SLOVENSKÁ ŠTATISTIKA a DEMOGRAFIA

SLOVAK STATISTICS and DEMOGRAPHY

3/2023
ročník/volume 33

Recenzovaný vedecký časopis so zameraním na prezentáciu moderných štatistických a demografických metód a postupov.

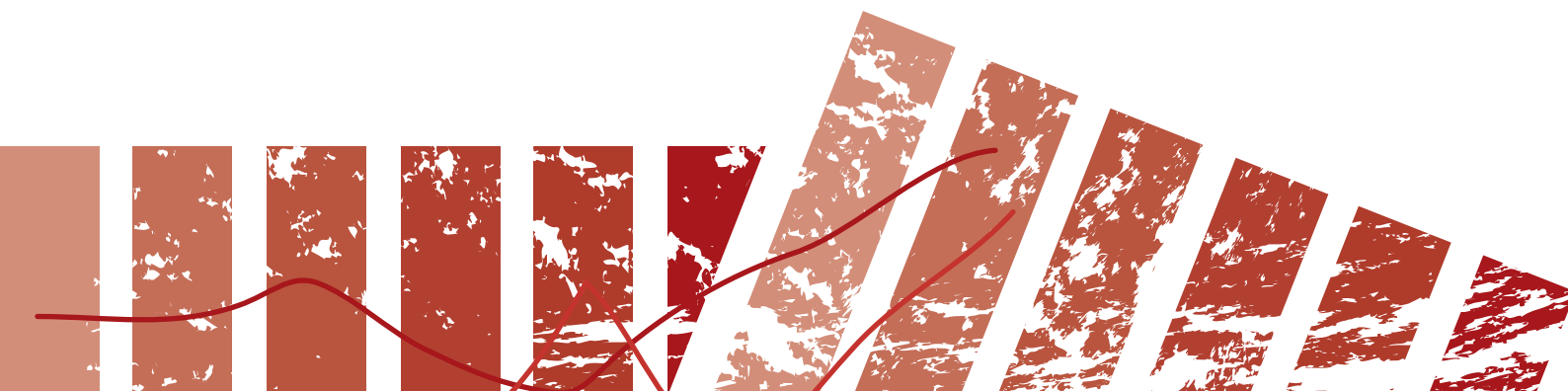
Scientific peer-reviewed journal focusing on the presentation of modern statistical and demographic methods and procedures.

Článok/Article: 2

Typ článku/Type of article: vedecký článok/scientific article

Strany/Pages: 7 – 20

Dátum vydania/Publication date: 15. júl 2023/July 15, 2023



Jacek BIAŁEK
Statistics Poland, Department of Trade and Services
University of Lodz, Department of Statistical Methods

SCANNER DATA PROCESSING AND PRICE INDEX CALCULATIONS IN THE PRICEINDICES R PACKAGE

SPRACOVANIE ÚDAJOV ZO SKENEROV A VÝPOČET CENOVÝCH INDEXOV V R BALÍKU CENOVÝCH INDEXOV

ABSTRACT
The main purpose of the work is to present the utility of the *PriceIndices* R-package in the field of analysing the dynamics of scanner prices. The package can be a useful IT tool for National Statistical Offices for at least several reasons. First, it enables a comprehensive transition from raw scanner data to price indices, taking into account all intermediate steps (e.g., product classification or product matching). Secondly, the package's functions, through their parameterization, allow experimental work to be carried out, such as setting thresholds for data filters or the length of the time window for multilateral indices. Third, the package is written in the free R environment, which does not generate any additional costs for its implementation.
The presentation of this R package is divided into the following areas: scanner data preparing, data set characteristics, bilateral index calculations, multilateral index calculations, extensions of multilateral indices, aggregation of index results, and comparison of price indices. The paper presents examples concerning main package features on the basis of data sets which are available in the *PriceIndices* package.

ABSTRAKT
Hlavným cieľom práce je predstaviť využitie balíka R cenových indexov v oblasti analýzy dynamiky scanner data (transakčné dáta nazývané aj údaje zo skenerov). Balík môže byť užitočným IT nástrojom pre národné štatistické úrady minimálne z niekoľkých dôvodov. Po prvé, umožňuje komplexný prechod od nespracovaných údajov zo skenerov k cenovým indexom, pričom zohľadňuje všetky čiastkové etapy (napr. klasifikáciu produktov alebo párovanie produktov). Po druhé, funkcie balíka prostredníctvom ich parametrizácie umožňujú vykonávať experimentálne práce, ako je nastavenie prahov pre dátové filtre alebo dĺžky intervalu (časového okna) pre multilaterálne indexy. Po tretie, balík je vytvorený vo voľnom prostredí R, čo nevytvára žiadne dodatočné náklady na jeho implementáciu.
Prezentácia balíka R je rozdelená do nasledujúcich oblastí: príprava skenerových dát, charakteristiky dátových súborov, výpočty bilaterálnych indexov, výpočty multilaterálnych indexov, rozšírenia multilaterálnych indexov, agregácia výsledkov indexov a porovnávanie cenových indexov. Článok prezentuje príklady týkajúce sa hlavných funkcií balíka na základe dátových súborov, ktoré sú dostupné v balíku cenových indexov.

KEY WORDS
scanner data, PriceIndices package, price indices, multilateral indices

KĽÚČOVÉ SLOVÁ
údaje zo skenerov, balík Cenových indexov, cenové indexy, multilaterálne indexy

1. INTRODUCTION

The term “scanner data” refers to transaction data that specify turnover and numbers of items sold by Global Trade Article Number (GTIN) code, European Article Number (EAN) code or other barcodes [10]. Scanner data have numerous advantages compared to traditional survey data: such data sets are much larger than the traditional ones and contain complete transaction records, i.e. information about prices and quantities along with the additional information about products, including the grammage, unit, label with description, VAT, etc. In other words, scanner data contain full information at the most detailed item level. The methodology for inflation measurement by using scanner data has strongly evolved over the last few years (see for instance: [1, 3, 5, 6, 7, 12, 13, 14, 20, 21, 23]).

Despite many benefits of using scanner data, their use is still associated with many methodological challenges and problems. The main challenge on the IT side is to build the right software that efficiently and quickly enables the transition from raw scanner data to price indices. An example of such software is the *PriceIndices* package written in the R environment [2] and available on CRAN and GitHub. Although some packages dedicated to scanner data and price indices are available in the R environment (e.g. *IndexNumR* by Graham White [10], *IndexNumber* package by Alejandro Saavedra-Nieves and Paula Saavedra-Nieves [15] or *multilateral* by Matthew Stansfield [16]), their functionalities are much lower compared to the *PriceIndices* package and they do not operate on a time variable in the sense of year-month-day. The PriceIndices package contains a rich set of functions for the proceding scanner data and price index functions (over 200 functions), which sets it apart from the other R packages in this area. As it was mentioned above, the package can be a useful IT tool for National Statistical Offices because: (1) it enables a comprehensive transition from raw scanner data to price indices, taking into account all intermediate steps (e.g., product classification or product matching); (2) the package's functions, through their parameterization, allow experimental work to be carried out, such as setting thresholds for data filters or the length of the time window for multilateral indices; (3) the package is written in the free R environment, which does not generate any additional costs for its implementation. The package has an open-source code so it can be modified and extended by the Statistical Office.

The main purpose of the paper is to present the utility of the *PriceIndices* R package, which can be divided into the following areas: scanner data preparing, data set characteristics, price index calculations, extensions of multilateral indices, aggregation of index results, as well as comparison of price indices [9]. The paper presents examples concerning main package features on the basis of data sets which are available in the *PriceIndices* package (e.g. *milk*, *sugar* or *coffee* datasets). As a result, the reader will have the opportunity to perform all the examples presented on their own.

The structure of the paper is as follows: Section 2 presents the main package features for scanner data preparing, including functions for classifying and matching products, data filtering and imputing missing prices; Section 3 shows how to obtain some statistical characteristics of the analysed scanner products, Section 4 describes price index methods which are available in *PriceIndices*, Section 5 discusses the aggregation methods; Section 6 shows some ways of presenting and comparing price indices and Section 7 provides the main conclusions. The attached Appendix contains

reproducible R-language codes using the PriceIndices package and its included scanner data sets.

2. SCANNER DATA PREPARING

In general, the procedure of scanner data processing consists of the following steps: data cleaning ► extracting information from the product description ► classification of products into COICOP (Classification of individual consumption by purpose) groups ► product matching ► product filtering ► data standardization ► imputing missing prices (optional) ► price index calculations. All these procedures can be performed in the *PriceIndices* package. Initial cleaning of the data set, i.e. removal of data gaps, removal of zero prices and quantities and standardization of variable types, can be performed using the **data_preparing** function. Sometimes a retail chain supplies scanner data frames with additional columns specifying the basis weight (grammage) and the sales unit of the products. This information, however, can be sometimes hidden in the product label and then needs to be extracted. The **data_unit** function returns the user's data frame (provided as a data frame by *data* parameter) 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: '2x50g', we will obtain: *grammage* = 100 and *unit* = g for that product (for the *multiplication* parameter set to 'x'). To get a good understanding of the function for extracting information from a product label, please run **Example 1** from the **Appendix.**

2.1 PRODUCT CLASSIFICATION

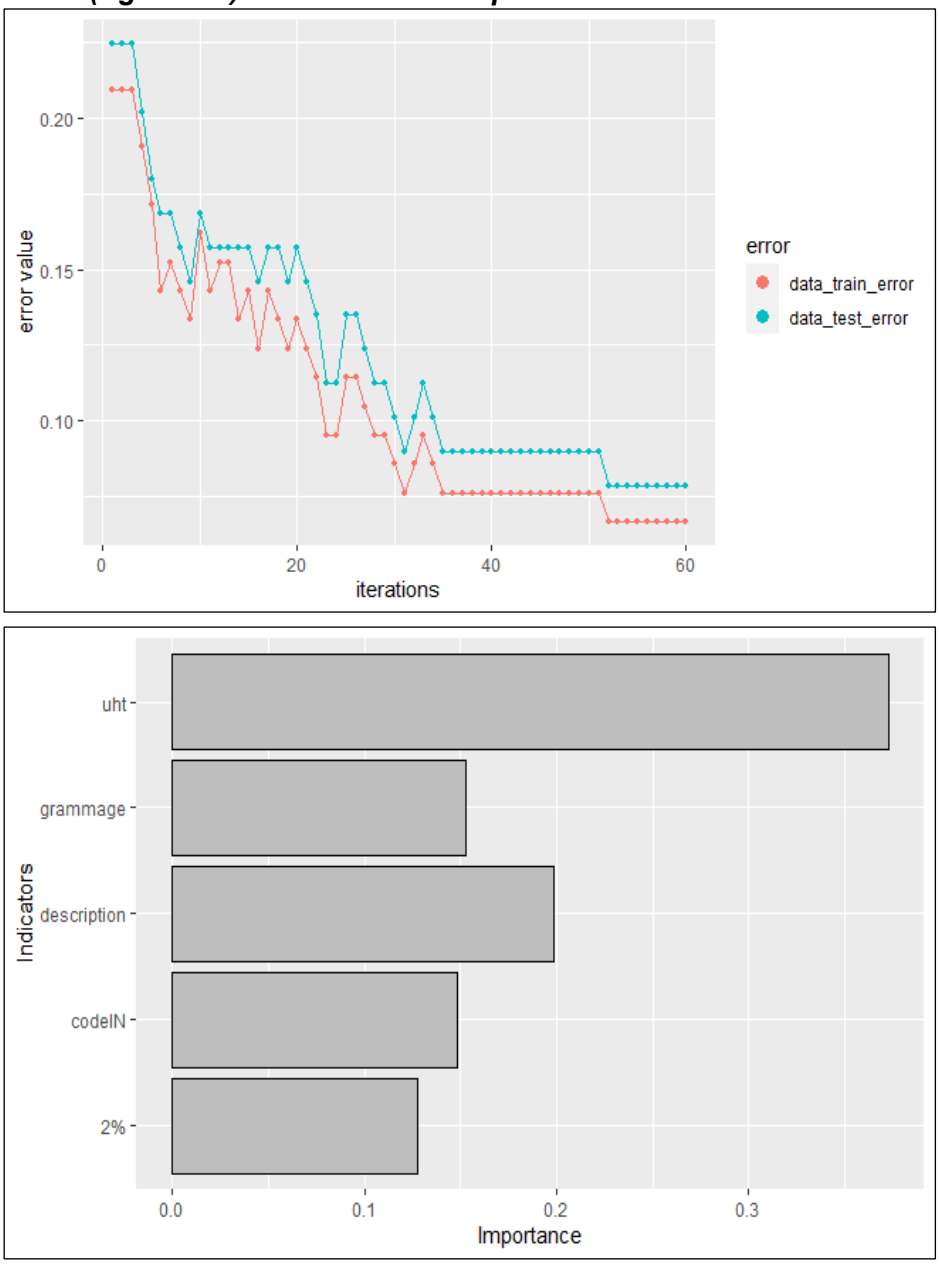
We can use the **data_selecting** or **data_classifying** function to classify products into COICOP (Classification of individual consumption by purpose) groups.

The **data_selecting** 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 *Coicop* column is optional and provides an a priori known product group). These parameters define those sets of phrases or keywords that can, must or cannot appear in the product label, respectively. The function is based on text analysis provided by *stringr* package [24] and is very effective when the product labels (*description* column) are of good quality. **Example 2** (Appendix) presents a selection of UHT milks excluding low-fat milks from the milk collection included in the *PriceIndices* package.

Alternatively, the **data_classifying** function can be used to classify the products into homogeneous groups. This function predicts product classification by COICOP levels using the selected, previously trained model for gradient-boosted decision trees (Chen et al., 2021). Thus, first we need to build a model for the product classification using the **model_classification** function, which is based on XGBoost algorithm from *xgboost* package by [4]. The algorithm is modified to take into account not only numeric columns (which is the case in the standard XGBoost algorithm) but also non-numeric columns (such as product labels) or artificially created binary columns based on the presence or absence of user-set keywords in the product description. We can save the built model to disk (**save_model**) and import it at any time **(load_model**). To get a good understanding of the function for product ML classification, please run **Example 3** (Appendix). In the above-mentioned example the *data COICOP* data set with correctly classified milk products to local COICOP 6 product groups is used. The user

may control the *accuracy* value obtained on the basis of training and testing data sets (*data_train* and *data_test*, respectively) and observe the importance of the used indicators (see Graph No. 1).

Graph No. 1: The level of errors while ML model training (left side) and importance of the used indicators (right side) obtained in Example 3



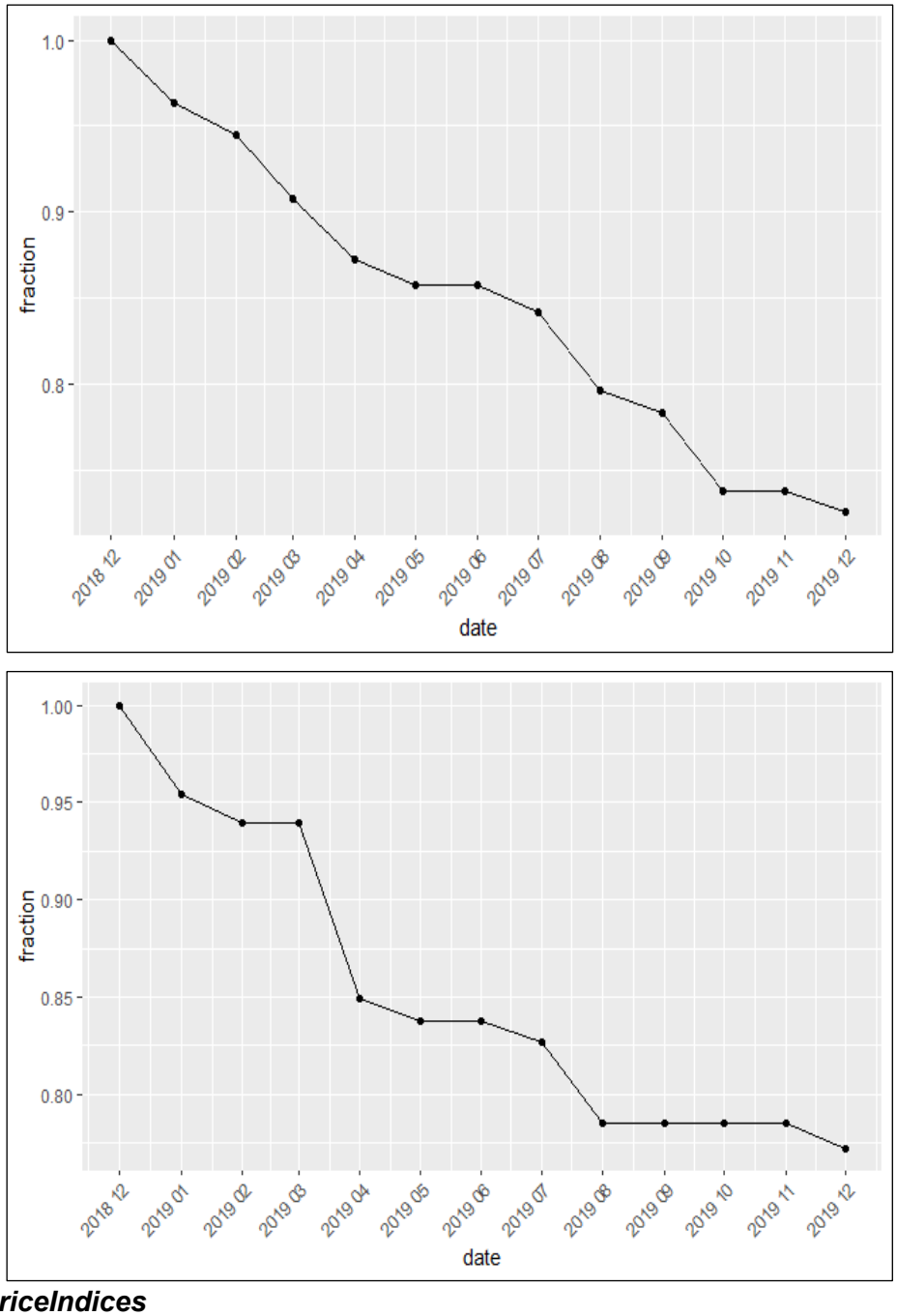
Source: PriceIndices

2.2 PRODUCT MATCHING

The next step in the procedure is to match products over time. This is an important step, especially when considering the lowest level of data aggregation (barcode level). This is because it can happen that a product changes the colour of its packaging and consequently its description and the EAN code, while in terms of quality it is the same product. Therefore, it is important to make sure that we follow the price evolution of the same product. The function for product matching (**data_matching**) is based on *recling* R package by van der Laan (2018) and it depends on the set of chosen

columns for matching. Product matching can be done only on the basis of its label (in which case the *onlydescription* parameter has the value TRUE) or with the use of the retailer codes (*codeIN*) and/or external codes (EAN, SKU - see the *codeOUT* parameter). These two approaches can lead to different numbers of the matched products (see **Example 4**). Generally, in most cases of data matching, we compare descriptions of products, and *PriceIndices* uses the Jaro-Winkler string similarity measure for this purpose. One should also be aware that the ratio of the matched products to all the available products is generally a decreasing function of time (see Graph No. 2 with results for *milk* and *coffee* products obtained after using the **matched_fig** function).

Graph No. 2: The ratio of the matched and available milk (left side) and coffee (right side) products obtained via the matched_fig function



Source: PriceIndices

2.3 DATA FILTERING

In the literature there is an ongoing discussion about whether or not to use data filters for scanner data, and if so, what kind of filters to apply. As a rule, scanner data indices are calculated using a dynamic approach, with most countries opting for the monthly chain Jevons index. This method is commonly referred to as the dynamic method [8]. 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 determined by the number of products in a given product group (the *low sales filter*). A filter for removing products with extreme price changes (*extreme price filter*) and a filter for eliminating recalled products from sale (*dump price filter*) are also often considered [19]. These filters can be used separately or independently together using the **data_filtering** function (see **Example 5** for coffee products). Sometimes filtering is even applied to the weighted multilateral indices, since the reduction of the data set always results in savings in terms of time needed for index computations.

2.4 DATA STANDARDIZATION

If we know the product's grammage and unit of sale, standardizing prices and quantities to a fuller unit seems reasonable. After all, manufacturers often use the "trick" of lowering the grammage of a product with an unchanged price, which de facto means an increase in the price of the product. In such a case, a qualitative adjustment to normalize price and quantity is needed. In the *PriceIndices* package, this can be achieved by using the **data_norm** function (see **Example 6**).

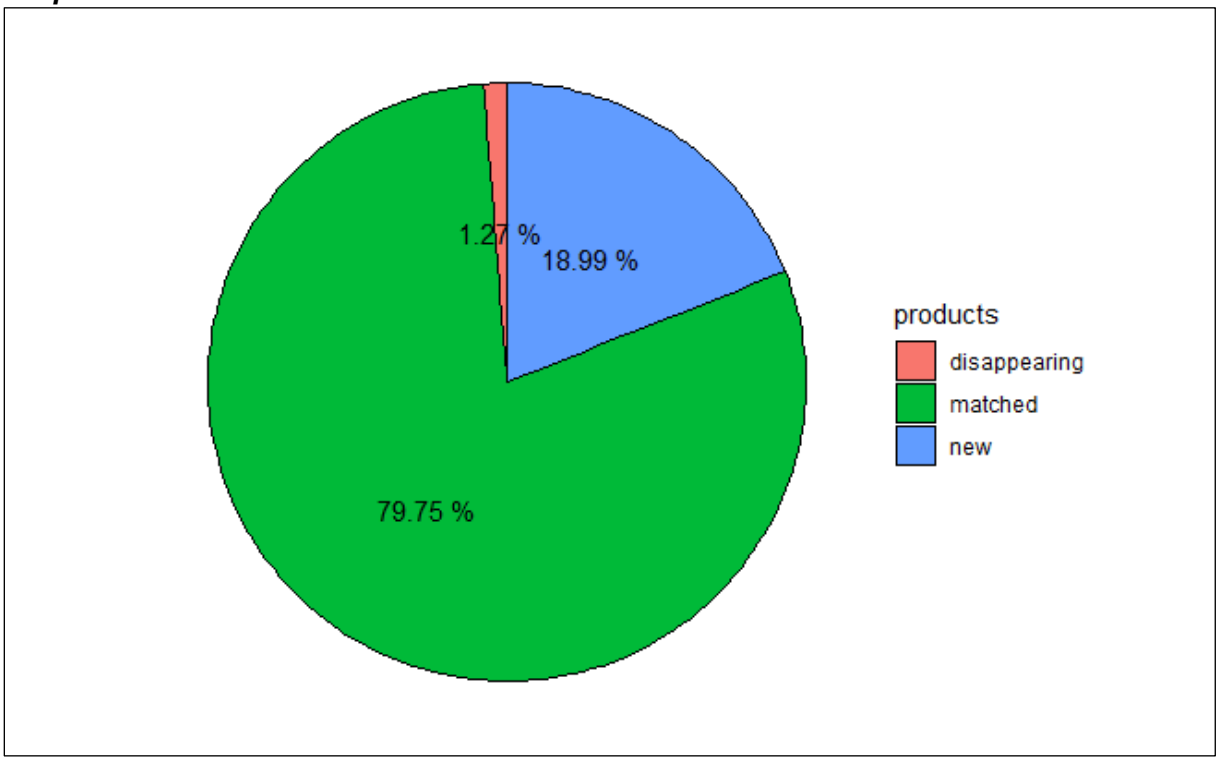
2.5 IMPUTING MISSING OR ZERO PRICES

The **data_imputing** 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 the imputed prices are set to zeros. The function returns a data frame (monthly aggregated) which is ready for price index calculations**.**

3. DATA SET CHARACTERISTICS

The *PriceIndices* package includes a number of functions for determining the characteristics of the analyzed scanner data sets (see [2]). In particular, we can analyze the differences in product sales levels (**sales_group**), correlations between product the prices and the quantities (**pqcor_fig**) or the level of product matching (**matched_fig**). One useful feature is certainly the **product** function, which analyzes the relationship the between matched products, new products, disappearing products and all available products in the given period (see **Example 7** and Graph No.3).

Graph No. 3: The matched, new and disappering coffee products – results obtained via the products function



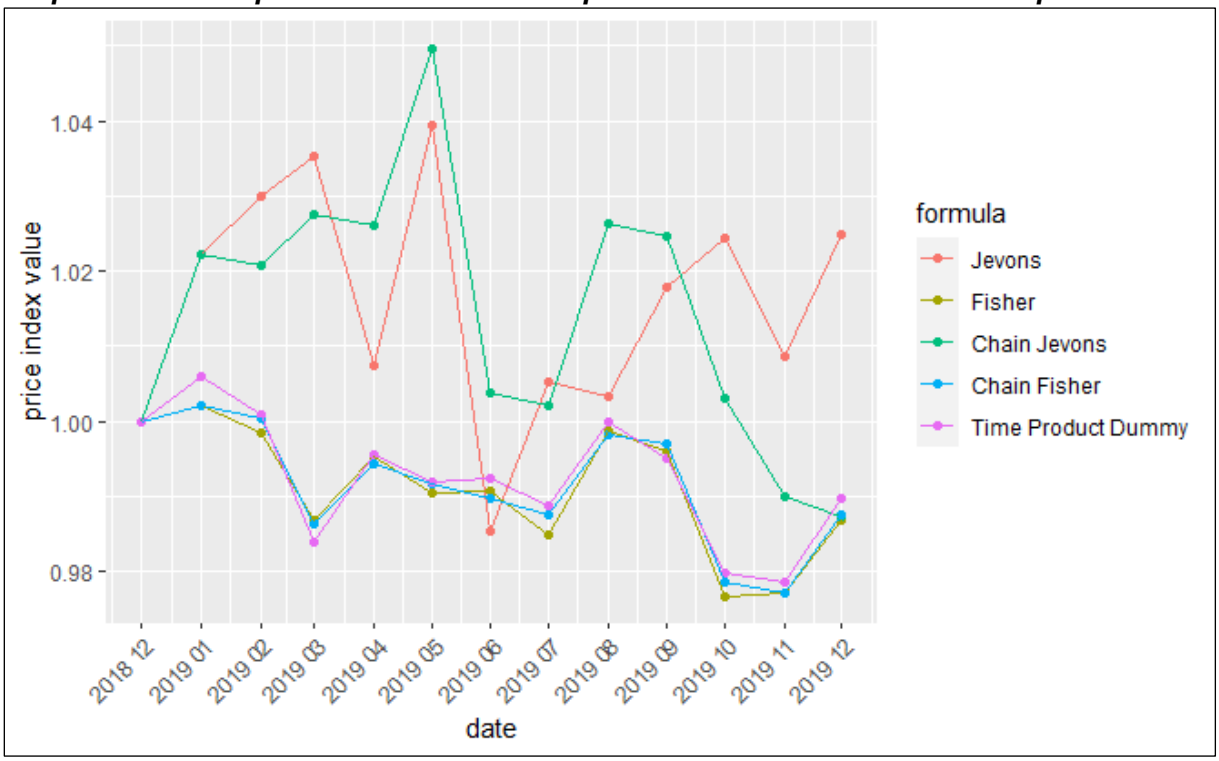
Source: PriceIndices

4. PRICE INDEX CALCULATIONS

The current version of the *PriceIndices* package (ver. 0.1.7) includes: 6 functions for calculating unweighted price indices, 30 functions for determining weighted bilateral indices, 35 functions dedicated to chain indices, and 22 functions for determining multilateral indices (unweighted and weighted – see CPI Manual (2004)). The **price_indices** function is a general function that allows the User to calculate price indices on a given dataset and for different parameters (such as the time window length in case of multilateral indices or the elasticity of substitution in case of the CES index). The above-mentioned function provides a data frame with index results. To compare the price indices determined in this way, we can use the **compare_indices_df** function (see **Example 8** and Graph No. 4).

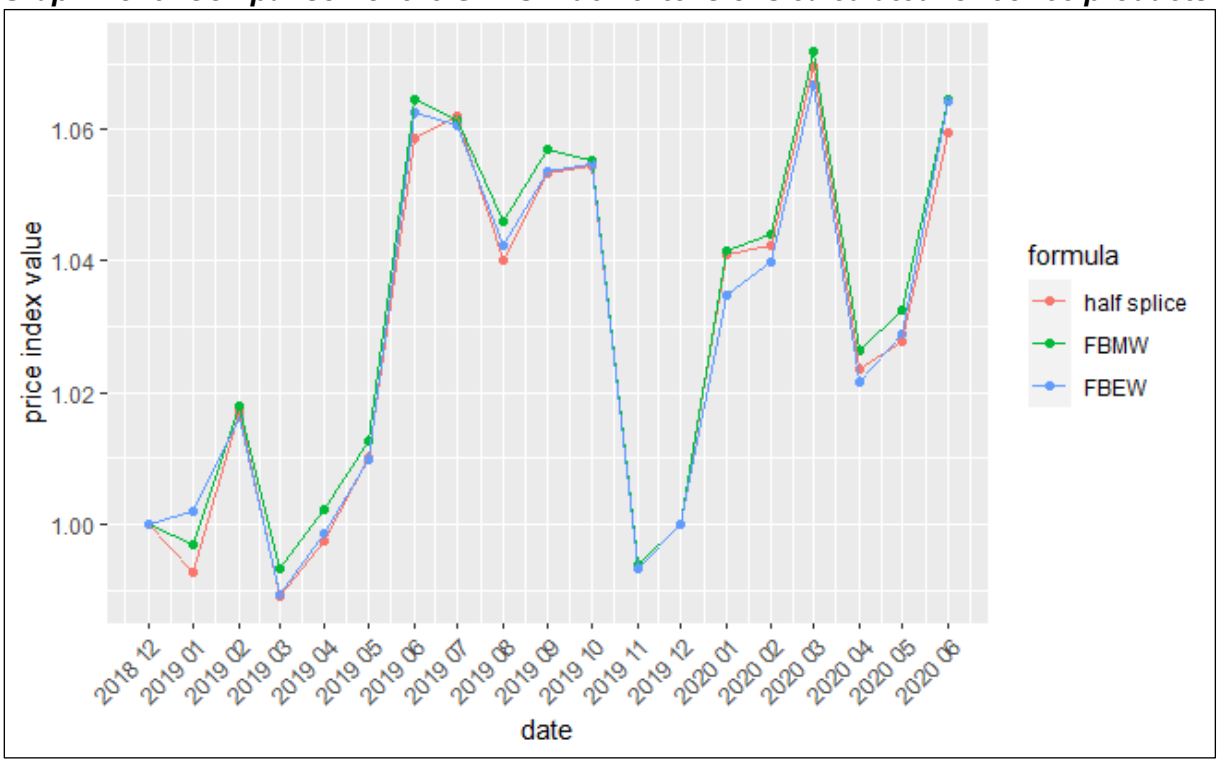
The package also includes extensions known from the literature for multilateral indices [9], which allow the inclusion of data from the following month without having to revise the price indices determined for the previous time window. Among the above-mentioned methods there are not only splicing methods (combining the old and the new time windows), but also FBEW (Fixed Base Expanding Window) or FBMW (Fixed Base Moving Window) methods are included - cf. [2, 3]. **Example 9** and Graph n. 5 present the multilateral GEKS index calculated for coffee products and extended by using a half splice method together with the FBEW and the FBMW methods.

Graph NO. 4: Comparison of the selected price indices calculated for milk products



Source: PriceIndices

Graph No. 5: Comparison of the GEKS index extensions calculated for coffee products



Source: PriceIndices

5. AGGREGATION OF PARTIAL INDEX RESULTS

Aggregation of partial price indices over outlets within a retail chain is advisable, as long as the retail chain follows a regional pricing policy. It should be borne in mind that if the number of outlets of a given retailer is large (e.g., it is several hundred), the time required to determine the final price index may increase substantially.

In the *PriceIndices* package, the **final_index** function allows the User to calculate any price index with optional consideration of aggregation of sub-indices calculated for the subgroups of products (then the User needs to indicate the grouping variable) and/or outlets. The user can choose from several possible functions that aggregate partial results, such as an arithmetic mean, geometric mean, the Laspeyres, Paasche or Fisher formula. **Example 10** demonstrates the aggregation of the Fisher indices over subgroups of milk products (*groups* = TRUE) and the over outlets (defined in the *retID* column with *outlets* = TRUE) where the Laspeyres formula is used for that aggregation (*aggr* = "laspeyres"). The obtained results, with the January 2019 as the fixed base month, are presented in Tab. No. 1.

Table No. 1: The final (fixed base) Fisher price index obtained after using the Laspeyres aggregation over subgroups of milk products and over outlets

Period	The final Fisher index
2019-01	1.0000000
2019-02	1.0002910
2019-03	0.9803976
2019-04	0.9929258
2019-05	0.9895008
2019-06	0.9872751

Source: PriceIndices

6. COMPARISON OF INDICES

The *PriceIndices* package includes two functions for a simple graphical comparison of price indices and two functions for calculating distances between indices. The first one, i.e. **compare_indices_df**, 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. The second function, i.e. **compare_indices_list**, has a general character since its first argument is a list of data frames which contain results obtained by using the **price_indices** or **final_index** functions. The third one, i.e. **compare_distances**, calculates (average) distances between price indices, i.e. the mean absolute distance or root mean square distance is calculated. The next function, **compare_to_target**, allows to compute distances between indices from the selected index group and the indicated target price index. The last function, **compare_indices_jk**, presents a comparison of selected indices obtained by using the jackknife method.

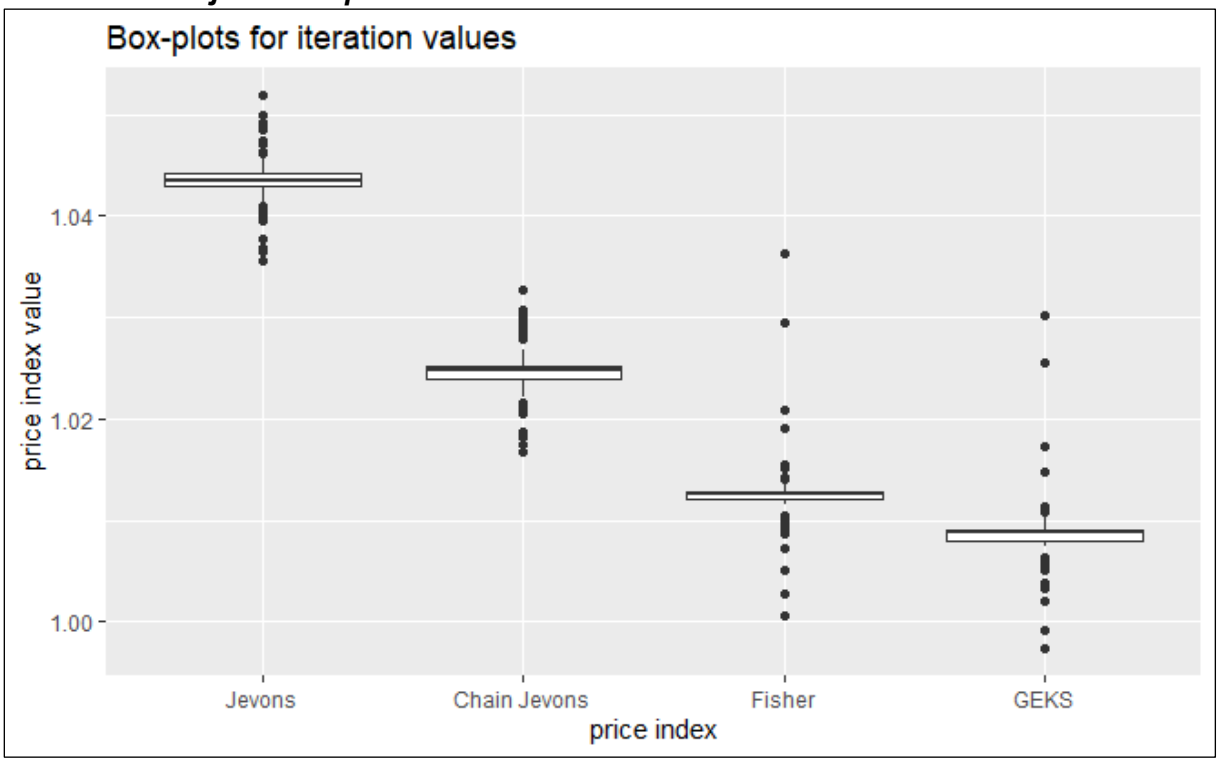
Example 11 and Table No. 2 presents a comparison of the selected monthly price indices calculated for the coffee products sold in 2019 in one of Polish retail chains and **Example 12** and Graph No 6 presents sensitivity of these indices on the product rotation in terms of the jackknife procedure (i.e. in each iteration one product is removed from the sample).

Table No 2: **Mean absolute distances (p.p.) between the selected monthly price indices calculated for the coffee products**

Formula	Jevons	Chain Jevons	Fisher	GEKS
Jevons	0.000	0.696	1.797	1.785
Chain Jevons	0.696	0.000	1.634	1.466
Fisher	1.797	1.634	0.000	0.812
GEKS	1.785	1.466	0.812	0.000

Source: PriceIndices

Graph No. 6: Sensitivity of the selected yearly price indices on the product rotation in terms of the jackknife procedure



Source: PriceIndices

7. CONCLUSIONS

The PriceIndices package is a full-featured IT tool for the procedure of scanner data. Its main advantages are: (a) no maintenance costs (R environment); (b) working with a time variable of the year-month-day type and operating on unit values, which makes the package very close to the practice of statistical offices; (c) wide functionality allowing to carry out the whole path from raw scanner data to the calculation of price indices; (d) wide range of available price index formulas.

The PriceIndices package can be useful for both the academics conducting research on the theory and practice of price indices and for the employees of statistical offices. Since many important features of the package are not discussed in this article (e.g., the author's price indices: the Białek index, the hybrid index, the GEKS-L, GEKS-GL, GEKS-AQU, GEKS-AQI indices, and the Bennet and Montgomery price and quantity indicators as well as many others) we refer the interested reader to the package's documentation available at:
https://cran.r-project.org/web/packages/PriceIndices/PriceIndices.pdf.

The author hopes that the added value of this article is the presentation of the PriceIndices package based on real scanner data sets which are available in the package. Some contribution of the paper is also the demonstration of the various stages of scanner data proceeding supported by reproducible R language codes included in the Appendix.

ACKNOWLEDGEMENTS

The author would like to thank the organizers of the workshop *Modernization of the consumer price statistics* held under the auspices of the Statistical Office of the Slovak Republic in Bratislava on March 23-24, 2023, for the invitation and the opportunity to present the *PriceIndices* package. The presentation delivered at this workshop formed the basis for this article.

BIBLIOGRAPHY

[1] BIAŁEK, J.: The general class of multilateral indices and its two special cases, Paper presented at the 17th Meeting of the Ottawa Group on Price Indices, Rome, Italy, 2022

[2] BIAŁEK, J.: Scanner data processing in a newest version of the PriceIndices package, Statistical Journal of the IAOS, 38 (4), p. 1369 – 1397, 2022.

[3] BIAŁEK, J. – ROSZKO-WÓJTOWICZ, E.: Potential reasons for CPI chain drift bias while using electronic transaction data, Technological and Economic Development of Economy, 29(2), 2023, p. 564-590.

[4] CHEN, T., HE, T. – BENESTY, M. – KHOTILOVICH, V. – TANG, Y – CHO, H. - CHEN, K. – MITCHELL, R. – CANO, I. – ZHOU, T. – LI, M. – XIE, J. – LIN, M. - GENG, Y. – LI, A.: Xgboost: Extreme Gradient Boosting. R package version 1.3.2.1, 2021.

[5] CHESSA, A. – GRIFFIOEN. R.: Comparing scanner data and web scraped data for consumer price indices. Report, Statistics Netherlands, 2016.

[6] CHESSA, A.: Comparisons of QU-GK indices for different lengths of the time window and updating methods. In: Second meeting on multilateral methods organised by Eurostat, 2017.

[7] DIEWERT, W. E. – FOX, K.J.: Substitution bias in multilateral methods for CPI construction using scanner data. UNSW Business School Research Paper (2018-13), 2018.

[8] EUROSTAT: Practical guide for processing supermarket scanner data. Harmonised Index of Consumer Prices. Luxembourg: Publications Office of the European Union, 2018, ISBN 978-92-79-76861-3.

[9] EUROSTAT: Guide on Multilateral Methods in the Harmonised Index of Consumer Prices. Luxembourg: Publications Office of the European Union, 2022, ISBN 978-92-76-44354-4.

[10] GRAHAM, W. : IndexNumR: A Package for Index Number Calculation. R package version 0.5.0, 2022.

[11] International Labour Office: Consumer price index manual: Theory and practice. Geneva, 2004.

[12] IVANCIC, L. – DIEWERT, W. E. – FOX, K. J.: Scanner data, time aggregation and the construction of price indices. Journal of Econometrics 161(1), 2011, p. 24 – 35.

[13] KRSINICH, F.: The FEWS index: Fixed effects with a window splice–non-revisable quality-adjusted price indices with no characteristic information. In meeting of the group of experts on consumer price indices, 2014, p. 26 – 28.

[14] MEHRHOFF, J.: Towards a new paradigm for scanner data price indices: applying big data techniques to big data. In Paper presented at the 16th Meeting of the Ottawa Group on Price Indices, Rio de Janeiro, Brazil, 2019.
[15] SAAVEDRA-NIEVES, A. – SAAVEDRA-NIEVES, P.: IndexNumber: Index Numbers in Social Sciences. R package version 1.3.1, 2021.
[16] STANSFIELD, M.: multilateral: Generalised Function to Calculate a Variety of Multilateral Price Index Methods. R package version 1.0.0, 2022.
[17] QUENOUILLE, M.H.: Notes on bias in estimation. Biometrika, 43 (3–4), 1956, p.353 – 360.
[18] VAN DER LAAN, J.: reclin: Record Linkage Toolkit. R package version 0.1.1, 2018.
[19] VAN LOON, K. V. – ROELS, D.: Integrating big data in the Belgian CPI. In Paper presented at the meeting of the group of experts on consumer price indices, Geneva, Switzerland, 2018.
[20] VON AUER, L.: The nature of chain drift. In Paper presented at the 17th Meeting of the Ottawa Group on Price Indices, Rio de Janerio, Brasil, 2019.
[21] WEBSTER, M. - TARNOW-MORDI, R. C.: Decomposing multilateral price indices into the contribution of individual commodities. Journal of Official Statistics (2), 2019, p. 461–486.
[22] WHITE, G.: IndexNumR: Index Number Calculation. R package version 0.5.0, 2022.
[23] ZHANG, L. – JOHANSEN, I. – NYAGAARD, R.: Tests for price indices in a dynamic item universe. Journal of Official Statistics 35(3), 2019, p. 683 – 697.
[24] WICKHAM, H.: stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.4, 2019.

RESUME
The article presents the basic functionality of the PriceIndices package, which was developed for both academics and practitioners involved in implementing scanner data to measure inflation. Specifically, the paper demonstrates how to prepare scanner data sets, classify products into the appropriate COICOP groups, match products over time and filter out the irrelevant sales. Of course the demonstration of the package also covers the determination of price indices, their comparison and the aggregation of the partial results over the retailer outlets and the product subgroups. The whole presentation is supported by reproducible codes written in R language, which can be found in the Appendix.

RESUMÉ
Článok predstavuje základnú funkcionalitu balíka Cenových indexov, ktorý bol vyvinutý pre akademikov aj odborníkov z praxe, ktorí sa podieľajú na implementácii údajov zo skenerov slúžiacich na meranie inflácie. Dokument konkrétne ukazuje, ako vytvoriť súbory údajov zo skenerov, klasifikovať produkty do vhodných skupín klasifikácie COICOP, priraďovať produkty v priebehu rokov a vytriediť irelevantné predaje. Ukážka balíka zahŕňa aj zisťovanie cenových indexov, ich porovnávanie a agregáciu čiastkových výsledkov maloobchodných predajní a podskupiny produktov. Súčasťou prezentácie sú reprodukovateľné kódy vytvorené v jazyku R, ktoré nájdete v prílohe.

CURRICULUM VITAE
Dr hab. Jacek Białek *has been an employee of Statistics Poland since 2018 where he is an expert in the Department of Trade and Services. Working in this position, he is involved in the analysis of scanner data and their application in the measurement of inflation. At the same*

time, Jacek Białek is an associate professor at the University of Lodz in Poland, where he works in the Department of Statistical Methods. Jacek Bialek's research interests revolve around the theory and practice of price indices. He is the author of more than 90 papers on price index theory, financial mathematics and open pension funds.

CONTACT
J.Bialek@stat.gov.pl,
jacek.bialek@uni.lodz.pl

APPENDIX

Example 1
```
> library('PriceIndices')
> data_unit(dataU, units = c("g", "ml", "kg", "l"), multiplication = "x")
```

Example 2
```
> milkUHT<-data_selecting(milk, must="uht", exclude="low")
> head(milkUHT)
```

Example 3
```
#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,coicop="coicop6",grid=my.grid,
> indicators=c("description","codeIN","grammage"),
> key_words=c("uht","2%"),rounds=60)
#Accuracy (while training)
> ML$figure_training
#Importance of the indicators:
> ML$figure_importance
#Data classification
> data_classifying(ML, data_test)  #please compare the last two columns!
```

Example 4
```
> df1<-data_matching(dataMATCH, start="2018-12",end="2019-02",
> onlydescription=TRUE, interval=TRUE)
> df2<-data_matching(dataMATCH, start="2018-12",end="2019-02",
> precision=0.98, interval=TRUE)
> length(unique(df1$prodID))
> length(unique(df2$prodID))
```

Example 5
```
> data_filtering(coffee, start="2018-12",end="2019-03",
> filters=c("extremeprices","lowsales","dumpprices"),
> plimits=c(0.25,2), lambda=1.25, dplimits=c(0.7,0.7))
```

Example 6
```
> data<-data_unit(dataU,units=c("g","ml","kg","l"), multiplication="x")
# Normalization of grammage units
> data_norm(data, rules=list(c("ml","l",1000),c("g","kg",1000)))
```

Example 7
```
> list<-products(coffee, "2018-12","2019-12")
> list$details
> list$statistics
> list$figure
```

Example 8
```
> df_indices<-price_indices(milk, start = "2018-12", end = "2019-12",
> formula=c("jevons", "fisher", "chjevons","chfisher", "tpd"),
> names=c("Jevons", "Fisher", "Chain Jevons", "Chain Fisher",
> "Time Product Dummy"), window=c(13), interval = TRUE)
> compare_indices_df(df_indices)
```

Example 9
```
> df_geks<-price_indices(coffee, start = "2018-12", end = "2020-06",
> formula=c("geks_splice", "geks_fbmw", "geks_fbew"),
> splice=c("half"), names=c("half splice", "FBMW","FBEW"),
> window=c(13), interval = TRUE)
> compare_indices_df(df_geks)
```

Example 10
```
> final_index(milk, start = "2019-01", end = "2019-06",
> formula = "fisher", groups = TRUE, outlets = TRUE,
> aggr = "laspeyres", by = "description", interval = TRUE)
```

Example 11
```
#Creating a data frame with index values
> DF<-price_indices(coffee,
> formula=c("jevons", "chjevons", "fisher", "geks"),
> start="2019-01", end="2019-12",
> window=c(13), interval=TRUE)
#Calculating average distances between indices (in p.p)
> compare_distances(DF)
```

Example 12
```
#This is a time-consuming procedure:
> comparison<-compare_indices_jk(coffee,
> formula=c("jevons", "chjevons", "fisher", "geks"),
> start="2019-01", end="2019-12", window=c(13),
> names=c("Jevons", "Chain Jevons", "Fisher", "GEKS"),
> title_iterations="Box-plots for iteration values")
> comparison$figure_iterations
```