Title Calculating Bilateral and Multilateral Price Indexes

Version 0.2.0


Depends R (>= 3.5.0)

Imports lubridate (>= 1.7.4), dplyr (>= 0.8.3), ggplot2 (>= 3.2.0), reshape, reclin2, stringr, xgboost, caret, strex

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Jacek Bialek [aut, cre] (<https://orcid.org/0000-0002-0952-5327>)

Maintainer Jacek Bialek <jacek.bialek@uni.lodz.pl>

Repository CRAN

Date/Publication 2024-08-16 10:00:06 UTC

Contents

agmean .......................................................... 6
available ........................................................ 7
banajree ....................................................... 8
bennet .......................................................... 9
bialek .......................................................... 10
bmw ............................................................ 11
carli ............................................................ 12
ccdi .......................................................... 13
ccdi_fbew ................................................. 14
ccdi_bmw ................................................. 15
ccdi_splice ............................................... 16
chagmean ................................................ 18
chbanajre ............................................... 19
chbialek ................................................ 20
chbmw ..................................................... 21
chcarli ................................................... 22
chcswd .................................................... 23
chdavies .................................................. 24
chdikhanov .............................................. 25
chdrobisch ............................................... 26
chdutot .................................................... 27
chfisher ................................................... 28
chgeary_khamis ......................................... 29
chgeohybrid ............................................ 30
chgeolaspeyres ......................................... 31
chgeolowe ................................................ 32
chgeopaasche .......................................... 33
chgeoyoung ............................................. 34
chharmonic ............................................... 35
chhybrid .................................................. 36
chiQMp .................................................... 37
chjevons .................................................. 38
chlaspeyres ............................................. 39
chlehr .................................................... 40
chlloyd_moulton ........................................ 41
chlowe ..................................................... 42
chmarshall_edgeworth ................................ 43
chpaasche ................................................ 44
chpalgrave ............................................... 45
chQMp ..................................................... 46
chQMq ..................................................... 47
chsato_vartia ........................................... 48
chstuvel .................................................. 49
chtornqvist .............................................. 50
chvartia .................................................. 51
chwalsh ................................................... 52
chyoung ................................................... 53
coffee ...................................................... 54
compare_distances .................................... 55
compare_indices_df ................................... 56
compare_indices_jk ................................... 56
compare_indices_list ................................ 59
compare_to_target .................................... 60
cswd ...................................................... 61
dataAGGR ................................................ 62
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>geksj</td>
<td>118</td>
</tr>
<tr>
<td>geksj_fbew</td>
<td>119</td>
</tr>
<tr>
<td>geksj_fbmw</td>
<td>120</td>
</tr>
<tr>
<td>geksj_splice</td>
<td>121</td>
</tr>
<tr>
<td>geksl</td>
<td>123</td>
</tr>
<tr>
<td>gekslm</td>
<td>124</td>
</tr>
<tr>
<td>gekslm_fbew</td>
<td>125</td>
</tr>
<tr>
<td>gekslm_fbmw</td>
<td>126</td>
</tr>
<tr>
<td>gekslm.splice</td>
<td>127</td>
</tr>
<tr>
<td>geksl_fbew</td>
<td>129</td>
</tr>
<tr>
<td>geksl_fbmw</td>
<td>130</td>
</tr>
<tr>
<td>geksl_splice</td>
<td>131</td>
</tr>
<tr>
<td>geksqm</td>
<td>133</td>
</tr>
<tr>
<td>geksqm_fbew</td>
<td>134</td>
</tr>
<tr>
<td>geksqm_fbmw</td>
<td>135</td>
</tr>
<tr>
<td>geksqm.splice</td>
<td>136</td>
</tr>
<tr>
<td>geksw</td>
<td>138</td>
</tr>
<tr>
<td>geksw_fbew</td>
<td>139</td>
</tr>
<tr>
<td>geksw_fbmw</td>
<td>140</td>
</tr>
<tr>
<td>geksw_splice</td>
<td>141</td>
</tr>
<tr>
<td>geks_fbew</td>
<td>143</td>
</tr>
<tr>
<td>geks_fbmw</td>
<td>144</td>
</tr>
<tr>
<td>geks_splice</td>
<td>145</td>
</tr>
<tr>
<td>generate</td>
<td>146</td>
</tr>
<tr>
<td>generate_CES</td>
<td>148</td>
</tr>
<tr>
<td>geohybrid</td>
<td>149</td>
</tr>
<tr>
<td>geolaspeyres</td>
<td>150</td>
</tr>
<tr>
<td>geolowe</td>
<td>151</td>
</tr>
<tr>
<td>geopaasche</td>
<td>152</td>
</tr>
<tr>
<td>geoyoung</td>
<td>153</td>
</tr>
<tr>
<td>gk</td>
<td>154</td>
</tr>
<tr>
<td>gk_fbew</td>
<td>155</td>
</tr>
<tr>
<td>gk_fbmw</td>
<td>156</td>
</tr>
<tr>
<td>gk_splice</td>
<td>157</td>
</tr>
<tr>
<td>harmonic</td>
<td>159</td>
</tr>
<tr>
<td>hybrid</td>
<td>160</td>
</tr>
<tr>
<td>IQMp</td>
<td>161</td>
</tr>
<tr>
<td>jevons</td>
<td>162</td>
</tr>
<tr>
<td>laspeyres</td>
<td>163</td>
</tr>
<tr>
<td>lehr</td>
<td>164</td>
</tr>
<tr>
<td>lloyd_moulton</td>
<td>165</td>
</tr>
<tr>
<td>load_model</td>
<td>166</td>
</tr>
<tr>
<td>lowe</td>
<td>167</td>
</tr>
<tr>
<td>marshall_edgeworth</td>
<td>168</td>
</tr>
<tr>
<td>matched</td>
<td>169</td>
</tr>
<tr>
<td>matched_fig</td>
<td>170</td>
</tr>
<tr>
<td>matched_index</td>
<td>171</td>
</tr>
<tr>
<td>mbennet</td>
<td>172</td>
</tr>
</tbody>
</table>
Contents

milk ................................................................. 173
mmontgomery ....................................................... 174
model_classification ............................................. 175
montgomery ......................................................... 177
paasche ............................................................. 178
palgrave ............................................................ 179
pqcor ............................................................... 180
pqcor_fig ......................................................... 181
PriceIndices ....................................................... 182
prices ............................................................... 188
price_indices ..................................................... 189
products ........................................................... 190
products_fig ....................................................... 191
QMp ................................................................. 192
QMq ................................................................. 193
QU ................................................................. 194
quantities .......................................................... 195
sales ............................................................... 196
sales_groups ...................................................... 197
sales_groups2 ..................................................... 198
sato_vartia ......................................................... 199
save_model ......................................................... 200
shrinkflation ....................................................... 201
SPQ ................................................................. 203
stuvel ............................................................... 204
sugar ............................................................... 205
tindex ............................................................. 206
tornqvist ......................................................... 207
tpd ................................................................. 208
tpd_fbew ........................................................... 209
tpd_fbmw .......................................................... 210
tpd_splice ......................................................... 211
unit_value_index .................................................. 212
utpd ................................................................. 213
utpd_fbew ........................................................ 214
utpd_fbmw ........................................................ 215
utpd_splice ....................................................... 216
value_index ......................................................... 218
vartia .............................................................. 219
walsh .............................................................. 220
wgeks ............................................................. 221
wgeksaqi .......................................................... 222
wgeksaqi_fbew .................................................... 223
wgeksaqi_fbmw ................................................... 224
wgeksaqi_splice .................................................. 225
wgeksaqu .......................................................... 227
wgeksaqu_fbew ................................................... 228
wgeksaqu_fbmw ................................................... 229
Calculating the bilateral AG Mean price index

This function returns a value (or vector of values) of the bilateral AG Mean price index.

Usage

agmean(data, start, end, sigma = 0.7, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric), and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. ’2020-03’.

end The research period (as character) limited to the year and month, e.g. ’2020-04’.

sigma The elasticity of substitution parameter (as numeric)

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).
Value

The function returns a value (or vector of values) of the bilateral AG Mean price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

agmean(sugar, start="2019-01", end="2020-01", sigma=0.5)
agmean(milk, start="2018-12", end="2020-01", interval=TRUE)

available

Providing values from the indicated column that occur at least once in one of the compared periods or in a given time interval

Description

The function returns all values from the indicated column (defined by the type parameter) which occur at least once in one of the compared periods or in a given time interval.

Usage

available(data, period1, period2, type = "prodID", interval = FALSE)

Arguments

data The user’s data frame. It must contain a column time (as Date in format: year-month-day,e.g. '2020-12-01') and also a column indicated by the type parameter.
period1 The first period (as character) limited to the year and month, e.g. "2019-03".
period2 The second period (as character) limited to the year and month, e.g. "2019-04".
type This parameters defines the column which is used in the procedure. Possible values of the type parameter are: retID, prodID, codeIN, codeOUT or description.
interval A logical parameter indicating whether the procedure is to work for the whole time period between period1 and period2 (then it is TRUE).
The function returns all values from the indicated column (defined by the type parameter) which occur at least once in one of the compared periods or in a given time interval. Possible values of the type parameter are: retID, prodID, codeIN, codeOUT or description. If the interval parameter is set to FALSE, then the function compares only periods defined by period1 and period2. Otherwise the whole time period between period1 and period2 is considered.

Examples

available(milk, period1="2018-12", period2="2019-12", interval=TRUE)
available(milk, period1="2018-12", period2="2019-12", type="description")

Description

This function returns a value (or vector of values) of the bilateral Banajree price index.

Usage

banajree(data, start, end, interval = FALSE)

Arguments

data: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start: The base period (as character) limited to the year and month, e.g. "2020-03".

end: The research period (as character) limited to the year and month, e.g. "2020-04".

interval: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Banajree price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

```r
banajree(sugar, start="2018-12", end="2019-12")
banajree(milk, start="2018-12", end="2020-01", interval=TRUE)
```

---

**bennet**

*Calculating the Bennet price and quantity indicators*

**Description**

This function returns the Bennet price and quantity indicators and optionally also the price and quantity contributions of individual products.

**Usage**

```r
bennet(
  data,  # The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.
  start,  # The base period (as character) limited to the year and month, e.g. "2020-03".
  end,  # The research period (as character) limited to the year and month, e.g. "2020-04".
  interval = FALSE,  # A logical parameter indicating whether calculations are to be made for the whole time interval (TRUE) or no (FALSE).
  matched = FALSE,  # A logical parameter indicating whether the matched sample approach is to be used (if yes, the parameter has the value TRUE).
  contributions = FALSE,  # A logical parameter indicating whether the price and quantity contributions of individual products are to be returned (if yes, the parameter has the value TRUE).
  prec = 2  # The number of decimal places to which the results are rounded.
)
```

**Arguments**

- `data` The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric) and `prodID` (as numeric, factor or character). A column `quantities` (as positive numeric) is also needed because this function uses unit values as monthly prices.
- `start` The base period (as character) limited to the year and month, e.g. "2020-03".
- `end` The research period (as character) limited to the year and month, e.g. "2020-04".
- `interval` A logical parameter indicating whether calculations are to be made for the whole time interval (TRUE) or no (FALSE).
- `matched` A logical parameter indicating whether the matched sample approach is to be used (if yes, the parameter has the value TRUE).
- `contributions` A logical parameter indicating whether the price and quantity contributions of individual products are to be returned (if yes, the parameter has the value TRUE).
- `prec` The number of decimal places to which the results are rounded.
contributions  A logical parameter indicating whether contributions of individual products are to be displayed. If it is TRUE, then contributions are calculated for the the base period start and the current period end.

prec  A numeric vector indicating precision, i.e. the number of decimal places for presenting results.

Value

This function returns the Bennet price and quantity indicators and optionally also the price and quantity contributions of individual products.

References


Examples

```r
bennet(milk, "2018-12", "2019-12", matched=TRUE, contributions=TRUE)
bennet(coffee, start="2018-12", end="2019-03", interval=TRUE)
```

### bialek

**Calculating the bilateral Bialek price index**

Description

This function returns a value (or vector of values) of the bilateral Bialek price index.

Usage

```r
bialek(data, start, end, interval = FALSE)
```

Arguments

- **data**  The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- **start**  The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**  The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).
Value
The function returns a value (or vector of values) of the bilateral Bialek price index depending
on the interval parameter. If the interval parameter is set to TRUE, the function returns a
vector of price index values without dates. To get information about both price index values and
 corresponding dates, please see functions: price_indices or final_index. The function does
not take into account aggregating over outlets or product subgroups (to consider these types of
aggregating, please use the final_index function).

References
(Ekonometria). 1(35), 76-83.
Bialek, J. (2013). *Some Remarks on the Original Price Index Inspired by the Notes of Peter von der
Lippe*. Econometrics (Ekonometria), 3(41), 40-54.

Examples
biallek(sugar, start="2018-12", end="2019-12")
biallek(milk, start="2018-12", end="2020-01", interval=TRUE)

bmw

Calculating the unweighted BMW price index

Description
This function returns a value (or vector of values) of the unweighted Balk-Mehrhoff-Walsh (BMW)
price index.

Usage
bmw(data, start, end, interval = FALSE)

Arguments
data User's data frame with information about sold products. It must contain columns:
time (as Date in format: year-month-day, e.g. "2020-12-01"), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities
(as positive numeric) is also needed because this function uses unit values as monthly prices.
start The base period (as character) limited to the year and month, e.g. "2020-03".
end The research period (as character) limited to the year and month, e.g. "2020-04".
interval A logical value indicating whether the function is to compare the research period
defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all
months from the time interval <start,end> are considered and start defines
the base period (interval is set to TRUE).
Value

The function returns a value (or vector of values) of the unweighted bilateral BMW price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
bmw(sugar, start="2018-12", end="2019-12")
bmw(milk, start="2018-12", end="2020-01", interval=TRUE)
```

---

carli

*Calculating the unweighted Carli price index*

Description

This function returns a value (or vector of values) of the unweighted bilateral Carli price index.

Usage

`carli(data, start, end, interval = FALSE)`

Arguments

data: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start: The base period (as character) limited to the year and month, e.g. "2020-04".

end: The research period (as character) limited to the year and month, e.g. "2020-04".

interval: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).
Value

The function returns a value (or vector of values) of the unweighted bilateral Carli price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

carli(sugar, start="2018-12", end="2019-12")
carli(milk, start="2018-12", end="2020-01", interval=TRUE)

ccdi(data, start, end, wstart = start, window = 13)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. ”2020-03”.

end The research period (as character) limited to the year and month, e.g. ”2020-04”.

wstart The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. ”2020-01”.

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
Value

This function returns a value of the multilateral CCDI price index (to be more precise: the GEKS index based on the Tornqvist formula) which considers the time window defined by \texttt{wstart} and \texttt{window} parameters. It measures the price dynamics by comparing period \texttt{end} to period \texttt{start} (both \texttt{start} and \texttt{end} must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References


Examples

ccdi(milk, start="2019-01", end="2019-08", window=10)
ccdi(milk, start="2018-12", end="2019-12")

\texttt{ccdi\_fbew}

\textit{Extending the multilateral CCDI price index by using the FBEW method.}

Description

This function returns a value of the multilateral CCDI price index (GEKS based on the Tornqvist formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

\texttt{ccdi\_fbew(data, start, end)}

Arguments

data The user’s data frame with information about sold products. It must contain columns: \texttt{time} (as Date in format: year-month-day,e.g. '2020-12-01'), \texttt{prices} (as positive numeric), \texttt{quantities} (as positive numeric) and \texttt{prodID} (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2019-12".
end The research period (as character) limited to the year and month, e.g. "2020-04".
Value

This function returns a value of the multilateral CCDI price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

ccdi_fbew(milk, start="2018-12", end="2019-08")

---

**Description**

This function returns a value of the multilateral CCDI price index (GEKS based on the Tornqvist formula) extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

ccdi_fbmw(data, start, end)

**Arguments**

data: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start: The base period (as character) limited to the year and month, e.g. "2019-12".

end: The research period (as character) limited to the year and month, e.g. "2020-04".
Value

This function returns a value of the multilateral CCDI price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)-1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product sub-groups (to consider these types of aggregating, please use the final_index function).

References


Examples

ccdi_fbmw(milk, start="2019-12", end="2020-04")

---

ccdi_splice

Extending the multilateral CCDI price index by using window splicing methods.

Description

This function returns a value (or values) of the multilateral CCDI price index (GEKS based on the Tornqvist formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

ccdi_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)
**ccdi_splice**

**Arguments**

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

- **window**: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

- **splice**: A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".

- **interval**: A logical value indicating whether the function is to provide the price index comparing the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by `start`).

**Value**

This function returns a value or values (depending on `interval` parameter) of the multilateral CCDI price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in `start` and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**


**Examples**

```r
ccdi_splice(milk, start="2018-12", end="2020-02", splice="half")
```
Calculating the monthly chained AG Mean price index

Description

This function returns a value (or vector of values) of the monthly chained AG Mean price index.

Usage

chagmean(data, start, end, sigma = 0.7, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

sigma The elasticity of substitution parameter (as numeric).

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained AG Mean price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chagmean(sugar, start="2019-01", end="2019-04", sigma=0.5)
chagmean(milk, start="2018-12", end="2020-01", interval=TRUE)
**chbanajree**  
*Calculating the monthly chained Banajree price index*

**Description**

This function returns a value (or vector of values) of the monthly chained Banajree price index.

**Usage**

`chbanajree(data, start, end, interval = FALSE)`

**Arguments**

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start,end>` are considered and `start` defines the base period (interval is set to TRUE).

**Value**

The function returns a value (or vector of values) of the monthly chained Banajree price index depending on the `interval` parameter. If the `interval` parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**


**Examples**

```r
cbanajree(sugar, start="2018-12", end="2019-04")
cbanajree(milk, start="2018-12", end="2020-01", interval=TRUE)
```
Calculating the monthly chained Bialek price index

Description

This function returns a value (or vector of values) of the monthly chained Bialek price index.

Usage

chbialek(data, start, end, interval = FALSE)

Arguments

data: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start: The base period (as character) limited to the year and month, e.g. "2020-03".

date: The research period (as character) limited to the year and month, e.g. "2020-04".

interval: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start, end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Bialek price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chbialek(sugar, start="2018-12", end="2019-04")
chbialek(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the monthly chained BMW price index

Description

This function returns a value (or vector of values) of the monthly chained Balk-Mehrhoff-Walsh (BMW) price index.

Usage

chbmw(data, start, end, interval = FALSE)

Arguments

data: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start: The base period (as character) limited to the year and month, e.g. "2020-03".

end: The research period (as character) limited to the year and month, e.g. "2020-04".

interval: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained BMW price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chbmw(sugar, start="2018-12", end="2019-04")
chbmw(milk, start="2018-12", end="2020-01", interval=TRUE)
chcarli

Calculating the monthly chained Carli price index

Description
This function returns a value (or vector of values) of the monthly chained Carli price index.

Usage
chcarli(data, start, end, interval = FALSE)

Arguments
- **data**: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value
The function returns a value (or vector of values) of the monthly chained Carli price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References

Examples
chcarli(sugar, start="2018-12", end="2019-04")
chcarli(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the monthly chained CSWD price index

Description

This function returns a value (or vector of values) of the monthly chained Carruthers-Sellwood-Ward-Dalen (CSWD) price index.

Usage

chcswd(data, start, end, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. "2020-12-01"), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained CSWD price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

cchcswd(sugar, start="2018-12", end="2019-04")

cchcswd(milk, start="2018-12", end="2020-01", interval=TRUE)

Calculating the monthly chained Davies price index

Description

This function returns a value (or vector of values) of the monthly chained Davies price index.

Usage

chdavies(data, start, end, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Davies price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Calculating the monthly chained Dikhanov price index

Description

This function returns a value (or vector of values) of the monthly chained Dikhanov price index.

Usage

```
chdikhanov(data, start, end, interval = FALSE)
```

Arguments

- **data**: The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric) and `prodID` (as numeric, factor or character). A column `quantities` (as positive numeric) is also needed because this function uses unit values as monthly prices.
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start,end>` are considered and `start` defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Dikhanov price index depending on the `interval` parameter. If the `interval` parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```
chdikhanov(sugar, start="2018-12", end="2019-04")
chdikhanov(milk, start="2018-12", end="2020-01", interval=TRUE)
```
Calculating the monthly chained Drobisch price index

Description

This function returns a value (or vector of values) of the monthly chained Drobisch price index.

Usage

chdrobisch(data, start, end, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

date The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start, end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Drobisch price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chdrobisch(sugar, start="2018-12", end="2019-04")
chdrobisch(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the monthly chained Dutot price index

Description

This function returns a value (or vector of values) of the monthly chained Dutot price index.

Usage

chdutot(data, start, end, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Dutot price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chdutot(sugar, start="2018-12", end="2019-04")
chdutot(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the monthly chained Fisher price index

Description

This function returns a value (or vector of values) of the monthly chained Fisher price index.

Usage

chfisher(data, start, end, interval = FALSE)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Fisher price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chfisher(sugar, start="2018-12", end="2019-04")
chfisher(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the monthly chained Geary-Khamis price index

Description

This function returns a value (or vector of values) of the monthly chained Geary-Khamis price index.

Usage

chgeary_khamis(data, start, end, interval = FALSE)

Arguments

data

The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start

The base period (as character) limited to the year and month, e.g. "2020-03".

end

The research period (as character) limited to the year and month, e.g. "2020-04".

interval

A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Geary-Khamis price index depending on the interval parameter (please use gk function to calculate the multilateral Geary-Khamis price index). If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (see the final_index function).

References


Examples

chgeary_khamis(sugar, start="2018-12", end="2019-04")
chgeary_khamis(milk, start="2018-12", end="2020-01", interval=TRUE)

chgeohybrid  Calculating the the monthly chained geohybrid price index

Description

This function returns a value (or vector of values) of the monthly chained geohybrid price index. The geohybrid index was proposed by Bialek (2020) and it uses correlation coefficients between prices and quantities.

Usage

chgeohybrid(data, start, end, base = start, interval = FALSE)

Arguments

data  The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  The base period (as character) limited to the year and month, e.g. "2020-03".

end  The research period (as character) limited to the year and month, e.g. "2020-04".

base  The prior period used in the geohybrid price index formula (as character) limited to the year and month, e.g. "2020-01".

interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained geohybrid price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References

Examples

cchgeohybrid(sugar, start="2019-12", end="2020-05", base="2018-12")
cchgeohybrid(milk, start="2019-12", end="2020-08", base="2018-12", interval=TRUE)

chgeolaspeyres

Calculating the monthly chained geo-logarithmic Laspeyres price index

Description

This function returns a value (or vector of values) of the monthly chained geo-logarithmic Laspeyres price index.

Usage

chgeolaspeyres(data, start, end, interval = FALSE)

Arguments

data

The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start

The base period (as character) limited to the year and month, e.g. "2020-03".

end

The research period (as character) limited to the year and month, e.g. "2020-04".

interval

A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained geo-logarithmic Laspeyres price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chgeolaspeyres(sugar, start="2018-12", end="2019-04")
chgeolaspeyres(milk, start="2018-12", end="2020-01", interval=TRUE)

Calculating the monthly chained geometric Lowe price index

Description
This function returns a value (or vector of values) of the monthly chained geometric Lowe price index.

Usage

chgeolowe(data, start, end, base = start, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2020-03".
end The research period (as character) limited to the year and month, e.g. "2020-04".
base The prior period used in the geometric Lowe price index formula (as character) limited to the year and month, e.g. "2020-01".
interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value
The function returns a value (or vector of values) of the monthly chained geometric Lowe price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References
chgeopaasche

Examples

chgeolowe(sugar, start="2019-01", end="2019-04", base="2018-12")
chgeolowe(milk, start="2018-12", end="2020-01", interval=TRUE)

chgeopaasche Calculating the monthly chained geo-logarithmic Paasche price index

Description

This function returns a value (or vector of values) of the monthly chained geo-logarithmic Paasche price index.

Usage

chgeopaasche(data, start, end, interval = FALSE)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained geo-logarithmic Paasche price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chgeopaasche(sugar, start="2018-12", end="2019-04")
chgeopaasche(milk, start="2018-12", end="2020-01", interval=TRUE)

Description

This function returns a value (or vector of values) of the monthly chained geometric Young price index.

Usage

chgeoyoung(data, start, end, base = start, interval = FALSE)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. "2020-12-01"), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

date The research period (as character) limited to the year and month, e.g. "2020-04".

base The prior period used in the geometric Young price index formula (as character) limited to the year and month, e.g. "2020-01".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained geometric Young price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


chharmonic

Examples
chgeoyoung(sugar, start="2019-01", end="2019-04", base="2018-12")
chgeoyoung(milk, start="2018-12", end="2020-01", interval=TRUE)

chharmonic  Calculating the monthly chained harmonic price index

Description
This function returns a value (or vector of values) of the monthly chained "unnamed" harmonic price index.

Usage
chharmonic(data, start, end, interval = FALSE)

Arguments
data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value
The function returns a value (or vector of values) of the monthly chained harmonic price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References
Examples

chharmonic(sugar, start="2018-12", end="2019-04")
chharmonic(milk, start="2018-12", end="2020-01", interval=TRUE)

Description

This function returns a value (or vector of values) of the monthly chained hybrid price index. The hybrid index was proposed by Bialek (2020) and it uses correlation coefficients between prices and quantities.

Usage

chhybrid(data, start, end, base = start, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day,e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-01".

end The research period (as character) limited to the year and month, e.g. "2020-04".

base The prior period used in the hybrid price index formula (as character) limited to the year and month, e.g. "2020-01".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start, end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained hybrid price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References

Examples

chIQMp(sugar, start="2019-01", end="2020-01")
chIQMp(sugar, start="2019-01", end="2020-01", r=1.3, interval=TRUE)

---

chIQMp

Calculating the monthly chained implicit quadratic mean of order r price index

Description

This function returns a value (or vector of values) of the monthly chained implicit quadratic mean of order r price index.

Usage

chIQMp(data, start, end, r = 2, interval = FALSE)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2020-03".
end The research period (as character) limited to the year and month, e.g. "2020-04".
r The real and non-zero parameter.
interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained implicit quadratic mean of order r price index - see CPI Manual (2004), Section 17.37, formula 17.32 (page 321).

References


Examples

chIQMp(sugar, start="2019-01", end="2020-01")
chIQMp(sugar, start="2019-01", end="2020-01", r=1.3, interval=TRUE)
Calculating the monthly chained Jevons price index

Description

This function returns a value (or vector of values) of the monthly chained Jevons price index

Usage

chjevons(data, start, end, interval = FALSE)

Arguments

data: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start: The base period (as character) limited to the year and month, e.g. "2020-03".

end: The research period (as character) limited to the year and month, e.g. "2020-04".

interval: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Jevons price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chjevons(sugar, start="2018-12", end="2019-04")
chjevons(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the monthly chained Laspeyres price index

Description

This function returns a value (or vector of values) of the monthly chained Laspeyres price index.

Usage

chlaspeyres(data, start, end, interval = FALSE)

Arguments

data
The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start
The base period (as character) limited to the year and month, e.g. "2020-03".

end
The research period (as character) limited to the year and month, e.g. "2020-04".

interval
A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Laspeyres price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chlaspeyres(sugar, start="2018-12", end="2019-04")
chlaspeyres(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the monthly chained Lehr price index

Description

This function returns a value (or vector of values) of the monthly chained Lehr price index.

Usage

\texttt{chlehr(data, start, end, interval = FALSE)}

Arguments

data \hspace{1cm} The user's data frame with information about sold products. It must contain columns: \texttt{time} (as Date in format: year-month-day, e.g. '2020-12-01'), \texttt{prices} (as positive numeric), \texttt{quantities} (as positive numeric) and \texttt{prodID} (as numeric, factor or character).

start \hspace{1cm} The base period (as character) limited to the year and month, e.g. "2020-03".

end \hspace{1cm} The research period (as character) limited to the year and month, e.g. "2020-04".

interval \hspace{1cm} A logical value indicating whether the function is to compare the research period defined by \texttt{end} to the base period defined by \texttt{start} (then \texttt{interval} is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval \langle \texttt{start}, \texttt{end} \rangle are considered and \texttt{start} defines the base period (\texttt{interval} is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Lehr price index depending on the \texttt{interval} parameter. If the \texttt{interval} parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References

Lehr, J. (1885). \textit{Beitrage zur Statistik der Preise, insbesondere des Geldes und des Holzes}. J. D. Sauerlander, Frankfurt am Main.


Examples

\texttt{chlehr(sugar, start="2018-12", end="2019-04")}
\texttt{chlehr(milk, start="2018-12", end="2020-01", TRUE)}
Calculating the monthly chained Lloyd-Moulton price index

Description

This function returns a value (or vector of values) of the monthly chained Lloyd-Moulton price index.

Usage

chlloyd_moulton(data, start, end, sigma = 0.7, interval = FALSE)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

sigma The elasticity of substitution parameter (as numeric).

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start, end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Lloyd-Moulton price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

chlloyd_moulton(sugar, start="2018-12", end="2019-04", sigma=0.9)
chlloyd_moulton(milk, start="2018-12", end="2020-01", interval=TRUE)

Calculating the monthly chained Lowe price index

Description
This function returns a value (or vector of values) of the monthly chained Lowe price index.

Usage
chlowe(data, start, end, base = start, interval = FALSE)

Arguments
- data: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- start: The base period (as character) limited to the year and month, e.g. "2020-03".
- end: The research period (as character) limited to the year and month, e.g. "2020-04".
- base: The prior period used in the Lowe price index formula (as character) limited to the year and month, e.g. "2020-01".
- interval: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value
The function returns a value (or vector of values) of the monthly chained Lowe price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References
Examples

chlowe(sugar, start=“2019-01”, end=“2019-04”, base=“2018-12”)
chlowe(milk, start=“2018-12”, end=“2020-01”, interval=TRUE)

chmarshall_edgeworth  Calculating the monthly chained Marshall-Edgeworth price index

Description

This function returns a value (or vector of values) of the monthly chained Marshall-Edgeworth price index.

Usage

chmarshall_edgeworth(data, start, end, interval = FALSE)

Arguments

data  The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  The base period (as character) limited to the year and month, e.g. “2020-03”.

end  The research period (as character) limited to the year and month, e.g. “2020-04”.

interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start, end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Marshall-Edgeworth price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


chpaasche


Examples

chmarshall_edgeworth(sugar, start="2018-12", end="2019-04")
chmarshall_edgeworth(milk, start="2018-12", end="2020-01", interval=TRUE)

chpaasche Calculating the monthly chained Paasche price index

Description

This function returns a value (or vector of values) of the monthly chained Paasche price index.

Usage

chpaasche(data, start, end, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Paasche price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

```r
chpaasche(sugar, start="2018-12", end="2019-04")
chpaasche(milk, start="2018-12", end="2020-01", interval=TRUE)
```

---

Calculating the monthly chained Palgrave price index

Description

This function returns a value (or vector of values) of the monthly chained Palgrave price index.

Usage

```r
chpalgrave(data, start, end, interval = FALSE)
```

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Palgrave price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

chpalgrave(sugar, start="2018-12", end="2019-04")
chpalgrave(milk, start="2018-12", end="2020-01", interval=TRUE)

---

### chQMp

**Calculating the monthly chained quadratic mean of order r price index**

**Description**

This function returns a value (or vector of values) of the monthly chained quadratic mean of order r price index.

**Usage**

chQMp(data, start, end, r = 2, interval = FALSE)

**Arguments**

data: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start: The base period (as character) limited to the year and month, e.g. "2020-03".

end: The research period (as character) limited to the year and month, e.g. "2020-04".

r: The real and non-zero parameter.

interval: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

**Value**

The function returns a value (or vector of values) of the monthly chained quadratic mean of order r price index - see CPI Manual (2004), Section 17.40, formula 17.35 (page 321).
chQMq

References


Examples

chQMp(sugar, start="2019-01", end="2020-01")
chQMp(sugar, start="2019-01", end="2020-01", r=1.3, interval=TRUE)

chQMq | Calculating the monthly chained quadratic mean of order r quantity index

Description

This function returns a value (or vector of values) of the monthly chained quadratic mean of order r quantity index.

Usage

chQMq(data, start, end, r = 2, interval = FALSE)

Arguments

data | The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start | The base period (as character) limited to the year and month, e.g. "2020-03".
end | The research period (as character) limited to the year and month, e.g. "2020-04".
r | The real and non-zero parameter.
interval | A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained quadratic mean of order r quantity index - see CPI Manual (2004), Section 17.35, formula 17.30 (page 321).

References

Examples

```r
chQMq(sugar, start="2019-01", end="2020-01")
chQMq(sugar, start="2019-01", end="2020-01", r=1.3, interval=TRUE)
```

### chsato_vartia

**Calculating the monthly chained Vartia-II (Sato-Vartia) price index**

#### Description

This function returns a value (or vector of values) of the monthly chained Vartia-II (Sato-Vartia) price index.

#### Usage

```r
chsato_vartia(data, start, end, interval = FALSE)
```

#### Arguments

- `data` The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- `start` The base period (as character) limited to the year and month, e.g. "2020-03".
- `end` The research period (as character) limited to the year and month, e.g. "2020-04".
- `interval` A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to `FALSE`) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start,end>` are considered and `start` defines the base period (`interval` is set to `TRUE`).

#### Value

The function returns a value (or vector of values) of the monthly chained Vartia-II (Sato-Vartia) price index depending on the `interval` parameter. If the `interval` parameter is set to `TRUE`, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

#### References


Examples

```r
chsato_vartia(sugar, start="2018-12", end="2019-04")
chsato_vartia(milk, start="2018-12", end="2020-01", interval=TRUE)
```

Description

This function returns a value (or vector of values) of the monthly chained Stuvel price index.

Usage

```r
chstuvel(data, start, end, interval = FALSE)
```

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. "2020-12-01"), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In the latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Stuvel price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

chtuvel(sugar, start="2018-12", end="2019-04")
chtuvel(milk, start="2018-12", end="2020-01", interval=TRUE)

chtornqvist

*Calculating the monthly chained Tornqvist price index*

Description

This function returns a value (or vector of values) of the monthly chained Tornqvist price index.

Usage

chtornqvist(data, start, end, interval = FALSE)

Arguments

data  The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day,e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  The base period (as character) limited to the year and month, e.g. "2020-03".

end  The research period (as character) limited to the year and month, e.g. "2020-04".

interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Tornqvist price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples
```
chtornqvist(sugar, start="2018-12", end="2019-04")
chtornqvist(milk, start="2018-12", end="2020-01", interval=TRUE)
```

**Description**
This function returns a value (or vector of values) of the monthly chained Vartia-I price index.

**Usage**
```
chvartia(data, start, end, interval = FALSE)
```

**Arguments**
- **data**: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

**Value**
The function returns a value (or vector of values) of the monthly chained Vartia-I price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

chvartia(sugar, start="2018-12", end="2019-04")
chvartia(milk, start="2018-12", end="2020-01", interval=TRUE)

chwalsh  
*Calculating the monthly chained Walsh price index*

Description

This function returns a value (or vector of values) of the monthly chained Walsh price index.

Usage

chwalsh(data, start, end, interval = FALSE)

Arguments

data  
The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  
The base period (as character) limited to the year and month, e.g. "2020-03".

end  
The research period (as character) limited to the year and month, e.g. "2020-04".

interval  
A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the monthly chained Walsh price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: *price_indices* or *final_index*. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the *final_index* function).
References


Examples

chwalsh(sugar, start="2018-12", end="2019-04")
chwalsh(milk, start="2018-12", end="2020-01", interval=TRUE)

### Description

This function returns a value (or vector of values) of the monthly chained Young price index.

### Usage

chyoung(data, start, end, base = start, interval = FALSE)

### Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. "2020-12-01"), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **base**: The prior period used in the Young price index formula (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start,end>` are considered and `start` defines the base period (interval is set to TRUE).

### Value

The function returns a value (or vector of values) of the monthly chained Young price index depending on the `interval` parameter. If the `interval` parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
chyoung(sugar, start="2019-01", end="2019-04", base="2018-12")
chyoung(milk, start="2018-12", end="2020-01", interval=TRUE)
```

<table>
<thead>
<tr>
<th>coffee</th>
<th>A real data set on sold coffee</th>
</tr>
</thead>
</table>

Description

A collection of scanner data on the sale of coffee in one of Polish supermarkets in the period from December 2017 to October 2020

Usage

coffee

Format

A data frame with 6 columns and 42561 rows. The used variables are as follows:

- **time**: Dates of transactions (Year-Month-Day)
- **prices**: Prices of sold products [PLN]
- **quantities**: Quantities of sold products [kg]
- **prodID**: Unique product codes (data set contains 79 different prodIDs)
- **retID**: Unique codes identifying outlets/retailer sale points (data set contains 20 different retIDs)
- **description**: Descriptions of sold coffee products (data set contains 3 different product descriptions)
compare_distances  Calculating distances between price indices

Description
The function calculates distances between price indices.

Usage
compare_distances(
  data = data.frame(),
  measure = "MAD",
  pp = TRUE,
  first = FALSE,
  prec = 3
)

Arguments
- **data**: A data frame containing values of indices which are to be compared.
- **measure**: A parameter specifying what measure should be used to compare the indexes. Possible parameter values are: "MAD" (Mean Absolute Distance) or "RMSD" (Root Mean Square Distance).
- **pp**: Logical parameter indicating whether the results are to be presented in percentage points (then pp = TRUE).
- **first**: A logical parameter that determines whether the first row of the data frame is to be taken into account when calculating the distance between the indices (then first = TRUE). Usually, the first row concerns the index values for the base period - all indexes are then set to one.
- **prec**: Parameter that determines how many decimal places are to be used in the presentation of results.

Value
The function calculates average distances between price indices and it returns a data frame with these values for each pair of price indices.

Examples
# Creating a data frame with unweighted bilateral index values
df <- price_indices(milk, formula=c("jevons","dutot","carli"), start="2018-12", end="2019-12", interval=TRUE)
# Calculating average distances between indices (in p.p)
compare_distances(df)
compare_indices_df  A function for graphical comparison of price indices

Description

This function returns a figure with plots of selected price indices.

Usage

```r
compare_indices_df(
  data,
  names = colnames(data)[2:length(colnames(data))],
  date_breaks = "1 month"
)
```

Arguments

- **data**: The user’s data frame with price index values. It must contain columns: `time` (as character in format: year-month, e.g. '2020-12') and columns with index values.
- **names**: A vector of strings indicating names of indices which are to be used in the figure’s legend.
- **date_breaks**: A string giving the distance between breaks on the X axis like "1 month" (default value) or "4 months".

Value

This function returns a figure with plots of previously calculated indices (together with dates on X-axis and a corresponding legend). Indices must be provided as a data frame, where the the first column must includes dates limited to the year and month (e.g.: "2020-04").

Examples

```r
df<-price_indices(milk, start = "2018-12", end = "2019-12",
  formula=c("laspeyres", "fisher"), interval = TRUE)
compare_indices_df(df)
```

compare_indices_jk  A general function to compare indices by using the jackknife method

Description

This function presents a comparison of selected indices obtained by using the jackknife method.
compare_indices_jk

Usage

```r
compare_indices_jk(
data,
start,
end,
by = "prodID", 
formula = c(),
window = c(),
splice = c(),
base = c(),
sigma = c(),
r = c(),
names = c(),
title_iterations = c(),
title_pseudovalues = c()
)
```

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also essential even if the selected index is an unweighted formula (unit values are calculated).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

by A character string which indicates a column name for creating product subgroups (in the classical jackknife method by should indicate prodID). In each, successive repetition, the indicated price indexes are counted on the set of products reduced by the subset determined by the successive element of the column indicated by the by parameter.

formula A vector of character strings indicating price index formulas that are to be calculated. To see available options please use the link: PriceIndices.

window A vector of integers. Each element of the vector defines the length of the time window of the corresponding multilateral index.

splice A vector of character strings. Each element of the vector indicates the splicing method is to be used for the corresponding multilateral index. Available values of vector elements are: "movement", "window", "half", "mean" and their additional variants: "window_published", "half_published" and "mean_published".

base The vector of prior periods used in the Young- or Lowe-type price indices or hybrid/geohybrid index. Each element of the vector (as character) must be limited to the year and month, e.g. "2020-01".

sigma The vector of elasticity of substitution parameters used in the Lloyed-Moulton, AG Mean or GEKS-LM indices (as numeric).

r The vector of non-zero parameters used in the quadratic mean of order r quantity / price index or in the GEKS-QM index (as numeric).
compare_indices_jk

names  
A vector of strings indicating names of indices which are to be used in the resulting data frame.

title_iterations
A character string indicating a title of the created box-plot for iteration index values.

title_pseudovalues
A character string indicating a title of the created box-plot for obtained (jackknife) index pseudovalues.

Value

This function presents a comparison of selected indices obtained by using the jackknife method. In particular, it returns a list with four elements: iterations, which is a data frame with basic characteristics of the calculated iteration index values (means, standard deviations, coefficients of variation and results for all sample), pseudovalues, which is a data frame with basic characteristics of the calculated index pseudovalues obtained in the jackknife procedure (i.e. the jackknife estimators and their standard deviations and coefficients of variation), figure_iterations which presents a box-plot for the calculated iteration index values, and figure_pseudovalues which presents a box-plot for the calculated index pseudovalues obtained in the jackknife procedure.

References


Examples

milk.<-dplyr::filter(milk, milk$prodID %in% sample(unique(milk$prodID),4))
#creating a list with jackknife results
comparison<-compare_indices_jk(milk.,
  formula=c("jevons","fisher"),
  start="2018-12",
  end="2019-12",
  names=c("Jevons","Fisher"),
  title_iterations="Box-plots for iteration values (milk products)",
  title_pseudovalues="Box-plots for pseudovalues (milk products)"
)
#displaying results
comparison$iterations
comparison$pseudovalues
comparison$figure_iterations
comparison$figure_pseudovalues
**Description**

This function returns a figure with plots of previously calculated price indices.

**Usage**

```r
compare_indices_list(data = list(), names = c(), date_breaks = "1 month")
```

**Arguments**

- `data` A list of data frames with previously calculated price indices. Each data frame must consist of two columns, i.e. the first column must include dates limited to the year and month (e.g., "2020-04") and the second column must indicate price index values for corresponding dates. The above-mentioned single data frame may be created manually in the previous step or it may be a result of functions: `price_index` or `final_index`. All considered data frames must have an identical number of rows.

- `names` A vector of character strings describing names of presented indices.

- `date_breaks` A string giving the distance between breaks on the X axis like "1 month" (default value) or "4 months".

**Value**

This function returns a figure with plots of previously calculated price indices. It allows for graphical comparison of price index values which were previously calculated and now are provided as a list of data frames (see `data` parameter).

**Examples**

```r
## Calculating two indices by using two different package functions:
index1<-final_index(data=milk, start="2018-12", end="2019-12", formula="walsh", interval=TRUE)
index2<-price_indices(milk, start="2018-12", end="2019-12", formula="geks", window=13, interval=TRUE)
## Graphical comparison of these two indices
compare_indices_list(data=list(index1,index2), names=c("Walsh index", "GEKS index"))
```
compare_to_target  
Calculating distances between considered price indices and the target price index

Description

The function calculates distances between considered price indices and the target price index

Usage

```r
compare_to_target(
  data = data.frame(),
  target,
  measure = "MAD",
  pp = TRUE,
  first = FALSE,
  prec = 3
)
```

Arguments

data  A data frame containing values of indices which are to be compared to the target price index
target  A data frame or a vector containing values of the target price index
measure  A parameter specifying what measure should be used to compare indices. Possible parameter values are: "MAD" (Mean Absolute Distance) or "RMSD" (Root Mean Square Distance).
pp  Logical parameter indicating whether the results are to be presented in percentage points (then pp = TRUE).
first  A logical parameter that determines whether the first row of the data frame and the first row of the 'target' data frame (or its first element if it is a vector) are to be taken into account when calculating the distance between the indices (then first = TRUE). Usually, the first row concerns the index values for the base period - all indexes are then set to one.
prec  Parameter that determines how many decimal places are to be used in the presentation of results.

Value

The function calculates average distances between considered price indices and the target price index and it returns a data frame with: average distances on the basis of all values of compared indices ('distance' column), average semi-distances on the basis of values of compared indices which overestimate the target index values ('distance_upper' column) and average semi-distances on the basis of values of compared indices which underestimate the target index values ('distance_lower' column).
Examples

#Creating a data frame with example bilateral indices
df<-price_indices(milk,
formula=c("jevons","laspeyres","paasche","walsh"),
start="2018-12",end="2019-12",interval=TRUE)
#Calculating the target Fisher price index
target_index<-fisher(milk,start="2018-12",end="2019-12",interval=TRUE)
#Calculating average distances between considered indices and the Fisher index (in p.p)
calculate_average_distances(df,target=target_index)

cswd

Calculating the unweighted CSWD price index

Description

This function returns a value (or vector of values) of the unweighted Carruthers-Sellwood-Ward-Dalen (CSWD) price index.

Usage

cswd(data, start, end, interval = FALSE)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the unweighted bilateral CSWD price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
dataAGGR

A small artificial scanner data set for a demonstration of data aggregation

Description

A collection of artificial scanner data on milk products sold in three different months

Usage

dataAGGR

Format

A data frame with 6 columns and 9 rows. The used variables are as follows:

- **time**: Dates of transactions (Year-Month-Day: 4 different dates)
- **prices**: Prices of sold products [PLN]
- **quantities**: Quantities of sold products [l]
- **prodID**: Retailer product codes (3 prodIDs)
- **retID**: Unique codes identifying outlets/retailer sale points (4 retIDs)
- **description**: Descriptions of sold products (two subgroups: goat milk, powdered milk)

References


Examples

cswd(sugar, start="2018-12", end="2019-12")
cswd(milk, start="2018-12", end="2020-01", interval=TRUE)
**Description**


**Usage**

dataCOICOP

**Format**

A data frame with 10 columns and 139600 rows. The used variables are as follows:

- **time** - Dates of transactions (Year-Month-Day)
- **prices** - Prices of sold products [PLN]
- **quantities** - Quantities of sold products
- **description** - Descriptions of sold products (original: in Polish)
- **codeIN** - Retailer product codes
- **retID** - Unique codes identifying outlets/retailer sale points
- **grammage** - Product grammages
- **unit** - Sales units, e.g.: kg, ml, etc.
- **category** - Product categories (in English) corresponding to COICOP 6 levels
- **coicop6** - Identifiers of local COICOP 6 groups (6 groups)

**Description**

A collection of scanner data on the sale of sample artificial products.

**Usage**

dataMATCH
Format

A data frame with 7 columns and 30 rows. The used variables are as follows:

- **time**: Dates of transactions (Year-Month-Day)
- **prices**: Prices of sold products [PLN]
- **quantities**: Quantities of sold products [liters]
- **codeIN**: Unique internal (retailer) product codes (data set contains 5 different codeINs)
- **codeOUT**: Unique external product codes (data set contains 5 different codeOUTs)
- **retID**: Unique codes identifying outlets/retailer sale points (data set contains 2 different retIDs)
- **description**: Descriptions of sold products (data set contains 3 different product descriptions)

**dataU**

An artificial, small scanner data set

Description

A collection of artificial scanner data on 6 products sold in Dec, 2018. Product descriptions contain the information about their grammage and unit.

Usage

dataU

Format

A data frame with 5 columns and 6 rows. The used variables are as follows:

- **time**: Dates of transactions (Year-Month-Day)
- **prices**: Prices of sold products [PLN]
- **quantities**: Quantities of sold products [item]
- **prodID**: Unique product codes
- **description**: Descriptions of sold products (data set contains 6 different product descriptions)
data_aggregating

Aggregating the user’s data frame

description

The function aggregates the user’s data frame over time and optionally over outlets.

Usage

data_aggregating(data, join_outlets = TRUE)

Arguments

data The user’s data frame.
join_outlets A logical value indicating whether the data aggregation over outlets should be also done.

Value

The function aggregates the user’s data frame over time and/or over outlets. Consequently, we obtain monthly data, where the unit value is calculated instead of a price for each prodID observed in each month (the time column gets the Date format: "Year-Month-01"). If the parameter join_outlets is TRUE, then the function also performs aggregation over outlets (retIDs) and the retID column is removed from the data frame. The main advantage of using this function is the ability to reduce the size of the data frame and the time needed to calculate the price index. Please note, that unnecessary columns are removed (e.g. description).

Examples

#Example 1
data_aggregating(dataAGGR, join_outlets = FALSE)
data_aggregating(dataAGGR, join_outlets = TRUE)

#Example 2 (data frame reduction)
nrow(milk)
nrow(data_aggregating(milk))

data_check

Checking the user’s data frame

description

The function checks if the argument data points to a data frame which is suitable for further price index calculation. In particular, the function checks whether the indicated data frame contains the required columns and whether they are of the appropriate type (if not, the function returns FALSE and an appropriate comment).
Usage

data_check(data)

Arguments

data Any R object but ultimately it is a data frame.

Value

The function returns TRUE if the data frame indicated by the data parameter is suitable for the calculation of price indices and returns FALSE otherwise.

Examples

data_check(milk)
data_check(iris)

---

data_classifying Predicting product classes via the machine learning model

Description

This function predicts product class levels via the selected machine learning model.

Usage

data_classifying(model = list(), data)

Arguments

model A list of 8 elements which identify the previously built machine learning model (the list is obtained via the model_classification function).
data A data set for the model (products with their characteristics). This data set must contain all the columns which were used in the built model.

Value

This function provides the indicated data set with an additional column, i.e. class_predicted, which is obtained by using the selected machine learning model.
Examples

```r
# Building the model
my.grid=list(eta=c(0.01,0.02,0.05),subsample=c(0.5,0.8))
data_train<-dplyr::filter(dataCOICOP,dataCOICOP$time<=as.Date("2021-10-01"))
data_test<-dplyr::filter(dataCOICOP,dataCOICOP$time==as.Date("2021-11-01"))
ML<-model_classification(data_train,data_test,class="coicop6",grid=my.grid,
indicators=c("description","codeIN", "grammage"),key_words=c("uht"),rounds=60)
# Data classification
data_classifying(ML, data_test)
```

---

**data_DOWN_UP_SIZED**

*An artificial data set on sold coffee*

---

**Description**

A collection of scanner data on the sale of coffee in the period from January 2024 to February 2024

**Usage**

**data_DOWN_UP_SIZED**

**Format**

A data frame with 6 columns and 51 rows. The used variables are as follows:

- **time** - Dates of transactions (Year-Month-Day)
- **prices** - Prices of sold products [PLN]
- **quantities** - Quantities of sold products [liters]
- **codeIN** - Unique internal product codes (retailer product codes)
- **codeOUT** - Unique external product codes (e.g. GTIN, EAN, SKU)
- **description** - Descriptions of sold coffee products

---

**data_filtering**

*Filtering a data set for further price index calculations*

---

**Description**

This function returns a filtered data set, i.e. a reduced user’s data frame with the same columns and rows limited by a criterion defined by filters.
Usage

data_filtering(
  data,
  start,
  end,
  filters = c(),
  plimits = c(),
  pquantiles = c(),
  dplimits = c(),
  lambda = 1.25,
  interval = FALSE,
  retailers = FALSE
)

Arguments

data
  The user’s data frame with information about products to be filtered. It must contain columns: 
time (as Date in format: year-month-day, e.g. ‘2020-12-01’), prices (as positive numeric) and quantities (as positive numeric).

start
  The base period (as character) limited to the year and month, e.g. “2020-03”.

end
  The research period (as character) limited to the year and month, e.g. “2020-04”.

filters
  A vector of filter names (options are: extremeprices, dumpprices and/or lowsales).

plimits
  A two-dimensional vector of thresholds for minimum and maximum price change (it works if one of the chosen filters is extremeprices filter).

pquantiles
  A two-dimensional vector of quantile levels for minimum and maximum price change (it works if one of the chosen filters is extremeprices filter).

dplimits
  A two-dimensional vector of thresholds for maximum price drop and maximum expenditure drop (it works if one of the chosen filters is dumpprices filter).

lambda
  The lambda parameter for lowsales filter (see References below).

interval
  A logical value indicating whether the filtering process concerns only two periods defined by start and end parameters (then the interval is set to FALSE) or whether the function is to filter products sold during the whole time interval <start, end>, i.e. any subsequent months are compared.

retailers
  A logical parameter indicating whether filtering should be done for each outlet (retID) separately. If it is set to FALSE, then there is no need to consider the retID column.

Value

This function returns a filtered data set (a reduced user’s data frame). If the set of filters is empty, then the function returns the original data frame (defined by the data parameter) limited to considered months. On the other hand, if all filters are chosen, i.e. filters=c(extremeprices, dumpprices, lowsales), then these filters work independently and a summary result is returned. Please note that both variants of extremeprices filter can be chosen at the same time, i.e. plimits and pquantiles, and they work also independently.
**References**


**Examples**

```r
data_filtering(milk, start="2018-12", end="2019-03",
              filters=c("extremeprices"), pquantiles=c(0.01, 0.99), interval=TRUE)
data_filtering(milk, start="2018-12", end="2019-03",
              filters=c("extremeprices", "lowsales"), plimits=c(0.25, 2))
```

**data_imputing**

*Imputing missing and (optionally) zero prices.*

**Description**

This function imputes missing prices and (optionally) zero prices by using carry forward/backward prices.

**Usage**

```r
data_imputing(data, start, end, zero_prices = TRUE, outlets = TRUE)
```

**Arguments**

- **data**: The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as numeric), `quantities` (as numeric - for future calculations) and `prodID` (as numeric, factor or character). A column `retID` (as factor, character or numeric) is also needed if the User wants to impute prices over outlets.

- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".

- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

- **zero_prices**: A logical parameter indicating whether zero prices are to be imputed too (then it is set to TRUE).

- **outlets**: A logical parameter indicating whether imputations are to be done for each outlet separately (then it is set to TRUE).

**Value**

This function imputes missing prices (unit values) and (optionally) zero prices by using carry forward/backward prices. The imputation can be done for each outlet separately or for aggregated data (see the `outlets` parameter). If a missing product has a previous price then that previous price is carried forward until the next real observation. If there is no previous price then the next real observation is found and carried backward. The quantities for imputed prices are set to zeros. The function returns a data frame (monthly aggregated) which is ready for price index calculations.
**Examples**

# Creating a small data set with zero prices:
```r
time<-c("2018-12-01","2019-01-01")
time<-as.Date(c(time., time.))
p1<-c(0,23)
p2<-c(14,0)
q1<-c(15,25)
q2<-c(44,79)
quantities<-c(q1,q2)
prices<-c(p1,p2)
prodID<-c(1,1,2,2)
my_data<-data.frame(time, prices, quantities, prodID)
```

# Price imputing:
```r
data_imputing(my_data, start="2018-12", end="2019-01", zero_prices=TRUE, outlets=FALSE)
```

# Preparing a data set with zero and missing prices:
```r
dataMATCH$prodID<-dataMATCH$codeIN
data<-dplyr::select(dataMATCH, time, prices, quantities, prodID, retID)
set1<-data[1:5,]
set1$prices<-0
set2<-data[6:30,]
set2$prices<-0
df<-rbind(set1, set2)
```

# Price imputing:
```r
data_imputing(df, start="2018-12", end="2019-03", zero_prices=TRUE, outlets=TRUE)
```

---

**data_matching**

**Matching products**

**Description**

This function returns a data set defined in the first parameter (data) with an additional column (prodID). Two products are treated as being matched if they have the same prodID value.

**Usage**

```r
data_matching(
  data,
  start,
  end,
  interval = FALSE,
  variables = c(),
  codeIN = TRUE,
  codeOUT = TRUE,
  description = TRUE,
  onlydescription = FALSE,
  precision = 0.95
)
```
**Arguments**

- **data**: The user’s data frame with information about products to be matched. It must contain columns: `time` (as Date in format: year-month-day, e.g. "2020-12-01") and at least one of the following columns: `codeIN` (as numeric, factor or character), `codeOUT` (as numeric, factor or character) and `description` (as character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the matching process concerns only two periods defined by `start` and `end` parameters (then the `interval` is set to FALSE) or whether that function is to match products sold during the whole time interval `<start, end>.
- **variables**: The optional parameter describing the vector of additional column names. Values of these additional columns must be identical for matched products.
- **codeIN**: A logical value, e.g. if there are retailer (internal) product codes (as numeric or character) written in `codeIN` column and there is a need to use that column while data matching, then that parameter should be set to TRUE. Otherwise it is set to FALSE.
- **codeOUT**: A logical value, e.g. if there are external product codes, such as GTIN or SKU (as numeric or character) written in `codeOUT` column and there is a need to use that column while data preparing then, that parameter should be set to TRUE. Otherwise it is set to FALSE.
- **description**: A logical value, e.g. if there are product labels (as character) written in `description` column and there is a need to use that column while data preparing, then that parameter should be set to TRUE. Otherwise it is set to FALSE.
- **onlydescription**: A logical value indicating whether products with identical labels (described in the `description`) are to be matched.
- **precision**: A threshold value for the Jaro-Winkler similarity measure when comparing labels (its value must belong to the interval [0,1]). Two labels are treated as similar enough if their Jaro-Winkler similarity exceeds the precision value.

**Value**

This function returns a data set defined in the first parameter (`data`) with an additional column (`prodID`). Two products are treated as being matched if they have the same `prodID` value. The procedure of generating the above-mentioned additional column depends on the set of chosen columns for matching. In most extreme case, when the `onlydescription` parameter value is TRUE, two products are also matched if they have identical descriptions. Other cases are as follows: Case 1: Parameters `codeIN`, `codeOUT` and `description` are set to TRUE. Products with two identical codes or one of the codes identical and an identical description are automatically matched. Products are also matched if they have identical one of codes and the Jaro-Winkler similarity of their descriptions is bigger than the precision value. Case 2: Only one of the parameters: `codeIN` or `codeOUT` are set to TRUE and also the `description` parameter is set to TRUE. Products with an identical chosen code and an identical description are automatically matched. In the second stage, products are also matched if they have an identical chosen code and the Jaro-Winkler similarity of their descriptions is bigger than the precision value. Case 3: Parameters `codeIN` and `codeOUT` are
set to TRUE and the parameter `description` is set to FALSE. In this case, products are matched if they have both codes identical. Case 4: Only the parameter `description` is set to TRUE. This case requires the `onlydescription` parameter to be TRUE and then the matching process is based only on product labels (two products are matched if they have identical descriptions). Case 5: Only one of the parameters: `codeIN` or `codeOUT` are set to TRUE and the `description` parameter is set to FALSE. In this case, the only reasonable option is to return the `prodID` column which is identical with the chosen code column. Please note that if the set of column names defined in the `variables` parameter is not empty, then the values of these additional columns must be identical while product matching.

**Examples**

```r
data_matching(dataMATCH, start="2018-12",end="2019-02",onlydescription=TRUE,interval=TRUE)
data_matching(dataMATCH, start="2018-12",end="2019-02",precision=0.98, interval=TRUE)
```

---

**data_norm**

*Normalization of grammage units and recalculation of prices and quantities with respect to these units*

**Description**

The function normalizes grammage units of products and recalculates product prices and quantities with respect to these normalized grammage units.

**Usage**

```r
data_norm(
  data = data.frame(),
  rules = list(c("ml", "l", 1000), c("g", "kg", 1000)),
  all = TRUE
)
```

**Arguments**

- **data** The user's data frame. The data frame must contain the following columns: `prices` (as positive numeric), `quantities` (as positive numeric), `grammage` (as numeric or character) and `unit` (as character).

- **rules** User rules for transforming grammage, unit, prices and quantities of products. For instance, a rule (`"ml", "l", 1000`) changes the 'old' grammage unit: ml into the new one: l on the basis of the provided relation: 1000ml=1l. As a consequence, for each product which is sold in liters l, the unit price and quantity are calculated.

- **all** A logical value indicating whether the resulting data frame is to be limited to products with detected grammage. Its default value is `TRUE` which means that not transformed rows (products) are also returned.
The function returns the user's data frame with two transformed columns: grammage and unit, and two rescaled columns: prices and quantities. The above-mentioned transformation and rescaling take into consideration the user rules. Recalculated prices and quantities concern grammage units defined as the second parameter in the given rule.

Examples

```
# Preparing a data set
data<-data_unit(dataU, units=c("g\mid ml\mid kg\mid l"), multiplication="x")
# Normalization of grammage units
data_norm(data, rules=list(c("ml","l",1000), c("g","kg",1000)))
```

Description

This function returns a prepared data frame based on the user’s data set. The resulting data frame is ready for further data processing (such as data selecting, matching or filtering) and it is also ready for price index calculations (if only it contains required columns).

Usage

```
data_preparing(  
data,  
time = NULL,  
prices = NULL,  
quantities = NULL,  
prodID = NULL,  
retID = NULL,  
description = NULL,  
codeIN = NULL,  
codeOUT = NULL,  
grammage = NULL,  
unit = NULL,  
additional = c(),  
zero_prices = FALSE,  
zero_quantities = TRUE  
)
```

Arguments

- **data** The user’s data frame to be prepared. The user must indicate columns: time (as Date or character type, allowed formats are, eg.: ‘2020-03’ or ‘2020-12-28’), prices and quantities (as numeric). Optionally, the user may also indicate
columns: prodID, codeIN, codeOUT, retID (as numeric, factor or character), description (as character), grammage (as numeric or character), unit (as character) and other columns specified by the additional parameter.

time A character name of the column which provides transaction dates.

prices A character name of the column which provides product prices.

quantities A character name of the column which provides product quantities.

prodID A character name of the column which provides product IDs. The prodID column should include unique product IDs used for product matching (as numeric or character). It is not obligatory to consider this column while data preparing but it is required while price index calculating (to obtain it, please see data_matching).

retID A character name of the column which provides outlet IDs (retailer sale points). The retID column should include unique outlet IDs used for aggregating subindices over outlets. It is not obligatory to consider this column while data preparing but it is required while final price index calculating (to obtain it, please see the final_index function).

description A character name of the column which provides product descriptions. It is not obligatory to consider this column while data preparing but it is required while product selecting (please see the data_selecting function).

codeIN A character name of the column which provides internal product codes (from the retailer). It is not obligatory to consider this column while data preparing but it may be required while product matching (please see the data_matching function).

codeOUT A character name of the column which provides external product codes (e.g. GTIN or SKU). It is not obligatory to consider this column while data preparing but it may be required while product matching (please see the data_matching function).

grammage A character name of the numeric column which provides the grammage of products

unit A character name of the column which provides the unit of the grammage of products

additional A character vector of names of additional columns to be considered while data preparing (records with missing values are deleted).

zero_prices A logical parameter indicating whether zero prices are to be acceptable.

zero_quantities A logical parameter indicating whether zero quantities are to be acceptable.

Value

The resulting data frame is free from: missing values, negative prices (if zero_prices is set to TRUE), zero or negative prices (if zero_prices is set to FALSE), negative quantities (if zero_quantities is set to TRUE) and zero and negative quantities (if zero_prices is set to FALSE). As a result, column time is set to be Date type (in format: ‘Year-Month-01’), columns prices and quantities are set to be numeric. If the column description is selected, then it is set to be character type. If columns: prodID, retID, codeIN or codeOUT are selected, then they are set to be factor type.
data_reducing

Examples

data_preparing(milk, time="time", prices="prices", quantities="quantities")
data_preparing(dataCOICOP, time="time",
prices="prices", quantities="quantities", additional="coicop6")

data_reducing Reducing products

Description

The function returns a reduced data set, i.e. a data set containing sufficiently numerous matched products in the indicated groups. The input data set (data frame) must contain matched products over time, i.e. it must contain the prodID column (as numeric, factor or character), or product descriptions, i.e. it must contain the description column (as character).

Usage

data_reducing(
data,
start,
end,
type = "prodID",
minN = 2,
outlets = FALSE,
by = c(),
interval = FALSE
)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’) and, depending on next parameter values, columns: prodID or description, and retID.

start The base period (as character) limited to the year and month, e.g. ”2020-03”.

end The research period (as character) limited to the year and month, e.g. ”2020-04”.

type This parameter indicates whether group counts are determined by different matched prodIDs (in which case the parameter has the value ‘prodID’) or different matched descriptions (in which case the parameter has the value ‘description’).

minN This parameter determines the minimum size of matched products in groups.

outlets This parameter determines whether grouping is to be done for each outlet separately. If so (if it is TRUE), the data set must contain a column identifying the outlets (retID).

by This parameter specifies the name of the grouping column (as character).
Interval

A logical value indicating whether the reducing process concerns only two periods defined by start and end parameters (then the interval is set to FALSE) or whether that function is to reduce products sold during the whole time interval <start, end>.

Value

The function returns a reduced data set, i.e. a data set containing sufficiently numerous matched products in the indicated groups. For each product group created and for selected periods, the procedure checks that the count of identical prodIDs (or identical product descriptions, which does not necessarily mean the same thing) is at least equal to minN. If it is not, such products are eliminated from the data set. The function performs the check either only for the base and current period (in which case the interval parameter is FALSE) or also for all intermediate months (in which case the interval parameter is TRUE). If the user wants to perform this check for each outlet separately, then the outlets parameter should be set to TRUE.

Examples

data_reducing(sugar, start="2018-12", end="2019-12", by="description", minN=5)

data_selecting

Selecting products from the user's data set for further price index calculations

Description

The function returns a subset of the user's data set obtained by selection based on keywords and phrases.

Usage

data_selecting(
  data,
  include = c(),
  must = c(),
  exclude = c(),
  sensitivity = FALSE,
  coicop = NULL
)

Arguments

data
  The user's data frame. It must contain a column description (as character).
include
  A vector consisting of words and phrases. The function reduces the data set to one in which the description column contains any of these values.
must
  A vector consisting of words and phrases. The function reduces the data set to one in which the description column contains each of these values.
data_unit

**exclude**  A vector consisting of words and phrases. The function reduces the data set to one in which the description column does not contain any of these values.

**sensitivity**  A logical parameter indicating whether sensitivity to lowercase and uppercase letters is taken into consideration (if yes, its value is TRUE).

**coicop**  An optional parameter indicating a value for an additional column coicop which is added to the resulting data frame.

**Value**

The function returns a subset of the user’s data set obtained by selection based on keywords and phrases defined by parameters: include, must and exclude (an additional column coicop is optional). Providing values of these parameters, please remember that the procedure distinguishes between uppercase and lowercase letters only when sensitivity is set to TRUE.

**Examples**

```r
data_selecting(milk, include=c("milk"), must=c("UHT"))
data_selecting(milk, must=c("milk"), exclude=c("paust"))
```

**data_unit**  *Providing information about the grammage and unit of products*

**Description**

The function returns the grammage and unit of products as two additional columns.

**Usage**

```r
data_unit(data = data.frame(), units = c("g|ml|kg|l"), multiplication = "x")
```

**Arguments**

- **data**  The user’s data frame. The data frame must contain the description column (as character).
- **units**  Units of products which are to be detected (e.g. "ml|g|kg")
- **multiplication**  A sign of the multiplication used in product descriptions (e.g. "x")

**Value**

The function returns the user’s data frame with two additional columns: grammage and unit. The values of these columns are extracted from product descriptions on the basis of provided units. Please note, that the function takes into consideration a sign of the multiplication, e.g. if the product description contains: '2x50 g', we obtain: grammage: 100 and unit: g for that product (for multiplication set to 'x').

**Examples**

```r
data_unit(dataU, units=c("g|ml|kg|l"), multiplication="x")
```
Calculating the bilateral Davies price index

Description
This function returns a value (or vector of values) of the bilateral Davies price index.

Usage
davies(data, start, end, interval = FALSE)

Arguments
data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2020-03".
end The research period (as character) limited to the year and month, e.g. "2020-04".
interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value
The function returns a value (or vector of values) of the bilateral Davies price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References

Examples
davies(sugar, start="2018-12", end="2019-12")
davies(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the unweighted Dikhanov price index

Description
This function returns a value (or vector of values) of the unweighted bilateral Dikhanov price index.

Usage

dikhanov(data, start, end, interval = FALSE)

Arguments

data
The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start
The base period (as character) limited to the year and month, e.g. "2020-03".

end
The research period (as character) limited to the year and month, e.g. "2020-04".

interval
A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value
The function returns a value (or vector of values) of the unweighted bilateral Dikhanov price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References

Examples

dikhanov(sugar, start="2018-12", end="2019-12")
dikhanov(milk, start="2018-12", end="2020-01", interval=TRUE)
dissimilarity  Calculating the relative price and/or quantity dissimilarity measure between periods

Description
This function returns a value of the relative price and/or quantity dissimilarity measure.

Usage
dissimilarity(data, period1, period2, type = "p")

Arguments
- **data**: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- **period1**: The first period (as character) limited to the year and month, e.g. '2019-03'.
- **period2**: The second period (as character) limited to the year and month, e.g. '2019-04'.
- **type**: The parameter indicates what type of dissimilarity measure is to be calculated. Possible values of the type parameter are: p (for the price dissimilarity measure calculation), q (for the quantity dissimilarity measure calculation) or pq (for the dSPQ measure calculation, i.e. the measure of relative price and quantity dissimilarity - see References).

Value
This function returns a value of the relative price (dSP) and/or quantity (dSQ) dissimilarity measure. In a special case, when the type parameter is set to pq, the function provides the value of dSPQ measure (the relative price and quantity dissimilarity measure calculated as min(dSP,dSQ).

References

Examples
dissimilarity(milk, period1="2018-12",period2="2019-12",type="q")
dissimilarity(milk, period1="2018-12",period2="2019-12",type="pq")
dissimilarity_fig  

Presenting the relative price and/or quantity dissimilarity measure over time

Description
This function presents values of the relative price and/or quantity dissimilarity measure over time.

Usage

```r
dissimilarity_fig(
  data, 
  start, 
  end, 
  type = "p", 
  benchmark = "end", 
  figure = TRUE, 
  date_breaks = "1 month"
)
```

Arguments

data  The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

start  The base period (as character) limited to the year and month, e.g. '2019-03'.

end  The research period (as character) limited to the year and month, e.g. '2019-07'.

type  The parameter indicates what type of dissimilarity measure is to be calculated. Possible values of the type parameter are: p (for the price dissimilarity measure calculation), q (for the quantity dissimilarity measure calculation) or pq (for the dSPQ measure calculation, i.e. the measure of relative price and quantity dissimilarity - see References).

benchmark  The benchmark period (as character) limited to the year and month, e.g. '2019-07'.

figure  A logical parameter indicating the resulting object. If it is TRUE, the function presents the above-mentioned dissimilarities over time via a figure. Otherwise, the function returns a dataframe.

date_breaks  A string giving the distance between breaks on the X axis like "1 month" (default value) or "4 months".

Value
This function presents values of the relative price and/or quantity dissimilarity measure over time. The user can choose a benchmark period (defined by benchmark) and the type of dissimilarity...
measure is to be calculated (defined by type). The obtained results of dissimilarities over time can be presented in a dataframe form or via a figure (the default value of figure is TRUE, which results in a figure).

References


Examples

dissimilarity_fig(milk, start="2018-12", end="2019-12", type="q", figure=FALSE)
dissimilarity_fig(milk, start="2018-12", end="2019-12", type="pq", benchmark="start")

drobisch Calculating the bilateral Drobisch price index

Description

This function returns a value (or vector of values) of the bilateral Drobisch price index.

Usage

drobisch(data, start, end, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Drobisch price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

drobisch(sugar, start="2018-12", end="2019-12")
drobisch(milk, start="2018-12", end="2020-01", interval=TRUE)

dutot  Calculating the unweighted Dutot price index

Description

This function returns a value (or vector of values) of the unweighted bilateral Dutot price index.

Usage

dutot(data, start, end, interval = FALSE)

Arguments

data  The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start  The base period (as character) limited to the year and month, e.g. ”2020-03”.

end  The research period (as character) limited to the year and month, e.g. ”2020-04”.

interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the unweighted bilateral Dutot price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

dutot(sugar, start="2018-12", end="2019-12")
dutot(milk, start="2018-12", end="2020-01", interval=TRUE)

elasticity Calculating the elasticity of substitution

Description

This function returns a value of the elasticity of substitution

Usage

elasticity(
  data,
  start,
  end,
  method = "lm",
  left = -10,
  right = 10,
  precision = 1e-06
)

Arguments

data The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. "2020-12-01"), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

method The index formula for which the CES index will be equated to calculate the elasticity. Acceptable options are `lm`, `f`, `t`, `w` and `sv`.

left The beginning of an interval for estimation of the elasticity of substitution (its default value is -10).

right The end of an interval for estimation of the elasticity of substitution (its default value is 10).

precision The precision of estimation (a 'stop' condition for the procedure). A default value of this parameter is 0.000001.
Value

This function returns a value of the elasticity of substitution. If the method parameter is set to lm, the procedure of estimation solves the equation: LM(\sigma) - CW(\sigma) = 0 numerically, where LM denotes the Lloyd-Moulton price index, the CW denotes a current weight counterpart of the Lloyd-Moulton price index, and \sigma is the elasticity of substitution parameter, which is estimated. If the method parameter is set to f, the Fisher price index formula is used instead of the CW price index. If the method parameter is set to t, the Tornqvist price index formula is used instead of the CW price index. If the method parameter is set to w, the Walsh price index formula is used instead of the CW price index. If the method parameter is set to sv, the Sato-Vartia price index formula is used instead of the CW price index. The procedure continues until the absolute value of this difference is greater than the value of the 'precision' parameter.

References


Examples

```
elasticity(coffee, start = "2018-12", end = "2019-01")
elasticity(coffee, start = "2018-12", end = "2019-01", method = "f")
elasticity(coffee, start = "2018-12", end = "2019-01", method = "sv")
```

Description

The function provides a data frame or a figure presenting elasticities of substitution calculated for time interval.

Usage

```
elasticity_fig(  
data,  
start,  
end,  
method = c("lm"),  
fixedbase = TRUE,  
figure = TRUE,  
date_breaks = "1 month",  
names = c(),  
left = -10,  
right = 10,  
```
precision = 1e-06

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

dernend The research period (as character) limited to the year and month, e.g. "2020-04".

method A vector indicating index formulas for which the CES index will be equated to calculate the elasticity. Acceptable options are lm, f, t, w and sv or their combinations.

fixedbase A logical parameter indicating whether the procedure is to work for subsequent months from the considered time interval (fixedbase=FALSE). Otherwise the period defined by start plays a role of fixed base month (fixedbase=TRUE)

figure A logical parameter indicating whether the function returns a figure (TRUE) or a data frame (FALSE) with values of elasticity of substitution.

date_breaks A string giving the distance between breaks on the X axis like "1 month" (default value) or "4 months".

names A character string indicating names of indices used for elasticity approximation (see the method parameter).

left The beginning of an interval for estimation of each elasticity of substitution (its default value is -10)

right The end of an interval for estimation of each elasticity of substitution (its default value is 10)

precision The precision of estimation (a 'stop' condition for the procedure). A default value of this parameter is 0.000001.

Value

The function provides a data frame or a figure presenting elasticities of substitution calculated for time interval (see the figure parameter). The elasticities of substitution can be calculated for subsequent months or for a fixed base month (see the start parameter) and rest of months from the given time interval (it depends on the fixedbase parameter). The above-mentioned parameters for compared months are calculated by using the elasticity function.

References


expenditures

Examples

```r
elasticity_fig (milk,start="2018-12",end="2019-04",figure=TRUE,
method=c("lm","f"),names=c("LM","Fisher"))
elasticity_fig (milk,start="2018-12",end="2019-06",figure=FALSE)
```

---

**expenditures**  
*Providing expenditures of sold products*

**Description**

The function returns expenditures of sold products with given IDs.

**Usage**

```
expenditures(data, period, set = c(), ID = FALSE)
```

**Arguments**

- **data**  
The user's data frame. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), quantities (as positive numeric) and prodID (as numeric, factor or character) with unique product IDs.

- **period**  
The time period (as character) limited to the year and month, e.g. "2019-03".

- **set**  
The set of unique product IDs to be used for determining expenditures of sold products (see also `data_matching`). If the set is empty, the function returns quantities of all products being available in period.

- **ID**  
A logical parameter indicating whether a data frame with prodIDs and quantities should be returned.

**Value**

The function analyzes the user's data frame and returns expenditures of products with given ID and being sold in the time period indicated by the `period` parameter. Please note that the function returns the expenditure values for sorted prodIDs and in the absence of a given prodID in the data set, the function returns nothing (it does not return zero). If the ID parameter is set to TRUE then the function returns a data frame with columns: by (IDs of products) and expend (expenditures of products).

**Examples**

```
expenditures(milk, period="2019-06")
expenditures(milk, period="2019-12", set=c(400032, 82919), ID=TRUE)
```
A general function to compute a final price index

Description

This function returns a value (or values) of the selected final price index for the selected type of aggregation of partial results.

Usage

```r
final_index(
  data = data.frame(),
  start = c(),
  end = c(),
  formula = c(),
  window = c(),
  splice = c(),
  base = c(),
  sigma = c(),
  r = c(),
  outlets = FALSE,
  groups = FALSE,
  by = c(),
  aggr = "fisher",
  interval = FALSE
)
```

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character). A column retID (as numeric, factor or character) is also essential if the aggregation over outlets is considered. A column with grouping variable (as numeric, factor or character - indicated by the by parameter) is essential if the aggregation over product subgroups is considered.

start The base period (as character) limited to the year and month, e.g. "2019-12".

data The research period (as character) limited to the year and month, e.g. "2020-04".

formula The character string indicating the price index formula is to be calculated. To see available options please use the link: PriceIndices.

window The length of the time window if the multilateral index is selected (as positive integer: typically multilateral methods are based on the 13-month time window).

splice A character string indicating the splicing method (if the multilateral splicing index is selected). Available options are: "movement", "window", "half", "mean" and their additional variants: "window_published", "half_published" and "mean_published".
The prior period used in the Young- or Lowe-type price indices (as character) limited to the year and month, e.g. "2020-01".

The elasticity of substitution parameter used in the Lloyed-Moulton, AG Mean or GEKS-LM indices (as numeric).

The non-zero parameter used in the quadratic mean of order r quantity / price index or in the GEKS-QM index (as numeric).

A logical parameter indicating whether the aggregation over outlets (defined by retID column) should be done.

A logical parameter indicating whether the aggregation over product subgroups (indicated by 'by' parameter) should be done.

A character string which indicates a column name for creating product subgroups.

The formula used for aggregating partial index results (available values are: "arithmetic", "geometric", "laspeyres", "paasche", "fisher", "tornqvist").

A logical value indicating whether the function is to provide price indices comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be presented (the fixed base month is defined by start).

This general function returns a value or values of the selected final price index for the selected type of aggregation of partial results. If the interval parameter is set to TRUE, then it returns a data frame where its first column indicates dates and the remaining columns show corresponding values of all selected price indices.

Examples

```r
final_index(coffee, start = "2018-12", end = "2019-12",
             formula = "fisher", groups = TRUE, outlets = FALSE,
             aggr = "tornqvist", by = "description")
final_index(milk, start = "2018-12", end = "2019-12",
             formula = "fisher", groups = TRUE, outlets = TRUE,
             aggr = "laspeyres", by = "description",
             interval = TRUE)
```

Description

This function returns a value (or vector of values) of the bilateral Fisher price index.

Usage

```r
fisher(data, start, end, interval = FALSE)
```
Arguments

data  The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ‘2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  The base period (as character) limited to the year and month, e.g. ”2020-03”.

end  The research period (as character) limited to the year and month, e.g. ”2020-04”.

interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Fisher price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

fisher(sugar, start=”2018-12”, end=”2019-12”)
fisher(milk, start=”2018-12”, end=”2020-01”, interval=TRUE)

geary_khamis  Calculating the bilateral Geary-Khamis price index

Description

This function returns a value (or vector of values) of the bilateral Geary-Khamis price index.

Usage

ageary_khamis(data, start, end, interval = FALSE)
Arguments

data        The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start       The base period (as character) limited to the year and month, e.g. "2020-03".

date        The research period (as character) limited to the year and month, e.g. "2020-04".

interval    A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Geary-Khamis price index depending on the interval parameter (please use gk function to calculate the multilateral Geary-Khamis price index). If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

gs = geary_khamis(sugar, start="2018-12", end="2019-12")
gs = geary_khamis(milk, start="2018-12", end="2020-01", interval=TRUE)

Description

This function returns a value of the multilateral GEKS price index (to be more precise: the GEKS index based on the Fisher formula).
Usage

geks(data, start, end, wstart = start, window = 13)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

wstart The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS price index (to be more precise: the GEKS index based on the Fisher formula) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geks(milk, start="2019-01", end="2019-08", window=10)
geks(milk, start="2018-12", end="2019-12")
Calculating the multilateral GEKS-AQI price index

Description

This function returns a value of the multilateral GEKS-AQI price index (to be more precise: the GEKS index based on the asynchronous quality adjusted price index formula).

Usage

geksaqi(data, start, end, wstart = start, window = 13)

Arguments

data          The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start         The base period (as character) limited to the year and month, e.g. "2020-03".
end           The research period (as character) limited to the year and month, e.g. "2020-04".
wstart        The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".
window        The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS-AQI price index (to be more precise: the GEKS index based on the asynchronous quality adjusted price index formula) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References

Examples

geksaqi(milk, start="2019-01", end="2019-08", window=10)
geksaqi(milk, start="2018-12", end="2019-12")

description

This function returns a value of the multilateral GEKS-AQI price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

geksaqi_fbew(data, start, end)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2019-12". 
end The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral GEKS-AQI price index (the GEKS index based on the asynchronous quality adjusted price index formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geksaqi_fbew(milk, start="2018-12", end="2019-08")

---

geksaqi_fbmw | Extending the multilateral GEKS-AQI price index by using the FBMW method.

Description

This function returns a value of the multilateral GEKS-AQI price index extended by using the FBMW (Fixed Base Moving Window) method.

Usage

geksaqi_fbmw(data, start, end)

Arguments

data | The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start | The base period (as character) limited to the year and month, e.g. "2019-12".

de | The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral GEKS-AQI price index (the GEKS index based on the asynchronous quality adjusted price index formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)-1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
gexaqi_fbmw(milk, start="2019-12", end="2020-04")
```

---

gexaqi_splice

*Extending the multilateral GEKS-AQI price index by using window splicing methods.*

Description

This function returns a value (or values) of the multilateral GEKS-AQI price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

```r
gexaqi_splice(  
data,  
start,  
end,  
window = 13,  
splice = "movement",  
interval = FALSE  
)
```

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
The base period (as character) limited to the year and month, e.g. "2019-12".

The research period (as character) limited to the year and month, e.g. "2020-04".

The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".

A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

This function returns a value or values (depending on interval parameter) of the multilateral GEKS-AQI price index (the GEKS index based on the asynchronous quality adjusted price index formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).


Examples

```r
geksaqi_splice(milk, start="2018-12", end="2020-02", splice="half")
```
Calculating the multilateral GEKS-AQU price index

Description

This function returns a value of the multilateral GEKS-AQU price index (to be more precise: the GEKS index based on the asynchronous quality adjusted unit value formula).

Usage

geksaqu(data, start, end, wstart = start, window = 13)

Arguments

data: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start: The base period (as character) limited to the year and month, e.g. "2020-03".

end: The research period (as character) limited to the year and month, e.g. "2020-04".

wstart: The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

window: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS-AQU price index (to be more precise: the GEKS index based on the asynchronous quality adjusted unit value formula) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


**geksaqu_fbew**

### Examples

```r
data(milk)
geksaqu(milk, start="2019-01", end="2019-08", window=10)
geksaqu(milk, start="2018-12", end="2019-12")
```

---

**Description**

This function returns a value of the multilateral GEKS-AQU price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

**Usage**

```r
geksaqu_fbew(data, start, end)
```

**Arguments**

- **data**: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

**Value**

This function returns a value of the multilateral GEKS-AQU price index (the GEKS index based on the asynchronous quality adjusted unit value formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**


**Examples**

```r
geksaqu_fbew(milk, start="2018-12", end="2019-08")
```

**Description**

This function returns a value of the multilateral GEKS-AQU price index extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

```r
geksaqu_fbmw(data, start, end)
```

**Arguments**

- `data` The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- `start` The base period (as character) limited to the year and month, e.g. "2019-12".
- `end` The research period (as character) limited to the year and month, e.g. "2020-04".

**Value**

This function returns a value of the multilateral GEKS-AQU price index (the GEKS index based on the asynchronous quality adjusted unit value formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as `year(end)-1`. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

geksaqu_fbmw(milk, start="2019-12", end="2020-04")

geksaqu_splice       Extending the multilateral GEKS-AQU price index by using window splicing methods.

Description

This function returns a value (or values) of the multilateral GEKS-AQU price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

geksaqu_splice(
  data,  
  start,  
  end,  
  window = 13,  
  splice = "movement",  
  interval = FALSE
)

Arguments

data       The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start  The base period (as character) limited to the year and month, e.g. "2019-12".
end    The research period (as character) limited to the year and month, e.g. "2020-04".
window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
splice A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".
interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value
This function returns a value or values (depending on interval parameter) of the multilateral GEKS-AQU price index (the GEKS index based on the asynchronous quality adjusted unit value formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References

Examples
geksaqu_splice(milk, start="2018-12", end="2020-02", splice="half")
Calculating the multilateral GEKS-GAQI price index

Description

This function returns a value of the multilateral GEKS-GAQI price index (to be more precise: the GEKS index based on the geometric asynchronous quality adjusted price index formula).

Usage

geksgaqi(data, start, end, wstart = start, window = 13)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2020-03".
end The research period (as character) limited to the year and month, e.g. "2020-04".
wstart The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".
window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS-GAQI price index (to be more precise: the GEKS index based on the geometric asynchronous quality adjusted price index formula) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


geksgaqi_fbew

**Examples**

geksgaqi(milk, start="2019-01", end="2019-08", window=10)
geksgaqi(milk, start="2018-12", end="2019-12")

---

**Description**

This function returns a value of the multilateral GEKS-GAQI price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

**Usage**

geksgaqi_fbew(data, start, end)

**Arguments**

data

The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start

The base period (as character) limited to the year and month, e.g. "2019-12".

end

The research period (as character) limited to the year and month, e.g. "2020-04".

**Value**

This function returns a value of the multilateral GEKS-GAQI price index (the GEKS index based on the geometric asynchronous quality adjusted price index formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

**References**


Examples

```r
geksgaqi_fbew(milk, start="2018-12", end="2019-08")
```

---

### geksgaqi_fbmw

**Extending the multilateral GEKS-GAQI price index by using the FBMW method.**

**Description**

This function returns a value of the multilateral GEKS-GAQI price index extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

```r
geksgaqi_fbmw(data, start, end)
```

**Arguments**

- **data**
  
The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

- **start**
  
The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**
  
The research period (as character) limited to the year and month, e.g. "2020-04".

**Value**

This function returns a value of the multilateral GEKS-GAQI price index (the GEKS index based on the geometric asynchronous quality adjusted price index formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods `end` and `start` and it uses a 13-month time window with a fixed base month taken as `year(end)-1`. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

geksgaqi_fbmw(milk, start="2019-12", end="2020-04")

---

geksgaqi_splice

*Extending the multilateral GEKS-GAQI price index by using window splicing methods.*

Description

This function returns a value (or values) of the multilateral GEKS-GAQI price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

```r
geksgaqi_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)
```

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
splice A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".

interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value

This function returns a value or values (depending on interval parameter) of the multilateral GEKS-GAQI price index (the GEKS index based on the geometric asynchronous quality adjusted price index formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geksgaqi_splice(milk, start="2018-12", end="2020-01", window=10)
Calculating the multilateral GEKS-GL price index

Description

This function returns a value of the multilateral GEKS-GL price index (to be more precise: the GEKS index based on the geometric Laspeyres formula).

Usage

geksgl(data, start, end, wstart = start, window = 13)

Arguments

data
  The user's data frame with information about sold products. It must contain
columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices
(as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start
  The base period (as character) limited to the year and month, e.g. "2020-03".

end
  The research period (as character) limited to the year and month, e.g. "2020-04".

wstart
  The beginning of the time interval (which is used by multilateral methods) limited
to the year and month, e.g. "2020-01".

window
  The length of the time window (as positive integer: typically multilateral meth-
ods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS-GL price index (to be more precise: the GEKS index based on the geometric Laspeyres formula) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geksl(milk, start="2019-01", end="2019-08",window=10)
geksl(milk, start="2018-12", end="2019-12")

Description

This function returns a value of the multilateral GEKS-GL price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

geksl_fbew(data, start, end)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral GEKS-GL price index (the GEKS index based on the geometric Laspeyres formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


**Examples**

```r
geksgl_fbw(milk, start="2018-12", end="2019-08")
```

**Description**

This function returns a value of the multilateral GEKS-GL price index extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

```r
geksgl_fbw(data, start, end)
```

**Arguments**

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. ’2020-12-01’), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. ”2019-12”.
- **end**: The research period (as character) limited to the year and month, e.g. ”2020-04”.

**Value**

This function returns a value of the multilateral GEKS-GL price index (the GEKS index based on the geometric Laspeyres formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods `end` and `start` and it uses a 13-month time window with a fixed base month taken as `year(end)-1`. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
geksgl_fbmw(milk, start="2019-12", end="2020-04")
```

geksgl_splice

*Extending the multilateral GEKS-GL price index by using window splicing methods.*

Description

This function returns a value (or values) of the multilateral GEKS-GL price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

```r
geksgl_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)
```

Arguments

- **data** The user's data frame with information about sold products. It must contain columns: `time` (as `Date` in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
start  The base period (as character) limited to the year and month, e.g. "2019-12".
end   The research period (as character) limited to the year and month, e.g. "2020-04".
window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
splice A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".
interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value

This function returns a value or values (depending on interval parameter) of the multilateral GEKS-GL price index (the GEKS index based on the geometric Laspeyres formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geksgl_splice(milk, start="2018-12", end="2020-02", splice="half")
Calculating the multilateral GEKS-IQM price index

Description

This function returns a value of the multilateral GEKS-IQM price index (to be more precise: the GEKS index based on the implicit quadratic mean of order r price index IQMp).

Usage

geksiqm(data, start, end, r = 2, wstart = start, window = 13)

Arguments

data

The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ‘2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start

The base period (as character) limited to the year and month, e.g. “2020-03”.

end

The research period (as character) limited to the year and month, e.g. “2020-04”.

r

The real and non-zero parameter used in the implicit quadratic mean of order r price index.

wstart

The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. “2020-01”.

window

The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS-IQM price index (to be more precise: the GEKS index based on the the implicit quadratic mean of order r price index IQMp) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

df <- geksiqm(milk, start="2019-01", end="2019-08", window=10)
df <- geksiqm(milk, start="2018-12", end="2019-12", r=1.6)

---

description

This function returns a value of the multilateral GEKS-IQM price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

df <- geksiqm_fbew(data, start, end, r)

Arguments

data: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. "2020-12-01"), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start: The base period (as character) limited to the year and month, e.g. "2019-12".

df <- geksiqm_fbew(data, start="2019-01", end="2019-08", window=10)

df <- geksiqm(milk, start="2018-12", end="2019-12", r=1.6)

---

Value

This function returns a value of the multilateral GEKS-IQM price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

gksiqm_fbw(milk, start="2018-12", end="2019-08", r=1.2)

---

**Description**

This function returns a value of the multilateral GEKS-IQM price index extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

gksiqm_fbw(data, start, end, r)

**Arguments**

- **data** The user’s data frame with information about sold products. It must contain columns: *time* (as Date in format: year-month-day, e.g. ’2020-12-01’), *prices* (as positive numeric), *quantities* (as positive numeric) and *prodID* (as numeric, factor or character).
- **start** The base period (as character) limited to the year and month, e.g. ”2019-12”.
- **end** The research period (as character) limited to the year and month, e.g. ”2020-04”.
- **r** The real and non-zero parameter used in the implicit quadratic mean of order r price index.

**Value**

This function returns a value of the multilateral GEKS-IQM price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)-1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
geksiqm_splice(milk, start="2019-12", end="2020-04", r=1.6)
```

**Description**

This function returns a value (or values) of the multilateral GEKS-IQM price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

**Usage**

```r
geksiqm_splice(
    data, start, end, r = 2, window = 13,
    splice = "movement", interval = FALSE
)
```

**Arguments**

- **data**: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. "2020-12-01"), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
geksiqm_splice

\( r \)  
The real and non-zero parameter used in the implicit quadratic mean of order \( r \) price index.

\text{window}  
The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

\text{splice}  
A character string indicating the splicing method. Available options are: "movement","window","half","mean","window_published","half_published","mean_published".

\text{interval}  
A logical value indicating whether the function is to provide the price index comparing the research period defined by \text{end} to the base period defined by \text{start} (then \text{interval} is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by \text{start}).

\text{Value}  
This function returns a value or values (depending on \text{interval} parameter) of the multilateral GEKS-IQM price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in \text{start} and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: \text{price_indices} or \text{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \text{final_index} function).

\text{References}  


\text{Examples}  

geksiqm_splice(milk, start="2018-12", end="2020-02", r=0.8, splice="half")
Calculating the multilateral GEKS price index based on the Jevons formula (typical notation: GEKS-J)

Description
This function returns a value of the multilateral GEKS-J price index (to be more precise: the GEKS index based on the Jevons formula).

Usage
geksj(data, start, end, wstart = start, window = 13)

Arguments
- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. ‘2020-12-01’), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character). A column `quantities` is needed because this function uses unit values as monthly prices.
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **wstart**: The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".
- **window**: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value
This function returns a value of the multilateral GEKS-J price index (to be more precise: the GEKS index based on the Jevons formula) which considers the time window defined by `wstart` and `window` parameters. It measures the price dynamics by comparing period `end` to period `start` (both `start` and `end` must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

geksj(milk, start="2019-01", end="2019-08", window=10)
geksj(milk, start="2018-12", end="2019-12")

Description

This function returns a value of the multilateral GEKS-J price index (i.e. the GEKS price index based on the Jevons formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

geksj_fbew(data, start, end)

Arguments

data
The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character). A column quantities is needed because this function uses unit values as monthly prices.

start
The base period (as character) limited to the year and month, e.g. "2019-12".

end
The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral GEKS-J price index (i.e. the GEKS price index based on the Jevons formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

```r
geksj_fbew(milk, start="2018-12", end="2019-08")
```

---

**geksj_fbew**

Extending the multilateral GEKS-J price index by using the FBMW method.

**Description**

This function returns a value of the multilateral GEKS-J price index (i.e. the GEKS price index based on the Jevons formula) extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

```r
geksj_fbew(data, start, end)
```

**Arguments**

- **data**
  
  The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character). A column `quantities` is needed because this function uses unit values as monthly prices.

- **start**
  
  The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**
  
  The research period (as character) limited to the year and month, e.g. "2020-04".

**Value**

This function returns a value of the multilateral GEKS-J price index (i.e. the GEKS price index based on the Jevons formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods `end` and `start` and it uses a 13-month time window with a fixed base month taken as `year(end)-1`. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
geksj_splice(milk, start="2019-12", end="2020-04")
```

---

**geksj_splice**

*Extending the multilateral GEKS-J price index by using window splicing methods.*

**Description**

This function returns a value (or values) of the multilateral GEKS-J price index (GEKS based on the Jevons formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

**Usage**

```r
geksj_splice(
  data,  
  start, 
  end, 
  window = 13, 
  splice = "movement", 
  interval = FALSE 
)
```

**Arguments**

- **data** The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character). A column quantities is needed because this function uses unit values as monthly prices.

- **start** The base period (as character) limited to the year and month, e.g. "2019-12".

- **end** The research period (as character) limited to the year and month, e.g. "2020-04".
window
The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

splice
A character string indicating the splicing method. Available options are: "movement","window","half","mean", "window_published","half_published","mean_published".

interval
A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value
This function returns a value or values (depending on interval parameter) of the multilateral GEKS-J price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples
geksj_splice(milk, start="2018-12", end="2020-02",splice="half")
Calculating the multilateral GEKS-L price index

Description

This function returns a value of the multilateral GEKS-L price index (to be more precise: the GEKS index based on the Laspeyres formula).

Usage

geksl(data, start, end, wstart = start, window = 13)

Arguments

data  
The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  
The base period (as character) limited to the year and month, e.g. "2020-03".

end  
The research period (as character) limited to the year and month, e.g. "2020-04".

wstart  
The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

window  
The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS-L price index (to be more precise: the GEKS index based on the Laspeyres formula) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geksl(milk, start="2019-01", end="2019-08", window=10)
geksl(milk, start="2018-12", end="2019-12")

---

gekslm Calculating the multilateral GEKS-LM price index

Description

This function returns a value of the multilateral GEKS-LM price index (to be more precise: the GEKS index based on the Lloyd-Moulton price index).

Usage

gekslm(data, start, end, sigma = 0.7, wstart = start, window = 13)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

sigma The elasticity of substitution (a parameter used in the Lloyd-Moulton index formula).

wstart The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS-LM price index (to be more precise: the GEKS index based on the Lloyd-Moulton price index) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
gekslm_fbew

References


Examples

```r
gekslm(milk, start="2019-01", end="2019-08", window=10)
gekslm(milk, start="2018-12", end="2019-12", sigma=0.5)
```

---

**gekslm_fbew**

*Extending the multilateral GEKS-LM price index by using the FBEW method.*

Description

This function returns a value of the multilateral GEKS-LM price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

```r
gekslm_fbew(data, start, end, sigma)
```

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

- **sigma**: The elasticity of substitution (a parameter used in the Lloyd-Moulton index formula).
Value

This function returns a value of the multilateral GEKS-LM price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

gekslm_fbew(milk, start="2018-12", end="2019-08", sigma=1.2)

Description

This function returns a value of the multilateral GEKS-LM price index extended by using the FBMW (Fixed Base Moving Window) method.

Usage

gekslm_fbmw(data, start, end, sigma)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. ”2019-12”.

The research period (as character) limited to the year and month, e.g. "2020-04".

The elasticity of substitution (a parameter used in the Lloyd-Moulton index formula).

### Value

This function returns a value of the multilateral GEKS-LM price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)−1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

### References


### Examples

```r
gksqm_fbmw(milk, start="2019-12", end="2020-04", r=1.6)
```

### Description

This function returns a value (or values) of the multilateral GEKS-LM price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

### Usage

```r
gkslm_splice(
  data,
  start,
  end,
)```
sigma = 0.7,
window = 13,
splice = "movement",
interval = FALSE
)

 Arguments

 data  The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ‘2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

 start  The base period (as character) limited to the year and month, e.g. "2019-12".

 end  The research period (as character) limited to the year and month, e.g. "2020-04".

 sigma  The elasticity of substitution (a parameter used in the Lloyd-Moulton index formula).

 window  The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

 splice  A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".

 interval  A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

 Value

 This function returns a value or values (depending on interval parameter) of the multilateral GEKS-LM price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

 References


**Examples**

```r
gekslm_splice(milk, start="2018-12", end="2020-02", sigma=0.8, splice="half")
```

---

**geksl_fbew**

*Extending the multilateral GEKS-L price index by using the FBEW method.*

---

**Description**

This function returns a value of the multilateral GEKS-L price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

**Usage**

```r
geksl_fbew(data, start, end)
```

**Arguments**

- `data`: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

- `start`: The base period (as character) limited to the year and month, e.g. "2019-12".

- `end`: The research period (as character) limited to the year and month, e.g. "2020-04".

**Value**

This function returns a value of the multilateral GEKS-L price index (the GEKS index based on the Laspeyres formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods `end` and `start`. The month of the `start` parameter must be December. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
geksl_fbew(milk, start="2018-12", end="2019-08")
```

---

**geksl_fbmw**

*Extending the multilateral GEKS-L price index by using the FBMW method.*

**Description**

This function returns a value of the multilateral GEKS-L price index extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

```r
geksl_fbmw(data, start, end)
```

**Arguments**

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
Value

This function returns a value of the multilateral GEKS-L price index (the GEKS index based on the Laspeyres formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods \texttt{end} and \texttt{start} and it uses a 13-month time window with a fixed base month taken as \texttt{year(end)}-1. If the distance between \texttt{end} and \texttt{start} exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the \texttt{start} parameter must be December. To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References


Examples

\begin{verbatim}
geksl_fbmw(milk, start="2019-12", end="2020-04")
\end{verbatim}

\begin{verbatim}
geksl_splice
\end{verbatim}

Extending the multilateral GEKS-L price index by using window splicing methods.

Description

This function returns a value (or values) of the multilateral GEKS-L price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).
Usage

geksl_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

splice A character string indicating the splicing method. Available options are: "movement", "window","half","mean", "window_published","half_published","mean_published".

interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value

This function returns a value or values (depending on interval parameter) of the multilateral GEKS-L price index (the GEKS index based on the Laspeyres formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


**Examples**

```r
gexsl_splice(milk, start="2018-12", end="2020-02", splice="half")
```

**Description**

This function returns a value of the multilateral GEKS-QM price index (to be more precise: the GEKS index based on the quadratic mean of order r price index QMp).

**Usage**

```r
gexsm(data, start, end; r = 2, wstart = start, window = 13)
```

**Arguments**

- **data**: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **r**: The real and non-zero parameter used in the implicit quadratic mean of order r price index.
- **wstart**: The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".
- **window**: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
Value

This function returns a value of the multilateral GEKS-QM price index (to be more precise: the GEKS index based on the the quadratic mean of order r price index QMp) which considers the time window defined by \textit{wstart} and \textit{window} parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References


Examples

\begin{verbatim}
geksqm(milk, start="2019-01", end="2019-08",window=10)
geksqm(milk, start="2018-12", end="2019-12", r=1.6)
\end{verbatim}

\begin{verbatim}
geksqm_fbw
Extending the multilateral GEKS-QM price index by using the FBEW method.
\end{verbatim}

Description

This function returns a value of the multilateral GEKS-QM price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

\begin{verbatim}
geksqm_fbw(data, start, end, r)
\end{verbatim}

Arguments

\begin{verbatim}
data The user’s data frame with information about sold products. It must contain columns: \textit{time} (as Date in format: year-month-day, e.g. '2020-12-01'), \textit{prices} (as positive numeric), \textit{quantities} (as positive numeric) and \textit{prodID} (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2019-12".
end The research period (as character) limited to the year and month, e.g. "2020-04".
r The real and non-zero parameter used in the implicit quadratic mean of order r price index.
\end{verbatim}
Value

This function returns a value of the multilateral GEKS-QM price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods `end` and `start`. The month of the `start` parameter must be December. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product sub-groups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

geksqm_fbw(milk, start="2018-12", end="2019-08", r=1.2)

---

geksqm_fbw

Extending the multilateral GEKS-QM price index by using the FBMW method.

Description

This function returns a value of the multilateral GEKS-QM price index extended by using the FBMW (Fixed Base Moving Window) method.

Usage

geksqm_fbw(data, start, end, r)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2019-12".
The research period (as character) limited to the year and month, e.g. "2020-04".

r

The real and non-zero parameter used in the implicit quadratic mean of order r price index.

Value

This function returns a value of the multilateral GEKS-QM price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)-1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product sub-groups (to consider these types of aggregating, please use the final_index function).

References


Examples

gksqm_fbmw(milk, start="2019-12", end="2020-04", r=1.6)

---

gksqm_splice

Extending the multilateral GEKS-QM price index by using window splicing methods.

Description

This function returns a value (or values) of the multilateral GEKS-QM price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

gksqm_splice(
   data,
   start,
   end,
geksqm_splice

r = 2,
window = 13,
splice = "movement",
interval = FALSE
)

Arguments

data The user’s data frame with information about sold products. It must contain
columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices
(as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

r The real and non-zero parameter used in the implicit quadratic mean of order r
price index.

window The length of the time window (as positive integer: typically multilateral
methods are based on the 13-month time window).

splice A character string indicating the splicing method. Available options are: "movement",
"window","half","mean","window_published","half_published","mean_published".

interval A logical value indicating whether the function is to provide the price index
comparing the research period defined by end to the base period defined by
start (then interval is set to FALSE) or all fixed base multilateral indices are
to be presented (the fixed base month is defined by start).

Value

This function returns a value or values (depending on interval parameter) of the multilateral
GEKS-QM price index extended by using window splicing methods. Available splicing methods
are: movement splice, window splice, half splice, mean splice and their additional variants: window
splice on published indices (WISP), half splice on published indices (HASP) and mean splice
on published indices (see References). The time window starts in start and should consist of
at least two months. To get information about both price index values and corresponding dates,
please see functions: price_indices or final_index. The function does not take into account
aggregating over outlets or product subgroups (to consider these types of aggregating, please use
the final_index function).

References

presented at the 16th Meeting of the Ottawa Group on Price Indices, 8-10 May 2019, Rio de Janeiro,
Brazil.


Adjusted Price Indices with No Characteristic Information. Paper presented at the UNECE-ILO
Calculating the multilateral GEKS price index based on the Walsh formula (GEKS-W)

Description

This function returns a value of the multilateral GEKS-W price index, i.e. the GEKS price index based on the superlative Walsh index formula.

Usage

geksw(data, start, end, wstart = start, window = 13)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

wstart The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral GEKS-W price index (to be more precise: the GEKS index based on the Walsh formula) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

```
geksw(milk, start="2019-01", end="2019-08", window=10)
geksw(milk, start="2018-12", end="2019-12")
```

---

**Description**

This function returns a value of the multilateral GEKS-W price index (GEKS based on the Walsh formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

**Usage**

```
geksw_fbew(data, start, end)
```

**Arguments**

- `data` - The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. `'2020-12-01`), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- `start` - The base period (as character) limited to the year and month, e.g. `'2019-12`.
- `end` - The research period (as character) limited to the year and month, e.g. `'2020-04`.

**Value**

This function returns a value of the multilateral GEKS-W price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods `end` and `start`. The month of the `start` parameter must be December. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
geksw_fbmw(milk, start="2018-12", end="2019-08")
```

---

**Description**

This function returns a value of the multilateral GEKS-W price index (GEKS based on the Walsh formula) extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

```r
geksw_fbmw(data, start, end)
```

**Arguments**

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. ’2020-12-01’), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. ”2019-12”.
- **end**: The research period (as character) limited to the year and month, e.g. ”2020-04”.

**Value**

This function returns a value of the multilateral GEKS-W price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods `end` and `start` and it uses a 13-month time window with a fixed base month taken as `year(end)-1`. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

geksw_fbmw(milk, start="2019-12", end="2020-04")

---

geksw_splice

Extending the multilateral GEKS-W price index by using window splicing methods.

Description

This function returns a value (or values) of the multilateral GEKS-W price index (GEKS based on the Walsh formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

geksw_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".
The research period (as character) limited to the year and month, e.g. "2020-04".

The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".

A logical value indicating whether the function is to provide the price index comparing the research period defined by `end` to the base period defined by `start` (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by `start`).

This function returns a value or values (depending on interval parameter) of the multilateral GEKS-W price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in `start` and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).


geksw_splice(milk, start="2018-12", end="2020-02", splice="half")
geks_fbew

Extending the multilateral GEKS price index by using the FBEW method.

Description

This function returns a value of the multilateral GEKS price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

geks_fbew(data, start, end)

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral GEKS price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

geks_fbw(milk, start="2018-12", end="2019-08")

Description

This function returns a value of the multilateral GEKS price index extended by using the FBMW (Fixed Base Moving Window) method.

Usage

geks_fbw(data, start, end)

Arguments

data  The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  The base period (as character) limited to the year and month, e.g. "2019-12".

end  The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral GEKS price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)-1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geks_fbmw(milk, start="2019-12", end="2020-04")

Description

This function returns a value (or values) of the multilateral GEKS price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

geks_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)

Arguments

data  The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. "2020-12-01"), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start  The base period (as character) limited to the year and month, e.g. "2019-12".
end  The research period (as character) limited to the year and month, e.g. "2020-04".
window  The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
splice  A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".
interval  A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).
Value

This function returns a value or values (depending on interval parameter) of the multilateral GEKS price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geks_splice(milk, start="2018-12", end="2020-02", splice="half")

generate

Generating an artificial scanner dataset

Description

This function provides artificial scanner datasets where prices and quantities are lognormally distributed.

Usage

generate(
    pmi = c(),
    psigma = c(),
    qmi = c(),
    qsigma = c(),
)
generate

prec = c(2, 0),
n = 100,
n0 = 1,
r = 1,
r0 = 1,
start,
days = FALSE
)

Arguments

pmi  A numeric vector indicating \( m_i \) parameters for lognormally distributed prices from the subsequent months.

psigma  A numeric vector indicating \( \sigma \) parameters for lognormally distributed prices from the subsequent months.

qmi  A numeric vector indicating \( m_i \) parameters for lognormally distributed quantities from the subsequent months.

qsigma  A numeric vector indicating \( \sigma \) parameters for lognormally distributed quantities from the subsequent months.

prec  A two-dimensional numeric vector indicating precision, i.e. the number of decimal places, for presenting prices and quantities.

n  An integer parameter indicating the number of products which are to be generated.

n0  An integer parameter indicating the first (the smallest) prodID.

r  An integer parameter indicating the number of outlets (retailer sale points) for which prices and quantities are to be generated.

r0  r0 An integer parameter indicating the first (the smallest) retID.

start  The first period in the generated data frame (as character) limited to the year and month, e.g. '2019-12'.

days  A logical parameter indicating whether the trading day in a given month is to be randomised. The default value of days is FALSE, which means that each transaction for a given month takes place on the first day of the month.

Value

This function returns an artificial scanner dataset where prices and quantities are lognormally distributed. The characteristics for these lognormal distributions are set by pmi, psigma, qmi and qsigma parameters. This function works for a fixed number of products and outlets (see n and r parameters). The generated dataset is ready for further price index calculations.

References

Examples

```r
generate(pmi=c(1.02,1.03,1.04),psigma=c(0.05,0.09,0.02),qmi=c(3,4,4),
qsigma=c(0.1,0.1,0.15),start="2020-01",days=TRUE)
generate(pmi=c(1.02,1.03,1.04),psigma=c(0.05,0.09,0.02),qmi=c(6,6,7),
qsigma=c(0.1,0.1,0.15),start="2020-01",n=1000,n0=132578,r=10)
```

---

**generate_CES**

Generating an artificial scanner dataset in the CES model

**Description**

This function provides artificial scanner datasets where prices are lognormally distributed and quantities are obtained under a CES utility.

**Usage**

```r
generate_CES(
  pmi = c(),
  psigma = c(),
  prec = 2,
  elasticity = 0.7,
  S = 1000,
  alfa = c(),
  n = 100,
  n0 = 1,
  r = 1,
  r0 = 1,
  start,
  days = FALSE
)
```

**Arguments**

- **pmi**
  A numeric vector indicating mi parameters for lognormally distributed prices from the subsequent months.

- **psigma**
  A numeric vector indicating sigmam parameters for lognormally distributed prices from the subsequent months.

- **prec**
  A numeric value indicating precision, i.e. the number of decimal places, for generating prices.

- **elasticity**
  The elasticity of substitution. The default value is 0.7.

- **S**
  Sum of spending. The default value is 1000.

- **alfa**
  A numeric vector indicating positive weights that reflect the consumer preferences. By default, this vector is randomized based on a uniform distribution.

- **n**
  An integer parameter indicating the number of products which are to be generated.
An integer parameter indicating the first (the smallest) prodID.

An integer parameter indicating the number of outlets (retailer sale points) for which prices and quantities are to be generated.

r0

An integer parameter indicating the first (the smallest) retID.

The first period in the generated data frame (as character) limited to the year and month, e.g. '2019-12'.

da days

A logical parameter indicating whether the trading day in a given month is to be randomised. The default value of days is FALSE, which means that each transaction for a given month takes place on the first day of the month.

This function returns an artificial scanner dataset where prices are lognormally distributed, quantities are calculated under the assumption that consumers have CES (Constant Elasticity of Substitution) preferences and their spending on all products is S. The characteristics for the lognormal price distribution are set by pmi and psigma parameters. This function works for a fixed number of products and outlets (see n and r parameters). The generated dataset is ready for further price index calculations.


#Generating an artificial dataset (the elasticity of substitution is 1.25)
df<-generate_CES(pmi=c(1.02,1.03),psigma=c(0.04,0.03),
                  elasticity=1.25,start="2020-01",n=100,days=TRUE)
#Verifying the elasticity of substitution
elasticity(df, start="2020-01",end="2020-02")

description

Calculating the bilateral geohybrid price index

This function returns a value (or vector of values) of the bilateral geohybrid price index. The geohybrid index was proposed by Bialek (2020) and it uses correlation coefficients between prices and quantities.

Usage

dend start, end, base = start, interval = FALSE)
Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

base The prior period used in the geohybrid price index formula (as character) limited to the year and month, e.g. "2020-01"

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral geohybrid price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geohybrid(sugar, start="2019-12", end="2020-08", base="2018-12")
geohybrid(milk, start="2019-12", end="2020-08", base="2018-12", interval=TRUE)

describe economidecision, allow for optimal decision making.

geolaspeyres Calculating the bilateral geo-logarithmic Laspeyres price index

Description

This function returns a value (or vector of values) of the bilateral geo-logarithmic Laspeyres price index.

Usage

geolaspeyres(data, start, end, interval = FALSE)
Arguments

data  The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  The base period (as character) limited to the year and month, e.g. "2020-03".

end  The research period (as character) limited to the year and month, e.g. "2020-04".

interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral geo-logarithmic Laspeyres price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geolaspeyres(sugar, start="2018-12", end="2019-12")
geolaspeyres(milk, start="2018-12", end="2020-01", interval=TRUE)

description

This function returns a value (or vector of values) of the bilateral geometric Lowe price index.

Usage

geolowe(data, start, end, base = start, interval = FALSE)
Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

base The prior period used in the geometric Lowe price index formula (as character) limited to the year and month, e.g. "2020-01"

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral geometric Lowe price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

geolowe(sugar, start="2019-01", end="2020-01",base="2018-12")
geolowe(milk, start="2018-12", end="2020-01", interval=TRUE)

geopaasche Calculating the bilateral geo-logarithmic Paasche price index

Description

This function returns a value (or vector of values) of the bilateral geo-logarithmic Paasche price index.

Usage

geopaasche(data, start, end, interval = FALSE)
Arguments

data
The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start
The base period (as character) limited to the year and month, e.g. "2020-03".

end
The research period (as character) limited to the year and month, e.g. "2020-04".

interval
A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value
The function returns a value (or vector of values) of the bilateral geo-logarithmic Paasche price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

doubles(geogaasche(sugar, start="2018-12", end="2019-12")
doubles(geogaasche(milk, start="2018-12", end="2020-01", interval=TRUE)

Description
This function returns a value (or vector of values) of the bilateral geometric Young price index.

Usage

doubles(data, start, end, base = start, interval = FALSE)
Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

base The prior period used in the geometric Young price index formula (as character) limited to the year and month, e.g. "2020-01".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral geometric Young price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

goyoung(sugar, start="2019-01", end="2020-01",base="2018-12")
goyoung(milk, start="2018-12", end="2020-01", interval=TRUE)

gk Calculating the multilateral Geary-Khamis price index

Description

This function returns a value of the multilateral Geary-Khamis price index.

Usage

gk(data, start, end, wstart = start, window = 13)
Arguments

data  The user's data frame with information about sold products. It must contain columns: \textit{time} (as Date in format: year-month-day, e.g. '2020-12-01'), \textit{prices} (as positive numeric), \textit{quantities} (as positive numeric) and \textit{prodID} (as numeric, factor or character).

\textit{start}  The base period (as character) limited to the year and month, e.g. "2020-03".

\textit{end}  The research period (as character) limited to the year and month, e.g. "2020-04".

\textit{wstart}  The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

\textit{window}  The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral Geary-Khamis price index which considers the time window defined by \textit{wstart} and \textit{window} parameters. The Geary-Khamis price index is calculated by using a special iterative algorithm from Chessa (2016). It measures the price dynamics by comparing period \textit{end} to period \textit{start} (both \textit{start} and \textit{end} must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References


Examples

\begin{verbatim}
gk(milk, start="2019-01", end="2019-08", window=10)
gk(milk, start="2018-12", end="2019-12")
\end{verbatim}

\underline{gk\_fbew}  \textit{Extending the multilateral Geary-Khamis price index by using the FBEW method.}

Description

This function returns a value of the multilateral Geary-Khamis price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.
Usage

```r
gk_fbew(data, start, end)
```

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral Geary-Khamis price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods `end` and `start`. The month of the `start` parameter must be December. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
gk_fbew(milk, start="2018-12", end="2019-08")
```

---

**gk_fbmw**

*Extending the multilateral Geary-Khamis price index by using the FBMW method.*

Description

This function returns a value of the multilateral Geary-Khamis price index extended by using the FBMW (Fixed Base Moving Window) method.
gk_fbmw(data, start, end)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral Geary-Khamis price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)-1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

gk_fbmw(milk, start="2019-12", end="2020-04")

gk_splice

Extending the multilateral Geary-Khamis price index by using window splicing methods.

Description

This function returns a value (or values) of the multilateral Geary-Khamis price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).
Usage
gk_splice(data, start, end, window = 13, splice = "movement", interval = FALSE)

Arguments
data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. "2019-12".
end The research period (as character) limited to the year and month, e.g. "2020-04".
window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
splice A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".
interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value
This function returns a value or values (depending on interval parameter) of the multilateral Geary-Khamis price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References
Examples

gk_splice(milk, start="2018-12", end="2020-02", splice="half")

 harmonic Calculating the unweighted harmonic price index

Description

This function returns a value (or vector of values) of the unweighted "unnamed" harmonic price index.

Usage

harmonic(data, start, end, interval = FALSE)

Arguments

data  The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start  The base period (as character) limited to the year and month, e.g. "2020-03".

end  The research period (as character) limited to the year and month, e.g. "2020-04".

interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the unweighted bilateral harmonic price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

harmonic(sugar, start="2018-12", end="2019-12")
harmonic(milk, start="2018-12", end="2020-01", interval=TRUE)

hybrid  Calculating the bilateral hybrid price index

Description

This function returns a value (or a vector of values) of the bilateral hybrid price index. The hybrid index was proposed by Bialek (2020) and it uses correlation coefficients between prices and quantities.

Usage

hybrid(data, start, end, base = start, interval = FALSE)

Arguments

data  The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start  The base period (as character) limited to the year and month, e.g. ’2020-03’.
end  The research period (as character) limited to the year and month, e.g. ’2020-04’.
base  The prior period used in the hybrid price index formula (as character) limited to the year and month, e.g. ’2020-01’.
interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or a vector of values) of the bilateral hybrid price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References

IQMp

Examples

hybrid(sugar, start="2019-12", end="2020-08", base="2018-12")
hybrid(milk, start="2019-12", end="2020-08", base="2018-12", interval=TRUE)

IQMp

Calculating the implicit quadratic mean of order r price index

Description

This function returns a value (or vector of values) of the implicit quadratic mean of order r price index.

Usage

IQMp(data, start, end, r = 2, interval = FALSE)

Arguments

data
   The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start
   The base period (as character) limited to the year and month, e.g. “2020-03”.

end
   The research period (as character) limited to the year and month, e.g. "2020-04".

r
   The real and non-zero parameter.

interval
   A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the implicit quadratic mean of order r price index - see CPI Manual (2004), Section 17.37, formula 17.32 (page 321).

References


Examples

IQMp(sugar, start="2019-01", end="2020-01")
IQMp(sugar, start="2019-01", end="2020-01", r=1.3, interval=TRUE)
Calculating the unweighted Jevons price index

Description

This function returns a value (or vector of values) of the unweighted bilateral Jevons price index.

Usage

jevons(data, start, end, interval = FALSE)

Arguments

data
The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also needed because this function uses unit values as monthly prices.

start
The base period (as character) limited to the year and month, e.g. "2020-03".

end
The research period (as character) limited to the year and month, e.g. "2020-04".

interval
A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start, end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the unweighted bilateral Jevons price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

jevons(milk, start="2018-12", end="2020-01")
jevons(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the bilateral Laspeyres price index

Description

This function returns a value (or vector of values) of the bilateral Laspeyres price index.

Usage

```r
laspeyres(data, start, end, interval = FALSE)
```

Arguments

- `data` The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- `start` The base period (as character) limited to the year and month, e.g. "2020-03".
- `end` The research period (as character) limited to the year and month, e.g. "2020-04".
- `interval` A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Laspeyres price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
laspeyres(sugar, start="2018-12", end="2019-12")
laspeyres(milk, start="2018-12", end="2020-01", interval=TRUE)
```
**Calculating the bilateral Lehr price index**

**Description**

This function returns a value (or vector of values) of the bilateral Lehr price index.

**Usage**

```r
lehr(data, start, end, interval = FALSE)
```

**Arguments**

- `data` The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- `start` The base period (as character) limited to the year and month, e.g. "2020-03".
- `end` The research period (as character) limited to the year and month, e.g. "2020-04".
- `interval` A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start,end>` are considered and `start` defines the base period (`interval` is set to TRUE).

**Value**

The function returns a value (or vector of values) of the bilateral Lehr price index depending on the `interval` parameter. If the `interval` parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**

Lehr, J. (1885). *Beitrage zur Statistik der Preise, insbesondere des Geldes und des Holzes*. J. D. Sauerlander, Frankfurt am Main.


**Examples**

```r
lehr(sugar, start="2018-12", end="2019-12")
lehr(milk, start="2018-12", end="2020-01", interval=TRUE)
```
Calculating the bilateral Lloyd-Moulton price index

Description

This function returns a value (or vector of values) of the bilateral Lloyd-Moulton price index.

Usage

lloyd_moulton(data, start, end, sigma = 0.7, interval = FALSE)

Arguments

data
The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start
The base period (as character) limited to the year and month, e.g. "2020-03".

end
The research period (as character) limited to the year and month, e.g. "2020-04".

sigma
The elasticity of substitution parameter (as numeric).

interval
A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Lloyd-Moulton price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


load_model

**Examples**

lloyd_moulton(sugar, start="2018-12", end="2019-12", sigma=0.9)
lloyd_moulton(milk, start="2018-12", end="2020-01", interval=TRUE)

---

**Description**

This function loads a list of machine learning model elements from the disk, i.e. the needed 8 files are read.

**Usage**

load_model(dir = "ML_model")

**Arguments**

- **dir**: The name of the directory from which the machine learning model is to be loaded. The directory must be in the working directory.

**Value**

This function loads a list of ML model elements from the disk, i.e. the needed 8 files are read from the directory selected by `dir`. After loading the model it can be used for product classification by using `data_classifying` function.

**Examples**

```r
# Setting a temporal directory as a working directory
## Not run: wd<-tempdir()
## Not run: setwd(wd)
# Building the model
my.grid=list(eta=c(0.01,0.02,0.05),subsample=c(0.5,0.8))
data_train<-dplyr::filter(dataCOICOP,dataCOICOP$time<=as.Date("2021-10-01"))
data_test<-dplyr::filter(dataCOICOP,dataCOICOP$time==as.Date("2021-11-01"))
ML<-model_classification(data_train,data_test,class="coicop6",grid=my.grid,
indicators=c("description","codeIN", "grammage"),key_words=c("uht"),rounds=60)
# Saving the model
## Not run: save_model(ML, dir="My_model")
# Loading the model
## Not run: ML_fromPC<-load_model("My_model")
# classes predicting
## Not run: data_classifying(ML_fromPC, data_test)
```

---

**load_model**

*Loading the machine learning model from the disk*

**Description**

This function loads a list of machine learning model elements from the disk, i.e. the needed 8 files are read.
Calculating the bilateral Lowe price index

Description
This function returns a value (or vector of values) of the bilateral Lowe price index.

Usage
lowe(data, start, end, base = start, interval = FALSE)

Arguments
- **data**: The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **base**: The prior period used in the Lowe price index formula (as character) limited to the year and month, e.g. "2020-01".
- **interval**: A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value
The function returns a value (or vector of values) of the bilateral Lowe price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References

Examples
lowe(sugar, start="2019-01", end="2020-01", base="2018-12")
lowe(milk, start="2018-12", end="2020-01", interval=TRUE)
Description

This function returns a value (or vector of values) of the bilateral Marshall-Edgeworth price index.

Usage

```
marshall_edgeworth(data, start, end, interval = FALSE)
```

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start,end>` are considered and `start` defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Marshall-Edgeworth price index depending on the `interval` parameter. If the `interval` parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


matched

Examples

marshall_edgeworth(sugar, start="2018-12", end="2019-12")
marshall_edgeworth(milk, start="2018-12", end="2020-01", interval=TRUE)

matched

Providing values from the indicated column that occur simultaneously
in the compared periods or in a given time interval.

Description

The function returns all values from the indicated column (defined by the type parameter) which occur simultaneously in the compared periods or in a given time interval.

Usage

matched(data, period1, period2, type = "prodID", interval = FALSE)

Arguments

data The user’s data frame. It must contain a column time (as Date in format: year-month-day, e.g. ’2020-12-01’) and also a column indicated by the type parameter.
period1 The first period (as character) limited to the year and month, e.g. ”2019-03”.
period2 The second period (as character) limited to the year and month, e.g. ”2019-04”.
type This parameter defines the column which is used in the procedure. Possible values of the type parameter are: retID, prodID, codeIN, codeOUT or description.
interval A logical parameter indicating whether the procedure is to work for the whole time period between period1 and period2 (then it is TRUE).

Value

The function returns all values from the indicated column (defined by the type parameter) which occur simultaneously in the compared periods or in a given time interval. Possible values of the type parameter are: retID, prodID, codeIN, codeOUT or description. If the interval parameter is set to FALSE, then the function compares only periods defined by period1 and period2. Otherwise the whole time period between period1 and period2 is considered.

Examples

matched(milk, period1="2018-12", period2="2019-12", interval=TRUE)
matched(milk, period1="2018-12", period2="2019-12", type="description")
**matched_fig**

*Providing a time dependent matched_index() function*

**Description**

The function provides a data frame or a figure presenting the `matched_index` function calculated for the column defined by the `type` parameter and for each month from the considered time interval.

**Usage**

```r
matched_fig(
  data,
  start,
  end,
  base = "start",
  type = "prodID",
  fixedbase = TRUE,
  figure = TRUE,
  date_breaks = "1 month"
)
```

**Arguments**

- **data**: The user's data frame. It must contain a column `time` (as Date in format: year-month-day, e.g. '2020-12-01') and also a column indicated by the `type` parameter.
- **start**: The beginning of a time interval (as character) limited to the year and month, e.g. "2019-03".
- **end**: The end of a time interval (as character) limited to the year and month, e.g. "2019-04".
- **base**: The base period (as character) for product comparisons. Its possible values are: "start" and "end".
- **type**: This parameter defines the column which is used in the procedure. Possible values of the `type` parameter are: `retID`, `prodID`, `codeIN`, `codeOUT` or `description`.
- **fixedbase**: A logical parameter indicating whether the procedure is to work for subsequent months from the considered time interval (`fixedbase=FALSE`). Otherwise the period defined by `base` plays a role of fixed base month (`fixedbase=TRUE`)
- **figure**: A logical parameter indicating whether the function returns a figure (TRUE) or a data frame (FALSE) with `matched_index` values.
- **date_breaks**: A string giving the distance between breaks on the X axis like "1 month" (default value) or "4 months".
**matched_index**

**Value**

The function returns a data frame or a figure presenting the `matched_index` function calculated for the column defined by the type parameter and for each month from the considered time interval. The interval is set by start and end parameters. The returned object (data frame or figure) depends on the value of `figure` parameter. The returned values belong to \([0,1]\).

**Examples**

```r
matched_fig(milk, start="2018-12", end="2019-12")
matched_fig(milk, start="2018-12", end="2019-12", figure=FALSE)
```

**Description**

The function returns a ratio of number of values from the indicated column that occur simultaneously in the compared periods or in a given time interval to the number of all available values from the above-mentioned column (defined by the type parameter) at the same time.

**Usage**

```r
matched_index(data, period1, period2, type = "prodID", interval = FALSE)
```

**Arguments**

- **data**
  The user's data frame. It must contain a column `time` (as Date in format: year-month-day, e.g. '2020-12-01') and also a column indicated by the type parameter.

- **period1**
  The first period (as character) limited to the year and month, e.g. "2019-03".

- **period2**
  The second period (as character) limited to the year and month, e.g. "2019-04".

- **type**
  This parameter defines the column which is used in the procedure. Possible values of the type parameter are: `retID`, `prodID`, `codeIN`, `codeOUT` or `description`.

- **interval**
  A logical parameter indicating whether the procedure is to work for the whole time period between `period1` and `period2` (then it is TRUE).

**Value**

The function returns a ratio of number of values from the indicated column that occur simultaneously in the compared periods or in a given time interval to the number of all available values from the above-mentioned column (defined by the type parameter) at the same time. Possible values of the type parameter are: `retID`, `prodID` or `description`. If the interval parameter is set to FALSE, then the function compares only periods defined by `period1` and `period2`. Otherwise the whole time period between `period1` and `period2` is considered. The returned value belongs to \([0,1]\).
Examples

```
matched_index(milk, period1="2018-12", period2="2019-12", interval=TRUE)
matched_index(milk, period1="2018-12", period2="2019-12", type="retID")
```

### Description

This function returns the multilateral Bennet price and quantity indicators and optionally also the price and quantity contributions of individual products.

### Usage

```
mbennet(
  data,
  start,
  end,
  wstart = start,
  matched = FALSE,
  window = 13,
  interval = FALSE,
  contributions = FALSE,
  prec = 2
)
```

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data</code></td>
<td>The user's data frame with information about sold products. It must contain columns: <code>time</code> (as Date in format: year-month-day, e.g. &quot;2020-12-01&quot;), <code>prices</code> (as positive numeric) and <code>prodID</code> (as numeric, factor or character). A column <code>quantities</code> (as positive numeric) is also needed because this function uses unit values as monthly prices.</td>
</tr>
<tr>
<td><code>start</code></td>
<td>The base period (as character) limited to the year and month, e.g. &quot;2020-03&quot;.</td>
</tr>
<tr>
<td><code>end</code></td>
<td>The research period (as character) limited to the year and month, e.g. &quot;2020-04&quot;.</td>
</tr>
<tr>
<td><code>wstart</code></td>
<td>The first period of the time window (as character) limited to the year and month, e.g. &quot;2019-12&quot;.</td>
</tr>
<tr>
<td><code>matched</code></td>
<td>A logical parameter indicating whether the matched sample approach is to be used (if yes, the parameter has the value TRUE).</td>
</tr>
<tr>
<td><code>window</code></td>
<td>The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).</td>
</tr>
<tr>
<td><code>interval</code></td>
<td>A logical parameter indicating whether calculations are to be made for the whole time interval (TRUE) or no (FALSE).</td>
</tr>
</tbody>
</table>
contributions  A logical parameter indicating whether contributions of individual products are to be displayed. If it is TRUE, then contributions are calculated for the the base period start and the current period end.

prec        A numeric vector indicating precision, i.e. the number of decimal places for presenting results.

Value

This function returns the multilateral Bennet price and quantity indicators and optionally also the price and quantity contributions of individual products.

References


Examples

mbennet(milk, "2018-12", "2019-12", matched=TRUE, contributions=TRUE)
mbennet(coffee, start="2018-12", end="2019-03", interval=TRUE)

milk        A real data set on sold milk

Description

A collection of scanner data on the sale of milk in one of Polish supermarkets in the period from December 2018 to August 2020

Usage

milk

Format

A data frame with 6 columns and 4386 rows. The used variables are as follows:

time       - Dates of transactions (Year-Month-Day)
prices     - Prices of sold products [PLN]
quantities - Quantities of sold products [liters]
prodID     - Unique product codes (data set contains 68 different prodIDs)
retID      - Unique codes identifying outlets/retailer sale points (data set contains 5 different retIDs)
description Descriptions of sold milk products (data set contains 6 different product descriptions)
Calculating the multilateral Montgomery price and quantity indicators

Description

This function returns the multilateral Montgomery price and quantity indicators and optionally also the price and quantity contributions of individual products.

Usage

```
mmontgomery(
  data,
  start,
  end,
  wstart = start,
  matched = FALSE,
  window = 13,
  interval = FALSE,
  contributions = FALSE,
  prec = 2
)
```

Arguments

data: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. ’2020-12-01’), `prices` (as positive numeric) and `prodID` (as numeric, factor or character). A column `quantities` (as positive numeric) is also needed because this function uses unit values as monthly prices.

start: The base period (as character) limited to the year and month, e.g. ”2020-03”.

data: The research period (as character) limited to the year and month, e.g. ”2020-04”.

wstart: The first period of the time window (as character) limited to the year and month, e.g. ”2019-12”.

matched: A logical parameter indicating whether the matched sample approach is to be used (if yes, the parameter has the value TRUE).

window: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

interval: A logical parameter indicating whether calculations are to be made for the whole time interval (TRUE) or no (FALSE).

contributions: A logical parameter indicating whether contributions of individual products are to be displayed. If it is TRUE, then contributions are calculated for the base period `start` and the current period `end`.

prec: A numeric vector indicating precision, i.e. the number of decimal places for presenting results.
Value

This function returns the multilateral Montgomery price and quantity indicators and optionally also the price and quantity contributions of individual products.

References


Examples

```r
mmontgomery(milk, "2018-12", "2019-12", matched=TRUE, contributions=TRUE)
mmontgomery(coffee, start="2018-12", end="2019-03", interval=TRUE)
```

Description

This function provides a trained machine learning model to classify products into classes or any other groups defined by the user. In addition, the function returns the characteristics of the model and figures describing the learning process.

Usage

```r
model_classification(
  data_train = data.frame(),
  data_test = data.frame(),
  class = c(),
  indicators = c(),
  key_words = c(),
  sensitivity = FALSE,
  p = 0.9,
  w = 0.2,
  rounds = 200,
  grid = list()
)
```
Arguments

- **data_train**: Training data set for the model. This set must contain all the columns defined by the `indicators` parameter and the `class` column. If the `key_words` vector is non-empty, the set should also contain a `description` column. Ideally, the indicators should be of the numerical type. If the indicator is not of the numerical type, it will be converted to this type.

- **data_test**: A test set that is used to validate the machine learning model. This set should have the same structure as the training set, but it is not obligatory. If the test set is not specified by the user then the test set is drawn from the training set (see `p` parameter).

- **class**: A character string which indicates the column with classes (groups) of products (e.g. COICOPs).

- **indicators**: A vector of column names to be considered in building a machine learning model. Important: the indicated variables can be numeric but also categorical (factor or character types are acceptable).

- **key_words**: A vector of keywords or phrases that will be recognized in the `description` column. For each such keyword and or phrase, a new binary variable (column) will be created and included in the machine model training process.

- **sensitivity**: A logical parameter that indicates whether lowercase or uppercase letters are to be distinguished when the `key_words` vector is not empty.

- **p**: A parameter related to creating the testing set, if it has not been specified by the user. The test set is then created on the basis of a class-balanced subsample of the training set. The size of this subsample is 100p percents of the training set size.

- **w**: A parameter for determining the measure of choosing the optimal machine learning model. For each combination of parameters specified in the `grid` list, the error rate of the trained model is calculated on the basis of the error on the training set (error_L=1-accuracy_L) and the error on the testing set (error_T=1-accuracy_T). Final accuracy of the model is estimated as: w accuracy_L + (1-w) accuracy_T.

- **rounds**: The maximum number of iterations during the training stage.

- **grid**: The list of vectors of parameters which are taken into consideration during the Extreme Gradient Boosting training. The default value of this list is as follows: grid=list(eta=c(0.05,0.1,0.2),max_depth=c(6),min_child_weight=c(1),max_delta_step=c(0),subsample=c(1),gamma=c(0),lambda=c(1),alpha=c(0)). The complete list of parameters for the used Tree Booster is available online here.

Value

In general, this function provides a trained machine learning model to classify products into classes (or any other groups). In addition, the function returns the characteristics of the model and figures describing the learning process. The machine learning process is based on the XGBoost algorithm (from the XGBoost package) which is an implementation of gradient boosted decision trees designed for speed and performance. The function takes into account each combination of model parameters (specified by the `grid` list) and provides, inter alia, an optimally trained model (a model that minimizes the error rate calculated on the basis of a fixed value of the `w` parameter). After all, the function
returns a list of the following objects: model - the optimally trained model; best_parameters - a set of parameters of the optimal model; indicators - a vector of all indicators used; key_words - a vector of all key words and phrases used; classes - a dataframe with categorized classes; sensitivity - a value of the used 'sensitivity' parameter; figure_training - a plot of the error levels calculated for the training set and the testing set during the learning process of the returned model (error = 1 - accuracy); figure_importance - a plot of the relative importance of the used indicators.

References

Tianqi Chen and Carlos Guestrin (2016). *XGBoost: A Scalable Tree Boosting System*. 22nd SIGKDD Conference on Knowledge Discovery and Data Mining.

Examples

```r
my.grid=list(eta=c(0.01,0.02,0.05),subsample=c(0.5,0.8))
data_train<dplyr::filter(dataCOICOP,dataCOICOP$Time<=as.Date("2021-10-01"))
data_test<dplyr::filter(dataCOICOP,dataCOICOP$Time==as.Date("2021-11-01"))
ML<model_classification(data_train,data_test,class="coicop6",grid=my.grid,
indicators=c("description","codeIN","grammage"),key_words=c("uht"),rounds=60)
ML$best_parameters
ML$indicators
ML$figure_training
ML$figure_importance
```

**montgomery**

*Calculating the Montgomery price and quantity indicators*

**Description**

This function returns the Montgomery price and quantity indicators and optionally also the price and quantity contributions of individual products.

**Usage**

```r
montgomery(
  data,
  start,
  end,
  interval = FALSE,
  matched = FALSE,
  contributions = FALSE,
  prec = 2
)
```
Arguments

- **data**: The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric) and `prodID` (as numeric, factor or character). A column `quantities` (as positive numeric) is also needed because this function uses unit values as monthly prices.
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical parameter indicating whether calculations are to be made for the whole time interval (TRUE) or no (FALSE).
- **matched**: A logical parameter indicating whether the matched sample approach is to be used (if yes, the parameter has the value TRUE).
- **contributions**: A logical parameter indicating whether contributions of individual products are to be displayed. If it is TRUE, then contributions are calculated for the base period `start` and the current period `end`.
- **prec**: A numeric vector indicating precision, i.e. the number of decimal places for presenting results.

Value

This function returns the Montgomery price and quantity indicators and optionally also the price and quantity contributions of individual products.

References


Examples

```r
montgomery(milk, "2018-12", "2019-12", matched=TRUE, contributions=TRUE)
montgomery(coffee, start="2018-12", end="2019-03", interval=TRUE)
```

---

**paasche**

*Calculating the bilateral Paasche price index*

Description

This function returns a value (or vector of values) of the bilateral Paasche price index.

Usage

```r
paasche(data, start, end, interval = FALSE)
```
Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Paasche price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

paasche(sugar, start="2018-12", end="2019-12")
paasche(milk, start="2018-12", end="2020-01", interval=TRUE)

palgrave Calculating the bilateral Palgrave price index

Description

This function returns a value (or vector of values) of the bilateral Palgrave price index.

Usage

palgrave(data, start, end, interval = FALSE)
Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. ’2020-12-01’), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

- **start**: The base period (as character) limited to the year and month, e.g. ”2020-03”.

- **end**: The research period (as character) limited to the year and month, e.g. ”2020-04”.

- **interval**: A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start,end>` are considered and `start` defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Palgrave price index depending on the `interval` parameter. If the `interval` parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
palgrave(sugar, start="2018-12", end="2019-12")
palgrave(milk, start="2018-12", end="2020-01", interval=TRUE)
```

--

**pqcor**

- Providing a correlation coefficient for price and quantity of sold products

Description

The function returns correlation between price and quantity of sold products with given IDs.

Usage

```r
pqcor(data, period, set = c(), figure = FALSE)
```
pqcor_fig

Arguments

data The user’s data frame. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character) with unique product IDs.

period The time period (as character) limited to the year and month, e.g. ”2019-03”.

set The set of unique product IDs to be used for determining correlation between price and quantity of sold products (see also data_matching). If the set is empty, the function works for all products being available in period.

figure A logical parameter indicating whether the function returns a figure (TRUE) or a data frame (FALSE) with correlations between price and quantity of sold products.

Value

The function returns Pearson’s correlation coefficient between price and quantity of products with given IDs and sold in period.

Examples

pqcor(milk, period="2019-03")
pqcor(milk, period="2019-03",figure=TRUE)

Description

The function returns Pearson’s correlation coefficients between price and quantity of sold products with given IDs.

Usage

pqcor_fig(data, start, end, figure = TRUE, date_breaks = "1 month", set = c())

Arguments

data The user’s data frame. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character) with unique product IDs.

start The beginning of the considered time interval (as character) limited to the year and month, e.g. ”2020-03”.

end The end of the considered time interval (as character) limited to the year and month, e.g. ”2020-04”.
figure A logical parameter indicating whether the function returns a figure (TRUE) or a data frame (FALSE) with price-quantity correlations.
date_breaks A string giving the distance between breaks on the X axis like "1 month" (default value) or "4 months".
set The set of unique product IDs to be used for determining correlation between prices and quantities of sold products (see also data_matching). If the set is empty, the function works for all products being available in period.

Value

The function returns Pearson’s correlation coefficients between price and quantity of products with given IDs and sold in the time interval: <start, end>. Correlation coefficients are calculated for each month separately. Results are presented in tabular or graphical form depending on the figure parameter.

Examples

pqcor_fig(milk, start="2018-12", end="2019-12", figure=FALSE)
pqcor_fig(milk, start="2018-12", end="2019-12", figure=TRUE)

PriceIndices 

The list of package functions and their demonstration

Description

The **PriceIndices** package is a tool for Bilateral and Multilateral Price Index Calculations. A demonstration of package functions is here: README. The package documentation can be found HERE. The list of package functions is as follows:

Data sets in the package and generating artificial scanner data sets

dataAGGR
dataMATCH
dataCOICOP
data_DOWN_UP_SIZED
milk
sugar
coffee
dataU
generate
tindex
Functions for data processing

- data_check
- data_preparing
- data_aggregating
- data_reducing
- data_unit
- data_norm
- data_selecting
- data_classifying
- model_classification
- save_model
- load_model
- data_matching
- data_filtering
- shrinkflation

Functions providing dataset characteristics

- available
- matched
- matched_index
- matched_fig
- prices
- quantities
- sales
- sales_groups
- sales_groups2
- pqcor
- pqcor_fig
- dissimilarity_fig
- elasticity
- elasticity_fig

Functions for bilateral unweighted price index calculation

- bmw
- carli
- cswd
- dutot
Functions for bilateral weighted price index calculation

agmean
banajree
bialek
davies
drobisch
fisher
geary_khamis
geolaspeyres
geolowe
gopaasche
goegeoun
geohybrid
hybrid
laspeyres
lehr
lloyd_moulton
lowe
marshall_edgeworth
paasche
palgrave
sato_vartia
stuvel
tornqvist
vartia
walsh
young

Functions for chain price index calculation

chbmv
chcarli
chcswd
chdutot
Functions for multilateral price index calculation

ccdi
geks
wgeks
geksl
wgeksl
geksgl
Functions for extending multilateral price indices by using splicing methods

ccdi_splice
geks_splice
wgeks_splice
geksj_splice
geksw_splice
geksl_splice
wgeksl_splice
geksgl_splice
wgeksgl_splice
geksaqu_splice
wgeksaqu_splice
geksgaqi_splice
wgeksgaqi_splice
gk_splice
tpd_splice
Functions for extending multilateral price indices by using the FBEW method

ccdi_fbew
geks_fbew
wgeks_fbew
geksj_fbew
geksw_fbew
geksl_fbew
wgeksl_fbew
geksgl_fbew
wgeksgl_fbew
geksaqu_fbew
wgeksaqu_fbew
geksaqi_fbew
wgeksaqi_fbew
geksgaqi_fbew
wgeksgaqi_fbew
gk_fbew
tpd_fbew

Functions for extending multilateral price indices by using the FBMW method

ccdi_fbmw
geks_fbmw
wgeks_fbmw
geksj_fbmw
geksw_fbmw
geksl_fbmw
wgeksl_fbmw
geksgl_fbmw
wgeksgl_fbmw
geksaqu_fbmw
wgeksaqu_fbmw
geksaqi_fbmw
wgeksaqi_fbmw
geksgaqi_fbmw
wgeksgaqi_fbmw
gk_fbmw
tpd_fbmw
General functions for price index calculations

- `price_indices`
- `final_index`

Functions for comparisons of price indices

- `compare_indices_df`
- `compare_indices_list`
- `compare_indices_jk`
- `compare_distances`
- `compare_to_target`

---

**prices**

*Providing prices (unit values) of sold products*

---

**Description**

The function returns prices (unit values) of sold products with given IDs.

**Usage**

```
prices(data, period, set = c(), ID = FALSE)
```

**Arguments**

- **data**
  The user’s data frame. It must contain columns: `time` (as Date in format: year-month-day, e.g. ‘2020-12-01’), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character) with unique product IDs.

- **period**
  The time period (as character) limited to the year and month, e.g. “2019-03”.

- **set**
  The set of unique product IDs to be used for determining prices of sold products (see also `data_matching`). If the set is empty, the function returns prices of all products being available in period.

- **ID**
  A logical parameter indicating whether a data frame with prodIDs and prices (unit values) should be returned.

**Value**

The function analyzes the user’s data frame and returns prices (unit value) of products with given ID and being sold in the time period indicated by the `period` parameter. Please note, that the function returns the price values for sorted prodIDs and in the absence of a given prodID in the data set, the function returns nothing (it does not return zero). If the ID parameter is set to TRUE then the function returns a data frame with columns: `by` (IDs of products) and `uv` (unit values of products).
price_indices

Examples

```r
prices(milk, period="2019-06")
prices(milk, period="2019-12", set=c(400032, 82919), ID=TRUE)
```

price_indices

A general function to compute one or more price indices

Description

This function returns a value or values of the selected price indices.

Usage

```r
price_indices(
  data,
  start,
  end,
  formula = c(),
  window = c(),
  splice = c(),
  base = c(),
  sigma = c(),
  r = c(),
  interval = FALSE,
  names = c()
)
```

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric) and prodID (as numeric, factor or character). A column quantities (as positive numeric) is also essential even if the selected index is an unweighted formula (unit values are calculated).
- **start**: The base period (as character) limited to the year and month, e.g. ”2019-12”.
- **end**: The research period (as character) limited to the year and month, e.g. ”2020-04”.
- **formula**: A vector of character strings indicating price index formulas that are to be calculated. To see available options please use the link: PriceIndices.
- **window**: A vector of integers. Each element of the vector defines the length of the time window of the corresponding multilateral index.
- **splice**: A vector of character strings. Each element of the vector indicates the splicing method is to be used for the corresponding multilateral index. Available values of vector elements are: ”movement”, ”window”, ”half”, ”mean” and their additional variants: ”window_published”, ”half_published” and ”mean_published”. 
products

| base | The vector of prior periods used in the Young- or Lowe-type price indices or hybrid/geohybrid index. Each element of the vector (as character) must be limited to the year and month, e.g. "2020-01". |
| sigma | The vector of elasticity of substitution parameters used in the Lloyd-Moulton, AG Mean or GEKS-LM indices (as numeric). |
| r | The vector of non-zero parameters used in the quadratic mean of order r quantity / price index or in the GEKS-QM index (as numeric). |
| interval | A logical value indicating whether the function is to provide price indices comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be presented (the fixed base month is defined by start). |
| names | A vector of strings indicating names of indices which are to be used in the resulting data frame. |

Value

This general function returns a value or values of the selected price indices. If the interval parameter is set to TRUE, then it returns a data frame where its first column indicates dates and the remaining columns show corresponding values of all selected price indices. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

Examples

```r
price_indices(milk, start="2018-12",end="2019-12", formula=c("geks","ccdi","hybrid","fisher", "QMp","young","geks1_fbew"), window=c(13,13), base=c("2019-03","2019-03"), r=c(3), interval=TRUE)
price_indices(milk, start="2018-12",end="2019-12", formula=c("geks","ccdi","hybrid","fisher", "QMp","young","geks1_fbew"), window=c(13,13), base=c("2019-03","2019-03"), r=c(3), interval=FALSE)
```

products

Detecting and summarising available, matched, new and disappearing products.

Description

This function detects and summarises available, matched, new as well as disappearing products on the basis of their prodIDs.
Usage

\texttt{products(data, start, end)}

Arguments

data \hspace{1cm} \text{The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01') and prodID (as numeric, factor or character).}

start \hspace{1cm} \text{The base period (as character) limited to the year and month, e.g. "2020-03".}

date \hspace{1cm} \text{The research period (as character) limited to the year and month, e.g. "2020-04".}

Value

This function detects and summarises available, matched, new and disappearing products on the basis of their prodIDs. It compares products from the base period (start) with products from the current period (end). It returns a list containing the following objects: details with prodIDs of available, matched, new and disappearing products, statistics with basic statistics for them and figure with a pie chart describing a contribution of matched, new and disappearing products in a set of available products.

Examples

\texttt{list<-products(milk, "2018-12","2019-12")}
\texttt{list$details}
\texttt{list$statistics}
\texttt{list$figure}

\textbf{products\_fig} \hspace{1cm} \textit{Function for graphical comparison of available, matched, new as well as disappearing products.}

Description

This function returns a figure with plots of volume (or contributions) of available, matched, new as well as disappearing products.

Usage

\texttt{products\_fig(}
\hspace{1cm} \texttt{data,}
\hspace{1cm} \texttt{start,}
\hspace{1cm} \texttt{end,}
\hspace{1cm} \texttt{show = c(\textquoteleft available\textquoteleft, \textquoteleft matched\textquoteleft, \textquoteleft new\textquoteleft, \textquoteleft disappearing\textquoteleft),}
\hspace{1cm} \texttt{fixed\_base = TRUE,}
\hspace{1cm} \texttt{contributions = TRUE,}
\hspace{1cm} \texttt{date\_breaks = "1 month"}
\hspace{1cm} )}
Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ‘2020-12-01’) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

show A character vector indicating which groups of products are to be taken into consideration. Available options are available, matched, new and disappearing.

fixed_base A logical parameter indicating whether each month is to be compared to the base period (TRUE) or to the previous month (then it is set to FALSE).

contributions A logical parameter indicating whether contributions or volumes counted for available, matched, new and disappearing products are to be displayed.

date_breaks A string giving the distance between breaks on the X axis like "1 month" (default value) or "4 months".

Value

This function returns a figure with plots of volume (or contributions) of available, matched, new as well as disappearing products. The User may control which groups of products are to be taken into consideration (see the show parameter). Available options are available, matched, new and disappearing.

Examples

```r
products_fig(milk, "2018-12","2019-04",
fixed_base=TRUE, contributions=FALSE,
show=c("new","disappearing","matched","available"))
```

QMp Calculating the quadratic mean of order r price index

Description

This function returns a value (or vector of values) of the quadratic mean of order r price index.

Usage

```r
QMp(data, start, end, r = 2, interval = FALSE)
```

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ‘2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
**QMq**

Calculating the quadratic mean of order r quantity index

**Description**

This function returns a value (or vector of values) of the quadratic mean of order r quantity index.

**Usage**

QMq(data, start, end, r = 2, interval = FALSE)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).</td>
</tr>
<tr>
<td>start</td>
<td>The base period (as character) limited to the year and month, e.g. &quot;2020-03&quot;.</td>
</tr>
<tr>
<td>end</td>
<td>The research period (as character) limited to the year and month, e.g. &quot;2020-04&quot;.</td>
</tr>
<tr>
<td>r</td>
<td>The real and non-zero parameter.</td>
</tr>
</tbody>
</table>

**Value**

The function returns a value (or vector of values) of the quadratic mean of order r price index - see CPI Manual (2004), Section 17.40, formula 17.35 (page 321).

**References**


**Examples**

QMq(sugar, start="2019-01", end="2020-01")
QMq(sugar, start="2019-01", end="2020-01", r=1.3, interval=TRUE)
interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the quadratic mean of order r quantity index - see CPI Manual (2004), Section 17.35, formula 17.30 (page 321).

References


Examples

QMq(sugar, start="2019-01", end="2020-01")
QMq(sugar, start="2019-01", end="2020-01", r=1.3, interval=TRUE)

QU Calculating the quality adjusted unit value index (QU index)

Description

This function returns a value of the quality adjusted unit value index (QU index) for a given set of adjustment factors.

Usage

QU(data, start, end, v)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

v The data frame with adjustment factors for at least all matched prodIDs. It must contain two columns: prodID (as numeric or character) with unique product IDs and values (as positive numeric) with corresponding adjustment factors.
quantities

Value

This function returns a value of the quality adjusted unit value index (QU index) for a given set of adjustment factors (adjusted factors must be available for all matched prodIDs).

References


Examples

```r
## Creating a data frame with artificial adjustment factors
## (random numbers from uniform distribution U[1,2])
prodID<-unique(milk$prodID)
values<-stats::runif(length(prodID),1,2)
v<-data.frame(prodID,values)
## Calculating the QU index for the created data frame 'v'
QU(milk, start="2018-12", end="2019-12", v)
```

quantities

Providing quantities of sold products

Description

The function returns quantities of sold products with given IDs.

Usage

```r
quantities(data, period, set = c(), ID = FALSE)
```

Arguments

- **data**
  - The user's data frame. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), quantities (as positive numeric) and prodID (as numeric, factor or character) with unique product IDs.

- **period**
  - The time period (as character) limited to the year and month, e.g. "2019-03".

- **set**
  - The set of unique product IDs to be used for determining quantities of sold products (see also `data_matching`). If the set is empty, the function returns quantities of all products being available in period.

- **ID**
  - A logical parameter indicating whether a data frame with prodIDs and quantities should be returned.

Value

The function analyzes the user's data frame and returns quantities of products with given ID and being sold in the time period indicated by the `period` parameter. Please note that the function returns the quantity values for sorted prodIDs and in the absence of a given prodID in the data set, the function returns nothing (it does not return zero). If the ID parameter is set to TRUE then the function returns a data frame with columns: by (IDs of products) and q (quantities of products).
sales

Providing values of product sales

Description

The function returns values of sales of products with given IDs.

Usage

sales(data, period, set = c(), shares = FALSE, hist = FALSE)

Arguments

data The user's data frame. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character) with unique product IDs.

period The time period (as character) limited to the year and month, e.g. "2019-03".

set The set of unique product IDs to be used for determining product sales values (see also data_matching). If the set is empty, then the function returns sale values of all products being available in period.

shares A logical parameter indicating whether the function is to return shares of product sales.

hist A logical parameter indicating whether the function is to return histogram of product sales.

Value

The function analyzes the user's data frame and returns values of sales of products with given IDs and being sold in time period indicated by the period parameter (see also expenditures function which returns the expenditure values for sorted prodIDs).

Examples

sales(milk, period="2019-06", shares=TRUE, hist=TRUE)
sales(milk, period="2019-12", set=unique(milk$prodID)[1])
sales_groups

**Providing information about sales of products from one or more datasets**

**Description**

The function returns values of sales of products from one or more datasets or the corresponding barplot for these sales.

**Usage**

```r
sales_groups(
    datasets = list(),
    start,
    end,
    shares = FALSE,
    barplot = FALSE,
    names = c()
)
```

**Arguments**

- `datasets` A list of user’s data frames. Each data frame must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric) and `quantities` (as positive numeric).
- `start` The beginning of the considered time interval (as character) limited to the year and month, e.g. "2020-03".
- `end` The end of the considered time interval (as character) limited to the year and month, e.g. "2020-04".
- `shares` A logical parameter indicating whether the function is to calculate shares of product sales
- `barplot` A logical parameter indicating whether the function is to return barplot for product sales.
- `names` A vector of characters describing product groups defined by `datasets`.

**Value**

The function returns values of sales of products from one or more datasets or the corresponding barplot for these sales (if `barplot` is TRUE). Alternatively, it calculates the sale shares (if `shares` is TRUE).

**Examples**

```r
## Creating 3 subgroups of milk:
ctg<-unique(milk$description)
categories<-c(ctg[1],ctg[2],ctg[3])
```
sales_groups2

Providing information about sales of products

Description

The function returns values of sales of products or the corresponding barplot for these sales.

Usage

```r
sales_groups2(
  data = data.frame(),
  by, 
  start, 
  end, 
  shares = FALSE, 
  barplot = FALSE, 
  names = c()
)
```

Arguments

- **data**: The user’s data frame with subgroups of sold products (see by parameter). The data frame must contain columns: `time` (as Date in format: year-month-day, e.g. "2020-12-01"), `prices` (as positive numeric) and `quantities` (as positive numeric). An additional column indicated via by parameter is also needed.
- **by**: The column name indicating grouping variable, i.e. this column is used for creating subgroups of products.
- **start**: The beginning of the considered time interval (as character) limited to the year and month, e.g. "2020-03".
- **end**: The end of the considered time interval (as character) limited to the year and month, e.g. "2020-04".
- **shares**: A logical parameter indicating whether the function is to calculate shares of product sales.
- **barplot**: A logical parameter indicating whether the function is to return barplot for product sales.
- **names**: A vector of characters describing product groups defined by datasets.
Value

The function returns values of sales of products or the corresponding barplot for these sales (if barplot is TRUE). Alternatively, it calculates the sale shares (if shares is TRUE).

Examples

```r
outlets<-as.character(unique(milk$retID))
sales_groups2(milk,by="retID",start="2019-04",end="2019-04", shares=TRUE,barplot=TRUE,names=outlets)
```

---

**sato_vartia**  
*C Calculating the bilateral Vartia-II (Sato-Vartia) price index*

Description

This function returns a value (or vector of values) of the bilateral Vartia-II (Sato-Vartia) price index.

Usage

```
sato_vartia(data, start, end, interval = FALSE)
```

Arguments

- `data`  
The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- `start`  
The base period (as character) limited to the year and month, e.g. "2020-03".
- `end`  
The research period (as character) limited to the year and month, e.g. "2020-04".
- `interval`  
A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Vartia-II (Sato-Vartia) price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

sato_vartia(sugar, start="2018-12", end="2019-12")
sato_vartia(milk, start="2018-12", end="2020-01", interval=TRUE)

Description

This function saves a list of machine learning model elements on the disk, i.e. the resulting 8 files are written.

Usage

save_model(model = list(), dir = "ML_model")

Arguments

model A list of 8 elements which identify the previously built machine learning model (the list is obtained via the model_classification function).

dir The name of the directory where the selected model should be saved. The directory with all necessary files will be created in the working directory.

Value

This function saves a list of ML model elements on the disk, i.e. the resulting 8 files are written into the new directory specified by dir. The list should be obtained previously using the model_classification function. After saving the model, it can be loaded at any time by using the load_model function.
Examples

# Setting a temporal directory as a working director
## Not run: wd<-tempdir()
## Not run: setwd(wd)

# Building the model

# Building the model
my.grid=list(eta=c(0.01,0.02,0.05),subsample=c(0.5,0.8))
data_train<dplyr::filter(dataCOICOP,dataCOICOP$time<=as.Date("2021-10-01"))
data_test<dplyr::filter(dataCOICOP,dataCOICOP$time==as.Date("2021-11-01"))
ML<-model_classification(data_train,data_test,class="coicop6",grid=my.grid,
indicators=c("description","codeIN", "grammage"),key_words=c("uht"),rounds=60)

# Saving the model
## Not run: save_model(ML, dir="My_model")

---

shrinkflation  

Detecting and summarising downsized and upsized products.

Description

This function detects and summarises downsized and upsized products.

Usage

```r
shrinkflation(
data,
start,
end,
type = "shrinkflation",
min_p_change = 0,
max_p_change = Inf,
min_s_change = 0,
max_s_change = Inf,
prec = 3,
interval = FALSE
)
```

Arguments

- **data**  The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ‘2020-12-01’) and prodID (as numeric, factor or character), prices (with standardised prices!) and quantities (as numeric), grammage (as numeric), unit (as character) and description (as character). Important: prices must be standardized beforehand, that is, they must refer to the sales unit (the data_norm function can be used for this).
- **start**  The base period (as character) limited to the year and month, e.g. "2024-01".
- **end**    The research period (as character) limited to the year and month, e.g. "2024-02".
### type
A parameter specifying what phenomenon is to be included in the resulting elements of the returned list (i.e., in returned products_detected, df_detected and df_reduced). The available values are: shrinkflation, shrinkdeflation, sharkflation, unshrinkdeflation, unshrinkflation and sharkdeflation (default value is: shrinkflation).

### min_p_change
Lower limit for unit price change, i.e.: a product is considered if the percentage change in its unit price is greater than the value of this parameter. The default value is zero, possibly positive values can be considered (in percentage).

### max_p_change
Upper limit for unit price change, i.e.: a product is considered if the percentage change in its unit price is less than the value of this parameter. The default value is Inf, possibly positive values can be considered (in percentage).

### min_s_change
Lower limit for size change, i.e.: a product is considered if the percentage change in its size is greater than the value of this parameter. The default value is zero, possibly positive values can be considered (in percentage).

### max_s_change
Upper limit for size change, i.e.: a product is considered if the percentage change in its size is less than the value of this parameter. The default value is Inf, possibly positive values can be considered (in percentage).

### prec
Number of decimal places for the presented summary results.

### interval
A parameter that specifies whether the search for downsized products should consider the entire time interval, or only the compared months specified by the start and end parameters.

### Value
This function detects and summarises downsized and upsized products. The function detects phenomena such as: shrinkflation, shrinkdeflation, sharkflation, unshrinkdeflation, unshrinkflation, sharkdeflation (see the type parameter). It returns a list containing the following objects:

- **df_changes** - data frame with detailed information on downsized and upsized products with the whole history of size changes,
- **df_type** - data frame with recognized type of products,
- **df_overview** - a table with basic summary of all detected products grouped by the type parameter,
- **products_detected** with prodIDs of products indicated by the 'type' parameter,
- **df_detected** being a subset of the data frame with only detected products,
- **df_reduced** which is the difference of the input data frame and the data frame containing the detected products, and
- **df_summary** which provides basic statistics for all detected downsized and upsized products (including their share in the total number of products and mean price and size changes).

### References

### Examples
```r
# Data matching over time
df <- data_matching(data=data_DOWN_UP_SIZED, start="2024-01", end="2024-02", codeIN=TRUE, codeOUT=TRUE, description=TRUE, onlydescription=FALSE, precision=0.9, interval=FALSE)
# Extraction of information about grammage (if needed)
```
df<-data_unit(df,units=c("g|ml|kg|l"),multiplication="x")
# Price standardization
df<-data_norm(df, rules=list(c("ml","l",1000),c("g","kg",1000)))
# Downsized and upsized products detection
result<-shrinkflation(data=df, start="2024-01","2024-02",
prec=3, interval=FALSE, type="shrinkflation")
result$df_changes
result$df_type
result$df_overview
result$products_detected
result$df_detected
result$df_reduced
result$df_summary

SPQ  Calculating the multilateral SPQ price index

Description
This function returns a value of the multilateral SPQ price index which is based on the relative price and quantity dissimilarity measure.

Usage
SPQ(data, start, end, interval = FALSE)

Arguments
data  The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start  The base period (as character) limited to the year and month, e.g. '2019-03'.
end    The research period (as character) limited to the year and month, e.g. '2019-07'.
interval  A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start, end> are considered and start defines the base period (interval is set to TRUE).

Value
This function returns a value of the multilateral SPQ price index which is based on the relative price and quantity dissimilarity measure (see References). If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

SPQ(sugar, start="2018-12", end="2019-02")
SPQ(milk, start="2018-12", end="2019-12", interval=TRUE)

stuvel

Calculating the bilateral Stuvel price index

Description

This function returns a value (or vector of values) of the bilateral Stuvel price index.

Usage

stuvel(data, start, end, interval = FALSE)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. ”2020-03”.

end The research period (as character) limited to the year and month, e.g. ”2020-04”.

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Stuvel price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

```r
stuvel(sugar, start="2018-12", end="2019-12")
stuvel(milk, start="2018-12", end="2020-01", interval=TRUE)
```

---

**sugar**

A real data set on sold sugar

---

Description

A collection of scanner data on the sale of sugar in one of Polish supermarkets in the period from December 2017 to October 2020

Usage

sugar

Format

A data frame with 6 columns and 7666 rows. The used variables are as follows:

- **time** - Dates of transactions (Year-Month-Day)
- **prices** - Prices of sold products [PLN]
- **quantities** - Quantities of sold products [kg]
- **prodID** - Unique product codes (data set contains 11 different prodIDs)
- **retID** - Unique codes identifying outlets/retailer sale points (data set contains 20 different retIDs)
- **description** - Descriptions of sold sugar products (data set contains 3 different product descriptions)
### Description

This function calculates the theoretical value of the unweighted price index for lognormally distributed prices.

### Usage

```r
tindex(pmi = c(), psigma = c(), start, ratio = TRUE)
```

### Arguments

- **pmi**
  - A numeric vector indicating \( m_i \) parameters for lognormally distributed prices from the subsequent months.

- **psigma**
  - A numeric vector indicating \( \sigma \) parameters for lognormally distributed prices from the subsequent months.

- **start**
  - The first period in the generated data frame (as character) limited to the year and month, e.g. '2019-12'.

- **ratio**
  - A logical parameter indicating how we define the theoretical unweighted price index. If it is set to TRUE, then the resulting value is a ratio of expected price values from compared months; otherwise the resulting value is the expected value of the ratio of prices from compared months.

### Value

This function calculates the theoretical value of the unweighted price index for lognormally distributed prices (the month defined by `start` parameter plays a role of the fixed base period). The characteristics for these lognormal distributions are set by `pmi` and `psigma` parameters. The `ratio` parameter allows to control the definition of resulting theoretical price index values. The function provides a data frame consisting of dates and corresponding expected values of the theoretical unweighted price index. The generated dataset is ready for further price index calculations.

### Examples

```r
tindex(pmi=c(1,1.2,1.3),psigma=c(0.1,0.2,0.15),start="2020-01")
tindex(pmi=c(1,1.2,1.3),psigma=c(0.1,0.2,0.15),start="2020-01",ratio=FALSE)
```


**Calculating the bilateral Tornqvist price index**

**Description**

This function returns a value (or vector of values) of the bilateral Tornqvist price index.

**Usage**

tornqvist(data, start, end, interval = FALSE)

**Arguments**

- **data**
  - The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

- **start**
  - The base period (as character) limited to the year and month, e.g. "2020-03".

- **end**
  - The research period (as character) limited to the year and month, e.g. "2020-04".

- **interval**
  - A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to `FALSE`) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start,end>` are considered and `start` defines the base period (`interval` is set to `TRUE`).

**Value**

The function returns a value (or vector of values) of the bilateral Tornqvist price index depending on the `interval` parameter. If the `interval` parameter is set to `TRUE`, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**


**Examples**

tornqvist(sugar, start="2018-12", end="2019-12")
tornqvist(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the multilateral TPD price index

**Description**

This function returns a value of the multilateral TPD (Time Product Dummy) price index.

**Usage**

```r
tpd(data, start, end, wstart = start, window = 13)
```

**Arguments**

- `data`: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. ‘2020-12-01’), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- `start`: The base period (as character) limited to the year and month, e.g. “2020-03”.
- `end`: The research period (as character) limited to the year and month, e.g. “2020-04”.
- `wstart`: The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. “2020-01”.
- `window`: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

**Value**

This function returns a value of the multilateral TPD price index which considers the time window defined by `wstart` and `window` parameters. It measures the price dynamics by comparing period `end` to period `start` (both `start` and `end` must be inside the considered time window). Please note that a Weighted Least Squares (WLS) regression is run with the expenditure shares in each period serving as weights. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**


**Examples**

```r
tpd(milk, start="2019-01", end="2019-08", window=10)
tpd(milk, start="2018-12", end="2019-12")
```
tpd_fbew

| tpd_fbew | Extending the multilateral TPD price index by using the FBEW method. |

Description

This function returns a value of the multilateral TPD price index (Time Product Dummy index) extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

```r
tpd_fbew(data, start, end)
```

Arguments

- **data**: The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: `year-month-day`, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral TPD price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods `end` and `start`. The month of the `start` parameter must be December. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
tpd_fbew(milk, start="2018-12", end="2019-08")
```
tpd_fbmw

Extending the multilateral TPD price index by using the FBMW method.

Description

This function returns a value of the multilateral TPD price index (Time Product Dummy index) extended by using the FBMW (Fixed Base Moving Window) method.

Usage

```r
tpd_fbmw(data, start, end)
```

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral TPD price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods `end` and `start` and it uses a 13-month time window with a fixed base month taken as `year(end)-1`. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
tpd_fbmw(milk, start="2019-12", end="2020-04")
```
tpd.splice 211

---

**tpd.splice** *Extending the multilateral TPD price index by using window splicing methods.*

**Description**

This function returns a value (or values) of the multilateral TPD price index (Time Product Dummy index) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

**Usage**

```r
tpd.splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)
```

**Arguments**

- `data` The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- `start` The base period (as character) limited to the year and month, e.g. "2019-12".
- `end` The research period (as character) limited to the year and month, e.g. "2020-04".
- `window` The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
- `splice` A character string indicating the splicing method. Available options are: "movement", "window","half","mean", "window_published","half_published","mean_published".
- `interval` A logical value indicating whether the function is to provide the price index comparing the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by `start`).

**Value**

This function returns a value or values (depending on `interval` parameter) of the multilateral TPD price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices.
indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

tpd_splice(milk, start="2018-12", end="2020-02", splice="half")

unit_value_index Calculating the unit value index

Description

This function returns a value (or vector of values) of the unit value index

Usage

unit_value_index(data, start, end, interval = FALSE)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
utpd

**start**
The base period (as character) limited to the year and month, e.g. "2020-03".

**end**
The research period (as character) limited to the year and month, e.g. "2020-04".

**interval**
A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

**Value**
The function returns a value (or vector of values) of the unit value index. The value index is calculated as the unit value at time start divided by the unit value at time start.

**References**

**Examples**
```
unit_value_index(sugar, start="2019-01", end="2020-01")
unit_value_index(sugar, start="2019-01", end="2020-01", interval=TRUE)
```

---

**utpd**

*Calculating the unweighted multilateral TPD price index*

**Description**
This function returns a value of the unweighted multilateral TPD (Time Product Dummy) price index.

**Usage**
```
utpd(data, start, end, wstart = start, window = 13)
```

**Arguments**
```
data  The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start  The base period (as character) limited to the year and month, e.g. "2020-03".

end    The research period (as character) limited to the year and month, e.g. "2020-04".

wstart The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).```
Value

This function returns a value of the unweighted multilateral TPD price index which considers the time window defined by \( w_{\text{start}} \) and \( \text{window} \) parameters. It measures the price dynamics by comparing period end to period start (both \( \text{start} \) and \( \text{end} \) must be inside the considered time window). Please note, that the estimation procedure runs the Ordinary Least Squares (OLS) method instead of the Weighted Least Squares (WLS) method like in the case of the TPD index. To get information about both price index values and corresponding dates, please see functions: \text{price\_indices} or \text{final\_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \text{final\_index} function).

References


Examples

```
utpd(milk, start="2019-01", end="2019-08",window=10)
utpd(milk, start="2018-12", end="2019-12")
```

---

| utpd_fbew | Extending the unweighted multilateral TPD price index by using the FBEW method. |

Description

This function returns a value of the unweighted multilateral TPD price index (Time Product Dummy index) extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

```
utpd_fbew(data, start, end)
```

Arguments

- **data**
  - The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

- **start**
  - The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**
  - The research period (as character) limited to the year and month, e.g. "2020-04".
Value
This function returns a value of the unweighted multilateral TPD price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods \text{end} and \text{start}. The month of the \text{start} parameter must be December. If the distance between \text{end} and \text{start} exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: \text{price_indices} or \text{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \text{final_index} function).

References

Examples
utpd_fbew(milk, start="2018-12", end="2019-08")

utpd_fbmw Extending the unweighted multilateral TPD price index by using the FBMW method.

Description
This function returns a value of the unweighted multilateral TPD price index (Time Product Dummy index) extended by using the FBMW (Fixed Base Moving Window) method.

Usage
utpd_fbmw(data, start, end)

Arguments
data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start The base period (as character) limited to the year and month, e.g. ”2019-12”.
end The research period (as character) limited to the year and month, e.g. ”2020-04”.
Value

This function returns a value of the unweighted multilateral TPD price index extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as \( \text{year}(\text{end})-1 \). If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References


Examples

\begin{verbatim}
utpd_fbmw(milk, start="2019-12", end="2020-04")
\end{verbatim}

\begin{verbatim}
utpd_splice(data, start, end, window = 13, splice = "movement", interval = FALSE)
\end{verbatim}

Description

This function returns a value (or values) of the unweighted multilateral TPD price index (Time Product Dummy index) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

\begin{verbatim}
utpd_splice(
    data, 
    start, 
    end, 
    window = 13, 
    splice = "movement", 
    interval = FALSE
)
\end{verbatim}
Arguments

data
The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

`start`
The base period (as character) limited to the year and month, e.g. "2019-12".

`end`
The research period (as character) limited to the year and month, e.g. "2020-04".

`window`
The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

`splice`
A character string indicating the splicing method. Available options are: "movement", "window","half","mean", "window_published","half_published","mean_published".

`interval`
A logical value indicating whether the function is to provide the price index comparing the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by `start`).

Value

This function returns a value or values (depending on `interval` parameter) of the unweighted multilateral TPD price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in `start` and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


value_index

Examples

```r
utpd_splice(milk, start="2018-12", end="2020-02", splice="half")
```

---

Calculating the value index

Description

This function returns a value (or vector of values) of the value index.

Usage

```r
value_index(data, start, end, interval = FALSE)
```

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval `<start, end>` are considered and `start` defines the base period (`interval` is set to TRUE).

Value

The function returns a value (or vector of values) of the value index. The value index is calculated as sum of expenditures from period `end` divided by sum of expenditures from period `start`.

References


Examples

```r
value_index(sugar, start="2019-01", end="2020-01")
value_index(sugar, start="2019-01", end="2020-01", interval=TRUE)
```
Calculating the bilateral Vartia-I price index

Description

This function returns a value (or vector of values) of the bilateral Vartia-I price index.

Usage

vartia(data, start, end, interval = FALSE)

Arguments

data The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".

interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Vartia-I price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

vartia(sugar, start="2018-12", end="2019-12")
vartia(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the bilateral Walsh price index

Description

This function returns a value (or vector of values) of the bilateral Walsh price index.

Usage

walsh(data, start, end, interval = FALSE)

Arguments

- **data**: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **interval**: A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval <start,end> are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Walsh price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

walsh(sugar, start="2018-12", end="2019-12")
walsh(milk, start="2018-12", end="2020-01", interval=TRUE)
Calculating the multilateral weighted WGEKS price index

Description

This function returns a value of the multilateral weighted WGEKS price index (to be more precise: the weighted GEKS index based on the Fisher formula).

Usage

\[
\text{wgeks(data, start, end, wstart = start, window = 13)}
\]

Arguments

- **data**: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **wstart**: The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".
- **window**: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral weighted WGEKS price index (to be more precise: the weighted GEKS index based on the Fisher formula) which considers the time window defined by `wstart` and `window` parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

\[
\text{wgeks(milk, start="2019-01", end="2019-08", window=10)}
\]
\[
\text{wgeks(milk, start="2018-12", end="2019-12"}
\]
Calculating the multilateral weighted WGEKS-AQI price index

Description

This function returns a value of the multilateral weighted WGEKS-AQI price index (to be more precise: the weighted GEKS index based on the asynchronous quality adjusted price index formula).

Usage

\[ \text{wgeksaqi}(\text{data}, \text{start}, \text{end}, \text{wstart} = \text{start}, \text{window} = 13) \]

Arguments

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2020-03".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
- **wstart**: The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".
- **window**: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral weighted WGEKS-AQI price index (to be more precise: the weighted GEKS index based on the asynchronous quality adjusted price index formula) which considers the time window defined by `wstart` and `window` parameters. It measures the price dynamics by comparing period `end` to period `start` (both `start` and `end` must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References

Examples

```
pgeksaqi(milk, start="2019-01", end="2019-08", window=10)
pgeksaqi(milk, start="2018-12", end="2019-12")
```

### Description

This function returns a value of the multilateral weighted GEKS-AQI price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

### Usage

```
pgeksaqi_fbew(data, start, end)
```

### Arguments

- **data**: The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".

### Value

This function returns a value of the multilateral weighted GEKS-AQI price index (the weighted GEKS index based on the asynchronous quality adjusted price index formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods `end` and `start`. The month of the `start` parameter must be December. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
wgeksaqi_fbmw(milk, start="2018-12", end="2019-08")
```

---

**Description**

This function returns a value of the multilateral weighted GEKS-AQI price index extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

```r
wgeksaqi_fbmw(data, start, end)
```

**Arguments**

- **data**: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. "2020-12-01"), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- **start**: The base period (as character) limited to the year and month, e.g. "2019-12".
- **end**: The research period (as character) limited to the year and month, e.g. "2020-04".
Value

This function returns a value of the multilateral weighted GEKS-AQI price index (the GEKS index based on the asynchronous quality adjusted price index formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods `end` and `start` and it uses a 13-month time window with a fixed base month taken as `year(end)-1`. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
wgeksaqi_fbmw(milk, start="2019-12", end="2020-04")
```

```
 wgeksaqi_splice
      Extending the multilateral weighted GEKS-AQI price index by using window splicing methods.
```

Description

This function returns a value (or values) of the multilateral weighted GEKS-AQI price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).
Usage

wgeksaqi_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

splice A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".

interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value

This function returns a value or values (depending on interval parameter) of the multilateral weighted GEKS-AQI price index (the weighted GEKS index based on the asynchronous quality adjusted price index formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


**Examples**

```r
wgeksaqui_splice(milk, start="2018-12", end="2020-02", splice="half")
```

### Description

This function returns a value of the multilateral weighted WGEKS-AQU price index (to be more precise: the weighted GEKS index based on the asynchronous quality adjusted unit value formula).

### Usage

```r
wgeksaqu(data, start, end, wstart = start, window = 13)
```

### Arguments

**data**  
The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. "2020-12-01"), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

**start**  
The base period (as character) limited to the year and month, e.g. "2020-03".

**end**  
The research period (as character) limited to the year and month, e.g. "2020-04".

**wstart**  
The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

**window**  
The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
Value

This function returns a value of the multilateral weighted WGEKS-AQU price index (to be more precise: the weighted GEKS index based on the asynchronous quality adjusted unit value formula) which considers the time window defined by $w_{start}$ and $window$ parameters. It measures the price dynamics by comparing period $end$ to period $start$ (both $start$ and $end$ must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
wgeksaqu(milk, start="2019-01", end="2019-08",window=10)
wgeksaqu(milk, start="2018-12", end="2019-12")
```

---

**wgeksaqu_fbew**

*Extending the multilateral weighted GEKS-AQU price index by using the FBEW method.*

Description

This function returns a value of the multilateral weighted GEKS-AQU price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

```r
wgeksaqu_fbew(data, start, end)
```

Arguments

data

The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
The base period (as character) limited to the year and month, e.g. "2019-12".

The research period (as character) limited to the year and month, e.g. "2020-04".

This function returns a value of the multilateral weighted GEKS-AQU price index (the weighted GEKS index based on the asynchronous quality adjusted unit value formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
wgeksaqu_fbw(milk, start="2018-12", end="2019-08")
```

Description

This function returns a value of the multilateral weighted GEKS-AQU price index extended by using the FBMW (Fixed Base Moving Window) method.

Usage

```r
wgeksaqu_fbmw(data, start, end)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>The user's data frame with information about sold products. It must contain columns: <em>time</em> (as Date in format: year-month-day, e.g. '2020-12-01'), <em>prices</em> (as positive numeric), <em>quantities</em> (as positive numeric) and <em>prodID</em> (as numeric, factor or character).</td>
</tr>
<tr>
<td>start</td>
<td>The base period (as character) limited to the year and month, e.g. &quot;2019-12&quot;.</td>
</tr>
<tr>
<td>end</td>
<td>The research period (as character) limited to the year and month, e.g. &quot;2020-04&quot;.</td>
</tr>
</tbody>
</table>

Value

This function returns a value of the multilateral weighted GEKS-AQU price index (the GEKS index based on the asynchronous quality adjusted unit value formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)−1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: *price_indices* or *final_index*. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the *final_index* function).

References


Examples

```r
wgeksaqu_fbmw(milk, start="2019-12", end="2020-04")
```

---

**wgeksaqu_splice**

*Extending the multilateral weighted GEKS-AQU price index by using window splicing methods.*
Description

This function returns a value (or values) of the multilateral weighted GEKS-AQU price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

```r
wgeksaqu_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)
```

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

splice A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".

interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value

This function returns a value or values (depending on interval parameter) of the multilateral weighted GEKS-AQU price index (the weighted GEKS index based on the asynchronous quality adjusted unit value formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

wgeksaqu_splice(milk, start="2018-12", end="2020-02",splice="half")

wgeksgaqi

*Calculating the multilateral weighted WGEKS-GAQI price index*

Description

This function returns a value of the multilateral weighted WGEKS-GAQI price index (to be more precise: the weighted GEKS index based on the geometric asynchronous quality adjusted price index formula).

Usage

wgeksgaqi(data, start, end, wstart = start, window = 13)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2020-03".

end The research period (as character) limited to the year and month, e.g. "2020-04".
The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

This function returns a value of the multilateral weighted WGEKS-GAQI price index (to be more precise: the weighted GEKS index based on the geometric asynchronous quality adjusted price index formula) which considers the time window defined by \texttt{wstart} and \texttt{window} parameters. It measures the price dynamics by comparing period \texttt{end} to period \texttt{start} (both \texttt{start} and \texttt{end} must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

\textbf{References}


\textbf{Examples}

\begin{verbatim}
 wgeksgaqi(milk, start="2019-01", end="2019-08",window=10)
 wgeksgaqi(milk, start="2018-12", end="2019-12")
\end{verbatim}

\textbf{Description}

This function returns a value of the multilateral weighted GEKS-GAQI price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

\textbf{Usage}

\begin{verbatim}
 wgeksgaqi_fbew(data, start, end)
\end{verbatim}
Arguments

data   The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start   The base period (as character) limited to the year and month, e.g. ”2019-12”.

date   The research period (as character) limited to the year and month, e.g. ”2020-04”.

Value

This function returns a value of the multilateral weighted GEKS-GAQI price index (the weighted GEKS index based on the geometric asynchronous quality adjusted price index formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

```
wgkeksgaqi_fbew(milk, start=”2018-12”, end=”2019-08”)
```

Description

This function returns a value of the multilateral weighted GEKS-GAQI price index extended by using the FBMW (Fixed Base Moving Window) method.
Usage

wgeksgaqi_fbmw(data, start, end)

Arguments

data: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

start: The base period (as character) limited to the year and month, e.g. "2019-12".

data: The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral weighted GEKS-GAQI price index (the GEKS index based on the geometric asynchronous quality adjusted price index formula) extended by using the FBWM (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)-1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

wgeksgaqi_fbmw(milk, start="2019-12", end="2020-04")
Description

This function returns a value (or values) of the multilateral weighted GEKS-GAQI price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

wgeksgaqi_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. ”2019-12”.

end The research period (as character) limited to the year and month, e.g. ”2020-04”.

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

splice A character string indicating the splicing method. Available options are: ”movement”, ”window”, ”half”, ”mean”, ”window_published”, ”half_published”, ”mean_published”.

interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value

This function returns a value or values (depending on interval parameter) of the multilateral weighted GEKS-GAQI price index (the weighted GEKS index based on the geometric asynchronous quality adjusted price index formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).
References


Examples

wgeksgaqi_splice(milk, start="2018-12", end="2020-02", splice="half")

wgeksgl

Calculating the multilateral weighted WGEKS-GL price index

Description

This function returns a value of the multilateral weighted WGEKS-GL price index (to be more precise: the weighted GEKS index based on the geometric Laspeyres formula).

Usage

wgeksgl(data, start, end, wstart = start, window = 13)

Arguments

data

The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start

The base period (as character) limited to the year and month, e.g. ”2020-03”.

end

The research period (as character) limited to the year and month, e.g. ”2020-04”.

wstart

The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. ’2020-01’.

window

The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
**Value**

This function returns a value of the multilateral weighted WGEKS-GL price index (to be more precise: the weighted GEKS index based on the geometric Laspeyres formula) which considers the time window defined by \( w_{\text{start}} \) and \( \text{window} \) parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: \textit{price_indices} or \textit{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \textit{final_index} function).

**References**


**Examples**

\begin{verbatim}
wgeksgl(milk, start="2019-01", end="2019-08",window=10)
wgeksgl(milk, start="2018-12", end="2019-12")
\end{verbatim}

---

**wgeksgl_fbew**

Extended the multilateral weighted GEKS-GL price index by using the FBEW method.

**Description**

This function returns a value of the multilateral weighted GEKS-GL price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

**Usage**

\begin{verbatim}
wgeksgl_fbew(data, start, end)
\end{verbatim}

**Arguments**

- **data** The user’s data frame with information about sold products. It must contain columns: \textit{time} (as Date in format: year-month-day, e.g. "2020-12-01"), \textit{prices} (as positive numeric), \textit{quantities} (as positive numeric) and \textit{prodID} (as numeric, factor or character).
- **start** The base period (as character) limited to the year and month, e.g. "2019-12".
- **end** The research period (as character) limited to the year and month, e.g. "2020-04".
Value

This function returns a value of the multilateral weighted GEKS-GL price index (the weighted GEKS index based on the geometric Laspeyres formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

References


Examples

```r
wgeksgl_fbew(milk, start="2018-12", end="2019-08")
```

Description

This function returns a value of the multilateral weighted GEKS-GL price index extended by using the FBMW (Fixed Base Moving Window) method.

Usage

```
wgeksgl_fbmw(data, start, end)
```
**Arguments**

- **data**
  The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

- **start**
  The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**
  The research period (as character) limited to the year and month, e.g. "2020-04".

**Value**

This function returns a value of the multilateral weighted GEKS-GL price index (the GEKS index based on the geometric Laspeyres formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods `end` and `start` and it uses a 13-month time window with a fixed base month taken as year(`end`)-1. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**


**Examples**

```r
wgeksgl_fbmw(milk, start="2019-12", end="2020-04")
```

---

**wgeksgl_splice**

*Extending the multilateral weighted GEKS-GL price index by using window splicing methods.*
**Description**

This function returns a value (or values) of the multilateral weighted GEKS-GL price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

**Usage**

```r
wgeksgl_splice(
  data, 
  start, 
  end, 
  window = 13, 
  splice = "movement", 
  interval = FALSE 
)
```

**Arguments**

- `data`: The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- `start`: The base period (as character) limited to the year and month, e.g. "2019-12".
- `end`: The research period (as character) limited to the year and month, e.g. "2020-04".
- `window`: The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
- `splice`: A character string indicating the splicing method. Available options are: "movement", "window", "half", "mean", "window_published", "half_published", "mean_published".
- `interval`: A logical value indicating whether the function is to provide the price index comparing the research period defined by `end` to the base period defined by `start` (then `interval` is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by `start`).

**Value**

This function returns a value or values (depending on `interval` parameter) of the multilateral weighted GEKS-GL price index (the weighted GEKS index based on the geometric Laspeyres formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in `start` and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).
References


Examples

```r
wgeksl_splice(milk, start="2018-12", end="2020-02",splice="half")
```

### Description

This function returns a value of the multilateral weighted WGEKS-L price index (to be more precise: the weighted GEKS index based on the Laspeyres formula).

### Usage

```r
wgeksl(data, start, end, wstart = start, window = 13)
```

### Arguments

- `data`: The user's data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).
- `start`: The base period (as character) limited to the year and month, e.g. "2020-03".
- `end`: The research period (as character) limited to the year and month, e.g. "2020-04".
wgeksl_fbew

wstart  The beginning of the time interval (which is used by multilateral methods) limited to the year and month, e.g. "2020-01".

window  The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

Value

This function returns a value of the multilateral weighted WGEKS-L price index (to be more precise: the weighted GEKS index based on the Laspeyres formula) which considers the time window defined by wstart and window parameters. It measures the price dynamics by comparing period end to period start (both start and end must be inside the considered time window). To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

wgeksl(milk, start="2019-01", end="2019-08", window=10)
wgeksl(milk, start="2018-12", end="2019-12")

wgeksl_fbew

Extending the multilateral weighted GEKS-L price index by using the FBEW method.

Description

This function returns a value of the multilateral weighted GEKS-L price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.

Usage

wgeksl_fbew(data, start, end)
Arguments

data
The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start
The base period (as character) limited to the year and month, e.g. "2019-12".

df_result
The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral weighted GEKS-L price index (the weighted GEKS index based on the Laspeyres formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods end and start. The month of the start parameter must be December. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

wgeksl_fbew(milk, start="2018-12", end="2019-08")
wgeksl_fbmw

Extending the multilateral weighted GEKS-L price index by using the FBMW method.

Description

This function returns a value of the multilateral weighted GEKS-L price index extended by using the FBMW (Fixed Base Moving Window) method.

Usage

wgeksl_fbmw(data, start, end)

Arguments

data: The user's data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start: The base period (as character) limited to the year and month, e.g. "2019-12".

end: The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral weighted GEKS-L price index (the GEKS index based on the Laspeyres formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods end and start and it uses a 13-month time window with a fixed base month taken as year(end)-1. If the distance between end and start exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the start parameter must be December. To get information about both price index values and corresponding dates, please see functions: price_indices or final_index. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the final_index function).

References


Examples

wgeksl_fbmw(milk, start="2019-12", end="2020-04")

Description

This function returns a value (or values) of the multilateral weighted GEKS-L price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

wgeksl_splice(
  data, 
  start, 
  end, 
  window = 13, 
  splice = "movement", 
  interval = FALSE
)

Arguments

data       The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. ’2020-12-01’), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).
start      The base period (as character) limited to the year and month, e.g. ”2019-12”.
end        The research period (as character) limited to the year and month, e.g. ”2020-04”.
window     The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).
splice     A character string indicating the splicing method. Available options are: ”movement”, ”window”, ”half”, ”mean”, ”window_published”, ”half_published”, ”mean_published”.
interval   A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).
This function returns a value or values (depending on interval parameter) of the multilateral weighted GEKS-L price index (the weighted GEKS index based on the Laspeyres formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References). The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**


**Examples**

```r
wgeksl_splice(milk, start="2018-12", end="2020-02",splice="half")
```

**Description**

This function returns a value of the multilateral weighted GEKS price index extended by using the FBEW (Fixed Base Monthly Expanding Window) method.
Usage

\texttt{wgeks_fbew(data, start, end)}

Arguments

- **data**
  The user's data frame with information about sold products. It must contain columns: \texttt{time} (as Date in format: year-month-day, e.g. '2020-12-01'), \texttt{prices} (as positive numeric), \texttt{quantities} (as positive numeric) and \texttt{prodID} (as numeric, factor or character).

- **start**
  The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**
  The research period (as character) limited to the year and month, e.g. "2020-04".

Value

This function returns a value of the multilateral weighted GEKS price index (the weighted GEKS index based on the Fisher formula) extended by using the FBEW (Fixed Base Monthly Expanding Window) method. The FBEW method uses a time window with a fixed base month every year (December). The window is enlarged every month with one month in order to include information from a new month. The full window length (13 months) is reached in December of each year. The function measures the price dynamics between periods \texttt{end} and \texttt{start}. The month of the \texttt{start} parameter must be December. If the distance between \texttt{end} and \texttt{start} exceeds 13 months, then internal Decembers play a role of chain-linking months. To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References


Examples

\texttt{wgeks_fbew(milk, start="2018-12", end="2019-08")}
**wgeks_fbmw**  
*Extending the multilateral weighted GEKS price index by using the FBMW method.*

**Description**

This function returns a value of the multilateral weighted GEKS price index extended by using the FBMW (Fixed Base Moving Window) method.

**Usage**

\[
\text{wgeks_fbmw(data, start, end)}
\]

**Arguments**

- **data**
  
  The user’s data frame with information about sold products. It must contain columns: `time` (as Date in format: year-month-day, e.g. '2020-12-01'), `prices` (as positive numeric), `quantities` (as positive numeric) and `prodID` (as numeric, factor or character).

- **start**
  
  The base period (as character) limited to the year and month, e.g. "2019-12".

- **end**
  
  The research period (as character) limited to the year and month, e.g. "2020-04".

**Value**

This function returns a value of the multilateral weighted GEKS price index (the weighted GEKS index based on the Fisher formula) extended by using the FBMW (Fixed Base Moving Window) method. It measures the price dynamics between periods `end` and `start` and it uses a 13-month time window with a fixed base month taken as `year(end)-1`. If the distance between `end` and `start` exceeds 13 months, then internal Decembers play a role of chain-linking months. The month of the `start` parameter must be December. To get information about both price index values and corresponding dates, please see functions: `price_indices` or `final_index`. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the `final_index` function).

**References**


**Examples**

\[
\text{wgeks_fbmw(milk, start="2019-12", end="2020-04")}
\]
Extending the multilateral weighted GEKS price index by using window splicing methods.

Description

This function returns a value (or values) of the multilateral weighted GEKS price index extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).

Usage

wgeks_splice(
  data,
  start,
  end,
  window = 13,
  splice = "movement",
  interval = FALSE
)

Arguments

data The user’s data frame with information about sold products. It must contain columns: time (as Date in format: year-month-day, e.g. '2020-12-01'), prices (as positive numeric), quantities (as positive numeric) and prodID (as numeric, factor or character).

start The base period (as character) limited to the year and month, e.g. "2019-12".

end The research period (as character) limited to the year and month, e.g. "2020-04".

window The length of the time window (as positive integer: typically multilateral methods are based on the 13-month time window).

splice A character string indicating the splicing method. Available options are: "movement", "window","half","mean", "window_published","half_published","mean_published".

interval A logical value indicating whether the function is to provide the price index comparing the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base multilateral indices are to be presented (the fixed base month is defined by start).

Value

This function returns a value or values (depending on interval parameter) of the multilateral weighted GEKS price index (the weighted GEKS index based on the Fisher formula) extended by using window splicing methods. Available splicing methods are: movement splice, window splice, half splice, mean splice and their additional variants: window splice on published indices (WISP), half splice on published indices (HASP) and mean splice on published indices (see References).
The time window starts in start and should consist of at least two months. To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References


Examples

\begin{verbatim}
wgeks_splice(milk, start="2018-12", end="2020-02", splice="half")
\end{verbatim}

\begin{verbatim}
\begin{tabular}{ll}
young & \textit{Calculating the bilateral Young price index} \\
\end{tabular}
\end{verbatim}

Description

This function returns a value (or vector of values) of the bilateral Young price index.

Usage

\begin{verbatim}
young(data, start, end, base = start, interval = FALSE)
\end{verbatim}

Arguments

\begin{verbatim}
data & The user's data frame with information about sold products. It must contain columns: \texttt{time} (as Date in format: year-month-day, e.g. "2020-12-01"), \texttt{prices} (as positive numeric), \texttt{quantities} (as positive numeric) and \texttt{prodID} (as numeric, factor or character). \\
start & The base period (as character) limited to the year and month, e.g. "2020-03". \\
end & The research period (as character) limited to the year and month, e.g. "2020-04". \\
base & The prior period used in the Young price index formula (as character) limited to the year and month, e.g. "2020-01"
\end{verbatim}
interval A logical value indicating whether the function is to compare the research period defined by end to the base period defined by start (then interval is set to FALSE) or all fixed base indices are to be calculated. In this latter case, all months from the time interval \(<\text{start},\text{end}>\) are considered and start defines the base period (interval is set to TRUE).

Value

The function returns a value (or vector of values) of the bilateral Young price index depending on the interval parameter. If the interval parameter is set to TRUE, the function returns a vector of price index values without dates. To get information about both price index values and corresponding dates, please see functions: \texttt{price_indices} or \texttt{final_index}. The function does not take into account aggregating over outlets or product subgroups (to consider these types of aggregating, please use the \texttt{final_index} function).

References


Examples

\begin{verbatim}
young(sugar, start="2019-01", end="2020-01", base="2018-12")
young(milk, start="2018-12", end="2020-01", interval=TRUE)
\end{verbatim}
Index

* datasets
  coffee, 54
data_DOWN_UP_SIZED, 67
dataAGGR, 62
dataOICOP, 63
dataMATCH, 63
dataU, 64
milk, 173
sugar, 205

agmean, 6, 184
available, 7, 183

banajree, 8, 184
bennet, 9
bialek, 10, 184
bmw, 11, 183

carli, 12, 183
ccdi, 13, 185
ccdi_fbew, 14, 187
ccdi_fbmmw, 15, 187
ccdi_splice, 16, 186
chagmean, 18, 185
chbanajree, 19, 185
chbialek, 20, 185
chbmw, 21, 184
chcarli, 22, 184
chcswd, 23, 184
chdavies, 24
chdikhanov, 25, 185
chdrobisch, 26, 185
chdutot, 27, 184
chfisher, 28, 185
chgeary_khamis, 29, 185
chgeohybrid, 30, 185
chgeolaspeyres, 31, 185
chgeolowe, 32, 185
chgeopaasche, 33, 185
chgeoyoung, 34, 185
chharmonic, 35, 185
chhybrid, 36, 185
chIQMp, 37
chjevons, 38, 185
chlaspeyres, 39, 185
chlehr, 40, 185
chlloyd_moultion, 41, 185
chlowe, 42, 185
chmarshall_edgeworth, 43, 185
chpaasche, 44, 185
chpalgrave, 45, 185
chQMq, 46
chsato_vartia, 48, 185
chstauvel, 49, 185
chtornqvist, 50, 185
chvartia, 51, 185
chwalsh, 52, 185
chyoung, 53, 185
coffee, 54, 182
compare_distances, 55, 188
compare_indices_df, 56, 188
compare_indices_jk, 56, 188
compare_indices_list, 59, 188
compare_to_target, 60, 188
cswd, 61, 183
data_aggregating, 65, 183
data_check, 65, 183
data_classifying, 66, 183
data_DOWN_UP_SIZED, 67, 182
data_filtering, 67, 183
data_imputing, 69
data_matching, 70, 74, 78, 181–183, 188, 195, 196
data_norm, 72, 183
data_preparing, 73, 183
data_reducing, 75, 183
data_selecting, 74, 76, 183
data_unit, 77, 183
INDEX

dataAGGR, 62, 182
dataCOICOP, 63, 182
dataMATCH, 63, 182
dataU, 64, 182
davies, 78, 184, 185
dikhanov, 79, 184
dissimilarity, 80
dissimilarity_fig, 81, 183
drobisch, 82, 184
dutot, 83, 183
elasticity, 84, 183
elasticity_fig, 85, 183
expenditures, 87
fisher, 89, 184
geary_khamis, 90, 184
geks, 91, 185
gerks_fbew, 143, 187
gerks_fbmw, 144, 187
gerks_splice, 145, 186
gerksaqi, 93, 186
gerksaqi_fbew, 94, 187
gerksaqi_fbmw, 95, 187
gerksaqi_splice, 96, 186
gerksau, 98, 186
gerksau_fbew, 99, 187
gerksau_fbmw, 100, 187
gerksau_splice, 101, 186
gerksgqi, 103, 186
gerksgqi_fbew, 104, 187
gerksgqi_fbmw, 105, 187
gerksgqi_splice, 106, 186
gerksgl, 108, 185
gerksgl_fbew, 109, 187
gerksgl_fbmw, 110, 187
gerksgl_splice, 111, 186
gerksiqm, 113
gerksiqm_fbew, 114
gerksiqm_fbmw, 115
gerksiqm_splice, 116
gerksj, 118, 186
gerksj_fbew, 119, 187
gerksj_fbmw, 120, 187
gerksj_splice, 121, 186
gerksl, 123, 185
gerksl_fbew, 129, 187
gerksl_fbmw, 130, 187
gerksl_splice, 131, 186
gerkslm, 124
gerkslm_fbew, 125
gerkslm_fbmw, 126
gerkslm_splice, 127
gerksq, 133
gerksq_fbew, 134
gerksq_fbmw, 135
gerksq_splice, 136
gerksw, 138, 186
gerksw_fbew, 139, 187
gerksw_fbmw, 140, 187
gerksw_splice, 141, 186
generate, 146, 182
generate_CES, 148
gehybrid, 149, 184
gelaspeyres, 150, 184
geolowe, 151, 184
geopasche, 152, 184
geoyoung, 153, 184
gk, 29, 91, 154, 186
gk_fbew, 155, 187
gk_fbmw, 156, 187
gk_splice, 157, 186
geometric, 159, 184
hybrid, 160, 184
IQMp, 161
jevons, 162, 184
laspeyres, 163, 184
lehr, 164, 184
lloyd_moulton, 165, 184
load_model, 166, 183
lowe, 167, 184
marshall_edgeworth, 168, 184
matched, 169, 183
matched_fig, 170, 183
matched_index, 170, 171, 171, 183
mbennet, 172
milk, 173, 182
mmontgomery, 174
model_classification, 175, 183
montgomery, 177
paasche, 178, 184
palgrave, 179, 184
pqcor, 180, 183
pqcor_fig, 181, 183
PriceIndices, 57, 88, 182, 189
PriceIndices-package (PriceIndices), 182
prices, 183, 188
products, 190
products_fig, 191
QMp, 192
QMq, 193
QU, 186, 194
quantities, 183, 195
sales, 183, 196
sales_groups, 183, 197
sales_groups2, 183, 198
sato_vartia, 184, 199
save_model, 183, 200
shrinkflation, 183, 201
SPQ, 186, 203
stuvel, 184, 204
sugar, 182, 205
tindex, 182, 206
tornqvist, 184, 207
tpd, 186, 208
tpd_fbew, 187, 209
tpd_fbmw, 187, 210
tpd_splice, 186, 211
unit_value_index, 212
utpd, 213
utpd_fbew, 214
utpd_fbmw, 215
utpd_splice, 216
value_index, 218
vartia, 184, 219
walsh, 184, 220
wgeks, 185, 221
wgeks_fbew, 187, 247
wgeks_fbmw, 187, 249
wgeks_splice, 186, 250
wgeksaqi, 186, 222
wgeksaqi_fbew, 187, 223
wgeksaqi_fbmw, 187, 224
wgeksaqi_splice, 186, 225
wgeksaqi, 186, 227
wgeksaqi_fbew, 187, 228
wgeksaqi_fbmw, 187, 229
wgeksaqi_splice, 186, 230
wgeksaqi, 186, 232
wgeksaqi_fbew, 187, 233
wgeksaqi_fbmw, 187, 234
wgeksaqi_splice, 186, 235
wgeksgl, 186, 237
wgeksgl_fbew, 187, 238
wgeksgl_fbmw, 187, 239
wgeksgl_splice, 186, 240
wgeksl, 185, 242
wgeksl_fbew, 187, 243
wgeksl_fbmw, 187, 245
wgeksl_splice, 186, 246
young, 184, 251
wgeksaqi_fbew, 187, 228
wgeksaqi_splice, 186, 230
wgeksaqi, 186, 232
wgeksaqi_fbew, 187, 233
wgeksaqi_fbmw, 187, 234
wgeksaqi_splice, 186, 235
wgeksgl, 186, 237
wgeksgl_fbew, 187, 238
wgeksgl_fbmw, 187, 239
wgeksgl_splice, 186, 240
wgeksl, 185, 242
wgeksl_fbew, 187, 243
wgeksl_fbmw, 187, 245
wgeksl_splice, 186, 246
young, 184, 251