Package 'pricelevels'

July 22, 2025

Type Package

Title Spatial Price Level Comparisons

Version 1.4.0

Description Price comparisons within or between countries provide an overall measure of the relative difference in prices, often denoted as price levels. This package provides index number methods for such price comparisons (e.g., The World Bank, 2011, <doi:10.1596/978-0-8213-9728-2>). Moreover, it contains functions for sampling and characterizing price data.

License EUPL

Depends R (>= 4.0.1)

Imports data.table (>= 1.14.0), minpack.lm (>= 1.2-1)

Encoding UTF-8

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/sweinand/pricelevels

BugReports https://github.com/sweinand/pricelevels/issues

Language en-US

NeedsCompilation no

Author Sebastian Weinand [aut, cre]

Maintainer Sebastian Weinand <s.weinand90@googlemail.com>

Repository CRAN

Date/Publication 2025-07-22 13:00:50 UTC

Contents

`bilateral.index` *Bilateral price indices*

Description

Calculation of bilateral price indices. Currently, the following ones are implemented (see below in alphabetic order).

Usage

```
banerjee(p, r, n, q, base=NULL, settings=list())

bmw(p, r, n, base=NULL, settings=list())

carli(p, r, n, base=NULL, settings=list())

cswd(p, r, n, base=NULL, settings=list())

davies(p, r, n, q, base=NULL, settings=list())

drobisch(p, r, n, q, w=NULL, base=NULL, settings=list())

dutot(p, r, n, base=NULL, settings=list())

fisher(p, r, n, q, w=NULL, base=NULL, settings=list())

geolaspeyres(p, r, n, q, w=NULL, base=NULL, settings=list())

geopaasche(p, r, n, q, w=NULL, base=NULL, settings=list())

geowalsh(p, r, n, q, w=NULL, base=NULL, settings=list())

geoyoung(p, r, n, q, w=NULL, base=NULL, settings=list())

harmonic(p, r, n, base=NULL, settings=list())

jevons(p, r, n, base=NULL, settings=list())

laspeyres(p, r, n, q, w=NULL, base=NULL, settings=list())
```

```
lehr(p, r, n, q, base=NULL, settings=list())

lowe(p, r, n, q, base=NULL, settings=list())

medgeworth(p, r, n, q, base=NULL, settings=list())

paasche(p, r, n, q, w=NULL, base=NULL, settings=list())

palgrave(p, r, n, q, w=NULL, base=NULL, settings=list())

svartia(p, r, n, q, w=NULL, base=NULL, settings=list())

toernqvist(p, r, n, q, w=NULL, base=NULL, settings=list())

theil(p, r, n, q, w=NULL, base=NULL, settings=list())

uvalue(p, r, n, q, base=NULL, settings=list())

walsh(p, r, n, q, w=NULL, base=NULL, settings=list())

young(p, r, n, q, base=NULL, settings=list())
```

Arguments

p	A numeric vector of positive prices.
r, n	A character vector or factor of regional entities r and products n, respectively.
q, w	A numeric vector of non-negative quantities q or expenditure share weights w (see Section 'Details'). Either q or w must be provided for weighted indices. If both q and w are provided, q will be used.
base	A character specifying the base region to which all price levels are expressed. If NULL, base region is set internally.
settings	A list of control settings to be used. The following settings are supported:

- chatty : A logical specifying if warnings and info messages should be printed or not. The default is getOption("pricelevels.chatty").
- connect : A logical specifying if the data should be checked for connectedness or not. The default is getOption("pricelevels.connect"). If the data are not connected, price levels are computed within the biggest block of connected regions or the block of regions to which the base region belongs. See also connect().
- plot : A logical specifying if the calculated price levels should be plotted or not. If TRUE, the price ratios of each region are displayed as boxplots and the price levels are added as colored points. The default is getOption("pricelevels.plot").
- qbase : A character specifying the region b whose quantities (and prices) should be used by lowe(), young(), and geoyoung(). If NULL, prices are averaged and quantities added up for each product, i.e. $p_i^b = \sum_{r=1}^{R} p_i^r / R$ and $q_i^b = \sum_{r=1}^{R} q_i^r$.

Details

Before calculations start, missing values are removed from the data. Duplicated observations for r and n are aggregated, that is, duplicated prices p and weights w are averaged and duplicated quantities q added up. If there is more than one region in the data, products with prices in only one region r are removed.

The weights w must represent expenditure shares defined as $w_i^r = p_i^r q_i^r / \sum_{j=1}^{N} p_j^r q_j^r$. They are internally (re-)normalized such that they add up to 1 for each region r.

Value

A named vector of price levels.

Author(s)

Sebastian Weinand

References

ILO, IMF, OECD, UNECE, Eurostat and World Bank (2020). *Consumer Price Index Manual: Concepts and Methods*. Washington DC: International Monetary Fund.

Examples

```
# sample complete price data:
set.seed(123)
dt1 <- rdata(R=3, B=1, N=5)

# compute jevons and toernqvist index:
dt1[, jevons(p=price, r=region, n=product, base="1")]
dt1[, toernqvist(p=price, r=region, n=product, q=quantity, base="1")]

# compute lowe index using quantities of region 2:
dt1[, lowe(p=price, r=region, n=product, q=quantity, base="1",
           settings=list(qbase="2"))]

# add price data:
dt2 <- rdata(R=4, B=1, N=4)
dt2[, "region":=factor(region, labels=4:7)]
dt2[, "product":=factor(product, labels=6:9)]
dt <- rbind(dt1, dt2)
dt[, is.connected(r=region, n=product)] # non-connected now

# compute jevons index with base region 1:
dt[, jevons(p=price, r=region, n=product, base="1")]

# change base region:
dt[, jevons(p=price, r=region, n=product, base="4")]
```

cpd *CPD and NLCPD methods*

Description

The function `cpd()` estimates regional price levels by the Country-Product-Dummy (CPD) method, originally developed by Summers (1973). Auer and Weinand (2025) recently proposed a generalization of the CPD method. This nonlinear CPD method (NLCPD method) is implemented in the function `nlcpd()`.

Usage

```
cpd(p, r, n, q=NULL, w=NULL, base=NULL, simplify=TRUE, settings=list())

nlcpd(p, r, n, q=NULL, w=NULL, base=NULL, simplify=TRUE, settings=list(), ...)
```

Arguments

`p` A numeric vector of positive prices.

`r, n` A character vector or factor of regional entities `r` and products `n`, respectively.

`q, w` A numeric vector of non-negative quantities `q` or weights `w`. By default, no weights are used in the regression (`q=NULL` and `w=NULL`). While `w` can be any weights considered as appropriate for weighted regression, `q` will result in an expenditure share weighted regression (see Section 'Details'). If both `q` and `w` are provided, `q` will be used.

`base` A character specifying the base to which the estimated logarithmic regional price levels are expressed. When `NULL`, they refer to the (unweighted) regional average, similar to `contr.sum`.

`simplify` A logical indicating whether the full regression-object should be provided (`FALSE`) or a named vector of estimated regional price levels (`TRUE`).

`settings` A list of control settings to be used. The following settings are supported:

- `chatty` : A logical specifying if warnings and info messages should be printed or not. The default is `getOption("pricelevels.chatty")`.
- `connect` : A logical specifying if the data should be checked for connectedness or not. The default is `getOption("pricelevels.connect")`. If the data are not connected, price levels are computed within the biggest block of connected regions or the block of regions to which the `base` region belongs. See also `connect()`.
- `norm.weights` : A logical specifying if the weights `w` should be renormalized such that they add up to 1 for each region `r` or not. The default is `TRUE`.
- `plot` : A logical specifying if the calculated price levels should be plotted or not. If `TRUE`, the price ratios of each region are displayed as boxplots and the price levels are added as colored points. The default is `getOption("pricelevels.plot")`.

- `self.start` : Only if `par=NULL`, the strategy how parameter start values are internally derived by `nlcpd()`. Currently, values `s1`, `s2` and `s3` are allowed. For `s1`, simple price averages across products and regions are used as start values, while these are derived by the CPD method for strategies `s2` and `s3`. Start values for `delta` are either set to 1 or derived by their first-order condition if `s3`. By default, `self.start='s1'`.
- `use.jac` : A logical indicating if the jacobian matrix should be used by `nlcpd()` for the nonlinear optimization or not. The default is `FALSE`.
- `w.delta` : A named vector of weights for the `delta`-parameter (see section 'Method'). Vector length must be equal to the number of products, while names must match product names. If not supplied, δ_i weights are derived internally by `nlcpd()` from the weights `w`.

`...` Further arguments passed to `nls.lm`, typically arguments `control`, `par`, `upper`, and `lower`. For `par`, `upper`, and `lower`, vectors must have names for each parameter separated by a dot, e.g., `lnP.1`, `pi.2`, or `delta.3`.

Details

Before calculations start, missing values are removed from the data. Duplicated observations for `r` and `n` are aggregated, that is, duplicated prices `p` and weights `w` are averaged and duplicated quantities `q` added up. If there is more than one region in the data, products with prices in only one region `r` are removed.

If `q` is provided, expenditure shares are derived as $w_i^r = p_i^r q_i^r / \sum_{j=1}^{N} p_j^r q_j^r$ and used as weights in the regression. If only `w` is provided, the weights `w` are (re-)normalized by default. If the weights `w` do not represent expenditure shares, the (re-)normalization can be turned off by `settings=list(norm.weights=FALSE)`.

Value

For `simplify=TRUE`, a named vector of (unlogged) regional price levels. Otherwise, for `cpd()`, a `lm`-object containing the full regression output, and for `nlcpd()` the full output of `nls.lm()` plus element `w.delta`.

Method

The CPD method is a linear regression model that explains the logarithmic price of product i in region r, $\ln p_i^r$, by the general product price, $\ln \pi_i$, and the overall price level, $\ln P^r$:

$$\ln p_i^r = \ln \pi_i + \ln P^r + u_i^r$$

The NLCPD method inflates the CPD model by product-specific elasticities δ_i:

$$\ln p_i^r = \ln \pi_i + \delta_i \ln P^r + u_i^r$$

Both methods require a normalization of the estimated price levels $\widehat{\ln P^r}$ to avoid multicollinearity. If `base=NULL`, the normalization $\sum_{r=1}^{R} \widehat{\ln P^r} = 0$ is used by `cpd()` and `nlcpd()`; otherwise, one price level is set to 0. The NLCPD method additionally imposes the restriction $\sum_{i=1}^{N} w_i \widehat{\delta}_i = 1$, where the weights w_i can be defined by `settings$w.delta`. In `nlcpd()`, one $\widehat{\delta}_i$-parameter is derived residually from this restriction instead of being estimated.

Author(s)

Sebastian Weinand

References

Auer, L. v. and Weinand, S. (2025). The Country-Product-Dummy Method With Product-Specific Spatial Price Variation. *Review of Income and Wealth*, 71: e70005.

Summers, R. (1973). International Price Comparisons based upon Incomplete Data. *Review of Income and Wealth*, 19 (1), 1-16.

See Also

`lm`, `dummy.coef`, `nls.lm`

Examples

```
# sample complete price data:
set.seed(123)
R <- 3 # number of regions
B <- 1 # number of product groups
N <- 5 # number of products
dt1 <- rdata(R=R, B=B, N=N)

# compute expenditure share weighted cpd and nlcpd index:
dt1[, cpd(p=price, r=region, n=product, q=quantity)]
dt1[, nlcpd(p=price, r=region, n=product, q=quantity)]

# set individual start values in nlcpd():
par.init <- list("lnP"=setNames(rep(0, R), 1:R),
                 "pi"=setNames(rep(2, N), 1:N),
                 "delta"=setNames(rep(1, N), 1:N))
dt1[, nlcpd(p=price, r=region, n=product, q=quantity, par=par.init)]

# use lower and upper bounds on parameters:
dt1[, nlcpd(p=price, r=region, n=product, q=quantity,
            lower=unlist(par.init)-0.1, upper=unlist(par.init)+0.1)]

# change internal calculation of start values:
dt1[, nlcpd(p=price, r=region, n=product, q=quantity, settings=list(self.start="s2"))]

# add price data:
dt2 <- rdata(R=4, B=1, N=4)
dt2[, "region":=factor(region, labels=4:7)]
dt2[, "product":=factor(product, labels=6:9)]
dt <- rbind(dt1, dt2)
dt[, is.connected(r=region, n=product)] # non-connected now

# compute expenditure share weighted cpd and nlcpd index:
dt[, cpd(p=price, r=region, n=product, q=quantity, base="1")]
dt[, nlcpd(p=price, r=region, n=product, q=quantity, base="1")]
```

```
# compare with toernqvist index:
dt[, toernqvist(p=price, r=region, n=product, q=quantity, base="1")]


# computational speed in nlcpd() usually increases if use.jac=TRUE:
set.seed(123)
dt3 <- rdata(R=20, B=1, N=30)
system.time(m1 <- dt3[, nlcpd(p=price, r=region, n=product, q=quantity,
                              settings=list(use.jac=FALSE), simplify=FALSE,
                              control=minpack.lm::nls.lm.control("maxiter"=200))])
system.time(m2 <- dt3[, nlcpd(p=price, r=region, n=product, q=quantity,
                              settings=list(use.jac=TRUE), simplify=FALSE,
                              control=minpack.lm::nls.lm.control("maxiter"=200))])
all.equal(m1$par, m2$par, tol=1e-05)
```

---

geks *GEKS method*

Description

The function `index.pairs()` computes bilateral index numbers for all pairs of regions. Based on that, the function `geks()` derives regional price levels using the GEKS method proposed by Gini (1924, 1931), Elteto and Koves (1964), and Szulc (1964).

Usage

```
index.pairs(p, r, n, q=NULL, w=NULL, settings=list())

geks(p, r, n, q=NULL, w=NULL, base=NULL, simplify=TRUE, settings=list())
```

Arguments

`p`	A numeric vector of positive prices.
`r, n`	A character vector or factor of regional entities `r` and products `n`, respectively.
`q, w`	A numeric vector of non-negative quantities `q` or expenditure share weights `w` (see Section 'Details') to be used in the computation of weighted bilateral index numbers. Can be `NULL`, if the index formula specified in `type` does not require quantities or weights. If both `q` and `w` are provided, `q` will be used.
`base`	A character specifying the base region to which all price levels are expressed. When `NULL`, they refer to the (unweighted) regional average.
`simplify`	A logical indicating whether the full regression-object should be provided (`FALSE`) or a named vector of estimated regional price levels (`TRUE`).
`settings`	A list of control settings to be used. The following settings are supported: • `chatty` : A logical specifying if warnings and info messages should be printed or not. The default is `getOption("pricelevels.chatty")`.

- connect : A logical specifying if the data should be checked for connectedness or not. The default is getOption("pricelevels.connect") for geks() and FALSE for index.pairs(). If the data are not connected, price levels are computed within the biggest block of connected regions or the block of regions to which the base region belongs. See also connect().
- plot : A logical specifying if the calculated price levels should be plotted or not. If TRUE, the price ratios of each region are displayed as boxplots and the price levels are added as colored points. The default is getOption("pricelevels.plot"). Used only by geks().
- all.pairs : A logical indicating whether bilateral index numbers should be computed for all region pairs (TRUE, the default) or only for non-redundant ones (e.g., the index number P^{AB} is computed while P^{BA} and P^{AA} are excluded). For bilateral index formulas passing the country reversal test, the resulting price levels derived by geks() will be the same. The default is TRUE.
- type : A character specifying the bilateral index formula(s) used to aggregate individual prices into price indices for each pair of regions (first step of GEKS). See bilateral.index for allowed values. Multiple choices allowed. The default is jevons.
- wmethod : the weighting method (second step of GEKS). Allowed values are none for equal weighting of all bilateral price indices, obs for weighting the bilateral price indices according to the underlying number of intersecting observations, or shares for weighting according to the intersecting expenditure shares. The default is none. Used only by geks().
- qbase : relevant only if type is one of (lowe, young, geoyoung), see bilateral.index.

Details

Before calculations start, missing values are removed from the data. Duplicated observations for r and n are aggregated, that is, duplicated prices p and weights w are averaged and duplicated quantities q added up. If there is more than one region in the data, products with prices in only one region r are removed.

The weights w must represent expenditure shares defined as $w_i^r = p_i^r q_i^r / \sum_{j=1}^{N} p_j^r q_j^r$. They are internally (re-)normalized such that they add up to 1 for each region r.

Value

For index.pairs(), a data.table with variables base (the base region), region (the comparison region), and the price level between the two regions (variable names defined by settings$type).

For geks(), a named vector or matrix of (unlogged) regional price levels if simplify=TRUE. Otherwise, for simplify=FALSE, a lm-object containing the full regression output.

Method

The GEKS method is a two-step approach. First, prices are aggregated into bilateral index numbers, P^{sr}, using the index formulas given in type. This is done for all pairs of regions s and r via the function `index.pairs()`. Second, these bilateral index numbers are transformed into transitive index numbers, P^r, by estimating the following regression model:

$$\ln P^{sr} = \ln \left(P^r / P^s\right) + u^{sr}$$

The quantities `q` or weights `w` are used within the aggregation of prices into index numbers (first stage) while the subsequent transformation of these index numbers (second stage) usually does not rely on any weights (but can if specified in `settings$wmethod`).

Author(s)

Sebastian Weinand

References

Gini, C. (1924). Quelques Considerations au Sujet de la Construction des Nombres Indices des Prix et des Questions Analogues. *Mentron*, 4 (1), 3-162.

Gini, C. (1931). On the Circular Test of Index Numbers. *International Statistical Review*, 9 (2), 3-25.

Elteto, O. and Koves, P. (1964). On a Problem of Index Number Computation Relating to International Comparison. *Statisztikai Szemle*, 42, 507-518.

Szulc, B. J. (1964). Indices for Multiregional Comparisons. *Przeglad Statystyczny*, 3, 239-254.

See Also

bilateral.index

Examples

```
# example data:
set.seed(123)
dt1 <- rdata(R=3, B=1, N=5)

### Index pairs

# matrix of bilateral index numbers:
Pje <- dt1[, index.pairs(p=price, r=region, n=product, settings=list(type="jevons"))]
# if the underlying index satisfies the country-reversal
# test (like the Jevons index), the price index numbers of
# the upper-right triangle are the same as the inverse of
# the price index numbers of the lower-left triangle.
all.equal(Pje$jevons[3], 1/Pje$jevons[7]) # true
# hence, one could set all.pairs=FALSE without loosing any
# information. however, this is no longer true for indices
# that do not satisfy this test (like the Carli index):
Pca <- dt1[, index.pairs(p=price, r=region, n=product, settings=list(type="carli"))]
all.equal(Pca$carli[3], 1/Pca$carli[7]) # false

### GEKS method
```

```
# for complete price data (no gaps), the jevons index is transitive.
# hence, no adjustment is needed by the geks approach, which is
# why the index numbers are the same:
all.equal(
  dt1[, geks(p=price, r=region, n=product, base="1", settings=list(type="jevons"))],
  dt1[, jevons(p=price, r=region, n=product, base="1")]
) # true

# this is no longer true when there are gaps in the data:
dt1.gaps <- dt1[!rgaps(region, product, amount=0.25), ]
all.equal(
  dt1.gaps[, geks(p=price, r=region, n=product, base="1", settings=list(type="jevons"))],
  dt1.gaps[, jevons(p=price, r=region, n=product, base="1")]
) # now, differences

# weighting at the second step of GEKS can be done with respect
# to the intersection of products for each pair of region:
dt1.gaps[, geks(p=price, r=region, n=product, base="1",
                settings=list(type="jevons", wmethod="obs"))]

# add price data:
dt2 <- rdata(R=4, B=1, N=4)
dt2[, "region":=factor(region, labels=4:7)]
dt2[, "product":=factor(product, labels=6:9)]
dt <- rbind(dt1, dt2)
dt[, is.connected(r=region, n=product)] # non-connected now

# compute all index pairs and geks:
require(data.table)
as.matrix(dcast(
  data=dt[, index.pairs(p=price, r=region, n=product)],
  formula=base~region,
  value.var="jevons"), rownames="base")
dt[, geks(p=price, r=region, n=product, base="1", settings=list(type="jevons"))]
```

`gerardi` *Gerardi index*

Description

Calculation of regional price levels using the multilateral Gerardi index (Eurostat, 1978).

Usage

```
gerardi(p, r, n, q, w=NULL, base=NULL, simplify=TRUE, settings=list())
```

Arguments

p	A numeric vector of positive prices.
r, n	A character vector or factor of regional entities r and products n, respectively.

q, w — A numeric vector of non-negative quantities q or expenditure share weights w (see Section 'Details'). If both q and w are provided, q will be used.

base — A character specifying the base region to which all price levels are expressed. When NULL, they refer to the (unweighted) regional average.

simplify — A logical indicating whether a named vector of estimated regional price levels (TRUE) should be returned, or also the average product prices.

settings — A list of control settings to be used. The following settings are supported:

- chatty : A logical specifying if warnings and info messages should be printed or not. The default is getOption("pricelevels.chatty").
- connect : A logical specifying if the data should be checked for connectedness or not. The default is getOption("pricelevels.connect"). If the data are not connected, price levels are computed within the biggest block of connected regions or the block of regions to which the base region belongs. See also connect().
- plot : A logical specifying if the calculated price levels should be plotted or not. If TRUE, the price ratios of each region are displayed as boxplots and the price levels are added as colored points. The default is getOption("pricelevels.plot").
- variant : for original, the international prices are calculated as unweighted geometric means. This is the original approach. With adjusted, the international prices are calculated as weighted geometric means.

Details

Before calculations start, missing values are removed from the data. Duplicated observations for r and n are aggregated, that is, duplicated prices p and weights w are averaged and duplicated quantities q added up. If there is more than one region in the data, products with prices in only one region r are removed.

The weights w must represent expenditure shares defined as $w_i^r = p_i^r q_i^r / \sum_{j=1}^N p_j^r q_j^r$. They are internally (re-)normalized such that they add up to 1 for each region r.

Value

For simplify=TRUE, a named vector of regional price levels. Otherwise, for simplify=FALSE, a list containing the named vector of international product prices and regional price levels.

Author(s)

Sebastian Weinand

References

Balk, B. M. (1996). A comparison of ten methods for multilateral international price and volume comparisons. *Journal of Official Statistics*, 12 (1), 199-222.

Eurostat (1978), *Comparison in real values of the aggregates of ESA 1975*, Publications Office, Luxembourg.

Examples

```
require(data.table)

# example data:
set.seed(123)
dt1 <- rdata(R=3, B=1, N=5)

# Gerardi price index:
dt1[, gerardi(p=price, q=quantity, r=region, n=product)]

# add price data:
dt2 <- rdata(R=4, B=1, N=4)
dt2[, "region":=factor(region, labels=4:7)]
dt2[, "product":=factor(product, labels=6:9)]
dt <- rbind(dt1, dt2)
dt[, is.connected(r=region, n=product)] # non-connected now

# compute expenditure share weights:
dt[, "share" := price*quantity/sum(price*quantity), by="region"]

# Gerardi index with quantites or expenditure share weights:
dt[, gerardi(p=price, q=quantity, r=region, n=product)]
dt[, gerardi(p=price, w=share, r=region, n=product)]
```

`gkhamis` *Multilateral systems of equations*

Description

Calculation of regional price levels using the

- Geary-Khamis method (Geary, 1958; Khamis, 1972): `gkhamis()`
- Iklé method (Iklé, 1972; Dikhanov, 1997; Balk, 1996): `ikle()`
- Rao system (Rao, 1990): `rao()`
- Rao-Hajargasht method (Rao and Hajargasht, 2016): `rhajargasht()`

Usage

```
gkhamis(p, r, n, q=NULL, base=NULL, simplify=TRUE, settings=list())

ikle(p, r, n, q=NULL, w=NULL, base=NULL, simplify=TRUE, settings=list())

rao(p, r, n, q=NULL, w=NULL, base=NULL, simplify=TRUE, settings=list())

rhajargasht(p, r, n, q=NULL, w=NULL, base=NULL, simplify=TRUE, settings=list())
```

Arguments

- `p` — A numeric vector of positive prices.
- `r, n` — A character vector or factor of regional entities `r` and products `n`, respectively.
- `q, w` — A numeric vector of non-negative quantities `q` or expenditure share weights `w` (see Section 'Details'). If both `q` and `w` are provided, `q` will be used. Note that `gkhamis()` does not use weights `w`.
- `base` — A character specifying the base region to which all price levels are expressed. When `NULL`, they refer to the (unweighted) regional average.
- `simplify` — A logical indicating whether a named vector of estimated regional price levels (`TRUE`) should be returned, or also the average product prices.
- `settings` — A list of control settings to be used. The following settings are supported:
  - `chatty` : A logical specifying if warnings and info messages should be printed or not. The default is `getOption("pricelevels.chatty")`.
  - `connect` : A logical specifying if the data should be checked for connectedness or not. The default is `getOption("pricelevels.connect")` for `geks()` and `FALSE` for `index.pairs()`. If the data are not connected, price levels are computed within the biggest block of connected regions or the block of regions to which the `base` region belongs. See also `connect()`.
  - `plot` : A logical specifying if the calculated price levels should be plotted or not. If `TRUE`, the price ratios of each region are displayed as boxplots and the price levels are added as colored points. The default is `getOption("pricelevels.plot")`.
  - `solve` : the method used for solving the system of equations. The default for all indices is `iterative` for iterative solving until convergence. For `gkhamis()`, the analytical solution proposed by Diewert (1999) is also allowed by setting to `matrix`.
  - `tol` : the tolerance level when convergence is achieved if `solve="iterative"`. The default is `1e-9`.
  - `max.iter` : the maximum number of iterations if `solve="iterative"`. The default is `99`.

Details

Before calculations start, missing values are removed from the data. Duplicated observations for `r` and `n` are aggregated, that is, duplicated prices `p` and weights `w` are averaged and duplicated quantities `q` added up. If there is more than one region in the data, products with prices in only one region `r` are removed.

The weights `w` must represent expenditure shares defined as $w_i^r = p_i^r q_i^r / \sum_{j=1}^N p_j^r q_j^r$. They are internally (re-)normalized such that they add up to 1 for each region `r`.

Value

For `simplify=TRUE`, a named vector of regional price levels.

For `simplify=FALSE`, a list containing the named vector of international product prices and regional price levels, the number of iterations until convergence, and the achieved difference at convergence.

Method

The Geary-Khamis, Iklé and Rao-Hajargasht methods as well as the Rao system have in common that they set up a system of interrelated equations of international product prices and price levels, which is solved iteratively. It is only the definition of the international product prices and price levels that differ between the methods (see also package vignette).

In their original form, the four methods use quantities (or weights). However, Rao and Hajargasht (2016, p. 417) show that similar solutions exist for the unweighted definitions of international product prices and price levels. In the package, this is implemented in the functions where

- `gkhamis(q=NULL)` corresponds to a multilateral Dutot index;
- `ikle(q=NULL, w=NULL)` to a multilateral Harmonic mean index;
- `rao(q=NULL, w=NULL)` to a multilateral Jevons index;
- `rhajargasht(q=NULL, w=NULL)` to a multilateral Carli index.

Author(s)

Sebastian Weinand

References

Balk, B. M. (1996). A comparison of ten methods for multilateral international price and volume comparisons. *Journal of Official Statistics*, 12 (1), 199-222.

Diewert, W. E. (1999). Axiomatic and Economic Approaches to International Comparisons. In: *International and Interarea Comparisons of Income, Output and Prices*, edited by A. Heston and R. E Lipsey. Chicago: The University of Chicago Press.

Dikhanov, Y. (1994). Sensitivity of PPP-based income estimates to the choice of aggregation procedures. The World Bank, Washington D.C., June 10, paper presented at 23rd General Conference of the International Association for Research in Income and Wealth, St. Andrews, Canada.

Geary, R. C. (1958). A Note on the Comparison of Exchange Rates and Purchasing Power Between Countries. *Journal of the Royal Statistical Society. Series A (General)*, 121 (1), 97–99.

Iklé, D. M. (1972). A new approach to the index number problem. *The Quarterly Journal of Economics*, 86 (2), 188-211.

Khamis, S. H. (1972). A New System of Index Numbers for National and International Purposes. *Journal of the Royal Statistical Society. Series A (General)*, 135 (1), 96–121.

Rao, D. S. P. (1990). A system of log-change index numbers for multilateral comparisons. In: *Comparisons of prices and real products in Latin America. Contributions to Economic Analysis Series*, edited by Salazar-Carrillo and Rao. Amsterdam: North-Holland Publishing Company.

Rao, D. S. P. and G. Hajargasht (2016). Stochastic approach to computation of purchasing power parities in the International Comparison Program. *Journal of Econometrics*, 191 (2016), 414-425.

Examples

```
require(data.table)

# example data:
set.seed(123)
```

```
dt1 <- rdata(R=3, B=1, N=5)

# Gery-Khamis price index can be obtained in two ways:
dt1[, gkhamis(p=price, q=quantity, r=region, n=product, settings=list(solve="iterative"))]
dt1[, gkhamis(p=price, q=quantity, r=region, n=product, settings=list(solve="matrix"))]

# gkhamis(), ikle() and gerardi() yield same results if quantites the same:
dt1[, "quantity2" := 1000*rleidv(product)]
dt1[, gkhamis(p=price, r=region, n=product, q=quantity2)]
dt1[, gerardi(p=price, r=region, n=product, q=quantity2)]
dt1[, ikle(p=price, r=region, n=product, q=quantity2)]
dt1[, "quantity2":=NULL]

# add price data:
dt2 <- rdata(R=4, B=1, N=4)
dt2[, "region":=factor(region, labels=4:7)]
dt2[, "product":=factor(product, labels=6:9)]
dt <- rbind(dt1, dt2)
dt[, is.connected(r=region, n=product)] # non-connected now

# compute expenditure share weights:
dt[, "share" := price*quantity/sum(price*quantity), by="region"]

# Ikle index with quantites or expenditure share weights:
dt[, ikle(p=price, q=quantity, r=region, n=product)]
dt[, ikle(p=price, w=share, r=region, n=product)]
```

---

pricedata *Price data characteristics*

Description

Price data typically consist of prices (and purchased quantities) for multiple products and regions. Since not all products are usually available in all regions, the data exhibit gaps. In some situations, the gaps can lead to non-connected data, which prevents a price comparison between all regions.

The following functions are available to derive the characteristics of a data set:

- `is.connected()` checks if all regions in the data are connected either directly or indirectly by some bridging region
- `neighbors()` divides the regions into groups of connected regions
- `connect()` is a simple wrapper of `neighbors()`, connecting the data using the group of regions with the maximum number of observations
- `gaps()` computes the (percentage) number of gaps in the data
- `pairs()` derives the number of available bilateral index pairs
- `properties()` derives data characteristics for each group of connected regions

Usage

```
is.connected(r, n)

neighbors(r, n, simplify=FALSE)

connect(r, n)

gaps(r, n, relative=TRUE)

pairs(r, n)

properties(r, n)
```

Arguments

`r, n`	A character vector or factor of regional entities `r` and products `n`, respectively.
`simplify`	A logical indicating whether the results should be simplified to a vector of group identifiers (`TRUE`) or not (`FALSE`). In the latter case the output will be a list of connected regions.
`relative`	A logical indicating whether the absolute (`FALSE`) or relative (`TRUE`) number of data gaps should be computed.

Details

Before calculations start, missing values are removed from the data. Duplicated observations for `r` and `n` are counted as one observation. Products with prices in only one region `r` do not provide meaningful information for interregional comparisons. Such products are therefore not considered by `gaps()`, `pairs()` and `properties()`. This approach follows the default treatment of all index functions in this package.

Following World Bank (2013, p. 98), a "price tableau is said to be connected if the price data are such that it is not possible to place the countries in two groups in which no item priced by any country in one group is priced by any other country in the second group".

Value

The function

- `is.connected()` prints a single logical indicating if the data is connected or not
- `connect()` returns a logical vector of the same length as `r` and `n`
- `neighbors()` gives a list or vector of connected regions
- `pairs()` returns a single numeric for the number of bilateral pairs
- `gaps()` returns a single numeric for the percentage of data gaps

The function `properties()` provides a data.table with the following variables:

`group`	*integer*	group identifier
`size`	*integer*	number of regions belonging to that group

`regions`	*list*	regions belonging to that group
`pairs`	*integer*	number of available non-redundant region pairs, e.g., (AB,AC,BC)=3
`nprods`	*integer*	number of unique products
`nobs`	*integer*	number of observations
`gaps`	*numeric*	percentage of data gaps

Author(s)

Sebastian Weinand

References

World Bank (2013). *Measuring the Real Size of the World Economy: The Framework, Methodology, and Results of the International Comparison Program.* Washington, D.C.: World Bank.

Examples

```
### connected price data:
set.seed(123)
dt1 <- rdata(R=4, B=1, N=3)

dt1[, is.connected(r=region, n=product)] # true
dt1[, neighbors(r=region, n=product, simplify=TRUE)]
dt1[, gaps(r=region, n=product)]
dt1[, pairs(r=region, n=product)]
dt1[, properties(r=region, n=product)]

### non-connected price data:
dt2 <- data.table::data.table(
          "region"=c("a","a","h","b","a","a","c","c","d","e","e","f",NA),
          "product"=c(1,1,"bla",1,2,3,3,4,4,5,6,6,7),
          "price"=runif(13,5,6),
          stringsAsFactors=TRUE)

dt2[, is.connected(r=region, n=product)] # false
with(dt2, neighbors(r=region, n=product))
dt2[, properties(region, product)]
# note that the first two observations are treated as one
# while the observation [NA,7] is dropped. Observation [a,2]
# is still included even though it does not provide valueable
# information for interregional comparisons (the product is
# observed in only one region)

# connect the price data:
dt2[connect(r=region, n=product),]
```

`pricelevels` *Spatial price indices*

Description

Calculation of multiple spatial price indices at once.

Usage

```
# list all available price indices:
list.indices()

# compute all price indices:
pricelevels(p, r, n, q=NULL, w=NULL, base=NULL, settings=list())
```

Arguments

`p`	A numeric vector of positive prices.
`r, n`	A character vector or factor of regional entities `r` and products `n`, respectively.
`q, w`	A numeric vector of non-negative quantities `q` or expenditure share weights `w` (see Section 'Details'). Either `q` or `w` must be provided for weighted indices. If both `q` and `w` are provided, `q` will be used.
`base`	A character specifying the base region to which all price levels are expressed. If `NULL`, base region is set internally.
`settings`	A list of control settings to be used. The following settings are supported: • `chatty` : A logical specifying if warnings and info messages should be printed or not. The default is `getOption("pricelevels.chatty")`. • `connect` : A logical specifying if the data should be checked for connectedness or not. The default is `getOption("pricelevels.connect")`. If the data are not connected, price levels are computed within the biggest block of connected regions or the block of regions to which the `base` region belongs. See also `connect()`. • `plot` : A logical specifying if the calculated price levels should be plotted or not. If `TRUE`, the price ratios of each region are displayed as boxplots and the price levels are added as colored points. The default is `getOption("pricelevels.plot")`. • `type` : A character specifying the index method(s) used to aggregate individual prices into price indices. See `list.indices()` for allowed values. The default is `NULL` in which case all possible price indices are computed. • `...` : Further settings allowed for the index methods. Note that `settings$solve` is always set to `iterative`.

Details

Before calculations start, missing values are removed from the data. Duplicated observations for `r` and `n` are aggregated, that is, duplicated prices `p` and weights `w` are averaged and duplicated quantities `q` added up. If there is more than one region in the data, products with prices in only one region `r` are removed.

The weights `w` must represent expenditure shares defined as $w_i^r = p_i^r q_i^r / \sum_{j=1}^{N} p_j^r q_j^r$. They are internally (re-)normalized such that they add up to 1 for each region `r`.

Value

A matrix of price levels where the rows contain the index methods and the columns the regions.

Author(s)

Sebastian Weinand

Examples

```
# sample complete price data:
set.seed(123)
dt1 <- rdata(R=3, B=1, N=5)

# compute specific unweighted price indices:
dt1[, pricelevels(p=price, r=region, n=product, base="1",
                  settings=list(type=c("jevons","cswd","bmw")))]

# compute all unweighted price indices:
dt1[, pricelevels(p=price, r=region, n=product, base="1")]

# compute the price indices using all methods:
dt1[, pricelevels(p=price, r=region, n=product, q=quantity, base="1")]

# add price data:
dt2 <- rdata(R=4, B=1, N=4)
dt2[, "region":=factor(region, labels=4:7)]
dt2[, "product":=factor(product, labels=6:9)]
dt <- rbind(dt1, dt2)
dt[, is.connected(r=region, n=product)] # non-connected now

# compute all unweighted indices:
dt[, pricelevels(p=price, r=region, n=product, base="1")]

# change base region:
dt[, pricelevels(p=price, r=region, n=product, base="4")]
```

ratios *Calculation of price ratios*

Description

Calculation of regional price ratios per product with flexible setting of base prices.

Usage

```
ratios(p, r, n, q=NULL, w=NULL, base=NULL, settings=list())
```

Arguments

- `p` A numeric vector of positive prices.
- `r, n` A character vector or factor of regional entities `r` and products `n`, respectively.
- `q, w` A numeric vector of non-negative quantities `q` or expenditure share weights `w`. If both `q` and `w` are provided, `q` will be used. This is only relevant for the averaging of duplicated prices (see Section 'Details').
- `base` A character specifying the base region to be used for the calculation of price ratios. If `NULL`, price ratios are calculated with reference to the regional average price of a product (see Section 'Details')
- `settings` A list of control settings to be used. The following settings are supported:
  - `chatty` : A logical specifying if warnings and info messages should be printed or not. The default is `getOption("pricelevels.chatty")`.
  - `static` : A logical indicating whether the `base` region is static (`TRUE`), that is, always the same, or if another region than `base` is allowed to be used when prices for `base` are not available or `NA` for specific products. Only relevant if `base` is not `NULL`. The default is `TRUE`.

Details

If there are $k = 1, \dots, K_n^r$ duplicated prices for product n in region r, these are averaged using the quantities `q` (or similarly as a weighted arithmetic mean using the weights `w`) if provided:

$$\bar{p}_n^r = \sum_{k=1}^{K_n^r} p_{nk}^r q_{nk}^r \Big/ \sum_{k=1}^{K_n^r} q_{nk}^r$$

Price ratios are then derived for each product n by $p_n^r \Big/ \frac{1}{R} \sum_{s=1}^{R} \bar{p}_n^s$ if `base=NULL` and by $p_n^r / \bar{p}_n^{base}$ otherwise.

Value

A numeric vector of the same length as `p`. If `base` is not `NULL` and `static=FALSE`, the attribute `attr(x, "base")` is added to the output, providing the respective base region for each product.

Author(s)

Sebastian Weinand

Examples

```
### (1) unique price observations

set.seed(123)
dt1 <- rdata(R=3, B=1, N=4)
levels(dt1$region) <- c("a","b","c")

# calculate price ratios by product:
dt1[, ratios(p=price, r=region, n=product, base="b")]


### (2) unique price observations and missing base region


# drop two observations:
dt2 <- dt1[-c(5,10), ]

# now, region 'a' is base for product 2:
dt2[, "pr" := ratios(p=price, r=region, n=product, base="b",
                     settings=list(static=FALSE))]

# base regions are stored in attributes:
attr(dt2$pr, "base")

# with static base, NAs are produced:
dt2[, "pr_static" := ratios(p=price, r=region, n=product, base="b")]


### (3) duplicated prices

# insert duplicates and missings:
dt3 <- rbind(dt1[c(2,3),], dt1[-c(11),])
dt3[1:2, c("price","quantity") := list(price*1.1, quantity*0.95)]
anyDuplicated(dt3, by=c("region","product"))

# duplicated prices are divided by the weighted average base prices:
dt3[, ratios(p=price, r=region, n=product, q=quantity, base="b",
             settings=list(static=FALSE))]
```

`rdata` *Simulate random price and quantity data*

Description

Simulate random price and quantity data for a specified number of regions $(r = 1, \dots, R)$, product groups $(b = 1, \dots, B)$, and individual products $(n = 1, \dots, N_b)$ using the function `rdata()`.

The generation of prices follows the NLCPD model (see `nlcpd()`), while expenditure share weights for product groups can be sampled using the function `rweights()`. Purchased quantities are assigned to individual products. Moreover, random sales and gaps (using the function `rgaps()`) can be introduced in the simulated data.

Usage

```
rgaps(r, n, amount=0, prob=NULL, pairs=FALSE, exclude=NULL)

rweights(r, b, type=~1)

rdata(R, B, N, gaps=0, weights=~b+r, sales=0, settings=list())
```

Arguments

`r, n, b` A character vector or factor of regional entities `r`, individual products `n`, and product groups (or basic headings) `b`, respectively.

`R, B, N` A single integer specifying the number of regions `R` and product groups `B`, respectively, and a vector of length `B` specifying the number of individual products `N` in each product group.

`weights, type` A formula specifying the sampling of expenditure share weights for product groups. If `type=~1`, product groups receive identical weights, while weights are product group specific for `type=~b`. If weights should vary among product groups and regions, use `type=~b+r`. As long as there are no data gaps, the weights add up to 1 for each region.

`gaps, sales, amount` Percentage amount of gaps and sales (between 0 and 1), respectively, to be introduced in the data.

`prob` A vector of probability weights, see also `sample()`. Either `NULL` or the same length as `r` and `n`. Larger values make gaps occur more likely at this position.

`pairs` A logical indicating if gaps should be introduced such that there are always at least two observations per product available (`pairs=TRUE`). Only in this case, all products provide valuable information for a spatial price comparison. Otherwise, if `pairs=FALSE`, there can be products with only one observation. See also the Details section.

`exclude` Data.frame of two (character) variables `r` and `n`, specifying regions and products to be excluded from introducing gaps. Default is `NULL`, meaning that gaps are allowed to occur in all regions and products present in the data. Missing values (`NA`) are translated into no gaps for the corresponding product or region, e.g. `data.frame(r="r1", n=NA)` means that there will be no gaps in region `r1`.

`settings` A list of control settings to be used. The following settings are supported:

- `gaps.prob` : See argument `prob`.
- `gaps.pairs` : See argument `pairs`.
- `gaps.exclude` : See argument `exclude`.
- `sales.max.rebate` : Maximum allowed percentage price rebate for a sale (between 0.001 and 1). The default is `0.25`, meaning that prices may be reduced by no more than 25%.

- sales.max.qi : Maximum allowed quantity increase for a sale (between 1 and Inf). The default is 2, meaning that quantities may double at most.
- par.sd : named vector specifying the standard deviations used for sampling true parameters and errors. The default is c(lnP=0.1, pi=exp(1), delta=0.5, error=0.01).
- par.add : logical, specifying if the parameters underlying the data generating process should be added the function output. This is particularly useful if rdata() is applied in simulations. Default is FALSE.
- round : logical, specifying if prices should be rounded to two decimals or not. While prices usually have two decimal places in reality, this rounding can cause small differences between estimated and true parameter values. For simulation purposes, it is therefore recommended to use unrounded prices by setting round=FALSE.

Details

The function rgaps() ensures that gaps do not lead to non-connected price data (see is.connected()). Therefore, it could happen that the amount of gaps specified in rgaps() is only approximate, in particular, in cases where certain regions and/or products should additionally be excluded from exhibiting gaps by exclude.

If rgaps(pairs=FALSE), the minimum number of observations for a connected data set is $R+N-1$. Otherwise, for rgaps(pairs=TRUE), this number is defined by $2N+\max(0,R-N-1)$.

Note that setting sales>0 in function rdata() distorts the initial price generating process. Consequently, parameter estimates may deviate stronger from their true values. Note also that the expenditure share weights weight represent the relevance of product groups as (often) derived from national accounts and other data sources. Therefore, they cannot be derived from the simulated prices and quantities in the data, which would represent the expenditure shares of the individual products.

Value

Function rgaps() returns a logical vector of the same length as r where TRUEs indicate gaps and FALSEs no gaps.

Function rweights() returns a numeric vector of (non-negative) expenditure share weights of the same length as r.

Function rdata() returns a data.table with the following variables:

group	product group identifier (factor)
weight	expenditure share weight of product groups (numeric)
region	region identifier (factor)
product	product identifier (factor)
sale	are prices and quantities affected by sales? (logical)
price	price (numeric)
quantity	consumed quantity (numeric)

or a list with the simulated data and its underlying parameter values, if settings=list(par.add=TRUE).

Author(s)

Sebastian Weinand

Examples

```
# simulate price data for ten regions and five product groups
# containing three individual products each:
set.seed(1)
dt <- rdata(R=10, B=5, N=3)
boxplot(price~paste(group, product, sep=":"), data=dt)

# simulate price data for ten regions and five product groups
# containing one to five individual products:
set.seed(1)
dt <- rdata(R=10, B=5, N=c(1,2,3,4,5))
boxplot(price~paste(group, product, sep=":"), data=dt)

# simulate price data for three product groups (with one
# product each) in four regions:
dt <- rdata(R=4, B=3, N=1)

# add expenditure share weights:
dt[, "w1" := rweights(r=region, b=group, type=~1)] # constant
dt[, "w2" := rweights(r=region, b=group, type=~b)] # product-specific
dt[, "w3" := rweights(r=region, b=group, type=~b+r)] # product-region-specific

# weights add up to 1:
dt[, list("w1"=sum(w1),"w2"=sum(w2),"w3"=sum(w3)), by="region"]

# introduce 25% random gaps:
dt.gaps <- dt[!rgaps(r=region, n=product, amount=0.25), ]

# weights no longer add up to 1 in each region:
dt.gaps[, list("w1"=sum(w1),"w2"=sum(w2),"w3"=sum(w3)), by="region"]

# approx. 25% random gaps, but keep observation for product "n2"
# in region "r1" and all observations in region "r2":
no_gaps <- data.frame(r=c("r1","r2"), n=c("n2",NA))

# apply to data:
dt[!rgaps(r=region, n=product, amount=0.25, exclude=no_gaps), ]

# or, directly, in one step:
dt <- rdata(R=4, B=3, N=1, gaps=0.25, settings=list("gaps.exclude"=no_gaps))

# introduce systematic gaps:
dt <- rdata(R=15, B=1, N=10)
dt[, "prob" := data.table::rleidv(product)] # probability for gaps increases per product
dt.gaps <- dt[!rgaps(r=region, n=product, amount=0.25, prob=prob), ]
plot(table(dt.gaps$product), type="l")
```

Index