Package 'IndexNumR'

March 4, 2020

Type Package

Title Index Number Calculation

Version 0.1.3

Author Graham White

Maintainer Graham White <g.white@unswalumni.com>

Description Computes bilateral and multilateral index numbers.
It has support for several standard bilateral indices as well as
the GEKS multilateral index number methods
(see Ivancic, Diewert and Fox (2011) <doi:10.1016/j.jeconom.2010.09.003>) .
It also supports updating of GEKS indexes using several splicing methods.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Depends R (>= 3.5.0)

Imports utils

Suggests testthat, knitr, rmarkdown, covr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-04 21:30:02 UTC

R topics documented:

`CES_sigma_2` *Dataset of prices and quantities on four products*

Description

A constructed dataset containing the prices and quantities of four products over a twelve month period, assuming CES preferences.

Usage

```
CES_sigma_2
```

Format

A data frame with 48 rows and 4 columns:

time time period

prices constructed prices

quantities constructed quantities

prodID product identifier

Source

Computed using procedure in W.E. Diewert and K.J. Fox (2017), "Substitution Bias in Multilateral Methods for CPI Construction Using Scanner Data", Discussion Paper 17-02, Vancouver School of Economics, The University of British Columbia.

`elasticity` *Computes the elasticity of substitution*

Description

A function to estimate the elasticity of substitution

Usage

```
elasticity(x, pvar, qvar, pervar, prodID, compIndex = "ces",
  lower = -20, upper = 20)
```

Arguments

`x`	A dataframe
`pvar`	A character string for the name of the price variable
`qvar`	A character string for the name of the quantity variable
`pervar`	A character string for the name of the time variable. This variable must contain integers starting at period 1 and increasing in increments of 1 period. There may be observations on multiple products for each time period.
`prodID`	A character string for the name of the product identifier
`compIndex`	The index number with which the CES index will be equated to calcuate the elasticity. Acceptable options are lloydmoulton, fisher or satovartia. The lloyd-moulton option equates the 'base period' lloyd-moulton index with the 'current period' lloyd-moulton index.
`lower`	lower limit to search for sigma.
`upper`	upper limit to search for sigma.

Value

A list with three elements: sigma (the average elasticity over all time periods); allsigma (a T-1 by 1 matrix of the estimated elasticities for each time period, except period one); and diff (the value of the difference between the two indexes, check this is zero for all time periods).

Examples

```
elasticity(CES_sigma_2,pvar="prices",qvar="quantities",pervar="time",
prodID = "prodID")
```

`evaluateMatched` *Evaluate product overlap between periods*

Description

Evaluate the counts and expenditure for each period with and without matching items across periods.

Usage

```
evaluateMatched(x, pvar, qvar, pervar, prodID, output = "chained")
```

Arguments

`x`	A dataframe containing price, quantity, a time period identifier and a product identifier. It must have column names.
`pvar`	A character string for the name of the price variable
`qvar`	A character string for the name of the quantity variable
`pervar`	A character string for the name of the time variable. This variable must contain integers starting at period 1 and increasing in increments of 1 period. There may be observations on multiple products for each time period.
`prodID`	A character string for the name of the product identifier
`output`	A character string specifying whether the matching should be done assuming a chained index or a fixed base index. No index is actually computed, but the matching needs to know which periods are being compared. Default is chained.

Value

A list of two matrices, one for expenditures and one for counts. Each matrix has eight columns. The first four columns present the base period information base_index (the index of the base period), base (base period expenditure or count), base_matched (the expenditure or count of the base period after matching), base_share (share of total expenditure in the base period that remains after matching). Columns 5-8 are defined analagously for the current period. The matched numbers for the base period should be interpreted as the count or expenditure that remains after removal of products that exist in the base period, but not in the current period. That is, products that existed in the base period but no longer exist in the current period are removed by the matching. If new products exist in the current period that were not available in the base period, this does not affect the matched base period expenditure or count. The appearance of new products is captured in the current period matched expenditure and counts. Therefore, a base period share that is less than 1 indicates that products have disappeared, while a current period share less than 1 indicates that new products have appeared.

Examples

```
# create CES_sigma_2 dataset removing the observation in time period 4
# on product 1
df <- CES_sigma_2[!(CES_sigma_2$time==4 & CES_sigma_2$prodID==1),]
# evaluate the overlap between periods for this dataset assuming
# a chained index
evaluateMatched(df, pvar="prices", qvar="quantities", pervar="time",
prodID = "prodID", output="chained")
```

GEKSIndex *Compute a GEKS multilateral index*

Description

A function to calculate a GEKS multilateral price index

Usage

```
GEKSIndex(x, pvar, qvar, pervar, indexMethod = "tornqvist", prodID,
  sample = "matched", window = 13, splice = "mean")
```

Arguments

x	A dataframe containing price, quantity, a time period identifier and a product identifier. It must have column names.
pvar	A character string for the name of the price variable
qvar	A character string for the name of the quantity variable
pervar	A character string for the name of the time variable. This variable must contain integers starting at period 1 and increasing in increments of 1 period. There may be observations on multiple products for each time period.
indexMethod	A character string to select the index number method. Valid index number methods are fisher or tornqvist. The default is tornqvist.
prodID	A character string for the name of the product identifier
sample	A character string specifying whether matching is to be performed. The default is to use matching. If sample=matched then any products that are not present in comparison periods are removed prior to estimating the index for those periods.
window	An integer specifying the length of the GEKS window.
splice	A character string specifying the splicing method. Valid methods are window, movement or mean. The default is mean.

Details

The splicing methods are used to update the price index when new data become available without changing prior index values. The window and movement splicing methods first calculate an 'update factor' by calculating the ratio of the final index value in the new GEKS window to some base period and then multiply the relevant old GEKS index value by the update factor. Alternatives are "window","movement","half", and "mean". See the package vignette for more information.

References

Ivancic, L., W.E. Diewert and K.J. Fox (2011), "Scanner Data, Time Aggregation and the Construction of Price Indexes", Journal of Econometrics 161, 24-35.

Examples

```
# compute a GEKS mutlilateral index with mean splicing
GEKSIndex(CES_sigma_2, pvar = "prices", qvar = "quantities", pervar = "time",
prodID = "prodID", indexMethod = "tornqvist", window=11, splice = "mean")

# compute a GEKS multilateral index with window splicing and the Fisher index method
GEKSIndex(CES_sigma_2, pvar = "prices", qvar = "quantities", pervar = "time",
prodID = "prodID", indexMethod = "fisher", window=11, splice = "mean")
```

---

maximumSimilarityLinks

Finds periods to link using minimum dissimilarity.

Description

Function to compute the maximum similarity chain links from a measure of dissimilarity. The procedure works as described in Diewert and Fox (2017). It first links period 2 to period 1. Then for each period t, from periods 3,...,T it searches among the periods 1,...,t-1 for the period that is most similar (least dissimilar) to period t.

Usage

```
maximumSimilarityLinks(x)
```

Arguments

x	a matrix containing a dissimilarity measure where the first two columns are the indices and the third column is the dissimilarity measure.

Examples

```
# find the linking periods in the CES_sigma_2 dataset that maximise
# the similarity between periods, using the absolute dissimilarity measure.
disMat <- mixScaleDissimilarity(CES_sigma_2, pvar = "prices", qvar = "quantities",
pervar = "time", prodID = "prodID", measure = "absolute",
combine = "geomean")
maximumSimilarityLinks(disMat)
```

`mixScaleDissimilarity` *Computes mix, scale and absolute dissimilarity measures*

Description

This is a function to compute the Fox, Hill and Diewert 2004 dissimilarity measures.

Usage

```
mixScaleDissimilarity(x, pvar, qvar, prodID, pervar,
  measure = "absolute", combine = "geomean")
```

Arguments

`x`	A dataframe
`pvar`	string identifying the price variable in x
`qvar`	string identifying the quantity variable in x
`prodID`	string identifying the product id variable in x
`pervar`	string identifying the time period variable in x
`measure`	choice of dissimilarity measure. Valid options are mix, scale or absolute.
`combine`	specifies how to combine the price and quantity vectors. "stack" stacks the price and quantity vectors, "geomean" computes separate dissimilarity measures for prices and quantities then takes the geometric mean of these.

Value

A matrix where the first two columns are the possible combinations of periods and the third column is the dissimilarity measure.

References

Fox, K.J., R.J. Hill and W.E. Diewert (2004), "Identifying outliers in multi-output models", Journal of Productivity Analysis, 22, 73-94, 2004.

Examples

```
# estimate the dissimilarity between periods in the CES_sigma_2 dataset
# using the absolute measure of dissimilarity and the geometric mean
# to combine price and quantity information.
mixScaleDissimilarity(CES_sigma_2, pvar = "prices", qvar = "quantities",
pervar = "time", prodID = "prodID", measure = "absolute",
combine = "geomean")
```

monthIndex *Generate an index of months*

Description

A function to create a month index variable

Usage

```
monthIndex(x, overlapWeeks = "naive")
```

Arguments

x	A vector or column of dates
overlapWeeks	Tells monthIndex how to deal with weeks that cross over two adjacent months. Options are "naive", "majority", "wholeOnly" or "fourWeek". "naive" simply takes the month number of the observation, ignoring where the week of that observation falls. "majority" will allocate the observation to the month that owns the majority of days in that week, assuming that Monday is day one of the week. "fourWeek" first calculates a week index, then calculates the month index assuming that there are four weeks in each month. "wholeOnly" will return NA for any dates falling inside a week that overlaps two adjacent months; that is, only weeks that are wholly within a month are given an index value. The default is "naive".

Examples

```
# given a vector of dates
df <- data.frame(date = as.Date(c("2017-01-01","2017-02-01","2017-03-01","2017-04-01"),
format = "%Y-%m-%d"))
# calculate the time period variable
df$period <- monthIndex(df$date, overlapWeeks = "naive")
df
```

priceIndex *Computes a bilateral price index*

Description

A function to compute a price index given data on products over time

Usage

```
priceIndex(x, pvar, qvar, pervar, indexMethod = "laspeyres", prodID,
  sample = "matched", output = "pop", chainMethod = "pop",
  sigma = 1.0001, ...)
```

Arguments

x	A dataframe containing price, quantity, a time period identifier and a product identifier. It must have column names.
pvar	A character string for the name of the price variable
qvar	A character string for the name of the quantity variable
pervar	A character string for the name of the time variable. This variable must contain integers starting at period 1 and increasing in increments of 1 period. There may be observations on multiple products for each time period.
indexMethod	A character string to select the index number method. Valid index number methods are dutot, carli, jevons, laspeyres, paasche, fisher, cswd, harmonic, tornqvist, satovartia, walsh, CES, geomLaspeyres and geomPaasche.
prodID	A character string for the name of the product identifier
sample	A character string specifying whether a matched sample should be used.
output	A character string specifying whether a chained (output="chained") , fixed base (output="fixedBase") or period-on-period (output="pop") price index numbers should be returned. Default is period-on-period.
chainMethod	A character string specifying the method of chain linking to use if the output option is set to "chained". Valid options are "pop" for period-on-period, and similarity chain linked options "plspread" for the Paasche-Laspeyres spread, "asymplinear" for weighted asymptotically linear, "logquadratic" for the weighted log-quadratic, and "mixScale" for the mix, scale or absolute dissimilarity measures. The default is period-on-period. Additional parameters can be passed to the mixScaleDissimilarity function using ...
sigma	The elasticity of substitution for the CES index method.
...	this is used to pass additional parameters to the mixScaleDissimilarity function.

Examples

```
# period-on-period Laspeyres index for the CES_sigma_2 dataset
priceIndex(CES_sigma_2, pvar="prices", qvar="quantities", pervar="time",
prodID = "prodID", indexMethod = "laspeyres")

# chained Fisher index
priceIndex(CES_sigma_2, pvar="prices", qvar="quantities", pervar="time",
prodID = "prodID", indexMethod = "fisher", output="chained")

# chained Tornqvist index, with linking periods chosen by the
# weighted log-quadratic dissimilarity measure
priceIndex(CES_sigma_2, pvar="prices", qvar="quantities", pervar="time",
prodID = "prodID", indexMethod = "tornqvist", output="chained",
chainMethod = "logquadratic")
```

`priceIndicator` *Calculate a price indicator*

Description

This calculates a price indicator. This is calculated using the differences approach to index number theory, where the change in prices and quantities from one period to the next is additive. Therefore, the change in total value is the sum of the change in prices and the change in quantities. Such a value decomposition can be obtained using `valueDecomposition`.

See the vignette for more information on the calculations.

```
vignette(topic = "indexnumr",package = "IndexNumR")
```

Usage

```
priceIndicator(x, pvar, qvar, pervar, prodID, method, sample = "matched")
```

Arguments

`x`	data frame with input data
`pvar`	character string for the name of the price column
`qvar`	character string for the name of the quantity column
`pervar`	character string for the name of the time period variable
`prodID`	character string for the name of the product ID column
`method`	character string for the indicator method. Valid options are "laspeyres", "paasche", "bennet", or "montgomery".
`sample`	whether to use a matched sample (sample = "matched")

Value

an nx1 matrix containing the indicator

Examples

```
# compute a price indicator using the Montgomery method
priceIndicator(CES_sigma_2, pvar = "prices", qvar = "quantities",
prodID = "prodID", pervar = "time", method = "montgomery")
```

quantityIndex *Computes a bilateral quantity index*

Description

A function to compute a quantity index given data on products over time

Usage

```
quantityIndex(x, pvar, qvar, pervar, indexMethod = "laspeyres", prodID,
  sample = "matched", output = "pop", chainMethod = "pop",
  sigma = 1.0001, ...)
```

Arguments

x	A dataframe containing price, quantity, a time period identifier and a product identifier. It must have column names.
pvar	A character string for the name of the price variable
qvar	A character string for the name of the quantity variable
pervar	A character string for the name of the time variable. This variable must contain integers starting at period 1 and increasing in increments of 1 period. There may be observations on multiple products for each time period.
indexMethod	A character string to select the index number method. Valid index number methods are dutot, carli, jevons, laspeyres, paasche, fisher, cswd, harmonic, tornqvist, satovartia, walsh and CES.
prodID	A character string for the name of the product identifier
sample	A character string specifying whether a matched sample should be used.
output	A character string specifying whether a chained, fixed base or period-on-period price index numbers should be returned.
chainMethod	A character string specifying the method of chain linking to use if the output option is set to "chained". Valid options are "pop" for period-on-period, and similarity chain linked options "plspread" for the Paasche-Laspeyres spread, "asymplinear" for weighted asymptotically linear, "logquadratic" for the weighted log-quadratic, and "mixScale" for the mix, scale or absolute dissimilarity measures. The default is period-on-period. Additional parameters can be passed to the mixScaleDissimilarity function using ...
sigma	The elasticity of substitution for the CES index method.
...	this is used to pass additional parameters to the mixScaleDissimilarity function.

Examples

```
# chained Fisher quantity index for the CES_sigma_2 dataset
quantityIndex(CES_sigma_2, pvar="prices", qvar="quantities", pervar="time",
prodID = "prodID", indexMethod = "fisher", output="chained")
```

quantityIndicator *Compute a quantity indicator*

Description

This calculates a quantity indicator. This is calculated using the differences approach to index number theory, where the change in prices and quantities from one period to the next is additive. Therefore, the change in total value is the sum of the change in prices and the change in quantities. Such a value decomposition can be obtained using `valueDecomposition`.

See the vignette for more information on the calculations.

```
vignette(topic = "indexnumr",package = "IndexNumR")
```

Usage

```
quantityIndicator(x, pvar, qvar, pervar, prodID, method,
  sample = "matched")
```

Arguments

x	data frame with input data
pvar	character string for the name of the price column
qvar	character string for the name of the quantity column
pervar	character string for the name of the time period variable
prodID	character string for the name of the product ID column
method	character string for the quantity indicator method. Valid options are "laspeyres", "paasche", "bennet", or "montgomery".
sample	whether to use a matched sample (sample = "matched")

Value

an nx1 matrix containing the indicator

Examples

```
# compute a quantity indicator using the Bennet method
quantityIndicator(CES_sigma_2, pvar = "prices", qvar = "quantities",
prodID = "prodID", pervar = "time", method = "bennet")
```

quarterIndex *Generate an index of quarters*

Description

A function to create a quarter index variable

Usage

```
quarterIndex(x)
```

Arguments

x	A vector or column of dates

Examples

```
# given a vector of dates
df <- data.frame(date = as.Date(c("2017-01-01","2017-04-01","2017-07-01","2017-08-01"),
format = "%Y-%m-%d"))
# calculate the time period variable
df$period <- quarterIndex(df$date)
df
```

relativeDissimilarity *Computes measures of relative dissimilarity between all periods*

Description

A function to compute the relative price dissimilarity between two vectors of prices.

Usage

```
relativeDissimilarity(x, pvar, qvar, pervar, prodID,
  indexMethod = "fisher", similarityMethod = "logquadratic")
```

Arguments

x	A dataframe containing price, quantities, a time period index and a product identifier.
pvar	A string identifying the price variable.
qvar	A string identifying the quantity variable.
pervar	A string identifying the time index variable.
prodID	A string identifying the product ID.

indexMethod — A string identifying the index method to use in the calculation. Not relevent for similarityMethod = PLSpread. Supported methods are fisher and tornqvist. Default is Fisher.

similarityMethod — A string specifying the formula for calculating the relative dissimilarity. Valid options are logquadratic, asymplinear and PLSpread. Default is logquadratic.

Value

A matrix of dissimilarity measures. The first two columns are the possible combinations of bilateral comparisons and the third column is the dissimilarity measure.

References

Diewert, W.E. (2002). "Similarity and Dissimilarity Indexes: An Axiomatic Approach" Discussion Paper No. 0210, Department of Economics, University of British Columbia.

Examples

```
# estimate the dissimilarity between periods in the CES_sigma_2 dataset
# using the log quadratic measure of dissimilarity
relativeDissimilarity(CES_sigma_2, pvar = "prices", qvar="quantities",
pervar = "time", prodID = "prodID", indexMethod="fisher",
similarityMethod = "logquadratic")
```

---

unitValues *Aggregates prices to unit values and quantities to sums*

Description

A function to aggregate price and quantity data to unit values

Usage

```
unitValues(x, pvar, qvar, pervar, prodID)
```

Arguments

x — A dataframe containing price, quantity, a time period identifier and a product identifier. It must have column names.

pvar — A character string for the name of the price variable

qvar — A character string for the name of the quantity variable

pervar — character string for the name of the time variable. This variable must contain integers starting at period 1 and increasing in increments of 1 period. There may be observations on multiple products for each time period.

prodID — A character string for the name of the product identifier

Value

A dataframe containing columns for product identifier, time period, quantities, and unit values.

Examples

```
# suppose the CES_sigma_2 dataset contains 12 monthly observations
# and suppose we want quarterly unit values.
df <- CES_sigma_2
# convert the monthly time variable into quarterly
df$time <- ceiling(CES_sigma_2$time/3)
# compute unit values using the quarterly time variable
unitValues(df,pvar="prices",qvar="quantities",pervar="time",prodID="prodID")
```

---

valueDecomposition *valueDecomposition*

Description

Perform a decomposition of value change using price and quantity indicators. This is an additive decomposition so that change due to price plus change due to quantity equals the total value change.

Usage

```
valueDecomposition(x, pvar, qvar, pervar, prodID, priceMethod,
  sample = "matched")
```

Arguments

x	data frame with input data
pvar	character string for the name of the price column
qvar	character string for the name of the quantity column
pervar	character string for the name of the time period variable
prodID	character string for the name of the product ID column
priceMethod	character string for the price indicator method. Valid options are "laspeyres", "paasche", "bennet", or "montgomery". This parameter also determines the method used for the quantity indicator. If a laspeyres price indicator is chosen, then a paasche quantity indicator is used. If a paasche price indicator is used then a laspeyres quantity indicator is used. For bennet and montgomery indicators, the same method is used for both the price and quantity indicators.
sample	whether to use a matched sample (sample = "matched")

Value

a dataframe containing the price indicator, quantity indicator the value change and the value level.

Examples

```
# decompose the value changes in the CES_sigma_2 dataset using the Bennet method
valueDecomposition(CES_sigma_2, pvar = "prices", qvar = "quantities",
prodID = "prodID", pervar = "time", priceMethod = "bennet")
```

---

values *Compute values (price x quantity)*

Description

Compute the total value (expenditure), for each time period in the sample.

Usage

```
values(x, pvar, qvar, pervar, prodID, sample = "matched",
  matchPeriod = "previous")
```

Arguments

x	A dataframe containing price, quantity, a time period identifier and a product identifier. It must have column names.
pvar	A character string for the name of the price variable
qvar	A character string for the name of the quantity variable
pervar	A character string for the name of the time variable. This variable must contain integers starting at period 1 and increasing in increments of 1 period. There may be observations on multiple products for each time period.
prodID	A character string for the name of the product identifier
sample	A character string specifying whether a matched sample should be used.
matchPeriod	A character string specifying which period is used to determine the set of products used for matching. Options are "following" or "previous". "following" calculates the expenditures in the current period, filtering out any products that do not appear in the following period. "previous" is calculated similarly, using the set of products in the previous period to filter the current period sample.

Examples

```
values(CES_sigma_2, pvar = "prices", qvar = "quantities", pervar = "time",
prodID = "prodID", matchPeriod = "previous")
```

weekIndex *Generate an index of weeks*

Description

Function to create a week index variable with weeks determined as defined in ISO 8601. If the week (starting on Monday) containing 1 January has four or more days in the new year, then it is considered week 1. Otherwise, it is the 53rd week of the previous year, and the next week is week 1.

Usage

```
weekIndex(x)
```

Arguments

x	A vector of dates

Examples

```
# given a vector of dates
df <- data.frame(date = as.Date(c("2016-12-20","2016-12-27","2017-01-01","2017-01-07"),
format = "%Y-%m-%d"))
# calculate the time period variable
df$period <- weekIndex(df$date)
df
```

yearIndex *Generate an index of years*

Description

Function to create a year index variable

Usage

```
yearIndex(x)
```

Arguments

x	A vector or column of dates

Examples

```
# given a vector of dates
df <- data.frame(date = as.Date(c("2017-01-01","2018-04-01","2019-07-01","2019-08-01"),
format = "%Y-%m-%d"))
# calculate the time period variable
df$period <- yearIndex(df$date)
df
```

Index