if_gini()`, `if_quantile()`, `if_ratio_quantiles()`, `if_share_ratio()`

Examples

```
# On synthetic data
eq_synth <- rlnorm(30, 9, 0.7)
IF_synth <- if_qri(y = eq_synth)

# On real data
data(synthouse)
eq <- synthouse$eq_income[1:30]
w <- synthouse$weight[1:30]
IF_qri <- if_qri(y = eq, weights = w, type = 6)
```

`if_quantile` *Influence Function for Quantiles*

Description

Computes the influence function of sample quantiles, allowing for both simple random sampling and complex survey designs with sampling weights, in the context of finite population. See Hampel et al. (1986) for an explanation of influence function and Deville (1999) for its definition in finite population theory.

Usage

```
if_quantile(y, weights = NULL, probs, type = 6, na.rm = TRUE)
```

Arguments

`y`	A numeric vector of data values
`weights`	A numeric vector of sampling weights (optional)
`probs`	A numeric value specifying the probability for the quantile (e.g., 0.5 for median)
`type`	Quantile estimation type: integer `4–9` or `"HD"` for Harrell–Davis (default: `6`). See `csquantile`.
`na.rm`	Logical, should missing values be removed? (default: TRUE)

Details

From the definition in Van der Vaart (2000) and (Osier 2009), the population influence function of the quantile $Q(p)$ is defined as:

$$IF(Q(p))_k = \frac{p - \mathbf{1}(y_k \leq Q(p))}{f(Q(p))\, N},$$

where $f(Q(p))$ is the population density function evaluated at the quantile and N is the population size.

In the sample, this is estimated as:

$$\widehat{IF}(Q(p))_k = \frac{p - \mathbf{1}(y_k \leq \widehat{Q}(p))}{\widehat{f}(\widehat{Q}(p))\, \widehat{N}},$$

where $\widehat{Q}(p)$ is the weighted sample quantile estimated by `csquantile()`, and $\widehat{N} = \sum_{i \in s} w_i$ is the estimated population size.

The density $\widehat{f}(y)$ is estimated using a Gaussian kernel density function:

$$\widehat{f}(y) = \frac{1}{\widehat{N}\, h\sqrt{2\pi}} \sum_{j \in s} w_j \exp\left\{ -\frac{(y - y_j)^2}{2h^2} \right\},$$

with bandwidth $h = 0.79 \cdot IQR \cdot \widehat{N}^{-1/5}$

Value

A numeric vector containing the estimated influence function values for each observation.

References

Hampel FR, Ronchetti E, Rousseeuw P, Stahel W (1986). *Robust statistics: the approach based on influence functions*. John Wiley & Sons.

Deville J (1999). "Variance estimation for complex statistics and estimators: linearization and residual techniques." *Survey methodology*, **25**, 193–204.

Van der Vaart AW (2000). *Asymptotic statistics*, volume 3. Cambridge University Press.

Osier G (2009). "Variance estimation for complex indicators of poverty and inequality using linearization techniques." *Survey Research Methods*, **3**, 167–195.

See Also

`csquantile` for weighted quantile estimation.

Other influence functions: `if_gini()`, `if_qri()`, `if_ratio_quantiles()`, `if_share_ratio()`

Examples

```
# On synthetic data
eq_synth <- rlnorm(30, 9, 0.7)
IF_synth <- if_quantile(y = eq_synth, probs = 0.3)

# On real data
data(synthouse)
eq <- synthouse$eq_income[1:30] # First 30 observations
w <- synthouse$weight[1:30]
IF_quantile <- if_quantile(y = eq, weights = w, type = 6, probs = 0.5)
```

---

if_ratio_quantiles *Influence Function for the Ratio Between Quantiles*

Description

Computes the influence function of the ratio between two quantiles (e.g., P90/P10) for all observations in the sample. See Deville (1999) and Osier (2009) for the definition of influence functions in finite population theory.

Usage

```
if_ratio_quantiles(
  y,
  weights = NULL,
  type = 6,
  prob_numerator = 0.9,
  prob_denominator = 0.1,
  na.rm = TRUE
)
```

Arguments

`y`	A numeric vector of data values.
`weights`	A numeric vector of sampling weights (optional). If `NULL`, equal weights are assumed.
`type`	Quantile estimation type: integer `4`–`9` or `"HD"` for Harrell–Davis (default: `6`). See `csquantile`.
`prob_numerator`	Numeric in $(0, 1)$; order of the quantile at the numerator (default: `0.90`).
`prob_denominator`	Numeric in $(0, 1)$; order of the quantile at the denominator (default: `0.10`).
`na.rm`	Logical; remove missing values before computing? Default: `TRUE`.

Details

The influence function for the ratio $\widehat{R} = \widehat{Q}(p_n)/\widehat{Q}(p_d)$ is derived via the delta method applied to the quantile influence function of Deville (1999):

$$I\left(\frac{\widehat{Q}(p_n)}{\widehat{Q}(p_d)}\right)_k = \frac{\left(\frac{p_n - \mathbf{1}(y_k \le \widehat{Q}(p_n))}{\widehat{f}(\widehat{Q}(p_n))\,\widehat{N}}\right)\widehat{Q}(p_d) - \left(\frac{p_d - \mathbf{1}(y_k \le \widehat{Q}(p_d))}{\widehat{f}(\widehat{Q}(p_d))\,\widehat{N}}\right)\widehat{Q}(p_n)}{\widehat{Q}(p_d)^2}$$

where:

- $\widehat{Q}(p)$ is the weighted sample quantile of order p, computed via `csquantile`,
- p_n and p_d are the orders of the numerator and denominator quantiles, respectively,
- $\widehat{f}(\cdot)$ is the estimated density function,
- $\widehat{N} = \sum_i w_i$ is the estimated population size.

The density $\widehat{f}(y)$ is estimated via a Gaussian kernel:

$$\widehat{f}(y) = \frac{1}{\widehat{N}\,h\sqrt{2\pi}} \sum_{j \in s} w_j \exp\left\{-\frac{(y - y_j)^2}{2h^2}\right\}$$

with bandwidth $h = 0.79 \cdot \mathrm{IQR} \cdot \widehat{N}^{-1/5}$.

Value

A numeric vector of influence function values, one per observation.

References

Deville J (1999). "Variance estimation for complex statistics and estimators: linearization and residual techniques." *Survey methodology*, **25**, 193–204. Osier G (2009). "Variance estimation for complex indicators of poverty and inequality using linearization techniques." *Survey Research Methods*, **3**, 167–195.

See Also

`ratio_quantiles`, `csquantile`

Other influence functions: `if_gini()`, `if_qri()`, `if_quantile()`, `if_share_ratio()`

Examples

```
# On synthetic data
set.seed(1)
eq_synth <- rlnorm(30, 9, 0.7)
IF_synth <- if_ratio_quantiles(y = eq_synth, prob_numerator = 0.80,
                               prob_denominator = 0.20)

# On survey data
data(synthouse)
eq <- synthouse$eq_income[1:30]
w  <- synthouse$weight[1:30]
IF_vals <- if_ratio_quantiles(y = eq, weights = w, type = 6)
```

`if_share_ratio` *Influence Function for Quantile-Based Share Ratios*

Description

Computes the linearized variable (influence function) for the quantile-based share ratio (QBSR) using the linearization approach of Deville (1999) and the derivation in Langel and Tillé (2011).

Usage

```
if_share_ratio(
  y,
  weights = NULL,
  type = 6,
  prob_numerator = 0.8,
  prob_denominator = 0.2,
  na.rm = TRUE
)
```

Arguments

y	A numeric vector of strictly positive values (e.g. income, wealth).
weights	A numeric vector of sampling weights. If NULL, all observations are equally weighted.
type	Quantile estimation type: integer 4–9 or "HD" for Harrell–Davis (default: 6). See csquantile.
prob_numerator	Numeric in $(0, 1)$; quantile order for the numerator (default: 0.80).
prob_denominator	Numeric in $(0, 1)$; quantile order for the denominator (default: 0.20).
na.rm	Logical; remove missing values before computing? Default: TRUE.

Details

Langel and Tillé (2011) derived the influence function for the quintile share ratio, which generalises to any QBSR. Define p_n and p_d as the quantile orders for the numerator and denominator, respectively. The linearized variable is:

$$I(\widehat{QBSR})_k = \frac{y_k - I(\widehat{Y}_{p_n})_k}{\widehat{Y}_{p_d}} - \frac{(\widehat{Y} - \widehat{Y}_{p_n})\, I(\widehat{Y}_{p_d})_k}{\widehat{Y}^2_{p_d}}$$

where the influence function of the partial total $\widehat{Y}_p = \sum_{j \in s} w_j y_j \mathbf{1}(y_j \leq \widehat{Q}(p))$ is:

$$I(\widehat{Y}_p)_k = p\,\widehat{Q}(p) - \big(\widehat{Q}(p) - y_k\big)\mathbf{1}(y_k \leq \widehat{Q}(p))$$

and $\widehat{Y} = \sum_{j \in s} w_j y_j$ is the estimated total.

Value

A numeric vector of the same length as y containing the linearized variable $\widehat{z}_k$ for each observation.

References

Deville J (1999). "Variance estimation for complex statistics and estimators: linearization and residual techniques." *Survey methodology*, **25**, 193–204. Langel M, Tillé Y (2011). "Statistical inference for the quintile share ratio." *Journal of Statistical Planning and Inference*, **141**, 2976–2985.

See Also

share_ratio, csquantile

Other influence functions: if_gini(), if_qri(), if_quantile(), if_ratio_quantiles()

Examples

```
data(synthouse)
eq <- synthouse$eq_income
w  <- synthouse$weight

# QSR influence function (default: p_n = 0.80, p_d = 0.20)
z <- if_share_ratio(eq, weights = w)

# Palma influence function (p_n = 0.90, p_d = 0.40)
z_palma <- if_share_ratio(eq, weights = w,
                          prob_numerator = 0.90, prob_denominator = 0.40)
```

`inequantiles` *Quantile-based inequality indicators*

Description

Estimates one or more quantile-based inequality indicators simultaneously — QRI, quantile-based share ratio (QSR, Palma, or custom), percentile ratio — together with the Gini coefficient as a widely used benchmark. When standard errors are requested, all indicators are evaluated on the same bootstrap replicates, ensuring full comparability.

Usage

```
inequantiles(
  y,
  weights = NULL,
  indicators = "all",
  se = FALSE,
  type = 6,
  na.rm = TRUE,
  M = 100,
  B = 200,
  seed = NULL,
  data = NULL,
  strata = NULL,
  psu = NULL,
  N_h = NULL,
  m_h = NULL,
  verbose = TRUE
)

## S3 method for class 'inequantiles'
print(x, digits = 4, ...)
```

Arguments

y	A numeric vector of strictly positive values (e.g. income, wealth, expenditure).
weights	A numeric vector of sampling weights. If NULL, all observations are equally weighted.
indicators	Character vector specifying which indicators to compute. Use "all" (default) for all, or any subset of "qri", "qsr", "palma", "ratio_quantiles", "gini". "qsr" (quintile share ratio) and "palma" (Palma index) are special cases of share_ratio. "ratio_quantiles" computes the P90/P10.
se	Logical; if TRUE, standard errors are estimated via the rescaled bootstrap; see rescaled_bootstrap. Requires data and strata (see below).
type	Quantile estimation type: integer 4–9 or "HD" for Harrell–Davis (default: 6). See csquantile.
na.rm	Logical; remove missing values before computing? Default: TRUE.
M	Integer; number of quantile-ratio grid points for the QRI (default: 100). Only used when the QRI is estimated; see qri.
B	Integer; number of bootstrap replicates (default: 200). Only used when se = TRUE.
seed	Integer; random seed for reproducibility. Only used when se = TRUE.
data	A data frame containing the survey design variables (strata, PSU). Required when se = TRUE.
strata	Character string; name of the stratification column in data. Required when se = TRUE.
psu	Character string; name of the PSU column in data. Required for two-stage complex designs.
N_h	Optional named numeric vector of stratum population sizes for the finite population correction. See rescaled_bootstrap.
m_h	Optional vector of bootstrap sample sizes per stratum. Defaults to the Rao-Wu formula. See rescaled_bootstrap.
verbose	Logical; if TRUE (default), displays a progress bar during bootstrap iterations.
x	An object of class "inequantiles".
digits	Integer; number of decimal places for rounding (default: 4).
...	Further arguments passed to or from other methods.

Details

All quantile-based indicators are computed from the same specified csquantile type. When se = TRUE, a *single* bootstrap loop is run through the rescaled bootstrap method (see rescaled_bootstrap) and all indicators are evaluated on each replicate, so standard errors are based on identical resamples and are directly comparable.

The Gini coefficient is estimated following (Langel and Tillé 2013), equation 6, using a weighted formula based on cumulative weight sums.

Value

A list with components:

estimates	Numeric vector of point estimates of inequality indicators.
se	Numeric vector of standard errors, or NULL when se = FALSE.
B	Number of bootstrap replicates used, or NULL.
design	Sampling design type detected by the bootstrap, or NULL when se = FALSE.
call	The matched function call.

The argument x, invisibly.

See Also

qri, share_ratio, ratio_quantiles, rescaled_bootstrap

Other inequality indicators based on quantiles: plot_inequality_curve(), qri(), ratio_quantiles(), share_ratio(), superpop_qri()

Examples

```
data(synthouse)
eq <- synthouse$eq_income
w  <- synthouse$weight

# Point estimates only
inequantiles(eq, weights = w)

# Subset of indicators
inequantiles(eq, weights = w, indicators = c("qri", "palma"))



# With bootstrap standard errors (complex design)
inequantiles(eq, weights = w,
             se = TRUE, B = 50, seed = 42,
             data = synthouse, strata = "NUTS2",
             psu = "municipality",
             verbose = FALSE)
```

---

plot_inequality_curve *Plot the Inequality Curve*

Description

Plots the inequality curve $R(p) = Q(p/2)/Q(1-p/2)$ over $p \in [0, 1]$, from either sampling survey data or a parametric distribution. The shaded area between the curve and the line $R(p) = 1$ equals the QRI.

Usage

```
plot_inequality_curve(
  y = NULL,
  qfunction = NULL,
  qfun_args = list(),
  weights = NULL,
  M = 100,
  type = 6,
  na.rm = TRUE,
  shade = TRUE,
  add = FALSE,
  col = "steelblue",
  shade_col = NULL,
  lwd = 1.5,
  lty = 1,
  legend_qri = TRUE,
  label = NULL,
  xlab = "p",
  ylab = "R(p)",
  main = "Inequality curve"
)
```

Arguments

y	Numeric vector of strictly positive values (e.g. income). Provide either y (empirical mode) or qfunction (parametric mode), not both.
qfunction	A parametric quantile function, e.g. qlnorm, qweibull. Provide either qfunction or y, not both.
qfun_args	Named list of additional arguments passed to qfunction (e.g. if qfunction == qlnorm, then provide list(meanlog = 9, sdlog = 0.6)).
weights	Numeric vector of sampling weights. Only used in estimation mode. If NULL, all observations are equally weighted.
M	Integer; number of grid points for evaluating $R(p)$ (default: 200).
type	Quantile estimation type: integer 4–9 or "HD" for Harrell–Davis (default: 6). Only used in empirical mode. See csquantile.
na.rm	Logical; remove missing values? Default: TRUE.
shade	Logical; if TRUE (default), shades the area between $R(p)$ and the equality line $R(p) = 1$. The shaded area equals the QRI.
add	Logical; if TRUE, adds the curve to an existing plot without redrawing axes. Each successive call appends a new entry to the legend on the right of the plot.
col	Colour of the inequality curve (default: "steelblue").
shade_col	Colour for the shaded area. Defaults to a transparent version of col.
lwd	Line width (default: 1.5).
lty	Line type (default: 1).

legend_qri	Logical; if TRUE (default), maintains a legend outside the plot area on the right, showing the QRI for each curve. When add = TRUE the new curve is appended below the previous entries.
label	Character string; overrides the auto-generated legend label ("QRI = X.XXXX"). Useful for giving curves descriptive names. Default: NULL (auto-label).
xlab	x-axis label (default: "p").
ylab	y-axis label (default: "R(p)").
main	Plot title (default: "Inequality curve").

Details

The inequality curve $R(p)$ plots the ratio of symmetric quantiles around the median:

$$R(p) = \frac{Q(p/2)}{Q(1-p/2)}, \quad p \in [0,1],$$

against p. For a perfectly equal distribution $R(p) = 1$ for all p, and the curve coincides with the horizontal line at 1. The further the curve lies below the equality line, the more unequal the distribution. The QRI is the area between the equality line and the curve.

Boundary values $R(0) = 0$ and $R(1) = 1$ are set by convention (see Prendergast and Staudte (2018)).

Multiple curves can be overlaid by calling the function repeatedly with add = TRUE. The legend outside the plot accumulates an entry for each curve automatically.

Value

Beyond the plot, a named list with three elements:

p	Numeric vector of grid points in $[0,1]$.
Rp	Numeric vector of $R(p)$ values at each grid point.
qri	The estimated QRI (area between the equality line and the curve).

The list is returned invisibly, meaning it is not printed to the console when the function is called without assignment. Assign the output to a variable (e.g. out <- plot_inequality_curve(...)) to inspect it.

References

Prendergast LA, Staudte RG (2018). "A simple and effective inequality measure." *The American Statistician*, **72**, 328–343.

Scarpa S, Ferrante MR, Sperlich S (2025). "Inference for the quantile ratio inequality index in the context of survey data." *Journal of Survey Statistics and Methodology*. doi:10.1093/jssam/smaf024.

See Also

qri for the sample-based QRI estimator, superpop_qri for the theoretical QRI of a parametric distribution.

Other inequality indicators based on quantiles: inequantiles(), qri(), ratio_quantiles(), share_ratio(), superpop_qri()

Examples

```
# ------------------------------------------------------------------
# Parametric mode: single curve
# ------------------------------------------------------------------
plot_inequality_curve(
  qfunction = qlnorm,
  qfun_args = list(meanlog = 9, sdlog = 0.9),
  main = "Log-Normal inequality curve"
)

# ------------------------------------------------------------------
# Overlay multiple curves — legend accumulates automatically
# ------------------------------------------------------------------
plot_inequality_curve(
  qfunction = qlnorm,
  qfun_args = list(meanlog = 9, sdlog = 0.3),
  main  = "Log-Normal inequality curves",
  col   = "steelblue",
  label = "LogN(9, 0.3)"
)
plot_inequality_curve(
  qfunction = qlnorm,
  qfun_args = list(meanlog = 9, sdlog = 0.9),
  col   = "tomato", lty = 2, add = TRUE,
  label = "LogN(9, 0.9)"
)

# ------------------------------------------------------------------
# Empirical mode: survey data with sampling weights
# ------------------------------------------------------------------
data(synthouse)
out <- plot_inequality_curve(
  y       = synthouse$eq_income,
  weights = synthouse$weight,
  main    = "Inequality curve — synthouse"
)

# Inspect the returned list
out$qri          # estimated QRI
head(out$p)      # grid points
head(out$Rp)     # R(p) values
```

---

qri *Quantile Ratio Index*

Description

Computes the quantile ratio index (QRI) estimator for measuring inequality on simple and complex sampling data

Usage

```
qri(y, weights = NULL, M = 100, type = 6, na.rm = TRUE)
```

Arguments

y	A numeric vector of strictly positive values (e.g. income, wealth, expenditure).
weights	A numeric vector of sampling weights. If NULL, all observations are equally weighted.
M	Integer; number of quantile ratios to average (default: 100)
type	Quantile estimation type: integer 4–9 or "HD" for Harrell–Davis (default: 6). See csquantile.
na.rm	Logical; should missing values be removed? (default: TRUE)

Details

Consider a random sample s, where w_j, $j \in s$, defines the sampling weight associated to the j-th individual. The QRI estimator is defined as:

$$\widehat{QRI} = \frac{1}{M} \sum_{m=1}^{M} \left(1 - \frac{\widehat{Q}(p_{m/2})}{\widehat{Q}(1 - p_{m/2})} \right)$$

where $p_m = p_m = (m - 0.5)/M$ and $m = 1, \ldots, M$. The estimated quantiles $\widehat{Q}(p)$ are computed via the function csquantile(), which accounts for sampling weights and the specified quantile type. This allows $\widehat{QRI}$ to be used both for simple random samples and for complex survey data with design weights.

This index was proposed by Prendergast and Staudte (2018), and extended to survey data by Scarpa et al. (2025).

Value

A scalar numeric value representing the estimated inequality by the quantile ratio index (QRI).

References

Prendergast LA, Staudte RG (2018). "A simple and effective inequality measure." *The American Statistician*, **72**, 328–343.

Scarpa S, Ferrante MR, Sperlich S (2025). "Inference for the quantile ratio inequality index in the context of survey data." *Journal of Survey Statistics and Methodology*. doi:10.1093/jssam/smaf024.

See Also

qri_grouped for QRI estimation from grouped data, superpop_qri for the theoretical QRI of a parametric distribution, if_qri for the influence function used for linearization.

Other inequality indicators based on quantiles: inequantiles(), plot_inequality_curve(), ratio_quantiles(), share_ratio(), superpop_qri()

Examples

```
data(synthouse)

eq <- synthouse$eq_income # Income data

# Compute unweighted QRI with default type 6 quantile estimator
qri(y = eq)

# Consider the sampling weights and change quantile estimation type
w <- synthouse$weight
qri(y = eq, weights = w, type = 5)

# Compare QRI across macro-regions (NUTS1)
tapply(1:nrow(synthouse), synthouse$NUTS1, function(area) {
  qri(y = synthouse$eq_income[area],
      weights = synthouse$weight[area],
      type = 6)
})
```

---

qri_grouped *Quantile Ratio Index Estimator for Grouped Data*

Description

Computes the quantile ratio index (QRI) for measuring inequality from grouped frequency data using linear interpolation for quantile estimation. This function is intended to be used for administrative or tax data, which are very often in the form of grouped data. Therefore, sampling weights are not considered.

Usage

```
qri_grouped(
  freq,
  lower_bounds,
  upper_bounds,
  M = 100,
  midpoints = NULL,
  na.rm = TRUE
)
```

Arguments

freq	Numeric vector of class frequencies (counts). Must be non-negative.
lower_bounds	Numeric vector of lower class bounds.
upper_bounds	Numeric vector of upper class bounds.
M	Integer; number of quantile ratios to average (default: 100).

`midpoints`	Optional numeric vector of class midpoints. Used only as fallback when a quantile class has zero frequency.
`na.rm`	Logical; should missing values in frequencies be removed? (default: TRUE)

Details

Consider grouped data divided into L classes with known boundaries and observed frequencies $f_1, \ldots, f_L$. The QRI estimator for grouped data is approximated as:

$$QRI \approx \frac{1}{M} \sum_{m=1}^{M} \left(1 - \frac{\widetilde{Q}(p_m/2)}{\widetilde{Q}(1 - p_m/2)} \right)$$

where:

- $p_m = (m - 0.5)/M$ for $m = 1, \ldots, M$
- $\widetilde{Q}(p)$ denotes the p-th quantile computed from grouped data using linear interpolation (see `quantile_grouped`)
- M is the number of quantile ratios to average (default: 100)

The quantiles $\widetilde{Q}(p)$ are computed via `quantile_grouped()`, which uses linear interpolation within classes and automatically handles open-ended classes (with `-Inf` or `Inf` bounds).

The QRI ranges from 0 (perfect equality) to 1 (maximum inequality). The index measures inequality by averaging the relative differences between symmetric quantiles below and above the median, across the entire distribution.

Value

A scalar numeric value representing the estimated inequality by the quantile ratio index (QRI) for grouped data.

Comparison with Microdata QRI

When individual-level (microdata) are available, use `qri` instead, which provides more accurate estimates. The grouped data version `qri_grouped` should be used when only frequency distributions are available, such as in published statistical tables or administrative aggregates.

The grouped QRI will generally approximate the microdata QRI well when:

- Classes are sufficiently narrow
- The distribution within classes is approximately uniform
- Sample sizes within classes are adequate

References

Prendergast LA, Staudte RG (2018). "A simple and effective inequality measure." *The American Statistician*, **72**, 328–343.

See Also

qri for QRI estimation with microdata. superpop_qri for QRI computation on parametric distributions

Other grouped data functions: gini_grouped(), quantile_grouped()

Examples

```
# Basic example with closed classes
income_freq <- c(120, 180, 150, 80, 40, 20, 10)
income_lower <- c(0, 15000, 30000, 45000, 60000, 80000, 100000)
income_upper <- c(15000, 30000, 45000, 60000, 80000, 100000, 150000)

qri_grouped(income_freq, income_lower, income_upper)

# Example with open-ended classes (Italian MEF-style data)
wage_freq <- c(150, 200, 180, 220, 180, 50, 15, 5)
wage_lower <- c(-Inf, 0, 10000, 15000, 26000, 55000, 75000, 120000)
wage_upper <- c(0, 10000, 15000, 26000, 55000, 75000, 120000, Inf)

# Compute QRI (automatically handles open classes)
qri_grouped(wage_freq, wage_lower, wage_upper)
```

---

quantile_grouped *Quantile Estimator for Grouped (Binned) Data*

Description

Computes quantiles from grouped frequency data using linear interpolation within the quantile class.

Usage

```
quantile_grouped(
  freq,
  lower_bounds,
  upper_bounds,
  probs = 0.5,
  midpoints = NULL
)
```

Arguments

freq	Numeric vector of class frequencies (counts). Must be non-negative.
lower_bounds	Numeric vector of lower class bounds. Must be strictly increasing.

`upper_bounds`	Numeric vector of upper class bounds. Must be strictly increasing and greater than corresponding `lower_bounds`.
`probs`	Numeric vector of probabilities (between 0 and 1) for which to compute the quantiles. Default is 0.5 (median).
`midpoints`	Optional numeric vector of class midpoints. Used only as fallback when a quantile class has zero frequency. If `NULL`, the midpoint is computed as the arithmetic mean of class bounds.

Details

Consider grouped data divided into J classes with known boundaries. Let:

- L_j be the lower bound of the j-th quantile class
- U_j be the upper bound of the j-th quantile class
- $h_j = U_j - L_j$ be the j-th quantile class width
- C_{j-1} be the cumulative frequency up to the previous class
- f_j be the frequency within the quantile class j
- $N = \sum_{i=1}^{k} f_i$ be the total frequency

The quantile class for the p-th quantile is the first class j such that:

$$j = min\{i : C_i \geq pN\}$$

.

The p-th quantile $Q(p)$ is then estimated by linear interpolation within the quantile class:

$$\widetilde{Q(p)} = L_j + \frac{(pN - C_{j-1})}{f_j} \cdot h_j$$

The method assumes a uniform distribution of observations within each class interval. This is a standard approach for grouped data when individual observations are not available.

Value

A vector of estimated quantiles on grouped data corresponding to `probs`. Returns `NA` if total frequency is zero or missing.

Handling Open-Ended Classes

When dealing with administrative or tax data, the first class is often defined as negative income (or incomes below zero) and the last class as incomes above a certain threshold. In such cases, we have `-Inf` as the lower bound of the first class and `Inf` as the upper bound of the last class.

If `Inf` values are present in the given bounds, the function imputes reasonable bounds using the specified method:

For open left class (first lower bound = -Inf): The imputed first lower bound is given by:

$$L_1^* = U_1 - h_2$$

where U_1 is the upper bound of the first class and $h_2 = U_2 - L_2$ is the width of the second class. This assumes the first class has the same width as the second class.

For open right class (last upper bound = Inf):

The imputed upper bound is given by:

$$U_J^* = L_J + h_{J-1}$$

where L_J is the lower bound of the last class and $h_{J-1} = U_{J-1} - L_{J-1}$ is the width of the second-to-last class. This assumes the last class has the same width as the penultimate class.

Special Cases

- If the quantile class has zero frequency, the function returns the class midpoint as a fallback.
- If total frequency is zero or NA, the function returns NA for all requested quantiles.

See Also

quantile for quantiles of ungrouped data.

Other grouped data functions: gini_grouped(), qri_grouped()

Examples

```
# Basic usage: compute quartiles
freq <- c(5, 8, 10, 4, 3)
lower <- c(0, 10, 20, 30, 40)
upper <- c(10, 20, 30, 40, 50)

quantile_grouped(freq, lower, upper, probs = c(0.25, 0.5, 0.75))

# Compute deciles
quantile_grouped(freq, lower, upper, probs = seq(0.1, 0.9, by = 0.1))

# With custom midpoints
midpts <- c(5, 15, 25, 35, 45)
quantile_grouped(freq, lower, upper, probs = 0.5, midpoints = midpts)

# Income distribution example
income_freq <- c(120, 180, 150, 80, 40, 20, 10)
income_lower <- c(0, 15000, 30000, 45000, 60000, 80000, 100000)
income_upper <- c(15000, 30000, 45000, 60000, 80000, 100000, 150000)

# Compute median income
quantile_grouped(income_freq, income_lower, income_upper, probs = 0.5)

# Compute income quintiles
quantile_grouped(income_freq, income_lower, income_upper,
                   probs = seq(0.2, 0.8, by = 0.2))
```

`ratio_quantiles` *Ratio of Quantiles (e.g., P90/P10)*

Description

Estimates ratio of quantiles (e.g., P90/P10) on simple and complex sampling data

Usage

```
ratio_quantiles(
  y,
  weights = NULL,
  prob_numerator = 0.9,
  prob_denominator = 0.1,
  type = 6,
  na.rm = TRUE
)
```

Arguments

`y`	A numeric vector of data values
`weights`	A numeric vector of sampling weights (optional). If `NULL`, all observations are equally weighted.
`prob_numerator`	The percentile to be considered at the numerator (default `0.90`)
`prob_denominator`	The percentile to be considered at the denominator (default `0.10`)
`type`	Quantile estimation type: integer `4–9` or `"HD"` for Harrell–Davis (default: `6`). See `csquantile`.
`na.rm`	Logical, should missing values be removed? (default: TRUE)

Details

Consider a random sample s of size n, and let w_j, $j \in s$, define the sampling weight and y_j be the observed characteristics (i.e. income) associated to the j-th individual, $j = 1, \ldots, n$. Let p_n be the order of the quantile at the numerator and p_d be the order of the quantile at the denominator. For example, set $p_n = 0.90$ and $p_d = 0.10$. Then the popular P90/P10 ratio can be estimated by

$$\widehat{P90/P10} = \frac{\widehat{Q}(p_n = 0.9)}{\widehat{Q}(p_d = 0.1)}$$

where the estimated quantiles $\widehat{Q}(p)$ are computed via the function `csquantile()`, which accounts for sampling weights and the specified quantile type.

Value

A scalar numeric value representing the estimated ratio of quantiles

See Also

Other inequality indicators based on quantiles: `inequantiles()`, `plot_inequality_curve()`, `qri()`, `share_ratio()`, `superpop_qri()`

Examples

```
data(synthouse)

eq <- synthouse$eq_income # Income data

# Compute unweighted P90/P10 with default type 6 quantile estimator
ratio_quantiles(y = eq)

# Consider the sampling weights, change quantile estimation type and orders of quantiles
w <- synthouse$weight
ratio_quantiles(y = eq, weights = w, prob_numerator = 0.6, prob_denominator = 0.1, type = 5)

# Compare the P90/P10 across macro-regions (NUTS1)
tapply(1:nrow(synthouse), synthouse$NUTS1, function(area) {
  ratio_quantiles(y = synthouse$eq_income[area],
      weights = synthouse$weight[area])
})
```

---

rescaled_bootstrap *Rescaled Bootstrap Variance Estimation*

Description

Implements the *rescaled bootstrap* method for variance estimation in survey data, supporting both stratified simple random sampling and multistage complex designs.

Usage

```
rescaled_bootstrap(
  data,
  y,
  strata,
  N_h = NULL,
  psu = NULL,
  weights = NULL,
  estimator,
  by_strata = TRUE,
  B = 200,
  m_h = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`data`	A data frame containing the survey data.
`y`	A character string specifying the variable name to be used for the target variable.
`strata`	A character string specifying the stratification variable.
`N_h`	Optional vector of stratum population sizes, used for the finite population correction (FPC). Can be a single value (applied to all strata) or one value per stratum.
`psu`	Optional character string specifying the Primary Sampling Unit (PSU) variable. Required for multistage complex designs.
`weights`	Optional character string specifying the sampling weight variable. Required for complex designs with unequal inclusion probabilities.
`estimator`	A function that computes the statistic of interest, accepting arguments `estimator(y, weights)` for complex designs or `estimator(y)` for simple designs.
`by_strata`	Logical; if `TRUE`, variances are computed separately by stratum.
`B`	Integer; number of bootstrap replicates (default = 200).
`m_h`	Optional vector of bootstrap sample sizes per stratum (PSUs for complex designs). If `NULL`, defaults to $m_h = \lfloor (n_h - 2)^2/(n_h - 1) \rfloor$.
`seed`	Optional integer for reproducibility.
`verbose`	Logical; if `TRUE` (default), displays a progress bar during bootstrap iterations.

Details

The rescaled bootstrap is a resampling technique designed for complex survey data that preserves stratification and primary sampling unit (PSU) structure, providing consistent variance estimation for both smooth and non-smooth statistics. The methodology is based on Rao and Wu (1988) and Rao et al. (1992).

(1) Stratified Simple Random Sampling design

Consider a finite population divided into H strata, each of size N_h, with a sample of size n_h selected independently in each h stratum. Suppose to be interested in some θ parameter, with $\hat{\theta}$ sampling estimator. For each b bootstrap replicate, $b = 1, \ldots, B$ and stratum h:

1. Draw a bootstrap sample of size m_h with replacement from the n_h sampled units. By default, $m_h = \lfloor (n_h - 2)^2/(n_h - 1) \rfloor \approx n_h - 3$.
2. Compute rescaled bootstrap values:

$$\tilde{y}_{hj}^{*(b)} = \bar{y}_h + \sqrt{\frac{m_h(1 - f_h)}{n_h - 1}}(y_{hj}^{*(b)} - \bar{y}_h),$$

   where $y_{hj}^{*(b)}$ is the bootstrap observation, $1 - f_h$ is the FPC, with $f_h = n_h/N_h$, and $\bar{y}_h$ is the sample stratum mean.
3. Compute the statistic of interest $\hat{\theta}_h^{*(b)}$ using rescaled values.

The variance is then estimated by the bootstrap variance.

(2) Two-Stage Stratified Sampling design

For designs with PSUs and sampling weights:

1. Within each stratum h, draw m_h PSUs with replacement from the n_h sampled PSUs. By default, $m_h = \lfloor (n_h - 2)^2/(n_h - 1) \rfloor \approx n_h - 3$.
2. Let $m_{hi}^{(b)}$ denote the number of times PSU i is selected in replicate b. Each observation in the i-th PSU is assigned a rescaled bootstrap weight:

$$w_{hij}^{*(b)} = \left[1 - c_h + c_h \frac{n_h}{m_h} m_{hi}^{(b)}\right] w_{hij}, \qquad c_h = \sqrt{\frac{m_h}{n_h - 1}}.$$

   w_{hij} is the sampling weight associated to individual j in PSU i in stratum h
3. The statistic $\hat{\theta}_h^{*(b)}$ is computed using the rescaled weights.

The variance is then estimated by the bootstrap variance.

Multiple estimators

The `estimator` argument accepts any function with signature `f(y, weights)` (complex design) or `f(y)` (simple design), including functions from this package and user-defined ones. When `estimator` returns a *named numeric vector*, variances are computed for all outputs simultaneously from the same bootstrap replicates, so the resulting standard errors are directly comparable across indicators.

Value

A list containing:

`variance`	Bootstrap variance estimate
`boot_estimates`	Vector of B bootstrap estimates
`B`	Number of bootstrap replicates
`by_strata`	Logical; `TRUE` if variance is computed separately by stratum.
`design`	Character string: `"two-stage stratified"` or `"stratified SRS"`.
`strata_info`	Data frame with number of observations/PSUs per stratum.
`call`	The matched function call.

References

Rao J, Wu C (1988). "Resampling inference with complex survey data." *Journal of the American Statistical Association*, **83**, 231–241.

Rao J, Wu C, Yue K (1992). "Some recent work on resampling methods for complex surveys." *Survey methodology*, **18**, 209–217.

Kolenikov S (2010). "Resampling variance estimation for complex survey data." *The Stata Journal*, **10**, 165–199.

Scarpa S, Ferrante MR, Sperlich S (2025). "Inference for the quantile ratio inequality index in the context of survey data." *Journal of Survey Statistics and Methodology*. doi:10.1093/jssam/smaf024.

See Also

For a convenience wrapper that automatically computes all package inequality indicators and their standard errors in a single call, see `inequantiles`.

Examples

```
data(synthouse)


# ================================================================
# Example 1: Stratified Simple Random Sampling (SRS)
# ================================================================


# Use NUTS2 as strata
set.seed(123)

# Simulate population sizes per stratum (for FPC)
N_values <- sample(2000:5000, length(unique(synthouse$NUTS2)), replace = TRUE)
names(N_values) <- sort(unique(synthouse$NUTS2))

# Define a simple mean estimator
mean_estimator <- function(y) mean(y, na.rm = TRUE)

# Apply the rescaled bootstrap under stratified SRS
boot_srs <- rescaled_bootstrap(
  data = synthouse,
  y = "eq_income",
  strata = "NUTS2",
  N_h = N_values,
  estimator = mean_estimator,
  by_strata = TRUE,
  B = 50,  # small number for illustration
  seed = 123,
  verbose = FALSE
)

# View results
boot_srs$variance


# ================================================================
# Example 2: Two-stage Complex Design
# ================================================================

# Estimate the QRI estimator sampling variance.
boot_complex <- rescaled_bootstrap(
  data = synthouse,
  y = "eq_income",
  strata = "NUTS2",
  psu = "municipality",
  weights = "weight",
  estimator = qri,
  by_strata = TRUE,
  B = 50,
  seed = 456,
  verbose = FALSE
```

```
)

# Display variance and bootstrap estimates
summary(boot_complex$variance)

# Strata and PSU summary



# ================================================================
# Example 3: Multiple estimators in a single bootstrap loop
# ================================================================

# Create a function returning a named vector of estimates,
# including package functions and user-defined ones. All indicators share
# the same bootstrap replicates, ensuring directly comparable standard errors.

multi_estimator <- function(y, weights) {
  c(
    w_mean = sum(y * weights) / sum(weights),        # custom: weighted mean
    qri    = qri(y, weights = weights),               # package function
    qsr    = share_ratio(y, weights = weights)        # package function
  )
}

boot_multi <- rescaled_bootstrap(
  data      = synthouse,
  y         = "eq_income",
  strata    = "NUTS2",
  psu       = "municipality",
  weights   = "weight",
  estimator = multi_estimator,
  by_strata = FALSE,
  B         = 50,
  seed      = 42,
  verbose   = FALSE
)

# One variance per indicator, all from the same replicates
boot_multi$variance



# ================================================================
# Note:
# These examples use small B for speed. For actual analysis,
# use B >= 200 for stable estimates.
# ================================================================
```

share_ratio *Quantile-Based Share Ratio*

Description

Estimates a quantile-based share ratio (QBSR) for measuring inequality from simple or complex survey data.

Usage

```
share_ratio(
  y,
  weights = NULL,
  type = 6,
  na.rm = TRUE,
  prob_numerator = 0.8,
  prob_denominator = 0.2
)
```

Arguments

y	A numeric vector of strictly positive values (e.g. income, wealth).
weights	A numeric vector of sampling weights. If NULL, all observations are equally weighted.
type	Quantile estimation type: integer 4–9 or "HD" for Harrell–Davis (default: 6). See csquantile.
na.rm	Logical; remove missing values before computing? Default: TRUE.
prob_numerator	Numeric in $(0, 1)$; quantile order for the numerator (default: 0.80, corresponding to the QSR top share).
prob_denominator	Numeric in $(0, 1)$; quantile order for the denominator (default: 0.20, corresponding to the QSR bottom share).

Details

Consider a random sample s of size n, and let y_j and w_j, $j \in s$, define the observed value and the sampling weight associated to the j-th individual. Define p_n and p_d as the orders of the numerator and denominator quantiles, respectively. The QBSR estimator is defined as:

$$\widehat{QBSR} = \frac{\sum_{j \in s} w_j y_j \mathbf{1}\left\{y_j \geq \widehat{Q}(p_n)\right\}}{\sum_{j \in s} w_j y_j \mathbf{1}\left\{y_j \leq \widehat{Q}(p_d)\right\}}$$

where $\widehat{Q}(p)$ is computed via csquantile, which accounts for sampling weights and the specified quantile type.

The most well-known special cases are the quintile share ratio (QSR; Langel and Tillé (2011)), obtained with $p_n = 0.80$ and $p_d = 0.20$, and the Palma index (Palma (2006); Palma (2011)), obtained with $p_n = 0.90$ and $p_d = 0.40$.

Value

A scalar numeric value representing the estimated share ratio.

References

Langel M, Tillé Y (2011). "Statistical inference for the quintile share ratio." *Journal of Statistical Planning and Inference*, **141**, 2976–2985.

Palma JG (2006). "Globalizing Inequality: 'Centrifugal' and 'Centripetal' Forces at Work." United Nations, Department of Economics and Social Affairs.

Palma JG (2011). "Homogeneous middles vs. heterogeneous tails, and the end of the 'inverted-U': It's all about the share of the rich." *Development and Change*, **42**, 87–153.

See Also

`csquantile` for quantile estimation

Other inequality indicators based on quantiles: `inequantiles()`, `plot_inequality_curve()`, `qri()`, `ratio_quantiles()`, `superpop_qri()`

Examples

```
data(synthouse)
eq <- synthouse$eq_income
w  <- synthouse$weight

# QSR (default: top 20% vs bottom 20%)
share_ratio(y = eq, weights = w)

# Palma index (top 10% vs bottom 40%)
share_ratio(y = eq, weights = w, prob_numerator = 0.90, prob_denominator = 0.40)

# Compare across macro-regions (NUTS1)
tapply(1:nrow(synthouse), synthouse$NUTS1, function(idx) {
  share_ratio(y = synthouse$eq_income[idx],
              weights = synthouse$weight[idx])
})
```

`superpop_qri` *Quantile Ratio Index in Superpopulation*

Description

Computes the theoretical quantile ratio index (QRI) for measuring inequality for a given parametric distribution.

Usage

```
superpop_qri(qfunction, lower = 0, upper = 1, subdivisions = 1000L, ...)
```

Arguments

qfunction	A quantile function (e.g., qnorm, qlnorm, qgamma).
lower	Lower bound of integration. Default is 0.
upper	Upper bound of integration. Default is 1.
subdivisions	Maximum number of subintervals for integration. Default is 1000L.
...	Additional parameters to pass to qfunction (e.g., distribution parameters).

Details

The QRI was proposed by (Prendergast and Staudte 2018) for measuring economic inequality. Consider a random variable Y with positive support, which admits a continuous CDF F and quantile function $Q(p) = F^{-1}(p)$, for any $p \in (0, 1)$. It is calculated as:

$$QRI = 1 - \int_0^1 R(p)dp$$

where $R(p) = Q(p/2)/Q(1 - p/2)$ is the ratio of symmetric quantiles, with $R(0) = 0$ and $R(1) = 1$.

This function computes the (superpopulation) QRI for theoretical parametric distributions, as opposed to qri which estimates the QRI from sample data.

Value

A numeric value representing the theoretical QRI for the specified parametric distribution. Values range from 0 (perfect equality) to 1 (maximum inequality).

References

Prendergast LA, Staudte RG (2018). "A simple and effective inequality measure." *The American Statistician*, **72**, 328–343.

See Also

qri for the sample-based QRI estimator, plot_inequality_curve for its representation

Other inequality indicators based on quantiles: inequantiles(), plot_inequality_curve(), qri(), ratio_quantiles(), share_ratio()

Examples

```
# Log-normal distribution
superpop_qri(qlnorm, meanlog = 9, sdlog = 0.3)
superpop_qri(qlnorm, meanlog = 9, sdlog = 1.4)

# Weibull distribution
```

```
superpop_qri(qweibull, shape = 1.7, scale = 30000)
superpop_qri(qweibull, shape = 3, scale = 30000)
```

`synthouse` *Synthetic Household Survey Data*

Description

A realistic synthetic dataset based on the empirical structure of real IT-SILC (Italian Survey on Income and Living Conditions) 2024 data.

Usage

```
synthouse
```

Format

A data frame with 20,034 rows (individuals nested in 10,099 households) and 17 variables covering demographic, socio-economic, and geographic information:

person_id Character. Unique person identifier, composed of the household ID followed by a person index within the household (format: HH000001P1, HH000001P2, HH000002P1, ...)

hh_id Character. Household identifier. All individuals in the same household share this ID (format: HH000001, HH000002, ...)

NUTS1 Character. NUTS1 region code (5 macro-regions):

- N
- S
- NE
- NO
- C

NUTS2 Character. NUTS2 region code (30 regions, format: N01-N06, S01-S06, ...)

NUTS3 Character. NUTS3 province code (120 provinces, format: N01001-N01004, ...)

municipality Character. Municipality code (1,079 municipalities, format: N010010001-N010010008, ...)

age Integer. Age in years (0-85)

age_class Factor. Age class with 7 levels: "0-14", "15-17", "18-24", "25-34", "35-49", "50-64", "65+"

gender Integer. Gender code:

- 1 = Male
- 2 = Female

education_level Character. Education level (adults 18+ only, NA for minors):

- "Low" = No education, primary, or lower secondary (ISCED 0-2)
- "Medium" = Upper secondary or post-secondary non-tertiary (ISCED 3-5)
- "High" = Tertiary education (ISCED 6-8)

employment_status Character. Main activity status:

- "Employed" = In employment
- "Unemployed" = Unemployed
- "Retired" = Retired
- "Student" = Student or pupil
- "Other" = Other (unable to work, domestic tasks, etc.)

hh_size Integer. Household size (number of members): 1-7

hh_type Character. Household type:

- "Single" = One-person household
- "Couple" = Two adults without children
- "Single_parent" = Single parent with children
- "Family" = Household with children (2+ adults)
- "Other" = Other household types

eq_income Numeric. Equivalised disposable household income in euros. This is the total household income divided by the OECD modified equivalence scale. All household members share the same equivalised income.

hh_income Numeric. Total disposable household income in euros before equivalisation. All household members share the same total income.

oecd_scale Numeric. OECD modified equivalence scale for the household:

- First adult (14+): weight = 1.0
- Other adults (14+): weight = 0.5 each
- Children (< 14): weight = 0.3 each

Formula: modif_oecd_scale = 1.0 + 0.5 × (n_adults - 1) + 0.3 × n_children

weight Numeric. Sampling weight (inverse inclusion probability). Represents the number of individuals in the population represented by this sample unit. All household members share the same weight.

Details

The synthetic dataset was generated to reproduce key characteristics of IT-SILC data, but contains fictional values; it is therefore suitable for methodology illustration and testing, not for policy analysis. It is primarily intended to demonstrate the computation of quantile-based inequality indicators provided by the `inequantiles` package, such as quantiles, quantile-based indicators, influence functions, and variance estimation.

Geographic variables follow a hierarchical NUTS structure with realistic proportions across macroregions and were created randomly; they do not correspond to real codes. Individual characteristics (age, gender, education, ...) were assigned randomly based on conditional empirical distributions from IT-SILC. Income was generated using a regression model fitted to IT-SILC data:

$$\log(eq\_income) \sim education\_head + n\_employed + age\_head + age\_head^2 + hh\_size.$$

where the suffix _head identifies variables measured for the household head (e.g., education_head is the education level of the household head, age_head is their age). Sampling weights follow a lognormal distribution fitted to IT-SILC.

Key Statistics:

- Sample size: 20,034 individuals in 10,099 households
- Average household size: ~1.99 (matching IT-SILC)
- Estimated population: 15,749,925 individuals (the sum of the weights)
- Geographic coverage: 5 macro-regions, 30 NUTS2, 120 NUTS3, 1,079 municipalities

References

Eurostat (2024). *EU Statistics on Income and Living Conditions (EU-SILC): Methodology*. https://ec.europa.eu/eurostat/

Examples

```
# Load the dataset
data(synthouse)

# Basic structure
str(synthouse)
head(synthouse)

# Summary statistics
summary(synthouse$eq_income)
summary(synthouse$age)

# Number of households and individuals
length(unique(synthouse$hh_id))  # Households
nrow(synthouse)                  # Individuals

# Average household size
mean(table(synthouse$hh_id))

# Distribution of household types
table(unique(synthouse[, c("hh_id", "hh_type")])$hh_type)

# Age distribution
table(synthouse$age_class)

# Weighted quantiles
csquantile(synthouse$eq_income,
           weights = synthouse$weight,
           probs = c(0.25, 0.5, 0.75),
           type = 6)

# Quantile Ratio Index
qri(synthouse$eq_income,
    weights = synthouse$weight,
    type = 6)
```

Index

∗ **datasets**
synthouse, 35
∗ **grouped data functions**
gini_grouped, 4
qri_grouped, 21
quantile_grouped, 23
∗ **inequality indicators based on quantiles**
inequantiles, 14
plot_inequality_curve, 16
qri, 19
ratio_quantiles, 26
share_ratio, 32
superpop_qri, 33
∗ **influence functions**
if_gini, 5
if_qri, 7
if_quantile, 9
if_ratio_quantiles, 10
if_share_ratio, 12

csquantile, 2, *7–13*, *15*, *17*, *20*, *26*, *32, 33*

gini_grouped, 4, *23*, *25*

if_gini, 5, *8*, *10*, *12, 13*
if_qri, *6*, 7, *10*, *12, 13*, *20*
if_quantile, *6*, *8*, 9, *12, 13*
if_ratio_quantiles, *6*, *8*, *10*, 10, *13*
if_share_ratio, *6*, *8*, *10*, *12*, 12
inequantiles, 14, *18*, *20*, *27*, *29*, *33, 34*

plot_inequality_curve, *16*, 16, *20*, *27*, *33, 34*
print.inequantiles *(*inequantiles*)*, 14

qri, *8*, *15, 16*, *18*, 19, *22, 23*, *27*, *33, 34*
qri_grouped, *5*, *20*, 21, *25*
quantile, *25*
quantile_grouped, *5*, *22, 23*, 23

ratio_quantiles, *12*, *16*, *18*, *20*, 26, *33, 34*
rescaled_bootstrap, *15, 16*, 27

share_ratio, *13*, *15, 16*, *18*, *20*, *27*, 32, *34*
superpop_qri, *16*, *18*, *20*, *23*, *27*, *33*, 33
synthouse, 35