PriceIndices – a New R Package for Bilateral and Multilateral Price Index Calculations

Jacek Białek i [1] | *University of Lodz, Lodz, Poland*

Abstract

The methodology of price indices dedicated to scanner data is broad, multifaceted, and still contains many open problems. The main challenges include choosing the index formula and the time window width in multilateral methods, as well as determining splicing or other data updating methods. Many NSIs experiment with scanner data, their processing, classifying, matching, and finally using this type of data for CPI calculations. However, these activities are limited, which is partly due to the lack of widely available software in this field. On the one hand, R packages dedicated to price indices are available (e.g. *IndexNumR or micEconIndex*), on the other hand, their functionality and the scope of implemented methods are quite limited. The article discusses a new R package, i.e. *PriceIndices*, which is used to process scanner data and to calculate bilateral and multilateral price indices. The assumptions for the construction of the package were such that it would serve both practitioners and scientists through a multitude of methods and their parameterization. The main purpose of the article is to present the utility of the package in the field of analyzing the dynamics of scanner prices. All obtained results are based on the real scanner data set on milk obtained from one retailer chain in Poland and included in the *PriceIndices* package.[2]

Keywords

Scanner data, bilateral indices, multilateral indices, elementary indices, chain indices, superlative indices

JEL code

C43, E31

INTRODUCTION

Scanner data, alongside web-scraped data, have recently been a fairly popular alternative source of information in Consumer Price Index (CPI) measurement. The availability of electronic sales data for the calculation of the CPI has increased over the past 18 years. Scanner data mean transaction data

[1] Department of Statistical Methods, University of Lodz, Rewolucji 1905 r. 41, 90-214 Lodz, Poland. E-mail: jacek.bialek@uni.lodz.pl. Also Statistics Poland, Department of Trade and Services, Aleja Niepodległości 208, 00-925 Warsaw, Poland. E-mail: j.bialek@stat.gov.pl.

[2] This paper was created as part of the project "Construction of an integrated retail price statistics system" (INSTATCENY), financed by the National Center for Research and Development (1st edition of GOSPOSTRATEG, No. 1/382525/14/NCBR/2018).

that specify turnover and numbers of items sold by GTIN or another bar code and can be obtained from a wide variety of retailers (supermarkets, home electronics, internet shops, etc.). Scanner data contain expenditure information at the item level, which makes it possible to use expenditure shares of items as weights for calculating price indices at the lowest (elementary) level of data aggregation. They provide some additional information about products (such as the following attributes: size, color, package quantity, etc.) which may be useful in aggregating items into homogeneous groups.

Nevertheless, there are many challenges while using scanner data sets. The first group of challenges is connected with data processing, i.e. row scanner data must be cleaned, classified, matched and (optionally) filtered (Loon and Roels, 2018). This is a huge challenge – these stages usually require the use of advanced techniques of multivariate statistics and machine-learning and/or text-mining methods. Sometimes these stages force NSIs to build new IT systems and sometimes additional, separate modules are created in *R*, *Python*, *SAS, Mathematica* or other environments. The second group of challenges concerns choosing the optimal price index formula and the time window width in multilateral methods, determining splicing or other data updating methods. Experiments in this area are somewhat limited, because most statisticians do not have access to the appropriate software or do not have specialist knowledge in computer science to create above-mentioned scripts. Although the methodology for scanner data and multilateral indices is extensive and constantly evolving (see for instance Ivancic et al., 2011; Krsnich, 2014; Griffioen and Bosch, 2016; de Haan et al., 2016; Chessa and Griffioen, 2016; Chessa, 2017; Chessa et al., 2017; Diewert and Fox, 2017; von Auer, 2019; Mehrhoff, 2019; Białek and Bobel, 2019; Webster and Tarnow-Mordi, 2019; Abe and Rao, 2019; Zhang et al., 2019), functionality of available packages for price index calculations and the scope of implemented methods are still quite limited. For instance, in the case of the popular *IndexNumR* package, the only multilateral price index formula which is available in this package is the GEKS index. Moreover, extending the GEKS index is possible by using "only" three splicing methods: movement splice, window splice and mean splice, i.e. the half splice is not available here, along with the FBEW or FBMW methods (see Section 4). Another R package, *micEconIndex*, provides only a small number of bilateral indices, i.e. the Paasche, the Laspeyres and the Fisher indices. There are some other R packages (e.g. *multilaterals* or *productivity*) but still the list of available methods is quite poor and these packages are not strictly dedicated to the scanner data case.

The article discusses a new R package, i.e. *PriceIndices*, which is used to process scanner data and to calculate bilateral and multilateral price indices. The assumptions for the construction of the package were such that it would serve both practitioners and scientists through a multitude of methods and their parameterization. The main purpose of the article is to present the utility of the package in the field of analyzing the dynamics of scanner prices. All obtained results are based on the real scanner data set on milk obtained from one retailer chain in Poland and included in the *PriceIndices* package. Presentation of this package is divided into the following sections: *data processing, bilateral index calculations, multilateral index calculations, extensions of multilateral indices, aggregation* and *index comparisons*.

1 DATA PROCESSING

The released version of the Priceindices package can be installed from GitHub with the command: *install_github("JacekBialek/PriceIndices")* or from *CRAN: install.packages("PriceIndices")*. This section discusses the basic package functions for scanner data processing. Technical details are omitted since they can be found in the package documentation. Please note that the package uses monthly unit values as prices and, as a consequence, daily data on prices and quantities are aggregated to one month.

1.1 Data sets included in the *Priceindices* package

This package includes two data sets: artificial and real. The first one, **dataMATCH**, can be used to demonstrate the **data_matching** function and it will be described later (see Section 1.4). The other one, **milk**, is a collection of scanner data on the sale of milk in one of Polish supermarkets in the period from December 2018 to August 2020. It is a data frame with 6 columns and 4 281 rows. The used variables (columns) 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 obtained after product matching (data set contains 67 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 corresponding to subgroups of milk group). The set milk represents a typical data frame used in the package for most calculations and is organized as follows.

Table 1 First six rows of the "milk" data set

Time	Prices	Quantities	prodID	retID	Description
2018-12-01	7.98	6.5	400032	1311	goat milk
2018-12-01	7.98	91.5	400032	2210	goat milk
2018-12-01	7.98	19.5	400032	6610	goat milk
2018-12-01	7.98	15.5	400032	7611	goat milk
2018-12-01	7.98	43.	400032	8910	goat milk
2019-01-01	7.98	4.5	400032	1311	goat milk

Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

The milk data set contains transaction data on 6 milk subgroups: goat milk, powdered milk, full-fat milk UHT, full-fat milk pasteurized, low-fat milk UHT, and low-fat milk pasteurized.

1.2 Data preparation

Data_preparing is a function for data preparation. 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). The resulting data frame is free from missing values, zero or negative prices and quantities. As a result, the time column is set to be *Date* type (in format: 'Year-Month-01'), columns of prices and quantities are set to be *numeric*. If the description parameter is set to TRUE, then the description column is set to be *character* type (otherwise it is deleted). Please note that the **milk** set is an already prepared dataset but let us assume for a moment that we want to make sure that it does not contain missing values and we do not need the description column for further calculations. For this purpose, we use the data_preparing function as follows: `data_preparing(milk, description=FALSE)`.

1.3 Product classification

Advanced machine-learning methods use for product classification so-called *learning trials*. However, this requires the manual assigning of the appropriate COICOP group to product codes. When we have meticulous and accurate product descriptions, we can alternatively classify products based on the words or phrases appearing in their description. This is done by using the **data_selecting** function. 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** (see documentation). For instance, please use `data_selecting(milk, include=c("milk"), must=c("UHT"))` to obtain a subset of **milk** dataset limited to the UHT category (Table 2) or `data_selecting(milk, must=c("milk"), exclude=c("past","goat"))` to obtain a subset of **milk** dataset with products which are not pasteurized and which are not goat products (Table 3).

Table 2 First six rows of the UHT milk subset (*)

Time	Prices	Quantities	prodID	retID	Description
2018-12-01	2.99	113	60010	1311	full-fat milk UHT
2018-12-01	2.29	650	401350	1311	full-fat milk UHT
2018-12-01	2.68	304	402570	1311	full-fat milk UHT
2018-12-01	2.65	137	405419	1311	full-fat milk UHT
2018-12-01	2.99	560	60010	2210	full-fat milk UHT
2018-12-01	2.50	1914	401350	2210	full-fat milk UHT

Note: (*) Available values of description: *"full-fat milk UHT", "low-fat milk UHT".*
Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

Table 3 First six rows of the not-pasteurized and not-goat milk subset (*)

Time	Prices	Quantities	prodID	retID	Description
2018-12-01	19.58	10.5	403249	1311	powdered milk
2018-12-01	19.58	154.5	403249	2210	powdered milk
2018-12-01	19.58	88.5	403249	6610	powdered milk
2018-12-01	19.58	75.0	403249	7611	powdered milk
2018-12-01	19.58	18.0	403249	8910	powdered milk
2018-12-01	69.95	1.0	400033	2210	powdered milk

Note: (*) Available values of description: *"powdered milk", "full-fat milk UHT", "low-fat milk UHT".*
Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

1.4 Product matching

If the user has a dataset with information about products sold but these products are not matched, then the **data_matching** function can be used. In an optimal situation, an input data frame contains the **codeIN**, **codeOUT** and **description** columns (see documentation) which in practice will contain retailer codes, GTIN or SKU codes and product labels, respectively. The **data_matching** 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 available variables for matching. In the most extreme case, when the **onlydescription** parameter is set to TRUE (its default value is FALSE), two products are also matched if they have identical descriptions. Other 5 cases differ from each other with regard to the set of considered variables, for instance, the algorithm for product matching when both product codes (internal and external) are available differs from the algorithm when only one of these codes is available (see the package documentation). If the matching process is to compare product labels defined by the **description** column, then the Jaro-Winkler distance measure is used (Jaro, 1989; Winkler, 1990) to compare each pair of character strings. For instance, let us suppose we want to match products from the artificial data set (**dataMATCH**) included in the package (see Table 4). Let us assume that products with two identical codes (**codeIN** and **codeOUT**) or one of thecodes identical and an identical description are automatically matched. Products are also matched if they have one of the codes identical and the Jaro-Winkler similarity measure, calculated for their descriptions, is bigger than the fixed **precision** value, e.g. let us set its level to 0.98. Let us suppose also that we want to match all products sold in the interval: December 2018–February 2019. Using the following command:

data_matching(dataMATCH, start="2018-12", end="2019-02", codeIN=TRUE, codeOUT=TRUE, precision=.98, interval=TRUE), an additional column (**prodID**) will be added to the data frame (Table 5). Now the data set is ready for further processing (e.g. data filtering) and/or price index calculations.

Table 4 First six rows of the **dataMATCH** set before matching

Time	Prices	Quantities	codeIN	codeOUT	retID	Description
2018-12-01	9.416371	309	1	1	1	product A
2019-01-01	9.881875	325	1	5	1	product A
2019-02-01	12.611826	327	1	1	1	product A
2018-12-01	9.598252	309	3	2	1	product A
2019-01-01	9.684900	325	3	2	1	product A
2019-02-01	9.358420	327	3	2	1	product A

Source: PriceIndices R package *<https://CRAN.R-project.org/package=PriceIndices>*

Table 5 First six rows of the **dataMATCH** set after matching

Time	Prices	Quantities	codeIN	codeOUT	retID	Description	prodID
2018-12-01	9.416371	309	1	1	1	product A	24
2019-01-01	9.881875	325	1	5	1	product A	24
2019-02-01	12.611826	327	1	1	1	product A	24
2018-12-01	9.598252	309	3	2	1	product A	30
2019-01-01	9.684900	325	3	2	1	product A	30
2019-02-01	9.358420	327	3	2	1	product A	30

Source: PriceIndices R package *<https://CRAN.R-project.org/package=PriceIndices>*

1.5 Data filtering

The *PriceIndices* package includes the **data_filtering** function for data set reduction. 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** parameter (see documentation). If the set of filters is empty, then the function returns the original data frame (defined by data parameter). On the other hand, if both filters are chosen, i.e. *filters=c(extremeprices, 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. For example, let us assume we consider three filters for the **milk** data set: **filter 1** is to reject 1% of the lowest and 1% of the highest price changes comparing March 2019 to December 2018, **filter 2** is to reject products with price ratio being less than 0.5 or bigger than 2 in the same time, **filter 3** rejects the same products as filter2 rejects and also products with relatively low sale in compared months. An additional **filter 4** works for each pair of subsequent months from the considered time interval and under the condition that filtering is done for each outlet (retID) separately. The right commands for these filters and their impact on **milk** data set reduction (with 403 and 817 records when no filter is used in comparison of two months and the whole time interval respectively) are presented in Table 6.

Table 6 The impact of filters 1–4 on data set reduction

Type of filter	Command	No. of records after filtering
Filter 1	data_filtering(milk,start="2018-12",end="2019-03",filters=c("extremeprices"),pquantiles=c(0.01,0.99))	378
Filter 2	data_filtering(milk,start="2018-12",end="2019-03",filters=c("extremeprices"),plimits=c(0.5,2))	381
Filter 3	data_filtering(milk,start="2018-12",end="2019-03",filters=c("extremeprices","lowsales"),plimits=c(0.5,2))	180
Filter 4	data_filtering(milk,start="2018-12",end="2019-03",filters=c("extremeprices"),pquantiles=c(0.01,0.99),interval=TRUE, retailers=TRUE)	773

Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

1.6 Additional product characteristics

The *PriceIndices* package includes 10 additional functions providing dataset characteristics. Please note that if the **interval** parameter is set to FALSE, then these functions compare only periods defined by **period1** and **period2** parameters (see documentation). Otherwise the whole time period between period1 and period2 is considered. Table 7 summarizes these functions.

Table 7 Package functions providing dataset characteristics

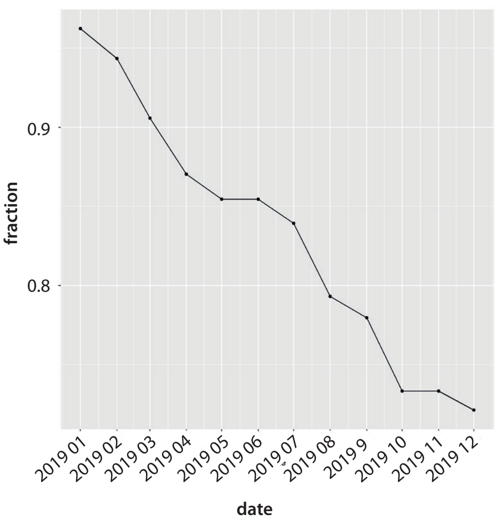
Function	Description
Available	The function returns all values from the indicated column (defined by the type parameter) which occur at least once in one of the compared periods or in a given time interval. Possible values of the type parameter are: **retID, prodID, codeIN, codeOUT or description.**
Matched	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**.
matched_index	The function returns a ratio of values from the indicated column that occur simultaneously in the compared periods or in a given time interval to 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, codeIN, codeOUT or description**. The returned value is from 0 to 1.
matched_fig	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 the **start** and **end** parameters. The returned object (data frame or figure) depends on the value of the **figure** parameter.
prices, quantities, sales	Functions return prices (unit value), quantities and sales (respectively) of products with given IDs (**prodID** column) and being sold in the time period indicated by the **period** parameter. The set parameter means a set of unique product IDs to be used for determining prices of sold products. If the set is empty, the function returns prices of all products being available in **period**.
sales_groups	The function returns **values of sales** of products from one or more **datasets** or the corresponding **barplot** for these sales (if **barplot** is set to TRUE). Alternatively, it calculates the **sale shares** (if **shares** parameter is set to TRUE).
pqcor	The function returns **Pearson's correlation coefficient** for price and quantity of products with given IDs (defined by the **set** parameter) and sold in **period**. If the **set** is empty, the function works for all products being available in **period**. A **figure** parameter indicates whether the function returns a figure with the correlation coefficient (TRUE) or just a correlation coefficient (FALSE).
pqcor_fig	The function returns **Pearson's correlation coefficients** between price and quantity of products with given IDs (defined by the **set** parameter) and sold in the time interval defined by the **start** and **end** parameters. If the **set** is empty, the function works for all available products. Correlation coefficients are calculated for each month separately. Results are presented in tabular or graphical form depending on the **figure** parameter.

Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

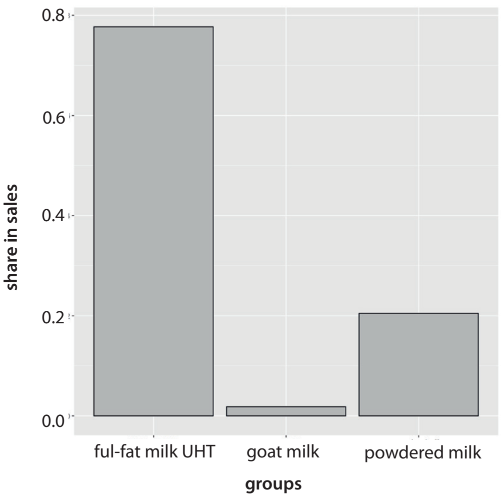
For instance, an example of the use of functions: **matched_fig, sales_groups, pqcor**, and **pqcor_fig** (respectively) for the **milk** dataset is presented in Figure 1.

Figure 1 Graphical results obtained by using functions: **matched_fig, sales_groups, pqcor, and pqcor_fig** for the milk dataset (*)

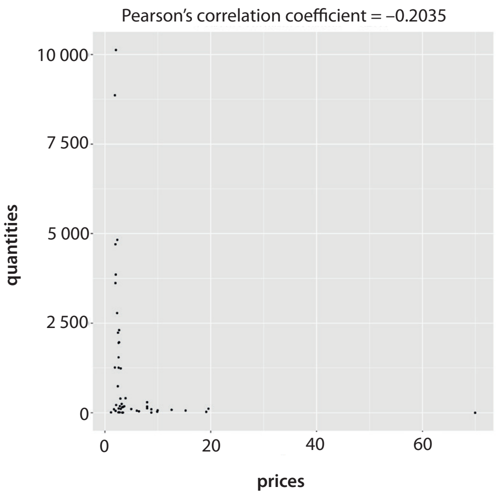
a) The use of the **matched_fig** function



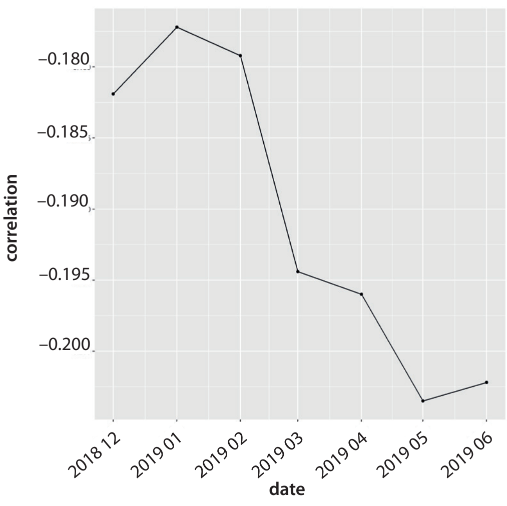
b) The use of the **sales_groups** function



c) The use of the **pqcor** function



d) The use of the **pqcor_fig** function



Note: (*) These figures are results of the following package commands: (a) matched_fig(milk, start="2018-12", end="2019-12", type="prodID"); (b) sales_groups(datasets=list(…), start="2019-04", end="2019-07", barplot=TRUE, shares=TRUE); c) pqcor(milk, period="2019-05", figure=TRUE); d) pqcor_fig(milk, start="2018-12", end="2019-06").
Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

2 BILATERAL PRICE INDEX CALCULATIONS

The *PriceIndices* package includes 6 functions for calculating bilateral, unweighted price indices and 24 functions for calculating bilateral, weighted price indices (see Table 8).

Table 8 Bilateral price indices included in the *PriceIndices* package

Unweighted price indices	
Price index formula	Package function
BMW (2007), Carli (1804), CSWD (1980, 1992), Dutot (1738), Jevons (1865), Harmonic	**bmw, carli, cswd, dutot, jevons, harmonic**
Weighted price indices	
Price index formula	Package function
AG Mean (2009), Banajree (1977), Bialek (2012; 2013), Davies (1924), Drobisch (1871), Fisher (1922), Geary-Khamis (1958, 1970), Geo-Laspeyres, Geo-Lowe, Geo-Paasche, Geo-Young, Laspeyres (1871), Lehr (1885), Lloyd-Moulton (1975, 1996), Lowe, Marshall-Edgeworth (1887), Paasche (1874), Palgrave (1886), Sato-Vartia (1976), Stuvel (1957), Törnqvist (1936), Vartia (1976), Walsh (1901), Young	**agmean, banajree, bialek, davies, drobisch, fisher, geary-khamis, geolaspeyres, geolowe, geopaasche, geoyoung, laspeyres, lehr, lloyd_moulton, lowe, marshall_edgeworth, paasche, palgrave, sato_vartia, stuvel, tornqvist, vartia, walsh, young**

Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

To get the chain version of any price index formula presented in Table 8, we need to add "ch" before its name, e.g. **chfisher** is a function for the chain Fisher index calculation (CPI Manual, 2004). Each of these 60 functions returns a value (or vector of values) of the selected bilateral price index depending on the **interval** parameter. If 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, we should use general functions: **price_index, price_indices or final_index** (see Section 7). None of these functions takes into account aggregating over outlets or product subgroups (to consider these types of aggregating, we need to use the **final_index** function, see Section 8).

For instance, the following command: **lloyd_moulton**(milk,start="2018-12",end="2019 06",sigma=0.7,interval=TRUE) provides values of the Lloyd-Moulton (CES) index for the month series from December 2018 to June 2019, where the fixed base month is December 2018 and the elasticity of the substitution parameter is set to 0.7. As a result, we obtain the following values: 1.0000000, 1.0155974, 1.0039722, 1.0032047, 1.0029064, 0.9943878, 1.0022053.

3 MULTILATERAL PRICE INDEX CALCULATIONS

This package includes 6 functions for calculating multilateral price indices and one additional, general function (**QU**) which calculates the Quality Adjusted Unit Value index (Table 9).

Table 9 Multilateral price indices included in the *PriceIndices* package

Multilateral price index	**Package function**
CCDI (Caves, Christensen, Diewert, 1982)	**ccdi**
GEKS, GEKS-J (GEKS based on the Jevons index), GEKS-W (GEKS based on the Walsh index) (Gini, 1931; Eltetö and Köves, 1964; Szulc, 1983)	**geks, geksj,geksw**
Geary-Khamis (Geary, 1958; Khamis, 1970)	**gk**
Quality Adjusted Unit Value (de Haan, 2004)	**QU**
Time Product Dummy (TPD) (de Haan and Krsinich, 2017)	**tpd**

Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

The above-mentioned 6 multilateral formulas consider the time window defined by the **wstart** and **window** parameters, where **window** is a length of the time window (typically multilateral methods are based on a 13-month time window). It measures the price dynamics by comparing the **end** period to the **start** period (both **start** and **end** must be inside the considered time window). None of these functions takes into account aggregating over outlets or product subgroups (to consider these aggregating, we need to use the **final_index** function, see Section 8). For instance, the following command:

geks(milk,start="2018-12",end="2019-12",window=13) provides the value of the full-window GEKS index comparing December, 2018 to December, 2012 , i.e. a 13-month time window is used. As a result, we obtain 0.9876098.

4 EXTENSIONS OF MULTILATERAL INDICES

4.1 Splicing methods in the *PriceIndices* package

In the case of bilateral methods, a fixed base month (period) is used and the current period is shifted each month. In monthly c{1,2,...,T}hained index methods, the base and the current month are both moved one month. The problem with proceeding with the next month arises in the case of multilateral index methods. Adding information from a new month may influence values of quality adjustment parameters and values of corresponding multilateral indices. In the literature, we can meet the following window updating methods (or splicing methods): a) *movement splice method* (MS), where a price index for the new month is calculated by chaining the month-on-month index for the last month of the shifted window to the index of the previous month (de Haan and van der Grient, 2011); b) *window splice method* (WS), which calculates the price index for the new month by chaining the indices of the shifted window to the index of T months ago (Krsinich, 2014). c) *half splice method* (HS), where the splicing period is chosen to be in the middle of the previous time window (de Haan, 2015); d) *mean splice method* (GMS), which uses the geometric mean of all possible choices of splicing, i.e. all months $\{1,2,...,T\}$ which are included in the current window and the previous one (Diewert and Fox, 2017). All the above-mentioned splicing methods are available in the *PriceIndices* package for any of the discussed multilateral indices (see Section 5). In particular, the following functions can be used: **ccdi_splice, geks_splice, geksj_splice, geksw_splice, gk_splice and tpd_splice**. For instance, let us calculate the extended Time Product Dummy index by using the half splice method with a 10-month time window with the following command (the resulting value is 1.002093):

tpd_splice(milk,start="2018-12",end="2020-02",window=10,splice="half").

4.2. Other extending methods in the *PriceIndices* package

Chessa (2016) proposed a method without using a monthly rolling window. Instead, it uses a time window with a fixed base month every year (December). The window is enlarged every month with one month (*Fixed Base Monthly Expanding Window* – FBEW). Lamboray (2017) proposed a mix of the FBEW method and the *movement splice*. His approach uses a rolling window where the last month of the window is compared to the previous December month. This December plays the role of fixed base, as in the FBEW method. This method is called the *Fixed Base Moving Window* method (FBMW). Both the FBEW and the FBMW methods are available in the package, i.e. to use them for any multilateral price index, we need to add "_fbew" or "_fbmw" to the corresponding index function. For instance, let us calculate the extended TPD index by using the FBEW method using the following command:

tpd_fbew(milk, start="2018-12", end="2020-02"). As a result, we obtain: 0.9977962.

Please note that December 2019 is the chain-linking month here and, following Diewert (2004), the Weighted Least Squares (WLS) method with expenditure shares as weights is used for estimation while calculating the TPD index.

5 GENERAL FUNCTIONS FOR PRICE INDEX CALCULATIONS

This package includes 3 general functions for price index calculation. These functions provide a value or values (depending on the **interval** parameter) of the selected price index formula or formulas. If the **interval** parameter is set to **TRUE**, then it returns a data frame with two columns: **dates** and **index values**. The first two general functions are described as below, the third and the most general function, i.e. **final_index**, is discussed in Section 8:

- **price_index** function.

This function returns a value or values of the selected price index. The **formula** parameter is a character string indicating the price index formula that is to be calculated. If the selected price index formula needs some additional information, it should be defined by additional parameters: **window** and **splice** (connected with multilateral indices), **base** (adequate for the Young and Lowe indices) or **sigma** (for the Lloyd-Moulton or AG mean indices). Table 10 presents an example of the use of the **price_index** function which runs the multilateral Geary-Khamis method for the **milk** dataset, i.e.

price_index(milk,start=”2018-12”,end=”2019-12”,formula=”gk”,interval=TRUE).

Table 10 Results of the use of the **price_index** function in the *PriceIndices* package

Date	gk
2018-12	1.0000000
2019-01	1.0066548
2019-02	1.0008807
2019-03	0.9817312
2019-04	0.9955483
2019-05	0.9918563
2019-06	0.9923588
2019-07	0.9886830
2019-08	1.0001154
2019-09	0.9940940
2019-10	0.9793358
2019-11	0.9779071
2019-12	0.9895014

Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

- **price_index** function.

This is an extended version of the **price_index** function because it allows us to compare many price index formulas by using one command. The general character of this function means that, for instance, one command may calculate two CES indices for two different values of the sigma parameter (the elasticity of substitution) or we can select several splice indices and calculate them by using different window lengths and different splicing methods. Please note that this function is not the most general in the package, i.e. all selected price indices are calculated for the same data set defined by the **data** parameter and the aggregation over subgroups or outlets is not taken into consideration here (to consider it, the **final_index** function should be used – see Section 8). Table 11 presents an example of the use of the **price_indices** function (for the **milk** dataset) which runs the Jevons index, the chain Fisher index, the AG mean index with the elasticity of substitution parameter **sigma**=0.5, the full-window GEKS and CCDI indices and the splicing TPD index, i.e. the TPD index extended by using the *movement splice* method and a 10-month time window, i.e.

```
price_indices(milk,start="2018-12",end="2019-12",
lateral=c("jevons","chfisher"),cesindex=c("agmean"),sigma=c(0.5),
fbmulti=c("geks", "ccdi"),fbwindow=c(13,13),splicemulti=c("tpd_splice"),
splicewindow=c(10),splice=c("movement"), interval=TRUE).
```

Table 11 Results of the use of the **price_indices** function in the *PriceIndices* package

Date	Jevons	Chain Fisher	AG Mean	GEKS	CCDI	Splicing TPD
2018-12	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
2019-01	1.0227271	1.0021874	1.0161907	1.0020440	1.0018258	1.0034052
2019-02	1.0306252	1.0004589	1.0041815	1.0001378	0.9998011	0.9997846
2019-03	1.0361275	0.9861511	1.0040160	0.9837980	0.9839374	0.9830298
2019-04	1.0076198	0.9943142	1.0033451	0.9935624	0.9931984	0.9949289
2019-05	1.0403077	0.9914703	0.9946718	0.9898290	0.9897645	0.9921140
2019-06	0.9850525	0.9897306	1.0027552	0.9889244	0.9887816	0.9907993
2019-07	1.0053768	0.9875189	1.0034281	0.9861619	0.9863439	0.9860539
2019-08	1.0034188	0.9981165	1.0094286	0.9980918	0.9978275	0.9994174
2019-09	1.0181678	0.9968423	1.0085949	0.9951837	0.9951218	0.9944681
2019-10	1.0248130	0.9784270	0.9838821	0.9774534	0.9771381	0.9789089
2019-11	1.0088363	0.9770267	1.0095095	0.9804598	0.9814365	0.9812340
2019-12	1.0255585	0.9873297	1.0000443	0.9876098	0.9875563	0.9901864

Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

6 FINAL INDEX – AGGREGATION OVER SUBGROUPS AND/OR OUTLETS

All previously presented functions for price index calculation do not take into consideration aggregation over subgroups or outlets. The most general package function, i.e. the **final_index**, returns a value or values of the selected price index taking into consideration aggregation over product subgroups and/or over outlets (retailer sale points defined in **retID** column). If this second option is selected, then for each outlet, i.e. each **retID** code, the set of considered **prodID** codes is limited to those codes which are available simultaneously in all considered months. To be more precise: if both types of aggregation are selected, then for each subgroup of products and for each outlet (point of sale) price indices are calculated separately and then aggregated (according to the aggregation methods indicated) to the form of the final price index. If the **interval** parameter is set to TRUE, then it returns a data frame with two columns: dates and final index values (after optional aggregating). The **datasets** parameter defines the user's list of data frames with subgroups of sold products. The **formula** parameter is a character string indicating the price index formula that is to be calculated. If the selected price index formula needs some additional information, it should be defined by additional parameters: **window** and **splice** (connected with multilateral indices), **base** (adequate for the Young and Lowe indices) or **sigma** (for the Lloyd-Moulton or AG mean indices). The **aggrret** parameter is a character string indicating the formula for aggregation over outlets. Available options are: **none**, **laspeyres**, **paasche**, **geolaspeyres**, **geopaasche**, **fisher**, **tornqvist**, **arithmetic**, and **geometric**. The first option means that there is no aggregating over outlets. The last two options mean unweighted methods of aggregating, i.e. the arithmetic or geometric mean is used. Similarly, the **aggrsets** parameter is a character string indicating the formula for aggregation over product subgroups with identical options as previously. To demonstrate the use of the **final_index** function, let us define four subgroups of milk:

```
g1<-dplyr::filter(milk, milk$description=="powdered milk"),
g2<-dplyr::filter(milk, milk$description=="full-fat milk UHT"),
g3<-dplyr::filter(milk, milk$description=="low-fat milk UHT"),
g4<-dplyr::filter(milk, milk$description=="goat milk").
```

Now, for the fixed time interval: December 2018–May 2019 using the **milk** dataset, let us calculate the (final) chain Fisher price index (the fixed base month is December 2018) taking into consideration the Laspeyres aggregation over subgroups **g1**, **g2, g3, g4** and the Törnqvist aggregation over outlets. The appropriate package command is as follows:

```
final_index(datasets=list(g1,g2,g3,g4), start="2018-12", end="2019-05",
formula="chfisher",aggrsets="laspeyres",aggrret="tornqvist",interval=TRUE),
```

and resulting final price index values are presented in Table 12.

Table 12 The final chain Fisher index calculated for **milk** and with aggregation over subgroups and outlets

Date	Chain Fisher
2018-12	1.0000000
2019-01	1.0021740
2019-02	1.0077480
2019-03	1.0129532
2019-04	1.0089259
2019-05	0.9960455

Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

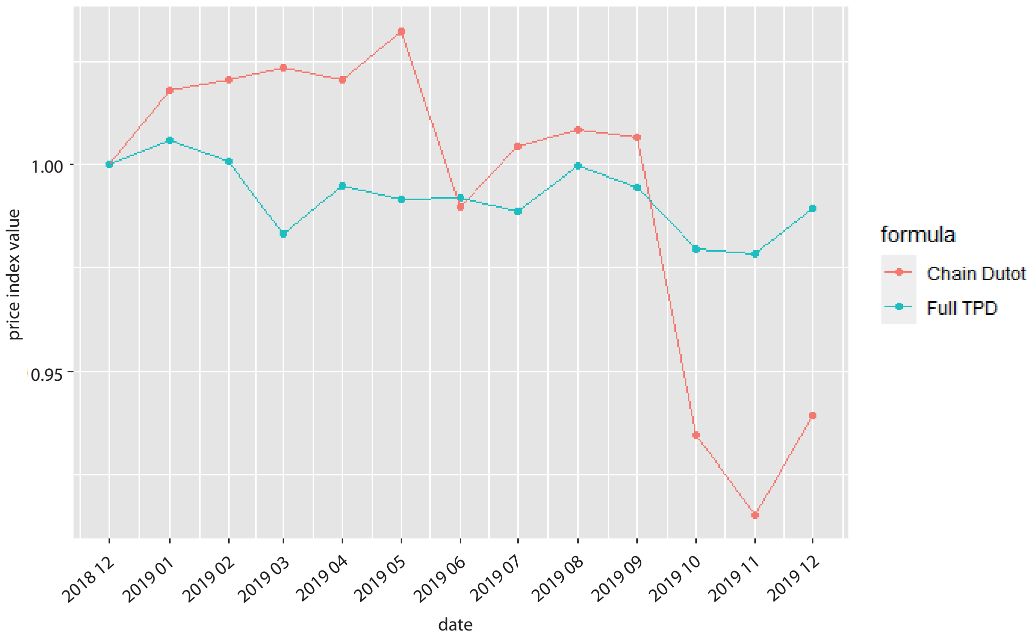
7 GRAPHICAL COMPARISON OF PRICE INDEX RESULTS

This package includes 2 functions for simple graphical comparison of price indices. The first one, i.e. **compare_indices**, is based on the syntax of the **price_indices** function, and thus it allows us to compare price indices calculated on the same data set. This function calculates selected bilateral or/and multilateral price indices and returns a figure with plots of these indices (together with dates on the X-axis and the corresponding legend). The function does not take into account aggregating over outlets or product subgroups. For instance, let us compare the price dynamics for the milk dataset for the time interval: December 2018–December 2019, calculated by using two price index formulas: the chain Dutot index and the full-window TPD index. The above-mentioned comparison can be made by the following command:

```
compare_indices(milk,start="2018-12",end="2019-12",
bilateral=c("chdutot"),fbmulti=c("tpd"),fbwindow=c(13),
namebilateral=c("Chain Dutot"), namefbmulti=c("Full TPD")),
```

and its result is presented in Figure 2.

Figure 2 An example of the use of the **compare_indices** function



Source: PriceIndices R package *<https://CRAN.R-project.org/package=PriceIndices>*

The second function, i.e. **compare_final_indices**, has a general character since its first argument is a list of data frames which contain results obtained by using the **price_index** or **final_index** functions. To be more precise: the **finalindices** parameter is 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 and the second column must indicate price index values for the corresponding dates. The above-mentioned single data frame may be created manually in the previous step or it may be a result of the following functions: price_index or final_index. All considered data frames must have an identical number of rows. The names parameter is a vector of character strings describing names of presented indices. For instance, let us compare the impact of the aggregating over outlets and subgroups on the price index results (e.g. the Laspeyres formula is the assumed aggregating method). For this purpose, let us calculate the full-window **GEKS index** in two cases: **case1** without the above-mentioned aggregation and **case2** which considers that aggregation. We use the **milk** dataset and the yearly time interval:

case1<-**price_index**(milk, start="2018-12",end="2019-12",formula="geks", interval=TRUE),

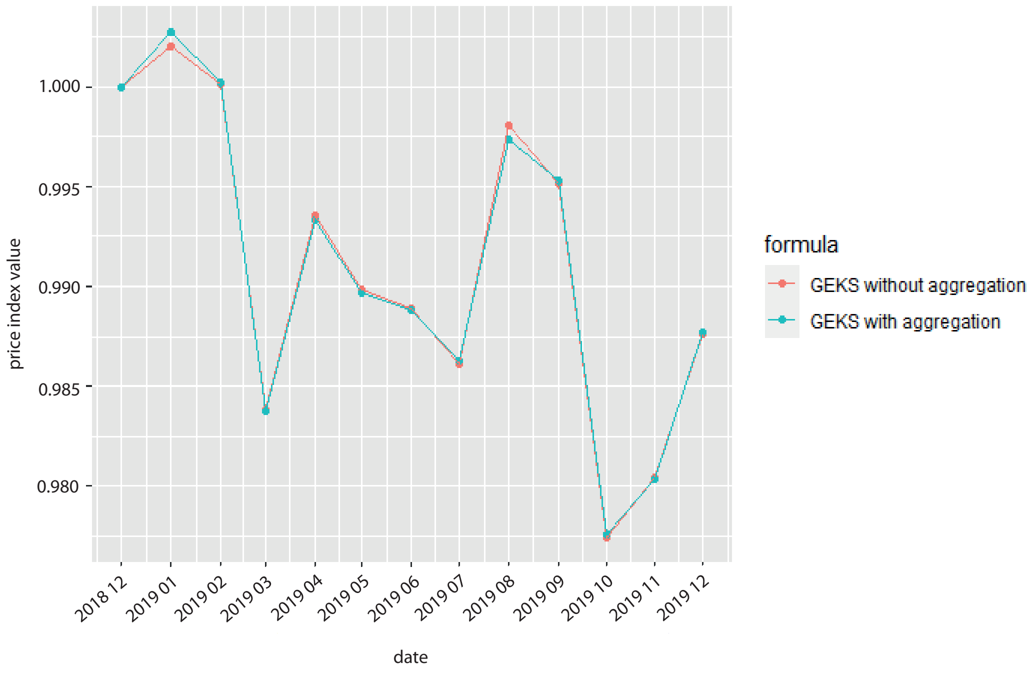
case2<-**final_index**(datasets=list(milk), start="2018-12", end="2019-12", formula="geks", aggrsets="laspeyres", aggrret = "laspeyres", interval=TRUE),

compare_final_indices(finalindices=list(case1, case2),

names=c("GEKS without aggregation", "GEKS with aggregation")).

The results of the above-mentioned comparison are presented in Figure 3. Differences between the calculated GEKS indices are negligible in the case of milk.

Figure 3 An example of the use of the **compare_final_indices** function



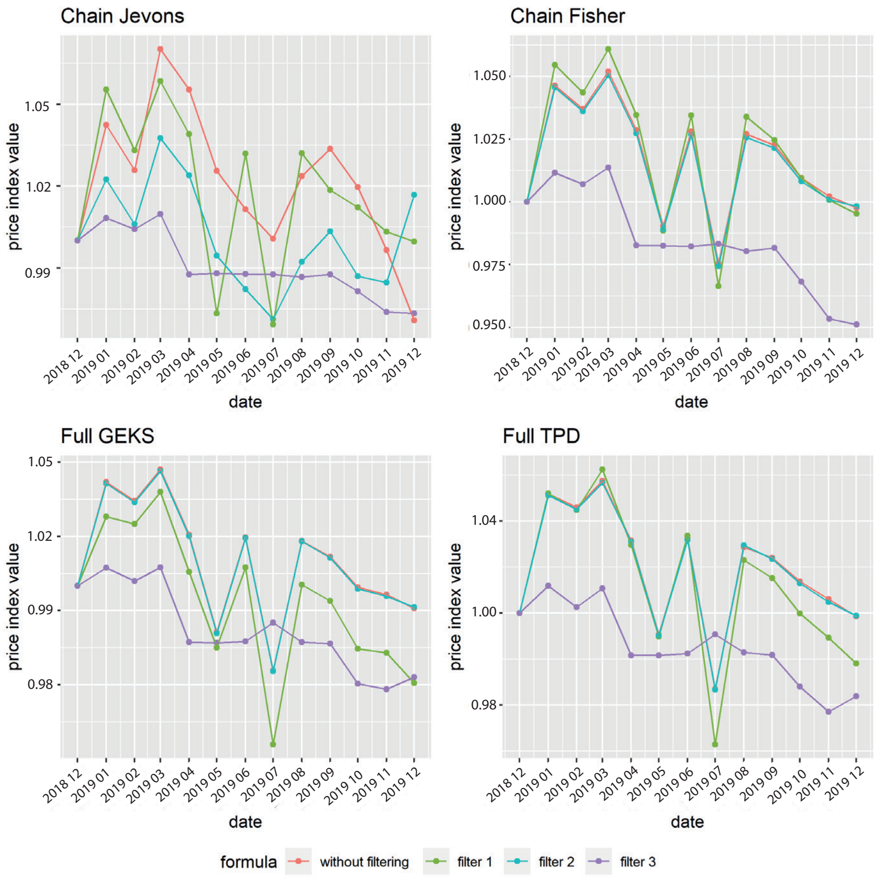
8 AN EXAMPLE OF AN EMPIRICAL STUDY USING THE *PRICEINDICES* PACKAGE

This section examines the influence of the choice of the data filter type on the final price index result. Discussion about whether and what types of data filters to use for scanner data is ongoing in the literature. As a rule, scanner data indices are calculated using a dynamic approach and most countries use the monthly chained Jevons index. This method is commonly named the dynamic method (Eurostat, 2017). The dynamic basket is determined using turnover figures of individual products in two adjacent months, i.e. the product is included in the sample if its turnover is above a fixed threshold being determined by the number of products in the considered product group. In the literature (Van Loon and Roels, 2018), we can meet the following condition for the above-mentioned rule which indicates whether the *i*-th product is taken into consideration while comparing months $t-1$ and t:

$$\frac{s_i^{t-1}+s_i^{t}}{2} > \frac{1}{n\lambda}\,, \tag{1}$$

where s_i^τ denotes the expenditure share of the *i*-th product at time τ, n is the number of considered products and λ is a fixed parameter (as a rule 1.25). We will call this kind of data filter the *low sale filter*. Supporters of using filters also assume that products that show extreme pricing changes from one month to another should be also excluded from the sample (*extreme price filter*). The list of possible data filters is longer, e.g. Statistics Belgium implements a filter for *dump prices* (van Loon and Roels, 2018). Filtering products is one thing and the decision to proceed with them is another. One possible option is the imputation of prices being flagged by filters but this raises questions about how to impute them. Another option is to remove flagged products from the sample if it does not change the sample size critically. This section examines the impact of the use of *low sale filter* and *extreme price filter* on price index values in the latter option. In the empirical illustration, we use the scanner data set (**milk.RData**) from one of retail chains in Poland, i.e. monthly data from 4 outlets on low-fat UHT milk (COICOP: 01142) sold during the period: Dec. 2018–Dec. 2019. We consider the *low sale filter* with $\lambda = 1.25$ (**filter1**) and the *extreme price filter* with thresholds for the minimum and maximum price change set to 50% and 200% (**filter2**). We use another variant of the *extreme price filter*, namely the filter which rolls out products with price changes (in compared months) being smaller and bigger than the 1th and 99th quantile of all observed price changes (**filter3**). We investigate the impact of the above-mentioned filters on the final price index results where the considered final index formulas are: the chain Jevons, the chain Fisher, the full-window GEKS and the full-window TPD indices. While calculating the final index, we consider aggregation over outlets, i.e. the aggregation is done by using the Törnqvist formula. Please note that filtering is done for each outlet separately, too. The whole procedure of the empirical study can be found in the Appendix. The obtained results are presented in Figure 4.

Figure 4 Impact of the *low sale filter* and *extreme price filters* on the final index value based on the example of **low-fat UHT milk** sold during the period: Dec. 2018–Dec. 2019



Source: PriceIndices R package <*https://CRAN.R-project.org/package=PriceIndices*>

The unweighted Jevons price index formula seems to be the most sensitive to the choice of the data filter. The quantile filter, i.e. the filter 3, seems to influence considered price indices in the same way, namely: as a rule, its use leads to the smallest values of final indices (e.g. the exception month is July, 2019). Presumably, this means that the distribution of monthly price changes is left-sided asymmetrical, i.e. products with larger monthly price changes than the average monthly price change dominate. In the case of weighted formulas (the Fisher, GEKS, TPD), the impact of filter 2 on the price index value is rather negligible. It can be easily justified since we do not observe any extreme price changes in the case of milk observed during the considered time interval. Moreover, differences in price changes between milk products with the biggest and the smallest sale values are probably small but not negligible, i.e. there is a substantial impact of the filter 1 (green line) on the index value observed especially for the chain Jevons and the full-window GEKS indices. In their case, the difference between the index values obtained with and without filter 1 may exceed 2–3 p.p.

CONCLUSIONS AND FINAL REMARKS

The presented R package, i.e. PriceIndices package, was created for both official price statistics and statisticians dealing with theory of price indices. The main assumptions when creating the package were such that, firstly, it would cover the largest set of price indices and methods, and secondly, would be as flexible as possible in controlling parameters related to price index formulas. The author of the package hopes that this is the case: the package contains 104 useful functions for data processing, price index calculations and price index comparisons, including 6 functions for calculating bilateral unweighted price indices, 24 functions for calculating bilateral weighted price indices, 30 functions for calculating weighted and unweighted chain price indices and 6 functions for calculating multilateral price indices. Moreover, the package allows us to extend multilateral price indices by using known and updated methods, i.e. splicing, the FBEW and FBMW methods. According to the best author's knowledge, this distinguishes the PriceIndices package from other packages dedicated to price indices. The package can be useful at any stage of dealing with scanner data: it allows for preliminary classification of products by labels, their matching and data filtering. Users (statisticians, NSIs, others) who have appropriate data frames prepared in the package or earlier by themselves have a variety of price indices and methods at their disposal, which makes it possible to conduct extensive experiments. The package, as part of the R environment, is free of charge and has the ability to be expanded, also by users. The presented version of the package (ver. 1.0) is its first installment (please use *install.packages("PriceIndices")* to install it) and its author expresses the hope that users will contribute to its extension with further useful functions or will improve its speed as well as reliability and find possible errors. The author of the package thanks in advance all users for any comments. It is also planned to constantly expand the package with methods and index formulas that will appear in the literature on an ongoing basis.

References

ABE, N. AND PRASADA RAO, D. S. Multilateral Sato-Vartia index for international comparison of prices and real expenditures. *Economic Letters*, 2019, 183(C), pp. 1–4.

BANAJREE, K. S. *On the factorial approach providing the true index of cost of living*. Göttingen: Vandenhoeck und Ruprecht, 1977.

BIALEK, J. Some Remarks on the Original Price Index Inspired by the Notes of Peter von der Lippe. *Econometrics (Ekonometria),* 2013, 3(41), pp. 40–54.

BIALEK, J. Simulation Study of an Original Price Index Formula. *Communications in Statistics – Simulation and Computation*, 2014, 43(2), pp. 285–297.

BIAŁEK, J. AND BOBEL, A. *Comparison of Price Index Methods for CPI Measurement using Scanner Data*. Paper presented at the 16th Meeting of the Ottawa Group on Price Indices, Rio de Janeiro, Brazil, 2019.

CARLI, G. Del valore e della proporzione de'metalli monetati. In: *Scrittori Classici Italiani di Economia Politica*, 1804, 13, pp. 297–336.

CARRUTHERS, A. G., SELLWOOD, D. J., WARD, P. W. Recent developments in the retail price index. *Journal of the Royal Statistical Society. Series D (The Statisticain)*, 1980, 29(1), pp. 1–32.

CAVES, D. W., CHRISTENSEN, L. R., DIEWERT, W. E. Multilateral comparisons of output, input, and productivity using superlative index numbers. *Economic Journal,* 1982, 92, pp. 73–86.

CHESSA, A. G. A New Methodology for Processing Scanner Data in the Dutch CPI. *Eurona*, 1/2016, pp. 49–69.

CHESSA, A. G. AND GRIFFIOEN, R. *Comparing Scanner Data and Web Scraped Data for Consumer Price Indices*. Report, Statistics Netherlands, 2016.

CHESSA, A. G. *Comparisons of QU-GK Indices for Different Lengths of the Time Window and Updating Methods*. Paper prepared for the 2nd meeting on multilateral methods organised by Eurostat, Luxembourg, 14–15 March, 2017, Statistics Netherlands.

CHESSA, A. G., VERBURG, J., WILLENBORG, L. *A comparison of price index methods for scanner data*. Paper presented at the 15th Meeting of the Ottawa Group on Price Indices, Eltville am Rhein, Germany, 10–12 May, 2017.

Consumer Price Index Manual. Theory and practice. Geneva: ILO, IMF, OECD, UNECE, Eurostat, The World Bank, International Labour Office (ILO), 2004.

DALEN, J. Recent developments in the retail price index. *The Statistician*, 1992, 29(1), pp. 1–32.

DAVIES, G. R. The Problem of a Standard Index Number Formula. *Journal of the American Statistical Association,* 1924, 19(146), pp. 180–188.

DIEWERT, W. E. *On the Stochastic Approach to Linking the Regions in the ICP*. Discussion Paper 04–16, Vancouver, Canada: Department of Economics, University of British Columbia, 2004.

DIEWERT, W. E., FOX, K. J. *Substitution Bias in Multilateral Methods for CPI Construction using Scanner Data*. Discussion paper 17–02, Vencouver, Canada: Vancouver School of Economics, University of British Columbia, 2017.

DE HAAN, J. *Estimating Quality-Adjusted Unit Value Indexes: Evidence from Scanner Data*. Paper presented at the SSHRC International Conference on Index Number Theory and the Measurement of Prices and Productivity, 30 June–3 July, 2004, Vancouver, Canada.

DE HAAN, J. AND VAN DER GRIENT, H. A. Eliminating chain drift in price indexes based on scanner data. *Journal of Econometrics*, 2011, 161, pp. 36–46.

DE HAAN, J. *A Framework for Large Scale Use of Scanner Data in the Dutch CPI.* Paper presented at the 14th Ottawa Group meeting, Tokyo, Japan, 2015.

DE HAAN, J., WILLENBORG, L., CHESSA, A. G. *An Overview of Price Index Methods for Scanner Data*. Paper presented at the Meeting of the Group of Experts on Consumer Price Indices, 2–4 May, 2016, Geneva, Switzerland.

DE HAAN, J. AND KRSINICH, F. Time Dummy Hedonic and Quality-Adjusted Unit Value Indices: Do They Really Differ? *Review of Income and Wealth*, 2017, 64(4), pp. 757–776.

DUTOT, C. F. *Reflexions Politiques sur les Finances et le Commerce.* The Hague: Les Freres Vaillant et Nicolas Prevost, 1738, Vol. 1.

DROBISCH M. W. Ueber einige Einwürfe gegen die in diesen Jahrbüchern veröffentlichte neue Methode, die Veränderungen der Waarenpreise und des Geldwerths zu berechten. *Jahrbücher für Nationalökonomie und Statistik*, 1871, Vol. 16, pp. 416–427.

EDGEWORTH, F. Y. Measurement of Change in Value of Money I. The first Memorandum presented to the British Association for the Advancement of Science, reprinted in *Papers Relating to Political Economy*, 1887, Vol. 1, New York, Burt Franklin, p. 1925.

ELTETÖ, Ö. AND KÖVES, P. On a Problem of Index Number Computation Relating to International Comparisons (in Hungarian). *Statisztikai Szemle*, 1964, 42, pp. 507–518.

EUROSTAT. Practical Guide for Processing Supermarket Scanner Data. In: *Harmonised Index of Consumer Prices*, 2017.

FISHER I. *The Making of Index Numbers*. Boston: Houghton Mifflin, 1922.

GEARY, R. G. A Note on Comparisons of Exchange Rates and Purchasing Power between Countries. *Journal of the Royal Statistical Society Series A*, 1958, 121, pp. 97–99.

GINI, C. On the Circular Test of Index Numbers. *Metron,* 1931, 9(9), pp. 3–24.

GRIFFIOEN, A. R. AND BOSCH, O. *On the Use of Internet Data for the Dutch CPI*. Paper presented at the UNECE-ILO Meeting of the Group of Experts on Consumer Price Indices, 2–4 May, 2016, Geneva, Switzerland.

IVANCIC, L., DIEWERT, W. E., FOX, K. J. Scanner Data, Time Aggregation and the Construction of Price Indices. *Journal of Econometrics*, 2011, 161(1), pp. 24–35.

JEVONS, W. S. The variation of prices and the value of the currency since 1782. *J. Statist. Soc. Lond.*, 1865, 28, pp. 294–320.

KHAMIS, S. H. A New System of Index Numbers for National and International Purposes. *Journal of the Royal Statistical Society Series A*, 1972, 135, pp. 96–121.

KRSINICH, F. *The FEWS Index: Fixed Effects with a Window Splice – Non-Revisable Quality-Adjusted Price Indices with No Characteristic Information.* Paper presented at the meeting of the group of experts on consumer price indices, 26–28 May, 2014, Geneva, Switzerland.

LAMBORAY, C. *The Geary Khamis index and the Lehr index: how much do they differ?* Paper presented at the 15th Ottawa Group meeting, 10–12 May, 2017, Elville am Rhein, Germany.

LASPEYRES, E. Die Berechnung einer mittleren Waarenpreissteigerung, *Jahrbücher für Nationalökonomie und Statistik*, 1871, 16, pp. 296–314.

LENT, J. AND DORFMAN, A. H. Using a Weighted Average of Base Period Price Indexes to Approximate a Superlative Index. *Journal of Official Statistics,* 2009, 25(1), pp. 139–149.

LEHR, J. *Beiträge zur Statistik der Preise, insbesondere des Geldes und des Holzes*. Frankfurt am Main: J. D. Sauerländer, 1885.

LLOYD, P. J. Substitution Effects and Biases in Non True Price Indices. *The American Economic Review*, 1975, 65, pp. 301–313.

MARSHALL, A. Remedies for Fluctuations of General Prices. *Contemporary Review*, 1887, 51, pp. 355–375.

MEHRHOFF, J. A linear approximation to the Jevons index. In: VON DER LIPPE. *Index Theory and Price Statistics*, Berlin, Germany: Peter Lang, 2007.

MEHRHOFF, J. *Towards a new paradigm for scanner data price indices: applying big data techniques to big data.* Paper presented at the 16th Meeting of the Ottawa Group on Price Indices, Rio de Janeiro, Brazil, 2019.

MOULTON, B. R. *Constant Elasticity Cost-of-Living Index in Share-Relative Form*. Washington DC: U. S. Bureau of Labour Statistics, 1996.

PAASCHE, H. Über die Preisentwicklung der letzten Jahre nach den Hamburger Borsennotirungen. *Jahrbücher für Nationalökonomie und Statistik*, 1874, 12, pp. 168–178.

PALGRAVE, R. H. I. *Currency and Standard of Value in England, France and India and the Rates of Exchange Between these Countries*. Memorandum submitted to the Royal Commission on Depression of trade and Industry, Third Report, 1886, Appendix B, pp. 312–390.

SATO, K. *The Ideal Log-Change Index Number. The Review of Economics and Statistics,* 1976, 58(2), pp. 223–228.

STUVEL, G. A New Index Number Formula. *Econometrica*, 1957, 25, pp. 123-131.

SZULC, B. Indices for Multiregional Comparisons (in Polish). *Przegląd Statystyczny*, 1964, 3, pp. 239–254.

TÖRNQVIST, L. The Bank of Finland's Consumption Price Index. *Bank of Finland Monthly Bulletin,* 1936, 10, pp. 1–8.

VAN LOON, K. V. AND ROELS, D. *Integrating big data in the Belgian CPI.* Paper presented at the meeting of the group of experts on consumer price indices, 8–9 May, 2018, Geneva, Switzerland.

VARTIA, Y. 0. *Ideal Log-Change Index Numbers. Scandinavian Journal of Statistics*, 1976, 3(3), pp. 121–126.

VON AUER, L. *The Nature of Chain Drift.* Paper presented at the 17th Meeting of the Ottawa Group on Price Indices, 8–10 May, 2019, Rio de Janerio, Brasil.

WALSH, C. M. *The Measurement of General Exchange Value*. New York: The MacMillan Company, 1901.

WEBSTER, M. AND TARNOW-MORDI, R. C. Decomposing Multilateral Price Indexes into the Contribution of Individual Commodities. *Journal of Official Statistics*, 2019, 35(2), pp. 461–486.

ZHANG, L., JOHANSEN, I., NYAGAARD, R. Tests for Price Indices in a Dynamic Item Universe. *Journal of Official Statistics*, 2019, 35(3), pp. 683–697.

APPENDIX

R procedure for the empirical study with the use of the *PriceIndices* package

```
library(“PriceIndices”)

library(“ggpubr”)

#time interval

t1<-”2018-12”

t2<-”2019-12”

#milk subgroup

data<-dplyr::filter(milk, milk$description==”low-fat milk UHT”)

#filters

filter1<-data_filtering(data,   start=t1,   end=t2,   filters=c(“lowsales”),
lambda=1.25, interval=TRUE, retailers=TRUE)

filter2<-data_filtering(data,  start=t1,  end=t2,  filters=c(“extremeprices”),
plimits=c(0.5,2), interval=TRUE, retailers=TRUE)

filter3<-data_filtering(data,  start=t1,  end=t2,  filters=c(“extremeprices”),
pquantiles=c(0.01,0.99), interval=TRUE, retailers=TRUE)

#Chain Jevons

CHJ_no<-final_index(datasets=list(data),start=t1,end=t2,formula=”chjevons”,
aggrsets = “none”, aggrret = “tornqvist”, interval=TRUE)

CHJ_f1<-final_index(datasets=list(filter1),start=t1,                 end=t2,
formula=”chjevons”, aggrsets = “none”, aggrret = “tornqvist”, interval=TRUE)

CHJ_f2<-final_index(datasets=list(filter2),start=t1,                 end=t2,
formula=”chjevons”, aggrsets = “none”, aggrret = “tornqvist”, interval=TRUE)

CHJ_f3<-final_index(datasets=list(filter3),start=t1,                 end=t2,
formula=”chjevons”, aggrsets = “none”, aggrret = “tornqvist”, interval=TRUE)

fig1<-compare_final_indices(finalindices=list(CHJ_no,CHJ_f1,CHJ_f2,CHJ_f3),
names=c(“without filtering”, “filter 1”, “filter 2”, “filter 3”))

fig1<-fig1+ggtitle(“Chain Jevons”)

#Chain Fisher

CHF_no<-final_index(datasets=list(data),start=t1, end=t2, formula=”chfisher”,
aggrsets = “none”, aggrret = “tornqvist”, interval=TRUE)

CHF_f1<-final_index(datasets=list(filter1),start=t1,                 end=t2,
formula=”chfisher”, aggrsets = “none”, aggrret = “tornqvist”, interval=TRUE)

CHF_f2<-final_index(datasets=list(filter2),start=t1,                 end=t2,
formula=”chfisher”, aggrsets = “none”, aggrret = “tornqvist”, interval=TRUE)

CHF_f3<-final_index(datasets=list(filter3),start=t1,                 end=t2,
formula="chfisher", aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

fig2<-compare_final_indices(finalindices=list(CHF_no,CHF_f1,CHF_f2,CHF_f3),
names=c("without filtering", "filter 1", "filter 2", "filter 3"))

fig2<-fig2+ggtitle("Chain Fisher")

#GEKS

G_no<-final_index(datasets=list(data),start=t1,   end=t2,   formula="geks",
aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

G_f1<-final_index(datasets=list(filter1),start=t1,end=t2,    formula="geks",
aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

G_f2<-final_index(datasets=list(filter2),start=t1,end=t2,    formula="geks",
aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

G_f3<-final_index(datasets=list(filter3),start=t1,end=t2,    formula="geks",
aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

fig3<-compare_final_indices(finalindices=list(G_no,G_f1,G_f2,G_f3),
names=c("without filtering", "filter 1", "filter 2", "filter 3"))

fig3<-fig3+ggtitle("Full GEKS")

#TPD

TPD_no<-final_index(datasets=list(data),start=t1, end=t2, formula="tpd",

aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

TPD_f1<-final_index(datasets=list(filter1),start=t1,  end=t2,  formula="tpd",
aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

TPD_f2<-final_index(datasets=list(filter2),start=t1,  end=t2,  formula="tpd",
aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

TPD_f3<-final_index(datasets=list(filter3),start=t1,  end=t2,  formula="tpd",
aggrsets = "none", aggrret = "tornqvist", interval=TRUE)

fig4<-compare_final_indices(finalindices=list(TPD_no,TPD_f1,TPD_f2,TPD_f3),
names=c("without filtering", "filter 1", "filter 2", "filter 3"))

fig4<-fig4+ggtitle("Full TPD")

#results

figure <- ggarrange(fig1, fig2, fig3, fig4,

                    common.legend = TRUE,

                    legend=c("bottom"),

                    ncol = 2, nrow = 2)

figure
ggexport(figure, filename = "result.png")

#end of procedure
```