economie
FPS Economy, S.M.E.s, Self-employed and Energy

An introduction to web scraping methods

Ken Van Loon
Statistics Belgium

UN GWG on Big Data for Official Statistics
Training workshop on scanner and on-line data
6-7 November 2017
Bogota, Colombia

Background

- Me
  - Statistician?
  - Mostly working on price statistics (consumer price indices/residential property indices)
  - Methodological issues (incl. scanner data and web scraping)

- Web scraping at Statistics Belgium
  - We have around 60 scripts running (some implemented others in test/research phase)
  - Currently we scrape data for the following segments:
    - Clothing
    - Footwear
    - Hotels
    - Airfares
    - Train tickets
    - Second-hand cars
    - Department stores
    - Books
    - DVD & Blu-ray
    - Video games
    - Consumer electronics
    - Student rooms
    - Supermarkets
    - …

Contents

- What is web scraping?
- HTML – CSS Selectors
- SelectorGadget
- Web scraping in R
  - Rvest
  - Scrape functions
  - RSelenium
- Experimental indices
- Monitoring tools

Definition

What is web scraping?

*Web scraping focuses on the **transformation of unstructured data** on the web, typically in **HTML** format, into structured data that can be stored and analyzed in*
***a central local database or spreadsheet**.*

(wikipedia ☺)

A technique to collect (scrape) data from the web automatically.

Implement web scraping:

- Programming skills
- Data collection
- Data processing

HTML

Webpages consist of HTML code/tags:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<a href="http://www.google.com">This is a link</a>

</body>
</html>
```

Knowing HTML is not really necessary for web scraping, but will make life easier!

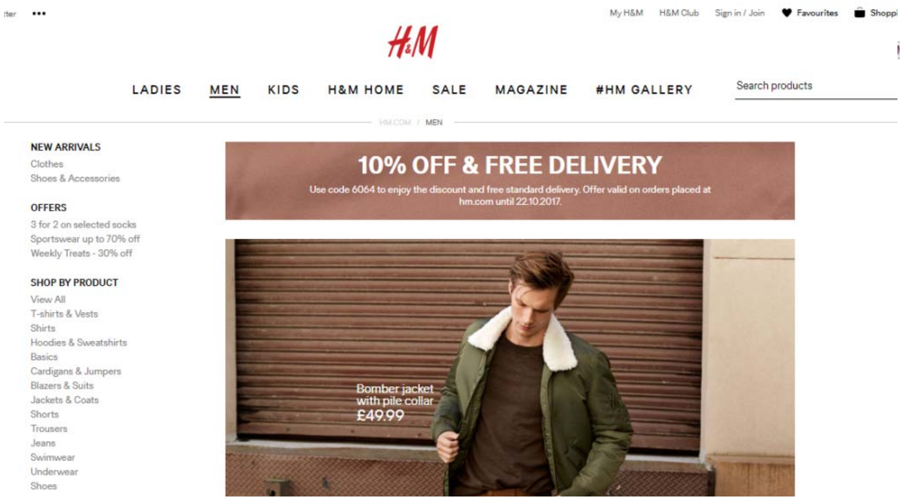
HTML

How to select the specific information on an HTML-page?

- Xpath – XML Path Language (query language for XML)
- **CSS selectors** – used to select elements you want to style

  → focus here on CSS selectors (personal opinion: more readable then Xpath)

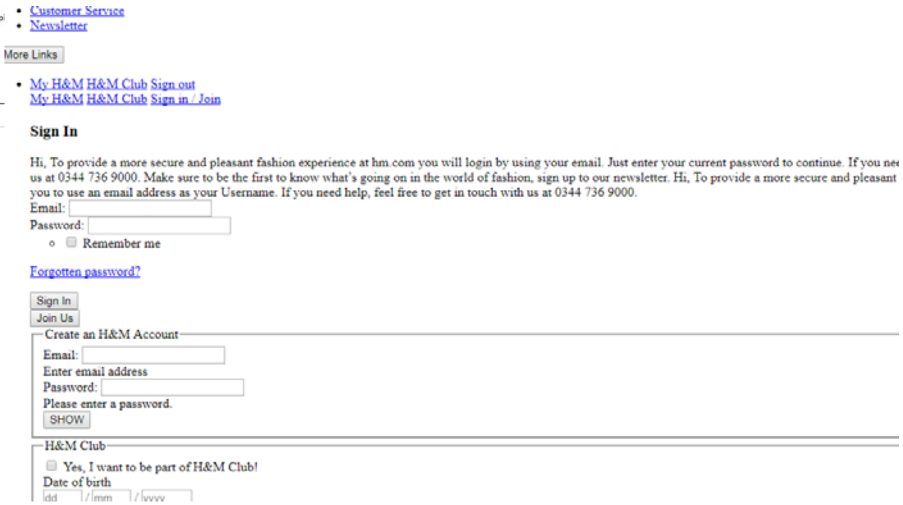
Cascading Style Sheets (**CSS**) are used to style websites:



10% OFF & FREE DELIVERY

Use code 6064 to enjoy the discount and free standard delivery. Offer valid on orders placed at hm.com until 22.10.2017.

with CSS



- Customer Service
- Newsletter

More Links

- My H&M H&M Club Sign out
  My H&M H&M Club Sign in / Join

Sign In

Hi, To provide a more secure and pleasant fashion experience at hm.com you will login by using your email. Just enter your current password to continue. If you nee
us at 0344 736 9000. Make sure to be the first to know what's going on in the world of fashion, sign up to our newsletter. Hi, To provide a more secure and pleasant
you to use an email address as your Username. If you need help, feel free to get in touch with us at 0344 736 9000.

Email:
Password:
○ □ Remember me

Forgotten password?

Sign In
Join Us

Create an H&M Account
Email:
Enter email address
Password:
Please enter a password.
SHOW

H&M Club
□ Yes, I want to be part of H&M Club!
Date of birth
dd / mm / yyyy

without CSS

HTML

How to select the specific information on an HTML-page?

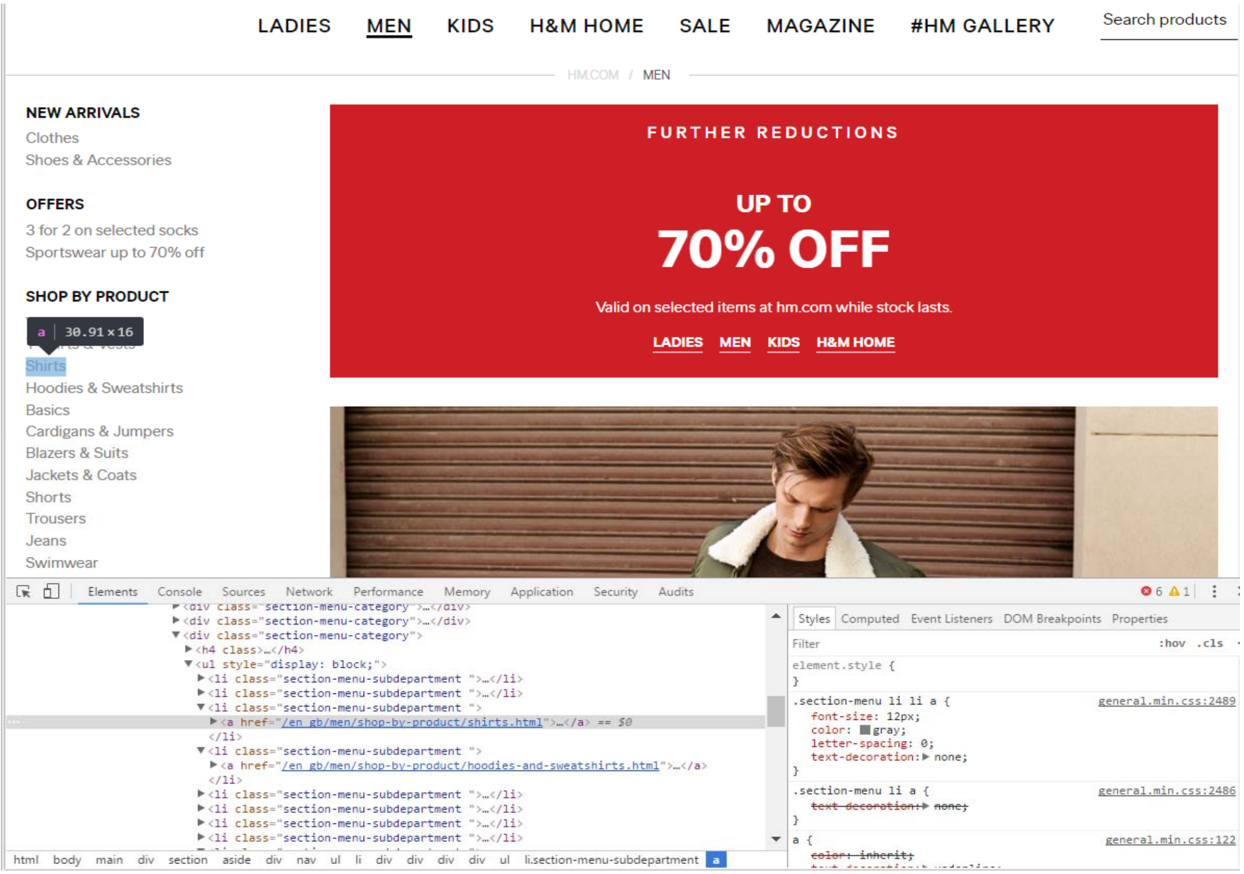
Tag selection in Chrome:

- Right click:
- Inspect screen
- (developer tools)

- Right click:
- Copy
- Copy selector

Also using ctrl + f:

- Just search for the tag



NEW ARRIVALS
Clothes
Shoes & Accessories

OFFERS
3 for 2 on selected socks
Sportswear up to 70% off

SHOP BY PRODUCT
a | 30.91 × 16
Shirts
Hoodies & Sweatshirts
Basics
Cardigans & Jumpers
Blazers & Suits
Jackets & Coats
Shorts
Trousers
Jeans
Swimwear

FURTHER REDUCTIONS

UP TO
70% OFF

Valid on selected items at hm.com while stock lasts.

LADIES MEN KIDS H&M HOME

```
▶<div class="section-menu-category">…</div>
▶<div class="section-menu-category">…</div>
▼<div class="section-menu-category">
  ▶<h4 class>…</h4>
  ▼<ul style="display: block;">
    ▶<li class="section-menu-subdepartment ">…</li>
    ▶<li class="section-menu-subdepartment ">…</li>
    ▼<li class="section-menu-subdepartment ">
      ▶<a href="/en_gb/men/shop-by-product/shirts.html">…</a> == $0
      </li>
    ▼<li class="section-menu-subdepartment ">
      ▶<a href="/en_gb/men/shop-by-product/hoodies-and-sweatshirts.html">…</a>
      </li>
    ▶<li class="section-menu-subdepartment ">…</li>
    ▶<li class="section-menu-subdepartment ">…</li>
    ▶<li class="section-menu-subdepartment ">…</li>
    ▶<li class="section-menu-subdepartment ">…</li>
html  body  main  div  section  aside  div  nav  ul  li  div  div  div  div  ul  li.section-menu-subdepartment  a
```

```
element.style {
}
.section-menu li li a {                general.min.css:2489
   font-size: 12px;
   color: ■gray;
   letter-spacing: 0;
   text-decoration:▶ none;
}
.section-menu li a {                   general.min.css:2486
   ~~text-decoration:▶ none;~~
}
a {                                    general.min.css:122
   ~~color: inherit;~~
```

HTML

How to select the specific information on an HTML-page?

- Using Chrome is quite tedious
- Would be much easier to have a point-and-click interface to select what you want
  - Luckily someone developed this ☺
    (Remember: you can always verify what you select with the CSS selector using Chrome developer tools)
- But before using the point-and-click interface we'll learn a bit how CSS work… by playing a game

CSS selectors

CSS Diner: https://flukeout.github.io/

```
<div class="table">
    <plate/>
    <plate/>
</div>
```

CSS selector “plate”: selects all plate elements

```
<div class="table">
    <bento/>
    <plate/>
    <bento/>
</div>
```

CSS selector “bento”: selects all bento tags

```
<div class="table">
        <plate id="fancy"/>
        <plate/>
        <bento/>
</div>
```

CSS selector “**#**fancy”: selects each tag with id=“fancy”.

“ **#** “ indicates the “id=”-tag

```
<div class="table">
        <apple/>
        <apple class="small"/>
        <plate>
                <banana class=“small"/>
        <plate/>
        <plate/>
</div>
```

CSS selector “**.**small”: selects each tag with class=“small”.
“ **.** ” indicates the “class=“-tag

```
<div class="table">
        <bento/>
        <plate>
             <apple/>
        <plate/>
        <apple/>
</div>
```

CSS selector “plate apple”: selects the apple tag within the plate tag

```
<div class="table">
        <bento>
                 <orange/>
        <bento/>
        <plate id="fancy">
                 <pickle/>
        <plate/>
        <plate>
                 <pickle/>
        <plate/>
</div>
```

CSS selector “#fancy pickle”: selects the tag pickle within the plate tag with id=fancy

```
<div class="table">
        <apple/>
        <apple class="small"/>
        <bento>
                <orange class=“large"/>
        <bento/>
        <plate>
                <orange/>
        <plate/>
        <plate>
                <orange class="small"/>
        <plate/>
</div>
```

CSS selector “orange.small”: selects the orange with class=small

```
<div class="table">
        <pickle class="small"/>
        <pickle/>
        <plate>
                <pickle/>
        <plate/>
        <bento>
                <pickle/>
        <bento/>
        <plate>
                <pickle/>
        <plate/>
        <pickle/>
        <pickle class="small"/>
</div>
```

CSS selector “plate, bento”: selects all plate and bento tags

```
<div class="table">
        <plate id="fancy">
                <orange class="small"/>
        <plate/>
        <plate>
                <pickle/>
        <plate/>
        <apple class="small"/>
        <plate>
                <apple/>
        <plate/>
</div>
```

CSS selector “plate *”: selects everything which includes plate (incl. subtags)

```
<div class="table">
<bento>
            <apple class="small"/>
<bento/>
<plate/>
<apple class="small"/>
<plate/>
<apple/>
<apple class="small"/>
<apple class="small"/>
</div>
```

CSS selector “plate + apple”: selects all apple tags directly following a plate tag, only the first apple tag after a plate tag is selected

“ **+** “ selects all tags directly (= first tag) following a specified tag

Note: with “ + “ only the first tag is selected

```
<div class="table">
        <pickle/>
        <bento>
                <orange class="small"/>
        <bento/>
        <pickle class="small"/>
        <pickle/>
        <plate>
                <pickle/>
        <plate/>
        <plate>
                <pickle class="small"/>
        <plate/>
</div>
```

CSS selector “bento ~ pickle”: selects all matching pickle tags after the bento tag

“ **~** ” selects all matching tags at the same level after the first specified tag

Note: with “ ~ “ all tags at the same level are selected, not only the first one (cfr. +)

```
<div class="table">
    <plate>
        <bento>
            <apple/>
        <bento/>
    <plate/>
    <plate>
      <apple/>
      <apple/>
    </plate>
    <plate>
      <apple/>
    <plate/>
    <apple/>
    <apple class="small"/>
</div>
```

CSS selector “plate > apple”: selects all apple tags which follow a plate tag at another level

“ **>** ” selects all direct ‘children’ of an element

economie
FPS Economy, S.M.E.s, Self-employed and Energy

Overview CSS selectors

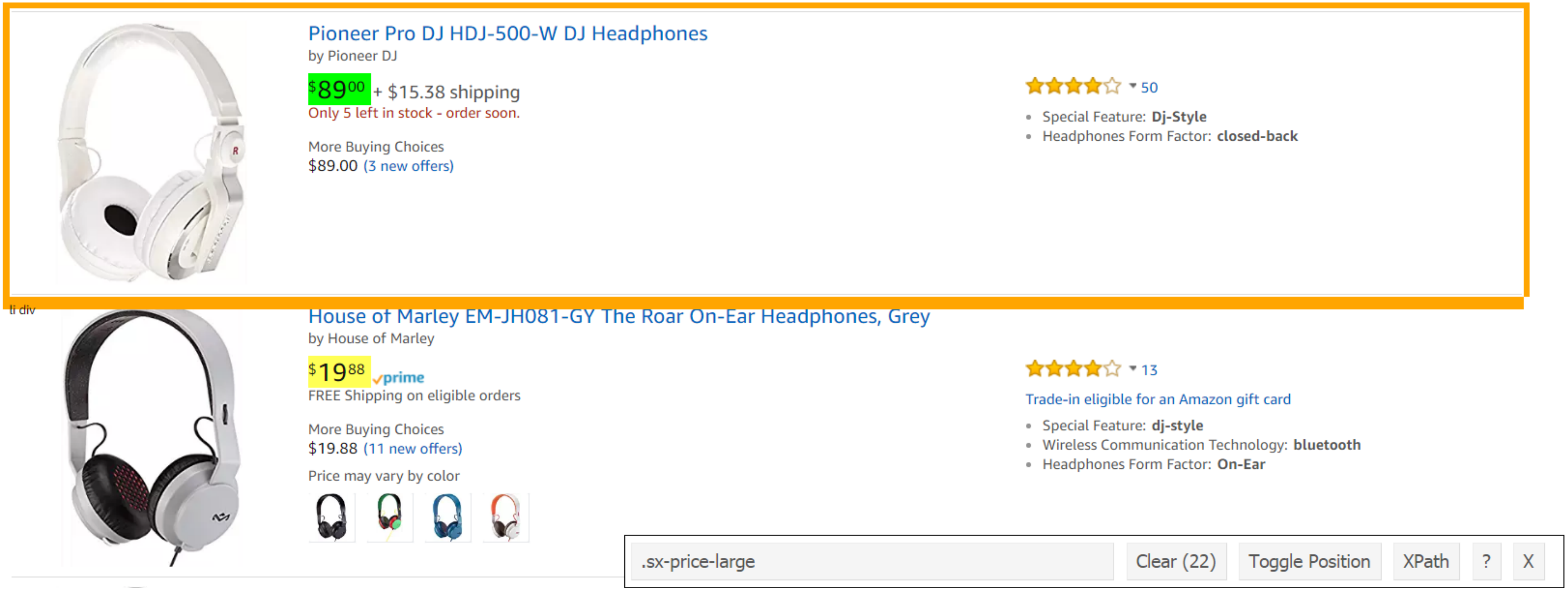
#	Selector	Example	Description
1	element	a	Selects all “a” tags
2	.class	.price	Selects all elements with the class=”price”
3	#id	#content	Selects all elements, in theory only one, with the id=”content”
4	element element	div a	Selects all “a” tags inside all “div” elements
5.a	.class element	.price a	Selects all “a” tags inside all elements with the class=”price”
5.b	#id element	#content a	Selects all “a” tags inside all elements with the id=”content”
6	element.class	div.price	Selects all “div” tags with the class=”price”
7	element, element	div, p	Selects all “div” tags and all “p” tags
8	element *	div *	Selects all elements within the “div” element
9	element+element	div+p	Selects “p” elements that follow directly after the “div” element (on the same level)
10	element~element	div~p	Selects all “p” elements that follow after the “div” element (on the same level)
11	element>element	div>p	Selects all “p” elements that are direct children of the “div” element
12	[attribute=”value”]	[size=”small”]	Selects all elements with size="small"

SelectorGadget

Identify CSS selectors with point-and-click interface

- Chrome extension
- Click on element you want to select
  - Selected item: marked in green
  - SelectorGadget makes a guess and marks all elements that matches the selector in yellow
- Deselect wrong elements: marked red
- CSS Selector / Tag can be used in programming languages such as R

SelectorGadget



Pioneer Pro DJ HDJ-500-W DJ Headphones

by Pioneer DJ

$89 00 + $15.38 shipping

Only 5 left in stock - order soon.

More Buying Choices

$89.00 (3 new offers)

50

- Special Feature: **Dj-Style**
- Headphones Form Factor: **closed-back**

li div

House of Marley EM-JH081-GY The Roar On-Ear Headphones, Grey

by House of Marley

$19 88 prime

FREE Shipping on eligible orders

More Buying Choices

$19.88 (11 new offers)

Price may vary by color

13

Trade-in eligible for an Amazon gift card

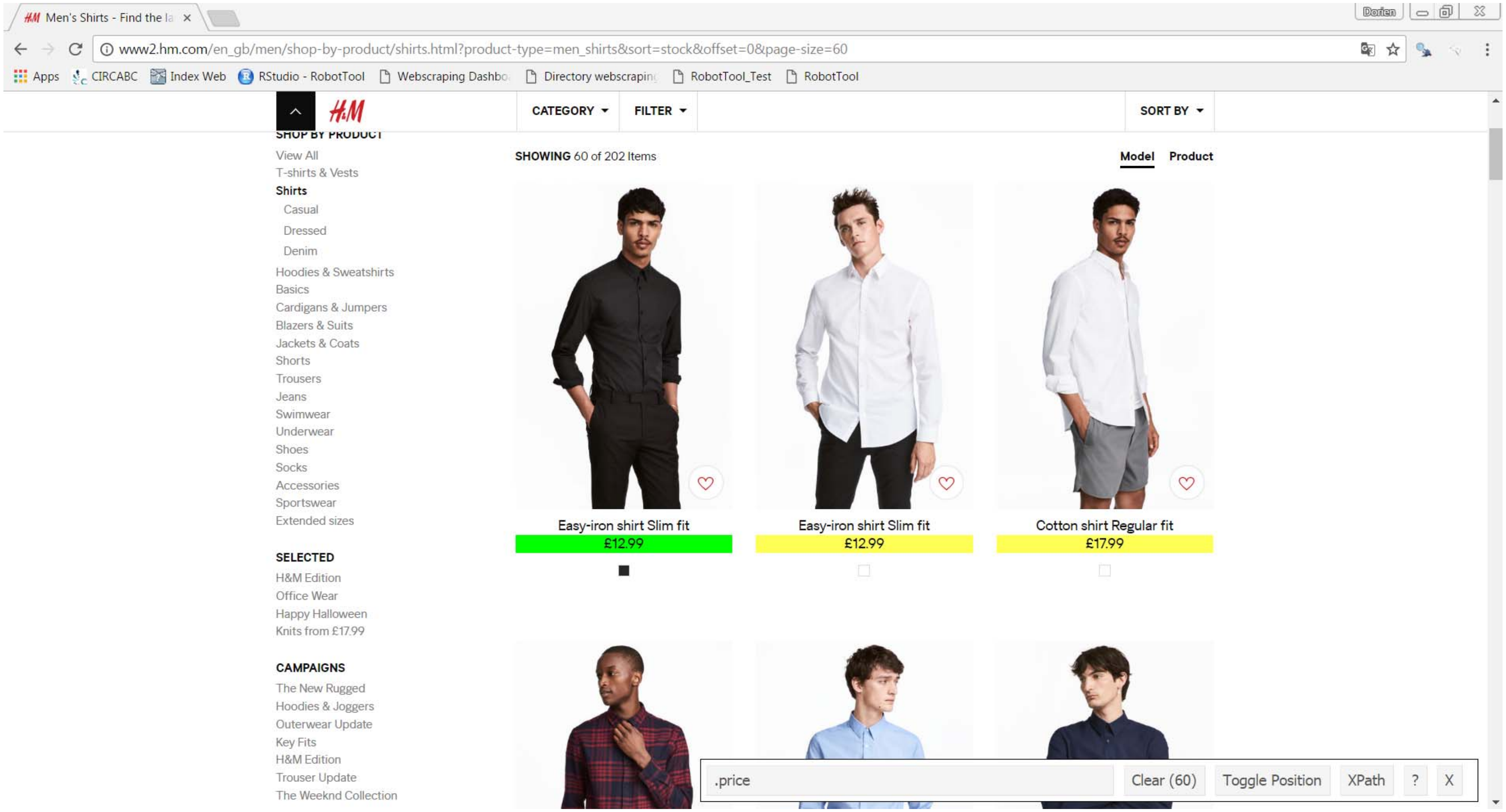
- Special Feature: **dj-style**
- Wireless Communication Technology: **bluetooth**
- Headphones Form Factor: **On-Ear**

.sx-price-large

To select the price on this website: use CSS selector ".sx-price-large"

SelectorGadget

Select element (price) on website → ".price"

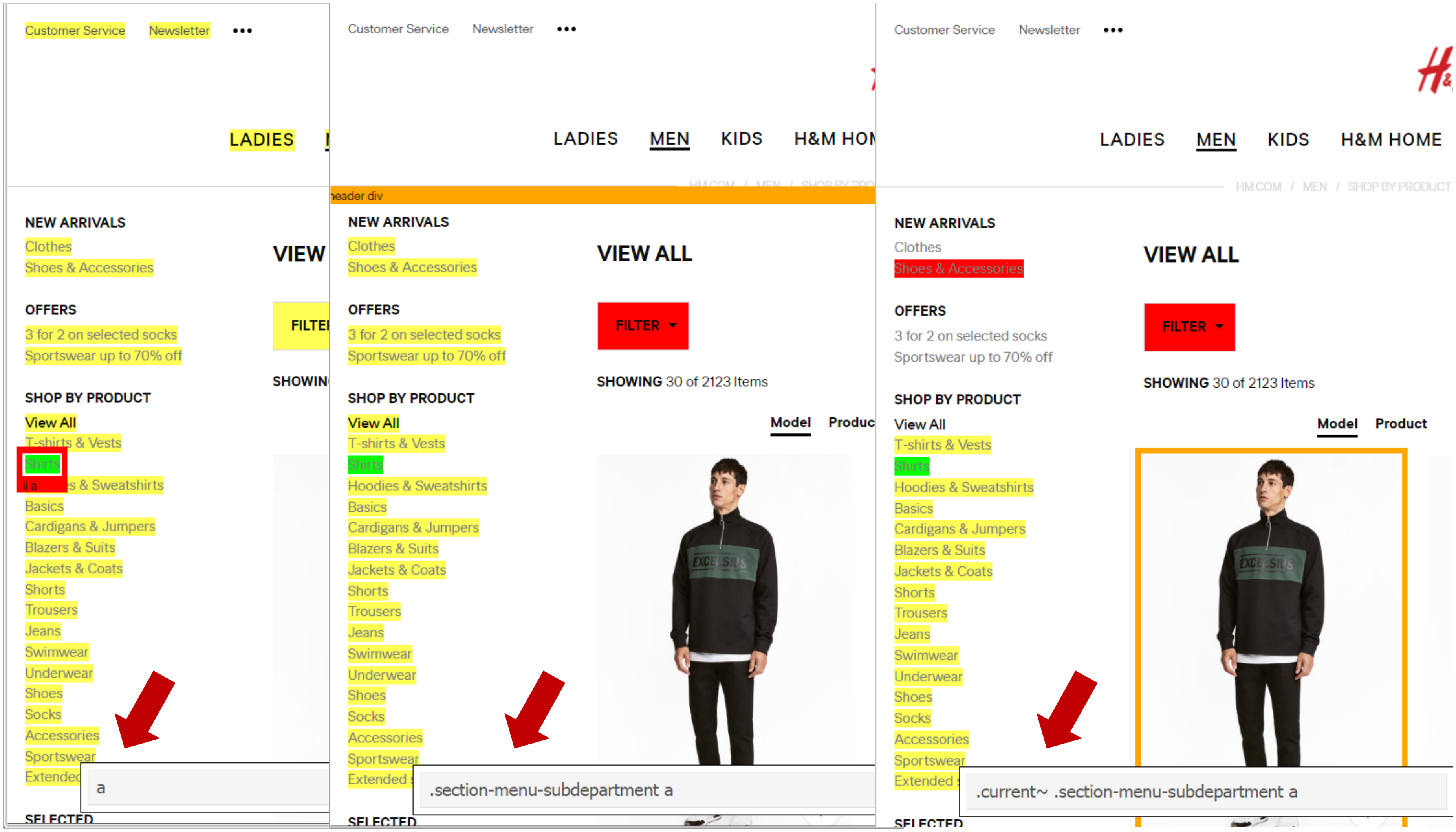


www2.hm.com/en_gb/men/shop-by-product/shirts.html?product-type=men_shirts&sort=stock&offset=0&page-size=60

.price

SelectorGadget

Deselect undesirable elements:



a

.section-menu-subdepartment a

.current~ .section-menu-subdepartment a

Web scraping with R

Web scraping with R

Use rvest package developed by Hadley Wickham (Chief Scientist at RStudio)

install.packages("rvest")

Most important functions:

- read_html(): creates an html document from a webpage
  - Without a proxy: e.g. read_html(“https://www.google.com”)
  - With a proxy: e.g. read_html(httr::GET(url,user_agent(agent), proxy))
- html_nodes(): select tags
  - e.g. html_nodes(“.sx-price-large”)
- html_node(): selects exactly one tag
  - e.g. html_node(“.sx-price-large”) will select only the first tag instead of all matching tags

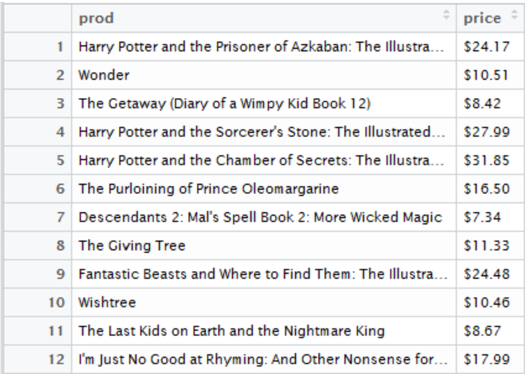
Web scraping with R

Most important functions (continued):

- html_text(): extracts text within tags, to be used after html_node(s)()
  - e.g. html_nodes(“.sx-price-large”) %>% html_text()
- html_attr(): extracts the value of the attribute, to be used after html_node(s)()
  - e.g. html_nodes(“a”) %>% html_attr(“href”) will select the url
- html_table(): extracts a table, to be used after html_node(s)()
  - e.g. html_node(“table css”) %>% html_table()
- All functions can be chained using the %>% (a.k.a. pipe) operator
  - e.g. read_html(“url”) %>% html_nodes(“css”) %>% html_text()

Simple example

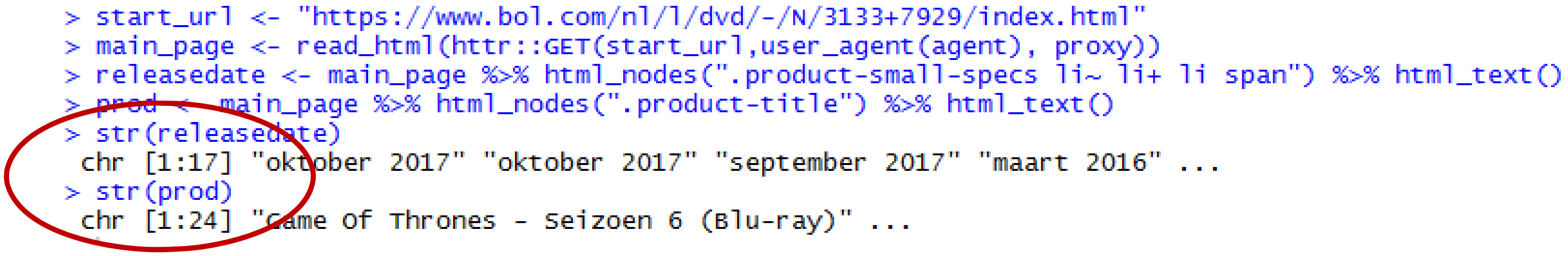
```
library(rvest)
library(stringr)

#url
start_url <- "https://www.amazon.com/s/ref=sr_nr_p_n_feature_browse-
b_1?fst=as%3Aoff&rh=n%3A283155%2Cn%3A%211000%2Cn%3A4%2Cp_n_feature_five_browse-
bin%3A2579000011%2Cp_n_feature_five_browse-bin%3A6118393011%2Cp_n_feature_browse-
bin%3A2656020011&bbn=4&ie=UTF8&qid=1507885684&rnid=618072011"

#load html page
main_page <- read_html(start_url)

#scrape price
price <- main_page %>% html_nodes(".sx-price-large") %>% html_text()
price <- str_trim(price)
price <- str_replace_all(price,"\n                            ", ".")

#scrape product name
prod <- main_page %>% html_nodes(".s-access-title") %>% html_text()

#store scraped data in data frame
data <- data.frame(prod=prod, price=price)
```



	prod	price
1	Harry Potter and the Prisoner of Azkaban: The Illustra...	$24.17
2	Wonder	$10.51
3	The Getaway (Diary of a Wimpy Kid Book 12)	$8.42
4	Harry Potter and the Sorcerer's Stone: The Illustrated...	$27.99
5	Harry Potter and the Chamber of Secrets: The Illustra...	$31.85
6	The Purloining of Prince Oleomargarine	$16.50
7	Descendants 2: Mal's Spell Book 2: More Wicked Magic	$7.34
8	The Giving Tree	$11.33
9	Fantastic Beasts and Where to Find Them: The Illustra...	$24.48
10	Wishtree	$10.46
11	The Last Kids on Earth and the Nightmare King	$8.67
12	I'm Just No Good at Rhyming: And Other Nonsense for...	$17.99

Simple example

```
library(rvest)


#url
start_url <- "https://www.bol.com/nl/l/dvd/-/N/3133+7929/index.html"

#load html page
main_page <- read_html(start_url)

#scrape release date
releasedate <- main_page %>% html_nodes(".product-small-specs li~ li+ li span") %>% html_text()

#scrape product name
prod <- main_page %>% html_nodes(".product-title") %>% html_text()

#check length of scraped data
str(releasedate)
str(prod)
```



```
> start_url <- "https://www.bol.com/nl/l/dvd/-/N/3133+7929/index.html"
> main_page <- read_html(httr::GET(start_url,user_agent(agent), proxy))
> releasedate <- main_page %>% html_nodes(".product-small-specs li~ li+ li span") %>% html_text()
> prod <- main_page %>% html_nodes(".product-title") %>% html_text()
> str(releasedate)
 chr [1:17] "oktober 2017" "oktober 2017" "september 2017" "maart 2016" ...
> str(prod)
 chr [1:24] "Game Of Thrones - Seizoen 6 (Blu-ray)" ...
```

≠

Web scraping with R

Problem: Missing values

e.g. number of prices ≠ number of products

Solution: Scrape Functions:

- scrape_css

```
scrape_css <- function(css, group) {
            txt <- main_page %>% html_nodes(group) %>% lapply(. %>% html_nodes(css) %>%
            html_text() %>% ifelse(identical(., character(0)), NA, .)) %>% unlist
            return(txt)
            }
```

- scrape_css_attr

```
scrape_css_attr <- function(css, group, attribute) {
                   txt <- main_page %>% html_nodes(group) %>% lapply(. %>% html_nodes(css) %>%
                   html_attr(attribute) %>% ifelse(identical(., character(0)), NA, .)) %>% unlist
                   return(txt)
                   }
```

Web scraping with R

Scrape functions

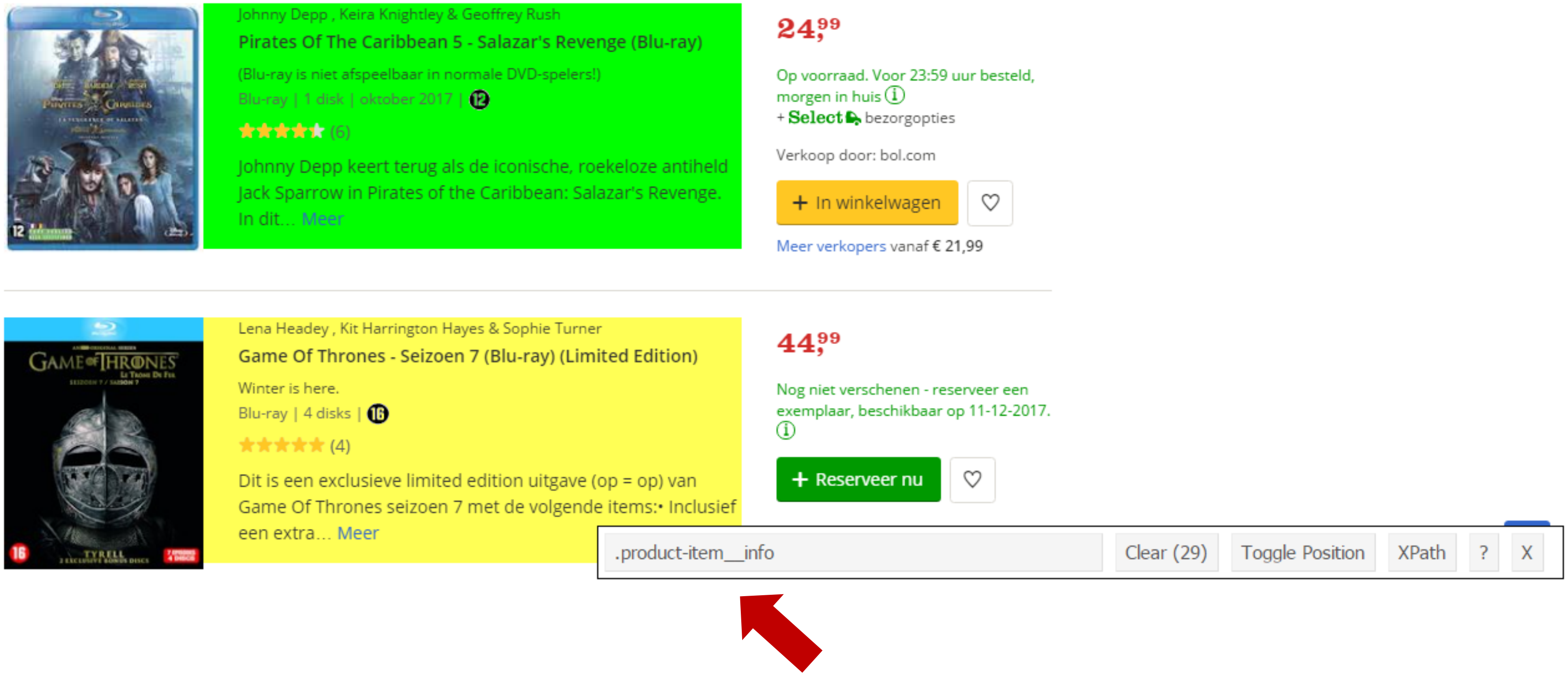
- scrape_css(css, group)
- scrape_css_attr(css, group, attribute)

  - css: specific element we want to scrape
  - group: refers to CSS selector that captures the whole observation including subcomponents in which we are interested
  - attribute: specific attribute we want to scrape (e.g. url)

Web scraping with R

Scrape functions

- scrape_css(css, **group**)
- scrape_css_attr(css, **group**, attribute)



Johnny Depp , Keira Knightley & Geoffrey Rush

Pirates Of The Caribbean 5 - Salazar's Revenge (Blu-ray)

(Blu-ray is niet afspeelbaar in normale DVD-spelers!)

Blu-ray | 1 disk | oktober 2017 | 12

★★★★★ (6)

Johnny Depp keert terug als de iconische, roekeloze antiheld Jack Sparrow in Pirates of the Caribbean: Salazar's Revenge. In dit... Meer

24,99

Op voorraad. Voor 23:59 uur besteld, morgen in huis ⓘ

+ Select bezorgopties

Verkoop door: bol.com

Meer verkopers vanaf € 21,99

Lena Headey , Kit Harrington Hayes & Sophie Turner

Game Of Thrones - Seizoen 7 (Blu-ray) (Limited Edition)

Winter is here.

Blu-ray | 4 disks | 16

★★★★★ (4)

Dit is een exclusieve limited edition uitgave (op = op) van Game Of Thrones seizoen 7 met de volgende items:• Inclusief een extra... Meer

44,99

Nog niet verschenen - reserveer een exemplaar, beschikbaar op 11-12-2017. ⓘ

.product-item__info

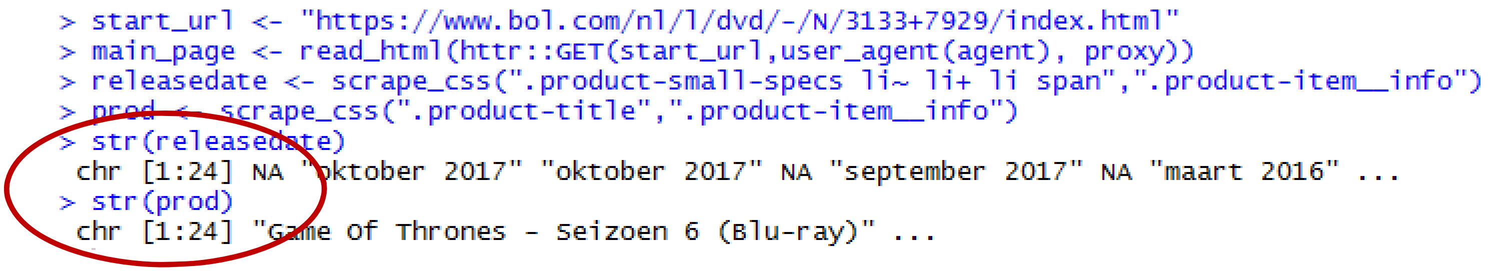
Simple example

```
library(rvest)

#url
start_url <- "https://www.bol.com/nl/l/dvd/-/N/3133+7929/index.html"

#load html page
main_page <- read_html(start_url)

#scrape release date
releasedate <- scrape_css(".product-small-specs li~ li+ li span",".product-item__info")

#scrape product name
prod <- scrape_css(".product-title",".product-item__info")

#check length of scraped data
str(releasedate)
str(prod)
```

group

group

=



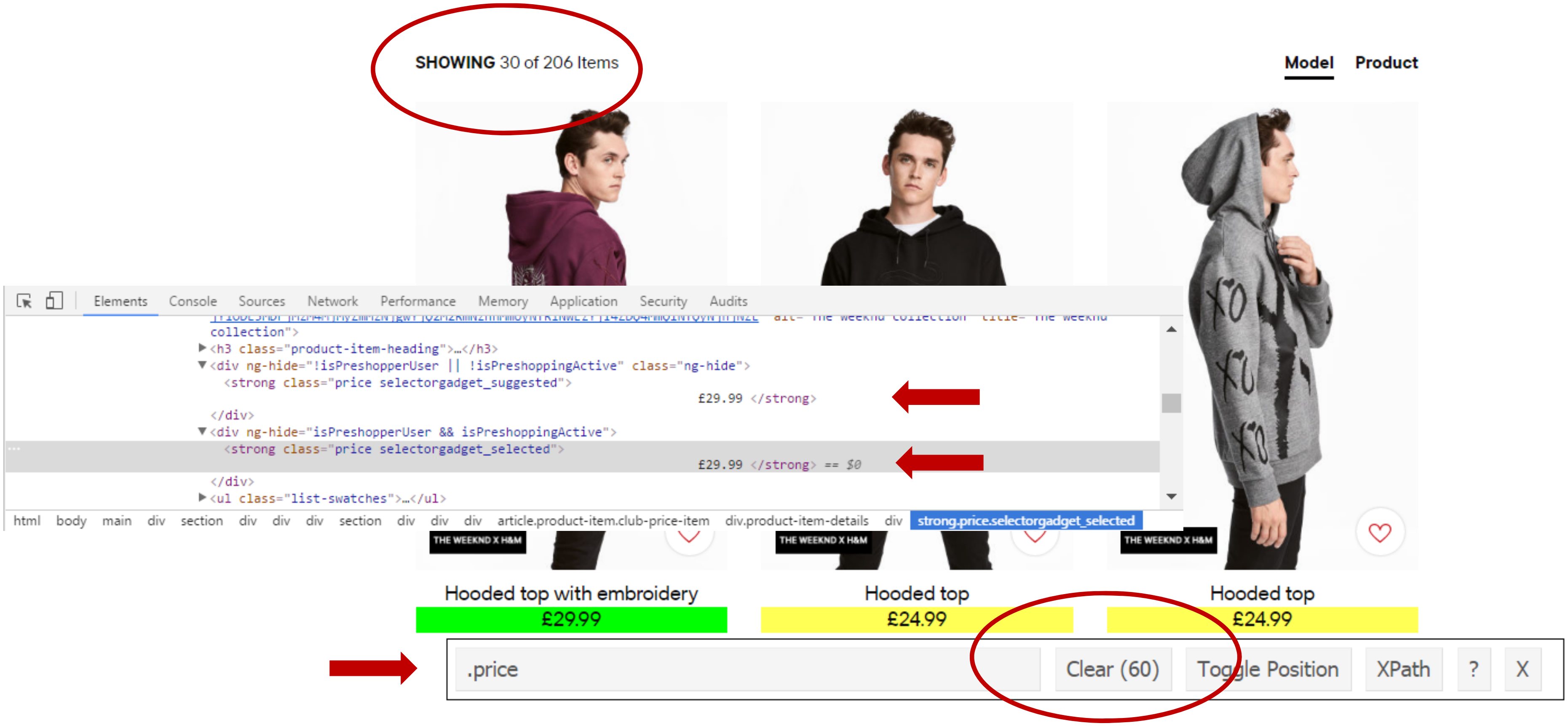
```
> start_url <- "https://www.bol.com/nl/l/dvd/-/N/3133+7929/index.html"
> main_page <- read_html(httr::GET(start_url,user_agent(agent), proxy))
> releasedate <- scrape_css(".product-small-specs li~ li+ li span",".product-item__info")
> prod <- scrape_css(".product-title",".product-item__info")
> str(releasedate)
 chr [1:24] NA "oktober 2017" "oktober 2017" NA "september 2017" NA "maart 2016" ...
> str(prod)
 chr [1:24] "Game Of Thrones - Seizoen 6 (Blu-ray)" ...
```

Simple example

```
library(rvest)
library(stringr)

#url
start_url <- "http://www2.hm.com/en_gb/men/shop-by-product/shirts.html"

#load html page
main_page <- read_html(start_url)

#scrape price
price <- main_page %>% html_nodes(“.price") %>% html_text()
price <- str_trim(price)

#scrape product name
prod <- main_page %>% html_nodes(".product-item-heading a") %>% html_text()

#store scraped data in data frame
data <- data.frame(prod=prod, price=price)
```

Simple example

```
#scrape price
price <- main_page %>% html_nodes(“.price") %>% html_text()
```

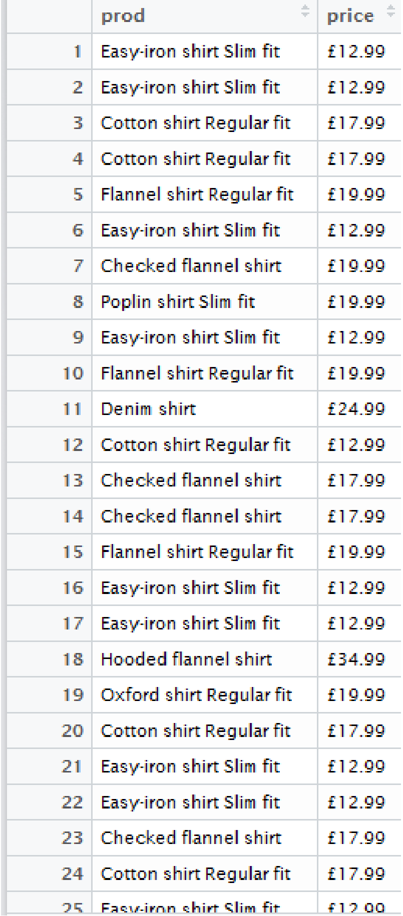


```
    collection">
  ▶<h3 class="product-item-heading">…</h3>
  ▼<div ng-hide="!isPreshopperUser || !isPreshoppingActive" class="ng-hide">
     <strong class="price selectorgadget_suggested">
                                                     £29.99 </strong>
   </div>
  ▼<div ng-hide="isPreshopperUser && isPreshoppingActive">
     <strong class="price selectorgadget_selected">
                                                     £29.99 </strong> == $0
   </div>
  ▶<ul class="list-swatches">…</ul>
```

html body main div section div div div section div div div article.product-item.club-price-item div.product-item-details div strong.price.selectorgadget_selected
THE WEEKND X H&M
THE WEEKND X H&M
THE WEEKND X H&M
Hooded top with embroidery
£29.99
Hooded top
£24.99
Hooded top
£24.99
.price
Clear (60)
Toggle Position
XPath
?
X

Simple example

```
library(rvest)
library(stringr)

#url
start_url <- "http://www2.hm.com/en_gb/men/shop-by-product/shirts.html"

#load html page
main_page <- read_html(start_url)

#scrape price
price <- main_page %>% html_nodes(“.product-item-details div ~ div .price")
            %>% html_text()
price <- str_trim(price)

#scrape product name
prod <- main_page %>% html_nodes(".product-item-heading a") %>% html_text()

#store scraped data in data frame
data <- data.frame(prod=prod, price=price)


#Alternative scrape functions
price <- scrape_css(“.ng-hide .price", ".product-item-details")
prod <- scrape_css(".product-item-heading a", ".product-item-details")
```



	prod	price
1	Easy-iron shirt Slim fit	£12.99
2	Easy-iron shirt Slim fit	£12.99
3	Cotton shirt Regular fit	£17.99
4	Cotton shirt Regular fit	£17.99
5	Flannel shirt Regular fit	£19.99
6	Easy-iron shirt Slim fit	£12.99
7	Checked flannel shirt	£19.99
8	Poplin shirt Slim fit	£19.99
9	Easy-iron shirt Slim fit	£12.99
10	Flannel shirt Regular fit	£19.99
11	Denim shirt	£24.99
12	Cotton shirt Regular fit	£12.99
13	Checked flannel shirt	£17.99
14	Checked flannel shirt	£17.99
15	Flannel shirt Regular fit	£19.99
16	Easy-iron shirt Slim fit	£12.99
17	Easy-iron shirt Slim fit	£12.99
18	Hooded flannel shirt	£34.99
19	Oxford shirt Regular fit	£19.99
20	Cotton shirt Regular fit	£17.99
21	Easy-iron shirt Slim fit	£12.99
22	Easy-iron shirt Slim fit	£12.99
23	Checked flannel shirt	£17.99
24	Cotton shirt Regular fit	£17.99
25	Easy-iron shirt Slim fit	£12.99

Scraping strategy

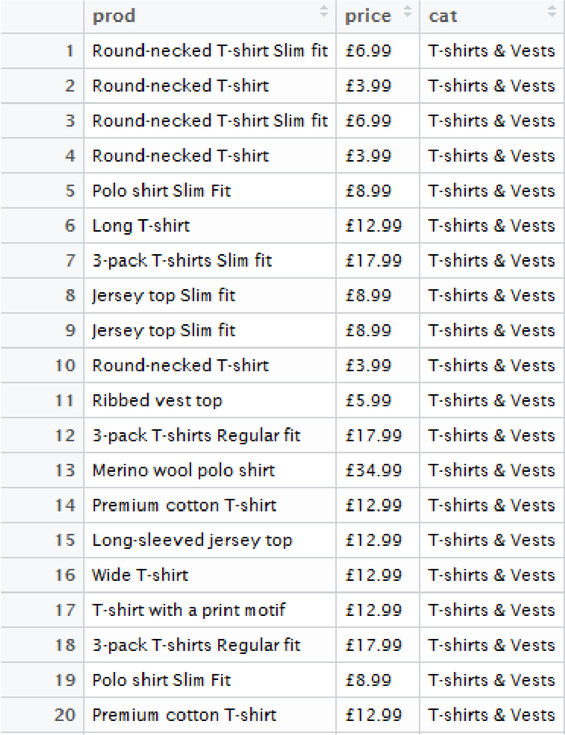
- Select a website
- Read homepage (read_html)
- Scrape all possible URLs (or predefine)
  - Subpages
  - Categories (html_nodes)
- Loop all of the previous URLs
- Scrape information you want
  - Product name
  - Price
  - …
- Store all data in a data frame
- Export data frame

Looping different categories

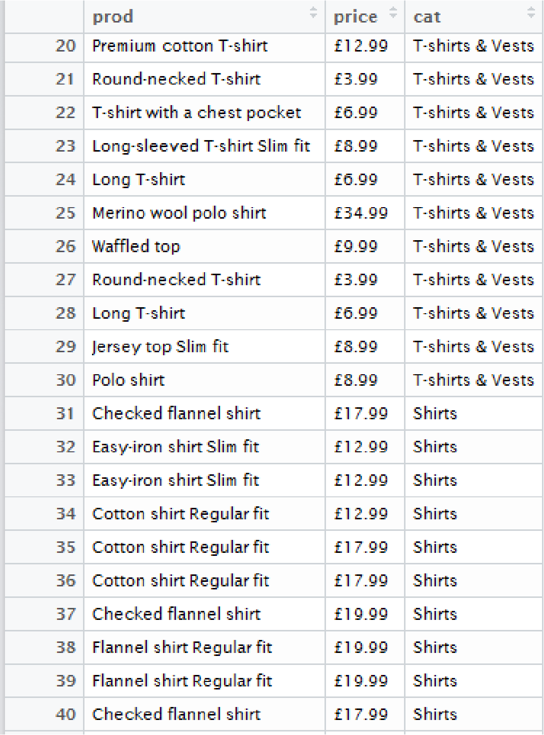
```
#url
main_url <- "http://www2.hm.com"
start_url <- "http://www2.hm.com/en_gb/men/shop-by-product/shirts.html"

#load html page
main_page <- read_html(start_url)

#Scrape subcategories
cat <- main_page %>% html_nodes(".section-menu-subcategory a") %>% html_text()
cat <- str_trim(cat)
cat_url <- main_page %>% html_nodes(". section-menu-subcategory a ") %>% html_attr("href")

#loop all categories and scrape price and product name
data<-NULL

for(i in 1:length(cat)){
  current_page <- as.character(paste0(main_url,cat_url[i]))
  main_page <- read_html(start_url)

  price <- str_trim(scrape_css(".price", ".product-item-details"))
  price <- str_trim(price)

  prod <- scrape_css(".product-item-heading a", ".product-item-details")

  data_cat <- data.frame(prod=prod, price=price, cat=cat[i])
  data <- rbind(data, data_cat)
  }
```

Looping different categories

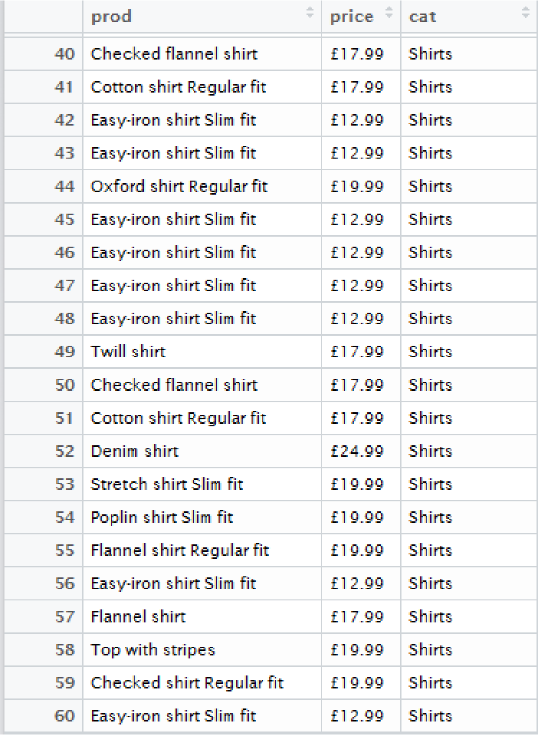
Result of the loop for 2 categories:



	prod	price	cat
1	Round-necked T-shirt Slim fit	£6.99	T-shirts & Vests
2	Round-necked T-shirt	£3.99	T-shirts & Vests
3	Round-necked T-shirt Slim fit	£6.99	T-shirts & Vests
4	Round-necked T-shirt	£3.99	T-shirts & Vests
5	Polo shirt Slim Fit	£8.99	T-shirts & Vests
6	Long T-shirt	£12.99	T-shirts & Vests
7	3-pack T-shirts Slim fit	£17.99	T-shirts & Vests
8	Jersey top Slim fit	£8.99	T-shirts & Vests
9	Jersey top Slim fit	£8.99	T-shirts & Vests
10	Round-necked T-shirt	£3.99	T-shirts & Vests
11	Ribbed vest top	£5.99	T-shirts & Vests
12	3-pack T-shirts Regular fit	£17.99	T-shirts & Vests
13	Merino wool polo shirt	£34.99	T-shirts & Vests
14	Premium cotton T-shirt	£12.99	T-shirts & Vests
15	Long-sleeved jersey top	£12.99	T-shirts & Vests
16	Wide T-shirt	£12.99	T-shirts & Vests
17	T-shirt with a print motif	£12.99	T-shirts & Vests
18	3-pack T-shirts Regular fit	£17.99	T-shirts & Vests
19	Polo shirt Slim Fit	£8.99	T-shirts & Vests
20	Premium cotton T-shirt	£12.99	T-shirts & Vests



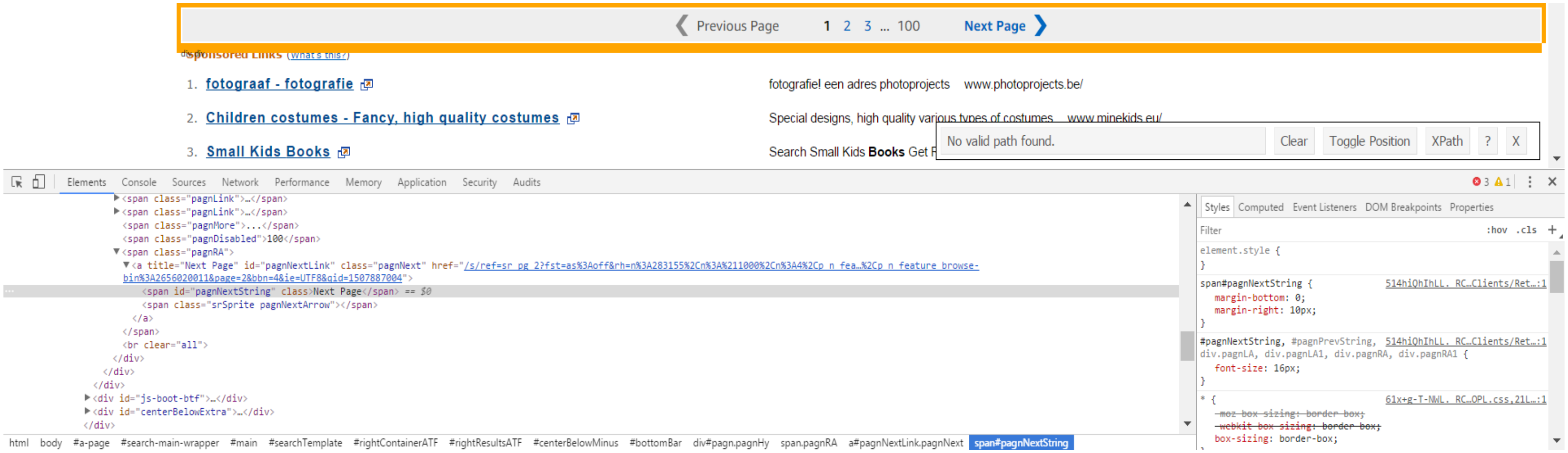
	prod	price	cat
20	Premium cotton T-shirt	£12.99	T-shirts & Vests
21	Round-necked T-shirt	£3.99	T-shirts & Vests
22	T-shirt with a chest pocket	£6.99	T-shirts & Vests
23	Long-sleeved T-shirt Slim fit	£8.99	T-shirts & Vests
24	Long T-shirt	£6.99	T-shirts & Vests
25	Merino wool polo shirt	£34.99	T-shirts & Vests
26	Waffled top	£9.99	T-shirts & Vests
27	Round-necked T-shirt	£3.99	T-shirts & Vests
28	Long T-shirt	£6.99	T-shirts & Vests
29	Jersey top Slim fit	£8.99	T-shirts & Vests
30	Polo shirt	£8.99	T-shirts & Vests
31	Checked flannel shirt	£17.99	Shirts
32	Easy-iron shirt Slim fit	£12.99	Shirts
33	Easy-iron shirt Slim fit	£12.99	Shirts
34	Cotton shirt Regular fit	£12.99	Shirts
35	Cotton shirt Regular fit	£17.99	Shirts
36	Cotton shirt Regular fit	£17.99	Shirts
37	Checked flannel shirt	£19.99	Shirts
38	Flannel shirt Regular fit	£19.99	Shirts
39	Flannel shirt Regular fit	£19.99	Shirts
40	Checked flannel shirt	£17.99	Shirts



	prod	price	cat
40	Checked flannel shirt	£17.99	Shirts
41	Cotton shirt Regular fit	£17.99	Shirts
42	Easy-iron shirt Slim fit	£12.99	Shirts
43	Easy-iron shirt Slim fit	£12.99	Shirts
44	Oxford shirt Regular fit	£19.99	Shirts
45	Easy-iron shirt Slim fit	£12.99	Shirts
46	Easy-iron shirt Slim fit	£12.99	Shirts
47	Easy-iron shirt Slim fit	£12.99	Shirts
48	Easy-iron shirt Slim fit	£12.99	Shirts
49	Twill shirt	£17.99	Shirts
50	Checked flannel shirt	£17.99	Shirts
51	Cotton shirt Regular fit	£17.99	Shirts
52	Denim shirt	£24.99	Shirts
53	Stretch shirt Slim fit	£19.99	Shirts
54	Poplin shirt Slim fit	£19.99	Shirts
55	Flannel shirt Regular fit	£19.99	Shirts
56	Easy-iron shirt Slim fit	£12.99	Shirts
57	Flannel shirt	£17.99	Shirts
58	Top with stripes	£19.99	Shirts
59	Checked shirt Regular fit	£19.99	Shirts
60	Easy-iron shirt Slim fit	£12.99	Shirts

Next page

```r
#url
main_url <- "https://www.amazon.com"

#Go to next page
next_page <- main_page %>% html_node("#pagnNextLink") %>% html_attr("href")
current_page<-as.character(paste0(main_url,next_page))
main_page <- read_html(start_url)
```



1. fotograaf - fotografie — fotografie! een adres photoprojects www.photoprojects.be/
2. Children costumes - Fancy, high quality costumes — Special designs, high quality various types of costumes www.minekids.eu/
3. Small Kids Books — Search Small Kids **Books** Get F

```
▶<span class="pagnLink">…</span>
▶<span class="pagnLink">…</span>
 <span class="pagnMore">...</span>
 <span class="pagnDisabled">100</span>
▼<span class="pagnRA">
 ▼<a title="Next Page" id="pagnNextLink" class="pagnNext" href="/s/ref=sr_pg_2?fst=as%3Aoff&rh=n%3A283155%2Cn%3A%211000%2Cn%3A4%2Cp_n_fea…%2Cp_n_feature_browse-
 bin%3A2656020011&page=2&bbn=4&ie=UTF8&qid=1507887004">
     <span id="pagnNextString" class>Next Page</span> == $0
     <span class="srSprite pagnNextArrow"></span>
   </a>
 </span>
 <br clear="all">
</div>
</div>
</div>
▶<div id="js-boot-btf">…</div>
▶<div id="centerBelowExtra">…</div>
</div>
```

html body #a-page #search-main-wrapper #main #searchTemplate #rightContainerATF #rightResultsATF #centerBelowMinus #bottomBar div#pagn.pagnHy span.pagnRA a#pagnNextLink.pagnNext span#pagnNextString

```
element.style {
}
span#pagnNextString {                    514hiQhIhLL._RC…Clients/Ret…:1
  margin-bottom: 0;
  margin-right: 10px;
}
#pagnNextString, #pagnPrevString,        514hiQhIhLL._RC…Clients/Ret…:1
div.pagnLA, div.pagnLA1, div.pagnRA, div.pagnRA1 {
  font-size: 16px;
}
* {                                      61x+g-T-NWL._RC…OPL.css,21L…:1
  ~~-moz-box-sizing: border-box;~~
  ~~-webkit-box-sizing: border-box;~~
  box-sizing: border-box;
```

RSelenium

Rvest downloads the HTML page and using rvest functions information can be selected

→ dynamic interaction is not possible

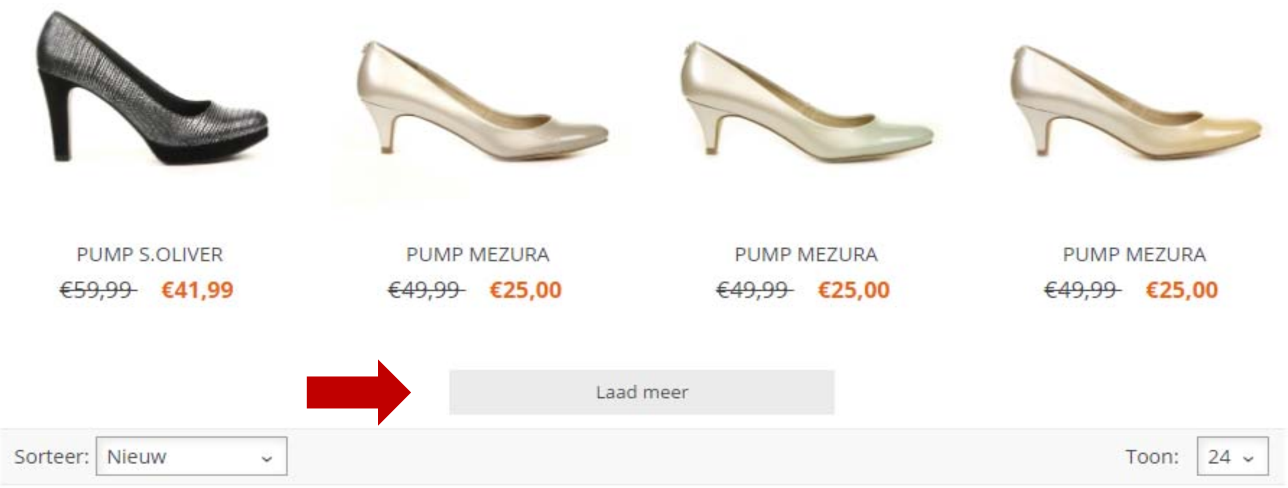
Dynamic interaction:

- Clicking a button to load more products
- Scrolling down to automatically load more products
- Filling in a form and click search button

Solution: RSelenium → provides R bindings for the Selenium Webdriver

RSelenium

Clicking a button to load more products (“Laad meer” = Load more)



The script on the next slide will open the webpage automatically in Chrome and click on the button until the button is not more available

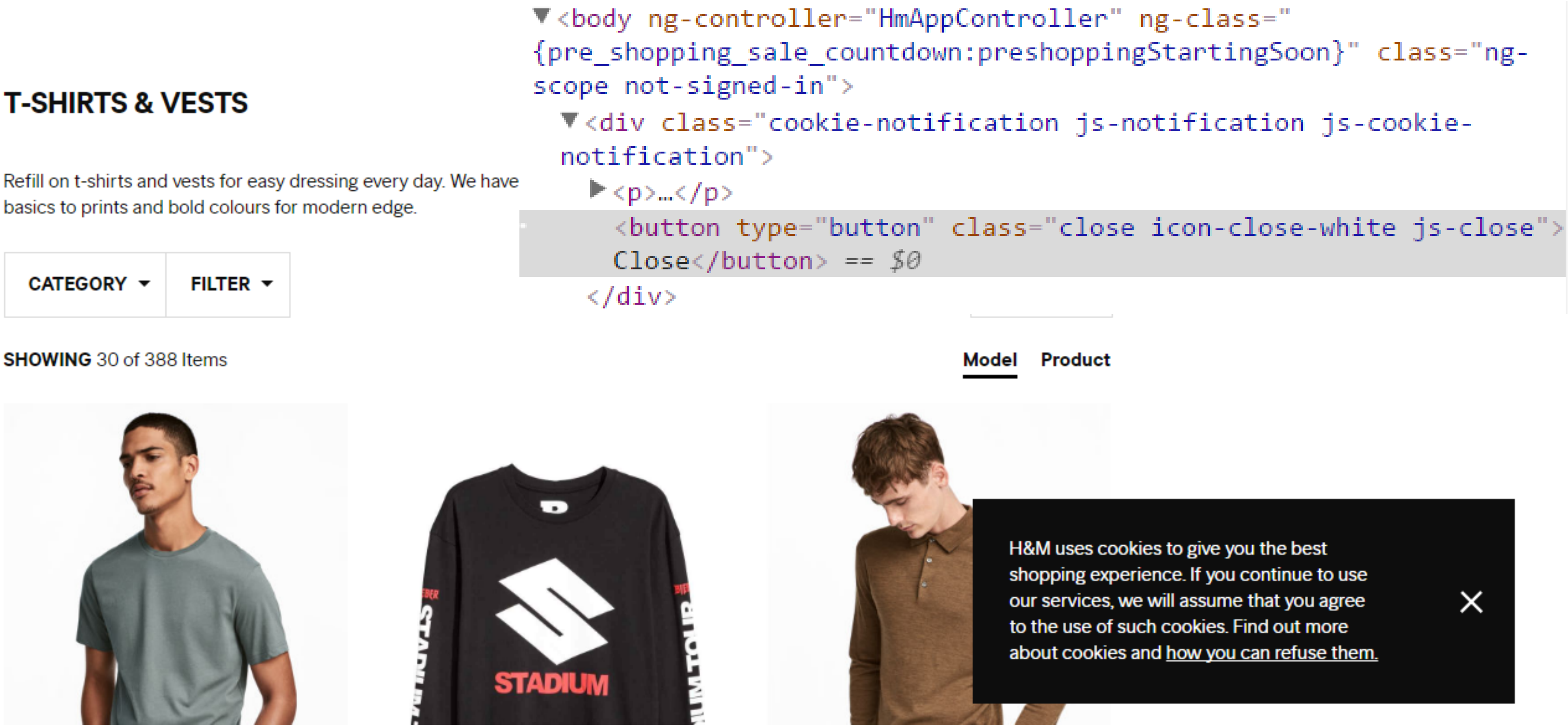
The whole webpage can then be scraped using rvest

RSelenium – Clicking a button

```
library(RSelenium)
library(rvest)
#url
start_url <- "https://be.avanceshoes.com/be/dames/schoenen/pumps.html"
#loading Selenium server and ChromeDriver
remDr <- remoteDriver(browserName = "chrome")
remDr$open()
Sys.sleep(2)
#navigate to the url
main_page <- remDr$navigate(start_url)

#code to find the button via a CSS selector and then clicking the button "laad meer"  (CSS selector
".i-next") #until it disappears
webElems <- remDr$findElements(using = 'css selector', ".i-next")
Sys.sleep(4)
while (length(webElems) != 0) {
  webElem <- remDr$findElement(using = 'css selector', ".i-next")
  webElem$clickElement()
  Sys.sleep(4)
  webElems <- remDr$findElements(using = 'css selector', ".i-next")
  Sys.sleep(4)
  }

#getting the final page via rvest
main_page<-read_html(unlist(remDr$getPageSource()),encoding="UTF-8")
#closing the Selenium session
remDr$close()
```

RSelenium – Clicking a button

```
#code to click on button "Accept Cookies"
webElems <- remDr$findElement(using = 'css selector', "body > div.cookie-notification.js-
notification.js-cookie-notification > button")
webElems$clickElement()
```



```
▼<body ng-controller="HmAppController" ng-class="
{pre_shopping_sale_countdown:preshoppingStartingSoon}" class="ng-
scope not-signed-in">
  ▼<div class="cookie-notification js-notification js-cookie-
  notification">
    ▶<p>…</p>
     <button type="button" class="close icon-close-white js-close">
     Close</button> == $0
    </div>
```

T-SHIRTS & VESTS

Refill on t-shirts and vests for easy dressing every day. We have basics to prints and bold colours for modern edge.

SHOWING 30 of 388 Items

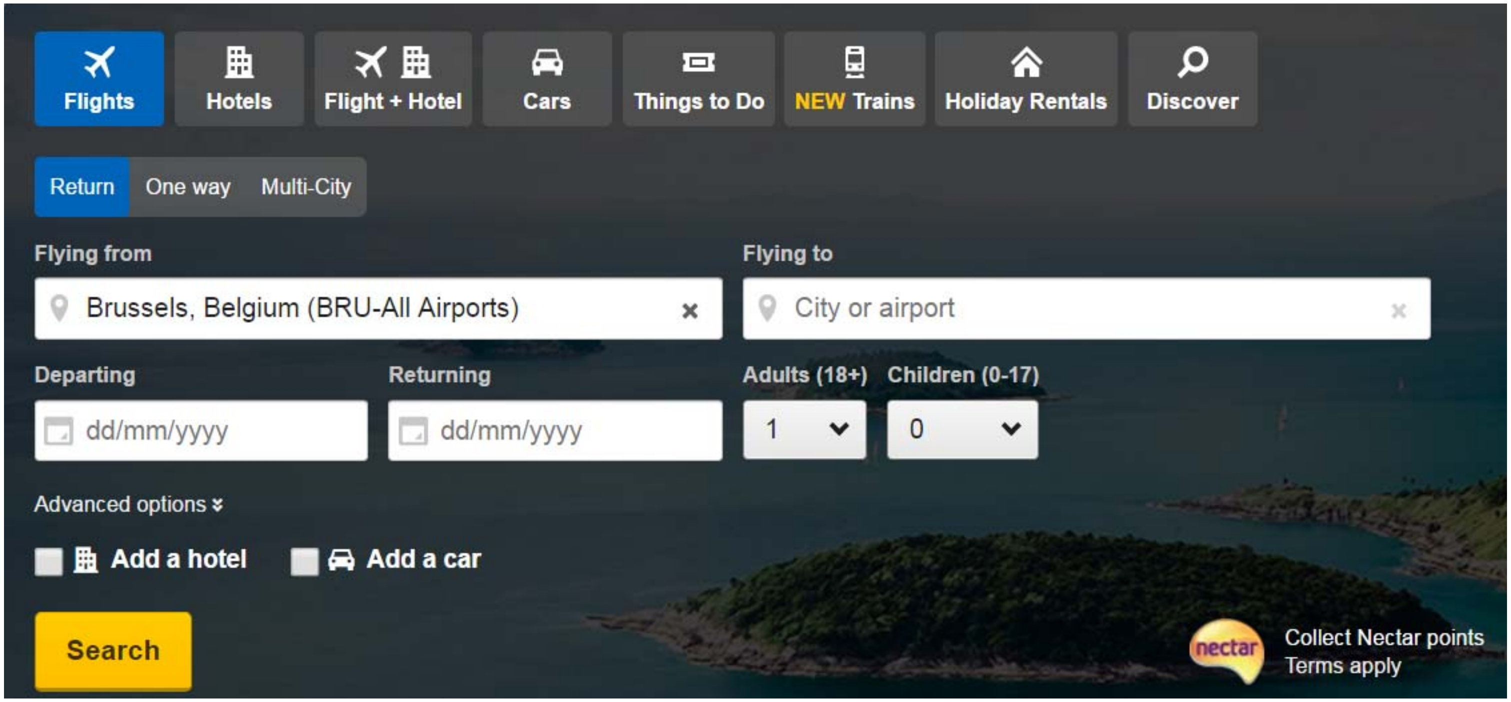
H&M uses cookies to give you the best shopping experience. If you continue to use our services, we will assume that you agree to the use of such cookies. Find out more about cookies and how you can refuse them.

RSelenium – Scrolling

```r
library(RSelenium)
library(rvest)

#url
start_url<-"http://www2.hm.com/en_gb/men/shop-by-product/t-shirts-and-vests.html"

#open the webpage
remDr <- remoteDriver(browserName = "chrome")
remDr$open()
main_page <- remDr$navigate(start_url)

#code to scroll, it scrolls 5 times a certain amount of pixels; in this case 10 000
for(i in 1:5){
  remDr$executeScript(paste("scroll(0,",i*10000,");"),list(""))
  Sys.sleep(3)
}
#getting the final page via rvest
main_page<-read_html(unlist(remDr$getPageSource()),encoding="UTF-8")

#closing ChromeDriver
remDr$close()
```

RSelenium – Filling in a form

Book hotel, flight, train tickets,…



Flying from	Brussels, Belgium (BRU-All Airports)
Flying to	City or airport
Departing	dd/mm/yyyy
Returning	dd/mm/yyyy
Adults (18+)	1
Children (0-17)	0

Fill in:

- Destination
- Departure date
- Return date

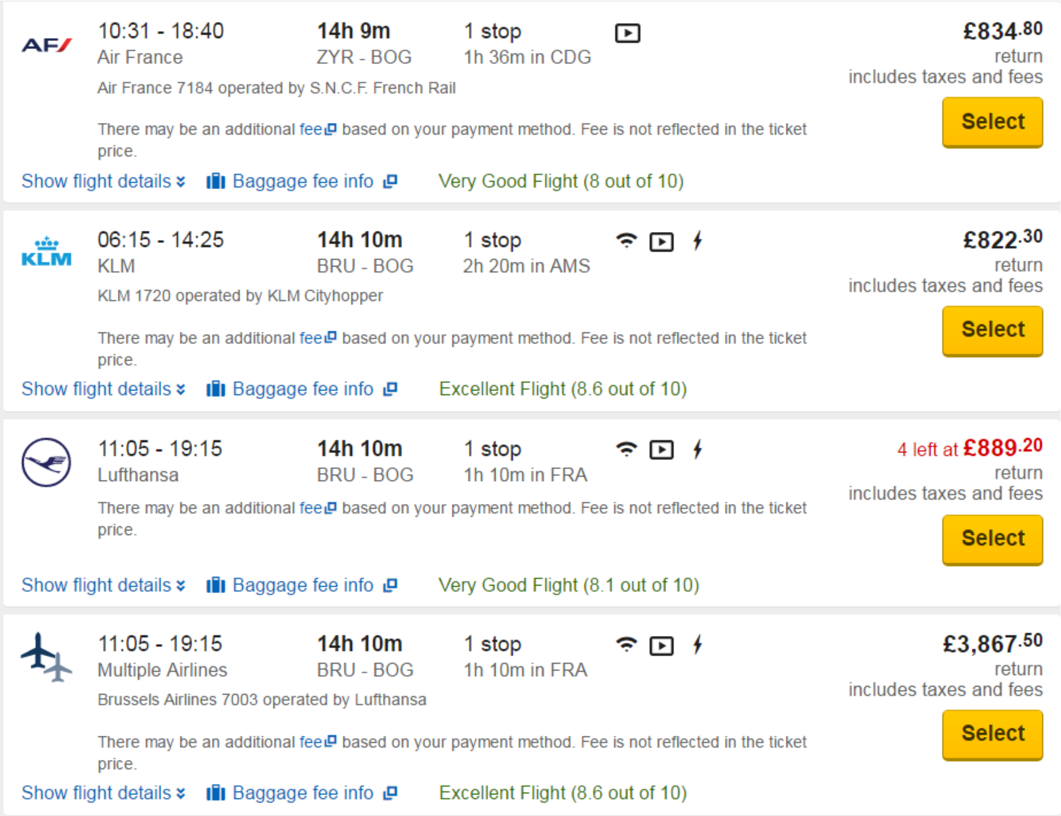
RSelenium – Filling in a form

Predefining:

- List of destinations
- Number of weeks booked in advance

Result: screen with different prices depending on Airline, stops, options,…

→ Rvest to scrape this data



10:31 - 18:40 | 14h 9m | 1 stop | £834.80
Air France | ZYR - BOG | 1h 36m in CDG | return
includes taxes and fees
Air France 7184 operated by S.N.C.F. French Rail
There may be an additional fee based on your payment method. Fee is not reflected in the ticket price.
Select
Show flight details | Baggage fee info | Very Good Flight (8 out of 10)

06:15 - 14:25 | 14h 10m | 1 stop | £822.30
KLM | BRU - BOG | 2h 20m in AMS | return
includes taxes and fees
KLM 1720 operated by KLM Cityhopper
There may be an additional fee based on your payment method. Fee is not reflected in the ticket price.
Select
Show flight details | Baggage fee info | Excellent Flight (8.6 out of 10)

11:05 - 19:15 | 14h 10m | 1 stop | 4 left at £889.20
Lufthansa | BRU - BOG | 1h 10m in FRA | return
includes taxes and fees
There may be an additional fee based on your payment method. Fee is not reflected in the ticket price.
Select
Show flight details | Baggage fee info | Very Good Flight (8.1 out of 10)

11:05 - 19:15 | 14h 10m | 1 stop | £3,867.50
Multiple Airlines | BRU - BOG | 1h 10m in FRA | return
includes taxes and fees
Brussels Airlines 7003 operated by Lufthansa
There may be an additional fee based on your payment method. Fee is not reflected in the ticket price.
Select
Show flight details | Baggage fee info | Excellent Flight (8.6 out of 10)

RSelenium – Filling in a form

```
#starting Selenium server
server<-startServer()
Sys.sleep(5)

#url
start_url<-"https://www.expedia.com/"
#gettting the date of today
current_date<-Sys.Date()
current_date_txt<-format(current_date,"%d/%m/%Y")
#departure date 28 days after the current date
dep_date<-current_date+(4*7)
dep_date_txt<-format(dep_date,"%m/%d/%Y")
#return date, 7 days after the departure date
ret_date<-dep_date+7
ret_date_txt<-format(ret_date,"%m/%d/%Y")
#destination Bogota, so a flight from Brussels to Bogota
departure <- "BRU"
destination <- "BOG"

#navigate to the url
remDr <- remoteDriver(browserName = "chrome")
remDr$open()
remDr$navigate(start_url)
#remDr$refresh()
```

RSelenium – Filling in a form

```
#close pop-ups
webElem <- remDr$findElement(using = 'css selector', 'button#join-rewards-close-btn')
webElem$clickElement()

webElem <- remDr$findElement(using = 'css selector', 'button.btn-close')
webElem$clickElement()

#click on flights
webElem <- remDr$findElement(using = 'css selector', '#primary-header-flight')
webElem$clickElement()
Sys.sleep(3)

#finding the CSS selector of the departure airport and filling it in with the airport
input_dep<-remDr$findElement(using="css selector","#flight-origin-flp")
input_dep$sendKeysToElement(list(departure))
blank<-remDr$findElement(using="css selector",".cols-nested+ .cols-nested")
blank$clickElement()
Sys.sleep(3)

input_dest<-remDr$findElement(using="css selector","#flight-destination-flp")
input_dest$sendKeysToElement(list(destination))
Sys.sleep(3)

input_date_dep<-remDr$findElement(using="css selector","#flight-departing-flp")
input_date_dep$clearElement()
Sys.sleep(1)
input_date_dep$sendKeysToElement(list(dep_date_txt))
Sys.sleep(3)
```

RSelenium – Filling in a form

```r
input_date_ret<-remDr$findElement(using="css selector","#flight-returning-flp")
input_date_ret$clearElement()
Sys.sleep(1)
input_date_ret$sendKeysToElement(list(ret_date_txt))
Sys.sleep(3)

close_calendar<-remDr$findElement(using="css selector",".datepicker-close-btn")
close_calendar$clickElement()
Sys.sleep(1)

search<-remDr$findElement(using="css","#flight-lap-or-seat-container-flp ~ .cols-nested .gcw-submit")
search$clickElement()

#getting the page with all the prices in rvest
main_page<-read_html(unlist(remDr$getPageSource()),encoding="UTF-8")

#closing ChromeDriver
remDr$close()
server$stop()
```

Implementation at Statistics Belgium

Web scraping – implementation

- Scripts are executed on a Linux server mostly at night
- Pauses are integrated into the script (Sys.sleep() function) to avoid overloading the website (netiquette!)
- Robot identifies itself as “Statistics Belgium”
  - Using proxy server
    - Read_html(httr::GET(start_url, user_agent(agent), proxy))
      - agent: identification to the website (e.g. NSI name)
- Data are saved first in csv files and loaded afterwards in the SAS Data Warehouse of Statistics Belgium
- All products are extracted (bulk scraping)
  - Exceptions: train tickets or airfares: a list of destinations and departure dates are predefined

Dashboard

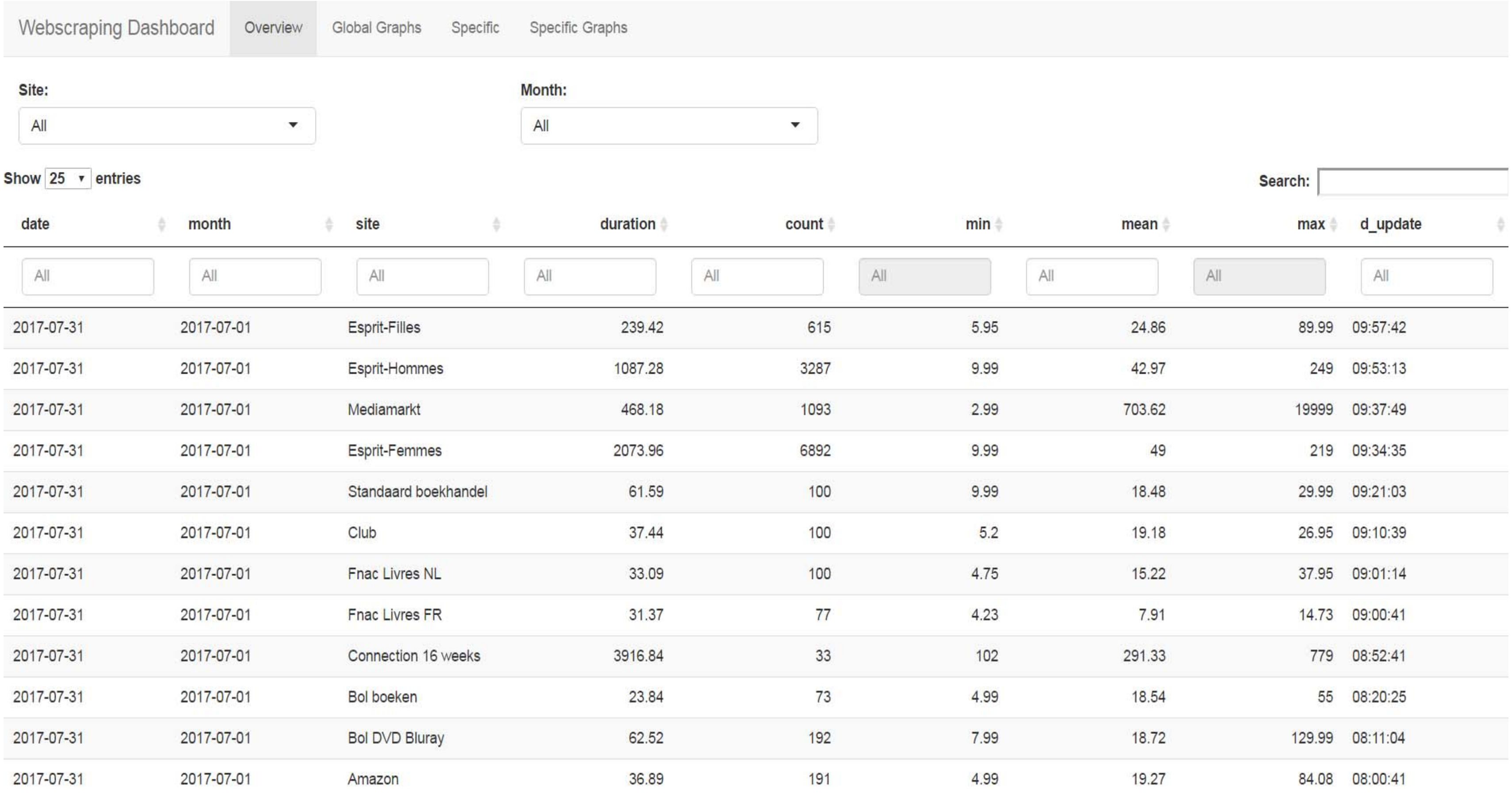
Monitoring results of scripts

- Check output
  - Number of records
  - Check results
- Change scripts in case of missing records
  - e.g. due to change in website
- Failed scripts (also receive an automatic mail)
  - Server problems
  - Change in website

economie
FPS Economy, S.M.E.s, Self-employed and Energy

Dashboard

Web scraping Dashboard Statistics Belgium – Overview:



Site: All

Month: All

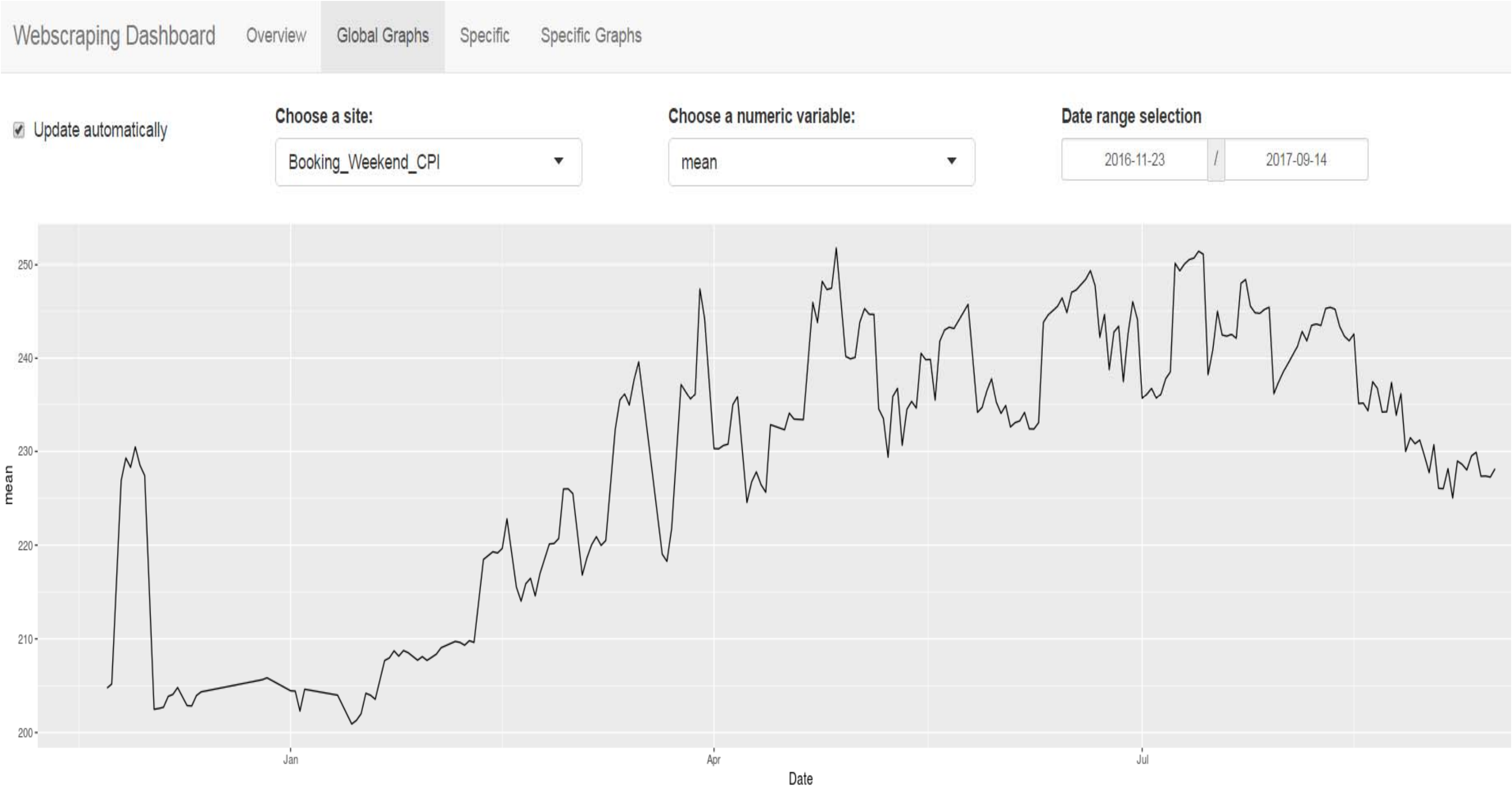
Show 25 entries

Search:

date	month	site	duration	count	min	mean	max	d_update
All	All	All	All	All	All	All	All	All
2017-07-31	2017-07-01	Esprit-Filles	239.42	615	5.95	24.86	89.99	09:57:42
2017-07-31	2017-07-01	Esprit-Hommes	1087.28	3287	9.99	42.97	249	09:53:13
2017-07-31	2017-07-01	Mediamarkt	468.18	1093	2.99	703.62	19999	09:37:49
2017-07-31	2017-07-01	Esprit-Femmes	2073.96	6892	9.99	49	219	09:34:35
2017-07-31	2017-07-01	Standaard boekhandel	61.59	100	9.99	18.48	29.99	09:21:03
2017-07-31	2017-07-01	Club	37.44	100	5.2	19.18	26.95	09:10:39
2017-07-31	2017-07-01	Fnac Livres NL	33.09	100	4.75	15.22	37.95	09:01:14
2017-07-31	2017-07-01	Fnac Livres FR	31.37	77	4.23	7.91	14.73	09:00:41
2017-07-31	2017-07-01	Connection 16 weeks	3916.84	33	102	291.33	779	08:52:41
2017-07-31	2017-07-01	Bol boeken	23.84	73	4.99	18.54	55	08:20:25
2017-07-31	2017-07-01	Bol DVD Bluray	62.52	192	7.99	18.72	129.99	08:11:04
2017-07-31	2017-07-01	Amazon	36.89	191	4.99	19.27	84.08	08:00:41

Dashboard

Web scraping Dashboard Statistics Belgium – Global Graphs



☑ Update automatically

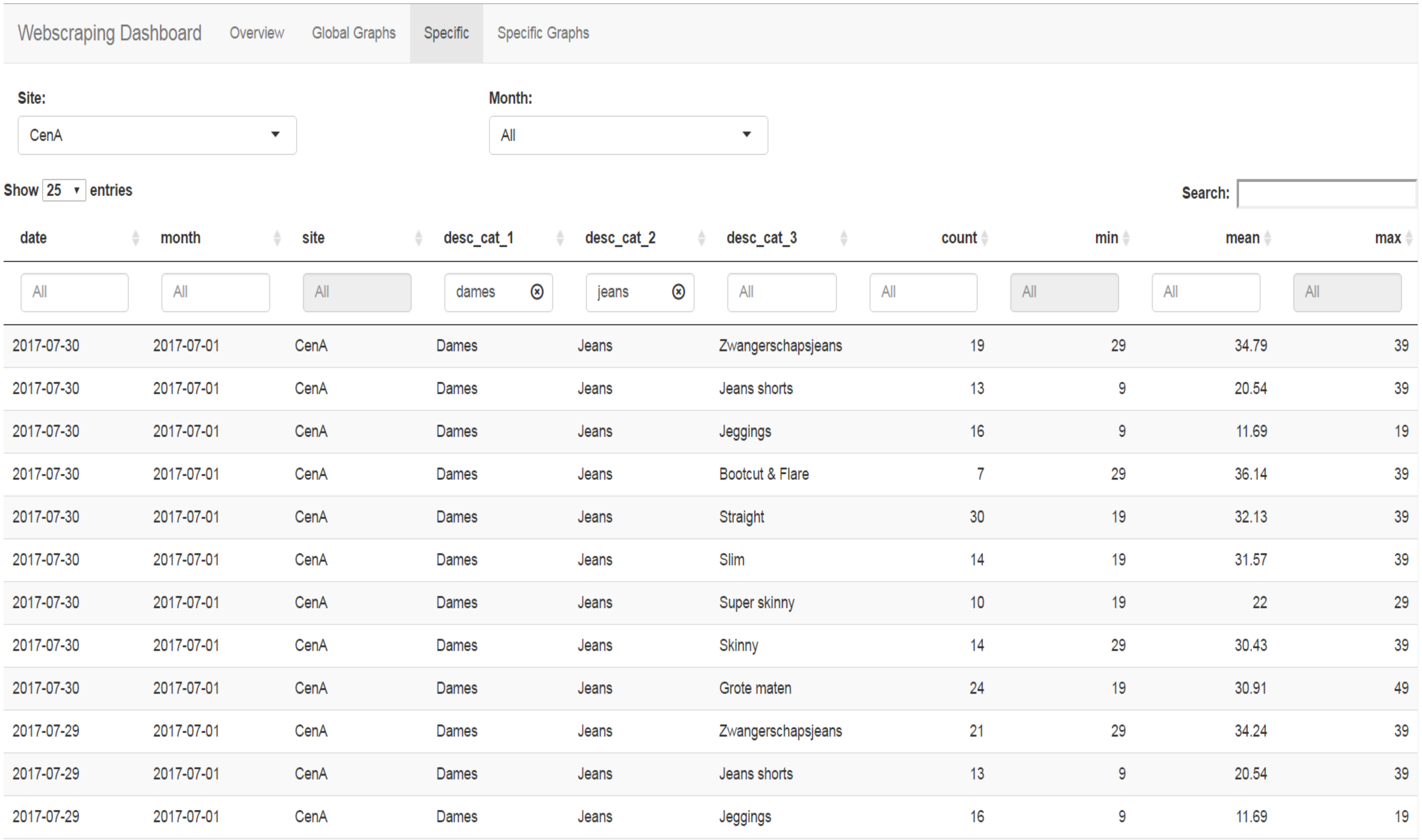
Choose a site: Booking_Weekend_CPI

Choose a numeric variable: mean

Date range selection: 2016-11-23 / 2017-09-14

Dashboard

Web scraping Dashboard Statistics Belgium – Specific



Site: CenA

Month: All

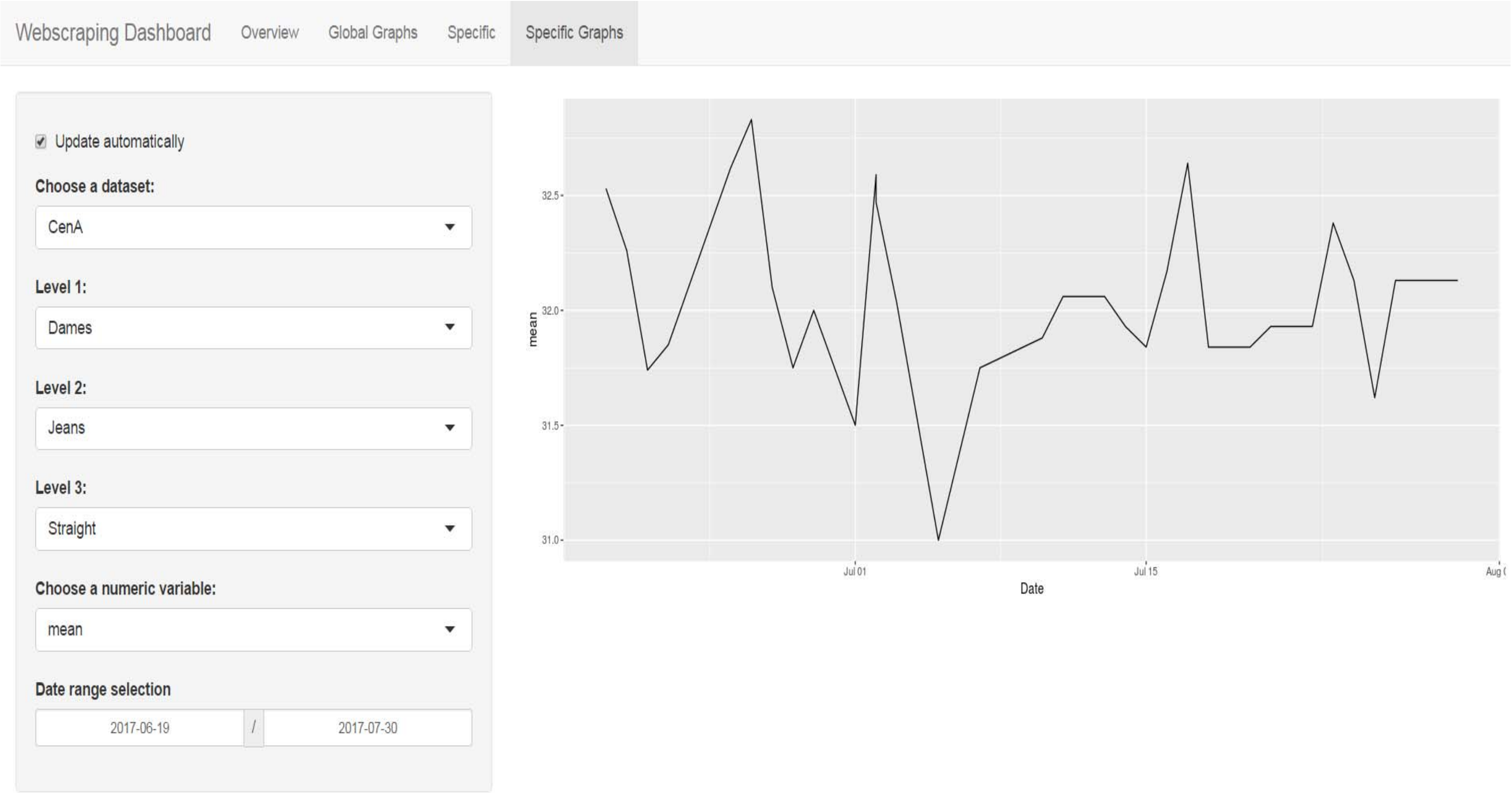
Show 25 entries

Search:

date	month	site	desc_cat_1	desc_cat_2	desc_cat_3	count	min	mean	max
All	All	All	dames ⊗	jeans ⊗	All	All	All	All	All
2017-07-30	2017-07-01	CenA	Dames	Jeans	Zwangerschapsjeans	19	29	34.79	39
2017-07-30	2017-07-01	CenA	Dames	Jeans	Jeans shorts	13	9	20.54	39
2017-07-30	2017-07-01	CenA	Dames	Jeans	Jeggings	16	9	11.69	19
2017-07-30	2017-07-01	CenA	Dames	Jeans	Bootcut & Flare	7	29	36.14	39
2017-07-30	2017-07-01	CenA	Dames	Jeans	Straight	30	19	32.13	39
2017-07-30	2017-07-01	CenA	Dames	Jeans	Slim	14	19	31.57	39
2017-07-30	2017-07-01	CenA	Dames	Jeans	Super skinny	10	19	22	29
2017-07-30	2017-07-01	CenA	Dames	Jeans	Skinny	14	29	30.43	39
2017-07-30	2017-07-01	CenA	Dames	Jeans	Grote maten	24	19	30.91	49
2017-07-29	2017-07-01	CenA	Dames	Jeans	Zwangerschapsjeans	21	29	34.24	39
2017-07-29	2017-07-01	CenA	Dames	Jeans	Jeans shorts	13	9	20.54	39
2017-07-29	2017-07-01	CenA	Dames	Jeans	Jeggings	16	9	11.69	19

Dashboard

Web scraping Dashboard Statistics Belgium – Specific Graphs



☑ Update automatically

Choose a dataset: CenA

Level 1: Dames

Level 2: Jeans

Level 3: Straight

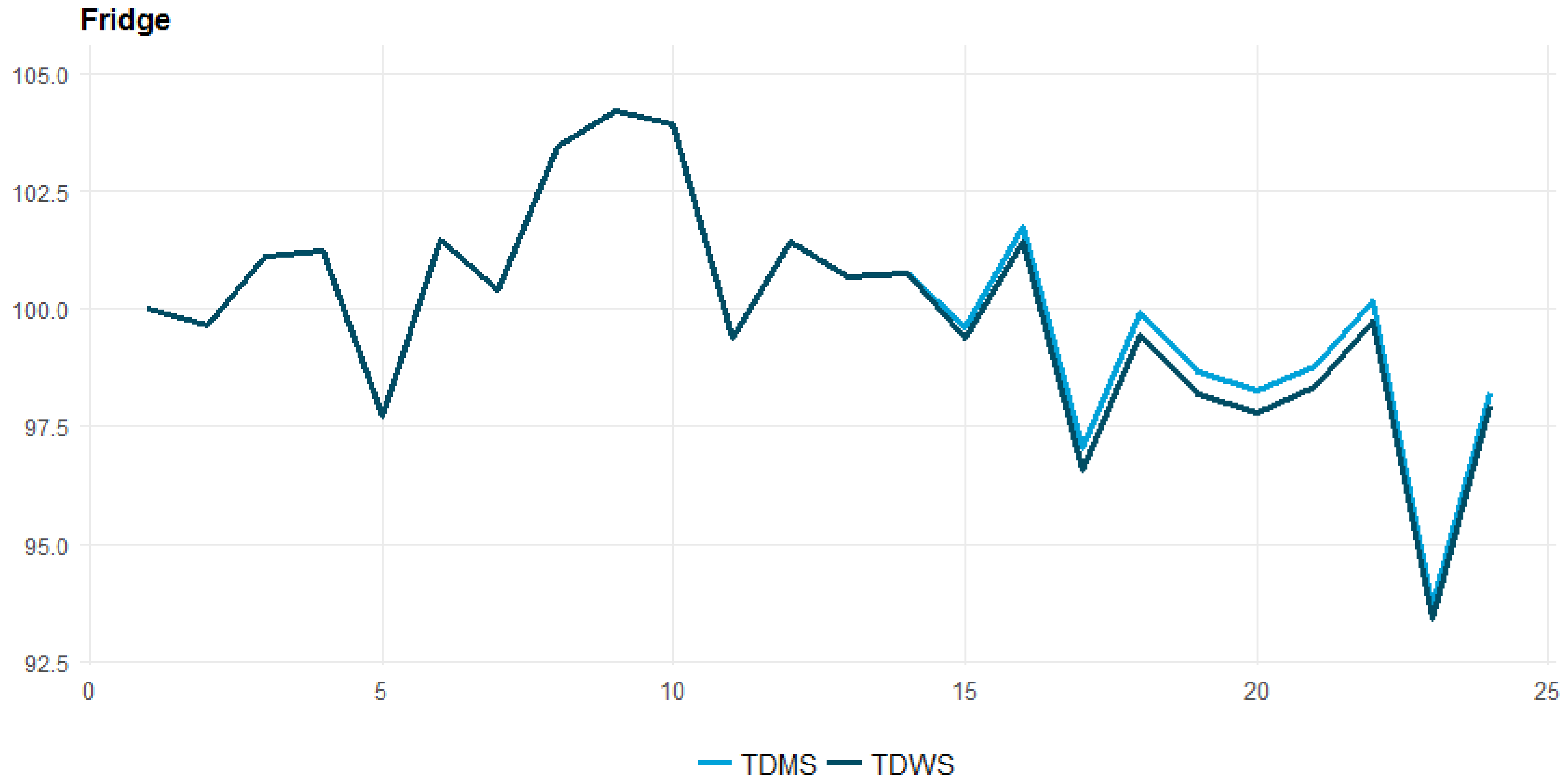
Choose a numeric variable: mean

Date range selection: 2017-06-19 / 2017-07-30

Experimental indices - Fridges

- Daily web scraping
- Bulk scraping
- Low attrition rate
- Hedonic regression: scraping of characteristics
  - Time Dummy with Movement Splice
  - Time Dummy with Window Splice

Experimental indices - Fridges



Experimental indices - Hotels

Manual price collection

- Sample of hotels
- Once a month virtual reservations are made
- 4 weeks before arrival date
  - One price quote for each hotel
- Booking for 2 adults – 2 nights
- Room type is kept stable (if possible)
- ‘Options’ (e.g. free cancelation) are kept stable (if possible)

Experimental indices - Hotels

Web scraping

- Daily web scraping
- 3 Destinations in Belgium: Brussels, Seaside, Ardennes
- 4 – 8 weeks before arrival date
- Arrival on Friday – Departure on Sunday
- Breakfast and free cancelation
- Star rating
- Stratification:

Destination
↓
Area
↓
Weeks booked before arrival date
↓
Hotel star rating

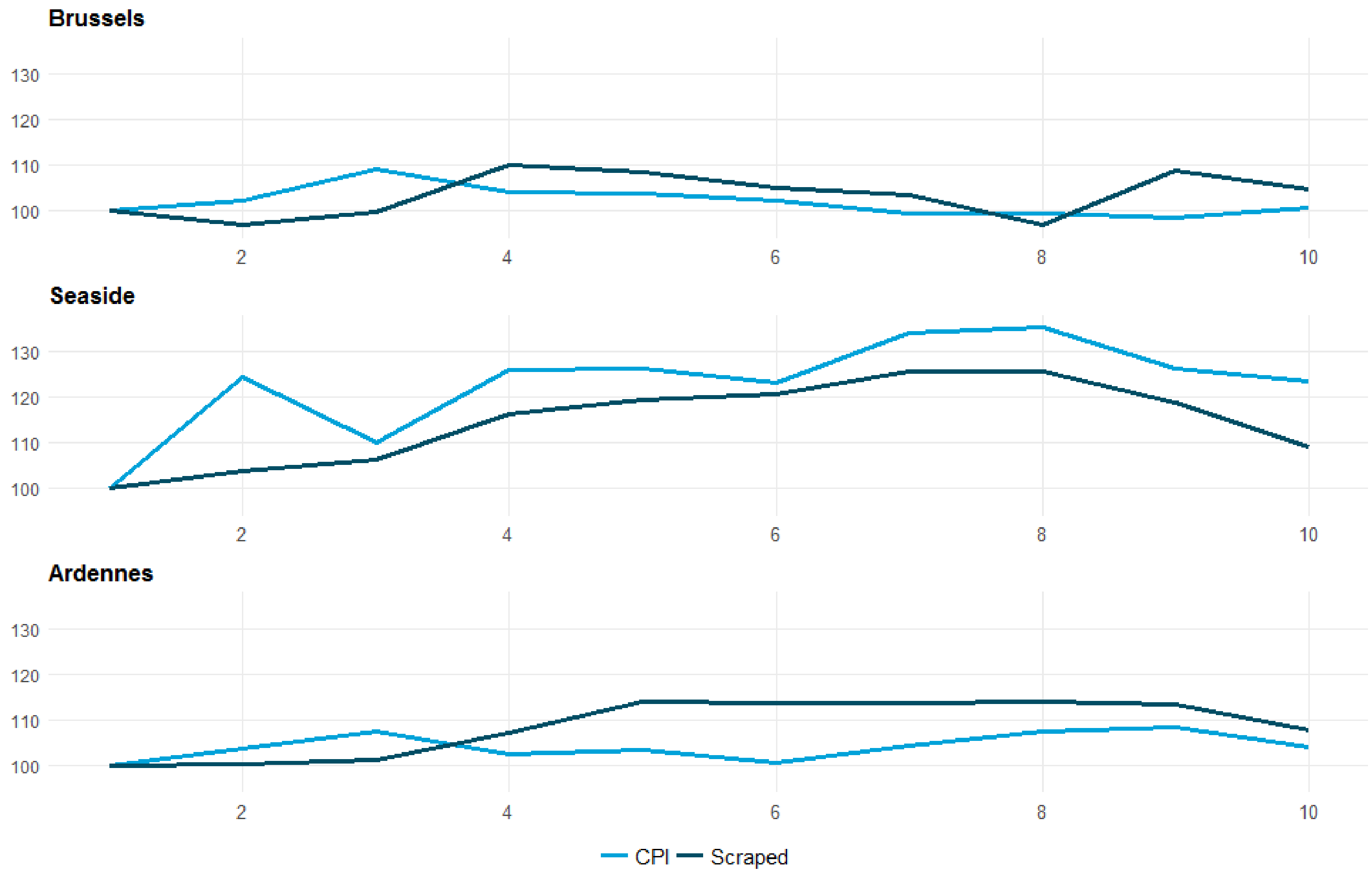
Experimental indices - Hotels

Sample size – number of prices:

Destination	Manual	Web scraping
Brussels	17	2,662
Seaside	25	12,614
Ardennes	15	23,552

economie
FPS Economy, S.M.E.s, Self-employed and Energy

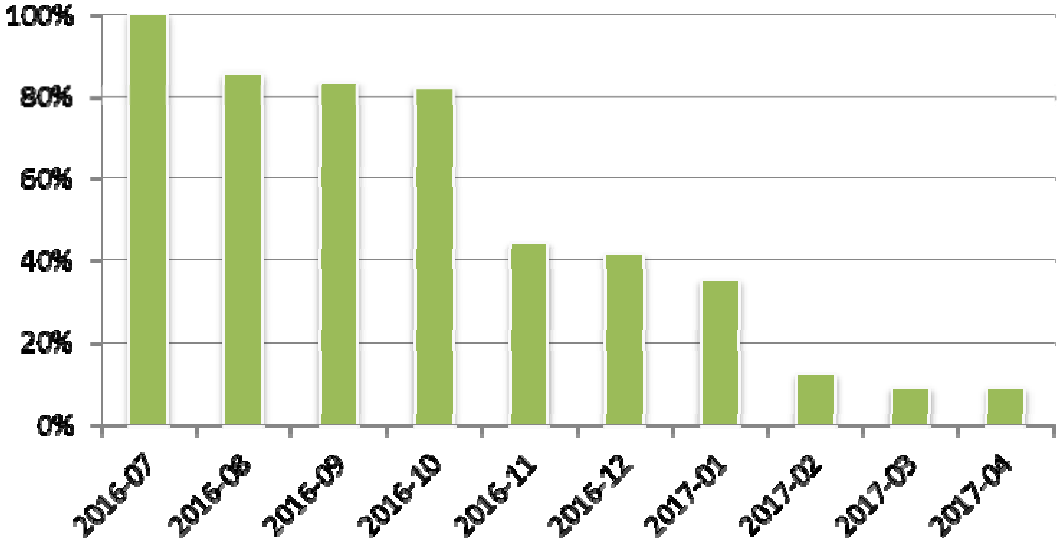
Experimental indices - Hotels



Experimental indices - Footwear

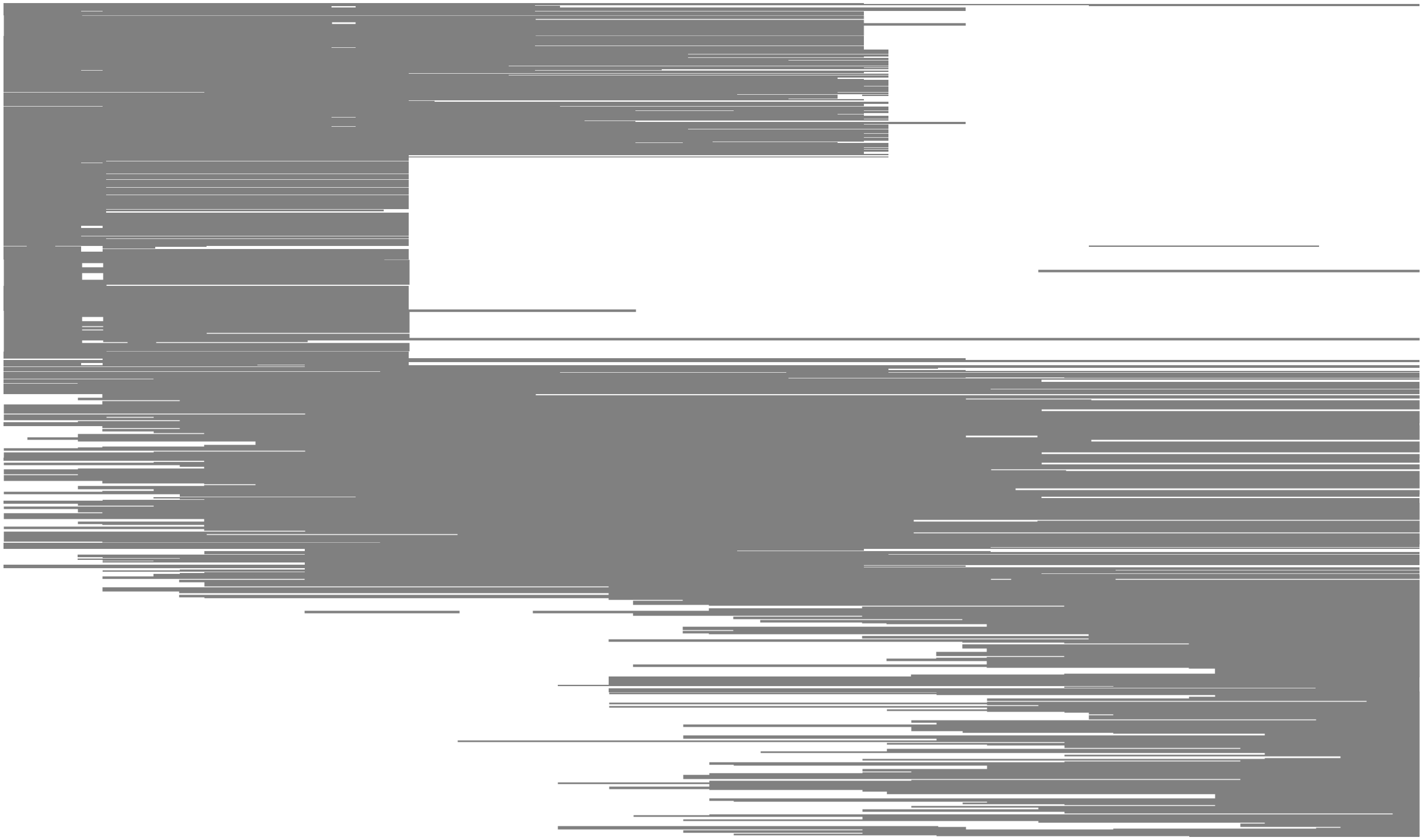
- Scraping multiple times a week
- Bulk scraping
- High attrition rate

% of matching items compared to 07-2016



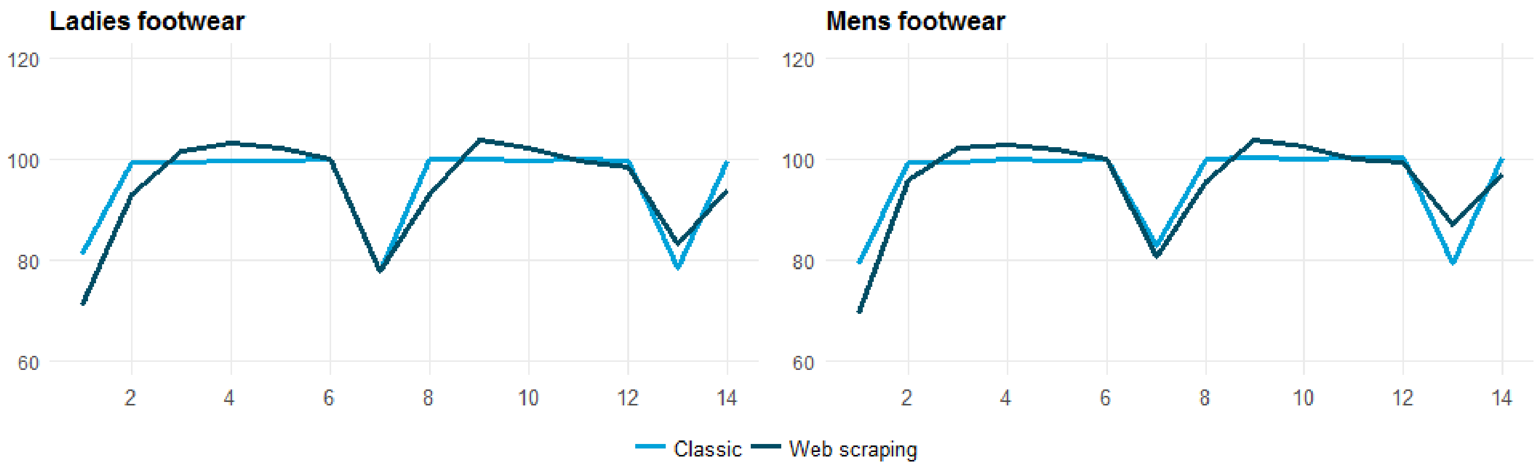
Experimental indices - Footwear

Availability of footwear:



Experimental indices - Footwear

- Non-matched model to avoid downward drift
- Stratification: men - women


Ladies footwear
Mens footwear
Classic
Web scraping