EUROPEAN COMMISSION
EUROSTAT

Directorate C: Macro-economic statistics
Unit C-4: Price statistics. Purchasing Power Parities. Housing statistics

How to start with web scraping in the HICP: Evidence from EU member states

Paper prepared by Pavel Belchev (pavel.belchev@ec.europa.eu) and Claude Lamboray (claude.lamboray@ec.europa.eu)
for the Group of Experts on Consumer Price Indices

UNECE, Online meeting, June 2021

How to start with web scraping in the HICP: Evidence from EU member states

Abstract

Consumers buy more and more goods and services on-line. The websites of the internet shops can be seen as large catalogues of products with their prices and characteristics, and therefore are a rich data source for official price statistics producers. Online price collection can be automated which makes it an efficient way of obtaining data. This paper reports on different techniques of automatic collection of online data and addresses some of the related legal, managerial, technological, methodological and index compilation matters that National Statistical Institutes (NSIs) might encounter in the different stages of their scraping projects.

The intention of the paper is to support NSIs that are still at an early stage of using web scrapers by identifying potential pitfalls and by giving recommendations on how to avoid them. The paper discusses the general web scraping workflow and gives practical advice for possible solutions at the different stages of the process. The material is generally based on the practice of some European NSIs that have already gained significant experience with web scraping for their CPI and HICP index compilation. Some concrete examples for certain product groups and website types are also included, as well as an overview of the web scraping practices implemented in the NSIs of the EU member states.

Contents

Introduction

This document has been prepared using input and material provided by the member states that have already some experience in online price collection. It will probably be most useful to statistical offices that still aim to implement web scraping, but colleagues that are at different stages of their web scraping projects should also be able to take advantage of some of the reported examples and good practices.

The document starts with some general information on web scraping, which is then followed by its five main sections. Section 1 discusses on some relevant legal considerations. Section 2 offers a summary of the most commonly used technological tools in online price collection. Sections 3 and 4 describe some practical approaches to take into account issues in relation to item coverage, sampling, data classification and validation. Section 5 provides guidance on index compilation and the integration of the scraped data.

The main sections are followed by a list of the used documents, as well as seven annexes with different illustrative examples from the experience of several national statistical offices (NSIs). The last annex concludes with a summary on the results of a questionnaire on the use web scraping in the HICP which countries have submitted during 2020.

1. General information

Internet purchases of consumer goods and services are getting more and more essential every year. The number of merchants that already maintain both traditional outlets and their own websites to sell has recently increased dramatically to the extent that for some retailers nowadays internet is often the main or even the only channel for communication with their customers. The web has grown to become an enormous source of information and this has an important impact on the economy. As we are spending an increasingly bigger part of our everyday life online, the official statistics providers should certainly not ignore this rich data source. Otherwise, they risk to lag behind and to establish a bias towards the traditional channels of economic activities. This is fully valid in the price statistics domain and if we do not want to see this coverage gap getting larger, it becomes necessary to integrate online prices in order to measure properly consumer price inflation.

In addition to that, collecting prices online has proven to be beneficial from a resource point of view as it reduces the costly manual price collection for the NSIs and the response burden for the statistical units. It can also accelerate statistical production processes as data can be collected with a higher frequency, but also in bigger volumes. This increased number of observations introduces a great potential for improvement and for the development of new indicators. Moreover, observing internet data sources helps statistical offices to understand better data patterns. Using automated web scraping technologies one may observe prices daily, hourly and even more frequently which could enrich our knowledge about their volatility and enhance the optimal collection strategies application. It also allows collecting additional metadata (e.g. product characteristics) which can help to improve price measurement.

It is necessary to handle online data in a more efficient way in order to manage the abundance of information which internet provides. The general goal is to transform information that is typically available in HTML format on the web, into centrally stored structures which enable more convenient ways for analysis and practical manipulation. This type of automatically retrieving data from a web source is known as web scraping. This is usually done by a computer program – a scraper, which is run on some server to collect and clean raw internet data for further statistical processing.

2. Legal aspects

The legal aspects of web scraping usually differ per country. Therefore, national applicable regulations need to be examined in detail. One can find a broad overview of national legislation in several member states in (Stateva ea. 2017). However, it is in any case preferable to also make your own research on the existing domestic case law as legally web scraping is a relatively recent topic which is currently a subject of essential change and development.

In general, the NSIs should have a clear legal mandate to collect and access information from multiple sources for European statistical purposes (Principle 2 of the European Statistics Code of Practice). Although this should facilitate access to websites for statistical purposes, the

owners of the websites might block the scrapers access to their data for different reasons – for example, because they may view them as something that destroys the statistics of their users activity. Still, commercial websites sometimes have benefit to be visited by different certain web scrapers so servers are often designed to consider this. In any case, the best approach is to contact retailers in advance asking about their website policies related to access and property rights, as well as about meta tags, privacy or database protection. Establishing this relationship could help the NSIs in future contacts with data owners, especially in cases of major website changes that may affect the performance of the scraper.

For these reasons it is recommendable that NSIs are transparent about what they are doing and to respect not only the mentioned policies that the websites owners might have in place but also the so-called netiquette (etiquette in the net). To follow the netiquette one should generally apply at least the following:

- the scraper is identified using the user-agent string as being from a NSI
- a sufficient pause (for example 1 second) is allowed between requests so that the servers are not overloaded
- the scrapers are run at night time or off-peak hours ([1])
- if the website is using a robots exclusion protocol it should be respected
- read and respect the 'Terms and Conditions' section of the website which may even include a 'no automated scraping' clause

3. Technology

Although essential for any successful web scraping project, it is not the purpose of this manual to discuss in a very detailed manner the purely technical aspects of web scraping[2]. These depend widely on the chosen sources and methods while at the same time the fast changing technologies would soon make such a description outdated. The intention of this section is rather to present some advantages and disadvantages of the different scraping options so that the potential CPI scrapers could choose better the method that is more appropriate for their context and circumstances. Thus, it should help them avoid making certain mistakes that could otherwise lead to potential drawbacks.

([1]) A disadvantage of scraping by night is that one cannot solve sudden bugs in the scraping process. At the same time running the scrapers only at night could luck representativeness in case prices differ between night and day time

([2]) Still, numerous detailed and illustrative script code examples could be found in some of the listed documents in the 'References' section.

Skills – Considerations who should do the web scraping

Data collection from the internet and its subsequent processing for producing statistics requires new skills and reorganizing work processes after switching from the traditional ways of data collection. CPI teams and data collectors need to understand the technical aspects of the internet in order to scrape it. They need to know how a HTML (CSS or other typical web language) page is structured, what an URL is and what is JavaScript, or the structure of data on the web (i.e. the Document Object Model – DOM). Knowledge in query languages, such as XPath and SQL, is helpful for the extraction and manipulation of online data.

Probably the most important decision to make in the beginning is who will be responsible for the web scraping project in the longer term and how to organize it in a sustainable way. As usually this is a function of available human and financial resources, it is obvious that having a higher number of IT qualified staff should make the process easier. Coding the scraper internally enables better options for control. It is much more customizable and allows for more complex solutions and less constraints. If the original developers of the web scrapers leave the project, the problem is who will be responsible for development and maintenance for production.

The ideal situation is that the IT department is in charge of maintaining the scraper, though it is necessary that it stays in close cooperation with the CPI staff. It is not in all statistics offices that IT-departments are directly part of the production process, while at the same time CPI teams usually have insufficient resources and/or knowledge to deal with this. So practice has shown that it would be preferable to have the scrapers designed in a way that the developers could train different, less IT savvy, team members to use them, for example by changing certain parameters in a file.

In any case, it is recommended that CPI teams enhance their knowledge of programming languages for processing and analysing purposes to facilitate better and more sustainable software engineering principles. To overcome the issue of insufficient IT-knowledge in the respective statistics department, standardisation of programs is advisable and if resources are available, the development of a user-friendly application which is programmed generically would make the usage of web scraping programs for a broad range of staff members possible.

Some websites may often change their layout or could suddenly be shut down. So you would have to change your scraping code accordingly to make sure that it keeps working. The recommendation is that the process is started gradually, testing it on a simple website and/or by starting with a single item which is selected because it is less versatile in terms of the different products or price offers that are related to it. However, before a decision is made on the items coverage it is best to first know which are the different web scraping methods.

Before selecting the method

In online price collection, it is better to view web scraping not as the goal but rather as an option of last resort as in some cases a better solution might be organized. For example, it is a good

practice that statistical offices ask data owners to provide them directly with open access to an API (Application Programming Interface). This enables even more solid technical solutions as it involves using a database, which is generally more stable compared to a website. The purpose of an API is to provide structured and controlled access to specific data or services on a server. APIs can be publicly available or locked by an API key. Some retailers may have a structured database that can be approached through such a web-based API which would return exactly the data of interest. A well-known example is the Amadeus API from where one can frequently retrieve thousands of air flight prices. Amadeus can serve as a good starting point to master using APIs ([3]).

It has already been mentioned that establishing a contact with the website owners in advance is recommended. This can lead to more efficient solutions as retailers might even agree to provide statistical offices with scanner data which is preferable for having information on the actual sales and quantities and this is invaluable for setting your weights right. It is always best to have a more stable relationship to enable direct communication with the web page owners. They know their data best and may even offer access to their back-office system, provide information regarding their web sites policies, inform in advance in cases of large website changes for which they can offer ready solutions and perhaps even grant insight to their pricing algorithms. In addition, if a lot of negotiations are needed to avoid being blocked it is probably worth considering to directly invest the effort in trying to obtain scanner data instead.

It is preferable to avoid third-party applications

If you could not get an API access, there are still different methods to scrape the web. There are third-party applications which can be used for this purpose. These are software tools which do not require programming environment or prior programming experience. They could be cloud services, or they could navigate using their own browsers or scrape using automated APIs. However, practice has shown that using third-party applications is rarely free of charge and is not convenient when you want to change scripts. Using them may also cause legal problems because the scraped data could be stored in a foreign country.

The so-called point-and-click functionality tools are also often available as third-party applications. They are probably more useful for detecting price changes in items which are kept constant for longer periods of time, i.e. cinema tickets, language or driving courses, hairdressers or similar services where the metadata in relation to the price is also checked for changes. The point-and-click tools make it easy for non-programmers to automate repetitive tasks but have limited applicability and are still better to combine with a fully featured programming language so it is preferable to have them developed in-house.

([3]) An example of how the Statistical office of Finland are using the Amadeus API can be found in Annex II

Scraping by coding

When it is feasible to develop and maintain scraping by coding internally, it is usually the ideal option. By coding (or scripting) all web browser interactions like opening a webpage, clicking and scrolling, can be automated and elements of interest can be extracted by executing precise queries. The standard that controls how web pages are formatted and displayed is usually HTML. An HTML page is essentially a tree of nodes having different meanings and content. The idea is to set the code so that it reads the tree by looking at the particular nodes to filter and extract the needed information.

The most common and recommended programming languages used to scrape websites for statistical purposes are R and Python. The process that the scraper would execute automatically is to make a request via a URL which will retrieve a HTML response, and then find the particular element from the response from which we want to have the data extracted. Minimizing navigation in this sense is important, as this will make the scraper robust to website changes and generally more stable ([4]). That is why it is necessary to first examine the HTML structure of the website closely so that the particular HTML elements of interest are identified. They will usually be defined with identical tags. Most web browsers have an option to inspect the HTML structure of the sites so it should not be a problem to identify them. The scraper should be instructed to focus on the selected nodes. Some of the most common expressions that your scraper could use to select the same types of nodes are the so-called Xpath or CSS selectors. The CSS is probably more readable but the Xpath generally has more powerful functionality but both can be used for selecting the correct tags in html documents.

Both R and Python also have dedicated libraries that are designed to be used for the purposes of web scraping. These libraries are essentially packages of ready-to-use code – they allow the scraper to download the HTML page and parse it using different functions to select the target information. A good library intended for web scraping with Python is BeautifulSoup, and the one for R is Rvest. The resources in these packages are usually enough for handling static websites. However, more and more webpages display interactive elements. This means that they are not necessarily server-based and their elements could be modified or created dynamically depending on their user actions. Much of their functionality is enabled by the execution of JavaScript codes. That is why it is good to note, that scraping programs run more stable and faster if loading of images is disabled and unnecessary JavaScript codes are blocked by a plug-in or in browser properties. In most cases, images and JavaScript codes do not contain relevant information for statistical purposes, such as price information or product descriptions.

A good solution for dynamic pages scraping is Selenium. Selenium is a standardized intermediary program tool which is mostly used for automated testing of web applications as it can automatically open browsers, fill in forms, scroll down and perform clicks. Selenium is not hard to maintain and it is perfectly compatible as an extension to the scraper. One can use the

([4]) Even though sometimes visiting more links is desired (i.e. it could add information on product characteristics or reveal that some offers are no longer active).

RSelenium library package to enable Selenium for R for dynamic website interaction. Selenium also supports Python so it is perfectly compatible with this language as well.

The natural enemy of a scraper is a captcha. Captchas are popups that often occur when navigating in a web shop. By integrating captchas, web shops try to reduce the amount of robots on their pages. Customers and robots are confronted with quests that change constantly. These quests are primitive for humans but not solvable for robots.

In general, by coding you can handle the data better to suit your needs. It is generally more flexible and cheaper in the longer run. Still, it has to be noted that web site changes can lead to plenty of malfunctions, so statistical offices developing and using the scrapers will, unless you develop a user-friendly generic program[5], need personnel with programming skills, as well as a server to run them on. Most sites will require a custom built scraper and dedicating one for each single site could be expensive, as it would have to be maintained for every small website change. That is why it is recommended to implement it gradually and to start with an item which is easier to handle for allowing the selection of sites which are expected to be generally more stable and not so complex to control.

The differentiation of 'bulk-' and 'targeted-' scraping

When eventually deciding on the usage of web scraping for the production of price indices, one also needs to decide which practice of web scraping suits best for staff members as well as compilation and index calculation methods. When performing so called **targeted scraping**, compilation and index calculation methods stay the same, only the actual price collection is automatized. The web scraping programs thereby imitate the actions of a human price collector. This approach is advisable when starting the usage of web scraping.

The products that are going to be scraped in the case of targeted scraping are predetermined. Thus, the workflow pattern would usually resemble the following: Upload of pre-defined sample for the respective month → let scraper find products by article number or specific URL → scrape all relevant information of product page → store information in data files → manual price collection and replacement of product failures.

On the contrary, when performing **bulk scraping**, the scraper will try to find as much relevant information online as possible. The supply of data is infinite but not quality adjusted. Compilation methods are needed to distinguish relevant items for statistical purposes as well as perhaps the introduction of new index calculation methods which allow big data.

The workflow of the applied scraper strategy is relatively straightforward and usually not much dependent on the programming language or the selected web site for scraping. Scrapers are generally started on virtual hardware connected to the internet. It is important to use a separate

([5]) An example of a similar program, which is developed and used by the statistical office of Austria, can be found in Annex I.

IP address for the scraper, so that potential problems would be limited to its IP. The output could be transferred automatically to data files on a daily basis.

As most sites have a similar functionality, their structure is initially filtered and then the scraper should navigate through the results in steps similar to the following:

Go to homepage → scrape all possible URLs of the subcategories of products → loop all of these URLs → scrape all product information (description, price, metadata) → store everything in data files (or databases).

If the web site is dynamic it may be necessary to provide for automatic interaction, for example by clicking on a button for previous or next page number in order to open observations, or by clicking directly on an item to go to more detailed information about it.

It is important that the scraper users understand the code syntax and are trained to the extent that they could update the necessary scraper parameters in case of web site changes. Such changes can be of different types. For example, differences in the structure nodes can generally be adjusted in a more straightforward manner, but a change in a URL or switch of the data presentation from text to a picture could cause more major problems to adjust the scrapers. It is also a reasonable practice to keep documentation – versioning and user guides should generally improve the scrapers maintenance and support.

4. Items coverage and sampling

Item coverage

There are different considerations to take into account when selecting a COICOP sub-index to scrape. These do not only relate to saving costs but can also involve methodological reasons – for example items that rarely need quality corrections; items for which it is known that online purchases are getting more and more representative; items for which the weights information is already available at the level at which they will be scraped. For most items, it is anyway clear that, when available, scanner data is simply the preferred data source.

Where NSIs are still to start their first web scraping project, the best strategy for them would be to follow a step-by-step approach by introducing scraping gradually until the main potential pitfalls have been neutralized. Once the process is organized and mastered in a sustainable manner the NSIs would have reached the point in which more items could fall within the coverage. Parallel training of the current personnel or hiring new specialists prepared with IT skills should help in this context. Hiring students for internships or new graduates to acquire work experience with IT-skills could also pay off at a later stage.

All of the following are areas where scraping can be very beneficial – fuels, electricity, insurances, books, clothing, medicines, electronics, airfares, accommodation. Though the last few of them are probably not the best to start with, as they may cause many classification

problems from the very beginning for the much higher expected diversity of available products and their characteristics, and they are also such that are often accompanied with captchas.

As a whole, the bigger number of observations would also have an impact on the required statistical methodology (replacement, quality adjustment, grouping). Nevertheless, it is always an option to still follow a targeted data processing approach (by focusing on a predefined list of a smaller number of items) even if the bulk type of web scraping was used (so data for all available products was extracted).

Sampling

Choosing the appropriate web site for scraping one has to consider several important factors. The following website characteristics should be assessed before the decision to scrape:

Representativity: Representative coverage from population or consumption point of view is an important dimension. A distinction must be made if prices are simply collected online so that there is no need for a price collector to visit an outlet if these prices genuinely represent e-commerce transactions. Additionally, in relation to the selectivity of products it has to be kept in mind that sometimes the assortments shown on websites may not be identical to those in the stores, or could even be of different prices.

Volume: The number of observations and the available variables. The more the better does not apply in all cases, especially if this is going to be your first scraping project. At a later stage, it may also be interesting to check how changing the number of observed prices could lead to different final results.

Content source: Not all sites offer original content. Often sites provide data collected from others. This additional layer between the actual data owners and the scraper could cause problems in the longer run.

Stability of the site: The long-term sustainability of the scraped data depends hugely on the steadiness of the web site. It is preferred to scrape sites that are not subject of regular changes and that are not likely to disappear from the web. As this is hard to know in advance, it is important to test the extraction for a sufficient period of time before putting it into production.

Technical characteristics: Prices in pictures or pdf can be harder to scrape. Static sites are easier to start with in comparison to sites with dynamic content where the online content involves JavaScript execution. Choosing a site with a clearly organized menu structure will also make scraping easier.

Methodological considerations: It is preferable to be able to directly query a web source per statistical unit. Linking scraped data to already available statistical unit items might help the consequent classification issues.

Target variables: The identifiability of a variable is always better to be guaranteed beforehand. It is also good to think in advance if the selected variables could be combined with other

available web sources, as this could even lead to the development of new indicators. It is good to know that extracting variables does not cause any more burden on a website so it is better to extract as much as possible from a page already loaded.

Metadata: For the creation of optimal stratification and homogeneity, it is preferable to scrape sites that offer metadata with the necessary detailed product descriptions. Price collection and metadata collection may actually work better also as two different scrapers with different schedules.

As a more general recommendation, once you have found a site that looks appropriate it is worth checking if you can get access to an API or if there is at least one other site that can also serve as a stable data source.

One of the important decisions to make is what should the scraping frequency be (temporal sampling). Prices collected on the internet can sometimes be highly volatile over time with some websites applying dynamic or personalized pricing strategies ([6]). The expected volatility of the observed prices could mean that more frequent extractions are made but for most items, it should be enough to do it on a daily or less frequent basis. As web sites are constantly subject to changes, it is a good idea to be able to adjust the frequency until the scraping process is stable. Thus, there will be time to react in cases of failure (for which it is reasonable to envisage an automatic restart script) or incomplete data collection (where it is good to provide for an option to impute lacking values).

5. Classification and data validation

Classification

The automatic classification of product-offers is probably the most important challenge to process bulk web scraped data efficiently. Product-offers are grouped into homogeneous products and they need to be assigned to product categories that are part of the index stratification (see section 6). This suggests the implementation of classification rules to be maintained in the long-term. The design of such rules is usually handled manually which implies the need to sustain a new set of rules for each new retailer. A typical strategy for classification is to conduct text string searches. This means that certain keywords are extracted from the scraped product descriptions. One could also do a mapping of the product categories used by the website to the product categories used in the index. If the datasets are small, the product-offers could even be classified manually by an expert. Smarter rules in this context are needed. With the help of supervised learning algorithms which classify product-offers taking into account the known classification of product-offers of a training data set, it is expected that

([6]) In Annex IV there is an example of rail transport prices being scraped at different anticipation dates so that they are most representative for the consumer behaviour. Similar strategies may be needed in items such as airfares or accommodation.

automatic classification techniques may appear in time which will further reduce seriously the cost of price collection.

Here is an example of some regular parameters which could be extracted from the scraped data to subset it for classification:

- ID – unique address (i.e URL) or code of an item
- Product type – classification (i.e. men, women, children clothing department)
- Name – name of the product not necessarily unique
- Description – item specific characteristics
- Price – 'offer price' as displayed on the site

In order to ensure that the same products are tracked over time, statistical offices could follow a GTIN or some other item code that the shops are using. Therefore, it is important that the ID is unique and stable over time. A check for identical IDs of different items is also a good practice. Checking for different language versions, as well as spelling errors, abbreviations or terminology could also be implemented.

If you have selected a well-structured web site with good quality of short item descriptions hierarchy and vocabulary, classification will be easier before the NSI becomes more advanced and experienced in web scraping. Still, at a later stage it is good to investigate the idea to add a more detailed, longer item descriptions which may prove useful to refine classification or for optimal methodological purposes. A more detailed classification should generally improve the compilation of price indices.

Monitoring and data validation

Offices have to continuously monitor the validity and usefulness of the tools chosen and the scraping mechanism applied, as it is often impossible to go back in time to recollect data if something went wrong. It is reasonable to monitor the data collection process automatically so that a warning is triggered in case of essential differences – adding a monitoring dashboard should help checking if the scrapers are run properly. A good proxy for detecting problems could be the data size or the number of items that are not classified. This information is already revealing for potential website changes or new collections.

Treatment of non-availability cases or server problems notifications should also be provided and offices are advised to have a backup solution prepared in advance for the periods of interruption. It is essential that monitoring happens early so that it would allow time to restart or redo the scraping if necessary. It is also a good practice to ensure that prices are taken from maintained parts of the site as otherwise they may no longer be valid offers, or the products are already not representative for being sold in very low quantities.

Some problems are not at all easy to detect especially if the scraped data contains thousands of observations. Therefore, it is better to introduce standard checks during data collection, for example to monitor for duplicate records, suspicious values, outliers, number of items found

which have values equal to zero etc. Data validation also of course depend much on the web source of interest, so it is good to take into account item related characteristics like differences between regular prices and sales; seasonality, etc.

In the very beginning, it is a good validation approach to do some parallel comparison with traditional price collection so that you are sure to be on the right track. If a NSI is already more advanced and has developed different scraping techniques, it is a good idea for some time to run independently and in parallel two measured collections of same prices to check the correctness of the scraped observations.

6. Index compilation and data integration

The goal is to compile a price index with the web scraped data. As an input for index compilation, we assume that for each scraped price, the following additional information is available: the date/time when the price has been scraped, the name of the website from which the price has been scraped, a product identifier, a description of the product and possibly other meta-information on the product (e.g. a product category). Each such data point corresponds to a 'product-offer', which is a specific product for which a price can be observed at given time on a website. A product-offer thus comprises the following dimensions over which the observed prices must be aggregated:

- A time dimension (i.e. date/time of scraping)
- A product dimension (i.e. product ID, product description, product category)
- An outlet dimension (i.e. website)

In general, the index compilation is conducted according to the following three steps:

1. The index compiler must make choices concerning the definition of the individual product. The product-offers may need to be aggregated across time, websites and products.

2. An elementary price index is obtained by aggregating across individual products. The index formula may, or may not, use some kind of weights or product characteristics.

3. The resulting elementary price indices are aggregated within an index structure defined below the ECOICOP subclass level.

These three steps are discussed below.

Step 1: Definition of the individual product

First we analyse aggregation across time, and then aggregation across products. If neither of these two steps is needed, the individual product coincides with the product-offer and one can proceed to the next steps.

Aggregation across time

Very often, data are scraped more than once per month (e.g. daily). It is even desirable to collect prices at a high frequency if they are known to fluctuate a lot within a month. If all points in time during a month are equivalent to the consumer, then the whole time period can be considered as homogeneous. Therefore, from a conceptual point of view, a unit value price would be the desired price target. However, weights are not available for each product-offer. Therefore, the calculation of a true average price is not possible. The practical solution is to compile either an unweighted arithmetic average or a geometric average of the scraped prices in a given month. We denote an observed price of a product-offer by $p_{i',t'}$, where *i'* refers to the specific product, and *t'* refers to the specific point in time and T' corresponds to the number of scraped prices in a month.

$$p_{i',t} = \frac{1}{T'}\sum_{t'} p_{i',t'} \qquad \text{or} \qquad p_{i',t} = \prod_{t'}(p_{i',t'})^{\frac{1}{T'}}$$

We say that $p_{i',t}$ is a price of a 'time-aggregated product-offer'. This price depends on the timing and frequency of the web scraping. It relies on the assumption that each product-offer is sold the same number of times. This may, or may not be a satisfactory, especially if prices are collected in different points in time during which more or less transactions are likely to happen. The objective is to come up with an average price that is representative for the reference month. It is recommended to scrap the prices every month at identical points in time.

The points in time to which the product-offer relate can sometimes be relevant from the consumer perspective, for instance in the field of transport services or accommodation services. In such a case, it may be more appropriate to distinguish product-offers with timings of 'different quality' and construct separate time-aggregated product-offers.

Aggregation across products

In general, some kind of product ID is the most granular product level readily available in the data set. In order to be sure to compare like with like, one could track the prices of (time-aggregated) product-offers having the same product ID. This can be a valid strategy, in which case there is no need to further aggregate across products. Moreover, an aggregation across products is not needed when detailed product characteristics are available and used in a hedonic model (see step 2).

Sometimes product IDs can change without any change in the quality of the respective product-offers. If product IDs are matched, the price change between the old ID and the new ID will not be captured. There is also a risk that the price indices will be downward biased if product IDs systematically disappear from the market at discounted prices. Therefore, if there is a high attrition rate in the data, and/or the matched model approach leads to biased results, it can be recommended to further group the time-aggregated product-offers and to constitute broader homogeneous products. Such a strategy appears to be especially appropriate for clothing and footwear.

The practical construction of homogeneous products is usually data driven and largely depends on circumstances (e.g. type of product, availability of product characteristics, etc.). In practice, the following steps could be done:

1. Make an exploratory analysis of the text strings describing the time-aggregated product-offers, as well as of any other structured metadata scraped from the website (for instance website specific product categories).

2. Derive a list of variables and their modalities which can be used for grouping. For instance, time-aggregated product-offers can be grouped according to brand, product type or some other specific product characteristics. In addition, one could also use the price itself as a discriminatory variable (e.g. to distinguish between high-end and low-cost products). Using the price levels to measure quality should however be done carefully, especially as price change is the primary measurement target. Identical prices do not necessarily imply identical quality. It is prudent to examine the item codes that are grouped together based on the price level. Price level may best be used only as an auxiliary criteria once the data has been pre-classified based on other characteristics.

3. For each time-aggregated product-offer, construct these auxiliary variables by extracting information from the scraped data.

4. Define the homogeneous product by imposing the variables to take certain values or modalities. For instance, a homogeneous product can be defined as those time-aggregated product-offers where the brand=X and the product-type=Y. A tighter definition would be to put a restriction not only on the brand and on the product-type, but also on some other product characteristics.

In general, the objective is to design homogeneous products which remain available through time while at same time being composed of time-aggregated product-offers of approximately the same quality. Trade-offs have to be made between stability over time and homogeneity. If feasible, one could try out different homogeneous product definitions and estimate the impact of these choices on the results.

The price for a homogeneous product is then obtained as the arithmetic average or the geometric average of the prices of the time-aggregated product-offers which fall under the definition of the homogeneous product in given month.

$$p_{i,t} = \frac{1}{N_t}\sum_{i'} p_{i',t} \qquad \text{or} \qquad p_{i,t} = \prod_{i'}\left(p_{i',t}\right)^{\frac{1}{N_t}}$$

The choice between an arithmetic and a geometric average is not obvious. The target price for a homogeneous product is a unit value price. In the absence of detailed weight information, it is not obvious if either the arithmetic or the geometric variant is a better estimate of the target price. When following the average price of a homogeneous product over time, the arithmetic variant can be seen as an unmatched Dutot index, whereas the geometric variant can be seen as an unmatched Jevons index. We call this 'unmatched' because the set of time-aggregated

product-offers differ across time. It reduces to a standard Dutot or Jevons index if one assumes that the missing (unmatched) prices in a given month are assumed to be equal to the arithmetic (or geometric) average of the observed prices of that same month. If geometric averaging is used for the aggregation across individual products (see section 3), geometric averaging could also be used at this stage, for the sake of consistency in aggregation. Either way, because price levels of the time-aggregated product-offers that belong to a homogeneous product are likely to be similar, the arithmetic and the geometric averages are expected to give similar results.

Step 2: Compilation of elementary price indices

Once the individual products have been constructed and prices for these individual products have been calculated, the aggregation *across* individual products must be performed. We suppose that the individual products have been classified into a category. The objective is to compile an elementary price index for this category. There are different options. With web scraped data, a limitation for index compilation is the absence of weight data. As it is usually not known how often an individual product was sold, care needs to be taken to ensure that the price indices are produced in an unbiased manner.

Aggregation without any weights

Weights are typically not available with web scraped data. Hence, it is natural to apply unweighted price index formulas. The most standard approach consists in using a fixed base Jevons index which compares the prices of the individual products available both in the base period (e.g. Dec of t-1) and in the current period:

$$I_{t,0} = \prod_{i \in N} \left(\frac{p_{i,t}}{p_{i,0}} \right)^{\frac{1}{|N|}}$$

In this approach, any individual product that is available in a given month t but not in the base month 0, or vice versa, is not taken into account. After one year, the price reference period is updated, allowing to include new individual products that appeared in the previous year.

A more dynamic approach consists in updating the base period every month and chaining the monthly links to obtain a continuous price series. However, such an approach should only be applied with caution as it can result in a bias if for instance the last price is always a discounted price. Another option would be to use a multilateral method such as GEKS-Jevons or a Time Product Dummy method (without weights). The downward bias could be reduced, albeit not necessarily completely removed with multilateral methods. Multilateral methods are not further discussed here.

There is also the possibility of using a fixed base Dutot price index. Note that in a Dutot index, the individual products are implicitly weighted by their price level in the base period.

$$I_{t,0} = \frac{\frac{1}{|N|}\sum_{i \in N} p_{i,t}}{\frac{1}{|N|}\sum_{i \in N} p_{i,0}}$$

When implementing unweighted index formulas, it is important to understand how representative the web scraped prices are. Large data sets may contain both very popular models and models rarely sold. Further cleaning and sampling may be required in order to obtain a representative aggregate price change.

Aggregation with weights

Although weights at the level of the individual product are not known, they could be estimated in a crude way. One practice consists in using the number of times that the product-offers underlying the individual product was scraped as a proxy for weights. Expenditure-type weights can thus be derived as follows for a given individual product *i*:

$$w_i = \sum_{t'} \sum_{i'} p_{i',t'}$$

There are also more advanced techniques to derive weights such as using popularity rankings of the website, or using external information. In any case, the aggregation across individual products can then be performed using these weights. In principle, there is the possibility of using either arithmetic aggregation (fixed base Laspeyres-type index) or geometric aggregation (fixed base geometric Laspeyres-type index)

$$I_{t,0} = \sum_{i \in N} \frac{w_i}{\sum_j w_j} \frac{p_{i,t}}{p_{i,0}} \quad \text{or} \quad I_{t,0} = \prod_{i \in N} \left(\frac{p_{i,t}}{p_{i,0}}\right)^{\frac{w_i}{\sum_j w_j}}$$

A more advanced approach would be to assign monthly weights to each individual product and to apply some multilateral method.

In principle, weighted indices are preferred over unweighted indices. However, the reliability of the proxy weights and of the resulting elementary indices should be closely monitored.

Aggregation with product characteristics

For certain product types (e.g. electronics), it is often possible to scrape the characteristics of the individual products, together with their prices. This makes it possible to compile a hedonic price index ([7]). There are different hedonic methods. In the context of web scraped data, one possibility is to use a time dummy hedonic method. This method is based on a regression where, in addition to product characteristics, time dummies are added as explanatory variables. In particular, the following model is estimated over a pooled period T:

([7]) An example of this can be found in Annex III.

$$\ln(p_{i,t}) = \alpha + \sum_k \beta_k x_{k,i} + \sum_t \gamma_t d_{t,i} + \varepsilon_{i,t} \qquad \forall i \in N_t, \forall t \in T$$

where $x_{k,i}$ corresponds to the characteristics k of individual product i, and where $d_{t,i}$ are dummy variables which equals 1 if the individual product is available in period t, and 0 otherwise. To avoid collinearity, there is no dummy variable for one month, for instance the base period. The index in period t compared to the base period is then obtained as follows:

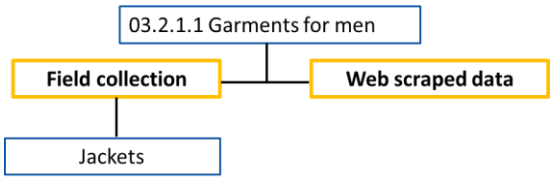
$$I_{t,0} = \exp(\gamma_t)$$

Note that the set of individual products can differ from one period to the next. In general, there are no weights which means that the model is estimated using OLS. This approach does not require grouping the product-offers into homogeneous products. Instead, product characteristics are used directly in the model.

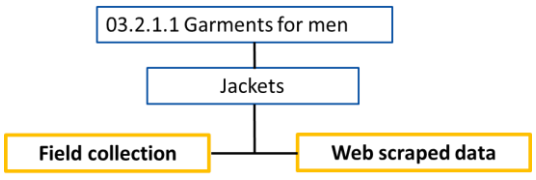
In practice, the model is re-estimated in each month. This implies that results for the periods prior to the compilation month are recalculated. There are different approaches to obtain non-revisable price indices.

Step 3: Aggregation structures below ECOICOP

The compiler must design an aggregation structure below the lowest ECOICOP level. Very often, there are refinements of the ECOICOP applied at the national level. Specific product categories can be defined and weights for these categories are available. A decision must be made at which level web scraped data is integrated in the HICP and combined with other data sources. For example, the web scraped data can be integrated directly below an ECOICOP 5-digit subclass. Alternatively, the web scraped data could also be attached to a more detailed product category already included in the basket.



Integration at ECOICOP5 level



Integration at a product category

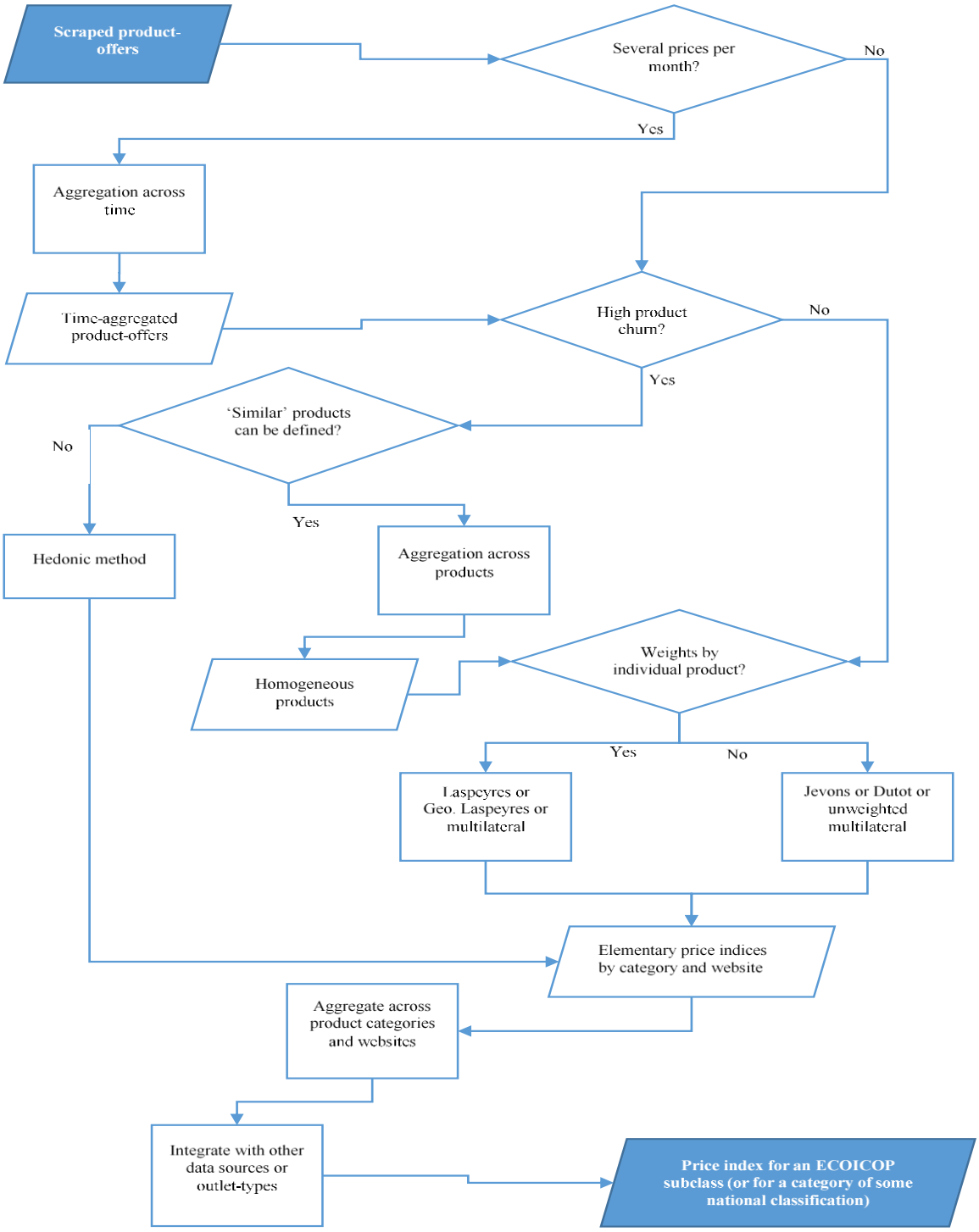
Integration at a higher level allows for more flexibility and for taking into account the advantages of web scraped data (for example a wider product coverage). Integration at a lower lever ensures a better consistency with, for example, manually collected prices. In any case, web scraped data corresponds to a stratum to which a weight must be assigned. Data sources for weights for this level can be Household Budget Surveys or retail trade statistics. For example, a distinction could be made between purchases made on the web versus purchases made in physical outlets. One could also argue that the prices collected on the web do not only represent purchases made online, but also purchases made through other channels. In that case, the weights for the stratum covering web scraped data has to be adjusted upwards accordingly.

The price index for web scraped data may be composed of several web-specific product categories. Price indices can be compiled for the different product categories using the strategies described in section 3. A decision must be made what these product categories are. Weights at this product category level can be more difficult to obtain. One strategy is to rely on the 'value' of the web scraped data. Another approach is to use external data sources. Efforts should be made for identifying such data sources because weights can help to reduce possible biases of the underlying data.

Possible aggregation structures for web scraped data

Data may be scraped from more than one website. Each website (or group of websites) could correspond to a separate stratum for which a weight must be derived. Data sources for weights are business registers, administrative sources (e.g. tax data), or business surveys. This may be especially relevant for important websites with a significant market share. The index for a website can then be customized to the specific assortment of that website. Alternatively, the prices from different websites could enter the same price index for a given product category. For example, the prices of similar models obtained from several websites may (or may not) be combined into a single homogenous product.

Flowchart for compiling prices indices with web scraped data



References

Stateva G., ten Bosch O., Maslankowski J., Righi A., Sannapieco M., Greenaway M., Swier N., Jansson I., *Legal aspects related to Web scraping of Enterprise Web Sites,* ESSnet Big Data Work Package 2, 2017.

Ken Van Loon, *An introduction to web scrapping methods*, presentation at UN GWG on Big Data for Official Statistics, Training workshop on scanner and online data, Bogota, 2017.

Ken Van Loon, Dorien Roels, *Integrating big data in the Belgian CPI*, paper presented at the Meeting of the Group of Experts on Consumer Price Indices, UNECE, Geneva, 2018.

Ken Van Loon, Dorien Roels, *Challenges and opportunities of using web scraping in the Belgian CPI*, paper presented at WGGES meeting: Session on Issues in Inflation Measurement, Lisbon, 2018.

Andrew Glassock and Michael Holt, *Experimental clothing indexes using Australian web scraped data*, paper presented at the 16th Ottawagroup meeting, Rio de Janeiro, 2019.

Robert Griffioen, Jan de Haan and Leon Willenborg, *Collecting clothing data from the internet*, paper presented at Group of Experts on Consumer Price Indices, Geneva, 2014.

Consumer Price Index Manual: Concepts and Methods, Appendix F to Chapter 5 on Web scraping, ILO, IMF, OECD, Eurostat, UNECE, WB, Draft version (September 2019)

Riccardo Giannini, Federico Polidoro, Francesco Pugliese and Antonio Virgilitto, *Web Scraping for collecting price data: Are we doing it right?*, ISTAT, 2017.

Daan Krijnen, Rinus Bot and Georgios Lampropoulos, *Automated web scraping APIs*, Leiden, 2014.

Olav ten Bosch and Dick Windmeijer, *On the use of internet robots for official statistics*, worker paper presented at the Meeting on the Management of Statistical Information Systems, MSIS, 2014.

Robert Griffionen and Olav ten Bosch, *On the use of internet data for the Dutch CPI*, paper presented at the Meeting of the Group of Experts on Consumer Price Indices, UNECE, Geneva, 2016.

Olav ten Bosch, Dick Windmeijer, Arnout van Delden and Guido van den Heuvel, *Web scraping meets survey design: combining forces*, presented at the Big Data Meets Survey Science Conference, Barcelona, 2018.

Robert Griffioen, Olav ten Bosch and Els Hoogteijling, *Challenges and solutions to the use of internet data in the Dutch CPI*, presented at the Workshop on Statistical Data Collection, The Hague, 2016.

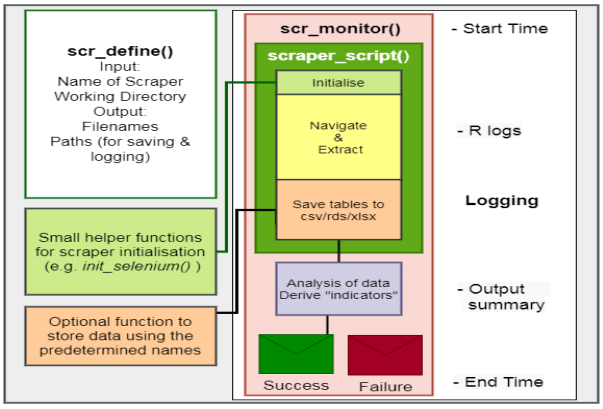
Annexes

ANNEX I

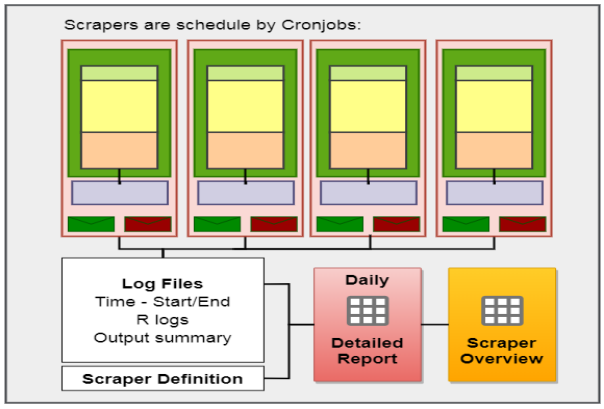
How to keep up with changing web content – Statistics Austria's approach using R

After Statistics Austria's first positive experiences with web scraping using R-scripts to view, navigate and extract websites, the number of targeted websites increased fast. In the beginning, scaling up the operation of scrapers by writing more scripts was easy. However, to maintain data quality and integrity over time, there was a need for a supporting framework around the otherwise independent scripts. The aim of the framework was to be able to keep track of the scraping schedule and to monitor the resulting data with automatically generated emails and reports. This is crucial, because otherwise even minor changes of websites may lead to the problem of undetected website changes that cause lost variables, lost observations, or even complete scraping failures. A scraping framework may save you from the frustration of discovering data problems too late when initially starting to work with a website's data for index compilation purposes and also increases the chance of a timely mitigation of collection errors in production processes.

To solve the problem of undetected website changes Statistics Austria created 'scrapeSTAT', an R toolset that wraps around the existing scripts and can be easily implemented with minimal risk of interference. Before the data collection the scraper is now defined with a unique name, and the output file names are either derived from this name or manually specified. After the collection process, the stored data can automatically be read again, analysed and the resulting information stored or sent via email. The process is shown below:



Single "ScrapeSTAT" Scraper



The system of scrapers

This process allowed us to focus on the data extraction in the scripts, taking away the need to think about logging and notifications for each scraper. As long as a scraping script produces a single or multiple tables as output with minor code changes it fits into this framework.

Each scraper enrolled in this system will at the end of its runtime automatically email a table with a set of indicators for each variable to specified recipients. Indicators are the number of unique values, the number of empty values, the number of numeric values and if the value contains numeric values, the minimum, maximum and mean values. On failure, the error

message is included in this message. In addition, each morning two reports are generated. The first reports lists all scrapers active since the last report and compares the indicators of the current run to the two previous runs and is archived daily. Large deviations are highlighted by a colour gradient. The second report lists all scrapers and their last runtime and compares runtime and the number of observations with an average of the last five runs. It is very useful to get an overview over all scraping processes and find overdue ones. A colleague receives reminders to check the newly available reports daily. The reports rely on R Markdown and are outputted to a single file with inline CSS and JavaScript. This allows for convenience functions like sorting or filtering.

So far the system proved to be very valuable and reliable. Monitoring the scraping process over time highlighted how different companies approach the maintenance of their website. Here are the most commonly found types of site changes, excluding changes in the actual page content:

The Relaunch: This is how most changes are introduced to web shops. Overnight the new version appears. These versions have already been internally tested and may stick around for a few years. If the majority of changes were to the front-end of the website, it is likely that all XPaths or CSS selectors of existing scrapers need updating, but there is a good chance that they will be similar. These relaunches are easy to detect, as existing scrapers tailored to the old site will fail.

Modular Change: Frequently, we encountered websites for which only specific parts of a website were updated, like the product filtering system and product lists. This may cause problems if the scraper relies on these features for navigation. In case the scraper fails, they are easy to detect. Changes to the way the pagination on the website works can be really problematic though. For example if you change the page by manipulating the parameter in the search bar, a change behind the scenes could lead to your scraper returning the same page multiple times. In our report this will be spotted by the indicator showing the number of unique values per variable.

In-house Team & other minor fixes: A few companies seem to employ dedicated staff to change minor parts of the website weekly. Small changes in class names, object ids, or the introduction of new pop-ups may all be detrimental to data quality. The detection of these minor changes is the largest benefit provided by ‘scrapeSTAT’. These problems are most of the time very easy to fix, and if undetected could lead to misleading data. Observing the data type and count of numeric or empty values is very helpful here.

All of the changes mentioned above are likely to be deployed suddenly. It is unlikely that the scraped data will decay in quality slowly over time due to changes by the website owner. Thus, when comparing the current run to the previous ones it will be easy to see when the changes occurred. Note that when the data is used in the production of price indices, additional and more thorough checks will have to be applied after the price collection anyway. But it is likely that you will have to collect data for some time until you gathered enough confidence to promote the data source to production use. Furthermore, early detection of issues increases the available time to find solutions and allows you to restart the scraper maybe even on the same day.

ANNEX II

Example from Statistics Finland on web scraping flight prices with Amadeus API

Statistics Finland has experimented web scraping with Amadeus API to collect the price information of flight prices. The benefit of using Amadeus is the fact that more prices may be collected via API compared to traditional price collection with less time spent on the collection.

The web scraping is done using Python programming language. The packages (datetime, numpy, pandas, sys) needed in Python script are downloaded and imported.

The GitHub-packages that allow us to host software packages, are available for everyone and can be downloaded from https://github.com/amadeus4dev/amadeus-python. The GitHub site also includes examples and other guides to help the starting of web scraping with Amadeus.

In short, the request with parameters on wanted flights is sent to API. The API returns information on prices and additional characteristics (e.g. which company operates the flight).

Methods

Login: A Python-function executes the retrieving of prices with given parameters. At first the function imports the Amadeus package (downloaded from the GitHub) and the credentials to API login are given in the code.

The mapping table is essential for functioning web scraping process and it also enables update of the sample and detailed flight information when necessary. The variables that are allowed to change, such as destination, weekdays, nights and class, are described in the mapping table-Excel-file. If any changes are needed, they are made to the mapping table and not directly in the Python request. The mapping table looks like following:

destinationName	destination	Weekdays ([8])	Nights ([9])	class
London	LON	0,2	1,3	ECONOMY
Luxembourg	LUX	4,5	5,7	BUSINESS

The example above defines the flights of which prices are collected:

1. Economy class flights destined to London on Monday or Wednesday (*weekday)* and timespan between departure and return dates one or three nights (*nights).*
2. Business class flights destined to Luxembourg on Friday or Saturday. The trip may last either five or seven days.

Processing of parameters: After updating the mapping table, the function extracts the parameters from the mapping table. Departure and return date times are deduced based on the information (nights, weekdays) in the mapping table and the date when the API-request is made.

([8]) weekdays notation: 0=Monday, 1=Tuesday, 2=Wednesday,…,6=Sunday
([9]) nights notation: 1=one night stay, 3=three nights stay…

After that, parameters origin, nonstop10 and currency are decided. In our case, the default value for departure airport (origin) is Helsinki (HEL), the default value for nonstop is 1 as we want to have direct flights, and the default value for currency is euro. These parameters are same in each request and thus can be hardcoded in the function. Now all the parameters needed to get the product offers from the Amadeus API have been defined:

- destination and travel class directly from mapping table
- departure date, return date, latest arrival time and latest return time calculated from mapping table and the API-request date
- default parameters origin, nonstop and currency described in the Python-function.

Through loops, the function takes one row of parameters at a time and enters parameter values to the function that collects the price and flight information on each flight that fulfills these constraints.

The API-request in practice

1) The needed packages and functions as well as the mapping table are imported to Python.

2) It is decided how much earlier the flight information need to be gathered. For example, one could want to get information on flights one month, three months and six months prior to actual departure date.

3) The mapping table is defined having all the flights belonging to sample

4) The actual code retrieving the data from the Amadeus API with nested loops is run:

a. loop going through each row in the mapping table

b. loop going through all the of dates prior to departure defined in step 2 (one month, three months, six months)

c. loop going through all the defined weekdays

d. loop going through suitable times between departure and return date (the duration of the entire trip) and collecting the information on each flight on given parameters with function:

1) the Amadeus package is imported to Python and credentials to API are given

2) parameters to flight search from API are defined and sent to API: destination and class are retrieved from mapping table and departure date, return date, latest arrival time, latest return time are deduced from weekdays and nights from mapping table. Origin, nonstop and currency are specified.

3) the flight metadata is collected

4) the flight price information is collected.

5) After all the loops have been completed, the collected information is saved as Excel-file.

([10]) nonstop notation: direct flight=1, flight with lay-overs=0

ANNEX III

Example from INSEE (Statistics France) on web scraping laptop prices to estimate hedonic models

When an item is missing and has to be replaced, the difference in quality between the dispparearing product and the new one must be taken into account in the consumer price index, in order to measure comparable prices. Hedonic regressions can be used to estimate this difference, using product characteristics as explanatory variables for the price. However, the quality of the models can be insufficient due to the small size of samples. Web scraping can be used in order to gather bigger volumes of information on prices and characteristics, in particular for electronic goods. Hedonic models can be estimated with traditional hedonic regressions or other predictive methods, including machine learning algorithms.

Web scraping programs have been developed using Python and the data collected on two French e-commerce websites have been analysed: rueducommerce.com, boulanger.com. Price, name, brand, technical characteristics of the product were collected for laptops. The website can change over time, which can lead to a disability of the program initially written and a need to recode it. Maintenance is thus very important in production. For hedonic models purposes this matter is less difficult to tackle, as the robot has to be executed only when a reestimation of the coefficients is needed, i.e. at each base month (December) for hedonic repricing. Despite the fact that web scraping brings overall better quality than manual collection due to its systematic aspect and the quantity of data which is generated, it also has drawbacks decreasing the quality:

- some of the variables are underfilled for an important proportion of the observations
- variables named differently may correspond to the same concepts
- modalities can be different for identical or similar products.

Hence, a lot of cleaning and variable selection work has been done. Infrequent brands and seldom used colour terms have been grouped together to limit sparse modalities. Missing values have been imputed with the most frequent value (qualitative variable) or the mean value (continuous variable) of filled observations. Removing all observations with missing values would have led to the deletion of too many observations (only observations with more than 50% of the missing variables were deleted).

The data we get from the e-commerce websites are quite exhaustive to describe the products. Many of these characteristics can contain redundant information, for example the length and width of the laptop and the size of the screen, or the base and turbo frequency, etc. Moreover, there are far too many variables: we do not want to overfit our model; many of the variables may have small importance; in practice, if manual collection is done on a monthly basis, we do not want the collection to cause too much work for the price collectors. A model with all these variables would not be possible in practice. Hence, it is important to select a subset of our set of variables.

To do this, we have mainly used two types of methods:

- methods based on trees (the observations are divided into homogeneous price groups according to the values of their characteristics and the predicted value is the average of the prices in each group
- the variable selection method consists in cutting the tree at a given level and retrieving all the variables (or nodes) that make it up)
- shrinkage methods (Ridge or LASSO penalized regressions make the coefficients of the least predictive variables decrease; it is then sufficient to observe the parameter values in order to know the most relevant features of the scraped database).

These algorithms were also used for prediction, as the goal of hedonic models is to reprice the product chosen at basis month with the technical features of the new product. When estimating the models on a training sample of 80 % of the scraped data and predicting the prices of the remaining 20 %, the results fall between 78% and 85% of accuracy on linear models, and between 83% and 87% for random forests, showing that machine learning prediction methods can be a promising way to reprice substitute products. However, the difference is not very large, and linear models would be used without losing too much precision. Moreover, the errors can be caused by a high variability of pricing by the seller, even for given technical characteristics.

Reference

J.D.Zafar, S. Himpens, *Webscraping laptop prices to estimate hedonic models and extensions to other predictive methods*, 2019.

ANNEX IV

Example from INSEE on their new methodology for calculating price indices for rail transport

Until 2019, the data used to calculate the mainline passenger transport index within the passenger transport by railway item (item 07.3.1.1.1) comes from catalogues listing the base price of each trip. These regulated prices are updated every six months and no longer reflect the prices actually paid by consumers. Indeed, 'yield management' techniques are used by railway companies: this consists of optimizing the turnover of a service by adapting, in real time, the price of the service according to the occupancy rate and the consumer. These methods are also used by airline companies, or in the tourism sector. For these services with very volatile prices, Eurostat recommends the constitution of a sample of prices representative of the behaviour of the consumers, by including in particular the anticipation of the purchase and the period during which the service is consumed.

Thus, an automated internet data collection (web scraping) has been implemented on the train ticket websites. An experiment was carried out for high-speed trains (classic or low-cost) in 2018 and 2019. Daily, a robot collects ticket prices with four anticipation dates (2 days, 10 days, 30 days and 60 days before the train departs), according to two consumer profiles (with or without a reduction card) for a sample of 250 journeys (one way), which corresponds to more than 10,000 requests. In 2020, this collection method was also extended to other mainline trains and regional trains.

The prices obtained are then aggregated by a geometric average at the elementary level by priority, profile, route and period (week or weekend). They are finally aggregated by a Laspeyres index. The weights for the different journeys were calculated from aggregated rail traffic data and data collected by webs craping, which allow to know the frequency and prices of trains on each line.

The new index obtained from data collected by web scraping therefore exhibits volatility - prices increasing sharply during the school holiday periods -, unlike the index calculated with the current method. The maximum difference on the passenger transport by railway item was 7.6 points in November (see Chart 1). The difference over one year between the two indices is -4.7 points. In May 2019, a major change in train ticket prices resulted in an observable decline from August (the first month in which all tickets were purchased with the new pricing). The impact of the methodological change was 1.5 points in November on the 'Transport services' group (07.3) and 0.04 points on the overall index[11].

([11]) In November 2019, the index obtained with the old method is 99.6 and the index obtained with web scraping is 92.0; the difference is therefore 7.6 points.

Chart 1: Passenger transport by railway item (07.3.1.1.1) (base 100 in December 2018)

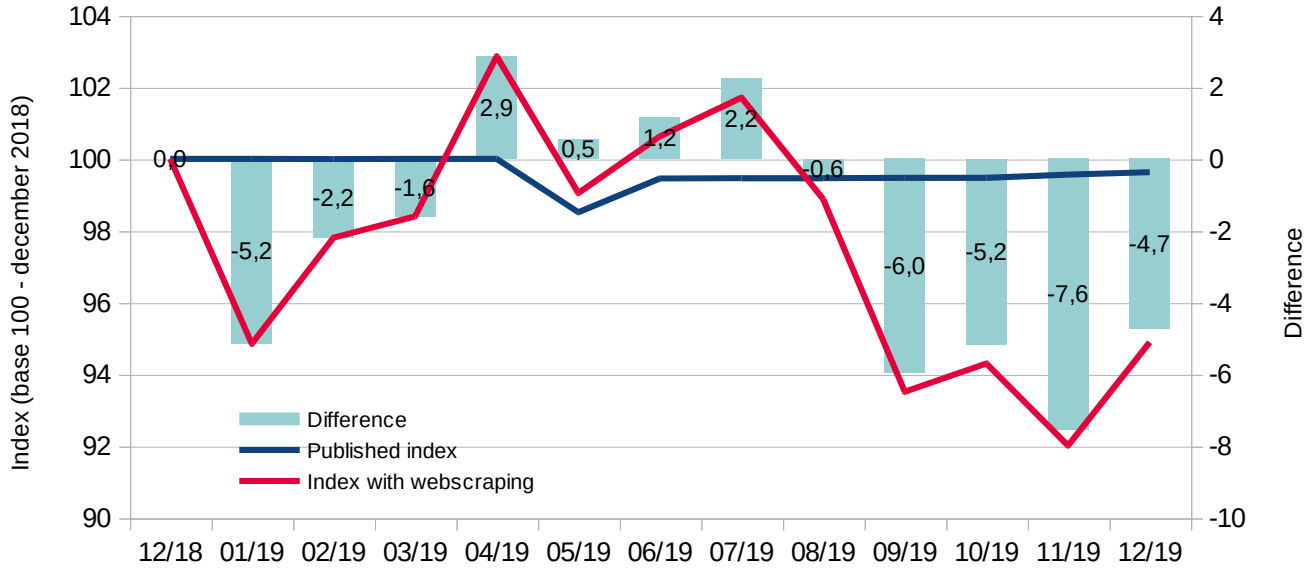
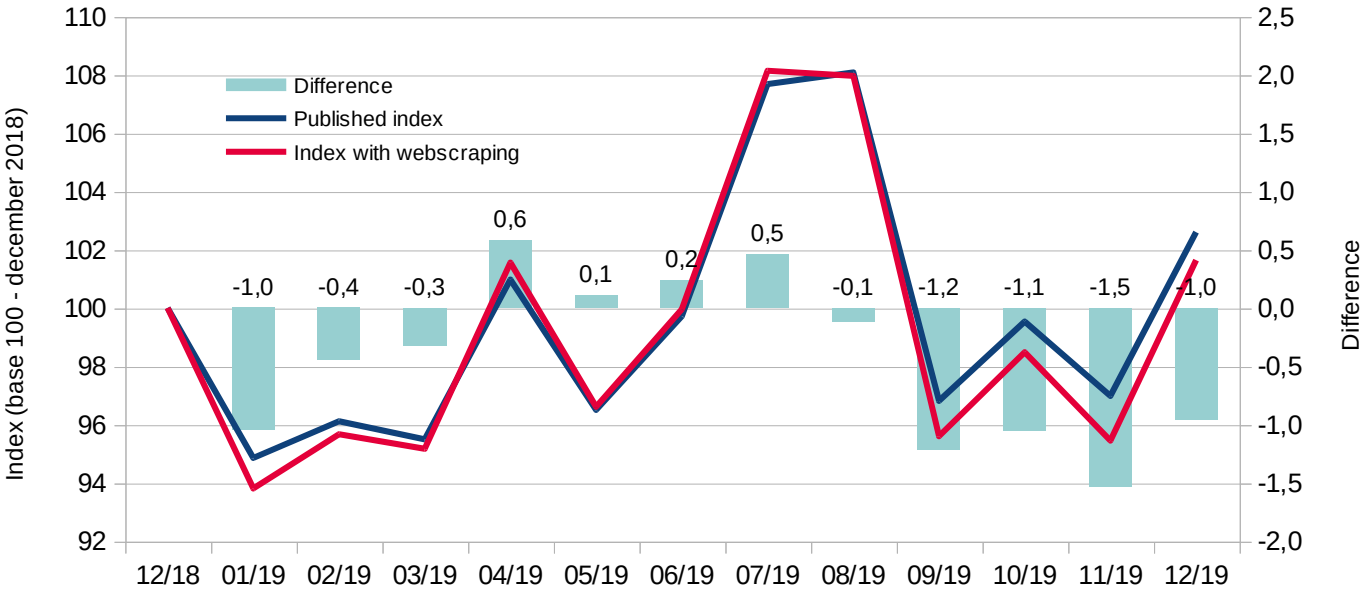


Chart 2: Transport services index (073) (base 100 in December 2018)



ANNEX V

Example from STATEC (Statistics Luxembourg) on electronic products scraping

The dataset

The bulk type of web scrapping, which is done by using Python code script, is being utilized three times a month to ensure that the data for all monthly available products is retrieved. The script scrapes all the information available on the webpage for every product: the product name and the EAN code, the internal shop code and classification, the price, and all the characteristics and specifications. As an example for a television, the script scrapes the screen size and type, the resolution, number of HDMI, LAN and USB ports, energy class and consumption as well as the physical characteristics. For production purposes, STATEC Luxembourg currently uses only the price information, the characteristics and specifications aren't automatically used for price index compilations but are used for manual replacements.

Current use of the data

STATEC Luxembourg is currently dubbing a manual price collection for production purposes. The current basket is based on the previous traditional physical price collection, which was defined together with the salesperson and is updated with recent products. Prices for predefined basket of products are selected from a final aggregated monthly dataset of products. This dataset includes prices of all products which have been available at least once throughout the analyzed month (prices of products which are available multiple times are averaged using a geometric mean). The current dataset also includes the respective labels and manufacturers, the internal shop code and classification. Price index compilation method, which is utilized for price index series construction, is a fixed base Jevons method. In case if a product from predefined basket of products cannot be found in the final aggregated monthly dataset of products, this product is manually replaced by a similar available product and the predefined basket of products is updated accordingly. This is mostly done by analyzing the most important characteristics like brands and other available information, and does not differ from a replacement in the traditional price collection. As an example if the current model of a specific mobile phone is not available anymore, the newer model is used as the replacement product.

Future plans

Analysis is being currently undertaken to be able to take full advantage of the web scraped metadata with the products description. The objectives are to use all the available web scraped information to better classify the products into their respective COICOP groups (some

combination of categories and sub-categories shall be considered) and to extract products quality characteristics, which will make it possible to automatically replace disappearing products with new ones. Moreover, in order to compile a more sophisticated price index with the available web scraped data, STATEC Luxembourg is analyzing alternative price index compilation methods. For instance, time product dummy regression method is being viewed as a possible alternative price index compilation method, if an extraction of products quality characteristics will not be possible. Moreover, if an extraction of products quality characteristics will be possible, time dummy hedonic regression method is being viewed as a possible alternative price index compilation method. One of the advantages of both price index compilation methods are that no products` replacements are needed, which significantly reduces time of manual intervention.

ANNEX VI

Dynamic web scraping using Python (Statistics Norway)

Web scraping has in various ways been tested and used by Statistics Norway for some years. Some web pages that are currently being scraped by Statistics Norway are more of a static nature. A static structured web page is characterized by having all the relevant information available immediately in the HTML structure when opening a page by the URL. No further interaction with the web page is needed to get the relevant information such as price, product ID, product name etc.

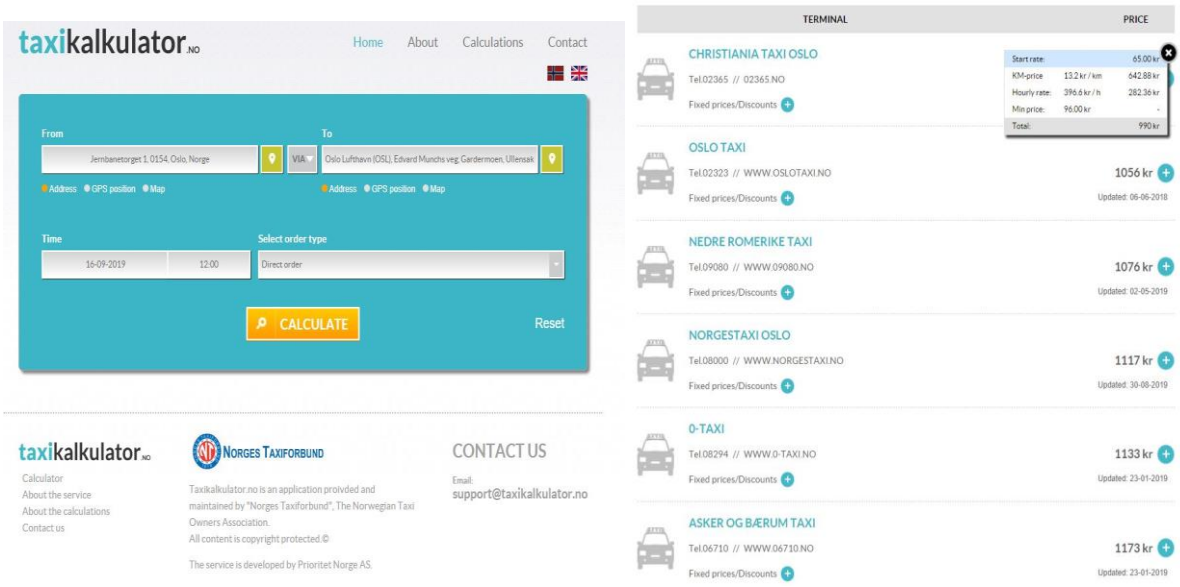
More complex methods opens for the possibility of scraping dynamic web pages when collecting data, web pages which require some form of input from the user before generating the relevant information. Examples of such web pages are booking sites of accommodation and travel services, but also sites which sells goods that can be ordered online. This text will present a user case regarding the taxi market.

The regulations regarding the Norwegian taxi market will see a change from November 2020. As of today, the taxi market is strictly regulated, only licensed taxi drivers that are operating out of a taxi dispatcher are allowed to provide taxi services. One of the main changes as of November 2020 is that the taxi market will open also for companies not operating out of a taxi dispatcher, providing taxi-related services through apps, like Uber, Lyft etc. This will of course change the population of taxi service providers, thus also the sample in the CPI/HICP. The new regulations encouraged us to have another look at the index of taxi service, both because of the new taxi services population, but also the reason that the data collection is performed manually, hence a tedious and labour-intensive task. The index for taxi services was a fitting subject to search for a more effective way of collecting data, and as such, was an inspiration for exploring the more complex possibilities of web scraping.

The web site that was used for this project was www.taxikalkulator.no, an official web site provided by the Norwegian Taxi Owners Association. The taxi calculator site provided a user-friendly price calculator regarding taxi services, which allowed user to search for a given destination and in return gave a price list for the various taxi dispatchers. The results provided information regarding which taxi dispatchers operate in the given area, as well as the associated price information, allowing the users to compare prices. Since this web site contains price information for all taxi dispatchers in a given area, it is an ideal web site to scrape data from.

The web scraper used for this project was built using the programming language Python. While most web scraper previously built was based on more static web sites, this time the web scraping programming had to act as a human user in order to get the data in question.

The taxi calculator is built so that users input their desired to- and from destinations, in addition to the time period. The resulting list will provide the taxi dispatchers in the area with prices in ascending order.



Given the taxi calculator, the web scraper needed to go through the same steps that a human user would do in order to get results: Click the search field for where to travel from, and then write the address. Next step is which address to travel to, as well as defining the desired date and hour of the taxi service. Finally, a mouse-click on Calculate to create the results. To make the program able to interact with the web site as a user, the Python library Selenium was used.

When all the steps mentioned above are performed, the calculator will provide the wanted results: all taxi dispatchers providing their service in the area, as well as the associated price information. When the results are produced, the actual scraping of the web site information will take place. The steps simulated in the web scraper program are the same if you want to collect information on taxi fares in Oslo or Bergen or any other city in Norway, the actual Python code is the same regardless. The only difference is the input needed to determine which area you want price information from, namely the to- and from destinations. This makes it possible to upscale the amount of data quite easily, as only minor changes are needed to get price information from all taxi dispatchers in a given area.

The web scraper of the taxi calculator web site proved promising, unfortunately however the entire web site was shut down shortly after the taxi calculator web scraper had been finalized. The Norwegian Taxi Owners Association has stopped the funding to keep the taxi calculator web site running, and as a result, the web site is not currently active. This means that the web scraper built for this user case sadly cannot be used for its purpose. However, even though this web scraper is not possible to use as of now, the building of the web scraper itself has been a valuable experience. One of the main lessons learned was that almost anything is possible when it comes to collecting data from the web. Programming languages such as Python, and in essence the Selenium driver, has made it possible to scrape data from web pages that require interaction to provide the desired information. The step from collecting data from more static web pages to more dynamic web pages has expanded the horizon regarding web scraping and inspired us to further experiment with the use of web scraping to collect data for the CPI.

ANNEX VII

Web scraping example from Statistics Slovenia (SURS)

At SURS we have developed our custom internet scraping tool which we use to scrape prices from internet shops. This document presents an example of how the tool and process work in the scope of scraping the data for products in ECOICOP classes Information processing equipment and Package holidays. The tool consists of 3 components: the Driver (a human-like behavior simulation that runs on browsers), the Scraper Manager (organizes multiple Drivers, inputs, and outputs to work concurrently), and the Instruction file (specific instructions for work on internet domain).

The general scraping process looks like this:

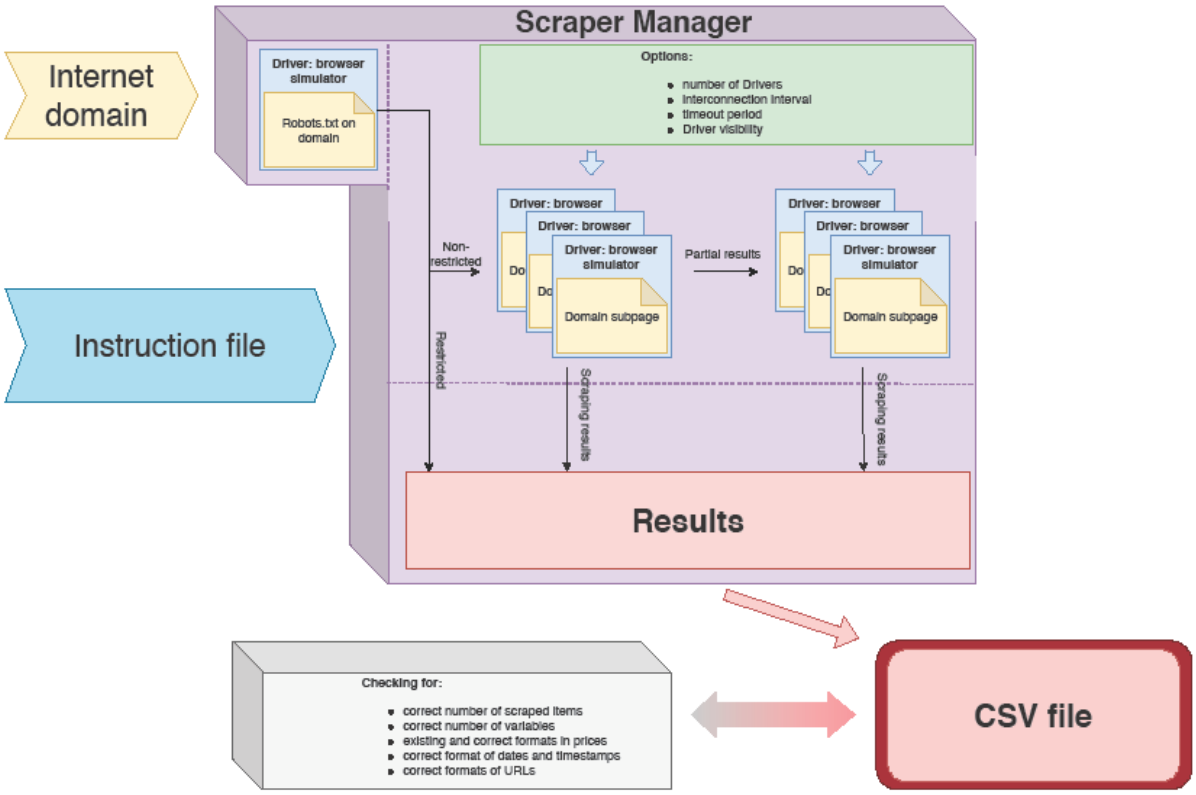


Figure 1: Diagram of price scraping

The actual work that is done behind the scenes when using the Scraper is that the Scraper Manager spawns a web browser, and requests the domain's Robots.txt page and reads the restrictions (can be seen on the figure below).

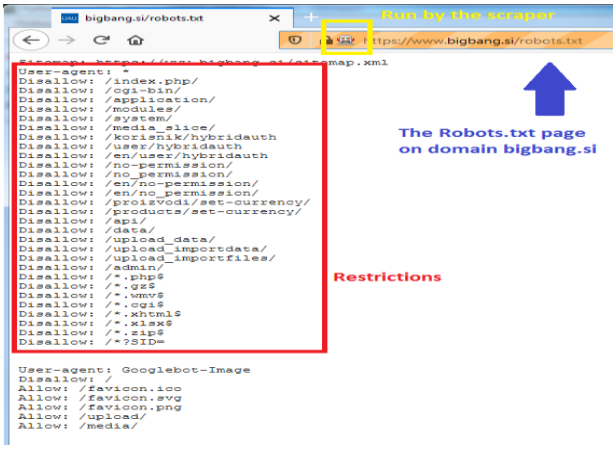


Figure 2: The Robots.txt page of an Internet shop

These are applied in all subsequent work done on this Scraper. The Scraper Manager then initializes multiple Drivers and loads pages into them. The pages might be pre-prepared and/or the Drivers will add them throughout the process of scraping. This is most commonly done due to listed categories of items or pagination. On the next picture below how what we see translates into HTML that the driver sees and uses.

Through use of HTML tags we can export item names, prices, Internet addresses.

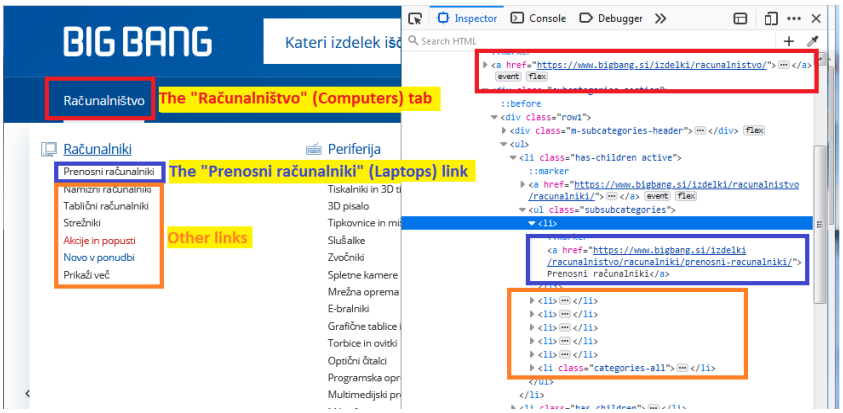


Figure 3: Understanding the structure of HTML

Working in a human simulator also enables us to interact with dynamic pages:

- waiting for HTML elements to become (in) visible, interactive etc.,
- inputting strings into input fields,
- choosing values from dropdown menus,
- loading so-called infinite scrolls,
- and much more.

Below are two examples of interaction on internet pages: pressing buttons and choosing options on a dropdown menu. We can also observe an invisible element in top part of the picture. The driver is able to determine if and when this element becomes visible or even non-existing, and can execute a procedure with such a change. Combining all these capabilities makes our tool quite successful at almost any action needed in the browser.

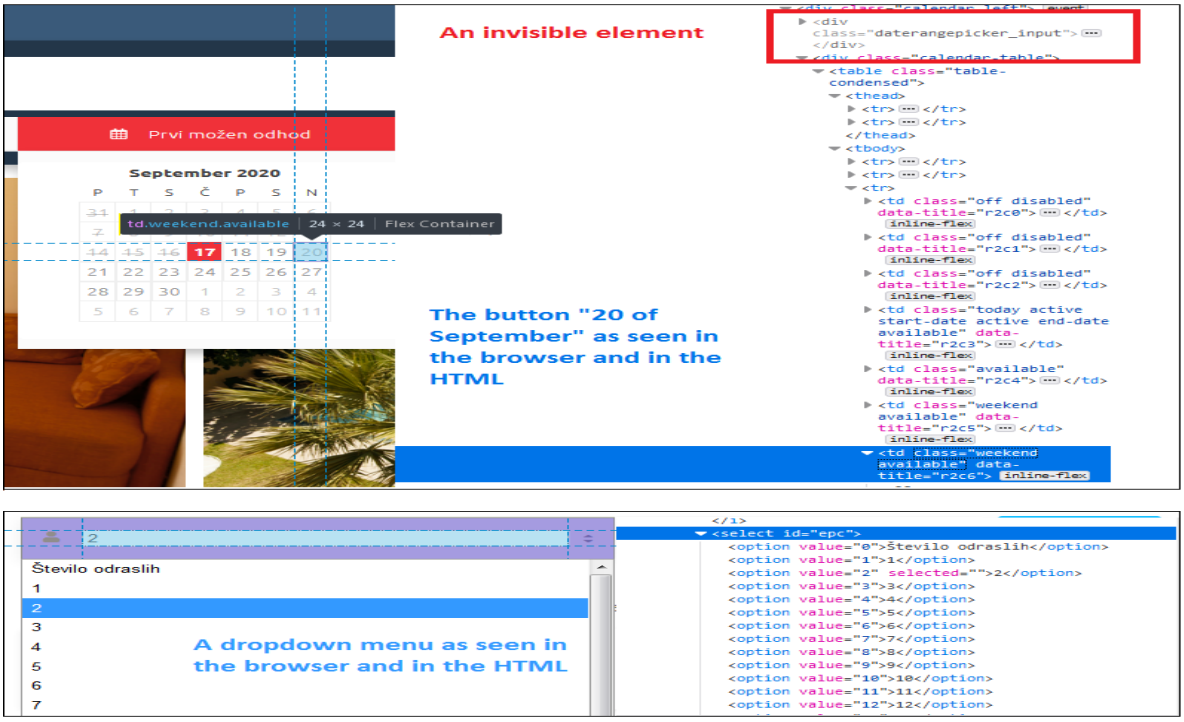


Figure 4: Some interactive elements

Here we can see one actual scraping step:

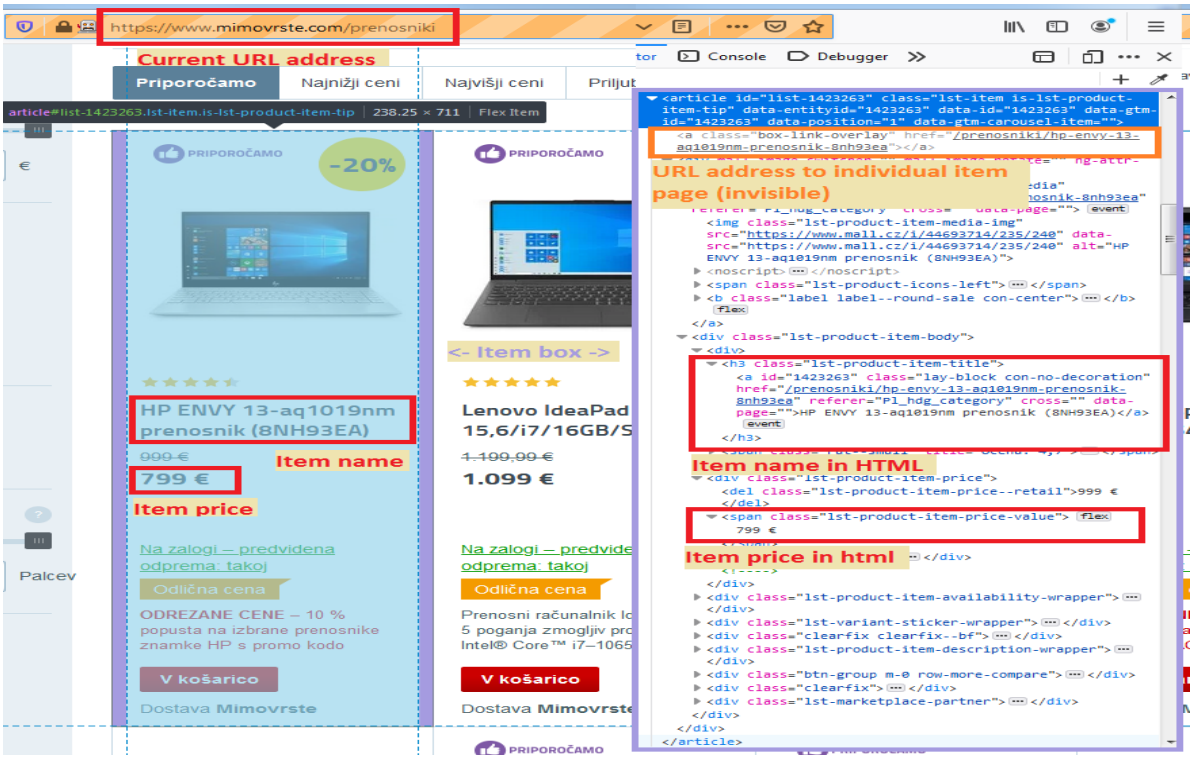


Figure 5: Scraping of information found in a shop

Apart from these 4 values we additionally also record the values of input page and current date and time. We collect these data for every item in 19 different item types in each shop. Afterwards we also collect item characteristics, article IDs used by the shop and "breadcrumbs" (navigation within the shop internet pages, used for ECOICOP classification) on their individual page.

ANNEX VIII

Summary of the 'Use of web scraping in the HICP' questionnaire answers (2020)

Eurostat and ECB have organized a questionnaire to explore the current web scraping practices across the countries. The NSIs submitted their answers during the first half of 2020 and below is presented a summary of the results.

Legal considerations

Most countries report that they have no specific national legal framework that explicitly covers web scraping. In Germany the legislation was updated to permit web scraping for CPI purposes. Following the respective case law is also preferable.

Statistical offices tend to inform the web page owners after checking their requirements (i.e. terms and conditions of use, robots.txt checks etc.). NSIs follow the so-called netiquette to minimize the burden of the servers – for example scraping at night, having a pause between scraping requests, indicating the purpose for the scraping etc.

General information

Some 20 countries of those that filled in the questionnaire are already using web scraping, though a few of them are still in different testing phases so they do not yet apply it in production. Most of the remaining countries also already have plans to introduce web scraping in the near future starting with products like airfares, taxi fares, cleaning of clothing, postal services, furniture, electronic devices etc. Approximately 70% of the countries apply different forms of a parallel price collection for the items they scrape – transaction and scanner data, survey or web questionnaires, tradition price collection or manual online collection.

At items level the split between bulk (40%) and targeted (60%) scraping is relatively equal. Around 2/3 of the scraped items are used in production, while the rest are used in testing or research (some 20% each). This is the list of the most scraped items as reported by the countries:

0312 – Garments
0321 – Shoes and other footware
053 – Household appliances
07112 – Second-hand motor cars
07311 – Passenger transport by train
0733 – Passenger transport by air
08202 – Mobile telephone equipment
09131 – Personal computers
0951 – Books
096 – Package holidays
11201 – Hotels, motels, inns and similar accommodation services

Identification when scraping

Half of the NSIs do not inform the stores that they scrape them, while the rest of the countries either do it, or inform at least some of the stores they scrape. The means of informing are usually by sending an official letter with specific information on the scraping techniques and intentions. Depending on the case, the letter is addressed to the company's general director, CEO or other responsible person for the websites.

Most of the NSIs do not identify their scrapers. The rest would be using different forms of identification so that their scraper user agents could be recognized (i.e. by using an abbreviation, hostname, contact email or the scraper IP address) especially in the cases that they have not already informed directly the website owners.

Technology

The most commonly used computer languages for the scraping are Python (most used libraries/tools are Selenium, Beautiful Soup, Scrapy, Pandas) and R (Rvest, Rselenium) or Java. Others use .NET framework with C# or Visual Basic, as well as different tools like Lynx, ScraT, Import.io or iMacros. Half of the countries are not using API while those that are using APIs have reported mostly the Amadeus airfares one.

Website changes

Approximately 75% of the countries already had to deal with website changes issues. In order to address them properly they developed two types of practices – prevention and monitoring. The preventive approach includes using more stable Xpaths (i.e. with identification of detailed classes, product IDs etc.), and also relying on good relationship with the site owners which inform them in advance about upcoming site changes. Monitoring relates to data observation rules – for example looking into missing values or the number of observations per scraped variable – in cases of differences with the expected values a warning is triggered.

Some problematic cases include inactive websites, web-site structural changes or concrete url/xpath changes. Most of these are fixed by editing the code of the scraper or another update. It seems that it is a good practice to have more than 1 in-house person that knows the language in which the scraper is coded, or to simply have the IT department responsible for its maintenance.

Product selection criteria

The most comment criteria for selection are methodological – representativity and the possibility to define homogenous products. So higher product weights are often selected, as well as representative sellers, which have a considerable amount of online purchases for the given product.

Other considerations seem to be of organizational, technical or pragmatic nature such as:

- comparability with physical stores (so that traditional price collection be fully replaced)
- better quality compared to current collection
- already having manual internet collection
- issues related to creation and maintenance of the scraper
- cost-efficiency (i.e. limited websites with large market shares or large number of products per store)
- volatility (need of a lot of prices to improve quality, or having dynamic prices or often changing items)

Frequency of scraping and number of scraped shops/products

Almost all reported items have 3 or less shops scraped. Still there are some products like clothing, personal computers, consumer electronics, mobile phones for which there are cases of even more than different 10 scraped sites. In the case of targeted scraping, the average number of different scraped products is around 20 per product category. However, there are also cases in which websites contain offers for a huge variety of products, so some countries report instances in which a much higher number of products is scraped. Often it is not the whole sample that is later used in production.

The number of scraped prices ranges between 100 prices per day to 60 000 per month and it seems to depend on the item specificity. The frequency of scraping also differs a lot as countries report relatively equal shares of items being scraped on daily, weekly or monthly basis.

Classification and tracking of same products

In many of the cases, countries report that classification is not needed (i.e. in cases of target scraping, or where an index is aggregated for the whole website). In the occasions where it is needed it is most often done manually or at least verified by a statistician. When it is done automatically, it is usually based on a predefined set of ECOICOP mapping rules or on some item-specific characteristics which are available on the website.

In the cases where countries need to track the same products they are doing this mostly using URL, search path or Xpath. When available, the NSIs would follow the GTIN of the products or some other item ID code, which the shops could be using. NSIs often select item codes (or models) that are then tracked over time. Another strategy is to follow homogenous products that are defined with respect to some attributes and to assign the scraped item codes or models to these homogenous products.

Aggregation to elementary index

The most commonly reported approach is using an unweighted geometric average (Jevons) of the collected prices, though some are using weighted averages too. Indices are then weighted using Laspeyres aggregation to the corresponding COICOP level. Some NSIs are also using hedonic methods. Many countries are still into testing phase so they do not yet calculate indices.