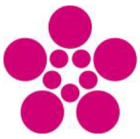


Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Diplomová práce

NÁVRH APLIKACE PRO AUTOMATIZOVANOU EXTRAKCI ATRIBUTŮ PRODUKTŮ WEBŮ

Vypracovala: Bc. Petra Chvostová

Vedoucí práce: doc. Ing. Ladislav Beránek, CSc.

České Budějovice 2022

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Petra CHVOSTOVÁ**
Osobní číslo:	**E18348**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Návrh aplikace pro automatizovanou extrakci atributů produktů webů**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Jednou z možností, kterou mají v sobě zabudovánu některé webové stránky, je možnost vyhledávat a porovnávat produkty z různých stránek. Cílem práce bude navrhnout aplikaci pro automatizovanou extrakci atributů produktů webů. Aplikace bude provádět extrakci sémantických částí stránek, následně bude provedena úprava získaných dat na základě lokálního kontextu. Získaná data pak budou rozdělena do předem definovaných kategorií, případně zpracována jiným vhodným způsobem pro další využití.

Metodický postup:

1. Analýza existujících postupů, teorie.
2. Návrh a popis vývoje a implementace výsledné aplikace.
3. Testy, experimenty, zhodnocení použitelnosti aplikace.
4. Doporučení, závěr.

Rozsah pracovní zprávy:	**50 – 60 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná**

Seznam doporučené literatury:

1. Archak, N., Ghose, A., & Ipeirotis, P. G. (2011). Deriving the Pricing Power of Product Features by Mining Consumer Reviews. *Management Science*, 57 (8), 1485-1509.
2. Ingersoll, G.S., Morton, T.S., & Farris, D. (2013). *Taming Text: How to Find, Organize, and Manipulate It.* Shelter Island: Manning Publications.
3. Munzert, S., Rubba, C., Meißner, P., Nyhuis, D. (2015). *Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining.* Hoboken: Wiley & Sons.

Vedoucí diplomové práce:	**doc. Ing. Ladislav Beránek, CSc.** Katedra aplikované matematiky a informatiky

Datum zadání diplomové práce: **15. ledna 2019**
Termín odevzdání diplomové práce: **14. dubna 2020**

V Českých Budějovicích dne 18. března 2019

L.S.

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Jana Klicnarová, Ph.D.
vedoucí katedry

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to - v nezkrácené podobě ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. Zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

…….. ……

Datum Podpis

Poděkování

Ráda bych tímto způsobem poděkovala panu doc. Ing. Ladislavu Beránkovi, CSc., vedoucímu mé diplomové práce, za jeho rady a pomoc při tvorbě této práce. Dále bych chtěla poděkovat svému okolí, rodině a přátelům, za podporu ve studiu.

Obsah

Úvod

Teoretická část této diplomové práce představuje základní technologie, které jsou předpokladem komunikace, výměny, ukládání a zobrazování informací na World Wide Web (HTTP, HTML, XML, JSON, AJAX, SQL). Zároveň poskytuje základní techniky pro dotazování webových dokumentů a datové sady (XPath a regulární výrazy). Tyto základy jsou užitečné zejména pro čtenáře, kteří neznají architekturu webu, ale mohou také sloužit jako osvěžení pro znalé uživatele. Následující část s názvem HTML je věnována způsobu tvorby zdrojového kódu. Závěrečná kapitola teoretické části, s názvem Extrakce[1] pomocí jazyka R[2], představuje funkce potřebné k samotné extrakci. Jsou zde také uvedeny právní aspekty extrakce a rady, jak se na internetu při této aktivitě chovat slušně.

Praktická část se věnuje analýze a stahování dat z vybraného e-shopu. Pro názorné provedení návrhu extrakce atributů z webu byl vybrán dynamický e-shop Vodácké a kempingové potřeby-Eshop Praha 5. Po představení a analýze webu byl vytvořen postup, jakým budou data stahována. Za tímto účelem byl vytvořen kód zadávaný v R, který je důkladně popsán v druhé kapitole praktické části. Pro přehlednost je kód v textu práce odlišen fontem (konkrétně Courier New). V závěru praktické části je uveden výsledek stažených dat z webu, který je možný uložit do souboru pro další použití.

Již v úvodu je potřeba zmínit, že autorka si je vědoma funkčností sestrojeného kódu pro extrakci atributů produktů z vybraného webu pouze pro podobně navržené e-shopy. Jedná se spíše o ukázku, jakým způsobem je možné kód sestrojit. Pro ostatní weby by bylo zapotřebí kód náležitě upravit. Také je nutné zdůraznit, že samotná extrakce má využití spíše pro firmy a jejich dodavatele, nikoliv pro běžného uživatele internetu.

---

[1] neboli web scrapingu, stahování dat z webu

[2] Extrakci z webů je možné provádět užitím i jiných jazyků, než je R. Například se nabízí Python nebo C# a další. Nás ale zajímá tvorba s vybraným jazykem v prostředí Rstudia. Ve třetí kapitole si více představíme důvody výběru.

Cíle a motivace

Cílem této práce je navrhnout možné řešení, jak pomocí jazyka R takzvaně škrabat, neboli stahovat data z webových stránek. Potřeba něco takového provést je na místě z důvodu vysoké chybovosti při ručním zadávání dat a při jejich kopírování. Navržený kód také šetří mnoho času, což šetří zároveň i peníze. Porovnání, v němž by jeden člověk měl například ručně sepsat nebo zkopírovat informace o několika tisících položek, versus pouhá úprava kódu navrženého v této práci, je rozdíl desítek hodin (v závislosti na množství dat, může se jednat i o dny či týdny) a množství chybných dat. Důležité je dodat, že nasbíraná data je potřeba někdy aktualizovat, tedy nemusí se jednat o jednorázovou aktivitu, což představuje několikanásobně větší časový rozdíl.

Zároveň je cílem této práce zkonstruovat takový kód, který je řešením konkrétního problému při sběru dat. Procesu sběru informací je téměř vlastní, že pole, kde jsou data sklízena (stahována), nejsou nikdy totožná a někdy rychle mění tvar. Proto je v praktické části kód upraven pro konkrétní e-shop. Na tomto příkladu bude ukázáno, jak vybrané techniky použít v praxi pro stahování určitých dat z webových stránek obchodů. Extrahovaná data jsou použita pouze pro účely této diplomové práce. Nejsou a nebudou mimo tuto práci použita. Důvodem výběru prostředí pro sepsání kódu je nejen jeho volná dostupnost, ale především následná možná práce s daty. Program Rstudio je statisticky zaměřen. Pokud nám tedy nejde pouze o stažení dat z webu, máme možnost následně s daty pracovat již v samotném programu, který je nám znám, jelikož jsme jej použili ke stahování z webových stránek. Není tudíž potřebné se učit pracovat v žádném dalším programu.

Autorka práce byla po nějaký čas součástí týmu, který pracoval na projektu tvorby webu, do kterého bylo potřeba vkládat informace dostupné z desítek jiných webových stránek. Tato práce byla celkem manuální. Bylo zapotřebí ručně vyhledávat informace dostupné na internetu a ty následně překopírovat na projektem tvořený web. Tato činnost trvala dny a bylo třeba ji několikrát ročně opakovat. Její chybovost při kopírování nebyla nulová. Byl to jeden z důvodů, proč si autorka vybrala z nabízených témat právě extrakci atributů produktů z webu. Použití navrženého kódu se jí jeví jako velmi užitečné a usnadňující pro práci na podobných projektech.

Teoretická část

1 Teoretický úvod

Rychlý růst World Wide Webu za poslední tři desetiletí nepředstavitelně změnil způsob, jakým sdílíme, shromažďujeme a publikujeme data. Firmy, veřejné instituce a soukromí uživatelé poskytují všechny představitelné typy informací a nové komunikační kanály generují obrovské množství dat nejen o lidském chování. Dříve byl zásadní problém, týkající se společenských věd, nedostatek a nedostupnost pozorování. Dnes se tento handicap rychle mění v nadbytek dat. Žádný obrat se obvykle ale neobejde bez problémů. Například tradiční techniky pro sběr a analýzu dat již nemusí stačit k překonání spletitých mas informací. Jedním z důsledků potřeby dát těmto informacím smysl byl vznik „datových vědců“, kteří probírají data a jsou výzkumníky i podniky velmi vyhledávaní.

Spolu s triumfálním rozvojem World Wide Webu přichází na scénu druhý trend, rostoucí popularita a síla open-source software, jako je R. Pro kvantitativní sociální vědce je R jedním z nejdůležitějších statistických softwarů. Jeho růst nabírá obrátek díky aktivní komunitě, která neustále vydává nové balíčky. Přesto je R více než jen bezplatná sada statistik. Obsahuje také rozhraní k mnoha dalším programovacím jazykům a softwarovým řešením, čímž výrazně zjednodušuje práci s daty z různých zdrojů.

V minulosti se často stávalo, že bylo třeba ručně skládat data z odlišných zdrojů a doufalo se, že nevyhnutelné chyby kódování, kopírování a vkládání jsou nesystematické. Je přinejmenším unavující shromažďovat výzkumná data nereprodukovatelným způsobem, který je náchylný k chybám, je těžkopádný a po čase by řadu pracovníků unudil k smrti. V důsledku toho se stále více začleňují procesy sběru a publikování dat do softwarového prostředí, které již pomáhá se statistickými analýzami, tedy prostředí Rstudia. Program nabízí skvělou infrastrukturu pro rozšíření každodenního pracovního postupu na kroky před a po skutečné analýzy dat (Munzert, 2015).

1.1 Primární zdroje a kvalita dat

Odborným způsobem, jak by se mělo postupovat při psaní jakékoliv odborné práce, je doporučeno postupovat při stahování informací z webů. Když se podíváme na online data, musíme mít na paměti jejich původ. Informace mohou být z první ruky, jako jsou například informace o produktech místních farmářů nebo data z druhé ruky, která byla zkopírována z offline zdroje nebo dokonce stažena odjinud. V takovém případě je na zvážení, zda sekundární data použít. Pokud nás například zajímají data určitého prodejce, který má ale své dovozce, bude pro nás primárním web námi vybraného prodejce. Ten je sice v tomto případě sekundárním zdrojem dat, ale pravdou je, že na svých stránkách bude mít uvedené například svoje ceny a svoje množství produktů na skladě. Pokud bychom tedy sbírali data z primárního zdroje, tedy od samotného dovozce, nenaplňovali bychom náš cíl, neboť bychom se dostali k jiným informacím, než nás ve skutečnosti zajímají. Proto je vždy nutností si nejprve co nejpřesněji zadat, která data jsou pro nás podstatná a která irelevantní.

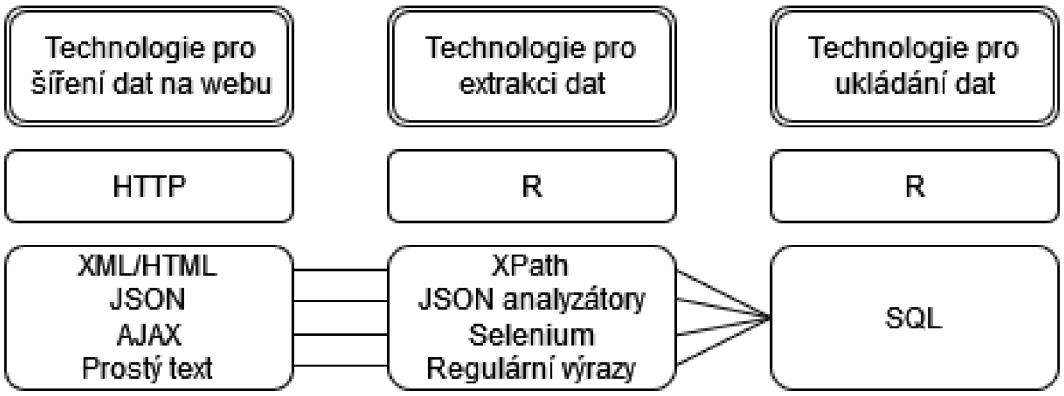
Stejně tak můžeme nahlížet na kvalitu stahovaných dat. Je opět na nás, jak k těmto informacím budeme přistupovat. Pokud se například budeme zajímat o produkty týkající se převážně jednoho konkrétního ročního období, záleží i na termínu, kdy budeme data z webu sbírat. Samotné weby, co se týče zdrojového kódu, mohou být statické či dynamické, o tom v dalších kapitolách v praktické části práce. Pokud ale budeme nahlížet na dynamiku webu v průběhu roku, vyplyne z ní, že například na jaře se zcela jistě na většině webech může objevit jiné portfolio produktů a cen. Proto je potřeba vždy pamatovat na účel našeho sběru dat.

1.2 Technologie pro šíření, extrahování a ukládání webových dat

Stahování dat z webu není vždy jednoduchou záležitostí. Pokud bychom hovořili o snazším sběru, mohlo by jít pouze o data uložená v tabulkách. Jenže obvykle nás zajímají data, která se nacházejí ve složitějších strukturách, než jsou tabulky HTML. Často se jedná o dynamické webové stránky nebo je potřeba získat informace z prostého textu. S automatizovaným sběrem dat pomocí R jsou spojeny určité náklady, což v podstatě znamená, že je nutné získat základní znalosti o webových technologiích.

Existují tři oblasti, které jsou důležité pro sběr dat z webu pomocí R. Na Obrázku 1 najdeme přehled těchto tří oblastí. Každou část si následně popíšeme.

Obrázek 1: Technologie pro šíření, extrahování a ukládání webových dat



Zdroj: (Munzert, 2015)

1.2.1 Technologie pro šíření dat na webu

V prvním sloupci na Obrázku 1 se setkáváme s technologiemi, které umožňují distribuci obsahu na webu. Existuje několik způsobů, jakými jsou data šířena, ale nejdůležitějšími technologiemi v tomto pilíři jsou XML/HTML, AJAX a JSON.

HTML

Na pozadí webu existuje skrytý standard, který strukturuje způsob zobrazování informací, jenž je nazýván jako jazyk hypertextových tagů neboli HTML. Ať už hledáme stránky na Googlu, kontrolujeme svůj bankovní účet na webu banky nebo se pohybujeme na sociálních sítích jako je Instagram, Facebooku nebo YouTube, tak používáním prohlížeče využíváme HTML. Ačkoli HTML není vyhrazený formát pouze pro ukládání dat, často obsahuje informace, které nás primárně zajímají. Data v něm uložená najdeme v textech, tabulkách, seznamech, odkazech nebo jiných strukturách. Důležité je mít na paměti, že je veliký rozdíl mezi tím, jak jsou data prezentována v prohlížeči na jedné straně a jak jsou uložena v HTML kódu na straně druhé. Aby bylo možné automaticky shromažďovat data z webu a zpracovávat je pomocí R, je zapotřebí základní porozumění HTML a způsobu, jakým jsou v něm informace uloženy (Munzert, 2015). Druhá kapitola je věnována konkrétní struktuře a jednotlivým prvkům HTML z pohledu, který bude potřebný pro stahování dat.

XML

Rozšiřitelný značkovací jazyk (XML) je jedním z nejpopulárnějších formátů pro výměnu dat napříč webem. Obdobně jako v případě HTML se jedná o značkovací jazyk. Zatímco HTML se používá k zobrazení informací, hlavním účelem XML je ukládání data. Dokumenty HTML tak prohlížeče interpretují a transformují do estetického výstupu, zatímco XML jsou „pouze“ data zabalená do uživatelsky definovaných tagů. Uživatelsky definované tagy činí XML mnohem flexibilnějším pro ukládání dat než HTML. Dokumenty ve stylu HTML i XML nabízejí přirozené, často hierarchické struktury pro ukládání dat. K rozpoznání a interpretaci takových struktur potřebujeme software, který je schopen těmto jazykům rozumět a adekvátně s nimi zacházet.

JSON

Dalším standardním formátem pro ukládání a výměnu dat, se kterým se na webu často setkáváme, je JavaScript Object Notation (zápis objektů pomocí javascriptu) neboli JSON. Podobně jako XML i JSON je používán mnoha webovými aplikacemi k poskytování dat pro webové vývojáře. Představte si XML i JSON jako standardy, které definují kontejnery pro data ve formátu prostého textu. Pokud vývojáři chtějí například[3] analyzovat trendy na Twitteru, mohou shromažďovat potřebná data z rozhraní, které Twitter nastavil k distribuci informací ve formátu JSON. Hlavním důvodem, proč jsou data přednostně distribuována ve formátech XML nebo JSON, se jeví jejich kompatibilita s mnoha programovacími jazyky a softwary, včetně R. Protože poskytovatelé dat nemohou předem znát software, který se používá k následnému zpracování informací, je vhodnější pro všechny zúčastněné strany, aby distribuovaly data ve formátech s všeobecně uznávanými standardy (Munzert, 2015).

AJAX

AJAX (zkratka z anglického Asynchronous JavaScript and XML) je skupina technologií, která je nyní pevně integrována do sady nástrojů moderního vývoje webu. AJAX hraje nesmírně důležitou roli v tom, že umožňuje webům vyžadovat data asynchronně na pozadí relace prohlížeče a dynamicky aktualizovat svůj vizuální vzhled.

[3] jak je uvedeno v citované knize

Přestože za velkou část sofistikovanosti moderních webových aplikací vděčíme AJAXu, tyto technologie představují pro webové extrakce obtíž a se standardními nástroji R se rychle dostáváme do slepé uličky (Munzert, 2015). Můžeme si představit následující příklad. Na e-shopu máme uvedeny produkty, které jsou nabízeny ve více variantách, ať už se jedná o velikost, barvu či jiné možné rozdělení. V takovém případě se při stahování stránky nikam neposuneme, dokud nezvolíme jednu z variant. Po provedení výběru již není nutné stránku aktualizovat, bude tak provedeno automaticky a nám se zobrazí konkrétní informace o námi zvoleném produktu a můžeme pokračovat v dalším stahování.

Prostý text

Při získávání informací z webu často nakládáme s daty ve formátu prostého textu. Svým způsobem je prostý text součástí každého dokumentu HTML, XML a JSON. Zásadní vlastností, kterou chceme zdůraznit, je, že prostý text jsou nestrukturovaná data. Je tomu tak alespoň pro počítačové programy, které jednoduše čtou textový soubor řádek po řádku. Proto pokud bychom chtěli specifičtěji pracovat se sesbíraným textem, nemusíme již hovořit v kontextu stahování dat, pouze o práci se samotným textem například v podobě stringového řetězce.

HTTP/HTTPS

Abychom mohli načíst data z webu, musíme našemu stroji umožnit komunikaci se servery a webovými službami. Jazykem komunikace na webu je hypertextový přenosový protokol (HTTP). Představuje nejběžnější standard pro komunikaci mezi webovými klienty a servery. Prakticky každá stránka HTML, kterou otevřeme, každý obrázek, který si prohlížíme v prohlížeči, každé video, které sledujeme, je doručeno protokolem HTTP nebo novějším a bezpečnějším HTTPS. Navzdory neustálému používání o protokolu většinou nevíme, protože výměny HTTP obvykle provádějí naše stroje. Dozvíme se, že u mnoha základních aplikací pro extrakci webu se nemusíme příliš starat o podrobnosti HTTP, protože R dokáže bez problémů převzít většinu nezbytných úkolů (Munzert, 2015).

1.2.2 Technologie pro extrakci dat

Druhý sloupec (viz Obrázek 1) představuje technologie pro sběr webových dat, které jsou potřebné k získávání informací ze shromažďovaných souborů. V závislosti na technice, která byla použita ke shromažďování souborů, existují specifické nástroje, jež jsou vhodné pro extrakci dat z těchto zdrojů. Tato část poskytuje první pohled na dostupné nástroje. Výhodou použití prostředí R pro extrakci informací je, že můžeme používat všechny technologie pomocí implementace prostřednictvím sady balíčků.

XPath

Prvním nástrojem, který máme k dispozici, je dotazovací jazyk XPath. Slouží k výběru konkrétních informací z označených dokumentů, jako je HTML, XML nebo jakákoli jejich varianta, například SVG nebo RSS. V typickém úkolu extrakce z webu je volání webových stránek důležité, ale obvykle je pouhým mezikrokem na cestě k dobře strukturovaným a vyčištěným datovým sadám. Abychom mohli naplno využít web jako téměř nekonečný zdroj informací, musíme po identifikaci a stažení příslušných webových dokumentů provést řadu kroků filtrování a extrahování. Hlavním účelem těchto kroků je přetvořit informace, které jsou uloženy v označených dokumentech, do formátů vhodných pro další zpracování a analýzu pomocí statistického softwaru. Tento úkol spočívá ve specifikaci dat, která nás zajímají, a jejich umístění v konkrétním dokumentu a následném přizpůsobení dotazu dokumentu, který extrahuje požadované informace (Munzert, 2015). Tuto možnost použití si ukážeme v praktické části práce.

JSON analyzátory

Na rozdíl od dokumentů HTML nebo XML jsou dokumenty JSON uživatelsky jednodušší a snáze se analyzují. Abychom extrahovali data z JSON, nevycházíme z konkrétního dotazovacího jazyka, ale spoléháme na funkcionalitu R na vysoké úrovni, která odvádí dobrou práci při dekódování dat JSON (Munzert, 2015).

Selenium

Extrahování informací z webových stránek obohacených o AJAX je pokročilejší a složitější scénář. Účinnou alternativu ke spouštění webových požadavků z konzole Rstuida představuje rámec Selenium jako praktický přístup k získání kontroly nad webovými daty. Selenium umožňuje přesměrovat příkazy, jako jsou kliknutí myší nebo vstupy z klávesnice do okna prohlížeče, právě pomocí prostředí R. Díky práci přímo

v prohlížeči je Selenium schopen obejít některé problémy, které se týkají webových stránek obohacených o AJAX (Munzert, 2015). Využití Selenia bude více popsáno v praktické části.

Regulární výrazy

Ústředním úkolem webového stahování je shromáždit relevantní informace pro náš výzkumný problém z množství textových dat. Obvykle se zabýváme systematickými prvky v textových datech, zvláště pokud chceme na výsledná data aplikovat kvantitativní metody. Systematickými strukturami mohou být čísla nebo názvy jako ceny nebo tagy. Jednou z technik, kterou můžeme použít k extrakci systematických složek informace, jsou regulární výrazy. Regulární výrazy jsou v podstatě abstraktní sekvence řetězců, které odpovídají konkrétním, opakujícím se vzorům v textu. Kromě jejich použití k extrahování obsahu z dokumentů ve formátu prostého textu je můžeme také použít na dokumenty HTML a XML, abychom identifikovali a extrahovali části dokumentů, které nás zajímají. I když je často vhodnější používat dotazy XPath na značkovací dokumenty, regulární výrazy mohou být užitečné, pokud jsou informace skryty v atomických hodnotách (Munzert, 2015).

Těžba textu

Kromě získávání smysluplných informací z textových dat ve formě čísel nebo názvů máme k dispozici ještě druhou techniku, kterou je text mining. Použití postupů v této třídě technik umožňuje výzkumníkům klasifikovat nestrukturované texty na základě podobnosti jejich slovních použití. Pro pochopení konceptu dolování textu je užitečné zamyslet se nad rozdílem mezi manifestní a latentní informací. Zatímco první popisuje informace, které jsou specificky spojeny s jednotlivými termíny, jako je adresa nebo měření teploty, druhý odkazuje na textové štítky, které nejsou explicitně obsaženy v textu. Při analýze vybraných zpráv je čtenáři mohou zařadit do určitých tematických kategorií, například politiky, médií nebo sportu. Postupy dolování textu poskytují řešení pro automatickou kategorizaci textu. To je užitečné zejména při analýze webových dat, která často přicházejí ve formě neoznačeného a nestrukturovaného textu (Munzert, 2015).

1.2.3 Technologie pro ukládání dat

Poslední, třetí sloupec technologií pro práci s webovými daty se zabývá možnostmi pro ukládání informací (viz Obrázek 1). R je většinou vhodný pro správu technologií ukládání dat, jako jsou databáze. Obecně lze říct, že spojení mezi technologiemi pro extrakci informací a technologiemi pro ukládání dat je méně zřejmé. Nejlepší způsob ukládání nemusí nutně záviset na původu dat.

SQL

Jednoduché a každodenní procesy, jakými jsou například online nakupování, procházení katalogů knihoven, převod peněz z bankovního účtu nebo dokonce nákup pečiva v supermarketu, to vše zahrnuje databáze. Málokdy si uvědomujeme, že databáze hrají tak důležitou roli, protože s nimi neinteragujeme přímo. Databáze pracují spíše v zákulisí. Kdykoli jsou data klíčová pro projekt, weboví administrátoři se budou spoléhat na databáze kvůli jejich spolehlivosti, efektivitě, uživatelskému přístupu, prakticky neomezené velikosti dat a možnostem vzdáleného přístupu. Co se týče automatizovaného sběru dat, databáze jsou zajímavé ze dvou důvodů. Prvním je možnost příležitostného udělení přímého přístupu k databázi, se kterou bych si měli být schopni poradit. Druhým důvodem se jeví, že ačkoliv Rstudio má mnoho zařízení pro správu dat, může být vhodnější ukládat data do databáze než do jednoho z nativních formátů. Pokud například pracujeme na projektu, kde je potřeba data zpřístupnit online, nebo pokud máme různé strany shromažďující konkrétní části našich dat, databáze může poskytnout potřebnou infrastrukturu. Navíc, pokud jsou data, která potřebujeme shromáždit, rozsáhlá a musíme s nimi často manipulovat, má také smysl nastavit databázi pro rychlost, s jakou je lze dotazovat (Munzert, 2015).

2 HTML

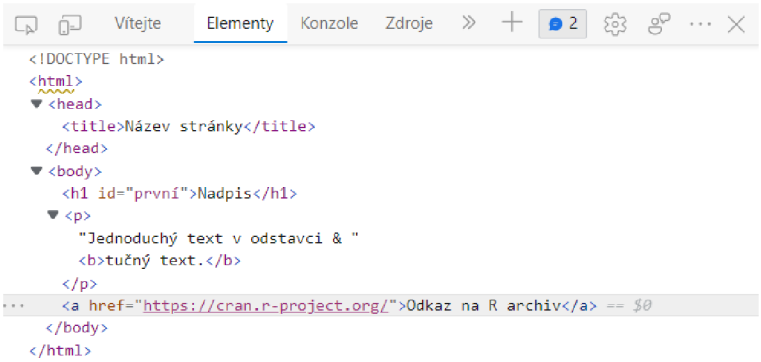
HTML je jazyk pro prezentaci obsahu na webu, který jako první navrhl Tim Berners-Lee roku 1989. Standard se od prvního zavedení neustále vyvíjí, nejnovější inkarnací je HTML5, který je vyvíjen World Wide Web Konsociem (z anglického Consortium, ve zkratce W3C) a Web Hypertext Application Technology Working Group (WHATWG) (Munzert, 2015).

Ačkoli každá revize HTML zavedla nové funkce a restrukturalizovala ty staré, základní gramatika HTML dokumentů se v průběhu let příliš nezměnila a pravděpodobně zůstane v dohledné době poměrně stabilní, což z ní činí jeden z nejdůležitějších standardů pro práci s webem. Tato kapitola představuje základy HTML z pohledu webového sběrače dat. Představena je logika značkovacích jazyků obecně a syntaxe HTML jako specifické instance značkovacího jazyka.

2.1 Prezentace zdrojového kódu

Soubor HTML není v podstatě nic jiného než prostý text, který lze otevřít a upravit pomocí libovolného textového editoru. To, co dělá HTML tak mocným, je jeho označená struktura. Značení HTML umožňuje definovat části dokumentu, které je třeba zobrazit jako nadpisy, části obsahující odkazy, části, jež by měly být uspořádány jako tabulky a mnoho dalších formulářů. Definice tagů se spoléhají na předdefinované sekvence znaků (tzv. tagy), které obklopují části textu. Tagy sdělují prohlížečům, jak je dokument strukturován a funkce různých částí. Pro představu je na Obrázku 2 jednoduchý zdrojový kód dokumentu html.html.

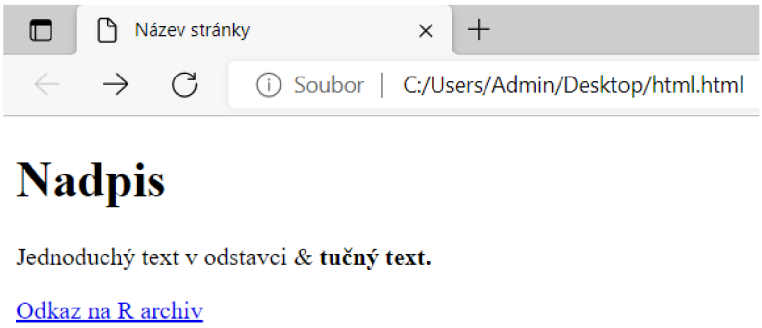
Obrázek 2: Zdrojový kód jednoduchého dokumentu html.html



Zdroj: vlastní

Výstup, který vidíme zobrazený ve svém prohlížeči (viz Obrázek 3), tedy není samotný HTML dokument, ale jeho interpretace. Pojďme si tuto myšlenku rozvést. Zdrojový kód obdobný s Obrázkem 2 dostaneme klepnutím pravým tlačítkem myši na vybranou část prohlížeče a z kontextové nabídky výběrem zobrazit zdrojový kód stránky. Za běžných okolností není důvod ke kontrole zdrojového kódu, ale při online sběru dat je to zásadní dovednost.

Obrázek 3: Vzhled stránky jednoduchého dokumentu html.html v prohlížeči



Zdroj: vlastní

Mohlo by se zdát, že mnoho informací ze zdrojového kódu se při interpretaci dokumentu ztratí. Koneckonců, ve zdrojovém kódu je podstatně více textu než jen pouhý výstup, který vidíme na Obrázku 3. Ve zdrojovém kódu je značné množství informací, které obsahují pokyny pro prohlížeč, a které nejsou vytištěny na obrazovce. Přesto je část informací ve skutečnosti zobrazena, ale jemnějšími způsoby. Podívejte se například na první řádek karty prohlížeče na Obrázku 3, kde je nadpis, který byl definován ve zdrojovém kódu jako <h1 id='první'>Nadpis</h1>.

2.2 Pravidla syntaxe

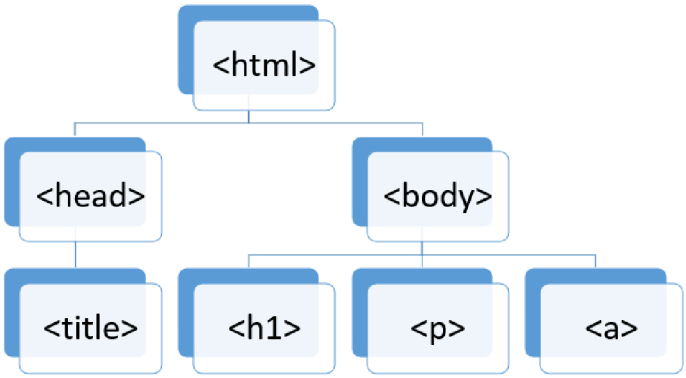
Nyní, když jsme se dozvěděli o rozdílu mezi interpretovanou verzí dokumentu a jeho zdrojovým kódem, pojďme se hlouběji ponořit do pravidel a konceptů, které jsou základem HTML. Tento jazyk má hierarchickou strukturu tvořenou prvky, které se skládají z počátečních tagů (např. <h1>), v nich volitelných atributů (id='první'), koncových tagů (jako </h1>) a obsahu, tedy textu mezi počátečním a koncovým tagem (vše mimo ostré závorky).

Ačkoli existuje několik verzí HTML je možné se setkat s weby, které dodržují starší standardy[4]. V každém případě jsou pro účely sběru dat rozdíly verzí zanedbatelné (Munzert, 2015).

2.2.1 Stromová struktura

Podívejme se znovu na zdrojový kód html.html na Obrázku 2. Ignorujeme-li prozatím <!DOC-TYPE html>, prvním prvkem v příkladu je prvek <html>. Mezi tagy tohoto prvku se otevírá a opět zavírá několik tagů: <head>, <title> a <body>. Tagy <head> a <body> jsou přímo uzavřeny v prvku <html>; prvek <title> je uzavřen tagem <head>. Dobrým způsobem, jak popsat více vrstev dokumentu HTML, je stromová analogie. Obrázek 4 ukazuje jednoduchou stromovou strukturu html.html. Element <html> je kořenový element, který se dělí na dvě větve, <head> a <body>. Za <head> následuje další větev s názvem <title> a za <body> další tři větve s názvem <h1>, <p> a <a>.

Obrázek 4: Stromová struktura html.html



Zdroj: vlastní

Prvky musí být přísně vnořeny do sebe v kvalitně vytvořeném a platném souboru HTML. Dvojice počátečních a koncových tagů musí být zcela uzavřena jinou dvojicí tagů. Zjevné porušení tohoto pravidla by bylo například:

<body><h1>toto</body>JE ŠPATNĚ</h1>

[4] v praktické části budeme pracovat s dokumenty, které dodržují pravidla HTML5.

2.2.2 Definice typu dokumentu

Jak můžeme vidět opět na Obrázku 2, první řádek HTML zní <!DOCTYPE html>. Obsahuje tzv. definici typu dokumentu (DTD), jež informuje prohlížeč o verzi HTML standardu, kterou dokument dodržuje. Jak již bylo zmíněno na začátku této kapitoly, HTML zaznamenalo za svou existenci určité přeformulování pravidel, které by mohlo vést k nesprávným interpretacím, pokud by HTML verze dokumentu nebyla explicitně uvedena. Nám postačí vědět, že DTD se nacházejí (pokud jsou zahrnuty) v prvním řádku dokumentu HTML. Níže naleznete různé DTD dle verze HTML.

- Pro verzi HTML5 (jako v našem případě):

<!DOCTYPE HTML>

- Pro starší verzi HTML 4.01 by první řádek měl vypadat následovně:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">

2.2.3 Vyhrazené a speciální znaky

Vyhrazené znaky se používají pro účely ovládání v jazyce. Obsah HTML je psán jako prostý text, což platí jak pro označení, tak pro obsahovou část dokumentu. Protože některé znaky jsou potřebné pro označení, nelze je v obsahu použít doslova. Například < a > se používají k vytváření tagů v HTML. Jsou to značkovací znaky. Představme si, že chceme v prohlížeči zobrazit něco takového: 1 < 4, ale 8 > 5. Není možné je zapsat do HTML souboru takto doslovně 1 < 4 but 8 > 5 (Munzert, 2015).

Entity

Příkladem, ve kterém již předpokládáme konflikt, je znak < a >. Primárně je potřeba tyto znaky interpretovat jako znak obklopující název tagu. Aby se znaky zobrazily doslova v okně prohlížeče, HTML se spoléhá na specifické sekvence znaků nazývané znakové entity nebo jednoduše entity. Všechny začínají ampersandem & a končí středníkem ;. Tedy < a > mohou být zahrnuty do obsahu souboru s jejich výrazy entity > a <. Při interpretaci souboru HTML nyní prohlížeč zobrazí znak, který tyto entity představují. Výše uvedený příklad je proto třeba přepsat následovně: <p>1 > 4 but 8 < 5 </p>.

Dalším důležitým příkladem jsou mezery a zalomení řádků ve zdrojovém kódu HTML, které se nepřekládají přímo do mezer a zalomení řádků v prezentaci prohlížeče. Zatímco konce řádků jsou zcela ignorovány, libovolný počet po sobě jdoucích mezer je prezentován jako jedna mezera. Abychom do interpretované verze dokumentu vnutili mezery, používáme entitu pevného prostoru   a tag
 pro zalomení řádku. Například takto:

První
   další
     další
 poslední

Vzhledem k tomu, že dokumenty HTML lze psát v mnoha jazycích, které často obsahují jiné než jednoduché latinské znaky, jako je O¨, E´ nebo Ø, existuje rozsáhlý seznam entit, který nalezneme například na stránce https://unicode-table.com/.

2.2.4 Tag, element a atribut

Elementy

Jak již bylo zmíněno, prostý text je přeměněn na HTML dokument pomocí tagů, které dokáže prohlížeč interpretovat. Lze si je představit jako pojmenované závorky, které uzavírají obsah a definují jeho strukturální funkci. Například tagy <title> v našem úvodním příkladu zhodnotily přiložený text jako nadpis, který se zobrazí v záhlaví karty prohlížeče. Kombinace počátečního tagu, obsahu a koncového tagu se nazývá prvek.

Počáteční tagy a koncové tagy jsou také známé jako otevírací a uzavírací. Tagy jsou vždy ohraničeny < a >, aby se odlišily od obsahu. Počáteční a koncové tagy mají stejný název, ale před koncovým tagem je lomítko /. Při odkazu na prvek je běžné vynechat lomené závorky a použít pouze název ve značkách, jako je například body tag, title tag a tak dále. Někdy zjistíme, že prvky a tagy se ve skutečnosti používají synonymně. Proto pokud se odkazujeme na počáteční tag (například <name>) odkazujeme tím na celý prvek.

Ačkoli se doporučuje, aby každý prvek měl počáteční a koncový tag, není to běžná praxe pro všechny typy prvků. Například tag
 označuje zalomení řádku a není uzavřen protějškem </br>. Tagy lze také uzavřít v rámci počátečního tagu přidáním lomítka na konec, jako v <body/>. Takové prvky nazýváme prázdné, protože neobsahují žádný obsah. Jinak by musely být zapsány jako <body></body>. Protože

standardní HTML nerozlišuje velká a malá písmena, je možné napsat tag jako <tag>, <TAG>, <Tag> nebo jakoukoli jinou kombinaci velkých a malých písmen. Přesto se doporučuje vždy používat malá písmena (Munzert, 2015).

Tagů je celá řada, protože všechny ale znát nepotřebujeme, představíme si tedy pouze několik tagů potřebných pro stahování webu.

Atributy

Další vlastností tagů jsou atributy. Příkladem široce používaného atributu je námi použitý: Odkaz na R archiv

Atributy jsou vždy umístěny v rámci počátečního tagu hned za jeho názvem. Tag může obsahovat více atributů, které jsou jednoduše odděleny mezerou. Atributy jsou vyjádřeny v páru obsahujícím jméno atributu a jeho hodnotu, jako například v href="https://cran.r-project.org/". Hodnota může být uzavřena jednoduchými nebo dvojitými uvozovkami. Pokud však samotná hodnota atributu obsahuje jeden typ uvozovek, je potřeba k uzavření hodnoty použít druhý typ: <příklad text='Řekla: "Máme krásný den", a pak se usmála.'> nebo <příklad text="Řekla: 'Máme krásný den', a pak se usmála.">.

Atributy obecně umožňují specifikaci možností, jak by se mělo nakládat s obsahem tagu. Které atributy jsou povoleny, závisí na konkrétním tagu.

Příklady tagů

Kotevní tag <a>

Tag <a> mění HTML z pouhého značkovacího jazyka na hypertextový značkovací jazyk tím, že umožňuje HTML dokumentům odkazovat na jiné dokumenty. Velká část navigace mezi weby v prohlížečích funguje prostřednictvím kotevních prvků. V našem případě umožňuje spojení textu: „Odkaz na R archiv“ s hypertextovým odkazem „https://cran.r-project.org/“, který ukazuje na jinou adresu. Atribut href="https://cran.r-project.org/" určuje cíl odkazu. Prohlížeče takové prvky automaticky formátují takovým způsobem, že obsah podtrhnou a umožní na něj kliknout (jak vidíme na Obrázku 3).

Atribut href umožňuje také odkazovat na konkrétní části dokumentu. Je možné propojit <a> v dokumentu, aby byla navigace na webu pohodlnější. Na stránce zadáme

referenční bod referenční bod a následně odkaz, kterým se k němu dostaneme Odkaz na referenční bod.

**Tag odstavce **

Tag označí po něm následující obsah jako odstavec a zajistí, aby byly konce řádku vloženy před a za jeho obsah:

Text v odstavci oddělený od zbytku textu zalomením řádku.

Tagy nadpisu <h1>, <h2>, <h3>, ...

Aby bylo možné definovat různé úrovně nadpisů (úroveň 1 až 6) HTML poskytuje řadu tagů <h1>, <h2>,...až po <h6>. Pro představu bychom si mohli přirovnat jednotlivé úrovně nadpisů k úrovním kapitol této práce:

<h1>Kapitola</h1><h2>1 Kapitola</h2>

Výpis obsahu pomocí <ul>, <ol> a <dl>

Pro seznam obsahu existuje několik tagů. Používají se v závislosti na tom, zda obtékají uspořádaný seznam (<ol>), neuspořádaný seznam (ul) nebo popisný seznam (<dl>). První dva tagy využívají k definování položek seznamu vnořené prvky <li>, zatímco třetí potřebuje dva další prvky: <dt> pro klíčové slovo a <dd> pro jeho popis. Příkladem neuspořádaného seznamu by bylo: Banány
Hrušky
Jablka
Švestky

**Tagy organizace a **

Dalším způsobem, jak definovat vzhled částí dokumentu HTML, jsou tagy a . Zatímco tagy a samy o sobě nemění vzhled obsahu, který obklopují, používají se k seskupování částí dokumentu. Tag se umožňuje definování skupin přes řádky, tagy a odstavce, zatímco tag se používá pro tzv. in-line seskupování, čili nezalamuje před sebou a za sebou řádky.

Seskupování částí dokumentu HTML je užitečné v kombinaci s kaskádovými styly (CSS), jazykem pro popis rozvržení HTML a dalších značkovacích dokumentů, jako jsou například XML. Níže jsou uvedeny příklady definic dvou stylů. První definice stylu platí pro všechny prvky třídy styl, zatímco druhá platí totéž pro prvky

:

```
div.styl { color:blue;
        font-family:"Courier New";
        font-size:110% }
span.styl { color:blue;
        font-family:"Courier New";
        font-size:110% }
```

Tyto definice stylů jsou běžně uloženy v samostatných souborech CSS, například stylKoduR.css, a později jsou zahrnuty prostřednictvím tagů <link> v záhlaví:

```
<link href="htmlresources/stylKoduR.css"
        rel="stylesheet" type="text/css"/>
```

Seskupení jsou následně v dokumentu předány prvku pomocí dalšího atributu třídy (class):

```
<div class="styl"><p>&#62; x &#60;&#45; 1</p></div>vypíše hodnotu
<span class="styl">&#62; x</span>
```

Účelem CSS je oddělit obsah od rozvržení a zlepšit tak přístupnost dokumentu. Definování stylů mimo HTML a jejich přiřazení pomocí atributu class umožňuje webovému návrháři opakovaně používat styly napříč prvky a dokumenty. To umožňuje vývojářům změnit styl na jednom místě, tedy v souboru CSS, s efekty na všechny prvky a dokumenty používající tento styl. V první řadě by si každý měl vždy dát záležet na stylu. Jelikož je CSS pro vývojáře tak užitečné, tagy , a třída class se používají často. Poskytují tak dokumentu HTML strukturu, kterou můžeme použít k identifikaci, kde jsou uloženy požadované informace (Munzert, 2015).

Tag <form> a jeho doprovodné prvky

Pokročilou funkcí HTML jsou formuláře. Tyto umí více než jen rozvržení obsahu. Umožňují uživatelům komunikovat se servery odesíláním dat na ně namísto toho, aby od nich pouze přijímali. Formuláře jsou představeny tagem <form> a podporovány dalšími tagy, jako jsou <fieldset>, <input>, <textarea>, <select> a <option> a jejich příslušné atributy. Tato obousměrná výměna informací mezi

uživatelem a serverem umožňuje dynamičtější procházení. Případy, kdy formuláře používáme denně, jsou vyhledávače typu Google. Do textového pole zadáme dotaz a na základě našeho požadavku se zavolá nový web. Příkladem k vysvětlení různých konceptů formulářů HTML může být:

<form name="submitPW" action="Passed.html" method="get">

password:

<input name="pw" type="text" value="">

<input type="submit" value="SubmitButtonText">

</form>(Munzert, 2015).

Tagy tabulky <table>, <tr>, <td> a <th>

Další skupina prvků umožňuje HTML zobrazovat tabulky. Pro začátek tabulky používáme <table>. Nové řádky začínáme pomocí <tr>. V rámci <tr> můžeme použít <td> pro definování buněk nebo <th> pro buňky záhlaví.

<table>

<tr> <th>Rank</th> <th>Město</th> <th>PSČ</th> </tr>

<tr> <td>1</td> <td>Vodňany</td> <td>389 01</td> </tr>

<tr> <td>2</td> <td>České Budějovice 5</td> <td>370 05</td> </tr>

</table>

3 Extrakce pomocí jazyka R

Statistický software R, který využívá jazyka R se dostal do popředí díky své flexibilitě jako efektivní jazyk, který staví most mezi vývojem softwaru a analýzou dat. Jednou ze silných stránek jazyka R[5] je například snadnost, s jakou jej lze vyvíjet a rychle se přizpůsobovat různým potřebám pocházejícím z komunity pro správu a analýzu dat a zároveň využívat jiné jazyky za účelem poskytování výpočetně efektivních řešení.

Při provádění web scrapingu se obvykle dostáváme do kontaktu s HTML ve dvou krocích. Nejprve zkontrolujeme obsah na webu a prozkoumáme, zda je atraktivní pro další analýzy. Za druhé importujeme HTML soubory do Rstudia a extrahujeme z nich informace. K analýze HTML dochází v obou krocích – pomocí prohlížeče, aby se obsah HTML pěkně zobrazil, a také pomocí analyzátorů v jazyce R pro vytváření užitečných reprezentací dokumentů HTML v našem programovacím prostředí (Munzert, 2015).

3.1 Výhody jazyka R

1. Rstudio je volně a snadno přístupný software využívající jazyk R. Můžeme si jej stáhnout, nainstalovat a používat kdekoli a kdykoli chceme. Nebýt specialistou na drahé proprietární programy má obrovské výhody, protože nejsme závislí na ochotě zaměstnavatelů platit licenční poplatky.
2. Pro softwarové prostředí s primárně statistickým zaměřením má R velkou komunitu, která neustále vzkvétá. R je používáno různými obory, jako jsou sociální vědci, lékařští vědci, psychologové, biologové, geografové, lingvisté a také v podnikání. Tato řada nám umožňuje sdílet kód s mnoha vývojáři a profitovat z kvalitně zdokumentovaných aplikací v různých nastaveních.
3. R je open source. To znamená, že můžeme snadno vysledovat, jak funkce pracují, a upravit je s minimálním úsilím. Znamená to také, že úpravy programu nejsou řízeny exkluzivním týmem programátorů, který se o produkt stará. I když nemáte zájem přispívat k vývoji R, stále budeme těžit z výhod přístupu k široké

[5] dále pouze R

škále volitelných rozšíření (balíčků). Jejich počet neustále roste a mnoho stávajících balíčků je často aktualizováno. Hodnotné přehledy oblíbených témat v používání R je možné nalézt na adrese http://cran.r-project.org/web/views/.

4. Rstudio je přiměřeně rychlý v běžných úlohách. Existují dokonce rozšíření pro urychlení R, například zpřístupněním C kódu z R, jako je balíček Rcpp.
5. Software R je mocný při vytváření vizualizací dat. Ačkoli to není zjevná výhoda pro sběr dat, při každodenním pracovním postupu by uživateli jistě chyběla grafická zařízení R.
6. Práce v Rstuidu je založena hlavně na příkazovém řádku. To může znít jako nevýhoda pro nováčky, ale je to jediný způsob, jak umožnit produkci reprodukovatelných výsledků ve srovnání s programy typu point-and-click.
7. Software R není vybíravý ohledně operačních systémů. Obecně lze spustit pod Windows, Mac OS a Linux.
8. Hlavní důvod pro zvolení prostředí R je, že se v něm můžeme pohybovat od začátku do konce. Příklad běžného výzkumného procesu se vyznačuje neustálým přepínáním mezi programy. Pro každý krok (sběr, manipulaci, analýzu i publikování dat) je vždy potřeba použití jiného programu. Výzkumný proces využívající R (jak je popsáno v praktické části této práce), probíhá v rámci jediného softwarového prostředí. V kontextu web scrapingu a zpracování textu to znamená, že se pro daný úkol uživatel nemá učit jiný programovací jazyk. Co je nutné se naučit, jsou některé základy značkovacích jazyků HTML a XML a logika regulárních výrazů a XPath, ale operace se provádějí z R (Munzert, 2015).

3.2 Balíčky a funkce

Pro práci s jednotlivými balíčky[6] je potřebné mít tyto balíčky instalovány. Pokud tomu tak není, je zapotřebí je instalovat pomocí příkazu nebo v nastavení vybrat úložiště, z něhož je chceme instalovat, pokud se nenachází na úložišti CRAN. Balíčků, které již existují je obrovské množství, proto si představíme jen ty balíčky a funkce (kromě základních, které není třeba explicitně zmiňovat), které nám budou nejužitečnější při stahování webu.

[6] rozšíření prostředí R, někdy také knihovny

3.2.1 Robotstxt

Balíček Robotstxt poskytuje funkce pro stahování a analýzu souborů 'robots.txt'. V konečném důsledku tento balíček usnadňuje kontrolu, zda mají roboti (pavouci, crawler, scrapery, ...) povolen přístup ke konkrétním zdrojům v doméně. Představme si některé jeho funkce, které využijeme v praktické části práce.

- paths_allowed() kontroluje, zda má robot oprávnění pro přístup ke stránkám (Meissne et al., September 3, 2020).

3.2.2 Rvest

Jedná se o tzv. „wrappery“ kolem balíčků 'xml2' a 'httr', které usnadňují stahování HTML a XML a následnou manipulaci s nimi. Představme si některé jeho funkce, které využijeme v praktické části práce.

- html_text() získá text prvku.

Tyto funkce lze použít jak v jednotném, tak množném čísle:

- html_attr() získá jeden atribut prvku; html_attrs() získá všechny atributy. Příklad použití je následující: html %>% html_elements("a") %>% html_attrs().
- html_element() a html_elements() najdou prvek HTML pomocí selektorů CSS nebo výrazů XPath. Selektory CSS jsou užitečné zejména ve spojení s https://selectorgadget.com/, což velmi usnadňuje nalezení požadovaného selektoru. Příklad použití je následující: li <- html %>% html_elements("li").
- html_table() analyzuje html tabulku do datového rámce. Příklad použití je následující: sample3 %>% html_element("table") %>% html_table() (Wickham, August 20, 2022).

3.2.3 Dplyr

Dplyr je rychlý a konzistentní nástroj pro práci s datovými rámci jako s objekty. Funkce, kterou budeme využívat:

- data.frame() vytváří datové rámce, těsně propojené kolekce proměnných, které sdílejí mnoho vlastností matic a seznamů, používané jako základní datová

struktura většinou modelovacího softwaru R ("A Grammar of Data Manipulation", August 31, 2022).

3.2.4 XML

XML balíček nabízí přístupy jak pro čtení, tak pro vytváření dokumentů XML (a HTML) (včetně DTD), a to jak místních, tak přístupných přes HTTP nebo FTP. Představme si některé jeho funkce, které využijeme v praktické části práce.

- xmlParse() a htmlParse() používají výchozí hodnotu pro parametr useInternalNodes TRUE, tzn. že pracují s interními uzly a vracejí je. Ty pak lze vyhledávat pomocí výrazů XPath přes xpathSApply().
- xpathSApply() je verze xpathApply(), která se pokouší zjednodušit výsledek, pokud jej lze převést na vektor nebo matici místo toho, aby byl ponechán jako seznam (Temple Lang et al., June 10, 2022).

3.2.5 Stringr

Stringr je konzistentní, jednoduchá a snadno použitelná sada kolem fantastického balení „stringi". Všechny názvy funkcí a argumentů (a pozic) jsou konzistentní, všechny funkce pracují s vektory, i s nulovou délkou stejným způsobem. Výstup z jedné funkce lze snadno vložit na vstup jiné.

- str_count() spočítá počet shod v řetězci (Wickham, August 21, 2022).

3.2.6 RSelenium

RSelenium je balíček, který představuje vazby R pro rozhraní WebDriver API v Selenium 2. Ke komunikaci se serverem Selenium používá protokol JsonWireProtocol (Harrison & Yeong Kim, September 2, 2022).

Hlavní referenční třídou RSelenia je třída s názvem remoteDriver. Chceme-li se připojit k serveru, musíme vytvořit instanci nového vzdáleného ovladače s příslušnými možnostmi. Představme si některé funkce, které využijeme v praktické části práce.

- clickElement() pro kliknutí na prvek.

- findElement(použití, hodnota) pro hledání prvku na stránce, počínaje kořenem dokumentu. Vstupy jsou: použití, což je schéma lokátoru, které se použije k vyhledání prvku, dostupná schémata jsou: „class name“, „css selector“, „id“, „name“, „link text“, „partial link text“, „tag name“, „xpath“.
- Výchozí hodnota je „xpath“.
- getElementText() pro získání textu, obsahu prvku.
- getPageSource(...) pro získání aktuálního zdrojového kódu stránky.
- goBack() pro stisknutí tlačítka Zpět v prohlížeči.
- navigate(url) pro přechod na danou adresu URL.
- sendKeysToElement(sendKeys) pro odeslání sekvence stisknutí kláves na prvek. Stisky kláves jsou odeslány jako seznam. Prostý text se zadává jako nepojmenovaný prvek seznamu. Položky klávesnice jsou definovány v „selKeys“ a měly by být uvedeny s názvem „key“.
- open() pro odeslání požadavku na vzdálený server k vytvoření instance prohlížeče.
- quit() pro smazání relace a zavření otevřeného prohlížeče.
- refresh() pro opětovné načtení aktuální stránky
  (Harrison & Yeong Kim, September 2, 2022).

3.2.7 Tidyverse

Tidyverse je sada balíčků, které fungují v harmonii, protože sdílejí společné reprezentace dat a design „API“. Tento balíček je navržen tak, aby se dalo snadno nainstalovat a načíst více 'tidyverse' balíčků v jediném kroku (“Easily Install and Load the 'Tidyverse'”, July 18, 2022).

3.2.8 Netstat

Rozhraní R pro obslužný program příkazového řádku 'netstat' používané k získávání a analýze běžně používaných síťových statistik, včetně dostupných a používaných portů protokolu TCP (Transmission Control Protocol).

- free_port() pro načtení portu protokolu TCP, který se aktuálně nepoužívá

  (Condylios et al., August 28, 2022).

3.3 Etické zásady při dolování webových dat

Stahování se týká celého data miningu a souvisejících technik, které se používají k automatickému objevování a extrahování informací z webových dokumentů a služeb. Při použití v obchodním kontextu a aplikování na určitý typ osobních údajů pomáhá společnostem vytvářet podrobné profily zákazníků a získávat marketingové informace. Stahování webu však představuje hrozbu pro některé důležité etické hodnoty, jako je soukromí a individualita. Ztěžuje jednotlivci autonomně řídit odhalování a šíření dat o svém soukromém životě. Při studiu těchto hrozeb rozlišujeme mezi „těžbou obsahu a struktury“ a „dolováním podle využití“. Dolování webového obsahu a struktury je důvodem k obavám, pokud jsou data publikovaná na webu v určitém kontextu těžena a kombinována s jinými daty pro použití v úplně jiném kontextu. Dolování z používání webu vyvolává obavy o soukromí, když jsou uživatelé webu vysledováni a jejich akce jsou analyzovány bez jejich vědomí. Kromě toho se oba typy extrakce často používají k vytváření souborů zákazníků se silnou tendencí posuzovat a zacházet s lidmi na základě skupinových charakteristik, dle jejich vlastností a předností (označované jako deindividualizace). Přestože existuje celá řada řešení problémů s ochranou soukromí, žádné z těchto řešení nenabízí dostatečnou ochranu. Pouze balíček kombinovaných řešení sestávající z řešení na individuální i kolektivní úrovni může přispět k uvolnění určitého napětí mezi výhodami a nevýhodami stahování webu. Hodnoty soukromí a individuality by měly být respektovány a chráněny, aby bylo zajištěno, že je s jednotlivci souzeno a zacházeno spravedlivě. Lidé by si měli být vědomi závažnosti situace a neustále o těchto etických otázkách diskutovat. To by měla být společná odpovědnost sdílená webovými těžaři (botadoptéry a vývojáři), uživateli webu a vládami (van Wel & Royakkers, 2004).

Před extrakcí webové stránky je dobré zkontrolovat, zda nabízí aplikační programové rozhraní (API), které uživatelům umožňuje rychle sbírat data přímo z databáze webu. Ačkoli by vám zde uvedené metody měly pomoci stahovat mnoho webových stránek, některé weby mohou zobrazovat informace v neobvyklých formátech, které znesnadňují jejich extrakci. Než vytvoříme kompletní scrapingový nástroj pro web, stojí za to zkontrolovat, zda si můžeme stáhnout a extrahovat informace z jedné stránky.

Při stahování webu je dobré vkládat mezi stahování webových stránek pauzy, protože to pomáhá rozprostřít provoz na stránkách. Webové extrakční nástroje mohou být vyloučeny z internetové stránky, pokud na ni kladou nepřiměřený důraz. Než zahájíme svůj vlastní projekt, bylo by dobré zkontrolovat zásady našeho institucionálního kontrolního výboru pro stahování webu; možná budeme muset vytvořit aplikaci pro etiku, nebo mohou být naše cílová data klasifikována jako archivní, která nevyžadují aplikaci pro etiku. Obecně platí, že veškeré informace uložené za uživatelským jménem a heslem jsou považovány za soukromé a neměly by být smazány z webu (Bradley & James, 2019).

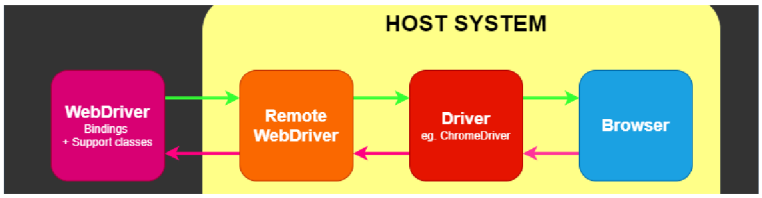
Je zapotřebí zdůraznit tyto body, než začneme sklízet gigabajty informací

1. Vždy mít na paměti, odkud data pocházejí, a kdykoli je to možné, poskytnout uznání těm, kteří je původně shromáždili a zveřejnili.
2. Neporušovat autorská práva, pokud plánujeme znovu publikovat data, která jsme našli na webu. Pokud jsme informace neshromáždili sami, je pravděpodobné, že k jejich reprodukci potřebujeme povolení od vlastníků.
3. Nedělejme nic nezákonného! Chceme-li získat představu o tom, co můžeme a nemůžeme při shromažďování dat dělat, podívejme se na Justia BlawgSearch (http://blawgsearch.justia.com/), což je vyhledávací stránka pro legální blogy. Hledání záznamů označených jako „web scraping“ nám může pomoci udržet si aktuální informace o právním vývoji a nedávných rozsudcích. Electronic Frontier Foundation (http://www.eff.org/) byla založena již v roce 1990 na obranu digitálních práv spotřebitelů a veřejnosti. Doufáme však, že se na jejich pomoc nikdy nebudeme muset spoléhat. (Munzert, 2015)

Metodika

Námi vybraný e-shop, podobně jako většina, je tvořen dynamicky. To znamená, že je zapotřebí provádět ve webovém prohlížeči určité kroky, po nichž se načtená stránka neaktualizuje, pouze se sama změní/upraví. Z toho důvodu není možné ze stránky stahovat bez použití vzdálené komunikace. Bez ní bychom byli schopni extrahovat pouze atributy, které se nemění, jsou statické na každé stránce. Jak již bylo zmíněno v předchozí kapitole, pro extrakci atributů produktů z webu budeme využívat nástroj Selenium v prostředí softwaru Rstudio pro vzdálenou komunikaci. Jak je uvedeno na Obrázku 5, je zapotřebí mít na počítači nainstalovaný takový prohlížeč (Browser), jež je možné ovládat pomocí WebDriver protokolu (v našem případě to je Google Chrome), dále ovladač (na obrázku anglicky Driver) pro zprostředkování komunikace mezi prohlížečem a Seleniem, a jelikož pracujeme v prostředí Rstudia, také v něm nainstalovaný balíček RSelenium. Díky němu budeme schopni implementovat protokol WebDriver potažmo vzdálený WebDriver (Remote WebDriver). Jak vidíme na zmiňovaném obrázku, Remote WebDriver běží na stejném systému jako ovladač a prohlížeč.

Obrázek 5: Vzdálená komunikace přes Selenium Server nebo RemoteWebDriver



Zdroj: (Understanding the components :: Documentation for Selenium)

Nejprve budeme analyzovat provedení e-shopu. To, jakým způsoben je tvořen, jakou cestou je možné se dostat k jednotlivým produktům a jejich konkrétním atributům. Předpokládáme, že jednotlivé stránky s produkty budou obsahovat atributy typu název, cena, dostupnost a nějaké shrnutí informací o něm. Po důkladné analýze se můžeme přesunout k tvorbě kódu v Rstudiu, kde využijeme metody popsané ve třetí kapitole, abychom z webu (z jeho zdrojového kódu) dostali námi vybrané informace. Následně, po stažení těchto dat, provedeme kontrolu, zda jsou informace korektní a extrakce byla provedena bez chyb. Tato stažená data následně bude možné uložit do souboru v počítači pro jejich další využití.

Praktická část

V následujících kapitolách si představíme, jakým způsobem nalézt a extrahovat informace z námi vybraného e-shopu. Nejprve si vybraný web důkladně prostudujme, abychom měli představu, jak funguje. Nebudeme studovat celý jeho zdrojový kód, ale budou nás zajímat pouze ty části, které budeme chtít stahovat. Zhodnotíme, jakým způsobem jsou zobrazeny produkty na stránce. Pokud se produkty dané kategorie nachází na více stránkách, zjistíme, zda je možnost zobrazit všechny najednou nebo je potřeba postupně procházet každou stránku zvlášť. Podíváme se na konkrétní produkt, zda je umožněn výběr variant. Extrahujeme data. Vše následně uložíme. Vždy záleží na nás, kolik informací využijeme.

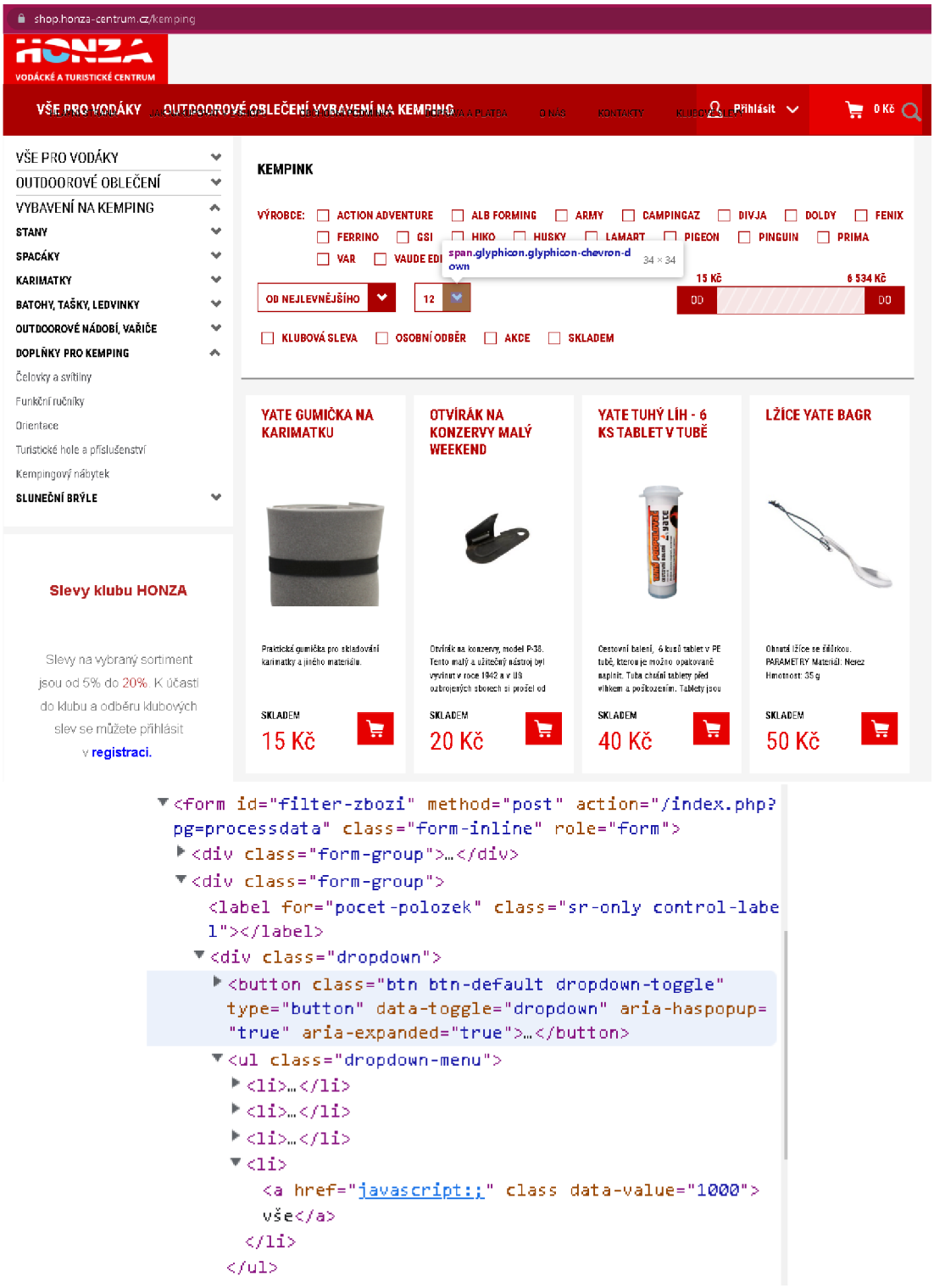
K identifikaci, jaké části zdrojového kódu odpovídají jakým prvkům v okně prohlížeče a naopak, můžeme použít inspektor prvků, který je implementován ve většině prohlížečů. Tak provedeme například kliknutím na vyhledávané místo v okně prohlížeče pravým tlačítkem myši a z kontextové nabídky vybereme Prozkoumat prvek. Prohlížeč zobrazí zdrojový kód části dokumentu HTML, která je zodpovědná za vybraný prvek (viz Obrázek 6). Proces můžeme také obrátit kliknutím na části zdrojového kódu a zvýraznit odpovídající části v interpretované verzi dokumentu.

4 Extrakce dat z vybraného e-shopu

Pro účely této diplomové práce byl vybrán e-shop Vodácké a kempingové potřeby - Eshop Praha 5 dostupný na URL adrese https://shop.honza-centrum.cz/. Jedná se o e-shop firmy Vodácké a turistické centrum Honza, s.r.o, se sídlem na adrese Sluneční náměstí 2583/11, 158 00 Praha 5, který nabízí vodácké, outdoorové a kempingové vybavení. Tento e-shop byl vybrán na základě určitých kritérií. Těmi jsou:

- Jedná se o dynamický e-shop.
- Obsahuje strukturovaná data.
- Je přehledný pro následnou kontrolu provedené extrakce.

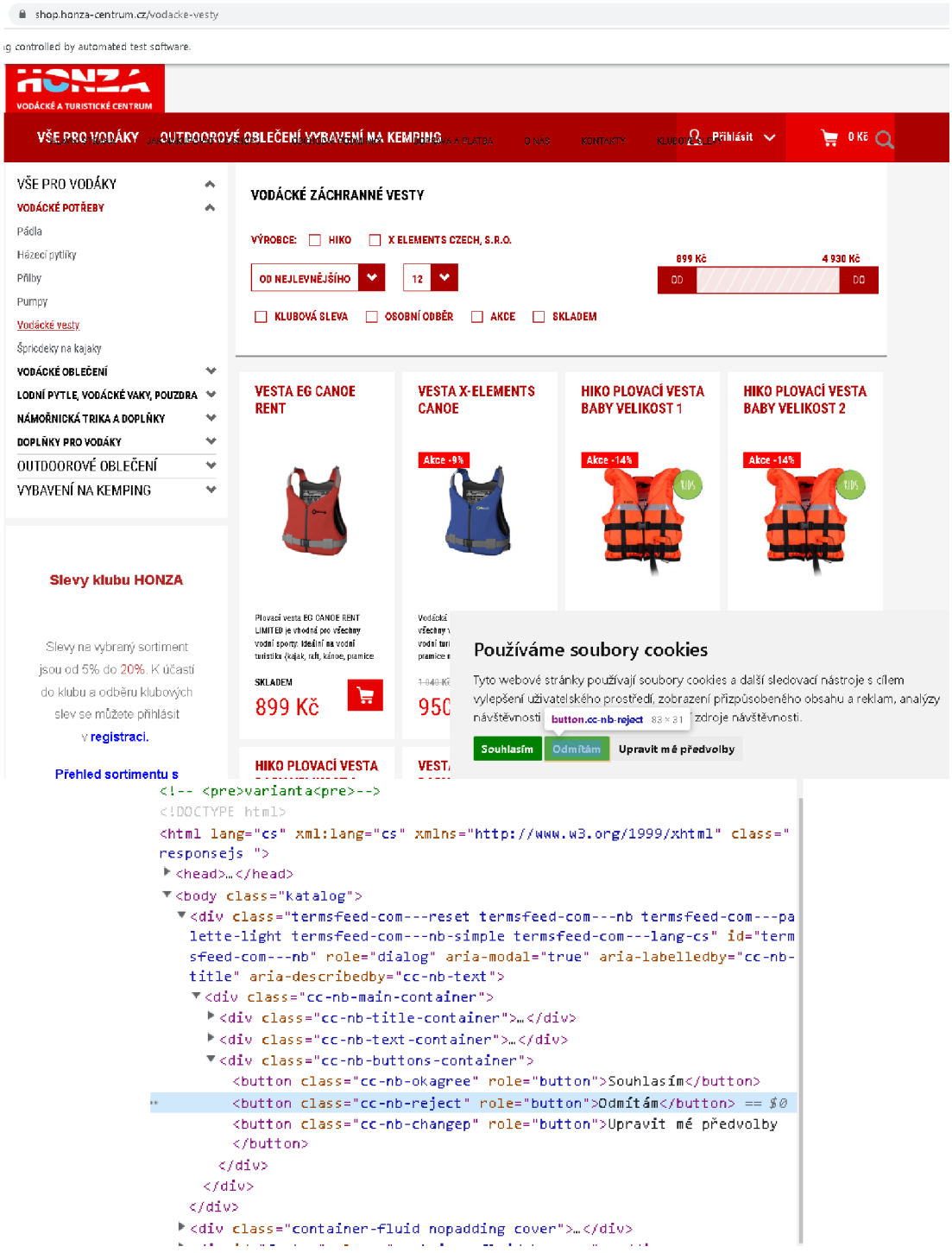
Obrázek 6: Vzhled stránky kategorie „VYBAVENÍ NA KEMPING“ se zdrojovým kódem7



Zdroj: (Vodácké a kempingové potřeby- Eshop Praha 5)

[7] U tohoto a následujících obrázků je vždy pod sebou část vzhledu webové stránky a část zdrojového kódu, která je zodpovědná za vybraný prvek na této stránce

Obrázek 7: Vzhled stránky podkategorie "VODÁCKÉ VESTY" se zdrojovým kódem



Zdroj: (Vodácké a kempingové potřeby- Eshop Praha 5)

4.1 Analýza stránek

Jako první při načtení stránek se objeví okno pro přijmutí cookies (viz Obrázek 7). Jelikož se zde nabízí možnost je odmítnout, tak ji zvolíme. Nejprve prozkoumáme, jaké kategorie web obsahuje. Z Obrázku 6 je patrné, že se zde vyskytují tři hlavní kategorie s názvy „VŠE PRO VODÁKY“, „OUTDOOROVÉ OBLEČENÍ“ a „VYBAVENÍ NA KEMPING“. Každá z těchto kategorií má ještě své podkategorie. Například v sekci „VYBAVENÍ NA KEMPING“ jsou to „STANY“, „SPACÁKY“, „KARIMATKY“, „BATOHY, TAŠKY, LEDVINKY“, „OUTDOOROVÉ NÁDOBÍ, VAŘIČE“, „DOPLŇKY PRO KEMPING“ a „SLUNEČNÍ BRÝLE“. I tyto podkategorie se mohou dál rozdělovat, jako například „DOPLŇKY PRO KEMPING“ se dále dělí na „Čelovky a svítilny“, „Funkční ručníky“, „Orientace“, „Turistické hole a příslušenství“ a „Kempingový nábytek“.

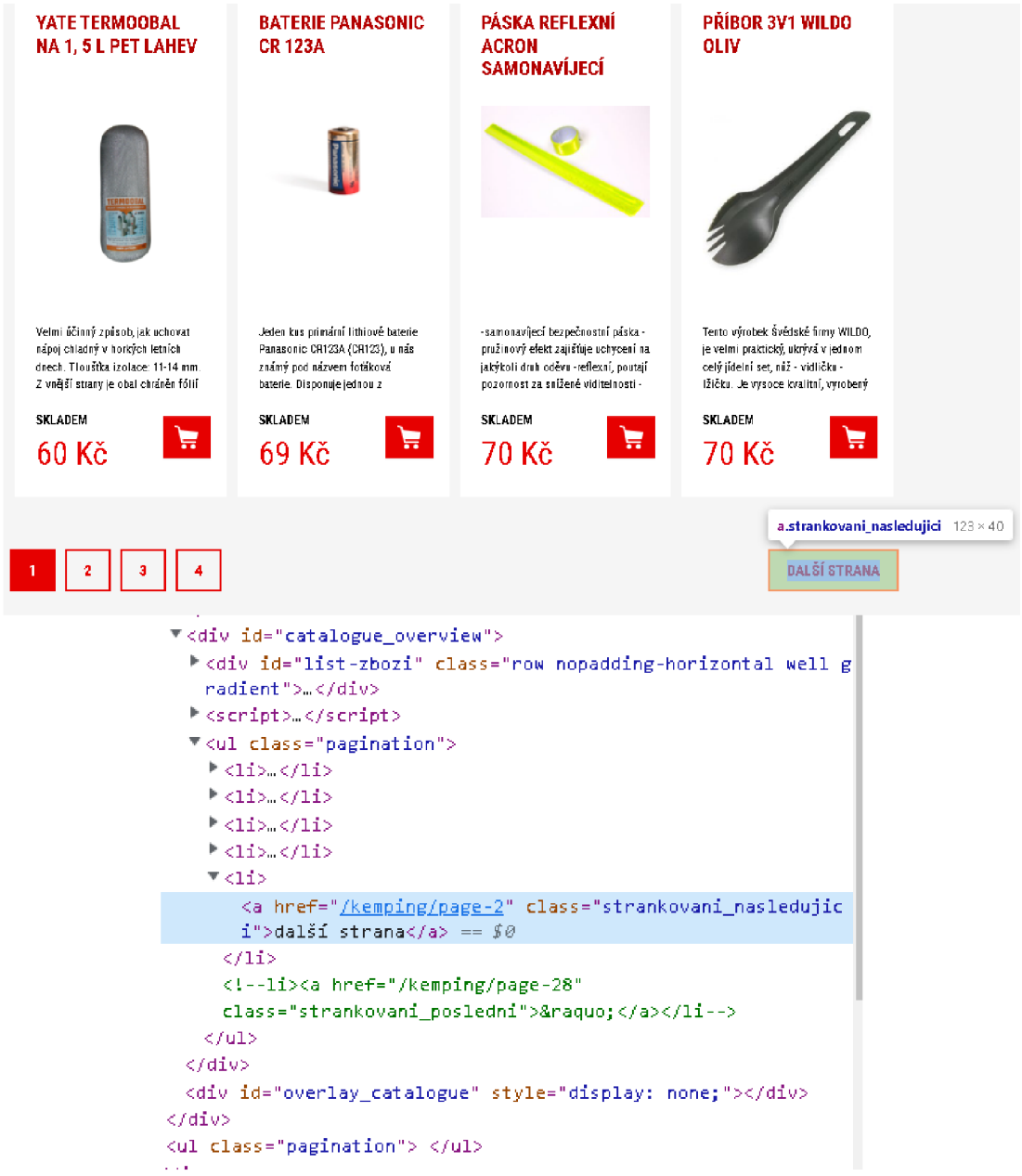
Jak vidíme na zmíněném obrázku, po výběru kategorie (v našem případě „VYBAVENÍ NA KEMPING“) jsou zobrazeny její produkty. Primárně je na stránkách zobrazeno prvních 12 produktů. Abychom nalezli všechny produkty dané kategorie, existuje obvykle několik cest.

Některé novější weby mohou další produkty načítat například pouze ve chvíli, kdy srolujeme (dostaneme se) na konec stránky a automaticky se načte další obsah. Na jiných webech můžeme nalézt také přímo možnost zobrazení všech produktů pomocí jednoho kliknutí.

Tento web nám nabízí dvě možnosti, jak zobrazit všechny produkty. Jednou je výběr počtu produktů, které chceme zobrazit na stránku. Pokud totiž rozklikneme výběr (na obrázku jej vidíme zvýrazněný uprostřed s číslovkou 12 a šipkou dolů), zobrazí se nám možnosti s názvy „12“, „24“, „36“ a „VŠE“. Nás bude primárně zajímat výběr „VŠE“, abychom načetli všechny produkty dané kategorie na stránku najednou.

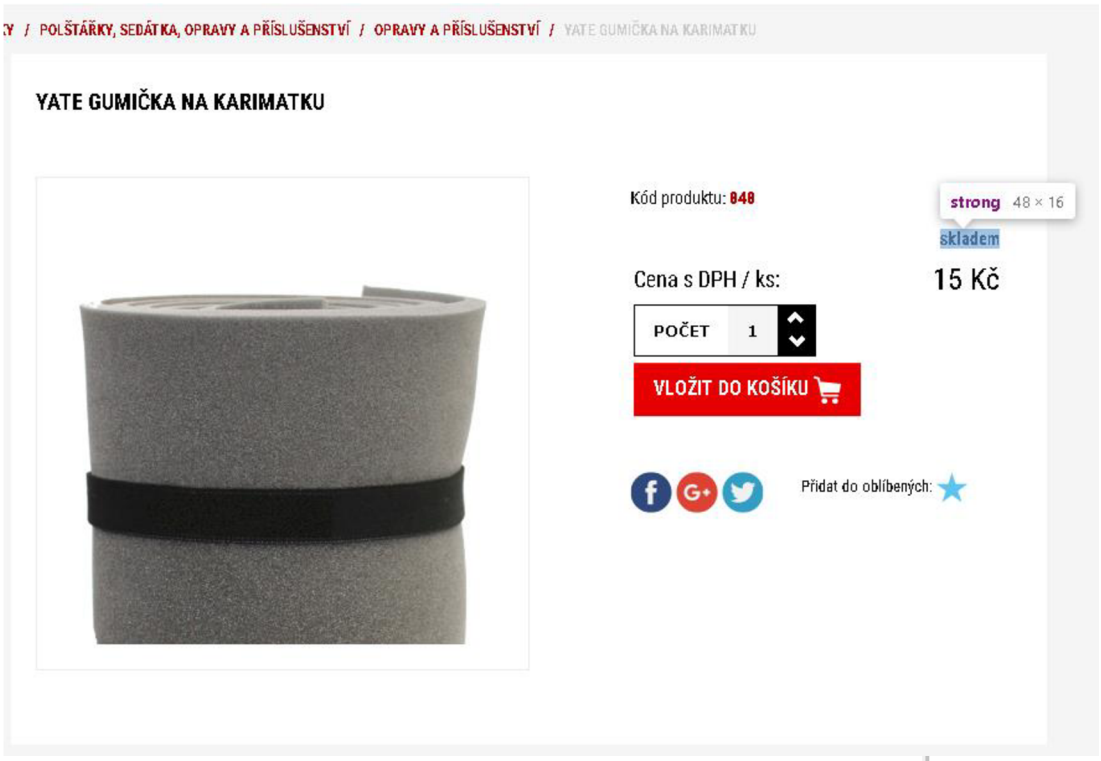
Druhá možnost se nachází na konci stránky, kde vidíme (na Obrázku 8) tlačítko „DALŠÍ STRANA“, díky němuž se dostaneme na další stránku s produkty. Takto bychom se dostali ke všem produktům, pokud bychom klikali na toto tlačítko, dokud by se na poslední stránce s produkty již nezobrazilo.

Obrázek 8: Vzhled konce stránky s produkty se zdrojovým kódem



Zdroj: (Vodácké a kempingové potřeby- Eshop Praha 5)

Obrázek 9: Vzhled stránky produktu „YATE GUMIČKA NA KARIMATKU“ a její zdrojový kód




```
<div id="foto" class="row">
  ::before
  <h1 itemprop="name">YATE gumička na karimatku</h1>
 <div class="main_foto col-sm-6">…</div>
<div class="col-sm-5 col-xs-12 pull-right">
 <ul class="list-group list-group-sm">…</ul>
 <div class="row">…</div>
<div class="product_detail_price_simple">
 <div id="cena" class itemprop="offers" itemscope
  itemtype="http://schema.org/Offer">
 <div class="widget-header widget-header-small form-inl
  ine-group no-radius">
   ::before
  <div class="widget-toolbar text-right pull-right">
   <div class="btn-group">
     <strong>skladem</strong> == $0
    </div>
   </div>
  <div class="clearfix">…</div>
  <div class="clearfix">…</div>
  <div class="widget-toolbar text-right bigger pull-ri
  ght" id="total-price">…</div>
  <div class="widget-toolbar bigger pull-left" id="tex
  t-price">…</div>
   ::after
 </div>
 <meta itemprop="currency" content="CZK">
</div>
</div>
```

Zdroj: (Vodácké a kempingové potřeby- Eshop Praha 5)

Rozhodneme se tedy pro výběr zobrazení všech produktů na jedné stránce. Klikneme na tlačítko s nápisem „12“ a z nabízených možností vybereme poslední s názvem „VŠE“. V toto chvíli máme zobrazeny všechny produkty na jedné stránce, aniž by se sama aktualizovala.

Jak vidíme (např. viz Obrázek 9[8]) u každého produktu se zobrazuje jeho název, stručný popis, dostupnost skladem a cena. Může se ale stát, že produkt bude například v akci a zobrazí se zde dvě ceny. Aktuální cena po slevě a zároveň přeškrtnutá před slevou. Prozkoumáme proto i jednotlivé stránky produktů.

První variantu toho, jak jsou uspořádány informace o produktech na jejich stránkách, vidíme na Obrázku 9. Název se nachází nad obrázkem produktu. Jeho dostupnost a cena jsou v pravé části hned pod sebou. Cena je uvedena v Kč.

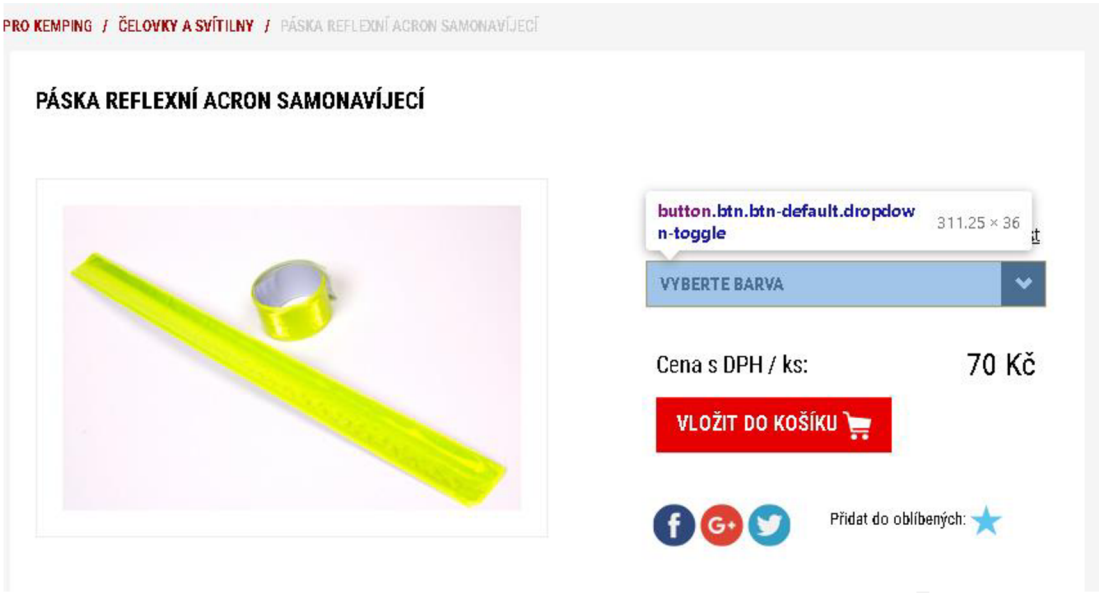
Vzhled druhé varianty produktu nalezneme na Obrázku 10. Zde vidíme stejné umístění názvu produktu i ceny. Můžeme tedy předpokládat, že toto rozložení se objeví i v dalších variantách. Cena je opět v pravé části, ale neobjevuje se zde dostupnost produktu. Místo toho je zde výběr z nabízených variant. Pokud na něj klikneme, zobrazí se seznam variant produktu. Může se stát, že nabídne pouze jednu variantu nebo několik. Konkrétně u tohoto produktu je reálné zvolit zpět výběr, žlutou nebo bílou barvu. Po provedení výběru (pokud je jedna možnost, vybereme tu, pokud jich je více, zvolíme až od druhé možnosti, abychom vynechali možnost vrátit se na začátek bez výběru) se zobrazí dostupnost vybrané varianty. Opět je tak provedeno, aniž by byla stránka automaticky aktualizována, pouze její obsah.

Na Obrázku 11 je zobrazen produkt, jenž opět potvrzuje pozici názvu i pozice výběru variant a ceny. Již se zde ale nenachází pouze jeden výběr, nýbrž dva. První volba zde nabízí jednu možnost, zatímco druhá jich nabízí několik. Pokud klikneme na první volbu a vybereme, dostupnost se ještě nezobrazí, dokud neprovedeme i druhou volbu. Na tomto obrázku se jedná také o produkt, jenž obsahuje dvě informace o ceně. První je šedivá a přeškrtnutá. Jedná se o cenu před slevou. Druhá je po slevě, zobrazena stejným stylem jako cena u předchozího Obrázku 9. Při extrakci nás bude zajímat primárně klubová cena, pokud produkt nebude

[8] V této kapitole se nachází obrázky, jejichž pravou část tvoří zdrojové kódy stránek, které budou následně využity v další kapitole

obsahovat, extrahujeme cenu v akci a pokud ani ta nebude uvedena, stáhneme cenu běžnou, protože za tyto částky bychom produkt reálně pořizovali.

Obrázek 10: Vzhled stránky produktu „PÁSKA REFLEXNÍ ACRON SAMONAVÍJECÍ“ a její zdrojový kód




```
<div class="panel panel-default row">
  ::before
  <div class="panel-body">
    ::before
    <form id="ax-variants" method="get" role="form"
    class="form-horizontal">
      <div class="form-group">
        ::before
        <div class="col-sm-12">
          <div class="dropdown">
            <button class="btn btn-default dropdown-toggl
            e" type="button" data-toggle="dropdown" aria-
            haspopup="true" aria-expanded="true"> == $0
              <span>vyberte barva</span>
              <span class="glyphicon glyphicon-chevron-do
              wn">
                ::before
              </span>
            </button>
            <ul class="dropdown-menu">…</ul>
            <select name="barva" id="barva" class="ajax_s
            elect form-control kosik-select input-sm">…
            </select>
          </div>
        </div>
        ::after
      </div>
      <script type="text/javascript">…</script>
    </form>
    ::after
  </div>
```

Zdroj: (Vodácké a kempingové potřeby- Eshop Praha 5)

Může se ale objevit i ta varianta, v níž se ve výběru nacházejí možnosti, které vybrat nelze (viz Obrázek 11). To se může stát, pokud jsme provedli jeden z výběrů, pro který není jedna z možností toho druhého dostupná. Na to je potřeba si dát pozor, zejména pokud provádíme výběry ihned po sobě. Například jsme provedli první volbu a v druhé byly všechny možnosti dostupné. Ve chvíli, kdy jsme se ale rozhodli provést znovu první výběr tentokrát pro jinou variantu, ta již nebyla dostupná, jelikož při druhém výběru už byla uložena jedna možnost, která pro tu z prvního výběru nebyla dosažitelná. Proto je zapotřebí vždy před opětovným prováděním obou výběrů také oba vrátit do původního stavu.

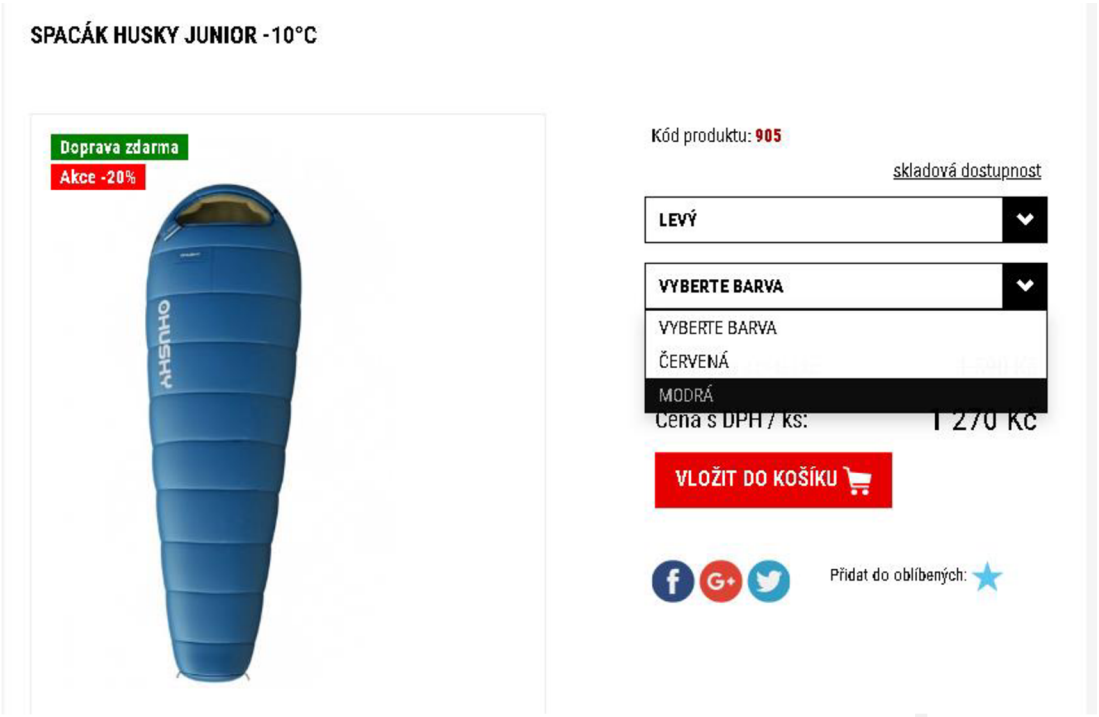
Je na nás, zda budeme stahovat informace o produktech z celého webu, nebo pouze z jeho částí. Pro náš návrh bude stahování provedeno pouze na určité části webu, a to na produktech spadajících pod kategorii „Vodácké vesty“, jelikož v době provádění extrakce se pouze zde nacházely produkty, které jsou v akci a zároveň mají klubovou cenu.[9]

Pro získání klubové ceny je zapotřebí se přihlásit (viz Obrázek 12) pomocí zadání e-mailu a hesla. To bude možné provést pouze pro registrované uživatele, proto se autorka práce na tomto webu zaregistrovala. Registrace není součástí navrženého kódu a měsíc po úspěšné obhajobě práce autorka změní své přihlašovací údaje, proto pro další kontroly bude potřeba se samostatně registrovat nebo požádat autorku o možný přístup. Po úspěšném přihlášení se u produktů s klubovou cenou spolu s prvními dvěma řádky zobrazí ještě třetí cena (viz Obrázek 13). Ta je ve stejném formátu, jako cena po slevě, ale pokud máme tu možnost nakupovat za klubové, nižší ceny, budou nás zajímat právě ty.

Takto vypadají všechny možné varianty produktů na tomto e-shopu. V následující podkapitole se pokusme vytvořit nástroj s použitím nasbíraných informací pro extrakci dat ze stránek.

[9] Nicméně sestrojený nástroj lze použít na všechny kategorie na e-shopu a bude plně funkční, nejen pro vybranou kategorii

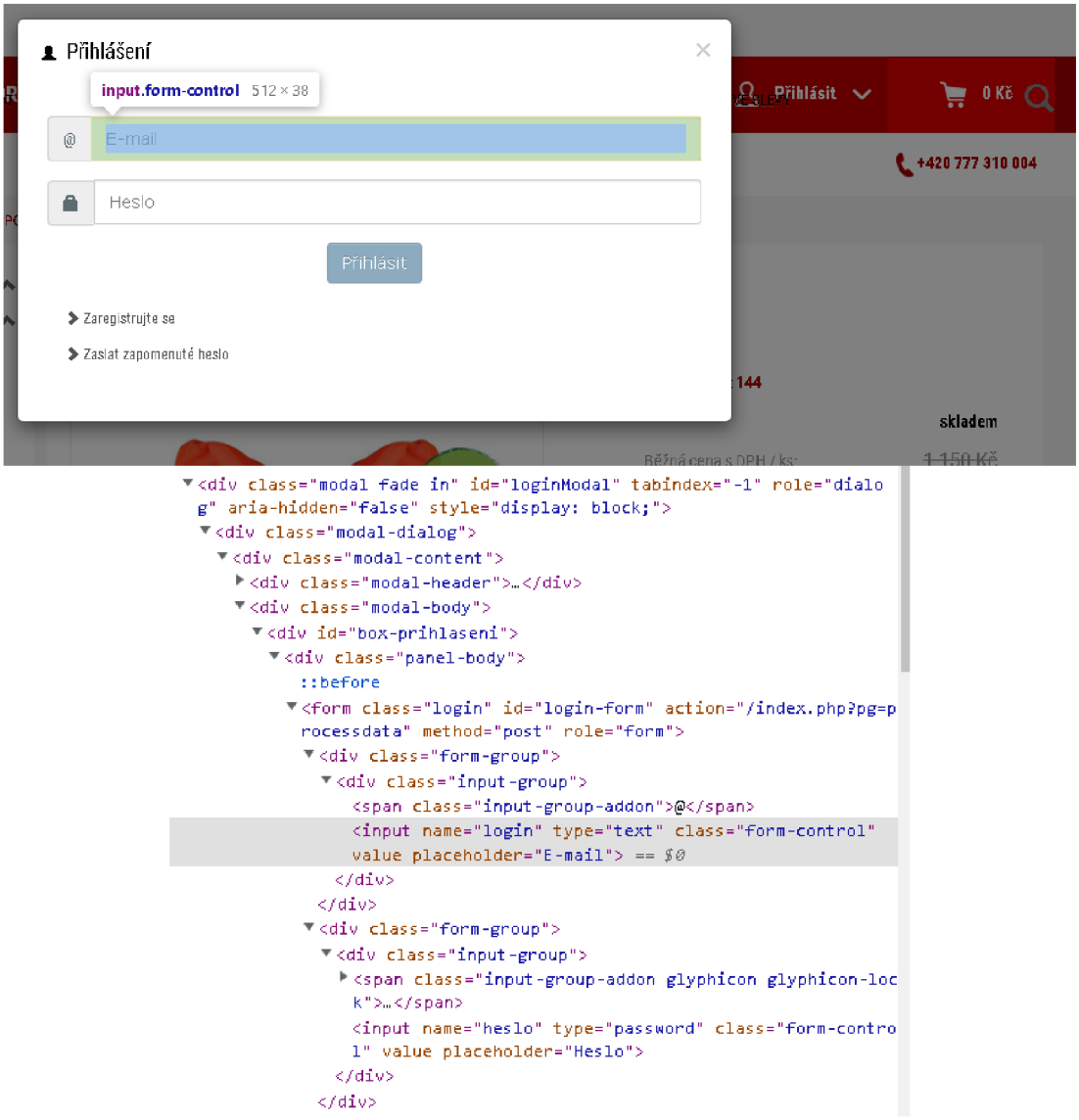
Obrázek 11: Vzhled stránky produktu „SPACÁK HUSKY JUNIOR -10°C“ a její zdrojový kód




```
<form id="ax-variants" method="get" role="form"
class="form-horizontal">
  <div class="form-group">…</div>
  <div class="form-group">
    ::before
    <div class="col-sm-12">
      <div class="dropdown open">
        <button class="btn btn-default dropdown-toggl
        e" type="button" data-toggle="dropdown" aria-
        haspopup="true" aria-expanded="true">…
        </button>
        <ul class="dropdown-menu">
          <li>…</li>
          <li>…</li>
          <li>
            <a href="javascript:;" class="disabled"
            data-value="Modr%C3%A1">Modrá</a> == $0
          </li>
        </ul>
        <select name="barva" id="barva" class="ajax_s
        elect form-control kosik-select input-sm">…
        </select>
      </div>
    </div>
    ::after
  </div>
```

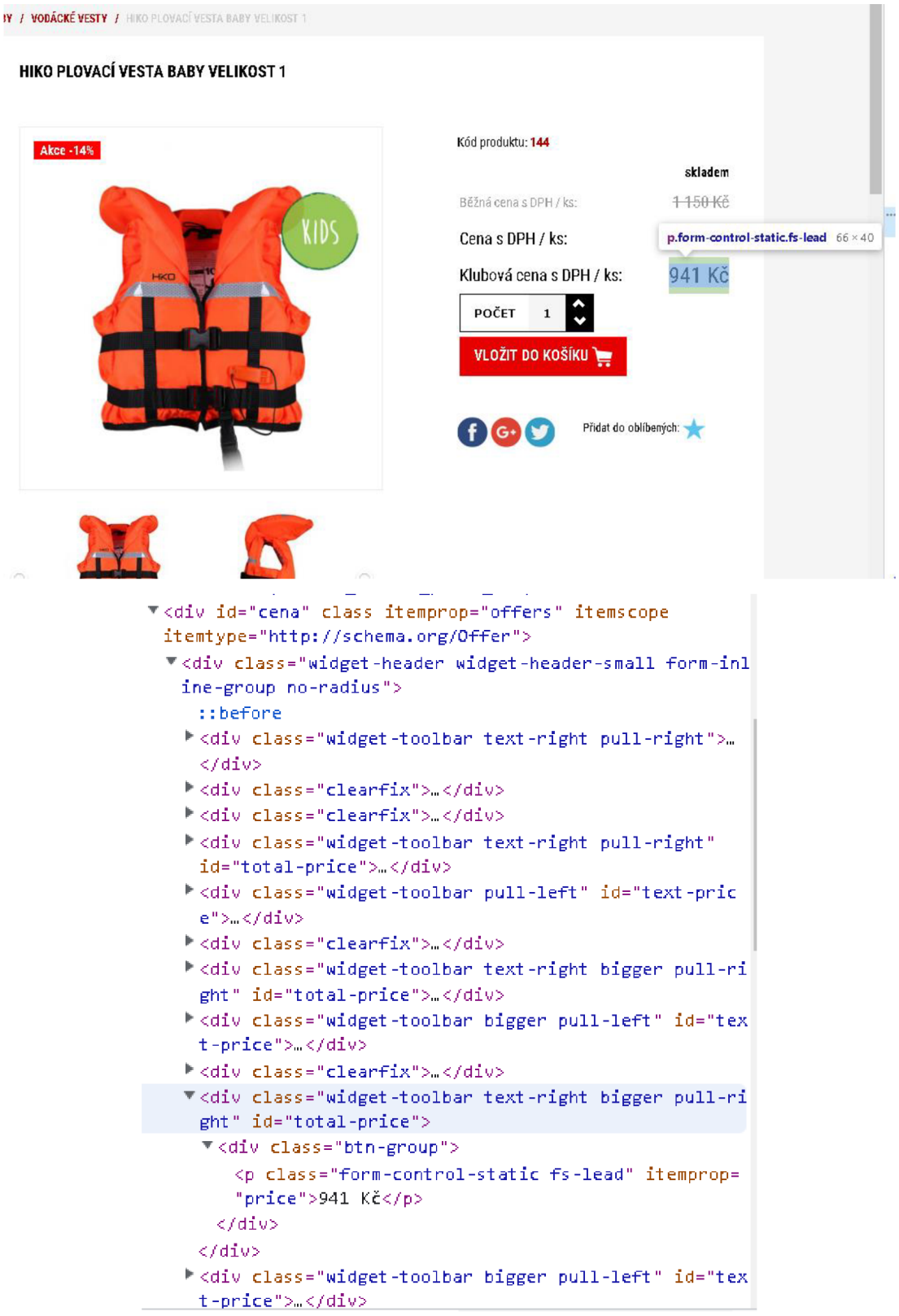
Zdroj: (Vodácké a kempingové potřeby- Eshop Praha 5)

Obrázek 12: Vzhled stránky s oknem pro přihlášení a její zdrojový kód



Zdroj: (Vodácké a kempingové potřeby- Eshop Praha 5)

Obrázek 13: Vzhled stránky produktu „HIKO PLOVACÍ VESTA BABY VELIKOST 1“ a její zdrojový kód

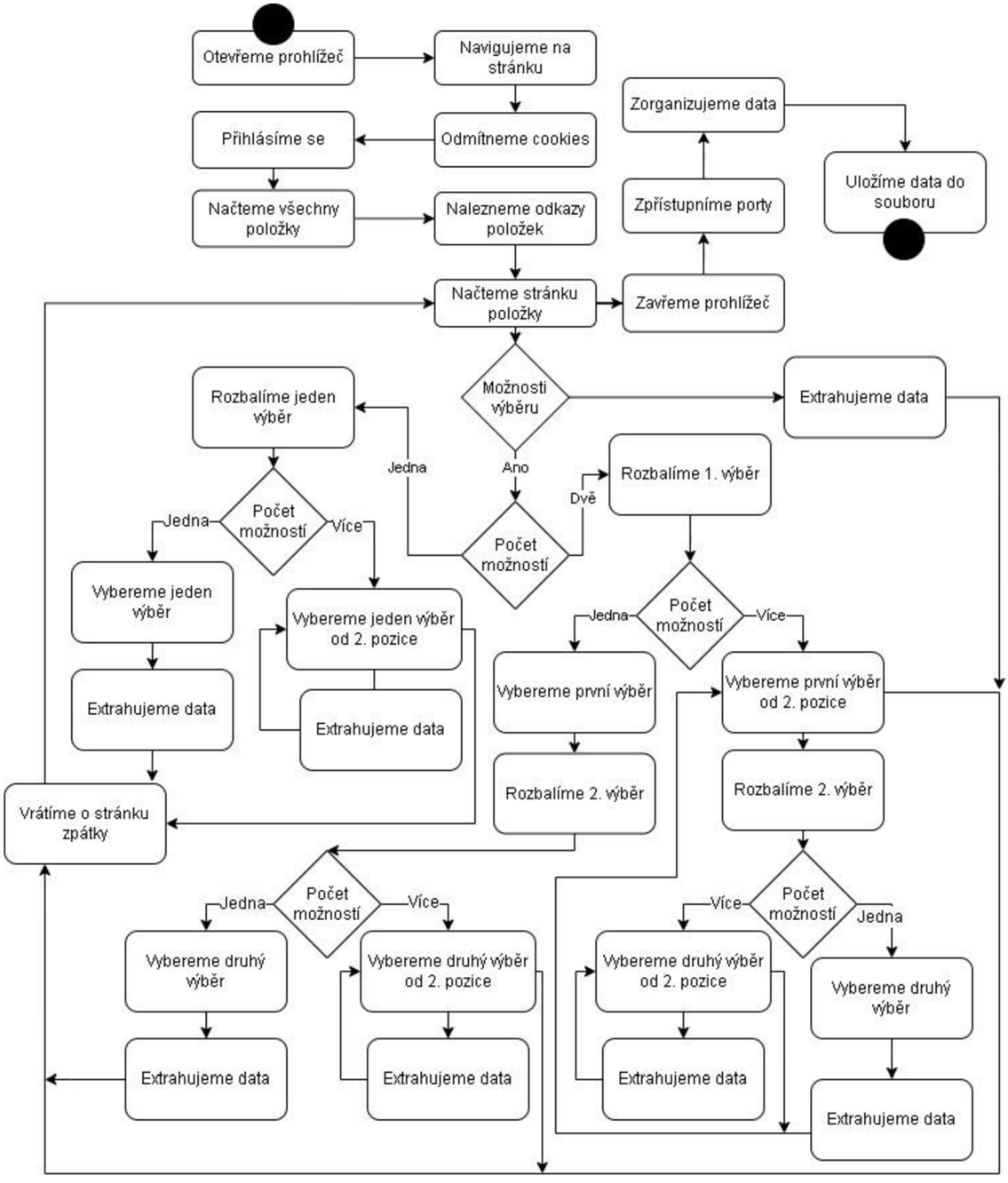


Zdroj: (Vodácké a kempingové potřeby- Eshop Praha 5)

4.2 Tvorba nástroje pro extrakci

Po provedení analýzy vybraného e-shopu využijeme nasbírané informace pro tvorbu postupu extrakce. Jak je vyobrazeno na Obrázku 14, nejprve otevřeme prohlížeč, v němž otevřeme vybranou stránku, kde odmítneme cookies. Následně se přihlásíme zadáním přihlašovacích údajů v okně pro přihlášení. Na stránce pak načteme všechny produkty a zjistíme jejich URL adresy. Jednotlivé produkty budeme postupně procházet od prvního po jejich celkový počet na stránce. Otevřeme odkaz každého produktu a tam zjistíme, zda je možné stáhnout informace nebo se nejprve proklikat výběrem. Pokud nenalezneme výběr, proběhne extrakce a vrátíme se zpět na všechny produkty. V případě jej nalezneme, je zapotřebí zjistit počet možných výběrů. Nalezneme-li jeden, opět buďto vybereme první možnost nebo postupně všechny v nabídce, kromě té první, která by výběr vrátila na začátek. V obou případech provedeme následně extrakci a vrátíme se zpět na všechny produkty. Pokud jsme ale nalezli dva výběry, je to složitější. V tuto chvíli může být v prvním výběru pouze jedna nebo více možností. Pokud je zde pouze jedna možnost, provedeme ji a v druhém výběru budeme postupovat opět dotazem na počet možností a provedením druhého výběru, jako kdyby zde byl celkově jeden. Jsou-li na stránce produktu ale dvě volby a první nabízí více možností, pak ji provedeme od druhé možnosti a druhou volbu opět buďto výběrem jedné nebo od druhé možnosti po počet nabízených. Jenže než se vrátíme k volbě další možnosti v prvním výběru, budeme je muset oba vrátit do původní podoby. Po provedení extrakce ze všech produktů budeme moci zavřít prohlížeč a zpřístupnit použité porty. Nasbíraná data bude možné zorganizovat například do datové tabulky a následně vytvořit soubor, do kterého tabulku uložíme.

Obrázek 14: Diagram postupu při extrakci



Zdroj: vlastní

V dalších podkapitolách nalezneme části kódu (příkazy) a některý jejich výstup po užití v konzoli aplikace Rstudio. V této aplikaci nejprve otevřeme nový R script, jež v první řadě uložíme do adresáře v počítači. Nyní se dostáváme k sepsání samotného kódu v konzoli.

4.2.1 Načtení balíčků

Níže vypsané příkazy nám umožní použít vypsané balíčky a následně jejich metody (funkce). Pokud bychom tyto příkazy neprovedli, v průběhu chodu navrženého kódu by nás konzole upozornila, že dané metody (spadající pod právě onen nepoužitý balíček) nezná.

```
library(rvest)

library(dplyr)

library(tidyverse)

library(robotstxt)

library(RSelenium)

library(XML)

library(netstat)
```

Může se stát, že konzole vrátí informaci, že nebylo možné některý z balíčků načíst. Proto je třeba před užitím těchto příkazů zkontrolovat, zda jsou tyto balíčky instalovány. Pokud tomu tak není, jednoduše je nainstalujeme také příkazem v konzoli. Například takto:

```
install.packages("RSelenium")
```

4.2.2 Kontrola povolení stahování webu

Do proměnné s názvem „honzaURL“ uložíme URL adresu, kterou jsme vybrali jako vstupní, z níž spustíme proces extrakce.

```
honzaURL<-("https://shop.honza-centrum.cz/vodacke-vesty")
```

Před prováděním extrakce je vždy potřeba zkontrolovat, zda je na dané stránce povolen přístup pro vstup „robota“, naší automatické extrakce.

```
robotstxt::paths_allowed(honzaURL)

shop.honza-centrum.cz

[1] TRUE
```

Výstup vrátil hodnotu TRUE, což znamená, že máme povoleno stahovat data z webové stránky `https://shop.honza-centrum.cz/vodacke-vesty`.

4.2.3 Přístup k dynamickému webu

Abychom mohli příkazem otevřít prohlížeč, potřebujeme znát aktuální verzi, která je používána v našem počítači. V našem případě používáme prohlížeč Google Chrome. Po jeho otevření klikneme na pole se třemi puntíky v pravém horním rohu. Zobrazí se možnosti, z nichž vybereme Nastavení (Settings) a následně O Chromu (About Chrome). Nyní se nám zobrazí informace o námi používané verzi: Version 105.0.5195.102 (Official Build) (64-bit). Pro naše účely nás zajímají první tři číslice dané verze. Tedy můžeme si pamatovat čislici 105.

V samotném příkazu pak můžeme využít balíček *binman* a jednu z jeho funkcí, list_versions(). Zajímá nás, jaké verze aplikace Google Chrome jsou dostupné dle platformy. Proto parametry, které uvedeme uvnitř funkce, jsou název aplikace, kterou chceme použít a druhý necháme defaultně nastavený na všechny platformy.

```
binman::list_versions("chromedriver")
```

```
$win32
```

```
[1] "105.0.5195.19" "105.0.5195.52" "106.0.5249.21"
```

Ve výstupu se objeví seznam použitelných verzí. Pro spuštění prohlížeče byla vybrána verze "105.0.5195.52", která je nejbližší nižší vůči aktuální verzi (105.0.5195.102) používaného prohlížeče Google Chrome.

Následujícím příkazem otevřeme okno prohlížeče za pomoci jeho názvu (chrome), verze (105.0.5195.102) a užití volného portu.

```
remdriver_objekt<-rsDriver(browser = 'chrome',
                           chromever = '105.0.5195.52',
                           verbose = FALSE,
                           port = free_port())
```

S prohlížečem lze pracovat ze strany serveru nebo klienta. My chceme zpřístupnit práci ze strany klienta, proto použijeme následující příkaz s uložením do proměnné „Okno“.

```
Okno<-remdriver_objekt$client
```

V průběhu práce s programem se může stát, že se okno po delší neaktivitě samo zavře, v tom případě jej otevřeme příkazem s použitím metody open().

```
Okno$open()
```

V tuto chvíli je možné použít uloženou URL adresu v proměnné „honzaURL“ pro načtení webu v prohlížeči s funkcí navigate().

```
Okno$navigate(honzaURL)
```

Po načtení stránky se do popředí dostane pole týkající se používání souborů cookies s možnostmi souhlasu, odmítnutí nebo úpravy předvolby (viz Obrázek 7). My vybereme Odmítnutí používání souborů cookies následujícím způsobem. Nejprve nalezneme tlačítko pro odmítnutí pomocí funkce findElement() pomocí třídy tohoto elementu. Ve zdrojovém kódu stránky se jedná o třídu s názvem „cc-nb-reject“ Poté na něj klikneme užitím metody clickElement() a tím odstraníme celé toto pole z námi zobrazené stránky.

```
Okno$findElement('class',value='cc-nb-reject')$clickElement()
```

Abychom se mohli přihlásit, je zapotřebí najít a kliknout na tlačítko pro přihlášení se k účtu. Opět tak provedeme užitím funkce findElement() a tentokrát třídy „account“ a užitím metody clickElement() na nalezené tlačítko.

```
Okno$findElement('class','account')$clickElement()
```

V okně prohlížeče se zobrazí nová kartička pro zadání přihlašovacích údajů. Je nutné zadat zaregistrovaný e-mail a k němu příslušné heslo. Pro užití se autorka na stránkách předem registrovala, protože předpokládá, že ten, kdo bude extrakci provádět, ji provede buď bez přihlášení k účtu nebo již bude registrován a nebude tak provádět pouze za účelem extrakce. Proto nalezneme kolonky pro vložení údajů použitím metody findElement() a jména (name) s hodnotou „login“ (viz Obrázek 12) pro pole zadání e-mailu a „heslo“ pro pole vepsání hesla. Funkcí sendKeysToElement() vyplníme nalezené kolonky textem v listu. Tyto přihlašovací údaje jsou osobní, proto budou změněny měsíc po úspěšné obhajobě práce. Pokud by tedy někdo chtěl později provádět tentýž postup, bude muset využít jiné (například platné po vlastní registraci). Po zadání údajů do pole hesla využijeme klíč s hodnotou „enter“. V podstatě tímto příkazem nebude potřeba hledat další tlačítko pro přihlášení. Je to obdobné, jako kdybychom použili tlačítko enter na klávesnici a přihlásili se.

```
Okno$findElement('name','login')$sendKeysToElement(list('chvosto
vapetra@seznam.cz'))

Okno$findElement('name','heslo')$sendKeysToElement(list('TheJerk
246', key = 'enter'))
```

4.2.4 Načtení všech položek

V tuto chvíli jsme přihlášeni. Můžeme tedy načíst všechny produkty dané kategorie na jednu stránku. Nalezneme tlačítko pro výběr počtu produktů zobrazených na stránce pomocí funkce findElement() a cesty (xpath) s hodnotou „//*[@id="filter-zbozi"]/div[2]/div/button“. V tomto případě nepoužijeme třídu („btn btn-default dropdown-toggle“, viz Obrázek 6), jelikož je tatáž třída použita i u sousedícího tlačítka pro styl zobrazení s textem „OD NEJLEVNĚJŠÍHO“. Cestu (xpath) je možné získat kliknutím na danou část ve zdrojovém kódu pravým tlačítkem myši, výběrem možnosti „Kopírovat“ a následně Kopírovat „XPath“. Nemusíme je tak složitě tvořit sami. Tlačítko proklikneme metodou clickElement().

```
Okno$findElement('xpath','//*[@id="filter-
zbozi"]/div[2]/div/button')$clickElement()
```

Zobrazí se možnosti výběru v podobě počtu produktů, které chceme zobrazit na stránce. Na tomto e-shopu se ukáží vždy čtyři možnosti, ale pokud bychom chtěli i tuto část postupovat automaticky (ne konkrétně vybráním čtvrté pozice), je to proveditelné.

Nalezneme pole všech možností funkcí findElement() a cesty (viz Obrázek6) „//*[@id="filter-zbozi"]/div[2]/div/ul“. Do proměnné názvem „pole“ užitím metody getElementText() uložíme text možností výběru.

```
pole<-Okno$findElement('xpath','//*[@id="filter-
     zbozi"]/div[2]/div/ul')$getElementText()
```

Pro zobrazení hodnot proměnné „pole“ pouze zadáme příkaz:

```
pole

[[1]]

[1] "12\n24\n36\nVŠE"
```

Jak vidíme výše, výstupem je text, jež bude potřeba upravit. Využijeme funkci gsub() pro nahrazení znaku „\n“ mezerou.

```
pole<-gsub("\n"," ",pole)
pole
[1] "12 24 36 VŠE"
```

V tuto chvíli je proměnná stále vedena jako jedna hodnota. Naším cílem je ji rozkouskovat na více hodnot, kde rozdělením bude námi zaměněná mezera. Tak provedeme funkcí strsplit(), ta rozdělí pole rozdělovačem námi udaným jako mezera.

```
pole<-strsplit(pole," ")
pole
[[1]]
[1] "12"  "24"  "36"  "VŠE"
```

Rozdělení na jednotlivé hodnoty bylo úspěšné. Zajímá nás, kolikátá je poslední hodnota. Tuto informaci můžeme zjistit funkcí length(), tedy délku proměnné „pole[[1]])“. V posledním výstupu z konzole vidíme, že pole je seznam s hodnotami na pozici „[[1]]“, proto ve funkci uvádíme přímo takto konkrétně.

```
pole<-length(pole[[1]])
pole
[1] 4
```

Dle výstupu jsme zjistili množství hodnot v poli pro výběr počtu zobrazených produktů a je umožněno tuto hodnotu použít pro výběr všech možností. Za pomoci metody str_c() spojíme dohromady části cesty „//*[@id="filter-zbozi"]/div[2]/div/ul/li[“, funkcí as.character() hodnotu „pole“ („4“) zapsanou jako znak a ne jako číslici a „]“. Tuto cestu užijeme pro nalezení možnosti výběru „VŠE“ s funkcí findElement() a následně ji vybereme metodou clickElement().

```
Okno$findElement('xpath',str_c('//*[@id="filter-
zbozi"]/div[2]/div/ul/li[',as.character(pole),']'))$clickElement
()
```

Po provedení výše uvedeného příkazu se na stránce načtou všechny produkty patřící do této kategorie. Abychom byly schopni je projít do posledního, je možné zjistit, kolik se jich na stránce nachází. Nejprve nalezneme zdrojový kód funkcí

getPageSource() (jedná se o seznam). Následně metodou unlist()zkonvertujeme seznam na vektor, jež užijeme ve funkci read_html() pro opětovné načtení zdrojového kódu. Je tak nutno provést, protože samotné provedení této funkce na původní zdrojový kód stránky by byl bez provedených výběrů. Výstup uložíme do proměnné „produkty“.

```
produkty <- read_html(unlist(Okno$getPageSource()))
```

V tuto chvíli použijeme metodu htmlParse(), abychom proměnnou „produkty“ (jež je typu xml_document) pomocí kódování „UTF-8“ přetvořili na typ HTMLInternalDocument a uložili do proměnné „produktyXML“.

```
produktyXML<-htmlParse(produkty,encoding = "UTF-8")
```

Užitím funkce capture.output() přetvoříme proměnnou „produktyXML“ na typ charakter string a metoda paste() tyto hodnoty zřetězí s oddělovačem „\n“. Vysledek uložíme do proměnné „produktyXMLtxt“

```
produktyXMLtxt<-paste(capture.output(produktyXML,
file=NULL),collapse = "\n")
```

Dostáváme se k proměnné „pocetNaStranku“, do níž uložíme hodnotu počtu produktů na stránce. Tu zjistíme užitím funkce str_count(), jež nalezne počet shod v proměnné „produktyXMLtxt“ se vzorem „item category col-xs-12 col-sm-6 col-lg-3“, což je název třídy použitý u jednotlivých produktů.

```
pocetNaStranku <- str_count(produktyXMLtxt,pattern = "item
category col-xs-12 col-sm-6 col-lg-3")

pocetNaStranku

[1] 11
```

Dle výstupu je zřejmé, že jsme provedli popsané operace korektně a hodnota proměnné „pocetNaStranku“ se shoduje s počtem produktů na stránce.

4.2.5 Vytvoření vlastních funkcí

Zde vytvoříme vlastní funkce, které následně využijeme v další podkapitole.

Metodu get_pole() je zapotřebí vytvořit, abychom na stránce konkrétního produktu zjistili, kolik je možností prvního výběru.

```
get_pole<-function(){
```

Funkcí findElement() a cestou „//*[@id="ax-variants"]/div/div/div/button“ (viz Obrázek 10) nalezneme tlačítko pro první výběr a metodou clickElement () jej rozbalíme.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/button')$clickElement()
```

Vždy, když použijeme funkci Sys.sleep() s hodnotou „0.5“, pozastavíme proces o půl vteřiny. To je zapotřebí, abychom nekladli přílišný tlak na server. Zpomalíme tak sice celou aktivitu, ale nebudeme tím riskovat, že by celý proces sám zastavil z důvodu přetížení.

```
Sys.sleep(0.5)
```

Metodou getElementText() nalezneme text v poli možností, jenž dostaneme funkcí findElement() a cestou „//*[@id="ax-variants"]/div[1]/div/div/ul“. Ten uložíme do proměnné „pole“.

```
pole<<-Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/ul')$getElementText()

pole

[1] "VYBERTE VELIKOST\n3XL\nL-XL\nS-M"
```

Výstupem u produktu „VESTA X-ELEMENTS CANOE“ (použijeme jej i jako příklad u dalších vytvořených funkcí) je jedna hodnota, kterou dále upravíme. Všimněme si, že k rozdělení jednotlivých polí je možné použít znak „\n“. Ten vyměníme za mezeru, ale ještě předtím, vyměníme mezery za nedělitelné, abychom ty původní nepoužili jako rozdělovač.

```
pole<<-gsub("\n"," ",gsub(" ","\u00A0",pole))

pole

[1] "VYBERTE VELIKOST 3XL L-XL S-M"
```

Proměnnou „pole“ je možné rozdělit mezerami na jednotlivá pole možností metodou strsplit().

```
pole<<-strsplit(pole," ")

pole

[[1]]
```

```
[1] "VYBERTE VELIKOST" "3XL"

[3] "L-XL"             "S-M"
```

Do proměnné tentokrát uložíme hodnotu délky seznamu na první pozici. Jinak řečeno, seznam má, jak ukazuje výstup, čtyři hodnoty.

```
pole<<-length(pole[[1]])}

pole

[1] 4
```

Funkci get_pole2() vytvoříme stejným způsobem jako get_pole() pouze s rozdílem, že zde použijeme cesty k tlačítku druhého výběru.

```
get_pole2<-function(){

Sys.sleep(0.5)

Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button')$clickElement()

pole2<<-Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/ul')$getElementText()

pole2

[[1]]

[1] "VYBERTE BARVA\nČERVENÁ\nMODRÁ\nZELENÁ"

pole2<<-gsub("\n"," ",gsub(" ","\u00A0",pole2))

pole2

[1] "VYBERTE BARVA ČERVENÁ MODRÁ ZELENÁ"

pole2<<-strsplit(pole2," ")

pole2

[[1]]

[1] "VYBERTE BARVA" "ČERVENÁ" "MODRÁ"

[4] "ZELENÁ"

pole2<<-length(pole2[[1]])}

pole2

[1] 4
```

Před extrakcí dat z webu zavedeme nejprve potřebné proměnné, do nichž budeme ukládat získané informace. Proměnná „cenaproduktu“ bude uchovávat ceny produktů, do „informace“ uložíme stručné informace o produktech, „dostupnost“ si zapamatuje, zda jsou produkty skladem, „odkazyProduktu“ bude obsahovat jejich URL odkazy, do „nazvyProduktu“ vložíme jejich názvy a nejen pro kontrolu si budeme také pamatovat v proměnných „prvnivyber“ a „druhyvyber“ hodnoty výběrů při extrakci.

```
cenaproduktu<-c()

informace<-c()

dostupnost<-c()

odkazyProduktu<-c()

nazvyProduktu<-c()

prvnivyber<-c()

druhyvyber<-c()
```

Zavedeme funkci get_atri(), jež bude extrahovat chtěné informace ze stránek v cyklu, který bude procházet jednotlivé produkty.

```
get_atri<-function(){

Sys.sleep(0.5)
```

Do proměnné „odkazyProduktu“ metodou append() přidáme jako další vektor hodnotu proměnné „odkaz“.

```
odkazyProduktu<<-append(odkazyProduktu, (odkaz))

odkaz

[1] " https://shop.honza-centrum.cz/vesta-xelements-canoe"
```

Za pomoci funkce str_c() spojíme dohromady části cesty „//*[@id="product-list"]/div[“, funkcí as.character() hodnotu „zac“ zapsanou jako znak a ne jako číslice a „]/div/div/div[1]/div[3]/ul/li/span“. Výstup uložíme do proměnné „path“.

```
path<<-str_c('//*[@id="product-
list"]/div[',as.character(zac),']/div/div/div[1]/div[3]/ul/li/sp
an')
```

Tu užijeme jako cestu k nalezení dat v proměnné „produktyXML“ pomocí metody xpathSApply() a v ní zadané funkce xmlValue() pro extrakci hodnot. Výstup uložíme do proměnné „informace1“. Příkaz append() jej přidá jako další vektor do proměnné „informace“.

```
informace<<-append(informace,informace1<-
xpathSApply(produktyXML, path=path,xmlValue))
```

```
informace1
```

```
[1] "Vodácká plovací vesta vhodná pro všechny vodní sporty.
Ideální na vodní turistiku (kajak, raft, kánoe, pramice nebo
paddleboard). Díky velmi vysoké odolnosti použitých materiálů
a snadné možnosti"
```

Metodou getElementText() nalezneme text na stránce, jež dostaneme funkcí findElement() a cestou „//*[@id="foto"]/h1“. Ten uložíme do proměnné „nazev“.

```
nazev<<-
Okno$findElement('xpath','//*[@id="foto"]/h1')$getElementText()
```

```
nazev
```

```
[[1]]
```

```
[1] "VESTA X-ELEMENTS CANOE"
```

Jedná se o seznam s hodnotami na pozici „[[1]]“, jež metodou append() přidáme jako další vektor do proměnné „nazvyProduktu“.

```
nazvyProduktu<<-append(nazvyProduktu,nazev[[1]])
```

Funkcí getElementText() nalezneme text na stránce, který dostaneme funkcí findElement() a cestou „//*[@id="cena"]/div/div[1]/div/strong“ (viz Obrázek 9). Ten uložíme do proměnné „dostupnost1“.

```
dostupnost1<<-
Okno$findElement('xpath','//*[@id="cena"]/div/div[1]/div/strong'
)$getElementText()
```

```
dostupnost1
```

```
[[1]]
```

```
[1] "skladem 4 ks"
```

Hodnoty se nachází na pozici „[[1]]“, jež příkazem append() přidáme jako další vektor do proměnné „dostupnost“.

```
dostupnost<<-append(dostupnost,dostupnost1[[1]])
```

Zavedeme funkci get_cenu(), jež bude extrahovat ceny ze stránek v cyklu, který bude procházet jednotlivé produkty.

```
get_cenu<-function(){
```

Jelikož se může stát, jak bylo zmíněno v analýze webu, že se na stránce produktu objeví několik typů cen, naším cílem je vybrat tu, za kterou bychom teoreticky produkt pořídili při nákupu. Proto provedeme následující operace.

Zjistíme zdrojový kód stránky funkcí getPageSource() a následně jej metodou unlist() zkonvertujeme na vektor, jenž užijeme ve funkci read_html() pro opětovné načtení zdrojového kódu. Je tak nutno provést, protože samotné provedení této funkce na původní zdrojový kód stránky by bylo bez provedených výběrů. Výstup uložíme do proměnné „stranka“.

```
stranka<-read_html(Okno$getPageSource() %>% unlist())
```

V tuto chvíli použijeme metodu htmlParse(), abychom proměnnou „stranka“ (která je typu xml_document) pomocí kódování „UTF-8“ přetvořili na typ HTMLInternalDocument a uložili do proměnné „strankaXML“.

```
strankaXML<-htmlParse(stranka,encoding = "UTF-8")
```

Užitím funkce capture.output() přetvoříme proměnnou „strankaXML“ na typ charakter string a metoda paste() tyto hodnoty zřetězí s oddělovačem „\n“. Výsledek uložíme do proměnné „strankaXMLtxt“.

```
strankaXMLtxt<-paste(capture.output(strankaXML,
file=NULL),collapse = "\n")
```

Dostáváme se k proměnné „ceny“, do níž uložíme hodnotu počtu produktů na stránce. Tu zjistíme užitím funkce str_count(), jež nalezne počet shod v proměnné „strankaXMLtxt“ se vzorem „total-price“, což je název id (viz Obrázek 13) použitý u jednotlivých produktů.

```
ceny<-str_count(strankaXMLtxt,pattern = "total-price")
ceny
```

```
[1] 2
```

Hodnota „ceny“ může nabývat hodnot od 1 do 3 (cena běžná, akční a klubová). Proto podle této hodnoty vybereme cenu příslušné pozice. Tedy příkazem if() se zeptáme, zda se hodnota „cena“ rovná 1. Pokud tato podmínka nebude splněna, else if() se dotáže na rovnost s další hodnotou (2). Pokud i tato informace bude nepravdivá, provedeme příkaz else a předpokládáme, že hodnota je rovna 3. U těchto příkazů získáme text funkcí getElementText() z části stránky, kterou nalezneme metodou findElement() a cestou „//*[@class="widget-header widget-header-small form-inline-group no-radius"]/div[X]“ (viz Obrázek 13), kde X nahradíme v prvním případě číslem 4, druhém 7 a třetím 10. Výstup zapíšeme do proměnné „cenaproduktu1“.

```
if(ceny==1){
cenaproduktu1<<-Okno$findElement('xpath','//*[@class="widget-header widget-header-small form-inline-group no-radius"]/div[4]')$getElementText()
}else if(ceny==2){
cenaproduktu1<<-Okno$findElement('xpath','//*[@class="widget-header widget-header-small form-inline-group no-radius"]/div[7]')$getElementText()
}else{
cenaproduktu1<<-Okno$findElement('xpath','//*[@class="widget-header widget-header-small form-inline-group no-radius"]/div[10]')$getElementText()}
cenaproduktu1
[[1]]
[1] "950 Kč"
```

Jelikož hodnota „ceny“ je 2, platí druhá podmínka a provede se příkaz s náhradou za číslo 7. Výstupem bude hodnota „950 Kč“. Při užití těchto tří funkcí tedy vždy dostaneme cenu v její číselné hodnotě a informaci, že se jedná o koruny. Abychom pak mohli s cenou ještě v budoucnu pracovat jako s číslicí, provádět matematické výpočty, je zapotřebí tyto hodnoty upravit před uložením do seznamu všech cen. Funkcí str_remove_all() odstraníme z hodnoty v seznamu „cenaproduktu1“, jež se nachází

na pozici „[[1]]“ hodnotu „ Kč“, tedy i s mezerou. Výsledek následně metodou as.numeric() zkonvertujeme na číslovku a funkcí append() ji jako další vektor přidáme do proměnné „cenaproduktu“.

```
cenaproduktu<<-
append(cenaproduktu,as.numeric(str_remove_all(cenaproduktu1[[1]]
,"[ Kč]"))) }
```

Poslední zavedenou funkci, jež bude extrahovat hodnoty dvou výběrů ve chvíli, kdy se objeví oba na stránce produktu.

```
get_vybery <- function(){
Sys.sleep(1)
```

Hodnotu prvního výběru dostaneme funkcí getElementText() z části stránky, kterou nalezneme metodou findElement() a cestou „//*[@id="ax-variants"]/div[1]/div/div/button“. Výstup uložíme do proměnné „prvnivyber1“.

```
prvnivyber1<-Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/button')$getElementText()
prvnivyber1
[[1]]
[1] "3XL"
```

Tuto hodnotu přidáme funkcí append() jako další vektor do proměnné „prvnivyber“.

```
prvnivyber<<-append(prvnivyber,prvnivyber1[1])
```

V druhém výběru dostaneme jeho hodnotu příkazem getElementText() z části stránky, kterou nalezneme metodou findElement() a cestou „//*[@id="ax-variants"]/div[2]/div/div/button“. Vše uložíme do proměnné „druhyvyber1“.

```
druhyvyber1<-Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button')$getElementText()
druhyvyber1
[[1]]
[1] "ČERVENÁ"
```

Výstup přidáme funkcí append() jako další vektor do proměnné „druhyvyber“.

```
druhyvyber<<-append(druhyvyber,druhyvyber1[1])}
```

4.2.6 Průchod jednotlivých produktů

Máme vše potřebné k tomu, abychom byli schopni procházet jednotlivé stránky produktů. Jelikož jsme si uchovali v proměnné „pocetNaStranku“ celkový počet načtených produktů, z nichž chceme extrahovat, můžeme tak provést pomocí příkazu for(). Jedná se o cyklus hodnot („zac“) od prvního po celkový počet načtených produktů postupně po jednom.

```
for (zac in seq(from=1,to= pocetNaStranku, by=1)) {
```

Nacházíme se na stránce vybrané kategorie (konkrétně je tato hodnota uložena v proměnné „honzaURL“). Je zapotřebí získat odkaz, jímž se přesuneme na stránku produktu. Cestu k jeho nalezení dostaneme funkcí str_c(), jež spojí části cesty „//*[@id="product-list"]/div[“, metodou as.character() hodnotu „zac“ zapsanou jako znak a ne jako číslici a „]/div/div/div[1]/div[1]/h4/a“.

```
cesta<-str_c('//*[@id="product-
list"]/div[',as.character(zac),']/div/div/div[1]/div[1]/h4/a')

cesta

[1]"//*[@id=\"product-list\"]/div[2]/div/div/div[1]/div[1]/h4/a"
```

Tu užijeme jako cestu k nalezení dat v proměnné „produktyXML“ pomocí metody xpathSApply() a v ní zadané funkce xmlGetAttr() a hledanou hodnotou „href“. Čímž získáme část domény, jež dále upravíme na celou adresu URL pomocí příkazu paste(), jímž spojíme část adresy, která chybí („https://shop.honza-centrum.cz“) a nalezený kus domény těsně za sebe bez mezery.

```
odkaz<-xpathSApply(produktyXML, path=cesta,xmlGetAttr,"href")%>%
paste("https://shop.honza-centrum.cz", ., sep = "")

odkaz

[1] "https://shop.honza-centrum.cz/vesta-xelements-canoe"
```

Přesměrujeme prohlížeč na nalezenou adresu produktu funkcí navigate() s hodnotou „odkaz“.

```
Okno$navigate(odkaz)
```

Zjistíme zdrojový kód stránky funkcí getPageSource() a následně jej metodou unlist() zkonvertujeme na vektor, jež užijeme ve funkci read_html() pro opětovné načtení zdrojového kódu. Je tak nutno provést, protože samotné provedení této funkce na původní zdrojový kód stránky by bylo bez provedených výběrů. Výstup uložíme do proměnné „stranka“.

```
stranka<-read_html(Okno$getPageSource() %>% unlist())
```

Nyní použijeme metodu htmlParse(), abychom proměnnou „stranka“ (jež je typu xml_document) pomocí kódování „UTF-8“ přetvořili na typ HTMLInternalDocument a uložili do proměnné „strankaXML“.

```
strankaXML<-htmlParse(stranka,encoding = "UTF-8")
```

Užitím funkce capture.output() přetvoříme proměnnou „strankaXML“ na typ charakter string a metoda paste() tyto hodnoty zřetězí s oddělovačem „\n“. Výsledek uložíme do proměnné „strankaXMLtxt“.

```
strankaXMLtxt<-paste(capture.output(strankaXML,
file=NULL),collapse = "\n")

Sys.sleep(0.5)
```

Dostáváme se k proměnné „klikani“, do níž uložíme hodnotu počtu sekcí[10], kde se nacházejí na stránce výběry. Tu zjistíme užitím funkce str_count(), jež nalezne počet shod v proměnné „strankaXMLtxt“ se vzorem „ax-variants“, což je název id použitý u jednotlivých produktů.

```
klikani<-str_count(strankaXMLtxt,pattern = "ax-variants")

klikani

[1] 1
```

Dotazem, zda je hodnota „klikani“ větší než 0, zjistíme, jestli se na stránce objevuje výběr či nikoli. Pokud je odpověď pravdivá, provedeme následující kroky. Pokud by byla lživá, provedeme kroky popsané na začátku stránky 67 pod příkazem else.

```
if(klikani>0){
```

[10] Sekce je v případě hodnoty 1 nalezena, v opačném je hodnota 0.

Zajímá nás, kolik je na stránce tlačítek pro výběr. Z proměnné „stranka“ nalezneme funkcí html_element() pomocí id[11] „ax-variants“ část zdrojového kódu odpovídající tomuto elementu. Hledáme celou část, kde se provádí výběry. Výstup převedeme na typ string příkazem toString.XMLNode() a hodnotu uložíme do proměnné „plocha“.

```
plocha<-stranka%>%html_element("#ax-
variants")%>%toString.XMLNode()
```

Proměnná „vybery“ slouží k uchování hodnoty počtu tlačítek výběrů, které dostaneme funkcí str_count() v proměnné „plocha“ se vzorem „form-group“, což je název třídy (viz Obrázek 11) použitý pro jednotlivé výběry.

```
vybery<-str_count(plocha,pattern = "form-group")
```

Provedeme dotaz, zda je hodnota „vybery“ rovna 1.

```
if(vybery==1){
```

Pokud je výstup pravdivý, užijeme následující příkazy. Funkce get_pole() rozbalí výběr a zjistí, kolik je v něm možností.

```
get_pole()
```

Následně se dotážeme, jestli je hodnota „pole“ shodná s číslem 1.

```
if(pole==1){
```

Je-li odpověď kladná, znamená to, že v poli výběru je pouze jedna možnost a pokračujeme následovně:

Pomocí cesty „//*[@id="ax-variants"]/div/div/div/ul/li/a“ a funkce findElement() nalezneme tlačítko výběru a metodou clickElement() na něj klikneme. Tím se vybere první nabízená možnost.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/ul/li/a')$clickElement()
```

Hodnotu výběru dostaneme funkcí getElementText() z části stránky, již nalezneme metodou findElement() a cestou „//*[@id="ax-variants"]/div[1]/div/div/button“. Výstup uložíme do proměnné „prvnivyber1“.

[11] znak # zde udává, že se jedná o id, v případě třídy by značení bylo tečkou

```
prvnivyber1<-Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/button')$getElementText()
```

Příkazem append() jej přidáme jako další vektor do proměnné „prvnivyber“.

```
prvnivyber<-append(prvnivyber,prvnivyber1)
```

Zároveň touto funkcí přidáme do proměnné „druhyvyber“ informaci, že se na stránce druhý výběr nenachází. Například text „NENÍ VÝBĚR“.

```
druhyvyber<-append(druhyvyber,"NENÍ VÝBĚR")
```

Metodami get_atri() a get_cenu() extrahujeme vybrané informace.

```
get_atri()
```

```
get_cenu()
```

Může nastat situace, kdy je hodnota „pole“ větší než 1. To znamená, že existuje více možností ve výběru, přičemž první nechceme vybrat, protože je neutrální. Vrátili by výběr na začáteční pozici, bez vybrání možnosti.

```
}else{
```

Proto jednotlivé hodnoty budeme postupně po jedné procházet od druhé pozice po celkový počet „pole“.

```
for(i in seq(from=2, to=pole, by=1)){
```

Ve chvíli, kdy se ve výběru nachází více možností, se může stát, že na některou z nich nebude možné kliknout. Zjistíme tedy, zda je daná možnost dostupná či nikoli.

Za pomoci metody str_c() spojíme dohromady části cesty „//*[@id="ax-variants"]/div/div/div/ul/li[“, funkcí as.character() hodnotu „i“ zapsanou jako znak a ne jako číslici a „]/a“. Tuto cestu užijeme pro nalezení možnosti výběru s funkcí findElement() a následně zjistíme třídu metodou getElementAttribute(). Výstup uložíme do proměnné „vyber1“.

```
vyber1<-Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div/div/div/ul/li[',as.character(i),']/a'))$getElemen
tAttribute("class")
```

Pokud se hodnota „vyber1“ shoduje s textem „disabled“ vracíme se do cyklu a budeme pokračovat v procházení dalších možností výběru. Častěji ale nastane situace, kdy se hodnoty neshodují a můžeme pokračovat následujícími příkazy.

```
if(vyber1!="disabled"){
```

Stejně tak, jako jsme o dva kroky zpátky nalezli element možnosti na něj zde místo funkce getElementAttribute() použijeme metodu clickElement() a klikneme na něj. Tím jsme ho vybrali.

```
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div/div/div/ul/li[',as.character(i),']/a'))$clickElem
ent()
```

Hodnotu výběru dostaneme funkcí getElementText() z části stránky, již nalezneme metodou findElement() a cestou „//*[@id="ax-variants"]/div[1]/div/div/button“. Výstup uložíme do proměnné „prvnivyber1“.

```
prvnivyber1<-Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/button')$getElementText()
```

Hodnotu „prvnivyber1“ přidáme funkcí append() jako další vektor do proměnné „prvnivyber“.

```
prvnivyber<-append(prvnivyber,prvnivyber1)
```

Zároveň touto funkcí přidáme do proměnné „druhyvyber“ informaci, že se na stránce druhý výběr nenachází. Například text „NENÍ VÝBĚR“.

```
druhyvyber<-append(druhyvyber,"NENÍ VÝBĚR")
```

Příkazy get_atri() a get_cenu() extrahujeme vybrané informace.

```
get_atri()
```

```
get_cenu()
```

Jelikož se nyní chceme vrátit do cyklu a pokračovat ve výběru, je zapotřebí rozbalit pole možností tohoto výběru.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/button')$clickElement() } } }
```

Tímto způsobem bychom měli mít prošlé varianty, kde je pouze jeden výběr. Dostáváme se k produktům, kde se nachází výběry dva.

```
}else if(vybery==2){
```

Funkce get_pole() rozbalí první výběr a zjistí, kolik je v něm možností.

```
get_pole()
```

Následně se dotážeme, jestli je hodnota „pole“ shodná s číslem 1.

```
if(pole==1){
```

Je-li odpověď kladná, znamená to, že v poli prvního výběru je pouze jedna možnost a pokračujeme následovně:

Pomocí cesty „//*[@id="ax-variants"]/div[1]/div/div/ul/li/a“ a funkce findElement() nalezneme první možnost výběru a metodou clickElement() na ni klikneme a vybereme.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/ul/li/a')$clickElement()
```

Funkce get_pole2() rozbalí druhý výběr a zjistí, kolik je v něm možností.

```
get_pole2()

Sys.sleep(0.5)
```

Zeptáme se, jestli je hodnota „pole2“ shodná s číslem 1.

```
if(pole2==1){
```

Je-li odpověď kladná, znamená to, že v poli druhého výběru je pouze jedna možnost.

S pomocí cesty „//*[@id="ax-variants"]/div[2]/div/div/ul/li/a“ a funkce findElement() nalezneme první možnost druhého výběru a metodou clickElement() na ni klikneme a vybereme.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/ul/li/a')$clickElement()
```

Na stránce se zobrazila informace o dostupnosti. V tuto chvíli pomocí příkazů get_atri(), get_cenu() a get_vybery() extrahujeme vybrané informace. Můžeme se vrátit do prvního cyklu a přejít na další produkt.

```
get_atri()

get_cenu()

get_vybery()
```

Je-li shoda hodnoty „pole2“ a čísla 1 záporná (nerovnají se), znamená to, že v poli druhého výběru je více možností. Ty postupně po jedné projdeme od druhé

až po celkový počet „pole2“. Až cyklus skončí, vrátíme se do prvního cyklu a přesuneme se na další produkt.

```
}else{
for(iiii in seq(from=2, to=pole2, by=1)){
```

Ve chvíli, kdy se ve výběru nachází více možností, se může stát, že na některou z nich nebude možné kliknout. Zjistíme tedy, zda je daná možnost dostupná či nikoli.

Za pomoci metody str_c() spojíme dohromady části cesty „//*[@id="ax-variants"]/div[2]/div/div/ul/li[“, funkcí as.character() hodnotu „iiii“ zapsanou jako znak a ne jako číslici a „]/a“. Tuto cestu užijeme pro nalezení možnosti výběru s funkcí findElement() a následně zjistíme třídu metodou getElementAttribute(). Výstup uložíme do proměnné „vyber2“.

```
vyber2<-Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[2]/div/div/ul/li[',as.character(iiii),']/a'))$get
ElementAttribute("class")
```

Pokud se hodnota „vyber2“ shoduje s textem „disabled“ vracíme se do cyklu a budeme pokračovat v procházení dalších možností výběru. Častěji ale nastane situace, kdy se hodnoty neshodují a můžeme pokračovat následujícími příkazy.

```
if(vyber2!="disabled"){
```

Na výše nalezený element možnosti zde místo funkce getElementAttribute() použijeme metodu clickElement() a klikneme. Tím ho vybereme.

```
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[2]/div/div/ul/li[',as.character(iiii),']/a'))$cli
ckElement()
```

Na stránce se zobrazila informace o dostupnosti. V tuto chvíli pomocí příkazů get_atri(), get_cenu() a get_vybery() extrahujeme vybrané informace.

```
get_atri()
get_cenu()
get_vybery()
```

Jelikož se nyní chceme vrátit do cyklu a pokračovat v druhém výběru, je zapotřebí rozbalit pole možností tohoto výběru.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button/')$clickElement() } } }
```

V případě situace, kdy hodnota „pole“ je vyšší než 1, je v poli prvního výběru více možností. Ty postupně po jedné projdeme od druhé po celkový počet „pole“. Až cyklus skončí, vrátíme se do prvního cyklu a přesuneme se na další produkt.

```
}else{
for(iii in seq(from=2, to=pole, by=1)){
```

Jelikož se ve výběru nachází více možností, může se stát, že na některou z nich nebude možné kliknout. Zjistíme, zda je daná možnost dostupná či nikoli.

Za pomoci metody str_c() spojíme dohromady části cesty „//*[@id="ax-variants"]/div[1]/div/div/ul/li[“, funkcí as.character() hodnotu „iii“ zapsanou jako znak a ne jako číslici a „]/a“. Tuto cestu užijeme pro nalezení možnosti výběru s funkcí findElement() a následně zjistíme třídu metodou getElementAttribute(). Výstup uložíme do proměnné „vyber1“.

```
vyber1<-Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[1]/div/div/ul/li[',as.character(iii),']/a'))$getE
lementAttribute("class")
```

Pokud se hodnota „vyber1“ shoduje s textem „disabled“ vracíme se do cyklu a budeme pokračovat v procházení dalších možností prvního výběru. Častěji ale nastane situace, kdy se hodnoty neshodují a můžeme pokračovat následujícími příkazy.

```
if(vyber1!="disabled"){
```

Pro element možnosti místo funkce getElementAttribute() použijeme metodu clickElement(), abychom na něj tentokrát klikli a vybrali ho.

```
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[1]/div/div/ul/li[',as.character(iii),']/a'))$clic
kElement()
```

Funkce get_pole2() rozbalí druhý výběr a zjistí, kolik je v něm možností.

```
get_pole2()
```

Zeptáme se, zda je hodnota „pole2“ shodná s číslem 1.

```
if(pole2==1){
```

Je-li výstup kladný, znamená to, že v poli druhého výběru je pouze jedna možnost. S pomocí cesty „//*[@id="ax-variants"]/div[2]/div/div/ul/li/a“ a funkce findElement() nalezneme první možnost druhého výběru a metodou clickElement() na ni klikneme a vybereme.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/ul/li/a')$clickElement()
```

Na stránce se zobrazila informace o dostupnosti. V tuto chvíli pomocí příkazů get_atri(), get_cenu() a get_vybery() extrahujeme vybrané informace. Můžeme se vrátit do cyklu a přejít na další možnost v prvním výběru.

```
get_atri()

get_cenu()

get_vybery()
```

Pokud se hodnota „pole2“ a čísla 1 nerovnají, znamená to, že v poli druhého výběru je více možností. Ty postupně po jedné projdeme od druhé až po celkový počet „pole2“. Až cyklus skončí, vrátíme se do cyklu prvního výběru a budeme pokračovat v možnostech.

```
}else{

for(iiii in seq(from=2, to=pole2, by=1)){
```

Ve chvíli, kdy se ve druhém výběru nachází více možností, se může stát, že na některou z nich nebude možné kliknout. Zjistíme tedy, zda je daná možnost dostupná či nikoli.

Za pomoci metody str_c() spojíme dohromady části cesty „//*[@id="ax-variants"]/div[2]/div/div/ul/li[“, funkcí as.character() hodnotu „iiii“ zapsanou jako znak a ne jako číslici a „]/a“. Tuto cestu (viz Obrázek 11) užijeme pro nalezení možnosti výběru s funkcí findElement() a následně zjistíme třídu metodou getElementAttribute(). Výstup uložíme do proměnné „vyber2“.

```
vyber2<-Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[2]/div/div/ul/li[',as.character(iiii),']/a'))$get
ElementAttribute("class")
```

Pokud se hodnota „vyber2“ shoduje s textem „disabled“ vracíme se do cyklu a budeme pokračovat v procházení dalších možností výběru. Častěji ale nastane situace, kdy se hodnoty neshodují a můžeme pokračovat následujícími příkazy.

```
if(vyber2!="disabled"){
```

Stejně tak, jako jsme o dva kroky zpátky nalezli element možnosti na něj nyní místo funkce getElementAttribute() použijeme metodu clickElement() a klikneme. Tím jsme ho vybrali.

```
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[2]/div/div/ul/li[',as.character(iiii),']/a'))$cli
ckElement()
```

Příkazy get_atri(), get_cenu() a get_vybery() extrahujeme vybrané informace.

```
get_atri()

get_cenu()

get_vybery()
```

Jelikož se nyní chceme vrátit do cyklu druhého výběru a pokračovat v možnostech, je zapotřebí toto pole rozbalit.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button')$clickElement() } } } }
```

Pokud byly provedeny dva výběry, je zapotřebí před návratem do cyklu prvního výběru oba vrátit na původní pozici následujícími čtyřmi příkazy. Provedeme rozbalení prvního výběru a v něm zvolení první možnosti.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/button')$clickElement()

Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/ul/li[1]/a')$clickElement()
```

Následně uskutečníme rozbalení druhého výběru a v něm zvolíme taktéž první možnosti.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button')$clickElement()

Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/ul/li[1]/a')$clickElement()
```

Zároveň máme v paměti, že do cyklu jsme vstupovali již s rozbaleným prvním výběrem, proto je zapotřebí tak opětovně provést, ještě před návratem pomocí následujícího příkazu.

```
Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/button')$clickElement() } } }
```

V posledních krocích je představena situace, kdy se na stránce produktu nenachází žádné tlačítko výběru.

```
}else{
```

Proto příkazem append() přidáme do proměnných „prvnivyber“ a „druhyvyber“ informaci, že se na stránce výběry nenachází. Například text „NENÍ VÝBĚR“.

```
prvnivyber<-append(prvnivyber,"NENÍ VÝBĚR")
druhyvyber<-append(druhyvyber,"NENÍ VÝBĚR")
```

Funkcemi get_atri() a get_cenu() extrahujeme vybrané informace.

```
get_atri()
get_cenu() } }
```

4.2.7 Ukončení přístupu k dynamickému webu

Právě jsme provedli extrakci dat ze všech produktů a metodou quit() zavřeme okno prohlížeče.

```
Okno$quit()
```

Příkazem system() a hodnotou „taskkill /im java.exe /f“ ukončíme proces „java.exe“ a znovu zpřístupníme dostupné porty (0 používaných).

```
system("taskkill /im java.exe /f")
SUCCESS: The process "java.exe" with PID 17340 has been
terminated.
[1] 0
```

4.2.8 Uložení extrahovaných dat

Nyní máme extrahované všechny žádané informace a chceme je uložit nejprve do zavedené proměnné „produktydata“, kterou využijeme jako datovou tabulku.

```
produktydata<-list()
```

Metodou rbind() vložíme hodnoty proměnných „nazvyProduktu“, „cenaproduktu“, „prvnivyber“, „druhyvyber“, „dostupnost“, „informace“ a „odkazyProduktu“ jako další na seznam datové tabulky „produktydata“.

```
produktydata<-
rbind(produktydata,data.frame(nazvyProduktu,cenaproduktu,prvnivy
ber,druhyvyber,dostupnost,informace,odkazyProduktu,stringsAsFact
ors=FALSE))
```

Pro větší přehlednost můžeme názvy sloupců přejmenovat funkcí colnames() a vektorem požadovaných názvů.

```
colnames(produktydata)<-c("Název produktu","Cena v Kč","První
výběr", "Druhý výběr","Dostupnost","Info o produktu","Odkaz na
stránku produktu")
```

Jelikož jsme si před psaním kódu soubor uložili, můžeme se metodou getwd() podívat na jeho umístění v adresáři počítače.

```
getwd()
```

Data v tabulce „produktydata“ budeme chtít uložit do souboru s koncovkou „.csv“, abychom se souborem mohli následně pracovat nejen v prostředí R, ale například taky v aplikaci Excelv nebo jiných tabulkových softwarech. Vytvoření a zápis do souboru provedeme příkazem write.csv(), kam uložíme hodnoty proměnné „produktydata“ a soubor nazveme například taktéž „produktydata.csv“

```
write.csv(produktydata, "produktydata.csv")
```

4.3 Kontrola správnosti provedení extrakce

Na první pohled nám může připadat podezřelé, že počet produktů na webu ve vybrané kategorii je jedenáct (jak bylo zmíněno i v kódu jako hodnota proměnné „pocetNaStranku“) a ve výsledné tabulce (viz Tabulka 1) se jich objevilo třicet devět. To má jednoduché vysvětlení. Na stránkách e-shopu jsou v kategoriích produkty zastoupeny pouze jednou. Pokud je nějaký produkt nabízen ve více variantách (například v modré a žluté barvě nebo velikosti L a M), my už tuto informaci chápeme a ukládáme jako dva produkty. Ty se totiž mohou lišit například cenou nebo dostupností.

Zkontrolovat úspěšnost stahování, všechny data v tabulce, zda souhlasí s informacemi na webu u většiny extrakcí nebude plně reálné. Záleží na množství stažených dat. To je další důvod, proč byla vybrána kategorie „VODÁCKÉ VESTY“. Protože na menším množství dat je reálné v poměrně krátké době překontrolovat opravdu všechny informace. Začneme třemi produkty, které jsou bez výběru, tedy „HIKO PLOVACÍ VESTA BABY VELIKOST 1, 2 a 3“. Na Obrázku 13 máme možnost přímé kontroly ceny a dostupnosti a vidíme, že tyto informace souhlasí. Cena byla stažena ze správné klubové pozice. Nabízí se například možnost uchovávat všechny tři varianty ceny. Záleží na účelu sběru dat. Pro ukázku sestrojení nástroje pro extrakci nás zajímal především způsob, jak tyto data získat. Proto jsme si zadali, že nám postačí pouze jedna informace, a to cena při aktuálním pořízení. Dalších sedm produktů postupně proklikáme a překontrolujeme na webu e-shopu, že opravdu pro konkrétní volby byla tato data správně stažena. Můžeme konstatovat, že extrakce dat z vybrané kategorie našeho e-shopu byla bezchybně provedena.

Při pohledu na pátý sloupeček s názvem „Dostupnost“ můžeme konstatovat, že ne vždy je uveden přesný počet kusů skladem. Proto je zapotřebí vzít v úvahu, jak bychom se staženými informacemi chtěli dále nakládat. Pokud bychom například prováděli extrakci z webu dodavatele a tyto produkty sami neměli skladem,

takto stažená data bychom mohli nechat v aktuální podobě. V případě, že ale máme některé produkty již na svém skladu, zajímal by nás především údaj udávající počet kusů. Proto bychom v tomto sloupečku museli upravit data s podmínkou, kde v případě hodnoty „skladem“ bez číslovky (tedy u produktů bez výběru) informaci ponecháme a je na nás, zda se budeme chtít následně dotázat dodavatel na upřesnění počtu kusů nebo na našem webu uvedeme například pouze „skladem u dodavatele“. V případě, že informace obsahuje číslovku, odebereme text „skladem “ a „ks“, použijeme funkci as.numeric(), aby ve sloupečku zůstala pouze ona číslice, se kterou bychom byli schopni pracovat, bude-li potřeba. Záleží vždy na nás, jak se staženými daty chceme naložit.

Také by nás mohli zajímat podrobnější informace o produktech, jako jsou například materiál nebo délka rukávu u oblečení, váha a rozměry u batohů a podobně. V tom případě je zapotřebí prozkoumat e-shop hlouběji, protože při tvorbě tohoto webu (pravděpodobně jako jakéhokoli jiného) jeho autor nezadal tyto informace u všech produktů stejným způsobem. Někdy jsou tato data uložena v tabulce, jindy neuspořádána v pouhém textu a někde dokonce úplně chybí. Proto by po takovéto analýze bylo nutné kód pro získání podrobnějších informací dále upravit.

Tabulka 1: Extrahované produkty uložené v souboru „produktydata.csv“

Název produktu	Cena v Kč	První výběr	Druhý výběr	Dostupnost	Info o produktu	Odkaz na stránku produktu
VESTA EG CANOE RENT	899	L/XL	TMAVĚ ČERVENÁ	skladem 2 ks	Plovací vesta EG CANOE RENT LIMITED je vhodná pro všechny vodr	https://shop.honza-centrum.cz/vesta-eg-canoe-rent
VESTA EG CANOE RENT	899	XXXL	TMAVĚ ČERVENÁ	skladem 3 ks	Plovací vesta EG CANOE RENT LIMITED je vhodná pro všechny vodr	https://shop.honza-centrum.cz/vesta-eg-canoe-rent
VESTA EG CANOE RENT	899	S-M	TMAVĚ ČERVENÁ	skladem 3 ks	Plovací vesta EG CANOE RENT LIMITED je vhodná pro všechny vodr	https://shop.honza-centrum.cz/vesta-eg-canoe-rent
VESTA X-ELEMENTS CANOE	950	3XL	ČERVENÁ	skladem 4 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
VESTA X-ELEMENTS CANOE	950	3XL	MODRÁ	skladem 6 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
VESTA X-ELEMENTS CANOE	950	3XL	ZELENÁ	skladem 7 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
VESTA X-ELEMENTS CANOE	950	L-XL	ČERVENÁ	skladem 4 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
VESTA X-ELEMENTS CANOE	950	L-XL	MODRÁ	skladem 7 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
VESTA X-ELEMENTS CANOE	950	L-XL	ZELENÁ	skladem 6 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
VESTA X-ELEMENTS CANOE	950	S-M	ČERVENÁ	skladem 4 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
VESTA X-ELEMENTS CANOE	950	S-M	MODRÁ	skladem 4 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
VESTA X-ELEMENTS CANOE	950	S-M	ZELENÁ	skladem 9 ks	Vodácká plovací vesta vhodná pro všechny vodní sporty. Ideální na	https://shop.honza-centrum.cz/vesta-xelements-canoe
HIKO PLOVACÍ VESTA BABY VELIKOST 1	941	NENÍ VÝBĚR	NENÍ VÝBĚR	skladem	Zkonstruována tak, aby dítěti při pádu do vody pomáhala v přetoč	https://shop.honza-centrum.cz/vesta-baby/1
HIKO PLOVACÍ VESTA BABY VELIKOST 2	941	NENÍ VÝBĚR	NENÍ VÝBĚR	skladem	Zkonstruována tak, aby dítěti při pádu do vody pomáhala v přetoč	https://shop.honza-centrum.cz/vesta-baby/2
HIKO PLOVACÍ VESTA BABY VELIKOST 3	941	NENÍ VÝBĚR	NENÍ VÝBĚR	skladem	Zkonstruována tak, aby dítěti při pádu do vody pomáhala v přetoč	https://shop.honza-centrum.cz/vesta-baby/3
VESTA X-ELEMENTS BABY	990	L 31-40 KG	ORANGE	skladem 5 ks	Záchranná vesta pro ty nejmenší vodáky zaručí bezpečí při každé si	https://shop.honza-centrum.cz/vesta-xelements-baby
VESTA X-ELEMENTS BABY	990	L 31-40 KG	REFLEXNÍ ZELENÁ	skladem 1 ks	Záchranná vesta pro ty nejmenší vodáky zaručí bezpečí při každé si	https://shop.honza-centrum.cz/vesta-xelements-baby
VESTA X-ELEMENTS BABY	990	M 21-30 KG	ORANGE	skladem 2 ks	Záchranná vesta pro ty nejmenší vodáky zaručí bezpečí při každé si	https://shop.honza-centrum.cz/vesta-xelements-baby
VESTA X-ELEMENTS BABY	990	M 21-30 KG	ŽLUTÁ	skladem 1 ks	Záchranná vesta pro ty nejmenší vodáky zaručí bezpečí při každé si	https://shop.honza-centrum.cz/vesta-xelements-baby
HIKO SWIFT 600 PLOVACÍ VESTA	1130	XS	ČERNÁ	skladem 1 ks	Univerzální lehká vesta z oděru vzdorného materiálu s barevným r	https://shop.honza-centrum.cz/vesta-hiko-swift-600-pfd
HIKO SWIFT 600 PLOVACÍ VESTA	1130	S/M	ČERNÁ	skladem 5 ks	Univerzální lehká vesta z oděru vzdorného materiálu s barevným r	https://shop.honza-centrum.cz/vesta-hiko-swift-600-pfd
HIKO SEAHORSE PLOVACÍ VESTA	1320	S/M	MODRÁ	skladem 1 ks	Univerzální, velmi lehká plovací vesta s širokým rozsahem použití	https://shop.honza-centrum.cz/vesta-hiko-seahorse-pfd
HIKO SEAHORSE PLOVACÍ VESTA	1320	2XL	MODRÁ	skladem 2 ks	Univerzální, velmi lehká plovací vesta s širokým rozsahem použití	https://shop.honza-centrum.cz/vesta-hiko-seahorse-pfd
HIKO K-TOUR PLOVACÍ VESTA	1395	XS	ČERVENÁ	skladem 2 ks	Vesta sportovní střihu vhodná pro široký rozsah použití, převážně	https://shop.honza-centrum.cz/vesta-hiko-k-tour
HIKO K-TOUR PLOVACÍ VESTA	1395	XS	MODRÁ	skladem 2 ks	Vesta sportovní střihu vhodná pro široký rozsah použití, převážně	https://shop.honza-centrum.cz/vesta-hiko-k-tour
HIKO K-TOUR PLOVACÍ VESTA	1395	S/M	MODRÁ	skladem 5 ks	Vesta sportovní střihu vhodná pro široký rozsah použití, převážně	https://shop.honza-centrum.cz/vesta-hiko-k-tour
HIKO K-TOUR PLOVACÍ VESTA	1395	L/XL	ČERVENÁ	skladem 2 ks	Vesta sportovní střihu vhodná pro široký rozsah použití, převážně	https://shop.honza-centrum.cz/vesta-hiko-k-tour
HIKO K-TOUR PLOVACÍ VESTA	1395	L/XL	MODRÁ	skladem 1 ks	Vesta sportovní střihu vhodná pro široký rozsah použití, převážně	https://shop.honza-centrum.cz/vesta-hiko-k-tour
HIKO K-TOUR PLOVACÍ VESTA	1395	XXL	ČERVENÁ	skladem 3 ks	Vesta sportovní střihu vhodná pro široký rozsah použití, převážně	https://shop.honza-centrum.cz/vesta-hiko-k-tour
HIKO K-TOUR PLOVACÍ VESTA	1395	XXL	MODRÁ	skladem 1 ks	Vesta sportovní střihu vhodná pro široký rozsah použití, převážně	https://shop.honza-centrum.cz/vesta-hiko-k-tour
HIKO CINCH HARNESS PLOVACÍ VESTA	3410	S/M	MODRÁ	skladem 2 ks	Výtlačná plovací vesta určená na divokou vodu.Konstrukčně je nav	https://shop.honza-centrum.cz/vesta-hiko-cinch-harness-l/xl-cerv
HIKO CINCH HARNESS PLOVACÍ VESTA	3410	L/XL	MODRÁ	skladem 2 ks	Výtlačná plovací vesta určená na divokou vodu.Konstrukčně je nav	https://shop.honza-centrum.cz/vesta-hiko-cinch-harness-l/xl-cerv
HIKO CINCH HARNESS PLOVACÍ VESTA	3410	XXL	ČERVENÁ	skladem 2 ks	Výtlačná plovací vesta určená na divokou vodu.Konstrukčně je nav	https://shop.honza-centrum.cz/vesta-hiko-cinch-harness-l/xl-cerv
HIKO CINCH HARNESS PLOVACÍ VESTA	3410	2XL	MODRÁ	skladem 1 ks	Výtlačná plovací vesta určená na divokou vodu.Konstrukčně je nav	https://shop.honza-centrum.cz/vesta-hiko-cinch-harness-l/xl-cerv
HIKO GUARDIAN 3.D	4930	S/M	ČERNÁ/STEALTH	skladem 1 ks	Anatomicky tvarovaná vesta umožňující volný pohyb s vysokým vz	https://shop.honza-centrum.cz/vesta-hiko-guardian-3d
HIKO GUARDIAN 3.D	4930	L/XL	ČERNÁ/STEALTH	skladem 1 ks	Anatomicky tvarovaná vesta umožňující volný pohyb s vysokým vz	https://shop.honza-centrum.cz/vesta-hiko-guardian-3d
HIKO GUARDIAN 3.D	4930	L/XL	ČERVENO-ORANŽOVÁ/INFERNO	skladem 1 ks	Anatomicky tvarovaná vesta umožňující volný pohyb s vysokým vz	https://shop.honza-centrum.cz/vesta-hiko-guardian-3d
HIKO GUARDIAN 3.D	4930	XXL	ČERNÁ/STEALTH	skladem 2 ks	Anatomicky tvarovaná vesta umožňující volný pohyb s vysokým vz	https://shop.honza-centrum.cz/vesta-hiko-guardian-3d
HIKO GUARDIAN 3.D	4930	XXL	MODRO-ORANŽOVÁ/WAIKIKI	skladem 1 ks	Anatomicky tvarovaná vesta umožňující volný pohyb s vysokým vz	https://shop.honza-centrum.cz/vesta-hiko-guardian-3d

Zdroj: vlastní

5 Porovnání funkčnosti nástroje na jiném e-shopu

Vytvořený nástroj pro extrakci při užití na e-shopu https://shop.honza-centrum.cz/ je plně funkční. Nabízí se otázka, je-li reálné ho použít i na jiné weby. Pro tuto tento pokus byl vybrán e-shop PandaOutdoor.cz dostupný na URL adrese https://www.pandaoutdoor.cz/. Jedná se o internetové stránky stejnojmenné firmy PandaOutdoor, se sídlem na adrese Prokopova 373/26, 397 01 Písek, na kterých nabízí vybavení pro cestovatele. Tento e-shop byl vybrán na základě stejných kritérií jako předchozí Vodácké a kempingové potřeby - Eshop Praha 5.

Na první pohled se může zdát, že díky podobnému vzhledu stránek bude možné nástroj použít. Začneme provádět analýzu tohoto webu. Z Obrázku 15 je zřejmé, že se zde nachází 13 kategorií, jež můžou být dále rozděleny na jednotlivé další části. Na stránce podkategorie „Oblečení“ nalezneme produkty, které se zobrazují na celkem čtyřiceti pěti stránkách. Na obrázku také vidíme tlačítko se textem „zobrazit vše“, po jehož kliknutí se načtou všechny produkty najednou. Při pohledu na stránky produktů (viz Obrázek 16) je možné nalézt název produktu, jeho cenu (běžnou, po určité slevě nebo pro registrované uživatele), dostupnost, nějaká pole výběru, podrobnější informace a samozřejmě odkaz URL.

Jenže vzhled může klamat. Proto je zapotřebí prozkoumat zdrojový kód, který uchovává námi žádané informace pro extrakci. Ten se liší totiž v mnoha ohledech.

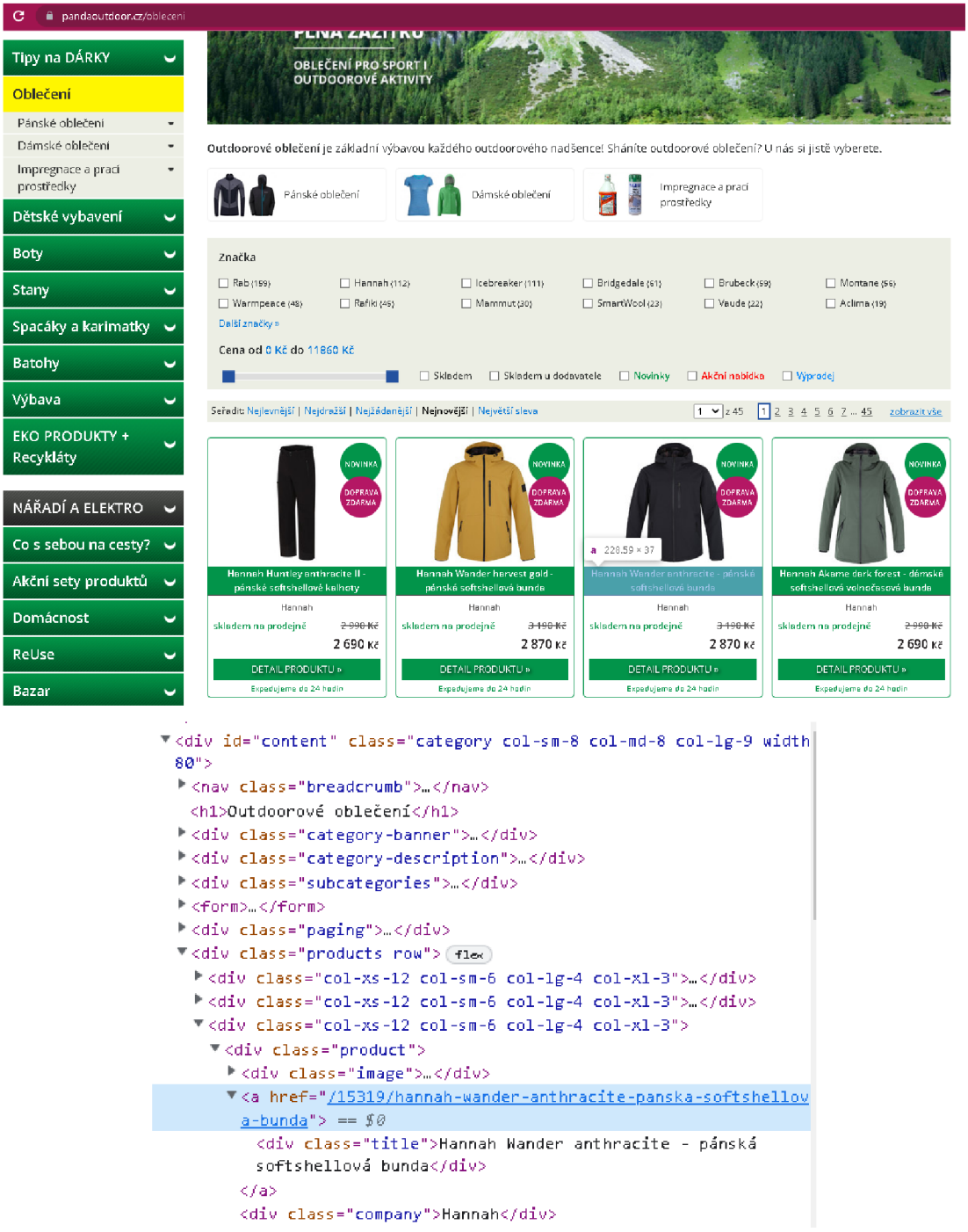
Pokud bychom pomocí vytvořeného nástroje chtěli nalézt například část URL odkazu jednotlivých produktů, cestu k tomuto elementu (kde je informace uchována) místo původní:

„//*[@id="product-list"]/div[',as.character(zac),']/div/div/div[1]/div[1]/h4/a“ užijeme následující: „//*[@id="content"]/div[5]/div[',as.character(zac),']/div/a“ (viz Obrázek 15). Informace ve zdrojových kódech jsou od sebe odlišné minimálně názvy id daných částí.

Stejně tak nalezení názvu produktu, by zde opět nebylo funkční s použitím cesty „//*[@id="foto"]/h1“, jelikož na tomto webu, je vzhled hledané části vytvořen pomocí jiného názvu id. Pro extrakci z tohoto webu bychom museli cestu změnit na hodnotu „//*[@id="content"]/h1“ (viz Obrázek 16).

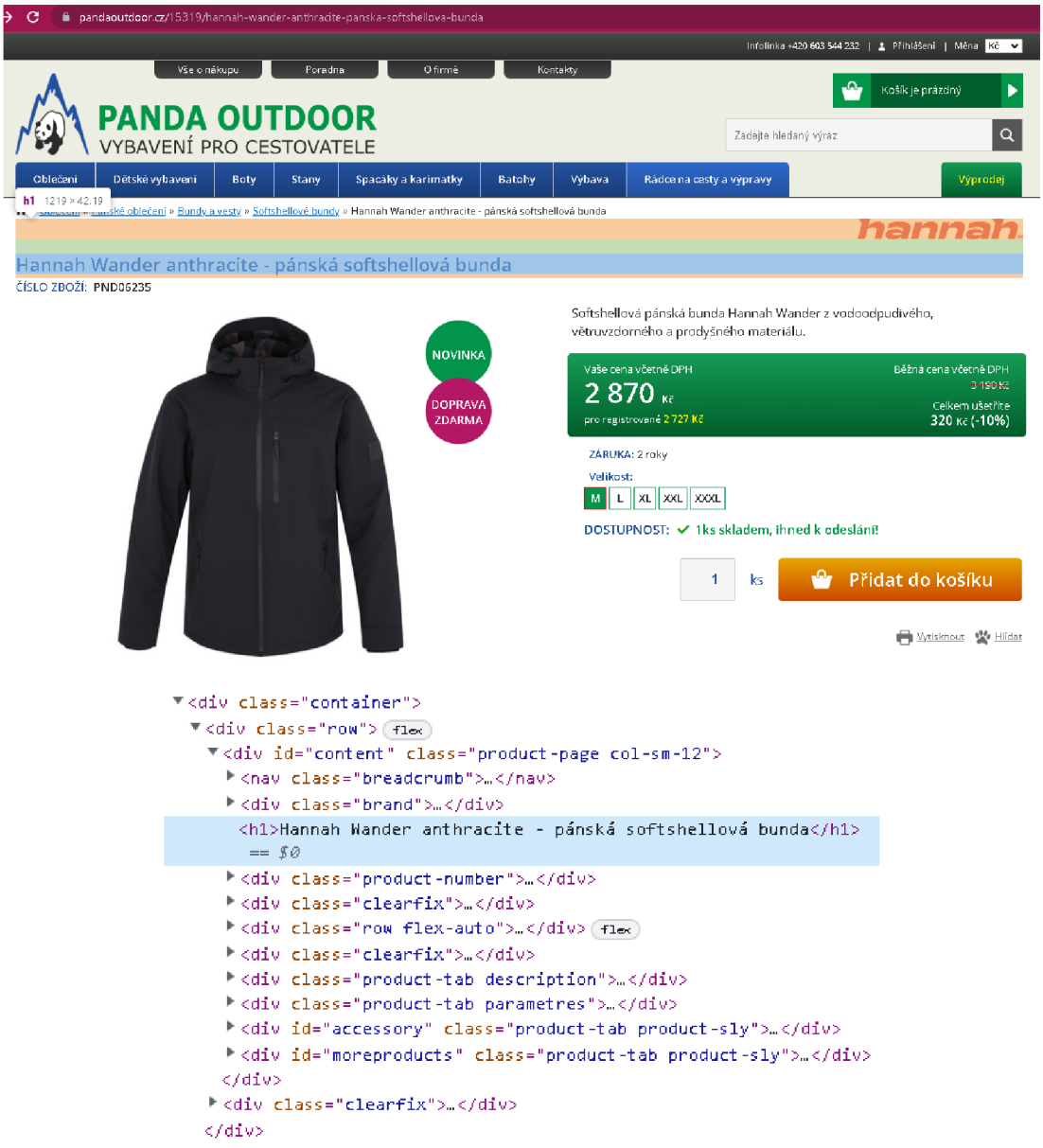
Takovéto úpravy je zapotřebí provést u všech příkazů, kde se budou elementy ve zdrojových kódech lišit. Jenže již při pohledu na stránku tohoto produktu je jasné, že ke stažení dat z e-shopu PandaOutdoor.cz budeme používat odlišný postup při výběru variant, jelikož zde jsou již všechny zobrazeny. Nebude zapotřebí rozklikávat pole voleb, pouze je postupně vybrat z nabízených přímo na stránce.

Obrázek 15: Vzhled stránky kategorie „Oblečení" se zdrojovým kódem



Zdroj: (Outdoor vybavení a sportovní potřeby - PandaOutdoor.cz)

Obrázek 16: Vzhled stránky produktu „Hannah Wander anthracite - pánská softshellová bunda“ se zdrojovým kódem



Zdroj: (Outdoor vybavení a sportovní potřeby - PandaOutdoor.cz)

Můžeme tedy konstatovat, že pro extrakci z odlišných webů je zapotřebí celý proces (analýzu, návrh a tvorbu nástroje) u každého dalšího e-shopu provést opakovaně.

Závěr

Vytvořený nástroj pro účel extrakce dat z webových stránek nám může usnadnit práci v řádu minut až hodin v závislosti na množství dat, která stahujeme. V opačném případě bychom strávili mnoho hodin až dnů manuální prací v podobě kopírování nebo psaní, při které bychom se dopouštěli množství chyb. Samozřejmě je nutné nějaký čas věnovat analýze webu a psaní samotného kódu. Následně budeme mít jistotu, že data jsou bez chyb a příště, až je budeme aktualizovat, už nebudeme muset trávit čas znovu psaním, pouze využijeme již námi vytvořený postup. Zpočátku se jedná o vyšší časovou investici pro vytvoření kódu, ta nám ale zařídí mnohonásobně větší úsporu času při stahování aktuálním i těch příštích. Nejedná se o nákladné řešení, neboť vybraný program Rstudio je volně dostupný pro všechny operační systémy.

Hlavní cíl práce, provedení návrhu extrakce atributů produktů z webů, je rozebrán v kapitole Extrakce dat z vybraného e-shopu. Sestrojený nástroj využijí spíše firmy, než běžní uživatelé webových stránek[12]. Pokud se podíváme z pohledu podniku na námi řešený problém, znamená to například, že existuje firma, která vlastní e-shop a má své dodavatele, kteří nabízejí produkty na svých stránkách. Jenže právě kvůli množství produktů odebíraných od dodavatele naše firma nemá na webu uvedeny všechny nabízené. Pro takovýto podnik, by byla možnost automatizovaného stahování dat z webu dodavatele obrovským přínosem. Jedno z odvětví, ve kterém autorka vidí využití poznatků z této práce, je například stavebnictví, kde nalezneme právě velkou řadu produktů. Informace o nich musí zaměstnanci stavebnin pracně stahovat z webů jednotlivých dodavatelů. Pokud se ovšem pro podrobnější informace jednoduše neodkazují na jejich weby, což běžného uživatele často odradí. Samozřejmě se firma může rozhodnout pro získávání dat z více důvodů. Nejen aby uváděla konkrétnější informace o produktech na svém webu, ale také například pro vlastní interní zdroje, aby databáze podniku měla podrobnější přehledy.

Při provedení srovnání funkčnosti vytvořeného nástroje pro extrakci z jiného e-shopu byla potvrzena teorie, že pokud chceme stahovat informace z dalších webů,

[12] Pro potřeby běžných uživatelů internetu již existují weby, které uchovávají podrobné informace o produktech z různých e-shopů (nazývané jako srovnávače). Příkladem může být Heureka.cz nebo Zboží.cz.

bude zapotřebí kód náležitě upravit. Stejně tak, pokud by nastala situace, kdy by se změnila struktura webu, ze kterého provádíme extrakci opakovaně[13], museli bychom kód těmto změnám náležitě přizpůsobit pro opětovnou funkčnost. Tato zjištěná nevýhoda má pro podniky odlišnou váhu. Pokud by bylo zapotřebí extrahovat data pouze z jednoho webu, jistě jí při rozhodování přiřadíme váhu nižší, než kdyby nás čekala úprava kódu pro více webů. Je tedy na dané firmě, jestli se rozhodne investovat finanční zdroje podniku a čas svých IT pracovníků[14] do sestrojení podobného nástroje pro automatizovanou extrakci, popřípadě bude nadále zaměstnávat pracovníky, kteří získávání dat provádí manuálně, a nebo nebude chtít žádné informace z dalších webů sbírat jakýmkoli způsobem.

Při tomto výběru strategie mějme na paměti, co řekl již L. A. Seneca: „*Není pravda, že máme málo času. Pravdou ale je, že ho hodně promarníme.*“ (Seneca, 2004).

Vypracování této diplomové práce bylo přínosem i pro samotnou autorku práce. Rozšířila si znalosti o užívání programu Rstudio, se kterým měla dříve pouze základní zkušenosti z odvětví matematiky či statistiky. Zároveň si oživila dovednosti v programování při tvorbě nástroje pro automatizovanou extrakci.

Na závěr lze říci, že nastavené cíle byly v této práci naplněny.

[13] Například by firma prováděla stahování pro aktualizaci dat ve frekvenci několikrát denně, týdně či ročně (záleží na účelu použití extahovaných informací).

[14] Popřípadě může využít služeb externí IT firmy.

I. Summary

This thesis deals with the so-called web scraping, specifically showing a possible way to extract product data from the web and web pages on a chosen example.

The theoretical part is devoted to the description of the website and its source code. In order to download data, you first need to understand this information. Next, the R language environment and some methods, which are subsequently used in the practical part during web extraction, are introduced.

The practical part contains procedures (analysis and design) for the creation of the tool used for scraping. Data is downloaded from a specific e-shop, and its extraction success is checked. The last part is dedicated to comparing the functionality of the created tool on another e-shop.

Keywords: Web scraping, extraction, RSelenium, dynamic website, product attributes

II. Seznam použité literatury

1) Aydin, O. (2018). *R Web Scraping Quick Start Guide: Techniques a tools to crawl a scrape data from websites*. Packt Publishing Limited. ISBN 978-1789138733.
2) Krotov, V., & Tennyson, M. (2021). Web Scraping in the R Language: A Tutorial. *Journal of the Midwest Association for Information Systems (JMWAIS), 2021 Issue 1*(Article 5), 61-77.
3) Munzert, S. (2015). *Automated data collection with R: a practical guide to Web scraping and text mining*. John Wiley. ISBN 978-1-118-83481-7.
4) Seneca, L. A. (2004). *O duševním klidu* (2. vyd). Baset. ISBN 80-86410-43-9, 80-7340-055-3.
5) Wickham, H., & Grolemund, G. (2016). *R for data science: import, tidy, transform, visualize and model data*. O'Reilly Media. ISBN 978-1-4919-1039-9.

Zdroje dostupné z internetu

6) A Grammar of Data Manipulation. (August 31, 2022). In (p. 80). https://cran.r-project.org/web/packages/dplyr/dplyr.pdf.
7) Beckman, M., Guerrier, S., Lee, J., Molinari, R., Orso, S. O., & Rudnytskyi, I. (2020). *An Introduction to Statistical Programming Methods with R*. https://smac-group.github.io/ds/
8) Bradley, A., & James, R. J. E. (2019). Web Scraping Using R. *Advances in Methods and Practices in Psychological Science*, *2*(3), 264-270. https://doi.org/10.1177/2515245919859535.
9) Condylios, S., Rudis, B., & Kiener, P. (August 28, 2022). Retrieve Network Statistics Including Available TCP Ports. In (p. 6). https://cran.r-project.org/web/packages/netstat/netstat.pdf.
10) Easily Install and Load the 'Tidyverse'. (July 18, 2022). In H. Wickham (p. 6). https://cran.r-project.org/web/packages/tidyverse/tidyverse.pdf.
11) Harrison, J., & Yeong Kim, J. (September 2, 2022). R Bindings for 'Selenium WebDriver'. In (p. 18). https://cran.r-project.org/web/packages/RSelenium/RSelenium.pdf.
12) Meissne, P., Ren, K., Keys, O., & Fitz John, R. (September 3, 2020). A 'robots.txt' Parser and 'Webbot'/'Spider'/'Crawler' Permissions Checker. In (p. 20). https://cran.r-project.org/web/packages/robotstxt/robotstxt.pdf.
13) *Outdoor vybavení a sportovní potřeby - PandaOutdoor.cz.* Retrieved September 15, 2022, from https://www.pandaoutdoor.cz/.

14) Temple Lang, D., Kalibera, T., & Team, C. (June 10, 2022). Tools for Parsing and Generating XML Within R and S-Plus. In (p. 172). https://cran.r-project.org/web/packages/XML/XML.pdf.

15) van Wel, L., & Royakkers, L. (2004). Ethical issues in web data mining. *Ethics and Information Technology*, *6*(2), 129-140. https://doi.org/10.1023/B:ETIN.0000047476.05912.3d.

16) *Understanding the components :: Documentation for Selenium*. Retrieved September 10, 2022, from https://www.selenium.dev/documentation/overview/components/.

17) *Vodácké a kempingové potřeby- Eshop Praha 5*. Retrieved September 15, 2022, from https://shop.honza-centrum.cz/.

18) Wickham, H. (August 20, 2022). Easily Harvest (Scrape) Web Pages. In (p. 13). https://cran.r-project.org/web/packages/rvest/rvest.pdf.

19) Wickham, H. (August 21, 2022). Simple, Consistent Wrappers for Common String Operations. In (p. 32). https://cran.r-project.org/web/packages/stringr/stringr.pdf.

III. Seznam obrázků

IV. Seznam tabulek

V. Seznam zkratek

AJAX – Asynchronous JavaScript and XML

CSS – Cascading Style Sheet

DOC TYPE – Document type declaration

DTD – Dokument type definition

FTP – File Transfer Protocol

HTML – HyperText Markup Language

HTTP/HTTPS – Hypertext Transfer Protocol (Secure)

JASON – JavaScript Object Notation

TCP – Transmission Control Protocol

XML – Extensible Markup Language

Xpath – XML dokument path

WHATWG – Web Hypertext Application Technology Working Group

WWW – World Wide Web

W3C – World Wide Web Consortium

VI. Seznam příloh

Sestrojený kód pro extrakci z vybraného webu v prostředí R

VII. Přílohy

```
library(rvest)
library(dplyr)
library(tidyverse)
library(robotstxt)
library(RSelenium)
library(XML)
library(netstat)
#načtení zdrojového kódu HTML stránky
honzaURL <- ("https://shop.honza-centrum.cz/vodacke-vesty")
#kontrola povolení škrabání
robotstxt::paths_allowed(honzaURL)
#použitím RSelenia načteme všechny produkty na jednu stránku
#otevřít chrome následovně
#binman::list_versions("chromedriver")#nabídne nejnovější verze
remdriver_objekt <- rsDriver(browser = 'chrome',
                             chromever = '105.0.5195.52',
                             verbose = FALSE,
                             port = free_port())
Okno <- remdriver_objekt$client
Okno$navigate(honzaURL)
#odmítnutí cookies
#Okno$deleteAllCookies()
Okno$findElement('class',value='cc-nb-reject')$clickElement()
#příhlášení
Okno$findElement('class','account')$clickElement()
#jmeno
```

```
Okno$findElement('name','login')$sendKeysToElement(list('chvosto
vapetra@seznam.cz'))

#heslo

Okno$findElement('name',
'heslo')$sendKeysToElement(list('TheJerk246', key = 'enter'))

#zobrazení všech produktů na jedné stránce

#vsechnyklik

Okno$findElement('xpath','//*[@id="filter-
zbozi"]/div[2]/div/button')$clickElement()

pole <- Okno$findElement('xpath','//*[@id="filter-
zbozi"]/div[2]/div/ul')$getElementText()

pole <- gsub("\n"," ",pole)

pole <- strsplit(pole," ")

pole <- length(pole[[1]])

Okno$findElement('xpath',str_c('//*[@id="filter-
zbozi"]/div[2]/div/ul/li[',as.character(pole),']'))$clickElement
()

#použijeme XML

produkty <- read_html(unlist(Okno$getPageSource()))

produktyXML <- htmlParse(produkty,encoding = "UTF-8")

produktyXMLtxt <- paste(capture.output(produktyXML,
file=NULL),collapse = "\n")

pocetNaStranku <- str_count(produktyXMLtxt,pattern = "item
category col-xs-12 col-sm-6 col-lg-3")

get_pole <- function(){

  Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/button')$clickElement()

  Sys.sleep(0.5)

  pole <<- Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/ul')$getElementText()

  pole <<- gsub("\n"," ",gsub(" ","\u00A0",pole))
```

```
  pole <<- strsplit(pole," ")
  pole <<- length(pole[[1]])
}
get_pole2 <- function(){
  Sys.sleep(0.5)
  Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button')$clickElement()
  pole2        <<-        Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/ul')$getElementText()
  pole2 <<- gsub("\n"," ",gsub(" ","\u00A0",pole2))
  pole2 <<- strsplit(pole2," ")
  pole2 <<- length(pole2[[1]])
}
#zavedení důležitých proměnných
cenaproduktu <- c()
informace <- c()
dostupnost <-c()
odkazyProduktu <- c()
nazvyProduktu <- c()
prvnivyber <- c()
druhyvyber <- c()
#funkce pro extrakci dat
get_atri <- function(){
  Sys.sleep(0.5)
  odkazyProduktu <<- append(odkazyProduktu, (odkaz))
  path                <<-                str_c('//*[@id="product-
list"]/div[',as.character(zac),']/div/div/div[1]/div[3]/ul/li/sp
an')
  informace     <<-     append(informace,xpathSApply(produktyXML,
path=path,xmlValue))
```

```
  nazev<<-
Okno$findElement('xpath','//*[@id="foto"]/h1')$getElementText()
  nazvyProduktu <<- append(nazvyProduktu,nazev[[1]])
  dostupnost1<<-
Okno$findElement('xpath','//*[@id="cena"]/div/div[1]/div/strong'
)$getElementText()
  dostupnost <<- append(dostupnost,dostupnost1[[1]])
  Sys.sleep(0.5)
}
#funkce pro extrakci ceny
get_cenu <- function(){
  stranka <- read_html(Okno$getPageSource() %>% unlist())
  strankaXML <- htmlParse(stranka,encoding = "UTF-8")
  strankaXMLtxt       <-       paste(capture.output(strankaXML,
file=NULL),collapse = "\n")
  ceny <- str_count(strankaXMLtxt,pattern = "total-price")
  if(ceny==1){
    cenaproduktu1<<-
Okno$findElement('xpath','//*[@class="widget-header      widget-
header-small                form-inline-group                no-
radius"]/div[4]')$getElementText()
  }else if(ceny==2){
    cenaproduktu1<<-
Okno$findElement('xpath','//*[@class="widget-header      widget-
header-small                form-inline-group                no-
radius"]/div[7]')$getElementText()
  }else {
    cenaproduktu1<<-
Okno$findElement('xpath','//*[@class="widget-header      widget-
header-small                form-inline-group                no-
radius"]/div[10]')$getElementText()
  }
```

```
  cenaproduktu<<-
append(cenaproduktu,as.numeric(str_remove_all(cenaproduktu1[[1]]
,"[ Kč]")))
}
#fce pro extrakci hodnot výběrů
get_vybery <- function(){
  Sys.sleep(1)
  prvnivyber1      <-      Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/button')$getElementText()
  prvnivyber <<- append(prvnivyber,prvnivyber1[[1]])
  druhyvyber1      <-      Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button')$getElementText()
  druhyvyber <<- append(druhyvyber,druhyvyber1[[1]])
}
#cyklus průchodu produktů
for (zac in seq(from=1,to= pocetNaStranku, by=1)) {
  cesta                <-                str_c('//*[@id="product-
list"]/div[',as.character(zac),']/div/div/div[1]/div[1]/h4/a')
  odkaz                <-                xpathSApply(produktyXML,
path=cesta,xmlGetAttr,"href")%>%      paste("https://shop.honza-
centrum.cz", ., sep = "")
    Okno$navigate(odkaz)
    stranka <- read_html(Okno$getPageSource() %>% unlist())
    strankaXML <- htmlParse(stranka,encoding = "UTF-8")
    strankaXMLtxt       <-       paste(capture.output(strankaXML,
file=NULL),collapse = "\n")
    Sys.sleep(0.5)
    klikani <- str_count(strankaXMLtxt,pattern = "ax-variants")
    if(klikani>0){
      plocha           <-           stranka%>%html_element("#ax-
variants")%>%toString.XMLNode()
```

```
      vybery <- str_count(plocha,pattern = "form-group")
        if(vybery==1){
          #pouze jeden výběr (např z barev)
          get_pole()
          if(pole==1){
            #vyberJedno
            Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/ul/li/a')$clickElement()
            prvnivyber1<- Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/button')$getElementText()
            prvnivyber <- append(prvnivyber,prvnivyber1)
            druhyvyber <- append(druhyvyber,"NENÍ VÝBĚR")
            get_atri()
            get_cenu()
            }else{
              for(i in seq(from=2, to=pole, by=1)){
                vyber1<-
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div/div/div/ul/li[',as.character(i),']/a'))$getElemen
tAttribute("class")
                if(vyber1!="disabled"){
                  #vyberJedno
                  Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div/div/div/ul/li[',as.character(i),']/a'))$clickElem
ent()
                  prvnivyber1<-
Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/button')$getElementText()
                  prvnivyber <- append(prvnivyber,prvnivyber1)
                  druhyvyber <- append(druhyvyber,"NENÍ VÝBĚR")
                  get_atri()
```

```
                get_cenu()
                #poleVyberu1
                Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/button')$clickElement()
              }
            }
          }
        }
        else if(vybery==2){
          get_pole()
          if(pole==1){
            #vyber1
            Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/ul/li/a')$clickElement()
            get_pole2()
            Sys.sleep(0.5)
            if(pole2==1){
              #vyber2
              Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/ul/li/a')$clickElement()
              get_atri()
              get_cenu()
              get_vybery()
            }else{
                for(iiii in seq(from=2, to=pole2, by=1)){
                  vyber2<-
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[2]/div/div/ul/li[',as.character(iiii),']/a'))$get
ElementAttribute("class")
                  if(vyber2!="disabled"){
```

```
                    #vyber2
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[2]/div/div/ul/li[',as.character(iiii),']/a'))$cli
ckElement()
                    get_vybery()
                    get_atri()
                    get_cenu()
                    #polevyberu2
                    Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button/')$clickElement()
                  }
                }
              }
            }
            else{
              for(iii in seq(from=2, to=pole, by=1)){
                #vyber1
                vyber1<-
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[1]/div/div/ul/li[',as.character(iii),']/a'))$getE
lementAttribute("class")
                if(vyber1!="disabled"){
                  Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[1]/div/div/ul/li[',as.character(iii),']/a'))$clic
kElement()
                  get_pole2()
                  if(pole2==1){
                    #vyber2
                    Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/ul/li/a')$clickElement()
                    get_vybery()
```

```
                    get_atri()
                    get_cenu()
                  }else{
                    for(iiii in seq(from=2, to=pole2, by=1)){
                      vyber2<-
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[2]/div/div/ul/li[',as.character(iiii),']/a'))$get
ElementAttribute("class")
                      if(vyber2!="disabled"){
                      Sys.sleep(0.5)
                      #vyber2
Okno$findElement('xpath',str_c('//*[@id="ax-
variants"]/div[2]/div/div/ul/li[',as.character(iiii),']/a'))$cli
ckElement()
                      get_vybery()
                      get_atri()
                      get_cenu()
                      #polevyberu2
                      Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button')$clickElement()
                      }
                    }
                  }
                }
                #"vynulovat" výběry
                Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/button')$clickElement()
                Okno$findElement('xpath','//*[@id="ax-
variants"]/div[1]/div/div/ul/li[1]/a')$clickElement()
                Sys.sleep(0.5)
```

```
                Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/button')$clickElement()
                Okno$findElement('xpath','//*[@id="ax-
variants"]/div[2]/div/div/ul/li[1]/a')$clickElement()
                Sys.sleep(0.5)
                #poleVyberu1
                Okno$findElement('xpath','//*[@id="ax-
variants"]/div/div/div/button')$clickElement()
              }
            }
        }
    }else{
      druhyvyber <- append(druhyvyber,"NENÍ VÝBĚR")
      prvnivyber <- append(prvnivyber,"NENÍ VÝBĚR")
      get_atri()
      get_cenu()
    }
}
Okno$quit()
#zavřít server a otevřít porty
system("taskkill /im java.exe /f")
produktydata <- list()
produktydata<-
rbind(produktydata,data.frame(nazvyProduktu,cenaproduktu,prvnivy
ber,druhyvyber,dostupnost,informace,odkazyProduktu,stringsAsFact
ors=FALSE))
colnames(produktydata) <- c("Název produktu","Cena v Kč","První
výběr", "Druhý výběr","Dostupnost","Info o produktu","Odkaz na
stránku produktu")
write.csv(produktydata,"produktydata.csv") # getwd()
```