ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

MASARYKŮV ÚSTAV VYŠŠÍCH STUDIÍ

DIPLOMOVÁ PRÁCE

Inovační projekt aplikace pro statistickou analýzu v prostředí prohlížeče a Node.js

Innovation Project of a Statistical Analysis Application in Browser and Node.js Environment

2023

Pavel Mužík

Studijní program: N0413A050002 - Projektové řízení inovací

Studijní obor: Projektové řízení

Vedoucí práce: Ing. Tomáš Löster, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mužík** Jméno: **Pavel** Osobní číslo: **481818**

Fakulta/ústav: **Masarykův ústav vyšších studií**

Zadávající katedra/ústav: **Institut ekonomických studií**

Studijní program: **Projektové řízení inovací**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Inovační projekt aplikace pro statistickou analýzu v prostředí prohlížeče a Node.js

Název diplomové práce anglicky:

Innovation Project of a Statistical Analysis Application in Browser and Node.js Environment

Pokyny pro vypracování:

Tvorba aplikace pro analýzu socio-ekonomických dat v prostředí Node.js

Seznam doporučené literatury:

Doležal, J. a kol.: Projektový management. Grada, 2016. ISBN 978-80-247-5620-2.
Brown, E.: Web Development with Node and Express 2e. O'Reilly, 2017. ISBN 978-1491988640.
HINDLS, Richard, ARLTOVÁ, Markéta, HRONOVÁ, Stanislava, MALÁ, Ivana, MAREK, Luboš, PECÁKOVÁ, Iva, ŘEZANKOVÁ, Hana. Statistika v ekonomii. 1. vyd. Příbram : Professional Publishing, 2018. 395 s. ISBN 978-80-88260-09-7.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Tomáš Löster, Ph.D. Masarykův ústav vyšších studií ČVUT v Praze

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **09.12.2022** Termín odevzdání diplomové práce: **27.04.2023**

Platnost zadání diplomové práce: _____

Ing. Tomáš Löster, Ph.D.	Mgr. František Hřebík, Ph.D.	prof. PhDr. Vladimíra Dvořáková, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ _____

Datum převzetí zadání Podpis studenta

MUŽÍK, PAVEL. *Inovační projekt aplikace pro statistickou analýzu v prostředí prohlížeče a Node.js*. Praha: ČVUT 2023. Diplomová práce. České vysoké učení technické v Praze, Masarykův ústav vyšších studií.

MASARYKŮV ÚSTAV
VYŠŠÍCH STUDIÍ
ČVUT V PRAZE

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně. Dále prohlašuji, že jsem všechny použité zdroje správně a úplně citoval a uvádím je v přiloženém seznamu použité literatury.

Nemám závažný důvod proti zpřístupňování této závěrečné práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Praze dne: 14. dubna 2023 Podpis:

Poděkování

Upřímné poděkování náleží vedoucímu práce, Ing. Tomáši Lösterovi, Ph.D., který po celou dobu jejího vzniku projevoval upřímný zájem o její kvalitu a tím přispěl k tomu, že výstupy aplikací přesahují původní plánovaný rámec, protože získaly další smysl v možném využití v praxi. Taktéž mu patří díky za zprostředkování některých testerů aplikace a za četné konzultace k vybraným oblastem. JUDr. Viktoru Rossmannovi tímto děkuji za diskuzi ohledně právní ochrany duševního vlastnictví. Klíčovým krokem v ověřování životaschopnosti produktu je jeho validace, která by se neobešla bez dobrovolníků, kteří testování i následným rozhovorům věnovali svůj čas a pozornost. Bez tisíců hodin práce open-source komunity by dnes ani jedna z aplikací nevypadala ani nefungovala tak, jak ji lze pozorovat. Proto i stovkám developerů, u nichž je často známa pouze přezdívka nebo avatár, patří významná zásluha na výsledku této práce.

Abstrakt

Práce se zabývá možnou inovací současných softwarových řešení pro statistickou analýzu ve smyslu přístupnosti a zohlednění uživatelských potřeb. Statistický software coby kategorie patří k jedné z nejdéle existujících oblastí ve výpočetní technice, přesto je vývoj celé kategorie méně dynamický než v řadě novějších oblastí. Klíčovým výstupem práce jsou dvě aplikace. Knihovna Retusa je výpočetním procesorem v Node.js, která uživateli zprostředkovává možnost provádět statistickou analýzu pomocí desítek metod, přičemž přináší nové prvky jako je validace dat, jejich automatická transformace, dokumentace či vícejazyčnost. Aplikace Retusa GUI je grafickým rozhraním pro knihovnu Retusa a umožňuje využívat její funkce v prostředí internetového prohlížeče i běžným uživatelům. Zákaznické testování naznačilo, že aplikace je dobře hodnocena a může být vítaným nástrojem pro vybrané segmenty uživatelů. Kapitola Komercializace se proto zabývá i možnostmi dalšího rozvoje obou aplikací, s ohledem na jejich budoucí monetizaci. Závěr práce diskutuje obecné problémy statistických softwarů a nabízí úvahu nad jejich možnou budoucí nápravou.

Klíčová slova

statistická analýza, JavaScript, Node.js, agilní řízení, software, uživatelské testování, strategie

Abstract

The thesis deals with the potential innovation of current software solutions for statistical analysis in terms of accessibility and consideration of user needs. Statistical software as a category is one of the longest-standing areas in computing, yet the development of the entire category is less dynamic than some newer areas. The key outputs of the work are two applications. The Retusa library is a computational processor in Node.js that provides users with the ability to perform statistical analysis using dozens of methods, while bringing new elements such as data validation, automatic transformation, documentation, and multilingualism. The Retusa GUI application is a graphical interface for the Retusa library and allows its functions to be used in the environment of a web browser by ordinary users. User testing indicated that the application is well-received and could be a welcome tool for selected user segments. Therefore, the last chapter also deals with the possibilities of further development of both applications, in respect to their future monetization. The conclusion discusses general problems with statistical software and offers considerations on their possible future improvement.

Keywords

statistical analysis, JavaScript, Node.js, agile management, software, user testing, strategy

Obsah

Úvod

Statistika slouží lidstvu nejpozději od starověku (1) a bez statistiky, potažmo statistické analýzy, by byl dnešní svět k nepoznání. Své uplatnění našla v nejrůznějších oborech lidské činnosti a zásadně pomohla uspíšit civilizační vývoj – mimochodem, dnes s nadějí i obavami vnímaný rozvoj umělé inteligence by bez statistiky nebyl vůbec možný. Moderní statistická analýza byla do značné míry podmíněna rozvojem výpočetní techniky, což se od 60. let minulého století projevovalo i skrze rodící se programy, určené pro tuto disciplínu, z nichž některé jsou, alespoň ve smyslu značky, dodnes živé.

V posledních letech si snad nešlo nevšimnout rostoucího důrazu na inovování všeho možného, což dokresluje vznik nejrůznějších inovačních center, bootcampů, hackathonů, zřizovaných či organizovaných státem, samosprávami, univerzitami, nadacemi i firmami. Tyto dva fenomény – statistická analýza a inovace – jsou pilíři, na kterých stojí předložená práce. Cílem bylo vytvořit inovativní řešení pro statistickou analýzu, a to v čistě praktické podobě. Výsledkem je nejen funkční program, ale i lepší pochopení současných neduhů, které lidé na statistické analýze vnímají a naznačení směru, kde lze pro tyto nedostatky hledat opatření. Zmíněný program představují dvě aplikace, které jsou nedílnou součástí této práce.

Mým přáním bylo, aby mě zpracování této práce bavilo a ideálně aby měl její výsledek i nějaký užitek. V prvním případě jsem zřejmě nemohl mít větší štěstí; navrhování a programování takového projektu je činnost mimořádně tvůrčí, svobodná a naplňující, jelikož v relativně krátkém čase vzniká z prázdného javascriptového souboru fungující software, který toho umí v určitých ohledech více, než například populární Excel. Vývoj každé nové části aplikace s sebou přinesl nápady na další inovace a zlepšení, které nebyly - vzhledem k omezenému času – zatím realizovány. Do značné míry a velmi slibně bylo naplněno i přání druhé - jemu se věnuje závěrečná část této práce.

1 Východisko

Pro tuto práci bylo třeba přesněji vymezit, co konkrétně aplikace pro statistickou analýzu znamená, neboť se jedná o velmi široký okruh možných řešení, která jsou mimořádně variabilní co do objemu funkcionalit, architektury, účelů užití, cílových skupin atd. Navíc bylo třeba se vypořádat se skutečností, že zadání této práce bylo realizováno dvěma způsoby (aplikacemi), kdy jedna aplikace – **Retusa**[1] - funguje jak v prostředí Node.js, tak v prohlížeči, zatímco druhá (Retusa GUI) je grafickým rozhraním pro aplikaci první. Z uživatelského i komercializačního hlediska je určující aplikace druhá, čímž je dáno, že se jedná o analytický nástroj s grafickým rozhraním. Vzhledem k předpokládanému využití aplikace (zejména běžné statistické testování, které provádí studenti ekonomických oborů či sociologie) byla podoba výstupu definována jako „**SPSS v prohlížeči**“[2]. Metodika Lean Canvas (2) pro tento typ definice užívá pojem srozumitelný opis (high-level concept) a jejím smyslem - kromě vymezení hranic produktu – je nabídnout i jeho snadno pochopitelný obsah. Koncept *SPSS v prohlížeči* je tedy možno dále chápat jako žádoucí výstup této práce.

Samotný koncept je třeba rozvíjet ještě dále, neboť takto volně pojatou definici lze brát i doslova. Základní charakteristikou projektu je časová omezenost, což by v případě úplné repliky SPSS pro prohlížeče, tvořené jedním vývojářem, byla spíše vlastnost nereálná. Upřesněme tedy, že SPSS je, velmi obecně řečeno, software s vlastním výpočetním enginem (jádrovou komponentou)[3], nad kterým je naprogramováno uživatelské rozhraní, kde může uživatel nahrávat a ukládat data, editovat je v tabulce a zejména nad nimi volat statistické metody, které své výsledky vrací ve formě čísel, tabulek a grafů. Předmětem (scope) inovačního projektu této práce je tedy **navrhnout a naprogramovat výpočetní knihovnu v prostředí Node.js a dále její grafické rozhraní, které bude fungovat v internetovém prohlížeči a bude nabízet podobný typ služeb jako běžné statistické aplikace**.

Klíčovým výstupem této práce jsou dvě aplikace, od nichž se zároveň očekává, že budou nějakým způsobem **inovovat současná řešení**. Pokud by byla vyvinuta pouze výpočetní aplikace v JavaScriptu, nejednalo by se de facto o inovaci, podobná řešení již ostatně existují (viz dále). Samotná inovativnost přístupů i výstupu se opírala o dva základní pilíře: analýzu současných řešení a uživatelské zkušenosti. Zatímco uživatelům se věnuje poslední kapitola, současným řešením následující text.

[1] Pracovní název aplikace (odvozeno o botanického názvu jednoho druhu palisandru, *Dalbergia retusa*) není výsledkem kreativního úsilí, ale prosté zkušenosti, že cílem bylo vytvořit unikátní značku alepoň v kontextu GitHub, kde byly veškeré relevantní názvy pro statistickou knihovnu již někým užívány (dokonce i jména cca 20 nejznámějším statističek a statistiků). Strom známý spíše jako cocobolo pochází z Jižní Ameriky a je velmi ceněn pro své vizuálně a akusticky výjimečné dřevo.

[2] SPSS je jednou z nejznámějších aplikací pro statistickou analýzu a svým způsobem i synonymem pro celou kategorii.

[3] To znamená, že SPSS má tzv. jádro, které provádí výpočetní operace. Např. poslední verze aplikace Statistica již od svého vlastního enginu upustila a aktuálně využívá jako procesor aplikaci RServe, serverový wrapper jazyka R.

1.1 Software pro statistickou analýzu

V roce 1990 proběhla na konferenci COMPSTAT-92 v tehdy ještě jugoslávském Dubrovníku panelová diskuze na téma budoucnost statistického softwaru (3). Po více než třiceti letech může působit až neuvěřitelně, jak přesně se diskutující trefili do dodnes plně neuspokojených přání uživatelů. Účastníci tehdy usoudili, že budoucí aplikace by měly disponovat mj. výjimečným grafických rozhraním a robustním systémem nápovědy, včetně tutoriálů. Jak uvidíme v průběhu této práce, jedná se o témata opakovaně poptávaná a opakovaně vykazující rezervy a tudíž prostor k uspokojení poptávky.

1.1.1 Historie

Ronald Fisher (1890-1962), jedna z klíčových postav statistické analýzy, začínal svou kariéru pouze s tužkou a papírem, nicméně již počátkem 20. let minulého století si za 200 britských liber pořídil první kalkulačku (4). Z dnešního pohledu byl následný rozvoj tehdejší výpočetní techniky natolik nepatrný, že teprve v době Fisherova penzijního věku se začaly objevovat první počítače společností Elliot a IBM, které znamenaly významný krok i v provádění statistické analýzy. Poptávka po zefektivnění práce statistiků se viditelně projevila prvně ve Velké Británii a USA, kde zájem přicházel od zpracovatelů zemědělských dat, ale i vědců ze společensky orientovaných oborů.

Vezmeme-li v úvahu délku životního cyklu dnes dobře známých systémů jako jsou Windows, iOS či Excel (nejsou dosud uzavřeny), ve skutečnosti se jedná o poměrně mladé produkty – alespoň oproti dodnes populárnímu **SPSS**, jehož první verze vznikla roku **1968**. V téže době vznikly i další dvě aplikace, dodnes podporované **SAS** a **GenStat**. Krátce nato se do povědomí (zejména akademické obce) dostal jazyk **S**, který byl populární zejména v 90. letech minulého století, než jej nahradila open-source alternativa **R** novozélandských autorů Ihaky a Gentlemana (5).

Vznik a rozvoj dalších desítek aplikací pro statistickou analýzu byl podmíněn (a akcelerován) několika faktory. Jednak šlo o postupně rostoucí počet dat, která do analýzy vstupují. Za druhé, změnil se způsob, jakým jsou data získávána (př. sociální sítě) a jak k nim lze přistupovat. Za třetí se změnily nároky na časové parametry vrácení výsledků, neboť pro některé pokročilé systémy (př. e-shopy) je statistická analýzy, prováděná v reálném čase, neodmyslitelnou součástí jejich úspěšného a udržitelného fungování. Za čtvrté (a zdaleka ne konečně) se konkurenční síly projevily v obchodních modelech: se stále běžnější dostupností bezplatných softwarů všeho druhu se rozšířila i nabídka statistických softwarů, jako je holandský **Jasp**, a zejména pak open-source řešení, které zastupuje i předmět této práce.

1.1.2 Současná řešení

Aplikace pro statistickou analýzu lze klasifikovat dle řady kritérií, např. dle statistických funkcí (jejich šíře a kvality zpracování), dle účelu využití, dle typické cílové skupiny a její velikosti či dle architektury (lokální x serverová, operační systém apod.). Vzhledem k tomu, jak je tento segment starý, je i vnitřně diferencovaný a málokterá očekávání uživatelů zůstávají opomenuta, jakkoliv je to s individuálními kombinacemi přání poněkud složitější. Ačkoliv do

rámce nástrojů pro statistickou analýzu patří i řešení pro data mining a big data (IBM Watson Studio, Rapidminer ad.[4]), tato práce se jimi blíže nezabývá, neboť se z mnoha úhlů pohledu vymykají dříve popsanému srozumitelnému opisu.

Jednou z možností, jak systematicky přistoupit k selektivnímu popisu statistických softwarů, je zvážit počet jejich klientů, resp. jejich utilizaci. Výrobci ohledně těchto statistik příliš sdílní nejsou, nicméně proběhlo několik pokusů o to využívání odhadnout nepřímo. Muenchen ve své studii (6) analyzoval profily potenciálních zaměstnanců v oboru statistika na serveru pracovních nabídek indeed.com; vyplynulo, že zájemci nejčastěji ovládali SAS (N = 1040), jazyk R (N = 1012), Stata (N = 176) a SPSS (N = 146)[5]. Jiný tým autorů provedl (7) metaanalýzu, kde sledoval na vzorku 10 596 biomedicinských studií, publikovaných mezi lety 1997 až 2017 na platformě PubMed.com, zmínky o užitém statistickém softwaru. Z jejich zjištění vyplývá, že zdaleka nejčastějším řešením v oboru bylo SPSS (52 % studií), následované SAS (13 %), Stata (13 %), Excelem (10 %) a jazykem R (9 %), přičemž SPSS bylo na počátku i konci sledovaného období volbou číslo jedna, a dvounásobný nárůst zaznamenalo mezi lety 1997 a 2007. Ačkoliv obě dvě studie zkoumaly kvalitativně odlišené vzorky, výčet hlavních nástrojů je u nich podobný. Výzkum provedený před nedávnem na českých univerzitách (8) na vzorku 763 respondentů tvrdí, že nejčastěji používanou aplikací u nás je Excel (53 %), s odstupem následovaný aplikací Statistica (19 %), SPSS (13 %) a R (10 %); autorky krom toho došly ke zjištění, že je běžnou praxí, že s analýzou pomáhá další osoba. Výsledky průzkumu jsou cenné mj. z toho důvodu, že pro aplikaci Retusa bude v případě produkční fáze klíčový domácí trh (konkrétně vysoké školy), který se od zahraniční trhů dříve popisovaných liší. Kromě výše uvedeného zájmu se autorky soustředily i na bariéry ve studiu a používání statistiky, kde shluková analýza identifikovala 3 klastry ze čtyř, kde studenti vykazují potenciál změnit svůj odmítavý či neutrální postoj ke statistice, pokud k tomu budou mít vhodné podmínky (např. odpovídající software).

Tři výše uvedené studie u různých oblastí se do značné míry shodují ve využívání konkrétních aplikací, byť jak bylo zřejmé, se zcela odlišnou vahou. Z pohledu případné komercializace a využití aplikací Retusa je smysluplné se více zabývat zejména řešeními uvedenými v poslední, domácí studii.

1.1.2.1 Excel

Excel kancelářského balíčku Office od Microsoftu (případně jeho různé alternativy, jako je Google Spreadsheets apod.) je natolik běžnou součástí většiny osobních počítačů (ale i dalších zařízení s jiným operačních systémem než Windows), že jej patrně není třeba obecně představovat. Oblastí jeho využití je nespočítatelně, což je do značné míry dáno i jeho poměrně velkou tolerancí vůči uživatelských návykům (např. že je velkou částí uživatelů používán pouze jako editor pro formátování tabulek). Funkcionality Excelu jsou přitom rozsáhlé, od editace dat přes jejich zobrazování formou tabulek a grafů, s možnostmi napojení na další (třeba databázové) zdroje a s podporou doplňků (tzv. *add-ons*) a vlastních skriptů ve VBA. Samozřejmou součástí Excelu jsou i funkce, které se definují v buňkách a mohou

[4] Existuje také řada privátních řešení, která si velké firmy nechávají postavit na klíč pro své specifické potřeby.

[5] Autor neuvádí velikost vzorku a nelze tudíž odvodit relativní podíly (předpokládejme, že respondenti mohli vyplnit víc než jednu odpověď).

konzumovat celou řadu argumentů. Část těchto funkcí je přímo určena pro statistickou analýzu.

Pro celou řadu případů je Excel dostačujícím řešením pro zpracování statistických úloh, ať sekvenčním výpočtem (za pomoci statistických funkcí buněk či bez nich), nebo pomocí doplňků, jako je bezplatná ***Analýza dat***, nabízející mj. výpočet vícenásobné regrese nebo dvou-faktorové analýzy rozptylu, tak doplňků třetích stran, jako je Xlstat[6], který již ve svém základním modulu nabízí například analýzu hlavních komponent. Podpora všech nástrojů balíčku Office je znatelná v prostředí Visual Studio, kde lze integrovat různé systémy (např. včetně jazyka R) a dosahovat výrazně vyšší efektivity vývoje, než je tomu při použití samotného VBA.

Vzhledem ke zjištěním na českých vysokých školách (8) lze navrhnout několik možných důvodů, proč je Excel převažujícím řešení. Jedním z nich je cena na uživatele, která je i díky množstevním slevám licencí u větších institucí oproti dalším placeným službám zanedbatelná. Dalším faktorem může být obecná elementární znalost prostředí napříč populací, kdy odpadá nutnost vysvětlovat základy obsluhy. Třetím faktorem může být pedagogický záměr: zatímco v aplikacích jako SPSS nebo Statistica potřebujeme pouze vědět, jaké výsledky chceme získat a k těm se následně pomocí několika kliknutí dostaneme, vyžaduje Excel (pokud nepoužíváme doplňky) daleko větší zapojení uživatele. Další otázkou zůstává, proč je – minimálně u nás – využívání doplňků do značné míry přehlížené. Náklady na roční licenci balíčku Premium výše uvedeného doplňku Xlstat se po slevě pohybují okolo 715 USD za rok, což je při obsahu funkcionalit patrně konkurenceschopná nabídka, ve srovnání se specializovanými softwary. Při volbě placených produktů přesto zákazníci zjevně preferují dražší řešení. Skutečný důvod je třeba zodpovědět pomocí empirických metod, nicméně se nabízí, že obecně nemají doplňky takovou důvěru jako kompaktní řešení, nejen u přímých klientů (uživatelů), ale i širší populace (příjemců analytických výstupů).

Ve vztahu k vývoji aplikací Retusa je Excel prototypem a vzorem pro design celé řady prvků v dalších aplikacích (tabulkové rozhraní, funkce v buňkách atd.), neboť nejen že se jedná o důkladně otestovaná řešení, ale zároveň řešení velmi dobře známá, která budou uživatelé očekávat. Z tohoto důvody byly mnohé prvky Excelu zohledněny jak v Node.js aplikaci (např. API distribučních funkcí), tak zejména v aplikaci pro prohlížeč.

1.1.2.2 SPSS

SPSS, celým názvem IBM SPSS Statistics, je jedno z nejstarších a dodnes nejpoužívanějších řešení pro statistickou analýzu. Jedná se o (mj. z hlediska obchodního modelu) modulární systém či pracovní prostředí (framework), do kterého si uživatel vybírá a nakupuje jednotlivé balíčky funkcionalit (regrese, neuronové sítě, conjoint analýzu atd.). SPSS je součástí širšího ekosystému, který se snaží pokrýt komplexní potřeby zákazníků, např. proces od sběru odpovědí respondentů (PS Quaestio PRO), přes jejich zpracování (SPSS) až po jejich reportování na straně klienta (PS Imago PRO).

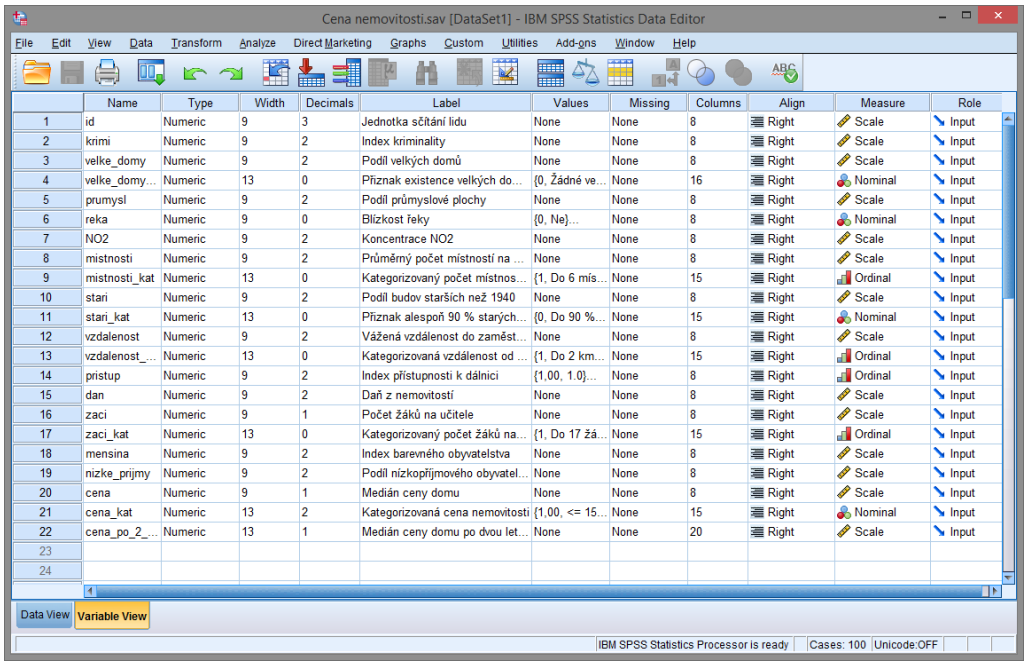
Funkcionality na straně SPSS lze rozdělit do několika skupin, přičemž záleží, jaké moduly má uživatel k dispozici. Pevnou součástí aplikace je tabulkové rozhraní podobné Excelu, kde lze

[6] https://www.xlstat.com/en/

definovat proměnné a ručně upravovat data. S ním souvisí i panely k další manipulaci s daty (filtrování, čištění, vážení apod.). Menu analýzy nabízí (v závislosti na pořízených modulech) variabilní množství statistických a dalších metod, roztříděných do smysluplných kategorií (porovnávání průměrů, regresní modely, redukce proměnných apod.) a výstupy jednotlivých metod jsou komplexní, tzn. že zahrnují post-testy, různé doplňující statistiky, tabulky aj. Součástí výstupů a též uživatelského prostředí je tvorba a generování grafů. Důležitou vlastností SPSS je možnost automatizace pomocí skriptování (SPSS syntax). Tento výčet funkcionalit není vyčerpávající a obecně platí, že potřeby uživatelů v SPSS se zřídkakdy dostanou na hranici možností samotné aplikace.

U nás a na Slovensku zastupuje SPSS společnost Acrea CR (resp. SR), která vedle prodeje zajišťuje širokou škálu služeb, od konzultací přes školení až po ad hoc extenze i systematické úpravy.

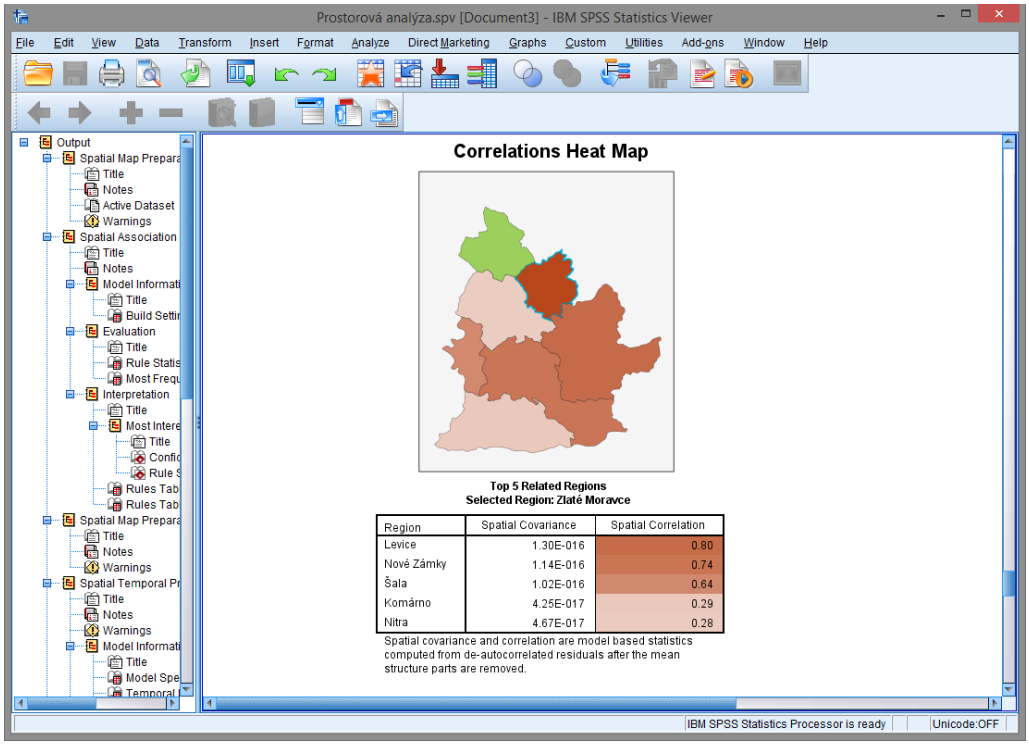
OBRÁZEK 1: SPSS - ROZHRANÍ TABULKY PRO SPECIFIKACI PROMĚNNÝCH[7]



	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure	Role
1	id	Numeric	9	3	Jednotka sčítání lidu	None	None	8	Right	Scale	Input
2	krimi	Numeric	9	2	Index kriminality	None	None	8	Right	Scale	Input
3	velke_domy	Numeric	9	2	Podíl velkých domů	None	None	8	Right	Scale	Input
4	velke_domy...	Numeric	13	0	Příznak existence velkých do...	{0, Žádné ve...	None	16	Right	Nominal	Input
5	prumysl	Numeric	9	2	Podíl průmyslové plochy	None	None	8	Right	Scale	Input
6	reka	Numeric	9	0	Blízkost řeky	{0, Ne}...	None	8	Right	Nominal	Input
7	NO2	Numeric	9	2	Koncentrace NO2	None	None	8	Right	Scale	Input
8	mistnosti	Numeric	9	2	Průměrný počet místností na ...	None	None	8	Right	Scale	Input
9	mistnosti_kat	Numeric	13	0	Kategorizovaný počet místnos...	{1, Do 6 mís...	None	15	Right	Ordinal	Input
10	stari	Numeric	9	2	Podíl budov starších než 1940	None	None	8	Right	Scale	Input
11	stari_kat	Numeric	13	0	Příznak alespoň 90 % starých...	{0, Do 90 %...	None	15	Right	Nominal	Input
12	vzdalenost	Numeric	9	2	Vážená vzdálenost do zaměst...	None	None	8	Right	Scale	Input
13	vzdalenost_...	Numeric	13	0	Kategorizovaná vzdálenost od ...	{1, Do 2 km...	None	15	Right	Ordinal	Input
14	pristup	Numeric	9	2	Index přístupnosti k dálnici	{1,00, 1,0}...	None	8	Right	Ordinal	Input
15	dan	Numeric	9	2	Daň z nemovitostí	None	None	8	Right	Scale	Input
16	zaci	Numeric	9	1	Počet žáků na učitele	None	None	8	Right	Scale	Input
17	zaci_kat	Numeric	13	0	Kategorizovaný počet žáků na...	{1, Do 17 žá...	None	15	Right	Ordinal	Input
18	mensina	Numeric	9	2	Index barevného obyvatelstva	None	None	8	Right	Scale	Input
19	nizke_prijmy	Numeric	9	2	Podíl nízkopříjmového obyvatel...	None	None	8	Right	Scale	Input
20	cena	Numeric	9	1	Medián ceny domu	None	None	8	Right	Scale	Input
21	cena_kat	Numeric	13	2	Kategorizovaná cena nemovitosti	{1,00, <= 15...	None	15	Right	Nominal	Input
22	cena_po_2_...	Numeric	13	1	Medián ceny domu po dvou let...	None	None	20	Right	Scale	Input
23											
24											

[7] Zdroj: https://acrea.cz/software/ibm-spss-statistics/

OBRÁZEK 2: SPSS - PŘÍKLAD ANALYTICKÉHO VÝSTUPU[8]



1.1.2.3 Statistica

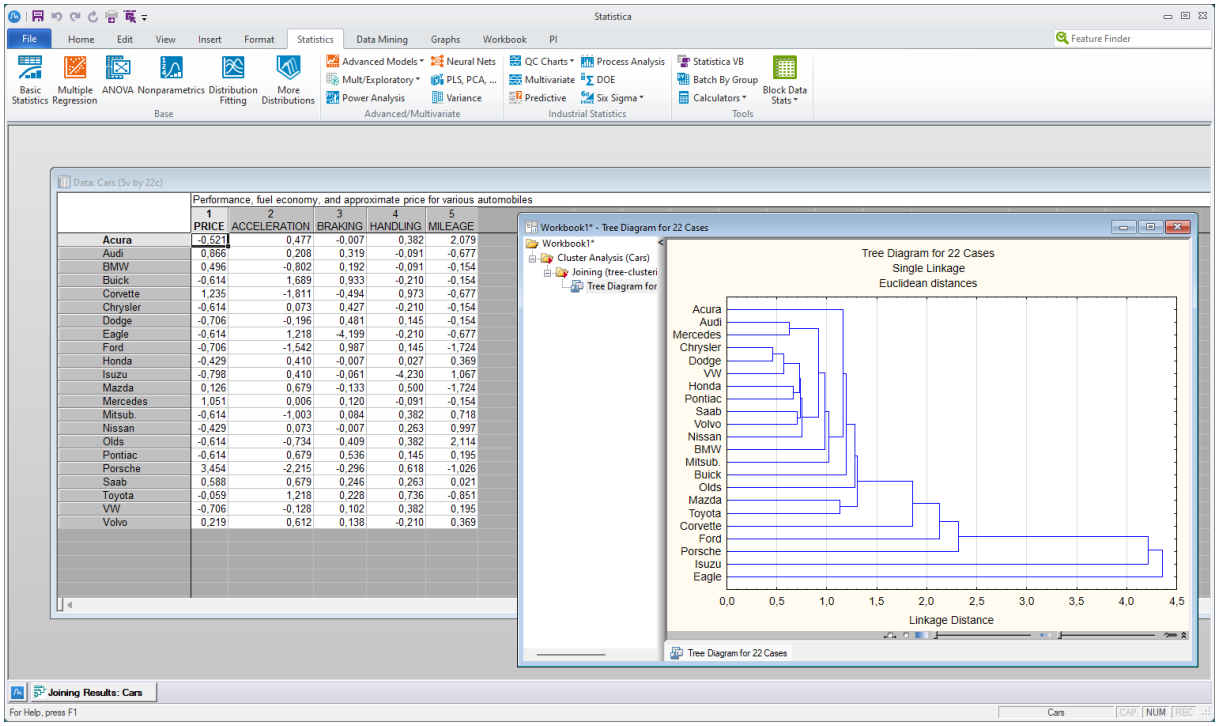
Ve výčtu nejčastěji používaných řešení v České republice je TIBCO Statistica nejbližší alternativou SPSS, zejména ve smyslu grafického rozhraní i způsobů obsluhy. Oproti SPSS využívá Statistica v posledních několika verzích (minimálně od verze 13) místo svého vlastního enginu procesor jazyka R (konkrétně modul Rserve), podobně jako Jasp, což svým způsobem potvrzuje revoluční přínos R pro statistickou analýzu – buďto ve smyslu nákladů (je levnější integrovat R, nežli si vyvíjet a udržovat vlastní procesor), nebo s ohledem na kvalitu zpracování. Podobně jako SPSS, i Statistica umožňuje uživatelské skriptování, buďto pomocí VBA, nebo přímo pomocí jazyka R.

S ohledem na podobnosti je případná uživatelská migrace mezi těmito dvěma řešeními nejsnazší a dle mé osobní zkušenosti převážně intuitivní. Statistica mj. podporuje čtení a zápis formátu *sav* (SPSS). Z pohledu škály statistických funkcí se obě dvě srovnávaná řešení mírně liší, nicméně základní výbava testů a vícerozměrných metod je velmi podobná.

V České republice má Statistica dlouhodobě obchodní zastoupení, aktuálně společností DataFriends s.r.o., která poskytuje podporu ve formě vzdělávacích programů či helpdesku.

[8] Zdroj: https://acrea.cz/software/ibm-spss-statistics/

OBRÁZEK 3: STATISTICA (VERZE 14) - PŘÍKLAD VSTUPU A VÝSTUPU SHLUKOVÉ ANALÝZY

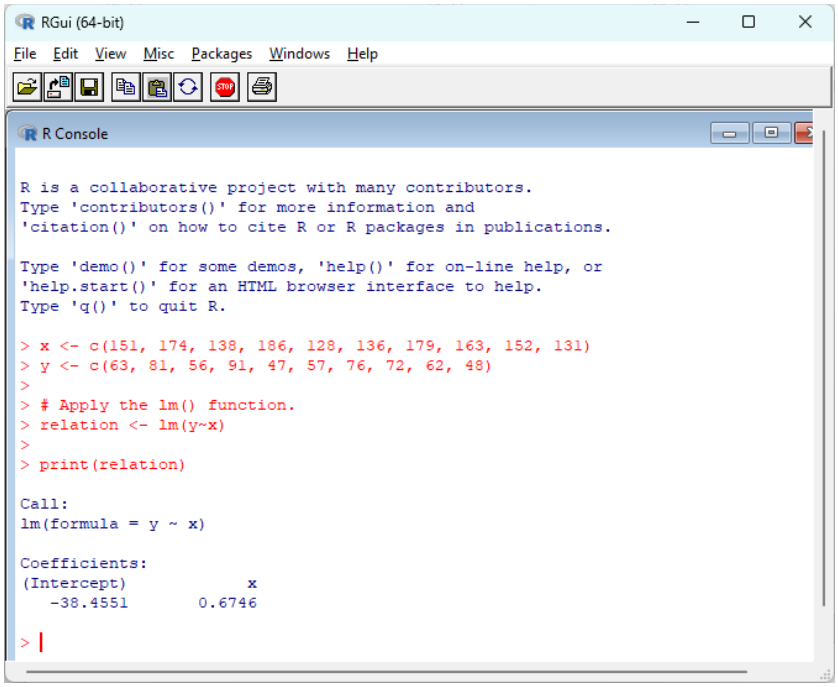


1.1.2.4 R

Zcela odlišný přístup ke statistické analýze představuje jazyk R, který vznikl v roce 1993 na Aucklanské univerzitě a jak název naznačuje, jedná se o programovací jazyk. Předchůdcem R je dříve zmíněný jazyk S, kterým byl R zásadně ovlivněn, do té míry, že dokáže zpracovat většinu syntaxe S. Je důležité upozornit, že na rozdíl od obsluhy dříve uvedených nástrojů je využívání R podmíněno osvojením si tohoto jazyka a jeho syntaxe, což je pro běžného uživatele nepoměrně náročnější. Navzdory tomu je popularita R neustále na vzestupu a využívá se v řadě oborů a pro řadu účelů. Nedávný průzkum (9) naznačuje, že v rámci akademických statistických analýz dochází od roku 2010 k systematickému růstu jeho využívání, a to v jednoznačný neprospěch někdejších favoritů SPSS a SAS, což naznačují i výpovědi respondentů prováděných za účelem testování aplikace Retusa (viz dále).

Z hlediska syntaxe je R objektově-orientovaný a interpretovaný jazyk, ke kterému lze přistupovat několika způsoby. Typické je využívání konzole RGui (viz příklad níže), do níž uživatel zadává argumenty a volá metody, kterých jazyk - i bez extenzí - obsahuje velké množství. Jazyk R lze ovšem využívat či volat i pomocí různých rozšíření. Příkladem, pro různé programovací jazyky existují tzv. wrappery, které R „zabalí“ do vlastního API a tak je zprostředkují vývojovým prostředím třetích stran (existuje i wrapper pro Node.js). Kromě toho je R schopno generovat statické grafy, které se vytváří zadáním příkazu a vykreslují se ve vlastním okně konzole (používáme-li RGui).

OBRÁZEK 4: RGUI - ROZHRANÍ KONZOLE S VÝPOČTEM KONSTANTY A KOEFICIENTU LINEÁRNÍ REGRESE



1.1.3 Node.js a statistické knihovny

Pro tuto práci je, vedle dříve popisovaných aplikací, stěžejní ještě jedna entita, a tou je **Node.js**. Jedná se o open-source softwarový systém, vyvinutý Raynem Dahlem a poprvé publikovaný v roce 2009. Systém běží na enginu **V8** od společnosti Google, stejném systému, na kterém funguje např. webový prohlížeč Chrome; jedná se o javascriptový runtime implementující zásady ECMAScript a WebAssembly, který byl vyvinut v jazyku C++ a je kompatibilní s novějšími verzemi operačních systémů Windows, macOS a některými verzemi Linux (10). Poslední stabilní verze systému Node.js byla v době vzniku tohoto textu (březen 2023) 19.8.1 (11).

Node.js je systém určený především pro **vývoj dynamických webových aplikací**, dle projektu W3Techs jej využívá 2 % všech jemu známých webových stránek[9], přičemž v roce 2015 to bylo pouze 0,1 % (12). Systém podporuje asynchronní zpracování HTTP požadavků v objemu tisícovek paralelních spojení, aniž by pro každé musel vytvářet vlastní vlákno (thread) – tím se liší od jiného populárního systému, PHP, který požadavky zpracovává paralelně (13). Díky tomu, že jazykem Node.js je JavaScript, otevírá se svět serverových aplikací milionům front-end vývojářům, pro které je přechod díky stejné syntaxi nesrovnatelně snazší než z jiných jazyků (14), což vede ke zjednodušení full-stack vývoje. Kromě toho, v porovnání s jinými systémy, jako je IIS nebo Apache, je konfigurace Node.js podstatně jednodušší (15). Specifikem systému je i to, že jeho knihovny nejsou distribuovány jako kompiláty ve strojovém kódu, nýbrž jako původní javascriptové balíčky (moduly). To znamená, že se libovolný uživatel dle potřeby dostane ke kterékoliv části kódu, pokud není nějak jinak chráněn.

[9] Mezi jeho nejznámější klienty patří Twitter, Spotify nebo Netflix, pro posledního jmenovaného je to systém páteřní (12).

To, že systém Node.js běží na stejném runtimu jako prohlížeč Chrome, znamená, že mnohé funkcionality JavaScriptu budou stejné a dostupné jak na straně serveru (back-end), tak v prohlížeči (front-end). Této vlastnosti je často využíváno, a to pomocí služeb jako je Webpack (16), knihovny, která kompiluje moduly Node.js do podoby vhodné pro běžné internetové prohlížeče s podporou JavaScriptu.

JavaScript a tudíž i Node.js mají řadu vestavěných typů, tříd, objektů a funkcí, které za běžného nároku na programování (ovšem většího, než v případě R) mohou provádět nejrůznější výpočetní operace. Tato vlastnost je samozřejmá u většiny jazyků a právě tak jako v jiných jazycích, i u Node.js nalezneme řadu knihoven, které se zaměřují přímo na některou oblast statistické analýzy. Jelikož jsou tyto knihovny z podstaty open-source, lze jejich řešení a obsah při dostatku času analyzovat do nejmenšího detailu a definovat jejich vhodné a nevhodné postupy. Protože těchto knihoven je mnoho desítek (většinou velice úzce zaměřených), zaměřuje se následující přehled pouze na ty, které jsou prokazatelně používané dalšími uživateli (na základě statistik GitHub) a které zároveň představují komplexní řešení.

Simple Statistics (17) je knihovna, která obsahuje kromě deskriptivních funkcí také metodu pro výpočet lineární regrese a několik málo dalších metod (např. jedno a dvou výběrový t-test), nicméně je orientovaná zejména právě na popisné statistiky. Oproti tomu knihovna **jStat** (18) nabízí vedle algebraických funkcí a menší nabídky statistických testů (např. ANOVA), robustní modul pro výpočet distribučních funkcí, které jsou přednostně implementovány (dle pozorování autora) při generování skriptů na straně chatGPT (viz dále). Právě ve smyslu rozsáhlosti i logiky struktury zpracování distribučních funkcí je jStat nejpokročilejší mezi sledovanými knihovnami pro Node.js. Co se týče statistických metod, je API knihovny postaveno tak, že vrací jen jednoduché (číselné) výsledky, takže např. pro výpočet statistik potřebných k popisu modelu analýzy rozptylu je potřeba volat každou potřeba proceduru zvlášť (pro výpočet F testu, p-hodnoty apod.). Kdysi populární knihovna **science.js** (19), jejíž poslední aktualizace proběhla před osmi lety, nabízí kromě řady funkcionalit uvedených v předchozích řešeních i pokročilejší metody, jako je logistická regrese. Balíček **Jerzy** (20), rovněž řadu let neaktualizovaný a od roku 2020 uzavřený, je orientován přímo na statistickou analýzu, v podobném smyslu, jako aplikace Retusa, popisovaná dále v této práci; kromě distribučních funkcí nabízí i několik testů (T-test, ANOVA, Pearsonova korelace, lineární regrese aj.), přičemž výsledky testů zpracovává komplexně (oproti jStat), tedy že na volání funkce vrací protokol metody (testy, p-hodnoty, popisné statistiky apod.) ve formátu objektu[10]. Stále podporovaná, ale šest let neaktualizovaná knihovna **statistics.js** (21) je architektonicky i účelově velmi podobná oběma výše uvedeným metodám, s tím, že kromě dříve uvedených metod obsahuje i několik neparametrických testů. Kromě toho je u ní specifickým předpokladem definování typu proměnných (nominální, ordinální, spojitá a metrická), které se u ostatních (zde probíraných) Node.js knihoven neobjevuje.

Na platformě GitHub lze nalézt mnoho dalších knihoven, které se buď přímo, nebo nepřímo věnují statistické analýze, přičemž v řadě případů se jedná o testovací řešení zaměřené např. na jedinou metodu – nejedná se tedy o systematické a komplexní řešení pro širší využití. Vedle nich stojí za pozornost projekt **EZ Statistics** (22), který je ve smyslu obsahu nejpodobnější dále

[10] Vzhledem k poměrně malé známosti/oblibě jsem tuto knihovnu objevil až po publikování obou aplikací Retusa. Je zajímavé, že obě dvě na sobě zcela nezávislá řešení (Jerzy a Retusa) přistupují k řešení konkrétních problémů identickým způsobem, např. konstrukce vektorů probíhá iniciací třídy, statistické metody vrací komplexní odpovědi apod.

řešené aplikaci Retusa GUI. Na rozdíl od dříve popisovaných řešení je koncipován jako front-end aplikace (s neoddělitelným výpočetním enginem) pro statistické testování přímo v oknu prohlížeče a nabízí systematicky utříděné parametrické a neparametrické testy pro porovnávání rozdílů (resp. centrality) u nezávislých i závislých vzorků a také dialog pro výpočet statistik lineární regrese či Pearsonova korelačního koeficientu. Aplikace nemá tabulkové rozhraní pro zadávání dat do matice a místo toho nabízí element typu input, do kterého uživatel zadává data oddělená čárkou. Výstupem metod jsou kromě testových hodnot i post-testy, grafy a také slovní interpretace výsledků. Aplikace je dle dostupných informací de facto neznámá, na druhou stranu je třeba zdůraznit, že se (ať úmyslně či náhodou) strefila nejen do dříve uvedené vize diskutujících na konferenci COMPSTAT-90 (3), ale i do potřeb, které zmiňovali respondenti během testování aplikace Retusa GUI – totiž že aplikace pro statistickou analýzu by měly umět výsledky srozumitelně vysvětlit.

TABULKA 1: UŽIVATELSKÉ STATISTIKY KNIHOVEN PRO STATISTICKOU ANALÝZU NA GITHUB.COM

Název knihovny	Počet uživatelů	Počet přispěvatelů	Obliba	URL
Simple Statistics	11 000	51	2 900	http://simple-statistics.github.io/
jStat	3 100	50	1 700	http://jstat.github.io/
science.js	83	7	875	https://github.com/jasondavies/science.js
Jerzy	-	2	40	https://github.com/pieterprovoost/jerzy
statistics.js	-	1	104	https://thisancog.github.io/statistics.js/
EZ statistics	-	1	1	https://github.com/jhagelback/ez_statistics

1.2 Shrnutí

Desítky let pokračující vývoj statistických softwarů dnes reprezentuje řada různých řešení, určených pro dílčí systémy, cílové skupiny či případy a využívané v desítkách či stovkách tisíců analytických projektů rok co rok. Trh těchto aplikací je, ve srovnání s jinými IT kategoriemi a jejich hráči, stabilní a de facto jediný náznak vychýlení mnohaletého kurzu trhu je příchod jazyka R, který narušil nepsané paradigma této kategorie, totiž že daný software nemusí být přístupný každému a zároveň že pokud je neortodoxní řešení dostatečně zajímavé, dokáže motivovat své uživatele k náročnějšímu studiu obsluhy. Pro řadu oblíbených aplikací (opět ve srovnání s dalšími kategoriemi) je specifický konzervativní vývoj – řada předních řešení vypadá a do značné míry i funguje mnoho let stejně. Ačkoliv výrobci běžně nezveřejňují metodiku vývoje aplikací coby uživatelského produktu, některé dříve uvedené studie a také rozhovory provedené pro tuto práci naznačují, že dnes klíčová koncentrace na potřeby zákazníka (23) nemusí být u tradičních výrobců prioritou. Statistika, jak uvádí nejen písnička, ale i empirická studie (8), není obor příliš oblíbený a při současné nabídce softwarů ji nelze provádět ani se znalostmi ze středoškolské matematiky; na provedení obstojné analýzy s efektivním vytěžením dat je potřeba nejen pokročilejších znalostí, ale i zkušeností.

Následující část práce pojednává o vývoji dvou aplikací, jejichž návrh a směřování navazují na zjištění uvedená v této kapitole. První aplikace, Retusa, která ze zadání této práce splňuje slovo Node.js, je paralelou k jazyku R – nikoliv v tom smyslu, že by představovala vlastní jazyk (jejím jazykem je JavaScript), ale spíše ve smyslu procesoru, který funguje podobně jako zmíněný jazyk. Druhá aplikace, nazvaná Retusa GUI, je zmíněným „SPSS v prohlížeči“ – její úlohou je zprostředkovat funkce první aplikace libovolnému uživateli (tedy i tomu, který vůbec

neumí programovat), a to při patrně nejmenších možných nárocích – pouze za pomoci internetového prohlížeče, bez potřeby jakékoliv instalace.

2 Aplikace

Klíčovým výstupem této práce jsou dvě aplikace: Retusa a Retusa GUI[11]. Jelikož je důležité obě dvě důsledně rozlišovat, zopakujme, že **Retusa** (zkráceně **RTS**) je back-end knihovnou pro výpočet statistických metod v prostředí Node.js. **Retusa GUI** (zkráceně **RUI**) je front-end aplikace pro webové prohlížeče, která slouží zejména jako svrchní vrstva pro RTS a umožňuje běžnému uživateli využívat funkcionalit RTS.

2.1 Retusa

Aplikace Retusa byla vyvinuta jako synchronní back-end (serverová) knihovna pro statistickou analýzu a odvislé operace, přičemž od samého začátku se předpokládalo i její využití na straně front-end aplikací, typicky v internetovém prohlížeči jako pomocné knihovny. Tento předpoklad měl za důsledek, že samotná RTS neobsahuje žádné závislosti na knihovnách třetích stran, a to s ohledem na potřebu a) kontrolovat optimální velikost cílové komprimované knihovny, b) minimalizovat riziko pramenící ze závislosti na cizích knihovnách a c) potřebě kontrolovat veškeré dílčí procesy, které zajišťují klíčové služby aplikace. Díky tomu má komprimovaná knihovna (soubor *bundle.min.js*) ve verzi 1.0.4 (k 21. březnu 2023) velikost pouhých 143 kB, a to včetně dvou jazykových mutací.

Zdrojový kód aplikace je veřejně přístupný na úložišti GitHub, a to ve třech verzích (neboli větvích, anglicky *branche*):

- **main:** hlavní (master) větev
- **dev:** vývojová větev, sloužící zejména k zálohování větve na lokálním uložišti a testování
- **cdn:** větev sloužící k distribuci přes CDN (content delivery network); aktuálně nevyužívaná

URL knihovny: https://github.com/muzikp/retusa/

2.1.1 Vývoj

Vývoj aplikace ve smyslu ontogeneze se řídil dvěma základními principy, a to **MVP** (minimum viable product) a analýzou existujících řešení (viz kapitolu Východisko). Myšlenka MVP (23), úzce spojená s agilním řízením, předpokládá, že k ověření potenciálu produktu na straně zákazníka stačí prezentovat minimální skupinu nepostradatelných vlastností daného produktu. Princip MVP se v rámci vývoje uplatnil v tom smysl, že většina prvků byla vyvíjena s maximálním ohledem na svou využitelnost a další komponenty byly přidávaný podle téhož

[11] GUI je zkratka pro Graphic User Interface, tedy grafické uživatelské rozhraní.

kritéria. Příkladem, metoda *correlPearson*, která vrací protokol Pearsonova korelačního koeficintu, byla nejdříve narychlo naprogramovaná bez jakékoliv validace, dokumentace apod. Po otestování byla pro prototyp funkce postavena definice metody (určující parametr třídy *MatrixAnalysis*), poté definice argumentů, výstupů a nakonec dokumentace, včetně lokálních mutací.

Analýza existujících řešení vycházela zejména s předchozího testování a identifikace entit (procesů, struktur apod.) dalších aplikací popsaných dříve. Postup spočíval v praktickém zkoušení existujících řešení a sledování, které prvky jsou následováníhodné, které vykazují nedostatky spojené s možnou inovací a také prvky, které byly přímo nevhodné. Příkladem, jako ideální (z hlediska vývoje i kvality řešení) se ukázalo implementovat distribuční funkce z knihovny jStat[12] (18), naopak jako nedostatečná se ukázala omezená konfigurovatelnost argumentů u metod typu ANOVA v SPSS či Statistica, která byla v RTS provedena formou flexibilnějšího API. Těchto postupů bylo v rámci vývoj použito několik desítek. Srovnávána nebyla pouze řešení v Node.js, ale obecně nástroje určené ke statistické analýze, jelikož architektura řešení RTS byla o počátku navržena tak, aby ji bylo možno využívat v řešeních třetích stran (jak v back-end prostředí, tak v prohlížeči). Obecně lze říci, že kompaktní aplikace typu SPSS inspirovaly obsahem svých výstupů a omezeními (viz výše), řešení v Node.js pak společnými znaky[13] a v některých případech i postupy.

Na rozdíl od aplikace RUI nebylo v rámci vývoje využito uživatelské testování (s výjimkou důkladného testování při integraci v RUI), architektura řešení je nicméně v souladu s běžnými standardy programování v Node.js.

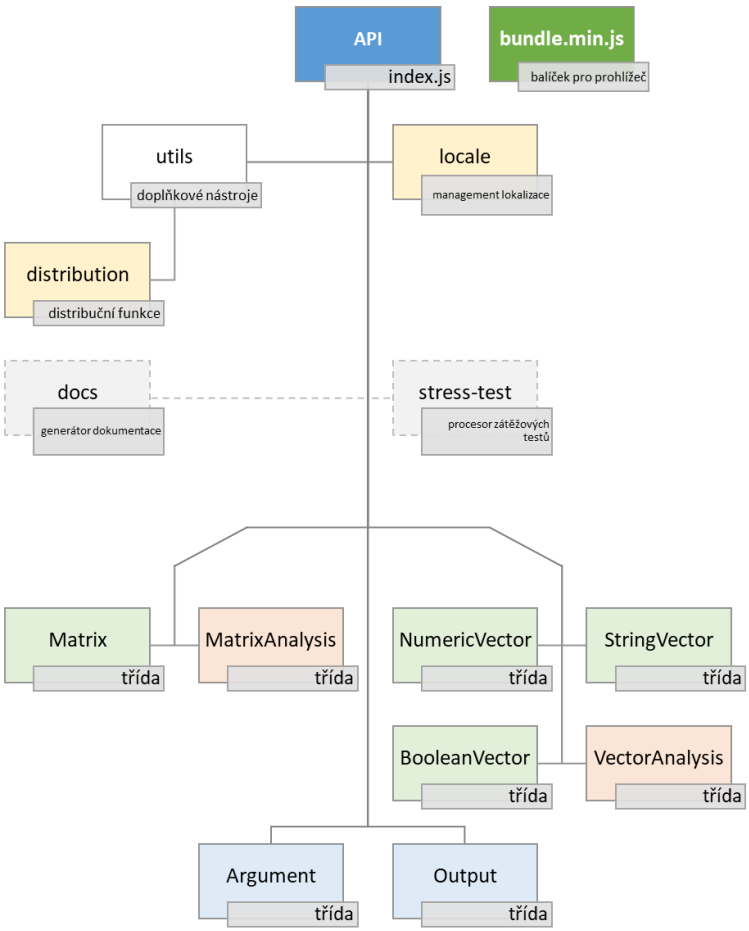
2.1.2 Struktura aplikace

V následujících částech této kapitoly je věnována pozornost jednotlivým funkcionalitám aplikace, nicméně bude užitečné alespoň stručně popsat strukturu rozhraní RTS jako API. To zprostředkovává přístup zejména k základním třídám a pomocným metodám a modulům, jak ukazuje schéma níže.

[12] Části scriptu byly zkopírovány do modulu distribution.js, kde byly mírně modifikovány tak, aby jejich API bylo co nejpodobnější rozhraní v MS Excel. Důvod, proč není knihovna zahrnuta do závislostí RTS (tzv. dependency) je ten, že výpočet rozdělení se v letech zásadně nemění a zároveň RTS z funkcí jStat využívá jen přesně ohraničenou výseč.

[13] Koncept tříd Vector a Matrix nacházíme u několika nejpoužívanějších knihoven (včetně jStat) a tato terminologie byla převzata do RTS z toho důvody, aby případná migrace mezi RTS a jinými aplikacemi byla pro uživatele co nejméně matoucí.

OBRÁZEK 5: RETUSA - STRUKTURA API ROZHRANÍ



Třídy *Matrix*, *NumericVector*, *StringVector* a *BooleanVector* slouží ve formě své instance jako datové kontejnery. Třídy *MatrixAnalysis* a *VectorAnalysis* fungují jako vlastní procesory analytických požadavků uživatele a kromě provádění výpočetních operací validují vstupní argumenty, zajišťují před-výpočetní zpracování dat a ukládají potřebné informace o výpočtu (vstupní/výstupní velikost vzorku, doba zpracování atd.). Třída *Argument* a potažmo celý modul, který ji publikuje, zprostředkovává rozhraní pro popis a validaci argumentů, třída *Output* zajišťuje zejména dokumentaci výstupů metod (strukturu a lokalizaci). Jedinou vlastností objektu *utils* je v současné době modul *distribution*, který obsahuje distribuční funkce, používané napříč výpočetními funkcemi. Modul *locale* zpracovává požadavky na lokalizované verze jazykových kódů. Celá struktura API se všemi funkcemi je pomocí služby webpack konvertována do souboru *bundle.min.js*, který je určen pro nasazení aplikace v internetovém prohlížeči. Kromě výše uvedených modulů jsou součástí aplikace ještě rozhraní pro provádění zátěžových testů (stress-test.js) a generování dokumentace (docs.js) do formátu markdown, ani jeden z nich není ovšem pro běžné využití podstatný.

2.1.3 Vektorové třídy

Ústředními třídami aplikace jsou vektor[14] a matice. V prvním případě se jedná o dědičné třídy mateřské třídy Array, v druhém případě o vektor druhého stupně, tedy vektor vektorů. Z pohledu koncového uživatele knihovny se nejedná o narušení konvenční syntaxe JavaScriptu, jelikož v konzoli třída Vector vypadá jako třída Array, neboť tuto třídu sama dědí. Rozdíly mezi entitami Vector a Array popisuje níže uvedená tabulka.

TABULKA 2: POROVNÁNÍ VLASTNOSTÍ TŘÍD ARRAY A VECTOR

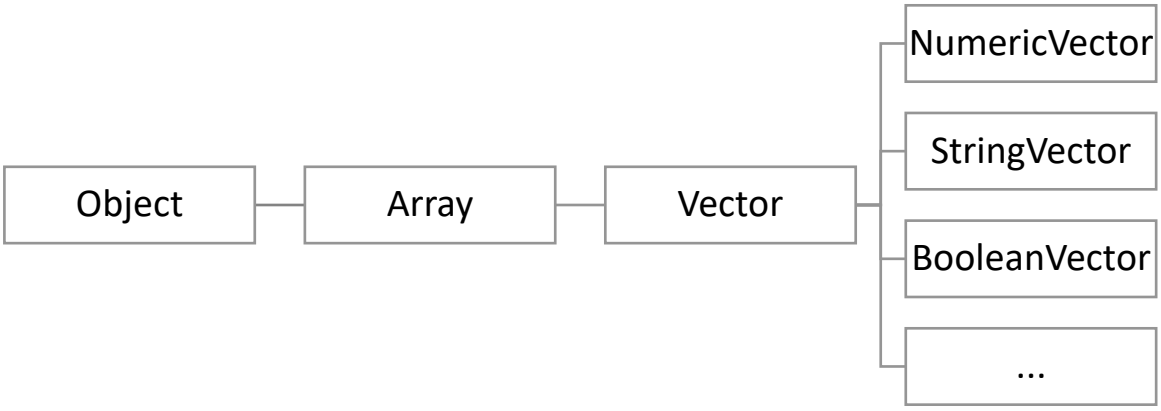
Array	**Vector**
Vlastnosti = hodnoty vektoru	Vlastnosti, které nemají tvořit hodnoty vektoru, je možné uchovávat pomocí funkcí do instance třídy WeakMap (např. název vektoru)
Do instance lze vložit libovolnou hodnotu	Do instance lze vložit pouze hodnotu akceptovatelnou validátorem třídy (např. u NumericVector pouze číselnou nebo prázdnou hodnotu)
Metody třídy jsou statické	Část (zejména statistických) metod je dynamicky generována; mj. lze modifikovat název metody, takže př. funkci sum() lze po vnější úpravě volat jako součet()

V modulu vector.js sice nalezneme samotnou třídu Vector, ta není ovšem vnějším uživatelům přístupná. Její úloha je mateřská (ve smyslu prototypu), a to vůči dědičným třídám NumericVector, StringVector a BooleanVector, které jsou – jak naznačuje jejich název – určeny pro práci s konkrétním typem hodnot[15] (číselným, textovým a binárním). Architektura tohoto větvení vektorů dle typů vychází z potřeby kontroly nad tím, jaké hodnoty uživatel vektorům připisuje, neboť jedním ze zásadních problémů dříve popsaných knihoven v Node.js je, že nedostatečně validují vstupy, což někdy vede k nesprávným výsledkům statistických metod.

[14] Pro běžné pochopení je místo termínu vektor (vector apod.) vhodnější používat pojmu proměnná, z hlediska konvence v podobných aplikacích je v této práci termín proměnná používán výjimečně.

[15] Pro aktuální operace jsou uvedené typy vektorů dostačující, nicméně v dalším vývoji se předpokládá jejich rozšiřování (např. TimeVector, MultipleVector apod.).

OBRÁZEK 6: RETUSA - SCHÉMA DĚDIČNOSTI VEKTOROVÝCH TŘÍD



Třída Vector definuje celou řadu metod popsaných dále, krom toho ale děděné vektory integrují i své vlastní, specifické metody, které jsou specifické pouze pro ně samotné.

2.1.3.1 Konstrukce vektoru

Instanci zděděných tříd mateřské třídy Vector (dále jen vektory) lze vytvořit několika různými způsoby, přičemž výsledek je vždy totožný (viz níže). Vektor lze vytvořit buďto běžným konstruktorem třídy (př. new NumericVector(…)), nebo metodou *vectorify*[16] (rozšíření třídy Array), která se pokusí sama rozpoznat typ vektorových hodnot a dle toho vytvoří odpovídající typ vektoru; eventuálně lze využít přesnějších metody *numerify*, *stringify* a *boolify*, které se pokusí převést řadu na konkrétní typ vektoru[17].

```
var score = new NumericVector(12, "", 3, 7, 6, 8, 9, 10);
var score = new NumericVector([12, undefined, 3, 7, 6, 8, 9, 10]);
var score = new NumericVector([12, false, 3, 7, 6], 8, 9, 10)
var score = [12, null, 3, 7, 6, 8, 9, 10].vectorify();
```

Pro konstrukci vektoru libovolného typu platí následující pravidla:

1. Argumenty lze zadávat buďto samostatně (oddělené čárkou; př. 1) nebo jako typ Array (př. 2)
2. Argumenty ve formátu vektoru (resp. Array) jsou rozloženy pomocí funkce *flat* na jednorozměrnou řadu (př. 3)
3. Obecně prázdné hodnoty (*null, undefined, NaN*) jsou převáděny na typ **null**, aby nedocházelo k zaměňování zdánlivě a skutečné prázdných hodnot[18]

Alternativou k výše uvedené a doporučené formě konstrukce vektoru je kmenová funkce API ***vector*** (s malým v na začátku). Je založena na implementaci rozšiřující metody ***vectorify*** třídy

[16] Metoda *vectorify* pracuje tak, že nejprve zjistí, zdali řada neobsahuje nic jiného, než hodnoty true, false nebo null. Pokud ne, vytvoří instance BooleanVector. Pokud obsahuje, zkusí vytvořit instanci NumericVector; pokud nelze, jako poslední možnost zvolí StringVector.

[17] Aktualizace RTS z 21.3.2023 zařadila do API funkce *vector* a *matrix*, které imitují konstrukci vektorů a matic u jStat a dalších řešení.

[18] V JavaScriptu je při komparaci druhého stupně (==) hodnota nula považována za prázdnou, což je v případě statistické analýzy zavádějící; proto je důsledně rozlišováno mezi nulou, nepravdou (false) a ostatními prázdnými hodnotami; v aplikace je jako prázdná hodnota používán výhradně typ null.

Array. Tato implementace vznikla s ohledem na dříve uvedený prvek, běžný v mnoha podobných řešení (jStat aj.).

```
var x = vector(5, 7, 8, 9, 4, 1, 2, null, 4, 7, 5, 6);
var xx = new NumericVector(5, 7, 8, 9, 4, 1, 2, null, 4, 7, 5, 6);
// x = xx
var y = vector("A", "B", "C");
var yy = new StringVector("A", "B", "C");
// y = yy
```

2.1.3.2 Validace hodnot

Principem různých typů vektorů je kontrola nad tím, jaké hodnoty smí nebo nesmí do vektoru vstupovat a následně jaké metody jsou pro daný typ vektory relevantní a jaké ne (např. součet u StringVector nedává smysl). Validace je zajištěna pomocí modifikace funkce **push** třídy Array, kdy je při každé změně (tedy i vložení) hodnoty posuzováno, zdali hodnota splňuje určitá kritéria. Validační funkce dokáže některé hodnoty převést (např. „20“ převést na 20), jiné ovšem ne a pak proces přeruší chybou.

```
var mix = new NumericVector(12, "20", true, false, 3, 7, 6, 8, 9, 10);
/* NumericVector(10) [12, 20, 1, 0, 3, 7, 6, 8, 9, 10] */

var mix = new NumericVector(12, "x", 3, 7, 6, 8, 9, 10);
/* Uncaught Error Tento typ vektoru akceptuje pouze numerické a prázdné
hodnoty.Vrácená chybná hodnota: x */
```

Vestavěné validátory jsou uloženy v privátním objektu *vectorParsers* (viz níže), který není přístupný z API, ale pro pochopení systému validace je jeho princip popsán níže. Přeloženo do běžného jazyka fungují jednotlivé validátory následovně:

1. Numeric (číselný)
    a. Pokud je hodnota nominální (string), je očištěna o mezery a případné čárky jsou změněny na tečky (konverze desetinného znaménka)
    b. Hodnoty 0, „0“ a false jsou převedeny na **0**
    c. Prázdné a chybové hodnoty (undefined, null, NaN ad.) jsou převedeny na **null**
    d. Hodnoty převoditelné na čísla jsou převedeny na **čísla**
    e. U ostatních hodnot (text, objekt, funkce atd.) je vrácena **chyba**
2. String (textový)
    a. Neprázdné hodnoty, nula a false jsou převedeny na **text** (např. „x“, „0“, „false“)
    b. Ostatní hodnoty jsou převedeny na **null**
    c. Validátor nikdy nezpůsobuje chybu
3. Boolean (binární)
    a. Neprázdné hodnoty jsou převedeny na **true** (pravda)

b. Hodnoty false a nula (i v textovém formátu) jsou převedeny na **false** (nepravda)

c. Ostatní hodnoty jsou převedeny na **null**

d. Validátor nikdy nezpůsobuje chybu

2.1.3.3 Společné vlastnosti a metody

Vlastnosti a zejména metody tříd typu Vector lze rozdělit na dvě skupiny: nevýpočetní, které jsou pevnou součástí tříd, a výpočetní (statistické), které jsou generovány jako prototypové metody dynamicky na základě nastavení (viz dále), případně jsou přístupné přes metodu ***analyze*** (viz dále).

Vektor nemá oproti mateřské javascriptové třídě Array žádné další vlastnosti (properties). Jelikož je třída Vector derivátem třídy Array, obsahuje zároveň všechny zděděné metody, k tomu navíc i rozsáhlou řadu rozšíření, která jsou definována v modulu *extensions.js* (seznam metod lze prohlížet přes vlastnost *Array.prototype*). V instanci vektoru je tak možné používat např. vlastnosti *length*, funkce *filter*, *find* atd.

Specifickými metodami vektorů jsou **referenční funkce**, které ukládají hodnoty, jako např. ***name*** (název vektoru), do instance WeakMap. Tímto způsobem lze zapisovat i číst virtuální vlastnosti vektoru, aniž by se tím narušila struktura vektoru s konkrétním datovým typem (např. do číselného vektoru nemohu najednou zapsat jako jednu z hodnot jméno). Způsob zadávání a získávání hodnoty byl inspirován knihovnou jQuery (24), kdy je hodnota zadána tím, že se zadá jako argument funkce (a je vrácena původní instance), hodnota se naopak získá tak, že se při volání metody nezadá žádný argument. Tímto způsobem fungují všechny metody, které jsou vázány na privátní instanci WeakMap.

```
var age = new BooleanVector(true, false, true);
console.log(age.name("věk"))
/* změní hodnotu name a vrátí vlastní vektorovou instanci => BooleanVector(3) [true,
false, true] */
console.log(age.name());
/* vrátí hodnotu name => věk */
```

Základní i děděné vektorové třídy mají řadu společných metod a vlastností, které jsou uvedeny dále.

2.1.3.3.1 Klonování a konverze vektorů

Metoda ***clone*** vytvoří kopii původního vektoru, včetně metadat. Volitelný argument metody *flush* (defaultně nepravda) při pravdivé hodnotě smaže původní hodnoty, čímž vrátí prázdný vektor se zachovanými metadaty (např. name). Obdobou metody clone je metoda ***reload***, která vytvoří identickou kopií původního vektoru, ovšem nahradí jeho hodnoty hodnotami uvedenými v argumentu. Pro případy, kdy je určitý typ vektoru třeba převést na jiný, je užitečná metoda ***convert***, která má dva argumenty: enumeraci typu (1 = numerický, 2 = textový, 3 = binární) a volitelně vlastní konverzní funkci jako druhý argument.

```
var V = new NumericVector(11, 15, 9, 4, 34, 17, 18, 14, 12, 13, 26,
31).name("respondent‘s score");
/* vrátí identickou kopii */
var _V = V.clone();
/* vrátí prázdnou kopii */
var __V = V.clone(true);
/* vrátí kopii vektoru s novými hodnotami z argumentu */
var C = V.reload(9, 8, 7, 6);
/* převede numerický vektor na textový */
var S = V.convert(2);
/* převede textový vektor na numerický za pomoci konverzní funkce */
var S = new StringVector("man", "man", "woman", "woman", "woman", null, "man",
"woman");
var cV = S.convert(1, function (value) { return value == "man" ? 1 : value == "woman"
? 2 : null })
```

2.1.3.3.2 Vzorkování

Metoda ***sample*** vybírá z hodnot vektoru N hodnot pseudonáhodným losováním. Velikost vzorku určuje jediný argument metody, *size*, kdy se jedná o kladné číslo větší nebo rovné nule. Chování metody na základě hodnoty atributu *size* lze shrnout následovně:

1. Pokud je nula nebo menší než nula, vrátí se kompletní kopie původního vektoru
2. Pokud je menší než jedna, bere se hodnota jako *size * 100 %* případů z celku (tedy při *size* = 0.5 se vybere náhodně 50 % případů)
3. Pokud je hodnota větší nebo rovná 1, vybere se počet případů odpovídajících *size*
4. Pokud je hodnota větší než počet případů ve vektoru, vrátí se kompletní kopie původního vektoru

2.1.3.3.3 Generování vektorů

Statická metoda ***generate*** vytvoří pomocí generátoru pseudonáhodných hodnot nový vektor s N hodnotami konkrétního typu (numerického, textového atd.), přičemž lze jednotlivé hodnoty vymezit vlastnostmi, které jsou specifikovány v objektu *config* (jediný argument metody). Společnými vlastnostmi tohoto objektu pro všechny typy vektorů jsou *total*, což je celé číslo indikující požadovaný počet generovaných příkladů (defaultně 100), a dále *nullprob*, což je hodnota udávající pravděpodobnost výskytu prázdné hodnoty (defaultně nula). Další vlastnosti se liší podle typu vektoru.

2.1.3.3.4 Generování numerických vektorů

V objektu *config* je možno nastavit hodnoty *min* a *max*, což jsou reálná čísla určující minimální a maximální možnou hodnotu generovaného čísla. Pokud uživatel hodnoty neupřesní, nabývají hodnot 2^{53} (max), resp. -2^{53} (min).

```
/* vygeneruje 100 hodnot mezi -2^52 a 2^52, bez prázdných hodnot.*/
var n = NumericVector.generate();
/* vygeneruje 1000 hodnot mezi mezi -2^52 a 2^52, bez prázdných hodnot.*/
var n = NumericVector.generate({ total: 1000 });
/* vygeneruje 1000 hodnot mezi nulou a mezi 2^52, bez prázdných hodnot.*/
var n = NumericVector.generate({ total: 1000, min: 0 });
/* vygeneruje 1000 hodnot mezi 0 a 200, bez prázdných hodnot.*/
var n = NumericVector.generate({ total: 1000, min: 0, max: 200 });
/* vygeneruje 100 hodnot mezi nula a 2^52, s 50 % šancí, že hodnota bude prázdná. */
var n = NumericVector.generate({ total: 100, min: 0, nullprob: 0.5 });
```

2.1.3.3.5 Generování nominálních a binárních vektorů

Metoda je založena na náhodném výběru z katalogu nominálních hodnot. Specifikaci rozsahu a možnosti hodnot lze provést v objektu *config* pomocí vlastnosti *list*. Ta může nabývat dvou typů:

1. Celé kladné číslo, které specifikuje počet možných kategorií (např. pro hodnotu 5 bude z vestavěného anglického katalogu náhodně vybráno 5 hodnot, ze kterých se bude náhodně losovat N případů)
2. Řada (instance Array), která určuje konkrétní hodnoty, ze kterých se mají případy losovat

Vlastnost *list* je povinná.

```
/* vygeneruje 100.000 případů s hodnotami A, B nebo C, s pravděpodobností prázdné
hodnoty 50 % */
var strings = StringVector.generate({ list: ["A", "B", "C"], total: 100000, nullprob:
0.5 })
/* vygeneruje 100.000 případů se 4 možnými hodnotami (např. "yellow", "horse",
"needle", "punk"), s pravděpodobností výskytu prázdné hodnoty 20 % */
var strings = StringVector.generate({ list: 4, total: 100000, nullprob: 0.175 })
/* vygeneruje 100 případů s 5 možnými hodnotami, s pravděpodobností výskytu prázdné
hodnoty 17.5 % */
var strings = StringVector.generate({ list: 5, nullprob: 0.1 })
/* vygeneruje 100 případů s 5 možnými hodnotami, bez prázdných hodnot */
var strings = StringVector.generate({ list: 5 })
```

Generování binárních vektorů je principiálně stejné jako generování nominálních vektorů, s tím, že nastavitelné jsou pouze parametry *total* a *nullprob*. Získání informací o typu vektoru

Základní metoda ***type*** vrací enumeraci typu vektoru (viz výše). Volitelným argumentem funkce je hodnota *verbose*, která – pokud je pravdivá (defaultně není) – vrátí slovní název třídy (např. StringVector), jinak vrací enumeraci třídy (v tomto případě 2). Vedle toho může být užitečná i vlastnost ***isVector***, která je statickou vlastností základní třídy Vector a vrací hodnotu *true* u základní i všech děděných vektorových tříd (je tak možno rychle ověřit, zdali se jedná o vektor či nikoliv).

```
var x = new StringVector("A", "B", "C");
console.log(x.type()); /* 2 */
console.log(x.type(true)); /* StringVector */
console.log(x.isVector); /* true */
console.log(["A", "B", "C"].isVector) /* false */
```

2.1.3.3.6 Serializace vektorů

Aby bylo možno vektor serializovat (tj. převést do textové podoby) včetně metadat, nelze použít metodu *JSON.stringify*, neboť ta metadata ignoruje. Místo toho je potřeba použít funkci ***serialize***, která vrátí objekt (pozor, nikoliv typ string) obsahující hodnoty vektoru ve vlastnosti *values* a metadata v dalších vlastnostech. Reverzní funkcí k *serialize* je statická metoda ***deserialize***, která výše zmíněná data naopak přetransformuje v instanci vektorové třídy. Vstupním argumentem pro druhou jmenovanou metodu může být jak objekt, tak hodnota typu string.

2.1.3.3.7 Volání metod statistické analýzy

Klíčovým prvkem vektorů jsou statistické metody. Ty lze volat dvěma způsoby, buďto přímo názvem metody, nebo pomocí zprostředkující funkce ***analyze***. V obou případech se jedná o volání metody *run* v instanci třídy VectorAnalysis, která je podrobněji popsána dále v textu. Je zde na místě zmínit statickou metodu ***listMethods***, která (bez argumentu) vrací jmenný seznam statistických metod a slouží jako určitá forma nápovědy, dostupná v reálném čase ve vývojovém prostředí.

```
var V = new NumericVector(11, 15, 9, 4, 34, 17, 18, 14, 12, 13, 26, 31);
var v = vector(11, 15, 9, 4, 34, 17, 18, 14, 12, 13, 26, 31);
var sum_a = V.sum();
var sum_b = V.analyze("sum").run().result;
var sum_c = v.sum();
var sum_d = v.analyze("sum").run().result;
// hodnoty sum_a, sum_b, sum_c i sum_d jsou totožné
```

2.1.3.3.8 Další metody

Společných metod pro vektory je kromě výše uvedených ještě několik. Z části ošetřují specifické případy procedur, které nabízí mateřská třída Array (např. push¸ filter atd.). Speciální, statickou metodou, je funkce ***register***, která uživatelům umožňuje do aplikace vkládat vlastní statistické metody.

2.1.4 Třída VectorAnalysis

Třída ***VectorAnalysis***, přesněji její instance, je vlastním procesorem statistické analýzy, v tomto případě statistických metod prováděných nad jednou proměnnou (vektorem). Z ontogenetického hlediska tato třída (a také třída *MatrixAnalysis*) prošla složitějším vývojem než ostatní třídy, zejména v kontextu analýzy chování komplexních softwarových řešení. Většina srovnatelných javascriptových knihoven pro statickou analýzu, která tato práce brala v úvahu, funguje tak, že na volání metody vrátí konkrétní výsledek volané metody (např. hodnoty *r* a *p* pro korelaci). Chování RTS je odlišné, již z toho důvodu, že na místo funkce používá celou třídu, která má kromě dílčích funkcí i vlastnosti a krom toho ji lze konfigurovat. Důvod tohoto přístupu je třeba hledat zpětně u analýzy existujících řešení. Pro interpretaci statistických testů a metod obecně je často užitečné sledovat i jiné než testové statistiky, např. velikost vzorku; také je třeba kontrolovat, že vstupní argumenty splňují požadavky standardně vyžadované. Uvažovaná třída má tedy celou řadu fundamentálních funkcí, které celý proces analýzy kontrolují a zároveň ukládají do metadat potřebné informace.

Dříve bylo zmíněno, že statistické metody lze volat dvěma způsoby, buďto přímo, nebo skrze funkci ***analyze***. Ještě než tyto přístupy popíšeme blíže, přibližme si funkční řešení třídy samotné. Podobně jako v případě nastavení uživatelského jména vektoru, i v případě klíčových funkcí třídy se uplatňuje podobný přístup, jaký je praktikován u knihovny jQuery (24) – tyto funkce nevrací konkrétní výsledek; místo toho jej ukládají do metadat a vrací samotnou instanci. Tímto způsobem je možné uchovávat průběžně metadata, aniž by vývojář musel vytvářet vlastní proměnné, do kterých bude metadata zapisovat (krom toho, že to vzhledem k několika sériovým krokům není možné). Zároveň, při kapacitně náročnějších výpočtech (např. testy manipulující s pořadím několika milionů případů) je díky tomu efektivnější provádět analýzu po krocích, např. nejdříve ověřit argumenty, čímž odpadá povinnost provádět všechny kroky analýzy v rámci jednoho volání.

Nejprve uveďme jednotlivé scénáře na příkladu volání metody *frequency*, tedy tabulky četností, což je metoda dostupná u všech typů vektorů.

```
var variable = new NumericVector(10, 12, 3, 12, 12, 15, 13, 15, 14, 13, 19, 17, 15, 3);
/* příklad 1 */
var f = variable.frequency();
/* příklad 2 */
var af = variable.analyze("frequency");
/* příklad 3 */
var af_with = variable.with(1);
var af_prepared = variable.prepare();
var af_ run = af_prepared.run();
var af_result = af _run_result;
/* příklad 4 */
var af_run = variable.run(1);
var af_result = af_run.result;
```

Příklad 1

Jedná se o přímé volání metody, jejíž vrácenou hodnotou je samotný výsledek metody (tabulka četností). Jelikož se konkrétně tato metoda obejde bez argumentu (v případě chybějícího parametru je načtena defaultní hodnota), lze ji volat bez specifikace.

Příklad 2

Volání metody přes pomocnou funkci ***analyze***, kde argumentem je název metody. Tato metoda nevrací výsledek, nýbrž samotnou instanci třídy VectorAnalyze.

Příklad 3

Vycházejme z předchozího příkladu: je uvedeno, jako metodu chceme vyvolat (*frequency*). V prvním řádku voláme funkci ***with***, která přijímá argumenty a následně je validuje. Následně voláme funkci ***prepare***, která vstupní data upraví do formátu vyžadovaného výpočetní funkcí. V dalším řádku voláme funkci ***run***, která samotnou metodu spouští (vypočítá). Metoda nevrací vlastní výsledek výpočtu, nýbrž jej uloží do vlastnosti ***result*** a vrací opět samotnou instanci. Čtvrtý řádek již zprostředkovává samotný výsledek metody.

Příklad 4

Jedná se o alternativu příkladu 3, s tím, že argumenty jsou upřesněny až v metodě ***run***, která zajišťuje výpočet i předchozí operace. Hodnota *f* z příkladu 1 je rovna hodnotě *af_result*.

Výše uvedené příklady ukazují, jak flexibilní je volání konkrétní statistické metody. V této souvislosti je třeba zdůraznit význam sekvenčního volání. V principu jde o to, že jednotlivé metody třídy VectorAnalysis (i MatrixAnalysis popsané dále) jsou na sobě závislé a automaticky se (dle potřeby) samy volají. Ve 4. příkladu výše se nejdříve autonomně vyvolá funkce ***with***, která validuje vstupní argumenty. Následně je volaná metoda ***prepare***, které validované argumenty převede do formátu přijatelného k výpočtu (vyčistí data, transformuje matici apod.). Metoda *prepare*, v dokumentaci uváděná jako před-výpočetní úprava dat, zároveň do objektu ***sample*** zaznamená počet vstupních (raw) a relevantních (net) hodnot, přičemž pro každou metodu je použitý specifický přepočet těchto hodnot (25). Kromě toho je do objektu ***time*** zaznamenáván časový začátek o konec zpracování metody, což slouží především k analýze náročnosti výpočtu; celková doba zpracování je přístupná v kmenové metodě třídy ***duration***. Dalšími metodami a vlastnostmi třídy jsou dokumentační objekty (*title, description*), zdrojový model metody (*model*), specifikace argumentů atd.

Flexibilita se projevuje i ve smyslu volnosti udávání argumentů. Ty lze zadávat dvojím způsobem:

1. V pořadí, které specifikuje dokumentace (argumenty odděleny čárkou)
2. Jmenovitě podle klíčů, které specifikuje dokumentace (objekt s vlastnostmi)

Níže uvedený příklad ukazuje různé způsoby zadávání argumentů u metody ***histogram***, která přijímá dva parametry - fix (pevná velikost intervalu) a max (maximální počet intervalů), které jsou nepovinné.

```
var score = new NumericVector(4.5, 3.9, 5, 6, 7, 5.7, 9.1, 5.3, 7.2, 6.9, 6, 7.5, 5.3,
7.1, 8.2, 1);
var h1 = score.histogram(null, 3);
var h2 = score.analyze("histogram").run(null, 3);
var h3 = score.histogram({ fix: 3 });
var h4 = score.analyze("histogram").run({ fix: 3 });
// h1 = h3 = h2.result = h4.result
```

Konkrétní varianty zadávání argumentů mají různou povahu a jsou popsány v dokumentaci (25). V případě vektorových metod se nejčastěji jedná o čísla, enumeraci, binární hodnotu, případně hodnotu libovolného typu (příklad metoda *pci*).

2.1.5 Třída Matrix

Třída ***Matrix*** (matice) představuje druhou klíčovou entitu celé aplikace. Tak, jako jsou vektorové třídy analogií ke sloupci v Excelu, je třída Matrix analogií k excelovému listu (přesněji oblasti). Podobně jako třída Vector zpřístupňuje uživateli vektorové funkce, třída Matrix zprostředkovává uživateli maticové funkce. Třída Matrix je, stejně jako Vector, dceřinou třídou prototypu Array, přesto i ona vykazuje oproti mateřské třídě některá specifika.

2.1.5.1 Konstrukce

Instanci matice lze konstruovat různými způsoby, přičemž argumenty konstruktoru musí tvořit vektory či řady, které lze transformovat na vektory – oba dva typy argumentů lze vzájemně kombinovat. Argumenty jsou zadávány seriálně, tedy odděleny čárkou, nemůže se jednat o set argumentů ve formě řady (Array), jelikož tu by konstruktor považoval za vektor a vrátil by chybu. Řešením je využít rozšiřovací statickou metodu ***matrify*** u třídy Array, která se pokusí základní třídu převést na matici (příklad proměnné M4 níže). Výsledek všech proměnných (M1, M2, M3 a M4) v kódu níže je totožný.

```
var M1 = new Matrix(new NumericVector(1, 2, 3), new StringVector("A", "B", "C"), new
BooleanVector(false, true, true));
var M2 = new Matrix([1, 2, 3], ["A", "B", "C"], [false, true, true]);
var M3 = new Matrix([1, 2, 3], new StringVector("A", "B", "C"), [false, true, true]);
var M4 = [[1, 2, 3], ["A", "B", "C"], [false, true, true]].matrify();
```

Princip validace hodnot matice je prostý, jelikož pouze ověřuje, zdali je každá hodnota instancí vektoru, či na ni může být převedena pomocí metod *vectorify*, *numerify* atd.

Alternativou ke konstrukci je volání kmenové funkce ***matrix*** (s malým m na začátku), které je, podobně jako v případě metody ***vector***, obdobou běžně používaného definování matice (např. jStat).

```
var m = matrix([11, 15, 9, 4, 34, 17, 18, 14, 12, 13, 26, 31], [34, 31, 35, 29, 28, 12,
18, 30, 14, 22, 10]);
var mm = new Matrix(new NumericVector(11, 15, 9, 4, 34, 17, 18, 14, 12, 13, 26, 31), new
NumericVector(34, 31, 35, 29, 28, 12, 18, 30, 14, 22, 10));
// m = mm
```

V souvislosti s konstrukcí můžeme zmínit i metodu ***clone***, která vytvoří identickou kopii instance, ze které je metoda volána.

2.1.5.2 Obecné vlastnosti a metody

Stejně jako Vector, i třída Matrix má celou řadu metod a vlastností, které lze dělit na obecné a výpočetní (statistické). Obecné metody mají funkci buďto modifikační, vyhledávací/třídící nebo informační.

2.1.5.2.1 Identifikátor vektoru

Jelikož se v základu chová třída Matrix jako Array, lze výběr vektoru řešit standardním způsobem. Nicméně, v řadě případů je jednodušší přistupovat k vektorům i jiným způsobem, např. vyhledáváním vektoru podle jeho jména. Metoda ***item*** umožňuje flexibilnější hledání, resp. umožnuje hledání s méně kódováním. Argumentem metody může být:

1. celé číslo (index vektoru v matici)
2. text (jméno vektoru, zadané pomocí metody ***name***)
3. vektor samotný

```
var a = new NumericVector(4, 5, 9, 7, 3, 2).name("A");
var b = new NumericVector(7, 5, 3, 5, 6, 7).name("B");
var c = new NumericVector(-2, -4, -7, -5, 8, 10).name("C");
var M = new Matrix(a, b, c);
var v1 = M.item(1);
var v2 = M.item("B");
var v3 = M.item(b);
/* v1 === v2 === v3 */
```

Implementace metody *item* je běžná napříč vestavěnými statistickými metodami matic, kde jsou vektory často jediným argumentem. Typy identifikátorů lze v rámci specifikace vstupů statistických metod volně kombinovat.

```
var a = new NumericVector(4, 5, 9, 7, 3, 2).name("A");
var b = new NumericVector(7, 5, 3, 5, 6, 7).name("B");
var c = new NumericVector(-2, -4, -7, -5, 8, 10).name("C");
var M = new Matrix(a, b, c);
var correl = M.correlPartial("A", b, 2);
```

2.1.5.2.2 Filtrování řádků matice

Metoda **filter** vytvoří kopii původní matice a aplikuje na ni filtr specifikovaný argumentem či argumenty. Je nutné upozornit, že ačkoliv je prekurzorem třídy Matrix prototyp Array, metoda *filter* může fungovat dvěma zcela odlišnými způsoby, a to dle zadaných argumentů:

- Pokud je zadán jeden argument
    - typu funkce, metoda filter funguje stejně jako mateřská funkce (tedy na základě argumentu filtruje vektory matice; př. 1)
    - typu Array s numerickými hodnotami, jsou vráceny řádky matice nalezené v tomto argumentu (př. 2)
- Pokud je zadán více než jeden argument, pak jsou filtrovány hodnoty vektorů, přičemž
    - lichý argument identifikuje vektor (viz metodu ***item***)
    - sudý argument představuje filtrační funkci vektoru
    - pokud je argumentů více než jeden, jejich počet musí být vždy sudý (tedy enkrát identifikátor vektoru a filtrační funkce), v opačném případě metoda vrátí chybu
        - výsledkem je matice s totožným výčtem vektorů; vráceny jsou pouze ty řádky, které splňují všechna zadaná kritéria v každém řádku – jednoduše řečeno, funguje přesně stejně jako filtry pro jeden a více sloupců v Excelu

```
/* příklad 1 - argumentem je funkce, která vrátí pravda pouze na vektory s názvem
"group" */

var M = new Matrix(new NumericVector(10, 12, 13, 20, 21, 25).name("score"), new Strin-
gVector("a", "a", "a", "b", "b", "b").name("group"));
var fM = M.filter(function (vector) { return vector.name() == "group" });

/*
fM =>Matrix(1) [StringVector(6)]
    0:StringVector(6) ['a', 'a', 'a', 'b', 'b', 'b']
 */

/* příklad 2 - argumentem je řada (celých čísel), jsou tedy vrácena všechny řádky s
indexy ze zadaného argumenty (zde tedy řádky 0 a 2) */
var M = new Matrix(new NumericVector(10, 12, 13, 20, 21, 25).name("score"), new Strin-
gVector("a", "a", "a", "b", "b", "b").name("group"));
var fM = M.filtrer([0, 2]);
/* fM = >Matrix(2) [NumericVector(2), StringVector(2)]
    0: NumericVector(2) [10, 13]
    1: StringVector(2) ['a', 'a']
*/

/* příklad 3 - argumentem jsou dva páry idenfitikátor-funkce, vrátí se pouze řádky,
kde je vektor "score" menší než 13 a zároveň se vektor "group" (index 1) rovná "a" */
var M = new Matrix(new NumericVector(10, 12, 13, 20, 21, 25).name("score"), new Strin-
gVector("a", "a", "a", "b", "b", "b").name("group"));
var fM = M.filter("score", function (value, index) { return value < 13 }, 1, function
(v) { return v === "a" });
/* fM => Matrix(2) [NumericVector(2), StringVector(2)]
    0:NumericVector(2) [10, 12]
    1:StringVector(2) ['a', 'a']
*/
```

2.1.5.2.3 Čistění dat matice

Metoda ***flush*** smaže hodnoty všech vektorů uvnitř matice, aniž by tím měnila strukturu matice či metadata vektorů.

2.1.5.2.4 Transformace matice

Pomocí metody ***pivot*** lze určitý vektor v matici roztřídit podle jiného vektoru v téže matici. Tato funkce je obzvláště účinná při volání metod jako je ANOVA či T-test, které coby vstupní argumenty konzumují konkrétní vektory (přesněji jejich identifikátory); v případě, že máme jednotlivé testované skupiny definované v řádcích třídícího vektoru, je potřeba je před voláním těchto metod upravit metodou *pivot*:

```
var M = new Matrix(new NumericVector(10, 12, 13, 20, 21, 25), new StringVector("a",
"a", "a", "b", "b", "b"));
var T = M.pivot(0, 1);
/* T => Matrix(2) [NumericVector(3), NumericVector(3)]
    0: NumericVector(3) [10, 12, 13]
    1:NumericVector(3) [20, 21, 25]
*/
var c1_title = T[0].name(); // 'a'
var c2_title = T[0].name(); // 'b'
```

Povšimněme si, že nově vytvořeným vektorům jsou automaticky přiřazeny názvy (vlastnost *name*) dle původních hodnot faktoru (v tomto případě tedy vznikly sloupce s názvy „a“ a „b“).

2.1.5.2.5 Výběr skupiny vektorů

Výběr vektorů je možné provést buďto dříve zmíněnou metodou ***filter***, nebo metodou ***select***, jejímiž argumenty jsou identifikátory vektorů. Obecně je metoda *select* vhodnější tam, kde se vybírají vektory „jmenovitě“, zatímco metoda *filter* tam, kde je nutná důkladnější analýza filtrovaných položek (např. vektory, které mají v matici index menší než 5). Výběr jediného vektoru je možné provádět dříve popsanou metodou ***item***.

2.1.5.2.6 Vzorkování

Princip vzorkování byl popsán u vektorů, v případě konceptu matic funguje velmi podobně, s tím, že náhodný výběr probíhá na úrovni řádků. Metoda ***sample*** vybírá z hodnot vektoru N hodnot pseudonáhodným losováním. Velikost vzorku určuje jediný argument metody, *size*, kdy se jedná o kladné číslo větší nebo rovné nule. Chování metody na základě hodnoty atributu *size* lze shrnout následovně:

1. Pokud je nula nebo menší než nula, vrátí se kompletní kopie původního matice
2. Pokud je menší než jedna, bere se hodnota jako *size * 100 %* případů z celku (tedy při *size* = 0.5 se vybere náhodně 50 % řádků matice)
3. Pokud je hodnota větší nebo rovná 1, vybere se počet případů odpovídajících velikosti *size* (po zaokrouhlení)
4. Pokud je argument větší než počet případů v nejdelším vektoru matice[19], vrátí se kompletní kopie původní matice

```
var M = new Matrix(new NumericVector(11, 15, 9, 4, 34, 17, 18, 14, 12, 13, 26, 31),
new NumericVector(34, 31, 35, 29, 28, 12, 18, 30, 14, 22, 10));
var sampledM = M.sample(0.5);
// => [[4,34,17,14,12,13],[29,28,12,30,14,22]
```

2.1.6 Třída MatrixAnalysis

Třída ***MatrixAnalysis*** je analogií k třídě *VectorAnalysis* a jako taková zprostředkovává nejen totožné API, ale stejně se i chová. Vývojově je tato třída starší než její vektorová obdoba, na analýze matic se totiž projevila potřeba systematicky řídit jednotlivé procesy a uchovávat metadata. Po určitou dobu vývoje byl prováděn výpočet vektorových a maticových metod dvěma jakostně odlišnými způsoby, v důsledku čehož vyšlo najevo, že metoda užívaná u matic je vhodnější než ta vektorová, která byla následně přizpůsobena maticovému modelu.

2.1.7 Statistické metody

Pro analýzu vektorů jsou klíčové **statistické metody**, které se generují dynamicky, tj. nejsou přímo skriptovány do prototypu třídy, ale vytváří se až po definování třídy Vector (tzn. bezprostředně po iniciaci). Příklad definice vektorové metody nalezneme níže. Obsahuje informace o názvu metody v API vektoru (zde *ttest*, tedy T-test pro jeden výběr), jakou funkcí je počítána (*fn*), před-výpočetní transformaci (*prepare*), název schématu vrácené hodnoty

[19] Vektory uvnitř matice mohou mít různou délku (počet členů).

(*output*), strukturu argumentů (*args*), popisky (*wiki*) atd. Uvedený objekt je použit jako argument konstruktoru u třídy VectorAnalysis, která následně zajišťuje validaci volání metody (argumenty) a ní spojené procesy. Doplňme, že architektura třídy MatrixAnalysis je totožná.

```
var ttestConfig = {
    name: "ttest",
    fn: Array.prototype.ttest,
    wiki: {
        title: "VEAt",
        description: "rbjM",
        preprocessor: preprocessors.removeEmpty.title,
    },
    prepare: preprocessors.removeEmpty.fn,
    type: [1],
    output: "ttest",
    args: {
        populationMean: {
            model: "number",
            config: {
                name: "populationMean",
                title: "GRoZ",
                description: "xtfz",
                required: true
            }
        }
    }
}
```

2.1.7.1 Metodika skriptování

Procesně byl vývoj metod popsán dříve, nicméně v souvislosti s tím, jak jsou statistické metody počítány a zpracovány, je třeba alespoň krátce klasifikovat a popsat užité zdroje. U některých metod byl vývoj založen na obecných znalostech (počet, součet, aritmetický průměr apod.), v dalších případech byly podklady pro konverzi vzorců (a často i konkrétních postupů) odvozovány z několik různých typů zdrojů. Zejména u maticových metod je zpracování výpočtu někdy poměrně komplikované a složené z řady kroků, nejčastěji u metod s řazením případů a v těchto scénářích bylo nutno imitovat postupy popsané v některém z níže uvedených typů zdrojů.

2.1.7.1.1 Literatura

Zejména výpočetně jednodušší metody byly zpracovány na základě samotných vzorců (26), některé metody s komplexnějšími výstupy pak s pomocí prakticky orientované Statistiky v příkladech (27), která postupy vysvětluje za použití MS Excel. Tento model vývoje (simulace v Excelu a paralelní programování v JavaScriptu) se osvědčil jako nejspolehlivější způsob kontroly algoritmu výpočtu a také obvykle vedl k nejpřímějšímu dosažení požadovaného výsledku (ve smyslu validace s SPSS).

2.1.7.1.2 Online zdroje

Jako mimořádně užitečné se projevily dva online projekty, které se zabývají popisem výpočtu a interpretace řady statistických metod s přednostním využitím Excelu a které jsou online obdobou výše uvedené příručky (27):

1. Real Statistics Using Excel (28)
2. Statistics How To (29)

Stranou odkazování na konkrétní zdroje, jako nápomocné se při výpočtu některých pokročilejších metod (např. zohlednění opakovaných pořadí při výpočtu z-testu pro Kendallovu korelaci) ukázaly I video tutoriály (YouTube apod.).

2.1.7.1.3 Knihovny třetích stran

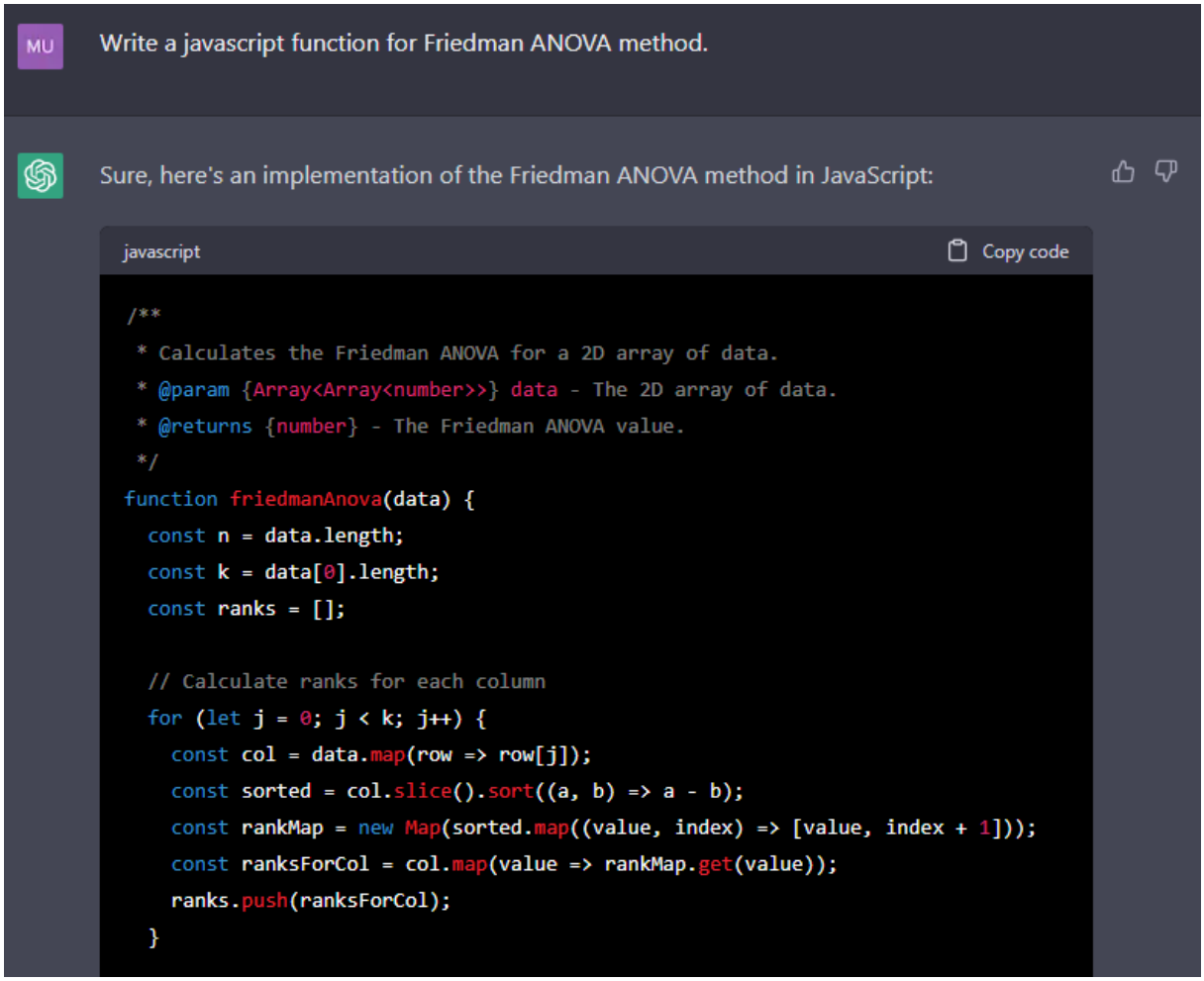
V průběhu vývoje vyšlo najevo, že přebírání existujících skriptů do RTS je, s ohledem na její specifickou architekturu, více problematické než přínosné a velká většina funkcí byla nakonec zpracována na základě výše uvedených přístupů. Hlavním důvodem, proč byly statistické metody vyvíjeny autonomně, bylo efektivní využívání dříve uvedených modulů a jejich tříd a funkcí (vektorové knihovny, extenze Array apod.) a také poměrně přísných pravidel při využívání tříd *VectorAnalysis* a *MatrixAnalysis*. Důležitou výjimku tvoří integrace části knihovny jStat (18), ze které byly převzaty distribuční funkce.

2.1.7.1.4 Umělá inteligence

Shodou okolností byla v době vývoje aplikace veřejnosti zpřístupněna služby ChatGPT[20] (dále AI) neziskové společnosti OpenAI, kterou od začátku jejího spuštění provázely velmi dobré reference ohledně interpretace kódů (tedy co kód konkrétně dělá). Přidaná hodnota ve vztahu k vývoji RTS byla spíše záporná, tj. využití AI bylo ve výsledků časově citelně náročnější, než vytvoření kódů některým z výše uvedených způsobů. Hlavní problém spočívá v tom, že AI sebou vytvořený kód nijak netestuje ani neinterpretuje a zjevně jej kompiluje z řady různých existujících skriptů. Generované metody byly soustavně validovány pomocí SPSS a s jedinou výjimkou (Goodman-Wallisův koeficient gama) nevracely srovnatelné výsledky. Během konverzace s AI pak vyšlo najevo, že algoritmus velice dobře „lže“, kdy po zadání správného výsledku uzná vlastní chybu a nějakým způsobem skript upraví, ovšem ani upravené výstupy nebyly po mnoha úpravách správné, nevedly ke správnému výsledku. Vzhledem k extrémně rychlému vývoji služby je zde třeba doplnit, že optimalizace učení může v krátké době vést i ke zlepšení těchto specifických úkonů, pro potřeby vývoje RTS bylo nicméně využití AI spíše kontraproduktivní.

[20] https://chat.openai.com/chat

OBRÁZEK 7: PŘÍKLAD KONVERZACE S CHATGPT



2.1.7.2 Přehled statistických metod

K 30. březnu 2023 nabízela RTS celkem 24 vektorových metod (popisné statistiky, histogram, testy ad.) a 16 maticových (korelace, lineární regrese, neparametrické testy aj.), jejichž soupis je uveden níže. Vývojový plán (back-log) metod nebyl předem nastaven, spíše se řídil simulací různých analytických úkolů v SPSS, načež byly potřebné metody doplňovány. Lze předpokládat, že při delší životnosti obou aplikací bude vhodné reflektovat potřeby uživatelů; respondenti během testování sami přicházeli s řadou vlastních nápadů a prioritních požadavků, které bude v produkční fázi nutné více zohledňovat (viz poslední kapitolu).

TABULKA 3: RETUSA - PŘEHLED STATISTICKÝCH METOD VEKTORŮ

funkce	metoda	numerický	nominální	binární
avg	aritmetický průměr	✓	-	-
count	počet	✓	✓	✓
frequency	tabulka četností	✓	✓	✓
geomean	geometrický průměr	✓	-	-
harmean	harmonický průměr	✓	-	-
histogram	histogram	✓	-	-
kstest	Kolmogorov-Smirnovův test	✓	-	-
kurtosis	špičatost	✓	-	-
max	maximální hodnota	✓	✓	✓
mci	interval spolehlivosti průměru	✓	-	-
median	medián	✓	-	-
min	minimální hodnota	✓	✓	✓
mode	modus	✓	✓	✓
pci	interval spolehlivosti podílu	✓	✓	✓
percentile	percentil	✓	-	-
range	variační rozpětí	✓	-	-
sem	střední chyba průměru	✓	-	-
swtest	Shapirův-Wilkův W test	✓	-	-
skewness	šikmost	✓	-	-
stdev	směrodatná odchylka	✓	-	-
sum	součet	✓	-	-
ttest	Jedno-výběrový t-test	✓	-	-
varc	variační koeficient	✓	-	-
variance	rozptyl	✓	-	-

TABULKA 4: RETUSA - PŘEHLED STATISTICKÝCH METOD MATIC

funkce	metoda
linreg	Lineární regrese
correlPearson	Pearsonův korelační koeficient
correlSpearman	Spearmanův korelační koeficient
correlGamma	Koeficient gama
correlKendall	Kendallova korelace
correlPartial	Parciální korelace
correlBiserial	Bodově biseriální korelace
anovaow	Jednofaktorová analýza rozptylu (ANOVA)
ttestind	Dvouvýběrový t-test
ttestpair	T-test (párový)
wcxind	Wilcoxonův test
mwu	Mann-Whitneyho test
kwanova	Kruskal-Wallisova ANOVA
wcxpair	Wilcoxonův znaménkový test pro dva závislé výběry
friedman	Friedmanova ANOVA
contingency	Kontingence

Každá metoda je popsána v dokumentaci, kterou lze nalézt pod níže uvedenými QR kódy.

Dokumentace v češtině

Dokumentace v angličtině

Dokumentace je zpracovaná v češtině a angličtině (pracovní verze). Soubory dokumentace zpracovává kompletně sama aplikace RTS, a to na základě metadat, kterou jsou součástí knihovny. Tato metadata jsou de facto popisy jednotlivých metod, chyb apod.; aktuálně lze dokumentaci připravovat s volnější závislostí na vývoji, kdy texty může připravovat podle předepsaných pravidel v Google Spreadsheets kdokoliv; speciální skript pak tato data dle potřeby stáhne a transformuje do javascriptové knihovny, která je součástí RTS.

Dokumentace každé metod sestává s následujících složek:

1. Lokální název metody
2. Stručný popis metody (liší se rozsahem)
3. Specifikace argumentů a jejich validátorů
4. Slovní specifikace před-výpočetní úpravy dat

5. Příklady syntaxe (kódu)
6. Schéma (diagram) výstupu se slovním popisem, co který indikátor znamená

OBRÁZEK 8: RETUSA - UKÁZKA DOKUMENTACE SPEARMANOVA KORELAČNÍHO KOEFICIENTU

Spearmanův korelační koeficient

Stanoví statistický protokol Spearmanova koeficientu pořadové korelace. Na rozdíl od Pearsonova korelačního koeficientu vychází hodnota testu z pořadí hodnot ve vstupních proměnných.

Argumenty

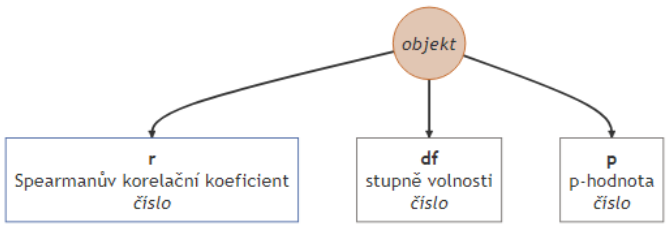
id	popis	typ hodnoty	validátor	povinný	defaultní hodnota
y	první proměnná	numerický vektor	Ověří, zdali je argument typově numerický vektor, nebo zdali se jedná o validní identifkátor numerického vektoru v matici, nebo - pokud je argument typu array - se pokusí řadu pomocí funkce 'numerify' převést na numerický vektor. Pokud se ani jedna z variant nezdaří, vyhodí chybu.	ano	
x	druhá proměnná	numerický vektor	Ověří, zdali je argument typově numerický vektor, nebo zdali se jedná o validní identifkátor numerického vektoru v matici, nebo - pokud je argument typu array - se pokusí řadu pomocí funkce 'numerify' převést na numerický vektor. Pokud se ani jedna z variant nezdaří, vyhodí chybu.	ano	

Před-výpočetní úprava dat Vstupní argumenty převede na vektory, ze kterých vytvoří matici. Z této matice následně odebere všechny řádky, které alespoň v jedné buňce obsahují prázdnou hodnotu. Vektory z této dceřiné matice poté přepíše původní argumenty, tzn. že vektory vstupují do metody již očištěné.

Příklady syntaxe

```
var M = new Matrix(
new NumericVector(3,7,5,10,9,8,4,1,6,2).name("design rating"),
new NumericVector(4,9,2,10,8,7,6,3,5,1).name("utility rating")
);
var rs_a = M.analyze("correlSpearman").run(0,1);
var rs_b = M.correlSpearman("design rating","utility rating");
// rs_a.result = rs_b
```

Schéma výsledku



2.2 Retusa GUI

Retusa GUI (RUI) je grafické rozhraní, které umožňuje provádět statistické operace v enginu Retusa pomocí editoru v browseru. Jedná se o prototyp jedné z možných podob grafických rozhraní, které zpřístupní jádrový procesor Retusa běžným uživatelům. Cílem nebylo dodat dokonalé řešení, nýbrž otestovat možnosti využitelnosti jádra v širší veřejnosti, které by bylo v případě RTS neúplné. Vrátíme-li se k definici MVP v první kapitole této práce, z hlediska uživatele byly jako klíčové definovány následující vlastnosti:

- Provádět analýzu vektorů a matic pomocí grafických komponent
- Zprostředkovat jednoduše interpretovatelné výsledky
- Možnost jednoduše importovat data z běžných tabulkových aplikací (Excel) a v aplikaci je upravovat

Aplikace je postavena primárně na grafickém prostředí Bootstrap (30) páté generace a knihovně jQuery, která zajišťuje interaktivnost v reálném čase. Tato kombinace je jedním z možných řešení a byla zvolena vzhledem k odhadovaným časovým nárokům na vývoj (vzhledem k dosavadním zvyklostem autora). Jelikož u obou aplikací mluvíme spíše o vývojových či beta verzích, produkční verze bude vyžadovat revizi a zvážení alternativních přístupů. Úspornost ve skriptování by např. přineslo dynamické vykreslování HTML objektů (zejména v kódování hůře přehledné generování formulářů metod atd.) nebo vázání datových a grafických struktur (pomocí VueJS, React apod.). Vzhledem k tomu, jak široké jsou možnosti dalšího rozvoje (viz závěrečnou kapitolu), inovace řešení musí být odvislá od komplexního plánu (mj. obchodního modelu). Zdůrazněme, že výše uvedené nemá žádný dopad na kvalitu výstupů ani komfort uživatele.

Specifickou vlastností RUI je to, že se jedná o čistě front-end řešení, tedy o sadu HTML stránky a javascriptových kódů, které běží pouze na počítači uživatele a nepotřebují žádnou serverovou podporu (kromě načítání knihoven z CDN serverů, zde se však nejedná o back-end služby v pravém slova smyslu). Částečně omezující byl tento přístup v úvodní fázi vývoje, neboť řada užitečných knihoven, které je možné spouštět v Node.js prostředí (coby serverovém systému), není spustitelná v prohlížeči, případně za vysokou cenu (kvůli velikosti), a proto bylo třeba tyto procesy naprogramovat a simulovat pro podmínky prohlížeče[21]. Nicméně, zmíněné omezení je benefitem pro koncového uživatele, neboť významně urychluje načítání aplikace v prohlížeči a z toho důvodu se s ním i od počátku počítalo. Zohlednění velikosti (souborů) v celkové architektuře obou aplikací bylo takové, že se – pohledem pamětníků - obě vejdou na jednu čtvrtpalcovou disketu.

Zatímco RTS je řešení, které není (s výjimkou javascriptového procesoru) závislé na žádné další službě, RUI systematicky využívá celou řadu knihoven třetích stran, zejména pro grafické zpracování dat. Jakkoliv je taková architektura běžná, je třeba jí dát do souvislosti agilním řízením vývoje, které zde bylo uplatněno. Principem štíhlých řešení (31) je obecně minimalizovat odpad (waste), který by byl v tomto případě reprezentován čímkoliv, co neřeší

[21] Příkladem je HTML renderer PUG, který se používá mj. v serverovém balíčku Express.js, kde pomáhá za pomoci minimálního kódování vytvářet dynamicky generované HTML soubory. Velikost balíčku určeného pro prohlížeč má velikost 2 MB (srovnejme se 140 kB celé RTS aplikace).

primární cíl aplikace – tedy vykreslování grafů, informačních oken apod. Dále, platí to co u RTS – závislost na knihovnách třetích stran je potenciální riziko, posílené tím, že se jedná o open-source zdroje s licencemi typu MIT, které se de facto zříkají jakékoliv zodpovědnosti, např. za udržování kompatibility. Jedním z důležitých úkolů budoucího vývoje bude určení priorit jednotlivých přebíraných knihoven a též velikost rizika jejich nestabilnosti, tak, aby se co nejefektivněji ošetřila možná zranitelnost nejen aplikace, ale i uživatelů samotných.

Aplikace je veřejně přístupná na adrese **t.ly/F-Q7** nebo pod níže uvedených QR kódem.

2.2.1 Kompatibilita

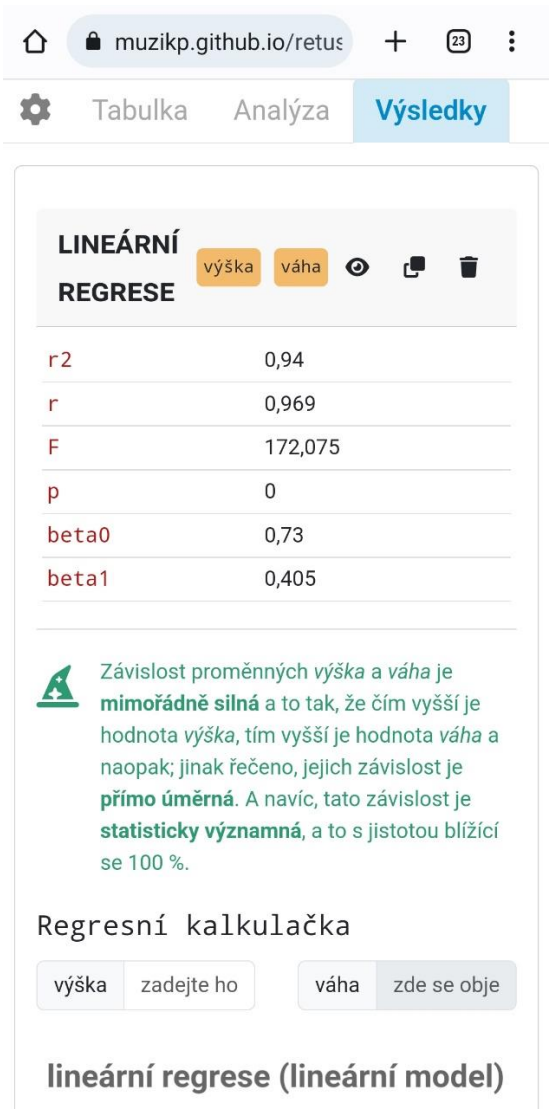
Funkčnost aplikace (verze 1.0.5) byla testována v následujících prohlížečích:

Aplikace	verze	Operační systém
Chrome	111.0.5563.65	Windows 11 (PC)
Mozilla (Firefox)	111.0.1	Windows 11 (PC)
Edge	111.0.1661.44	Windows 11 (PC)
Safari	111.0.5563.72	iOS (iPad)
Safari	111.0.5563.72	iOS (iPhone)
Chrome	105.0.5195.79	Android 12

Veškeré funkce (výpočetní) se projevují napříč platformami stejně. Graficky je aplikace postavena na knihovně Bootstrap, která efektivně zohledňuje různá rozlišeními displeje, je tedy vhodná pro optimalizaci rozložení jak u standardní obrazovky, tak u chytrých telefonů. RUI je plně funkční i na hlavních mobilních operačních systémech, byť se nepředpokládá, že by na nich byla hojněji používána. Potřebné zohlednění zařízení lze ve většině případů ošetřit pouze pomocí CSS souboru, např. úpravu velikosti fontu, cílenější distribuce elementů do sloupců apod.

OBRÁZEK 9: RETUSA GUI - ZOBRAZENÍ NA MOBILNÍM ZAŘÍZENÍ

Android



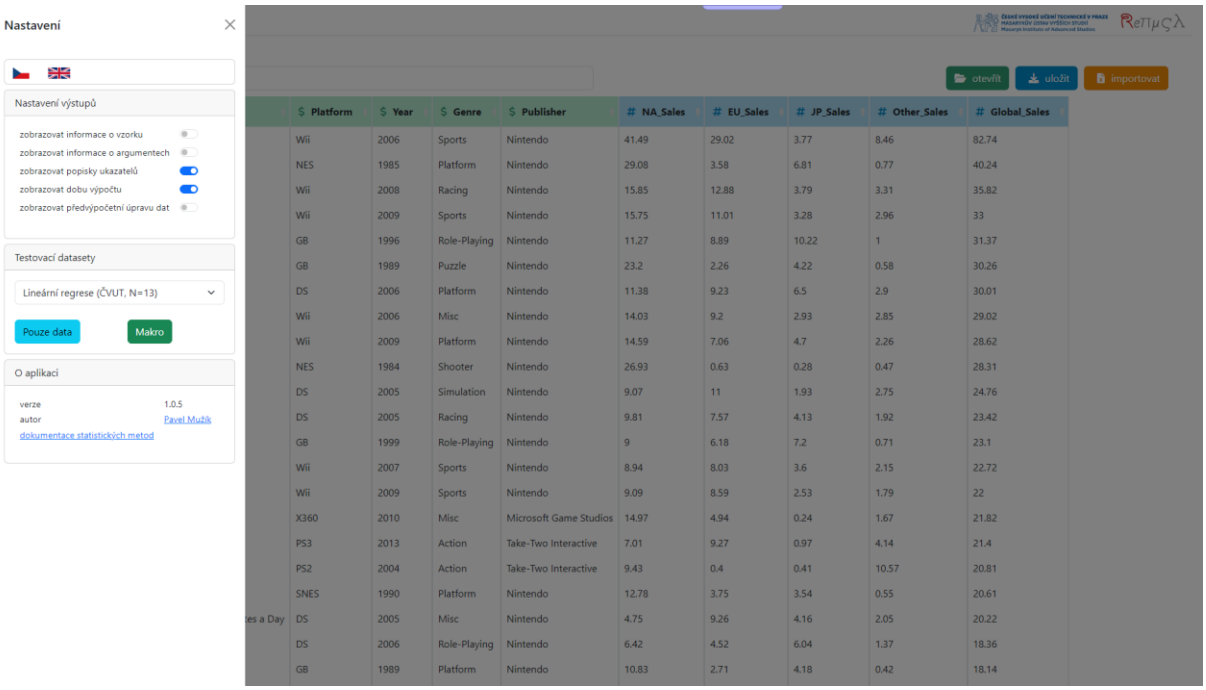
iPhone

2.2.2 Hlavní komponenty

2.2.2.1 Panel nastavení

Ikonou ozubeného kolečko v levém horní rohu lze zobrazovat (a skrývat) nabídku neanalytických funkcionalit aplikace.

OBRÁZEK 10: RETUSA GUI - PANEL NASTAVENÍ



Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33
GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1	31.37
GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31
DS	2005	Simulation	Nintendo	9.07	11	1.93	2.75	24.76
DS	2005	Racing	Nintendo	9.81	7.57	4.13	1.92	23.42
GB	1999	Role-Playing	Nintendo	9	6.18	7.2	0.71	23.1
Wii	2007	Sports	Nintendo	8.94	8.03	3.6	2.15	22.72
Wii	2009	Sports	Nintendo	9.09	8.59	2.53	1.79	22
X360	2010	Misc	Microsoft Game Studios	14.97	4.94	0.24	1.67	21.82
PS3	2013	Action	Take-Two Interactive	7.01	9.27	0.97	4.14	21.4
PS2	2004	Action	Take-Two Interactive	9.43	0.4	0.41	10.57	20.81
SNES	1990	Platform	Nintendo	12.78	3.75	3.54	0.55	20.61
DS	2005	Misc	Nintendo	4.75	9.26	4.16	2.05	20.22
DS	2006	Role-Playing	Nintendo	6.42	4.52	6.04	1.37	18.36
GB	1989	Platform	Nintendo	10.83	2.71	4.18	0.42	18.14

Menu obsahuje čtyři sekce:

- **Lokalizace**
  - aktuálně v českém a anglickém jazyce, funguje v reálném čase, tzn. automaticky se přepisuje (s výjimkou popisků grafů, viz Doplňky) veškerý, tedy i existující obsah. Tato funkcionalita je řešena pomocí speciálních atributů (__text, __title apod.), které jsou při generování HTML zapisovány do jednotlivých elementů a odkazují se na konkrétní kódy jazykových mutací. V případě jQuery událostí změn jsou příslušné elementy upraveny.
- **Nastavení výstupů**
  - Slouží k uživatelské specifikace, které výstupy se mají zobrazovat a které ne
- **Testovací datasety**
  - Aktuálně slouží zejména k rychlejšímu testování vývoje, ovšem podobné testovací datasety má ve své základní konfiguraci většina aplikací pro statistickou analýzu, neboť začínajícím uživatelům pomáhají lépe pochopit požadovanou strukturu datové zdroje pro vybrané metody.

- **O aplikaci**
    - Informace o verzi, autorovi a odkaz na dokumentaci v příslušeném jazyce na GitHub

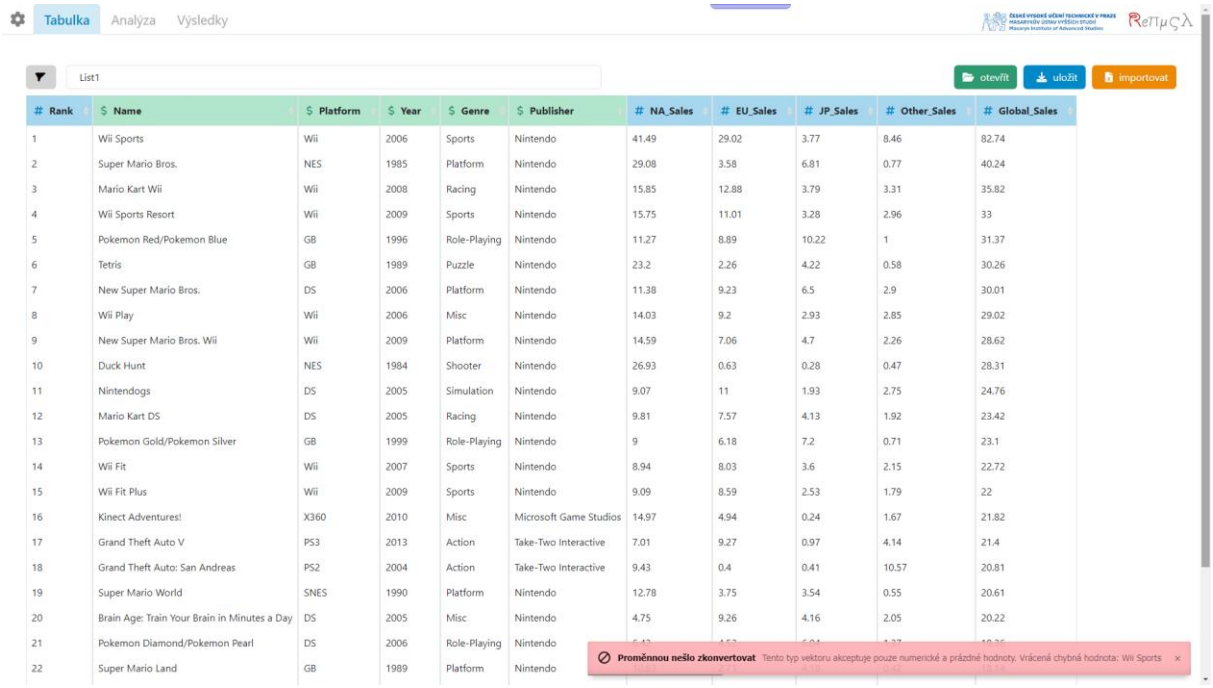
2.2.2.1.1 Jazykové mutace

Vzhledem k tomu, že základní knihovna RTS podporuje vícejazyčnost, zobrazují se v RUI převzaté hodnoty ve zvoleném jazyce (vývoj probíhá v české mutaci, po dokončení prototypu bude doplněna angličtina).

2.2.2.1.2 Chybové a jiné zprávy

Asynchronní komunikace s uživatelem je zajištěna pomocí zobrazování zpráv v pravém dolním rohu. Vykreslování zpráv zajišťuje knihovna iziToast (32). Většina potenciálních chyb vzniká při zadání nesprávného argumentu statistické metody. Validaci argumentů provádí do detailní úrovně jádrová aplikace (RTS). Text chyby je jazykově lokalizován a poměrně přesně popisuje danou chybu. Uživateli RUI se tedy obvykle zobrazují chyby nikoliv z RUI, ale ty vyvolané RTS.

OBRÁZEK 11: RETUSA GUI - PŘÍKLAD CHYBOVÉHO HLÁŠENÍ



# Rank	$ Name	$ Platform	$ Year	$ Genre	$ Publisher	# NA_Sales	# EU_Sales	# JP_Sales	# Other_Sales	# Global_Sales
1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33
5	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1	31.37
6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
7	New Super Mario Bros.	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
8	Wii Play	Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
9	New Super Mario Bros. Wii	Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31
11	Nintendogs	DS	2005	Simulation	Nintendo	9.07	11	1.93	2.75	24.76
12	Mario Kart DS	DS	2005	Racing	Nintendo	9.81	7.57	4.13	1.92	23.42
13	Pokemon Gold/Pokemon Silver	GB	1999	Role-Playing	Nintendo	9	6.18	7.2	0.71	23.1
14	Wii Fit	Wii	2007	Sports	Nintendo	8.94	8.03	3.6	2.15	22.72
15	Wii Fit Plus	Wii	2009	Sports	Nintendo	9.09	8.59	2.53	1.79	22
16	Kinect Adventures!	X360	2010	Misc	Microsoft Game Studios	14.97	4.94	0.24	1.67	21.82
17	Grand Theft Auto V	PS3	2013	Action	Take-Two Interactive	7.01	9.27	0.97	4.14	21.4
18	Grand Theft Auto: San Andreas	PS2	2004	Action	Take-Two Interactive	9.43	0.4	0.41	10.57	20.81
19	Super Mario World	SNES	1990	Platform	Nintendo	12.78	3.75	3.54	0.55	20.61
20	Brain Age: Train Your Brain in Minutes a Day	DS	2005	Misc	Nintendo	4.75	9.26	4.16	2.05	20.22
21	Pokemon Diamond/Pokemon Pearl	DS	2006	Role-Playing	Nintendo					
22	Super Mario Land	GB	1989	Platform	Nintendo					

2.2.3 Tabulka

Tabulkové rozhraní je svým způsobem synonymem, přesněji možná symbolem, aplikací pro statistickou analýzu. Paradoxně nemusí být tabulka pro uživatele důležitá a v některých případech, např. při analýze databázových dat o milionech řádcích, slouží pouze jako náhled na typ dat. Nicméně s ohledem na MVP bylo tabulkové rozhraní nejen pevně danou položkou, ale stalo se i důležitým rozhraním pro samotnou analýzu (vektorů), čímž se liší od ostatních statistických softwarů.

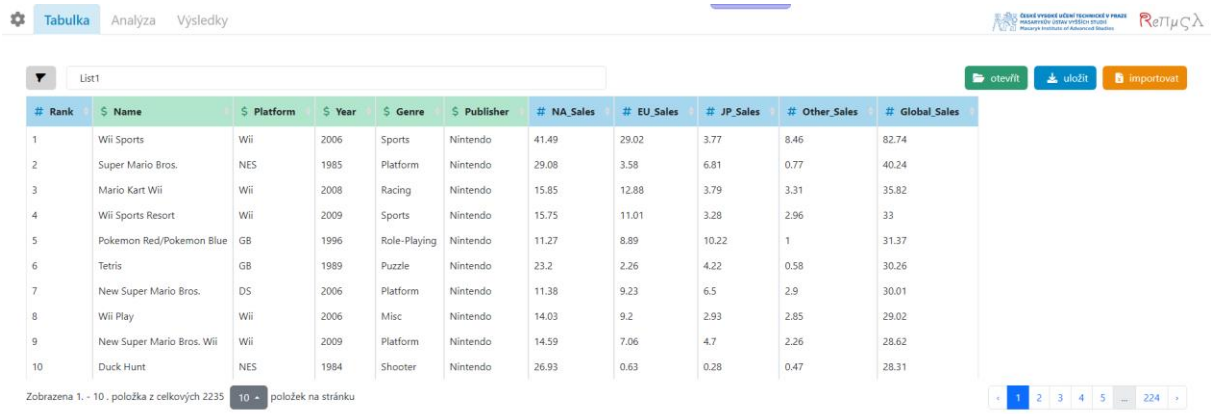
2.2.3.1 Soubory

RUI, kromě podpory importování souborů typu XLSX , využívá svůj vlastní formát pro ukládání a načítání dat. Soubor s příponou **ret** je textovou kopií JSON objektu, obsahujícího serializovanou matici s možností přidávání dalších vlastností v budoucnu (např. logy vyvolaných metod s jejich argumenty pro replikaci). Aktuálně není využíváno žádné komprimace[22]. Soubor kromě samotné zdrojové matice obsahuje i metadata, jako je název tabulky, hodnoty filtrů jednotlivých proměnných atd.

2.2.3.2 Zobrazení dat

Samotná tabulka je založena na knihovně bootstrap-table (33), která rozšiřuje komponentu Table v Bootstrap o řadu funkcí (třídění dat, řádků, sloupců atd.). Pro optimalizaci využití paměti jsou data tabulky načítána metodou ***ajax***, rozšiřující původní třídu Matrix v RTS, čímž se zamezuje duplicitnímu ukládání souborů dat to paměti. Podpora AJAX komunikace je původní vlastností knihovny bootstrap-table a je typicky využívána zejména pro komunikaci se serverem. Tato vlastnost tedy za určitých podmínek umožňuje v prohlížeči uchovávat pouze data, která může uživatel fyzicky vidět na obrazovce (konkrétní výběr maximálně několika desítek řádků), zatímco ve struktuře serveru mohou být k dispozici miliony řádků. To znamená, že pokud by se aplikace rozdělila striktně na front-end a back-end, mohl by uživatel pomocí asynchronních metod se zpětným voláním provádět paralelně větší množství výpočetně náročných úkonů, jelikož paměť by nezatěžovaly velké datasety.

OBRÁZEK 12: RETUSA GUI - PŘÍKLAD TABULKY DAT



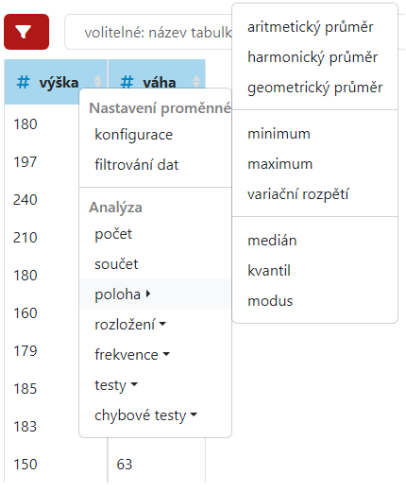
# Rank	$ Name	$ Platform	$ Year	$ Genre	$ Publisher	# NA_Sales	# EU_Sales	# JP_Sales	# Other_Sales	# Global_Sales
1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33
5	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1	31.37
6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
7	New Super Mario Bros.	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
8	Wii Play	Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
9	New Super Mario Bros. Wii	Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31

2.2.3.3 Kontextové menu

Nastavení vektorů a přistupování k jeho statistickým funkcím probíhá pomocí kontextového menu v podobě widgetu (miniaplikace), který se zobrazí po najetí kurzorem myši na záhlaví sloupce a stisknutí pravého tlačítka myši. Kontextové menu je hierarchický strom, kterým lze jak rozevřít nastavení vektoru (a tak upravit název či ho konvertovat), tak nastavení filtru (viz dále) a konečně také spouštět jednotlivé statistické metody.

[22] Soubory *.xlsx Microsoft Excel i dalších aplikací Office jsou de facto zazipované adresáře.

OBRÁZEK 13: RETUSA GUI - KONTEXTOVÉ MENU VEKTORU

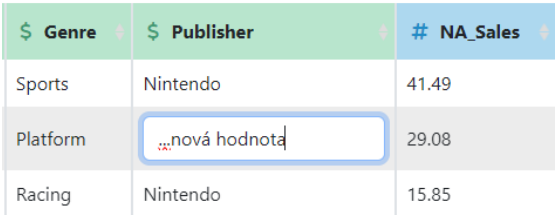


2.2.3.4 Editace hodnot

Knihovna bootstrap-table sice nabízí extenzi pro editaci hodnot, která ovšem k únoru 2023 nebyla funkční. Z tohoto důvodu rozšiřuje knihovna RUI původní balíček (33) o funkcionalitu editace, která mj. využívá validaci na straně jádrové knihovny RTS. Chování aplikace při editování hodnoty funguje následujícím způsobem[23]:

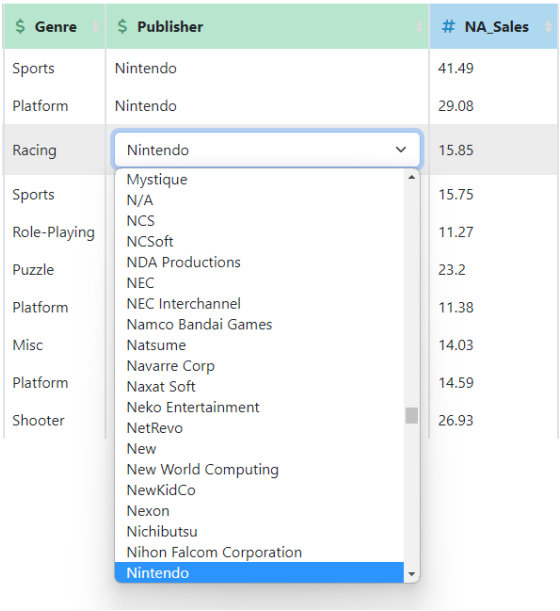
- Kliknutí na buňku – otevře se editor hodnoty (input)
    - Při zmáčknutí *Esc* nebo při události *mousedown* je editor opuštěn a ponechána původní hodnota
    - Při *Enter* je hodnota validována
        - Pokud vyhovuje danému vektoru, je zanesena do buňky
        - Pokud ne, je zobrazeno chybové hlášení a akce je přerušena
    - Při *Ctrl* se změní původní input na kontrolku select s unikátními hodnotami daného vektoru
        - Při výběru je hodnota zanesena do buňky
        - Při *mouseleave* nebo *Esc* podobně jako u a.

OBRÁZEK 14: RETUSA GUI - EDITACE HODNOTY TABULKY POMOCÍ ELEMENTU INPUT



[23] Platí u Windows, u iOS a dalších systémů je potřeba volit odpovídající analaogie.

OBRÁZEK 15: RETUSA GUI - EDITACE HODNOTY TABULKY POMOCÍ ELEMENTU SELECT

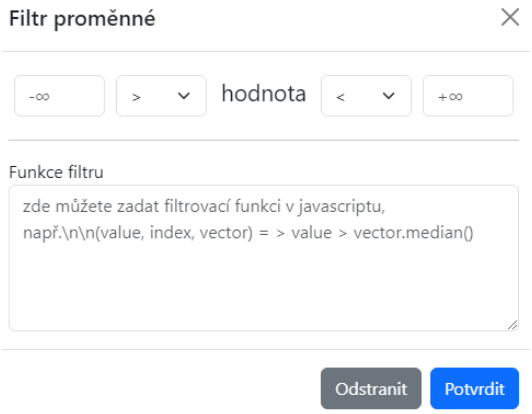


2.2.3.5 Filtry a řazení proměnných

Filtrování proměnných se projevuje jak na úrovni zobrazení (v tabulce), tak ve smyslu vstupu analýz. Funguje podobně jako v aplikaci Excel, tzn. že filtr jedné proměnné se promítá i do filtrování ostatních proměnných – vyřazené řádky z jedné proměnné se vyřadí i z ostatních proměnných a vztah filtrů napříč proměnnými je reciproční.

Nastavení filtrů má dvě úrovně. Na úrovni jednotlivých sloupců (proměnných) se filtr zadává přes kontextové menu (druhá položka). Existují tři typy filtrů: numerický (input), výběrový (select) a funkční (function). Funkční filtr je k dispozici bez ohledu na typ proměnné (vektoru) a funguje stejně jako funkce *filter* u javascriptové instance *Array* (viz příklad níže). Analogií k tomuto způsobu je v SPSS využití syntaxe.

OBRÁZEK 16: RETUSA GUI - UKÁZKA FILTRU PRO NUMERICKÝ VEKTOR

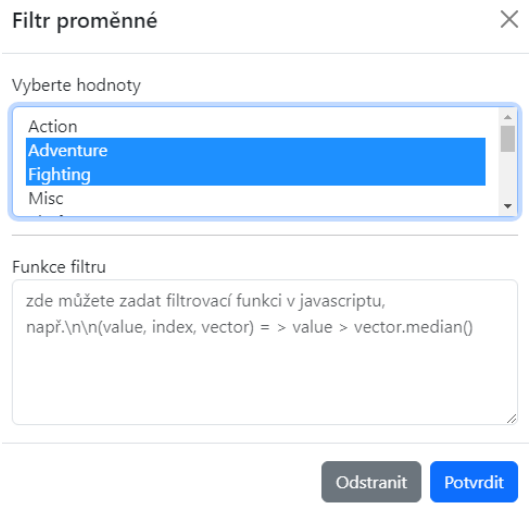


Pro numerické proměnné je defaultně přístupný zároveň filtr typu „range“, tedy výběr mezi dvěma hraničními hodnotami, s možností upřesnit, zdali se jedná o hodnotu rovnou či větší apod. U nominálních a binárních proměnných je defaultně nastaven filtr typu „select“, tedy

výběr z unikátních hodnot v rámci proměnné. Pokud je zadána filtrovací funkce, jsou filtry typu input i select ignorovány.

Filtry zároveň ovlivňují, jaký dataset bude vstupovat do jednotlivých analytických metod (vektorových i maticových); tímto způsobem tak lze např. rychle optimalizovat nastavení vstupní nominální hodnoty T-testu, pokud by daná proměnná měla více než 2 unikátní hodnoty.

OBRÁZEK 17: RETUSA GUI - UKÁZKA VÝBĚROVÉHO FILTRU VEKTORU



Filtry je možné hromadně aktivovat a deaktivovat pomocí tlačítka s ikonkou trychtýře v levé části horního panelu tabulky. Tímto způsobem se filtry pouze hromadně aktivují/deaktivují, avšak jejich nastavení zůstávají zachována, čili je opětovně lze zapnout týmž tlačítkem, aniž by bylo nutné vše znovu nastavovat znovu. Nastavení filtrů se serializuje, při uložení do souboru *.ret a opětovném načtení se tedy obnoví i nastavení filtrů při uložení.

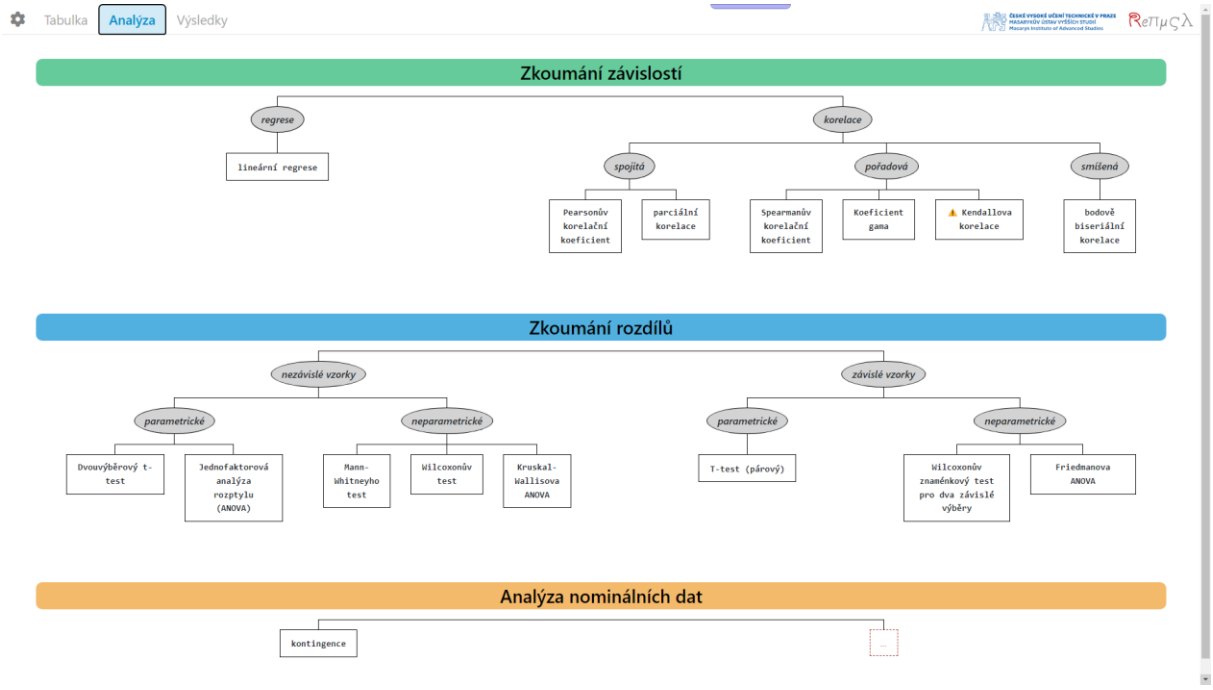
Vektory lze vizuálně řadit (A-Z nebo Z-A), seřazení jedné proměnné má vliv na řazení ostatních proměnných, nemá ovšem vliv na řazení ve zdrojovém objektu. Vestavěnou funkcí knihovny bootstrap-table je stránkování, které efektivně ošetřuje využití paměti tím, že zobrazuje pouze data, která může uživatel prakticky vidět na displeji (dávky lze měnit pomocí nabídky ve spodní liště).

2.2.3.6 Analýza

Záložka analýza je rozhraním pro vizualizaci nabídky maticových metod RTS. Je koncipována jako rozhodovací strom, který uživateli usnadňuje zvolit vhodnou metodu podle analytického cíle i s ohledem na data v tabulce. Vzhledem k rozsáhlým zkušenostem a návykům uživatelů s SPSS jsou části uzlů a jejich metod inspirovány právě touto aplikací - např. skupina Compare means je v anglické verzi RTS nazvána Compare centrality a obsahuje podobné metody.

Jednotlivé maticové metody jsou volány kliknutím na požadovanou metodu (u vektorů kliknutím na položku v kontextovém menu záložky tabulka); pokud nejsou v tabulce žádná, logicky se ani nenabídne dialog pro požadovanou analýzu.

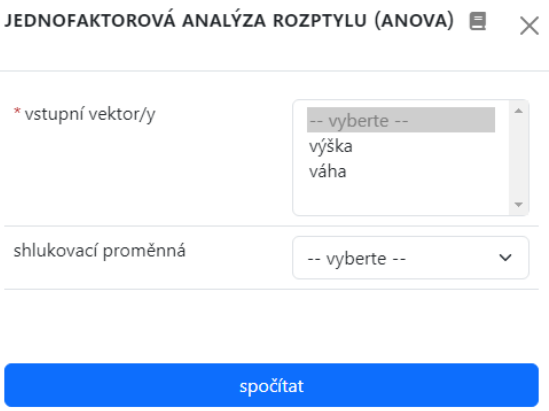
OBRÁZEK 18: RETUSA GUI - STROM ANALÝZY



2.2.3.6.1 Dialogové okno analýzy

Vyjma části vektorových funkcí, které žádný argument nevyžadují (součet, medián atp.) má řada vektorových a všechny maticové metody své vlastní dialogové okno (podobně jako SPSS atd.), pomocí kterého uživatel definuje parametry a vstupy metody. U vektorových metod je vstup předen dán, u maticových metod se vždy vybírají z tabulky konkrétní vektory. Dalším typem argumentů, které uživatel specifikuje, jsou numerace, konkrétní hodnoty – zde se lze odkázat na dokumentaci RTS, neboť dialogová okna jsou dynamicky sestavována právě na základě konstruktoru třídy VectorAnalysis či MatrixAnalysis, který obsahuje všechny podstatné informace.

OBRÁZEK 19: RETUSA GUI - PŘÍKLAD DIALOGOVÉHO OKNA

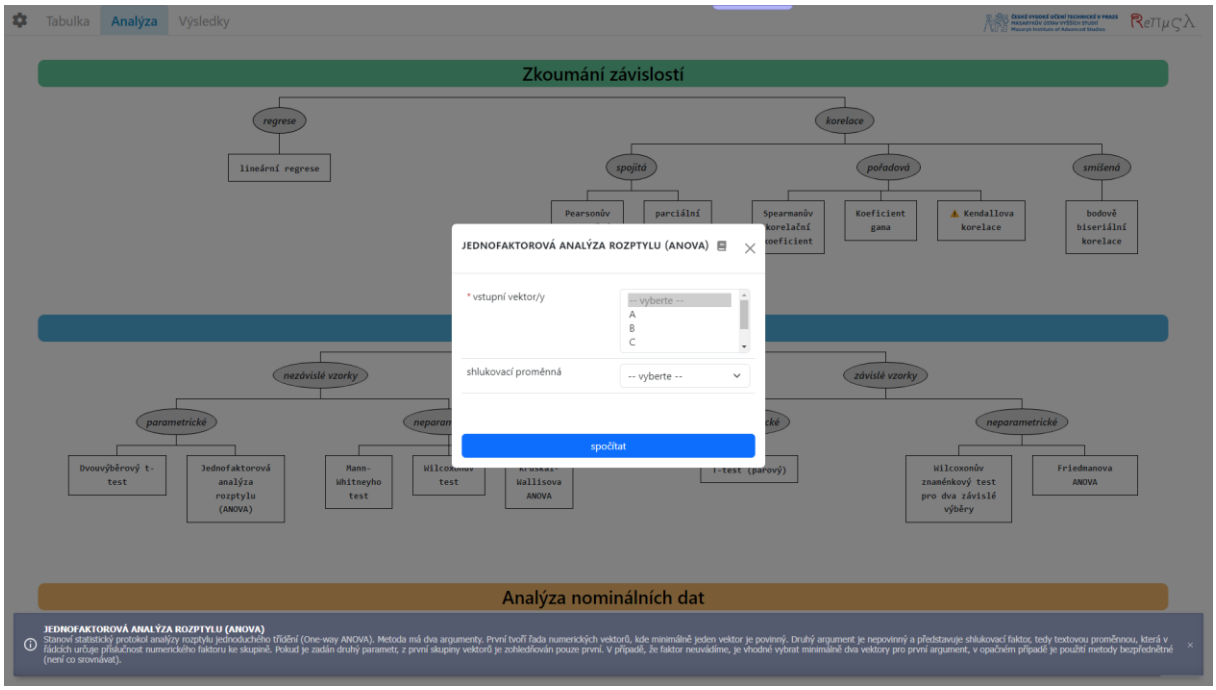


2.2.3.6.2 Kontextová nápověda

Dialog pro zadávání parametrů metody je v naprosté většině případů opatřen nápovědou k dané metodě. Typy nápovědy jsou dva. První popisuje obecně danou metodu (viz strukturu

dokumentace dříve) a její zobrazení je možné pomocí kliknutí na ikonu knihy vpravo od názvu metody; nápověda se zobrazí v dolní liště plochy. Druhý typ nápovědy se vztahuje ke konkrétním parametrům a nápověda je dostupná, pokud je u názvy metody ikona otazníku – tehdy je možné ji zobrazit najetím myší. Povinné parametry jsou označeny červenou hvězdičkou u svého názvu.

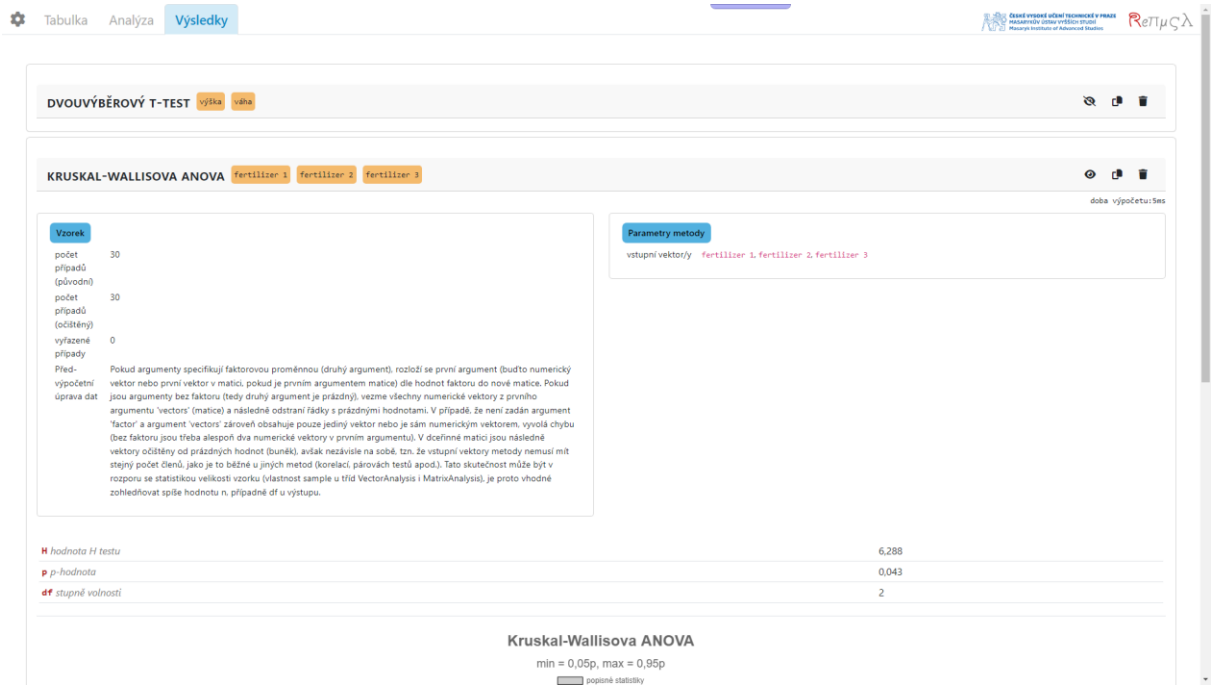
OBRÁZEK 20: RETUSA GUI - UKÁZKA NÁPOVĚDY K MATICOVÉ METODĚ



2.2.3.7 Výsledky

Záložka *Výsledky* funguje jako kontejner graficky zpracovaných výstupů jednotlivých metod, podobně jako je tomu u SPSS a dalších aplikací (s tím rozdílem, že zde není využíváno vlastní okno). Dílčí výstupy se zobrazují coby karty, v jejichž záhlaví se uvádí kromě názvu metody a vstupních vektorů (pokud se jedná o výstup maticové metody) také nástroje k minimalizování, kopírování a smazání karty. Každá karta se skládá z fixní a variabilní části. Fixní část obsahuje widget **Vzorek**, který popisuje velikost vstupního a analyzovaného souboru a také popi před-výpočetní operace, dále widget **Parametry metody**, který rekapituluje parametry funkce definované uživatelem (případně definovaná sama sebou, pakliže mají předurčenu hodnotu) a samotný **výsledek** metody, jehož grafická struktura se řídí instrukcemi RTS. Variabilní část se skládá z rozšiřujících komponent v souboru addons.js (viz dále). Oba dva dříve uvedené widgety lze dle potřeby zobrazit či skrýt (viz Panel nastavení).

OBRÁZEK 21: RETUSA GUI - ZÁLOŽKA VÝSLEDKŮ



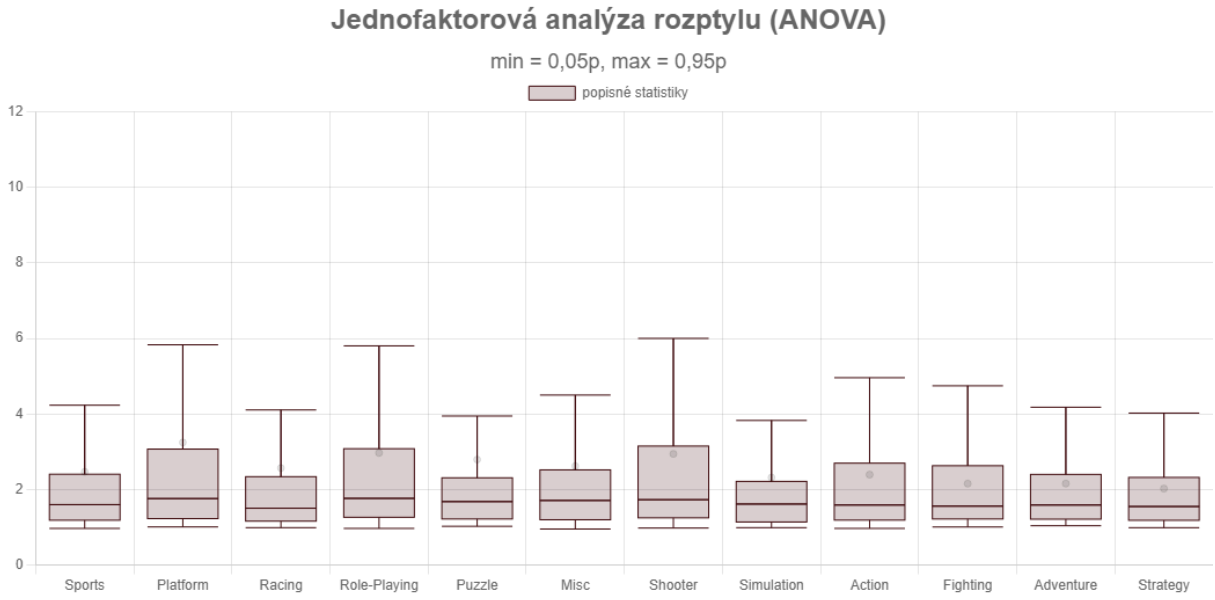
2.2.3.8 Doplňky

Knihovna addons.js zahrnuje balíček rozšiřujících operací pro vybrané metody, kde operacemi míníme zobrazení grafu, vytvoření tabulky, kalkulačky apod. Knihovnu mohou programátorsky zdatní uživatelé rozšiřovat o vlastní doplňky. Pevně daným vstupem každé rozšiřují metody je instance VectorAnalysis či MatrixAnalysis, která v sobě nese všechna potřebná data k další analýze, včetně vstupního i očištěného souboru. V aktuálních balíčku jsou obsaženy následující typy doplňků (týká se jak maticových, tak vektorových metod).

2.2.3.8.1 Grafy

Pro vykreslování grafů je používána knihovna ChartJS (34); jedná se o front-end balíček se základní sadou běžně užívaných grafů (sloupcový, koláčový apod.) a možností uživatelského rozšiřování (díky tomu lze RUI zobrazovat i krabičkové grafy). Samotné výstupy mají publikovatelnou kvalitu, příjemným zpestřením pro uživatele je pak jejich interaktivnost, kdy lze např. najetím myši na bod zobrazit jeho souřadnice, dle potřeby lze skrývat řady apod.

OBRÁZEK 22: RETUSA GUI - PŘÍKLAD DOPLŇKU TYPU GRAF



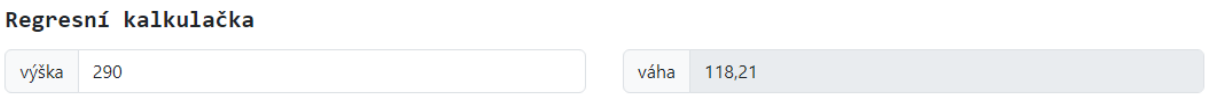
2.2.3.8.2 Tabulky

Pro vybrané metody (např. kontingence) je běžné k výsledkům přidávat agregované matice, které umožní lepší vhled do dat. Tabulky jsou zpracovávány pouze za použití komponent Bootstrap a funkcionalit jQuery. Výstupem některých metod (např. tabulky četností či histogramu) je samotná tabulka, která funkci doplňků nevyužívá.

2.2.3.8.3 Kalkulačky

Kalkulačky umožňují simulovat určité scénáře, typicky u regresních modelů, jejichž výstupem v RTS je i regresní funkce. Aktuálně je doplněk typu kalkulačka integrován u jediné regresní metody (lineární). Kalkulačka funguje jednoduše tak, že do příslušných polí se zadá hodnota jedné či více nezávislých proměnných a v sousedním poli se v reálném čase dopočítává výsledek.

OBRÁZEK 23: RETUSA GUI - PŘÍKLAD DOPLŇKU TYPU KALKULAČKA

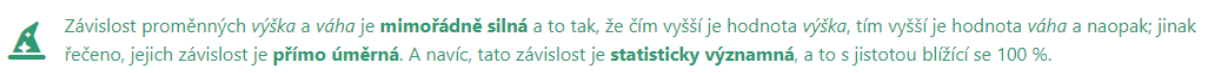


2.2.3.8.4 Humanizer

Koncept humanizer vychází přímo z uživatelského testování (a shodou okolností z oné více jak 30 let staré diskuze), kdy uživatelé, bez ohledu na pokročilost, požadují „polidštění“ výstupů analýz. Humanizer je funkce, která interpretuje výsledky, podobným způsobem, jako dříve zmíněná aplikace EZ Statistics (22), s tím rozdílem, že se snaží o co nejpřirozenější formulaci. Koncept humanizer respektuje vícejazyčnost a jeho architektura tak dokáže v reálném čase (při změně lokalizace) automaticky převést interpretované výsledky do požadovaného jazyka.

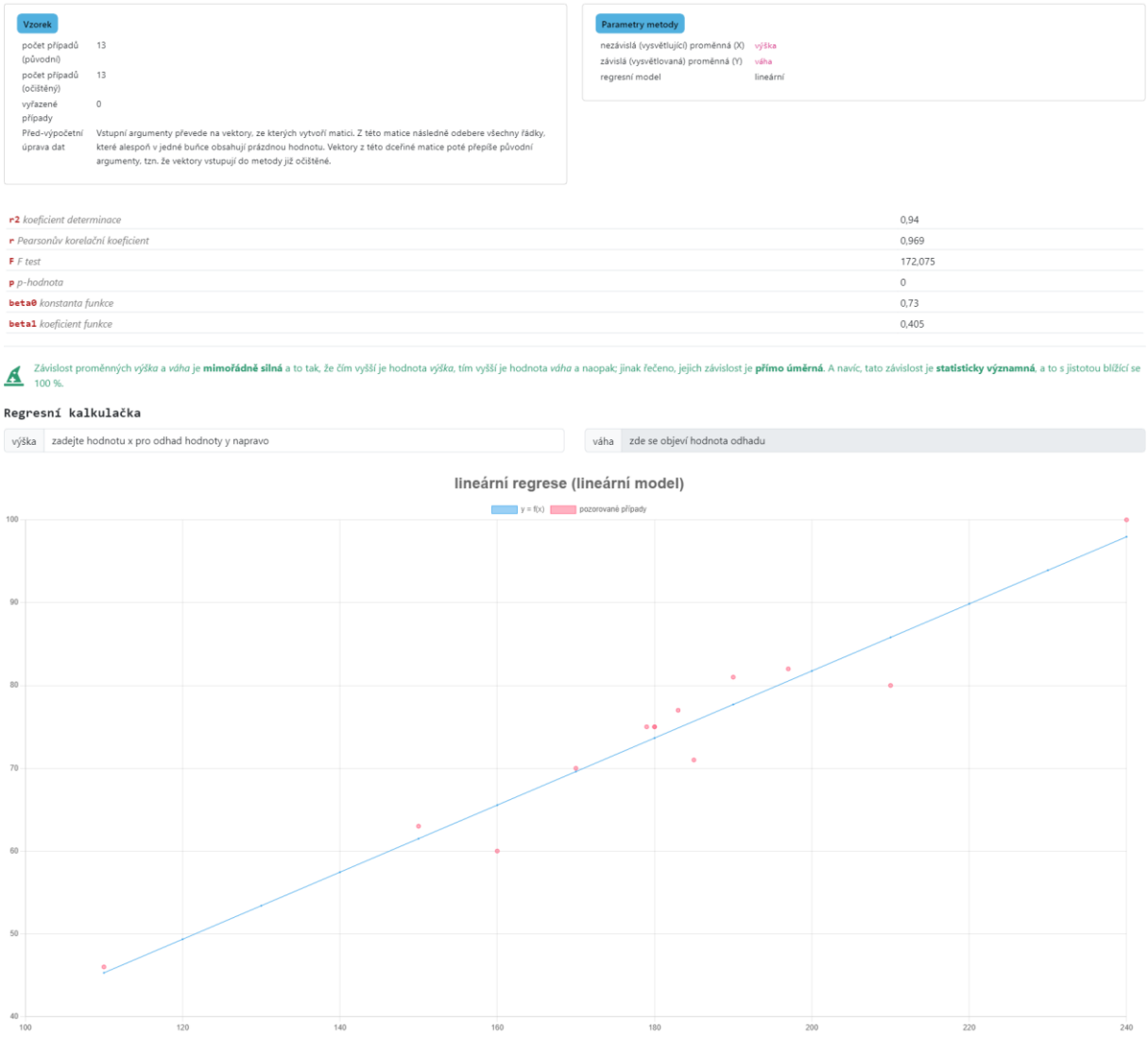
Aktuálně je tento doplněk zpracován pouze pro Pearsonův korelační koeficient (resp. i lineární regresi, jejímž dílčím výstupem tato metoda je), plánuje se ovšem postupné programování balíčků i pro ostatní metody.

OBRÁZEK 24: RETUSA GUI - PŘÍKLAD DOPLŇKY TYPU HUMANIZER



Příklad[24] lineární regrese níže ukazuje všechny čtyři typy doplňků v jednom výstupu. S výjimkou grafů jsou veškeré výstupy citlivé na změnu lokalizace (jazyka), proto se při jeho změně uživatelem automaticky přeloží, a to bez vazby na jakékoliv jiné knihovny či zdroje (jako je třeba Google API).

OBRÁZEK 25: RETUSA GUI - PŘÍKLAD DOPLŇKŮ



[24] Z důvodu přehlednosti byly z výstupu odtraněny informace o vzorku a argumentech.

2.2.4 Aplikační vztahy

Vzhledem k tomu, že RUI ve svém vývoji maximálně zohledňovala architekturu RTS a její možnosti (a zároveň dávala zpětně návrhy k vylepšení), existuje řada příkladů, kdy se některé inovace v RTS organicky promítly do designu RUI. Typických příkladem je ošetření způsobu zadávání argumentů u metod typu ANOVA, nezávislý T-test, Mann-Whitneyho test apod. Ve dvou důležitých referenčních aplikacích, SPSS a Statistica, je nutné zadávat dva argumenty: závislou a faktorovou proměnnou. Přitom není důvod, aby uživatel nemohl zadat onu závislou proměnnou jako balíček vektorů (tedy definovat přímo jednotlivé skupiny – proměnné). Obě dvě aplikace Retusa proto umožňují volnější práci s argumenty, jak bylo uvedeno dříve, přičemž RUI koncepty RTS pouze "přenáší na obrazovku". Podobně je v referenčních softwarech nutné u T-testu pro dva nezávislé výběru (a podobných metod) zadávat dvě faktorové hodnoty, a to i v případě, že proměnná faktoru přesně dvě hodnoty má. V Retusa lze tento výběr buďto ponechat na samotné aplikace, anebo lze v RUI ve filtru hodnoty vybrat ručně.

Zopakujme princip závislostí těchto knihoven. **Retusa (RTS) je zcela nezávislá na Retusa GUI** a může jí používat libovolný uživatel buďto v rámci vlastních projektů v Node.js, případně ve webových aplikacích pomocí balíčku *bundle.min.js*. **Retusa GUI je naopak na RTS zcela závislá**, neboť nejen, že pomocí ní vypočítává statistické metody, ale také z ní čerpá většinu textů, používá jejích nástrojů validace atd. Při teoretické úvaze, zdali vývoj jedné z aplikací ukončit je tedy v obou případech nutné zohledňovat právě tyto faktory.

2.3 Inovativnost řešení

Na závěr doposud převážně technického popisu aplikace je třeba shrnout inovace, které obě popsaná řešení přináší do segmentu softwaru pro statistickou analýzu. Vzhledem k velikosti projektu (ve smyslu času a zdrojů) byly inovační plány, na rozdíl od vize popsané v závěrečné kapitole, poměrně skromné. Část inovací vycházela ze zkušeností s existujícími aplikacemi, náměty na vývoj rovněž přicházely od testerů RUI.

V rámci Node.js řešení přináší Retusa inovativní přístupy zejména v následujících oblastech:

- Koncept vektorů a matic založený na vlastních třídách, s možností ukládání metadat
- Systematické ošetření validace hodnot a argumentů na úrovni aplikace samotné
- Rozšíření životního cyklu statistické analýzy pomoci tříd VectorAnalysis a MatrixAnalysis (mj. možnost krokového provádění analýzy)
- Lokalizaci výstupů i vnitřního prostředí (např. vracení chyb v nastaveném jazyku)
- Víceúčelové využívání vestavěných prvků (např. popisy argumentů a výstupů jsou používány pro generování souborů dokumentace i jako komunikační prostředek s uživatelem v RUI)
- Architektura je navržena pro přidávání dalších metod třetími stranami

Jelikož je RUI do určité míry definována možnostmi RTS, jsou vybrané prvky uvedené výše vlastní i této front-end aplikaci. K dalším inovativním vlastnostech RUI patří:

- Uživatelsky orientované prostředí (množství nápovědy, interpretace ukazatelů, koncept Humanizer, redukce nutnosti častých překlikávání apod.)
- Změna lokalizace v reálném čase (u většiny známých aplikací, podporující lokalizaci, je třeba aplikaci před projevením změn restartovat, u RUI nikoliv)
- Koncept all-in-one-place (uživatel pracuje na jedné pracovní ploše, bez nutnosti vracení se na předchozí stránku apod.)

2.4 Provozní test a validace

Podstatným milníkem v celkovém vývoji řešení je testování, které bylo rozděleno na dvě části. První skupinu tvořilo technické testování aplikace a zaměřilo se na provozní (zátěžové) testy a také na ověření samotných výsledků formou porovnání s výstupy z existujících řešení. Druhou skupinu představuje uživatelské testování RUI a je popsáno v poslední kapitole této práce.

2.4.1 Provozní test

Měření výkonnosti (statistik doby zpracování) bylo provedeno u maticových metod, které jsou výpočetně zpravidla náročnější (z hlediska algoritmů, počtu navazujících operací atd.) než metody vektorové. Testování probíhalo vždy v sérii N iterací (smyček), kdy obsah jedné iterace představovalo vygenerování vstupní matice (není započteno do statistik) a následný výpočet. Výstupem každé iterace byla vždy kompletní struktura výsledku, tj. včetně výpočtu p-hodnoty a dalších testů (př. korelační koeficient u lineární regrese apod.) a také implementace předvýpočetní úpravy dat – přesněji řečeno, výstupem byla instance třídy MatrixAnalysis. Z výstupu iterace byla pro statistickou analýzu, provedenou fyzicky v RUI, využita pouze doba zpracování každé iterace. Samotné testování bylo zautomatizované v jádrové aplikaci (modul stress-test.js). Veškeré výsledky uvedené níže byly tedy vygenerovány a následně zpracovány pouze pomocí dvou aplikací popisovaných v této práci.

TABULKA 5: SPECIFIKACE TESTOVACÍHO POČÍTAČE

Parametr	**Hodnota**
Operační systém	Microsoft Windows 11 Home (10.0.22621 Build 22621)
Procesor	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 2592 Mhz, jádra: 6, logické procesory: 12
BIOS	LENOVO EFCN31WW
Typ systému	x64-based PC
Fyzická RAM	16 Gb
V8 (verze)	11.0.226.16
Node.js (verze)	18.12.1

TABULKA 6: SPECIFIKACE METOD A PARAMETRŮ

metoda	**iterace**[25]	**velikost matice**[26]	**konfigurace matice**[27]
ANOVA	10.000	10.000	1x numerický vektor 1x nominální vektor (faktorová proměnná, 5 možných hodnot)
Biseriální korelace	10.000	10.000	1x binární vektor 1x numerický vektor
Friedmanova ANOVA	10.000	10.000	5x numerický vektor
Kendallova korelace	10.000	10.000	2x numerický vektor
Koeficient gamma (Goodman-Kruskal)	10.000	10.000	2x numerický vektor
Kontingence	10.000	1.000	1x nominální vektor (řádky, 10 možných hodnot) 1x nominální vektor (sloupce, 10 možných hodnot)
Kruskal-Wallisova ANOVA	1.000	10.000	1x numerický vektor 1x nominální vektor (faktorová proměnná, 5 možných hodnot)
Lineární regrese	10.000	10.000	2x numerický vektor
Mann-Whitneyho test	1.000	500	2x numerický vektor
Parciální korelace	10.000	10.000	3x numerický vektor
Pearsonův korelační koeficient	10.000	10.000	2x numerický vektor
Spearmanův korelační koeficient (N=1.000)	10.000	1.000	2x numerický vektor
Spearmanův korelační koeficient (N=10.000)	1.000	10.000	2x numerický vektor
T-test (nezávislý)	10.000	10.000	1x numerický vektor 1x nominální vektor (faktorová proměnná o dvou možných hodnotách)
T-test (závislé výběry)	10.000	10.000	2x numerický vektor
Wilcoxonův test pro dva nezávislé výběry	10.000	1.000	1x numerický vektor 1x nominální vektor (faktorová proměnná o dvou možných hodnotách)
Wilcoxonův znaménkový test pro závislé výběry	1.000	1.000	2x numerický vektor

[25] Počet jedinečných opakování výpočtu metody; metoda je vždy počítána s jiným vzorkem.
[26] Počet případů v matici.
[27] Hodnoty vektorů jsou náhodně generované pomocí funkce *generate*.

TABULKA 7: VÝSLEDKY PROVOZNÍCH TESTŮ (DOBA ZPRACOVÁNÍ V MILISEKUNDÁCH)

Metoda	N[28]	průměr	chyba[29]	variační koeficient	medián	minimum	maximum
ANOVA	10.000	144	0,33	0,09	141	112	335
Biseriální korelace	10.000	31,7	0,12	0,15	30	27	80
Friedmanova ANOVA	10.000	40	0,12	0,11	39	34	97
Kendallova korelace	10.000	49,9	0,1	0,08	49	42	119
Koeficient gamma (Goodman-Kruskal)	10.000	76,1	0,1	0,05	76	66	120
Kontingence	10.000	71,7	0,25	0,14	70	48	212
Kruskal-Wallisova ANOVA	1.000	2 184	8,8	0,05	2 177	1 919	3 029
Lineární regrese	10.000	33,7	0,06	0,07	33	30	101
Mann-Whitneyho test	1.000	385,2	1,84	0,06	379	351	596
Parciální korelace	10.000	110,4	0,3	0,11	107	97	272
Pearsonův korelační koeficient	10.000	27,2	0,04	0,06	27	24	60
Spearmanův korelační koeficient (N=1.000)	10.000	27,7	0,13	0,18	27	16	76
Spearmanův korelační koeficient (N=10.000)	1.000	3 131	18	0,06	3 106	2 688	4 134
T-test (nezávislý)	10.000	81,5	0,3	0,14	78	70	191
T-test (závislé výběry)	10.000	25,4	0,08	0,12	24	22	64
Wilcoxonův test pro dva nezávislé výběry	10.000	181,8	0,22	0,05	180	164	286
Wilcoxonův znaménkový test pro závislé výběry	1.000	237	1,43	0,07	232	215	404

Výpočetní metody jsou z běžného uživatelského hlediska rychlé, dobu zpracování je možné vyjádřit v desítkách, maximálně málo stovkách milisekund, při nízké variabilitě (viz chybu nebo variační koeficient). Viditelná latence u RUI oproti výše uvedeným statistikám není způsobena rozdíly v době výpočtu, ale v době potřebné ke kompletnímu vykreslení výsledku na straně prohlížeče[30]. Některé metody vykazují výrazný nárůst v požadavcích na výpočetní kapacitu při změně velikosti vzorku. Tento případ je evidentní u Spearmanova korelačního koeficientu, kdy nárůst 1.000 na 10.000 analyzovaných případů zvyšuje výpočetní dobu více než stonásobně. U podobných případů – tedy zejména metod, které pracují s nominálními a pořadovými proměnnými - bude vhodné, s ohledem na uživatelskou přívětivost, hledat možnosti vylepšení algoritmu výpočtu. U těchto metod byl rovněž používán buďto menší vzorek vstupních případů, nebo byl snížen počet iterací.

2.4.2 Validace výsledků

Více než zátěžové testy bylo v rámci testování zásadní ověřit relevantnost výsledků metod, neboť přesnost (či lépe relevance) výstupů je z uživatelského hlediska zcela klíčová. Metodicky byla validace provedena tak, že nad konkrétním souborem dat byl napříč několika aplikacemi proveden výpočet pro danou metodu a následně byly porovnány výsledky jednotlivých metrik (testů apod.), které jsou výstupem dané metody u RTS. Do testovací sady byly zařazeny tři běžně používané aplikace (SPSS, Statistica, JASP) a jako doplňkový zdroj sada příkladů počítaných v předmětu Statistická analýza (viz tabulku níže). Některé validace metod nebyly přístupné ve všech kontrolních zdrojích a tyto jsou označeny vodorovnou čárkou.

[28] Počet hodnocených iterací.

[29] Velikost intervalové chyby odhadu průměru při hladině významnosti p = 0,01.

[30] Zpoždění je programaticky nastaveno tak, aby se uživateli zobrazili až kompletně renderované výsledky, včetně příloh (grafů, kalkulaček atd.).

TABULKA 8: ZDROJE VALIDACE

Zkratka	Aplikace	Popis
SPSS	IBM SPSS Statistics	Verze 20
Statistica	TIBCO Statistica	verze 14 (zkušební)
JASP	JASP	Verze 0.17.1 (desktop)
MÚVS	Výpočty prováděny v MS Excel 365	Příklady počítané v rámci předmětu Statistická analýza (K63C1102) ve studijním programu Projektové řízení inovací na Masarykově ústavů vyšších studií ČVUT; byly porovnávány všechny relevantní výstupy metod.

TABULKA 9: VÝSLEDKY VALIDACE

Metoda/zdroj srovnání	SPSS	Statistica	Jasp	MÚVS
ANOVA	✓	✓	✗	✓
Biseriální korelace	✓	✓	✓	-
Friedmanova ANOVA	✓	✓	-	-
Kendallova korelace	✓[31]	✓	-	-
Koeficient gamma (Goodman-Kruskal)	✓	✓	-	-
Kruskal-Wallisova ANOVA	✓	✓	✓	-
Kontingence	-[32]	-[33]	✓	✓
Lineární regrese	✓	✓	✓	✓
Mann-Whitneyho test	✓	✓[34]	-	-
Parciální korelace	✓	✓	✗[35]	-
Pearsonův korelační koeficient	✓	✓	✓	✓
Spearmanův korelační koeficient	✓[36]	✓	-[37]	✓
T-test (nezávislý)	✓	✓	✓	-
T-test (závislé výběry)	✓	✓	✓	-
Wilcoxonův test pro dva nezávislé výběry	✓	-	-	-
Wilcoxonův znaménkový test pro závislé výběry	✓	✓	-	-

Hodnota srovnávací tabulky výše má vícero účelů. Kromě samotné validace (kde jsou, zejména z hlediska konkurenceschopnosti RTS, klíčové shody s prvními dvěma aplikace) ukazuje přidanou hodnotu vůči relevantní konkurenci (horizontálně), tedy jaké funkcionality navíc

[31] Mírně se liší p-hodnota (oproti SPSS i Statistica) vzhledem k odlišnému výpočtu standardizované chyby, která nezohledňuje možná opakované pořadí (ties). Do budoucna se počítá s úpravou.

[32] Nebylo validováno Cramérovo V ani Pearsonovo V, podobně u Jasp. Shodný je χ^2 test a p-hodnota.

[33] Zpracování statistik kontingence je ve Statistice natolik procedurálně komplikované, že se výsledky nepodařilo porovnat.

[34] Hodnota U de facto totožná, pouze s opačným znaménkem.

[35] Nižší hodnota koeficientu než u RTS/RUI/SPSS.

[36] Minimální rozdíl způsobený dosud neintegrovaným zohledněním opakovaných pořadí.

[37] JASP sice výpočet Spearmanova i Kendallova koeficientu nabízí, vrací ovšem hodnotu NaN, tj. zřejmě dochází na straně aplikace k chybnému parsování číselných hodnot (identický zdroj funguje bez problémů jak u SPSS, tak u Statistica).

oproti nejbližším alternativám nabízí (zde je příkladem této konkurence Jasp). Při plánování dalšího vývoje bude klíčové pečlivě plánovat jednotlivé sprinty, jejichž předmět by mělo stanovovat uživatelské šetření, které přesně definuje, která další metoda je pro uživatele prioritní, jaký má být její výstup apod.

3 Komercializace

Závěrečná část této práce se věnuje, volně řečeno, otázce „co s Retusou dále“. Jak ukázala předchozí dokumentace a testy, obě aplikace mají reálnou schopnost rychle a správně spočítat zadání, která jsou běžnou součástí placených aplikací jako je SPSS a naopak nejsou součástí běžných nástrojů, např. Excelu, a dokáže či má předpoklady dokázat daleko více – pokud tomu bude věnován čas a úsilí. Z tohoto důvodu bylo žádoucí se zaobírat kontextem, ve kterém se aplikace nachází; ne ovšem technickým, ale především lidským. Předmětem dalších úvah bude zejména aplikace Retusa GUI, a to nejprve ve smyslu testování, tedy jak ji hodnotí sami potenciální uživatelé, a dále v načrtnutí možného postupu při dalším vývoji a zejména uvedení aplikace do produkční fáze.

3.1 Uživatelské testování

V rámci agilní inovace produktů lze k testování konceptů a prototypů přistupovat několika způsoby, přičemž zmiňme dva, které nesou řadu společných znaků. Prvním je Design Thinking neboli designové myšlení (35), které velmi obecně řečeno vychází z hledání zákaznického problému, který je následně formulován do podoby úkolu. Pro tento obecný úkol jsou následně pomocí různých technik (brainstorming apod. (36)) hledána možná i nereálná řešení, z nichž nejperspektivnější je přetaveno v prototyp (obvykle mock-up aplikace, storyboard apod.) a následně otestováno u zákazníků. Druhým přístupem je Sprint, rozvinutý ve společnosti Google Ventures (37), který je v porovnání s prvním přístupem vhodnější pro podnikové inovace a dává větší důraz na role členů týmu[38]. V obou případech se jedná o přístupy orientované na zákazníka, které cíleně hledají řešení jeho problémů a v obou přístupech je rozhodná zákaznická validace, prováděná zpravidla testováním produktu na straně uživatele.

Původně tato práce nepočítala se vznikem aplikace Retusa GUI; představou autora byla pouze konzole v prohlížeči, do které uživatel bude psát javascriptové sekvence kódu, které budou vracet odpovědi z knihovny rovněž v kódu (resp. formátu JSON) – na takovém produktu není však mnoho co testovat, minimálně z hlediska uživatelů. Pokusy s Retusa GUI byly ovšem natolik slibné, že bylo postupně naprogramováno komplexní prostředí, které však nebylo předem validováno uživateli (tedy nebyl vůbec ověřen samotný koncept). Právě a pouze z tohoto důvodu – tedy vzhledem k velmi dynamickému postupu - byla prováděna až validace hotového prototypu (demo aplikace).

3.1.1 Metodika

Testování mělo povahu kvalitativního výzkumu, konkrétně **individuálních hloubkových rozhovorů**, které byly prováděny s potenciálními uživateli aplikace Retusa GUI. Rozhovory byly uskutečněny postupně s pěti respondenty, přičemž se jednalo o tři vysokoškolské pedagogy statistiky, jednu analytičku se specializací v marketingovém výzkumu a jednu studentku MÚVS, která neměla se statistickými aplikacemi, vyjma Excelu, žádnou předchozí zkušenost. Velikost vzorku, která se u kvalitativního výzkumu stanovuje odlišnou logikou než u kvantitativního šetření, byla odvozena od běžné praxe v Google Ventures (37), kde autoři vycházeli z

[38] Týmová spolupráce je u obou přístupů klíčová, nicméně u metodiky Sprint je více využíváno individuální kreativní činnosti členů a obvykle jsou vynechávány metody jako brainstorming.

předpokladu, že pět testerů je orientačně dostačující počet pro validaci konceptu produktu[39]. V případě testování Retusa GUI však nebylo předmětem rozhovorů pouze schválení aplikace coby funkční/nefunkční, ale testování bylo zároveň rozšířeno o další témata (viz níže), která mohou pomoci produkt v budoucnosti lépe zacílit. Z pohledu zakotvené teorie (38) je předem stanovený počet respondentů zavádějící, neboť nijak nezaručuje zdárnou formulaci teorie, která není mezerovitá či zavádějící. Důsledkem toho je, že zde provedené testování je třeba chápat jako formu malého výzkumu, typického pro metodiku sprint, ze kterého nad rámec svého účelu vzešlo několik dalších poznatků, které by bylo zajímavé dále rozpracovat.

3.1.2 Výzkumná témata a scénář

Testování se zaměřilo, kromě validace aplikace, i na pochopení kontextu, ze kterého testeři vycházeli. Počáteční témata zůstala v průběhu dotazování stejná, ale měnila se (v souladu se zakotvenou teorií) jejich důležitost a výtěžnost. Zatímco na začátku rozhovorů bylo prioritním zájmem zjistit názor na aplikaci, poslední rozhovory se zaměřovaly více na kontext – např. na problémy, které obecně uživatelé při statistické analýz řeší apod.

Samotná výzkumná témata se týkala pochopení vztahu respondenta ke statistice, zmapování používaných řešení a jejich silných a slabých stránek, dále pochopení žádoucích vlastností statistických aplikací, a konečně samotné testování produktu. Témata a jejich dílčí otázky popisuje níže uvedený scénář.

3.1.2.1 Scénář rozhovorů

- ***TESTER: Řekněte mi něco o sobě***
  - *Co děláte, co vás baví?*
  - *Baví vás statistika? Proč se jí věnujete? Proč vás nebaví?*
- ***POUŽÍVANÉ APLIKACE: Jaký statistický software používáte?***
  - *Název*
  - *Proč tento software?*
  - *Co je na něm dobrého a co špatného?*
  - *Je nějaká aplikace, kterou jste v minulosti používal/a a už jí nepoužíváte? Proč?*
- ***MVP: Co má splňovat dobrý statistický software?***
  - *Jedna hlavní věc*
  - *Pak další dvě hlavní věci*
  - *Proč jsou tyto věci důležité?*
- ***TRŽNÍ PŘÍLEŽITOST: Co vám na aktuálních aplikacích chybí?***
  - *Co dělá špatný statistický software špatných?*
  - *Co vám chybí?*
- ***TESTOVÁNÍ RETUSA GUI***
  - *Jaký byl váš první dojem?*
  - *Co se Vám na aplikaci líbí/nelíbí?*
  - *Co chybí nejvíce?*

[39] Přesněji řečeno, Knap vychází z metaanalýzy Jakoba Nielsena, který dospěl k závěru, že pět rozhovorů odhalí v průměru 85 % existujících problémů a je tak přijatelným kompromisem mezi přesností a náklady. Zdroj: Nielsen, Jakob, a Thomas K. Landauer, „A Mathematical Model of the Finding of Usability Problems" (Matematický model výsledků u problémů s použitelností), sborník z konference ACM INTERCHI'93 (Amsterdam, 24.–29. dubna 1993), str. 206–213; dle Knap a další, 2012.

- *Kde se lze inspirovat a v čem?*
- *Byl byste za aplikaci ochoten/ochotná něco zaplatit? Proč ano/ne?*
- *Kdo by mohl být uživatelem? Pro koho je to vhodné?*

3.1.3 Výsledky

Testující měli možnost si aplikaci Retusa před rozhovorem vyzkoušet (což všichni udělali), a to bez jakýchkoliv instrukcí a nápovědy. Při dotazování byl dán maximální důraz na to, aby otázky nebyly návodné.

Aplikace byla hodnocena všemi respondenty kladně, byť obvykle s dodatkem, že zohlednili, že se jedná teprve o demo verzi, nikoliv definitivní řešení. Mezi argumenty, proč se aplikace líbí, byly jmenovány zejména **snadná přístupnost** (bez nutnosti instalace), **rychlost**, **design** (přehledná struktura, grafy ad.), **uživatelská přívětivost** (nápovědy, rozhodovací strom analýzy) a vybrané funkcionality a prvky. Výtky vůči aplikaci byly velmi úzce zaměřené a netýkaly se systému jako takového. Příkladem, jedna z testujících nenašla bez nápovědy kontextové menu vektoru a očekávala rozsáhlejší paletu funkcí datové tabulky. Jinému testujícímu se nepodařilo aplikaci otevřít v žádném prohlížeči. Další testující vyjádřil přání mít k protokolu výsledků i slovní interpretace. Dílem byly některé úpravy provedeny bezodkladně (např. doplněk *humanizer*), některé další návrhy (rozšíření funkcionalit tabulky ve stylu Excelu) jsou komplexnějším požadavkem a byly přidány do back-logu dalšího vývoje. Kromě výtek testující jmenovali i **vlastní návrhy na další vývoj**, které se často odvíjely od jejich vlastní praxe (možnost propojení s dalšími aplikacemi, automatizace, pokročilejší statistické metody apod.) a byly rovněž zařazeny do back-logu.

Vedle hodnocení aplikace Retusa byl věnován prostor i hodnocení softwarů, které testující využívají. Obecně platí, že **čím vyšších kompetencí uživatelé ve statistické analýze dosahují, tím více aplikací ovládají** (přesněji musí umět ovládat) a jejich užití volí s ohledem na účel (např. Excel pro výuku, EViews pro časové řady apod.). Nutnost mít k dispozici více nástrojů vnímají jako danou skutečnost, byť nepohodlnou a také finančně náročnou. Ač respondenti pocházeli z rozličných prostředí, shodují se v tom, že **vlastností kvalitního softwaru pro statistickou analýzu jsou přehlednost, rychlost, design a zohlednění toho, že uživatelem je člověk** (tzn. potřeba interpretace, nápovědy, tutoriálů, podpora ad.); explicitně nevyřčeným, ale všudypřítomných faktorem byla i cena. Testující se zároveň vyjadřovali i k tomu, co jim na stávajících řešeních nevyhovuje a vadí. Seznam těchto parametrů byl i na tak malém vzorku poměrně rozsáhlý, za **hlavní negativa současných řešení lze označit vysokou cenu**[40] **a malou uživatelskou přívětivost**, což jsou nedostatky systémové; kromě těch respondenti individuálně zmiňovali další vady, týkajících se vesměs detailů. Znalost žádoucích vlastností aplikací i dnes běžné nedostatky mohou být konkurenční výhodou pro další vývoj Retusy.

Tématem rozhovorů byla i vhodná **cílová skupina**. Vývoj Retusy neprobíhal s ohledem na konkrétní využití, byť se předpokládalo, že vzhledem k jednoduchosti aplikace bude moci být uplatňována jen pro prostší účely. Právě proto bylo důležité definovat, na jaký segment by se mělo řešení v budoucnu zaměřovat. Tři respondenti z řad vysokoškolských pedagogů se jednomyslně shodli, že aplikace může být vhodným řešením pro **studenty statistiky**, mj. s ohledem na svou přehledností a dostupností. Analytička z mediální agentury si dokázala představit, že by aplikaci mohla – po doplnění požadovaných funkcí – používat jako regulérní

[40] U placených služeb.

náhradu za SPSS, ze kterého používá jen zlomek funkcionalit. Úvahou jednoho z pedagogů byla využitelnost Retusy i ve firmách (např. bankách), kde analytici čas od času potřebují provést úkony, na které běžné funkce Excelu nestačí a zároveň nemá smysl kvůli tomu nakupovat speciální (a nákladné) řešení. Na základě těchto doporučení byly pro formulaci strategie vybrány coby potenciálně vhodná cílová skupina (konkrétně tzv. první vlaštovky) **české vysoké školy**[41].

3.2 Strategie

Výsledky uživatelského testování naznačují, že Retusa si může získat své uživatele a co pro to je (ze zákaznického pohledu) třeba udělat. Tato zjištění je možné chápat i jako impuls pro další rozvoj aplikací Retusa coby kompaktního produktu. Doposud ovšem nebylo prokázáno, že bude takové úsilí adekvátně zhodnoceno, tedy že koncept „SPSS v prohlížeči" může být nějakým způsobem rentabilní. Následující text odpověď na tuto otázku nepřináší – na to je příliš složitá a závažná, popisuje ovšem způsob, jakým způsobem lze k jejímu zodpovězení přistoupit a jak lze, za určitých okolností, pokračovat dále, směrem k vlastnímu podnikání. Z toho důvodu je strategické uchopení Retusy rozděleno na tři po sobě jdoucí fáze, z nichž první (validační) určí základní směřování produktu, druhá (rozvojová) mu zajistí klienty a třetí (konsolidační) z něj udělá ziskový business – pokud budou splněny všechny důležité podmínky popsané dále.

3.2.1 Validační fáze

Cílem této fáze je ověření komerčního potenciálu řešení. Tato aktivita byla naznačena během uživatelského testování, posláním validační fáze je proto provést důkladnou analýzu vnějších a vnitřních podmínek, konkrétně:

1) **podrobnou analýza vnějšího a vnitřního prostředí** (doplnění stávajících poznatků)
    a. PESTLE a SWOT analýza
    b. Syntéza konkurenčních analýz
    c. Analýza rizik
2) **kvalifikaci a kvantifikaci jednotlivých trhů**
    a. Segmentace trhů a targeting s ohledem na fázi a další faktory
    b. Stanovení velikosti trhů
    c. Kvalitativní vyhodnocení trhů, včetně analýzy rizik
    d. Odhad kupní síly trhů
3) **navrhnout obchodní modely a validovat je na straně zákazníků**

[41] Uživateli sice budou dle očekávání studenti, ovšem potenciálním (platícím či smluvním) zákazníkem bude sama škola.

a. Definice vhodných obchodních modelů, které zohledňují možná specifika nejdůležitějších trhů

b. Ověření těchto modelů u potenciálních zákazníků

4) definovat cíle rozvojové, případně konsolidační fáze

a. Definování toho, co jednotlivé trhy potřebují, pomocí empirického šetření (metody, funkcionality, vlastnosti, dodatečné služby apod.)

b. Formulace back-logů a jejich hierarchie (co bude provedeno přednostně apod.)

5) spočítat náklady rozvojové fáze

a. Stanovení nákladů na doručení výstupů jednotlivých back-logů (časové a mzdové náklady, režie, subdodávky, daně atd.)

6) formulovat misi a vizi produktu/podniku

Jedná se de facto o předprojektovou fázi, která určí, zdali má vůbec ekonomický smysl v rozvoji aplikace pokračovat (validace trhů a obchodního modelu), co pro to je třeba udělat (scope) a s jakými přibližnými náklady je třeba počítat. Z jiného úhlu pohledu se jedná o stanovení projektového trojimperativu, kde faktory nákladů, času a zadání (scope) určují kvalitu výstupu. Výsledky této analýzy budou tvořit, spolu s dříve popsanými zjištění, zejména těmi týkajícími se konkurence, studií proveditelnosti, jejíž technická část již byla poměrně detailně zpracována a ověřena v předchozí kapitole.

Jelikož sama tato fáze je projektem, je třeba se alespoň obecně zaobírat i jejím vlastním trojimperativem. Předmět plnění projektu lze odvodit z aktivit popsaných výše, výstupem je tedy **detailní rozhodovací analýza pro další vývoj služby**. Z hlediska struktury nákladů lze uvažovat v rozhodující většině o **personálních nákladech** potřebných k provedení jednotlivých analytických úkolů (sekundární výzkum, rozhovory, návrh obchodních modelů, finanční analýza atd.), v jednotlivých případech lze uvažovat o subdodávkách (např. právní poradenství). Pokud zohledníme pouze personální náklady, může se jednat odhadem o **20-30** MD (man-day), přičemž většinu aktivity může provést jedna fyzická osoba.

Z hlediska financování této fáze je třeba uvažovat v širším kontextu. Vzhledem k tomu, že se jedná v podstatě o studii proveditelnosti, nelze předpokládat, že bude mít nějaký přímý ekonomický užitek, vyjma toho, že může zamezit tvorbě ztrát, protože odradí od dalších investic do neperspektivního produktu. Protože můžeme o Retuse uvažovat jako o startupu, je běžné, že podobné **náklady na sebe většinou berou sami zakladatelé** (v tomto případě autor) a tyto činnosti posléze figurují jako nefinanční investice při stanovování podílů nově vznikajícího podniku (39), případně i jako proměnné použitelné při jeho valuaci (40).

3.2.2 Rozvojová fáze

Jak bylo naznačeno výše, teoreticky se může stát, že celý projekt Retusa bude ukončen na základě studie proveditelnosti, která ukáže, že má větší smysl věnovat své úsilí či prostředky něčemu jinému. Předpokládejme zde proto, že ekonomická hodnota budoucího produktu (businessu) bude kladná, že projekt bude obecně proveditelný. V tom případě bude dalším krokem tzv. rozvojová fáze. Jejími primárními cíli budou:

1) Dodání výstupu definovaného v předchozí fázi (scope, resp. back-logy)
2) Průběžné validace výstupů, zjišťování dalších uživatelských potřeb a problémů a jejich řešení, implemtace sprintů
3) Akvizice klientů definovaných v předchozí fázi bez důrazu na ziskovost
4) Nastavení KPI (indikátorů plnění cílů)

Výše je zdůrazněno, že v této fázi není cílem na produktu vydělávat, což zaslouží zdůvodnění. U začínajících podniků (startupů) se lze často dočíst, že je třeba se snažit co nejdříve začít vydělávat z prodeje produktu peníze (41). Toto doporučení má nejen logiku, ale funguje i jako bezpečnostní mechanismus: produkt, který není schopen se z dlouhodobého hlediska sám uživit, bude příčinou ztrát. V našem případě bude rentabilita alespoň teoreticky vysvětlena ve validační fázi, nicméně, reálné chování zákazníků bývá často diametrálně odlišné, než jak jej – při nejčistším svědomí – deklarují, čili že jejich ochota v budoucnu platit za dosud bezplatný produkt významně ochabne. Zároveň, základní podmínkou investorů, včetně jinak benevolentních business andělů, bývá alespoň část platících klientů, potažmo jinak ošetřené příjmy (u neziskových organizace např. dotace či granty). To znamená, že bez generování nějakého systematického příjmu nebude Retusa atraktivní investicí; zároveň platí, že bez investic se rozvojová fáze bude dát jen stěží realizovat. Na druhou stranu, velký počet (i) neplatících zákazníků v sobě obyčejně nese sice rizikový, ale vysoký potenciál pro nějakou formu monetizace produktu (i kdyby jen zobrazováním reklam). Mezi těmito faktory je třeba hledat řešení.

Cílem rozvojové fáze je **maximalizovat počet uživatelů**, potažmo utilizaci aplikace, s tím, že na **pilotní části zákazníků budou testovány zpoplatněné (příjmové) části služby** – může se jednat např. o přednostní přístup k nový funkcionalitám či intenzivnější podpora. Tímto způsobem by mohl být vytvořen, z pohledu investorů, produkt, který má jak vysoký potenciál (mnoho uživatelů), tak je u něj potvrzen potenciál komerční (někteří z nich již platí).

Pro interní vyhodnocení úspěšnosti strategie mohou být použity tyto tři metriky:

- celkový počet uživatelů
- podíl platících uživatelů
- celková utilizace (monitoring využívání funkcí a využitého času)

Jelikož budou konkrétní hodnoty trojimperativu rozvojové fáze definovány až ve validační fází, nelze v tuto chvíli přesněji určit ani jeden z parametrů. Předpokládejme, na základě srovnání s podobnými produkty, že trvání této fáze se může pohybovat mezi **jedním a třemi roky**, přirozeně záleží na zdrojích a podmínkách. Právě zdroje, na rozdíl od fáze předchozí, budou v tomto období rozhodující, neboť zásadním způsobem ovlivní čas i kvalitu produktu. V této fázi se investice již nemohou opírat pouze o dříve popsaný vklad zakladatelů, i kdyby jich bylo více (což patrně bude potřeba). Předpokládejme, že vzhledem k době trvání a díky odhadu z dosavadní náročnosti práce se budou pohybovat v řádech milionů Kč ve skupině personálních nákladů (ty budou nadále dominantní). Rostoucí podíl budou mít subdodávky, např. hosting cloudových služeb, marketing, služby typu účetnictví, poradenství apod. Finanční prostředky pro výše popsanou činnost mohou pocházet z několika zdrojů:

- Granty a dotace (např. program TA ČR GAMA 2 určený pro komercializaci)
- Neinstitucionální investice (vklady zakladatelů, FFF)
- Business andělé[42] (v závěru fáze, pokud bude úspěšná)

Je třeba pamatovat, že investice od úrovně business andělů výše jsou téměř vždy podmíněny možností finanční valuace podniku, jejichž výše a výše získaného podílu se odvíjí od budoucího výnosu. V úvodu této fáze nebude standardní výpočet kapitalizace podniku možný, jelikož kromě samotného podniku (i kdyby byl založen) nebudou k dispozici data k výpočtu jeho hodnoty. Klasický výpočet pomocí běžných finančních ukazatelů (např. 10 x EBIT) sice lze částečně nahradit některou pre-money valuační metodou (40), nicméně pro malé a středně velké investory typu business andělů a venture kapitálových společností se bude zdát investice pravděpodobně i tak příliš riziková. Z tohoto důvodu tedy předpokládejme, že samotnou rozvojovou fázi bude třeba z hlediska financování rozdělit do dílčích fází, které budou mít proměnlivé cíle, směřující k lepší dosažitelnosti finančních prostředků.

3.2.2.1 Podnik a tým

Předpokládejme, že nejpozději v průběhu této fáze bude, i s ohledem na financování, vhodné **přidružit produkt k právnické osobě** (zejména, pokud bude vyvíjen týmem). Možných scénářů je zde několik. Jedna varianta je vznik nové právnické osoby (např. s.r.o.), jejímiž podílníky budou zakládající členové. Další je vznik spin-off firmy ČVUT, který je jednak poměrně zevrubně otestován na desítkách stále aktivních subjektů a pro tuto fázi se jeví jako jedno z nejvhodnějších řešení. Produkt se může stát i součástí portfolia existující instituce, ať obchodní, tak neziskové (ta může být rovněž založena). Klíčové je v tomto zohlednit jednotlivé přínosy a rizika.

Bylo uvedeno, že tato fáze již pravděpodobně nebude realizovatelná pouze za účasti jediné osoby, neboť by to přinejmenším zpomalovalo vývoj a jeho rychlost je v tomto období klíčová. Možnou **strukturu týmu** ukazuje tabulka níže; zatímco jednotlivé role a kompetence vycházejí z běžných pozic ve srovnatelných projektech, velikosti úvazků jsou, vzhledem k nejasnosti budoucího plnění, pouze orientační. Zároveň je možné, že pozici developerů zaujme jediná osoba (tzv. full-stack developer), kde se velikosti úvazků sčítají.

[42] Role business andělů není pouze finanční, jejich investice mívají i podobu strategické podpory (propojování s klienty a partnery, poradenství apod.).

TABULKA 10: NÁVRH STRUKTURY TÝMU V ROZVOJOVÉ FÁZI

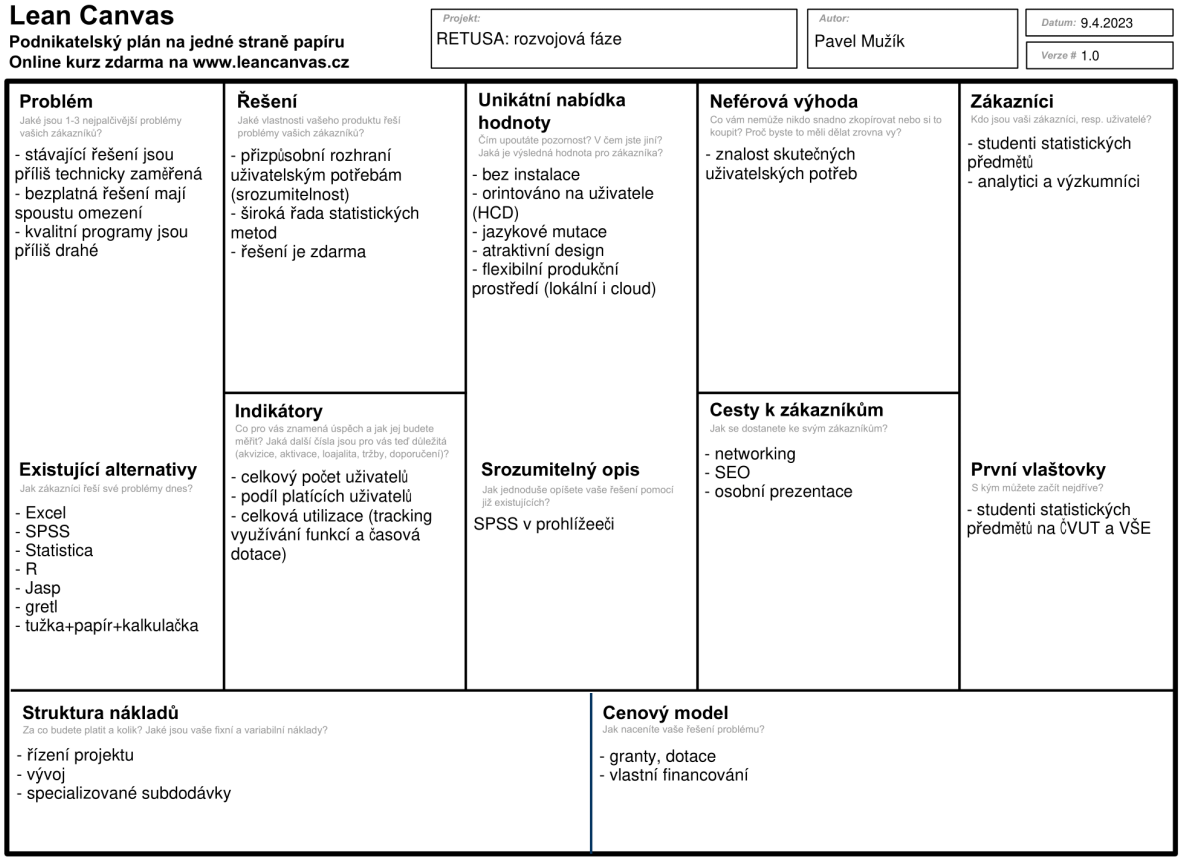
Role	Kompetence	Velikost úvazku
Product owner	• Řízení projektu • Obchod • Rozvoj produktu	0,75
Back-end developer	• Vývoj serverové aplikace (RTS)	0,5
Front-end developer	• Vývoj uživatelského rozhraní (RUI)	0,5
Specialista na statistiku	• Poradenství v oblasti statistické analýzy • Zpracování metodiky	0,1
Designer	• Uživatelské testování • rozvoj produktu	0,25
Back-office	• administrativa	0,1

3.2.2.2 Lean Canvas

Rozvojová fáze představuje překlenovací období mezi stavem ověřeného prototypu a veřejně využívaným produktem s částí platících klientů, integrovaným do právnické osoby v podobě podniku. Je třeba počítat s rizikem, že ani na konci této fáze nebude vlastní činnost v užším smyslu zisková; kladné peněžní toky bude třeba zajistit zejména dotacemi a investicemi. V každém případě by měla ale být rozvojová fáze zakončena dostatečně prokázaným potenciálem produktu stát se ziskovým a ideálně soběstačným ve třetí, konsolidační fázi.

Vlastní strategie rozvojové fáze je shrnuta pomocí metody **Lean Canvas** níže. Jelikož většina jejích oblastí byla popsána dříve, zaměřme se jen na vybrané z nich. Na základě dosavadního testování se jako segment tzv. *Early adopters* (první vlaštovky) jeví studenti statistiky na českých univerzitách (přesněji univerzity samy), nicméně platnost tohoto předpokladu prokáže až validační fáze, stejně tak jako vymezení ostatních uživatelů. Z hlediska marketingu lze při minimálním scénáři (např. pouze české vysoké školy) předpokládat, že k vlastní akvizici klientů bude dostačující **přímý prodej** skrze networking. Zvláštní oblastí nástroje Lean Canvas je tzv. neférová výhoda, která obvykle nemusí být pro úspěch businessu rozhodující, např. post-hoc zpracovaný příklad Facebooku kolem ní tápe (42); v kontextu dříve zjištěného může být touto výhodou důraz na **lidsky orientovaný přístup** (human-centered design).

OBRÁZEK 26: LEAN CANVAS (ROZVOJOVÁ FÁZE)



3.2.3 Konsolidační fáze

Hlavním cílem konsolidační fáze je přeměna produktu (resp. mateřského podniku) v **profitabilní a udržitelný business**. Vzhledem k tomu, že tato fáze bude definována dvěma předchozími a doposud teoretickými fázemi, je možno o ni uvažovat pouze velice obecně. Management by v této fázi měl soustředit svou pozornost zejména na **intenzivní akvizici nových klientů a maximalizovat podíl platících klientů** vůči celku (prozatím předpokládejme, že valná část funkcí bude bezplatná). Jednou z důležitých vlastností aplikace je vícejazyčnost, které umožňuje snadnější vstup na zahraniční trhy, kdy zároveň platí, že statistická analýza je daleko méně citlivá na kulturní specifika než další segmenty. Příjmy by se měly tvořit zejména na základě vlastní činnosti (prodej služeb a zboží), což pochopitelně platí zejména pro obchodní společnosti – Retusa se stejně tak dobře může stát páteřní službou neziskové organizace a kvalitně plnit své poslání i tak.

Růst zákaznické báze bude mít pravděpodobně za následek rostoucí nároky na vývojový tým, obchod, subdodávky atd. a tím pádem **rostoucí náklady**. Vzhledem k tomu, že se pohybujeme ve vyšších časových jednotkách (letech), je nutné přihlédnout také k celkové změně vnějšího prostředí – nejen ve smyslu očekávatelného rozvoje služeb na straně **konkurence**, ale také rozvoj **alternativních řešení** (např. dnes často skloňované umělé inteligence, která se může stát příslovečným game-changerem i ve výzkumu obecně). Slovo konsolidační zde tedy neznamená pouze konsolidaci financí podniku, ale obecně ustálení tržní pozice produktu samotného.

3.3 Strategické nástroje

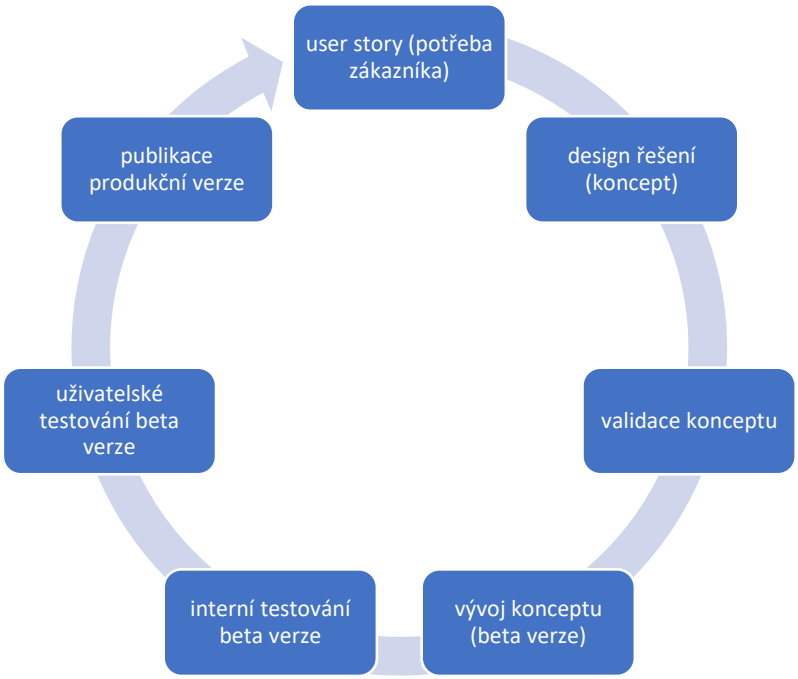
Níže popsané sekce se zaměřují na doplnění výše uvedené strategie o další poznatky a návrhy, které se v ní mohou uplatnit. Vycházejí částečně ze zkušeností, které byly shromážděny během vývoje aplikace a které nebyly zohledněny v předchozích částech této práce.

3.3.1 Optimalizace vývoje

Klíčovým faktorem pro úspěch strategie je čas. Dostatečně pohotový vývoj představuje nejen možnou konkurenční výhodu, ale zároveň urychluje dosažení bodu, kdy bude možno plošněji zavádět placené funkcionality či modely; krom toho se může jednat i o úsporu nákladů. Jednou z možností, jak efektivněji řídit budoucí vývoj aplikace (resp. komplexního produktu), který byl doposud řízen spíše intuitivně, je **implementace metodiky sprint**. V agilním řízení má termín sprint jiný význam, než ve dříve uvedeném případě: jedná se o iterativní cyklus procesů, kterým se postupně produkt vyvíjí, inovuje, modifikuje apod. V metodice Scrum (31) a dalších agilních přístupech se sprinty obvykle objevují v kontextu tzv. *User stories*, což jsou jednoduché popisy toho, co uživatel potřebuje, např. od softwaru. Tyto požadavky vstupují do tzv. back-logů (projektového a následně sprintového), což jsou de facto prioritizované soupisy úkolů pro produkční tým. Tyto úkoly jsou následně vypořádávány v cyklech (sprintech), které jsou povětšinou předem časově ohraničené a probíhají v několika krocích.

Pro budoucí produkci uvažujme synchronizovaný vývoj obou aplikací (RTS a RUI). V první fázi budou zjišťovány tzv. **user stories** (uživatelské potřeby); toto zjišťování může nabývat celé řady podob (ad hoc šetření, zjevné nedostatky v dosavadní verzi, výsledky z předchozích testování), důležité je, že nějakým způsobem reflektují to, co uživatel skutečně potřebuje, co mu nevyhovuje, co dává smysl apod. Potřeby budou následně prioritizovány dle své významnosti, realizovatelnosti a přidané hodnoty a formulovány do jednotlivých **back-logů**. Jelikož mohou být back-logy triviální a komplexní, bude i doba pro jejich splnění proměnlivá. Příkladem, vývoj a integrace maticové metody v RTS/RUI, včetně dokumentace, testování a naprogramování doplňků, trvala obvykle mezi 4 až 8 hodinami. Naopak integrace systematických změn (např. přesun části služeb do cloudu) bude otázkou desítek hodin, robustní analýzy celého systému a komplexního testování. Samotné back-logy budou realizovány pomocí **sprintů**, jejichž proces je zobrazen níže. Logika sady aktivit je velmi podobná té u metodiky Scrum, s tím, že v našem případě je fixně počítáno i s jinými možnostmi vývoje než lineárními – příkladem, po validaci konceptu není nutné přistupovat k vývoji beta verze, nýbrž je dle potřeby možno upravit samotný koncept, dokud nebude dokonalý; tento rekurzivní přístup je znám i u designového myšlení. Každý úspěšně provedený sprint je ukončen **publikací** (release) nové funkcionality; v případě RUI byl tento již postup zaveden, jednotlivé back-logy jsou součástí dokumentace na GitHub.

OBRÁZEK 27: NÁVRH VÝVOJOVÉHO SPRINTU



3.3.2 Ochrana duševního vlastnictví

Dalším faktorem, který je nutno zohlednit, je rámec, ve kterém se Retuse jako softwaru dostává právní ochrany. V České republice jsou zákony na ochranu duševního vlastnictví softwaru upraveny především v zákoně č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským (autorský zákon) a zákoně č. 89/2012 Sb., občanském zákoníku. Podle autorského zákona je počítačový program, tedy jakýkoliv software, bez ohledu na formu jeho vyjádření, včetně přípravných koncepčních materiálů, chráněn jako dílo literární, pokud splňuje základní podmínky pro autorskou ochranu. Software tak musí být jedinečným výsledkem tvůrčí činnosti autora a musí být vyjádřen v jakékoli objektivně vnímatelné podobě včetně podoby elektronické, trvale nebo dočasně, bez ohledu na jeho rozsah, účel nebo význam. To znamená, že softwarové dílo musí být originální a musí být vyjádřeno v nějaké konkrétní formě, obvykle v kódu. Autorský zákon chrání autora softwaru před neoprávněným užitím softwaru, tak, že umožňuje autorovi, aby se domáhal

a) určení autorství;

b) zákazu ohrožení svého práva, včetně hrozícího opakování nebo neoprávněného zásahu do svého práva, zejména zákazu neoprávněné výroby, neoprávněného obchodního odbytu, neoprávněného dovozu nebo vývozu originálu nebo rozmnoženiny či napodobeniny díla, neoprávněného sdělování díla veřejnosti, jakož i neoprávněné propagace, včetně inzerce a jiné reklamy;

c) sdělení údajů o původu neoprávněně zhotovené rozmnoženiny či napodobeniny díla, o způsobu a rozsahu jejího užití a o totožnosti osob, které se neoprávněného zhotovení, popř. neoprávněného rozšiřování účastní;

d) odstranění následků zásahu do práva, zejména:

    a. stažením neoprávněně zhotovené rozmnoženiny či napodobeniny díla z obchodování nebo jiného užití;

    b. zničením neoprávněně zhotovené rozmnoženiny či napodobeniny díla

e) poskytnutí přiměřeného zadostiučinění za způsobenou nemajetkovou újmu, zejména:

    a. omluvou;

    b. zadostiučiněním v penězích, pokud by se přiznání jiného zadostiučinění nejevilo postačujícím; výši peněžitého zadostiučinění určí soud, který přihlédne zejména k závažnosti vzniklé újmy a k okolnostem, za nichž k zásahu do práva došlo; tím není vyloučena dohoda o narovnání;

f) práva uveřejnit rozsudek na náklady účastníka, který ve sporu neuspěl;

g) náhrady škody;

h) vydání bezdůvodného obohacení.

Další právní ochranu autorům softwaru umožňují například občanský zákoník, zákon o vynálezech, průmyslových vzorech a zlepšovacích návrzích, zákon o užitných vzorech, zákon o vymáhání práv z průmyslového vlastnictví a ochraně obchodního tajemství a v neposlední řadě i trestní zákoník. Problematika ochrany autorských práv však svým rozsahem přesahuje téma této práce a nebude proto dále podrobněji rozebírána.

Konkrétně u interpretovaných jazyků, mezi které patří JavaScript, je praktická ochrana komplikovaná, jelikož zdrojový kód není kompilovaný a kdokoliv, kdo má k němu přístup (což je například uživatel, který si webovou stránku načte a její skripty zobrazí v konzoli prohlížeče), může kód libovolně měnit, kopírovat a distribuovat. Proto javascriptové knihovny z podstaty vnímáme jako open-source. Využití JavaScriptu je nejtypičtější právě v internetových prohlížečích.

Možné způsoby ochrany před zneužitím ve výše uvedeném významu jsou minimálně tři. První z nich tvoří tzv. **obfuskace**: jedná se o transformaci kódu do podoby, která se lidsky obtížně interpretuje a stěžuje reverzně-inženýrské dekódování vztahů uvnitř kódu. Typickým příkladem je minimalizování JavaScriptu. Většina původních kódů, vytvářených vývojáři, je psána ve struktuře, která zohledňuje hierarchii jednotlivých elementů (funkcí, kódu uvnitř atd.). Vývojáři také opatřují kódy poznámkami (formát JSDoc), mimoto používají deklarace proměnných či konstant pomocí lidsky srozumitelných řetězců (např. funkci *correlPearson* lze nazvat prostě *t* nebo _, časem ale vývojář zapomene, co toto *t* konkrétně znamená). Všechny uvedené vlastnosti jsou pro samotnou interpretaci jazyka počítačem nerelevantní a javascriptová obfuskace je založená právě na tom, že nadbytečné části kódu minimalizuje (correlPearson => t), ignoruje (komentáře), redukuje duplicity kódu atd. Javascriptová obfuskace je primárně způsobem, jak možné útoky zkomplikovat či zpomalit, ale její skutečná ochrana je sporná. I proto je dnes většina javascriptových knihoven dostupná i v původní, nekomprimované verzi. Mimochodem, minimalizované verze skriptů, dostupné běžně u webových stránek, jsou historicky vytvářeny nikoliv z hlediska ochrany kódu, ale především kvůli nižší přenosové zátěži.

Druhým způsobem je **ochrana pomocí licenční smlouvy**. Jedná se o obecný přístup k ochraně a standardizovaných smluvních vzorů je celá řada. Retusa je aktuálně chráněna smlouvou MIT Licence (43), která je mimořádně benevolentní vůči manipulaci s kódem, jediným požadavkem je de facto soustavné a pravdivé uvádění autora kódu a typu licence. Další smluvní vzory pro ochranu open-source kódů (44) mají z hlediska možné právní ochrany srovnatelnou účinnost. Klíčová je zde skutečnost, že open-source řešení – jak naznačuje název – je ze své podstaty řešení otevřené veřejnosti. Krom toho je ještě nutné zohlednit související skutečnost, že aplikace vznikla jako součást závěrečné vysokoškolské práce a k jejímu užití se vztahuje smluvní vztah mezi autorem a dotčenou vysokou školou (45).

Třetí způsob ochrany je povahově zcela odlišný od dvou předchozích. Spočívá v celkové změně architektury řešení, kdy jsou části kódů neveřejné a jejich **zpracování probíhá ve veřejně nepřístupném prostředí** (v serveru). Nejedná se přitom pouze o způsob ochrany, může jít i o obchodní model[43]. Ze všech tří uvedených způsobů ochrany má tento jako jediný skutečný potenciál ochrany (pakliže ji neprolomí např. zaměstnanec nebo hacker), neboť zdrojový kód chrání lépe než kompilace; navíc za určitých podmínek umožňuje i monetizaci částí služby, neboť prakticky přebírá náklady osobního počítače uživatele (ve smyslu spotřebované energie, výpočetní kapacity apod.) na sebe a zároveň zabraňuje tomu, aby byly provozovány ve prospěch jiného provozovatele.

3.3.3 Obchodní modely

Jak bylo uvedeno výše, zpoplatnění stávající formy aplikace je z řady důvodů komplikované – aplikace má podobu open-source, je tedy snadno kopírovatelná a právně de facto nechráněná. Na otázku, zdali by byl zákazník za takovou službu ochoten platit, je možno si odpovědět opět otázkou: proč by měl? Bezplatných řešení na trhu je více (Jasp, Gretl ad.) a lze předpokládat, že uživatel bude ochotný platit až za něco, **co zdarma jinde nezíská**, případně za něco, co vůbec nikde (ve srovnatelné kvalitě) nezíská a co má pro něj klíčovou hodnotu nebo do třetice za něco, co svou kvalitou převyšuje stávající nabídku. Jedním z cílů vývoje a zejména designování aplikace bude takové **funkce a vlastnosti identifikovat a co nejlépe je naplnit**. Tento postup řeší implementace sprintů popsaná výše.

Design produktu sám o sobě ještě není obchodním modelem, ale je jeho důležitou součástí. Dalším parametrem je to, za co a kolik budou zákazníci ochotni platit, a to i v kontextu jejich kupní síly (ta je na úrovni institucí obyčejně násobně vyšší než u individuálních uživatelů). **U existujících funkcionalit bude důležité zohlednit cenu u konkurence, u neexistujících (inovativních) hodnotu pro zákazníka**.

Z hlediska zákazníka často kosmetická, ale z pohledu podniku klíčová věc je **periodicita plateb**. U stále většího množství online (i desktopových) aplikací se setkáváme s modelem **předplatného** (subscription), které je založeno na pravidelných platbách (např. měsíčních), které pro podnik představují dobrou predikovatelnost cashflow a tím obecně lepší možnosti plánování dalších nákladů, včetně přímých investic do rozvoje produktu; zákazníkovi tento model většinou přislibuje určitou míru podpory a dalších benefitů (např. slevy). Obecně se

[43] Velké cloudové služby, jako je AWS, Azure či Google Cloud, umožňují monitorovat celkovou spotřebu svých služeb až na úroveň jednotlivých volání. Tímto způsobem tak lze měřit potřebu služeb jednotlivých zákazníků optikou výpočetní doby, spotřeby paměti virtuálních počítačů a využití navázaných cloudových služeb. U nás tento model s úspěchem využívá např. společnost Apify **Je zadán neplatný pramen.**.

tedy jedná o recipročně výhodnější smluvní vztah než jednorázové platby, byť větší přidanou hodnotu má spíše pro prodejce.

Aktuální limit výpočetní kapacity Retusy je roven výpočetní kapacitě uživatelova zařízení (počítače). Jedním z možných využití tohoto omezení je, že aplikace přesune část služeb a funkcionalit do cloudu, kde mj. nebude uživatel limitován výše uvedeným způsobem. Příkladem, v bezplatné verzi uživatel spočítá např. korelační koeficienty, ale pro výpočet analýzy hlavních komponent nebo pro výpočet korelací s desítkami milionů případů bude využit server. Aby mohl uživatel těchto funkcionalit využívat, musí si **předplatit jejich konkrétní objem**. Z pohledu výrobce jsou nákladem za tento objem poplatky poskytovateli cloudové služby, vyjádřené násobkem spotřebované paměti a doby trvání využití služby (př. služba AWS Serverless Lambda); na straně uživatele musí být tedy nastavena unikátní **konvertibilní měna** (např. Retusa bod), která bude obsahovat kromě určité výpočetní kapacity i další benefity, škálované dle výše předplatného (např. cena jednoho bodu či kvalita podpory). Jednotlivé úrovně předplatného je třeba nastavit tak, aby a) byly jako celek rentabilní a b) aby si mohl uživatel vybrat vhodnou variantu vzhledem ke svým potřebám a možnostem.

Níže uvedený příklad (nevychází z konkrétní kalkulace nákladů) ukazuje možné moduly předplatného. Většina podobných služeb dnes nabízí nějakou formu bezplatné licence, přičemž se mezi sebou liší rozsahem služeb, případně časovou omezeností. V ukázce je vědomě zvýhodněn klient institucionálního typu, kterému je nabídnuta nejen nejnižší cena za bod na uživatele, ale při ročním předplatném i největší sleva. Jednotlivé balíčky se liší i co do dodatečných služeb (opět jen naznačeny) či úrovně poskytované podpory. Pomocí tohoto modelu lze flexibilněji měnit ceník (buďto změnit cenu, nebo počet Retusa bodů), přičemž z psychologického hlediska není změna pro zákazníka tak patrná.

TABULKA 11: MODULY PŘEDPLATNÉHO (PŘÍKLAD)

LICENCE	FREE	STUDENT	SMALL TEAM	ENTERPRISE	CUSTOM
Počet uživatelů	1	1	10	100	*Kontaktujte nás*
Počet Retusa bodů/uživatel	100	1 000	2 000	5 000	*Kontaktujte nás*
Podpora	Fórum	Email	Chat Email	Chat Email Telefon	*Kontaktujte nás*
Další služby	Základní balíček	Vše z FREE + vícerozměrná analýzy	Vše ze STUDENT + zálohování 30 dní + API	Vše ze SMALL TEAM + migrace z projektů v SPSS	*Kontaktujte nás*
Cena/měsíc	**Zdarma**	**99 Kč**	**890 Kč**	**7 900 Kč**	*Kontaktujte nás*
Cena/rok	**Zdarma**	**89 Kč (10% sleva)**	**799 Kč (10% sleva)**	**6 320 Kč (20 % sleva)**	*Kontaktujte nás*

Součástí obchodního modelu mohou být, nad rámec předplatného, i různé zakázkové ad-hoc služby, jako jsou vzdělávací programy, outsourcing (zpracování analýzy bez předplatného), exkluzivní zákaznická rozšíření, dostupná pouze konkrétnímu klientovi a podobně. Pro rentabilitu jednotlivých zdrojů příjmů je nutné, kromě důkladné teoretické přípravy, zohlednit jejich „chování“ v praxi a dle toho obchodní model upravovat.

Kromě uvedených a dalších modelů komerčních existují i modely neziskové, které jsou nicméně pro služby uvedeného typů méně obvyklé. Jedním z důvodů je **velké rozpětí odhadů**

nákladů na vývoj softwaru, které u větších projektů znásobují z podstaty přítomnou nejistotu (v tomto případě nejistotu hodnoty nákladů a času). Jelikož náklady jsou vitální součástí obchodního modelu, připomeňme, že tvorba kalkulace se obvykle provádí pomocí různých pravděpodobnostních metod, které se snaží **minimalizovat odhad rozpětí nákladů** (46).

Uvedené návrhy je přirozeně nutné validovat v rámci první fáze. Ověření je zároveň nutné dát do souladu s produktem jako celkem, to znamená, že vztah produktu a obchodního modelu je reciproční. Doplňme, že u současné verze aplikace je již nyní možné implementovat obchodní model založený na **zobrazování reklamy**. Na druhou stranu, dlouhodobě klesající výnosy z display inzerce a averze uživatelů vůči ní budou mít pravděpodobně za následek to, že tato varianta bude již v první fázi vyloučena.

3.3.4 Analýza rizik

Důležitou součástí výstupu validační fáze bude analýza možných rizik, jejímž jedním z možných závěrů bude, že pokračování ve vývoji je celkově příliš rizikové. Protože řada možných rizik a jejich případných ošetření je známá již v tuto chvíli (na základě obecných zákonitostí i zkušeností z vývoje), byl pro účely testování připraven seznam vybraných rizik (níže), který bude dle potřeby ve validační fázi doplněn.

TABULKA 12: ANALÝZA RIZIK

Riziko	Popis rizika	Pravděpodobnost	Závažnost	Možnost řešení	Náročnost řešení
Chyby v návrhu aplikace	Chyby v návrhu aplikace mohou vést k nesprávné funkčnosti aplikace	Vysoká	Vysoká	Testování a revize návrhu aplikace	Střední
Závislost na knihovnách třetích stran	Závislost na knihovnách třetích stran může vést k nefunkčnosti aplikace, pokud se tyto knihovny změní nebo zastarají	Střední	Vysoká	Pravidelná aktualizace a testování knihoven	Střední
Problémy s kompatibilitou na různých platformách	Aplikace může být náchylná k problémům s kompatibilitou na různých operačních systémech nebo verzích prohlížečů	Střední	Vysoká	Testování na různých platformách	Střední
Omezené finanční zdroje	Omezené finanční zdroje mohou vést k omezenému rozvoji aplikace	Vysoká	Vysoká	Získání investice nebo financování od univerzit nebo jiných organizací	Vysoká
Ztráta zákazníků kvůli konkurenci	Konkurence může nabídnout podobnou aplikaci nebo produkt, což může vést ke ztrátě zákazníků	Střední	Vysoká	Vývoj nových funkcí a zlepšování stávajících funkcí, které zákazníky přilákají	Střední
Změna požadavků zákazníka	Změna požadavků zákazníka může vést ke zpoždění vývoje nebo nutnosti kompletního přepracování aplikace	Střední	Vysoká	Pravidelná komunikace s zákazníkem a průběžné aktualizace požadavků	Vysoká
Omezené znalosti o marketingu	Nedostatečné znalosti o marketingu mohou vést k nedostatečnému zaujetí zákazníků	Vysoká	Střední	Vyhledání odborníka na marketing nebo	Vysoká

				vzdělávání se v této oblasti	
Chybné výpočty	Chyba v implementaci výpočetních metod	Vysoká	Vysoká	Ladění kódu, testování	Vysoká
Nepoužitelnost na některých zařízeních	Nefunkčnost aplikace na určitých zařízeních	Střední	Vysoká	Testování na různých zařízeních, oprava chyb	Střední
Závislost na třetích stranách	Změny v knihovnách třetích stran mohou způsobit nefunkčnost aplikace	Vysoká	Vysoká	Aktualizace knihoven, testování aplikace	Vysoká
Ztráta dat	Ztráta neuložených dat kvůli pádu aplikace	Nízká	Vysoká	Implementace automatickéh o ukládání dat	Vysoká
Konkurence	Konkurence má lepší marketingovou strategii	Střední	Vysoká	Vylepšení marketingové strategie, zlepšení uživatelské zkušenosti	Střední
Změna legislativy	Změna zákona, který aplikace porušuje	Nízká	Vysoká	Úprava aplikace, aby vyhovovala novému zákonu	Vysoká
Nedostatek financí	Nedostatek financí na vývoj aplikace	Nízká	Vysoká	Hledání investora, úprava rozpočtu	Vysoká
Nedostatek času	Nedostatek času na vývoj aplikace	Střední	Vysoká	Přizpůsobení plánu vývoje, najmutí dalšího vývojáře	Vysoká
Nízká poptávka	Malý zájem o aplikaci ze strany zákazníků	Vysoká	Střední	Vylepšení marketingové strategie, zlepšení uživatelské zkušenosti	Střední
Nespokojenost uživatelů	Nespokojenost uživatelů s funkcionalitou aplikace	Střední	Vysoká	Zlepšení uživatelské zkušenosti, vylepšení funkcionality	Střední
Nesprávná interpretace dat	Nesprávná interpretace výsledků aplikace	Střední	Vysoká	Rozvoj extenzí typu humanizer	Střední

3.4 Shrnutí

Cílem této kapitoly bylo zhodnotit, zdali aplikace Retusa jako komplexní softwarové řešení vykazuje předpoklady pro možnou komercializaci a s ní spojenou potřebu či možnost produkt dále rozvíjet. Reakce testerů i (jakkoliv fragmentovaná) analýza prostředí naznačují, že segment aplikací pro statistickou analýzu nabízí celou řadu oblastí pro inovace, které jsou nejen technicky dosažitelné, ale i uživatelsky žádoucí. Jak bylo zmíněno dříve, původní rámec této práce zamýšlel pouze vývoj jádrové aplikace Retusa, s komprimovanou verzí pro

prohlížeč. Výsledky byly ovšem natolik motivující, že kromě toho, že byla naprogramována druhá aplikace (Retusa GUI), bylo následně provedeno i uživatelské testování, které naznačilo, že další rozvoj obou softwarů může mít smysl a užitek. Z tohoto důvodu slouží popis komercializace zejména jako úvaha nad dalšími potřebnými kroky, z nichž prvním je validace celého konceptu, provedená dílem empiricky, mj. komunikací s potenciálními obchodními partnery. Právě z důvodu (nejen) časové náročnosti zůstávají tyto aktivity pouze ve formě doporučení. Na druhou stranu, součástí validační fáze by byly za běžných okolností další, relativně náročné činnosti, jako je ověření technické proveditelnosti – tou byl však samotný vývoj aplikace, popsaný zevrubně v druhé části této práce. Design výstupu se odvíjel od detailní analýzy existujících řešení i potřeb a problémů uživatele. Tyto a další složky jsou tedy již vypracovanými odpověďmi na důležité otázky, které budou po rozhodnutí o dalším vývoji aplikace pokládány.

Závěr

Cílem této práce bylo navrhnout a realizovat inovativní softwarové řešení pro statistickou analýzu v prostředí internetového prohlížeče a Node.js. Toho bylo dosaženo průpravnou analýzou existujících řešení, nejen na úrovni specifických knihoven v Node.js, ale i běžných statistických softwarů a také vyhodnocením vybraných informací o uživatelích těchto řešení. Těžiště této práce spočívá zejména ve vývoji dvou vzájemně souvisejících aplikací, které představují klíčový výstup celé práce. Popisu obou dvou je věnována střední část této práce, druhá významná část, dokumentace, je – i s ohledem na pokračující vývoj – veřejně dostupná na internetu. Jedná se s v obou případech o plně funkční řešení, výrazně přesahující vlastnosti specifické pro prototyp; zatímco aplikaci Retusa v Node.js lze bez překážek používat v aplikacích třetích stran, aplikaci Retusa GUI si může libovolný uživatel otevřít z veřejně přístupné adresy ve svém prohlížeči (na libovolném zřízení) a provádět vlastní analýzu, a to většinou bez jakékoliv instruktáže – design aplikace se soustředí na intuitivnost ovládání. Výstupy řešením mají podobu komentovaných výsledků, grafů, tabulek a dalších doplňků, které mohou být dále rozšiřovány. Automatizované testování aplikace potvrdilo, že zpracování výsledků je nejen rychlé, ale zejména ve shodě s hlavními statistickými nástroji, které se dnes ve světě používají. Uživatelské testování výstupu vedlo k rozšíření této práce o část věnující se strategickému rozvoji aplikací, s důrazem na jejich budoucí monetizaci, neboť nejen že byla aplikace Retusa GUI plošně pozitivně hodnocena, ale další diskuze indikovala i její možné širší uplatnění v budoucnu.

Shrneme-li výše uvedené skutečnosti, lze tvrdit, že **předem stanované cíle byly splněny**. Je však třeba dodat, že byly nejen splněny, nýbrž předčeny. Samotná inovativnost výstupu této práce nespočívá pouze v tom, že bylo vyvinuto v podstatě ojedinělé a komplexní řešení v JavaScriptu. Druhým podstatným přínosem je i poznatek, že aktuální nabídka statistických softwarů jako systému není ani zdaleka dokonalá a že současná situace na trhu nabízí velký prostor právě pro inovování současných přístupů, zejména vůči specifickým segmentům zákazníků.

Věnujme těmto segmentům krátce pozornost. Jako perspektivní uživatel Retusy se zdá být, na základě nepřímých ukazatelů, studující, resp. kdokoliv, kdo potřebuje zpracovat jednoduchou statistickou analýzu a nechce být ponechán napospas abstraktním ukazatelům, jejichž správná interpretace není jen otázkou vzdělání, ovšem i zkušenosti. Uvážíme-li dnes hojně prosazovaný a zcela oprávněný koncept, že produkt má především řešit skutečné problémy uživatelů, zjistíme, že se na tuto poměrně širokou skupinu uživatelů trochu zapomnělo. Ať již začínající analytik používá Excel nebo specializovaný software, při orientaci v aplikaci a zejména jejích výsledcích se od něj očekává dobrá schopnost interpretace různých testů, což stěží povede k větší popularitě statistiky mezi nestatistiky, je to naopak překážka. Doplňme, že se to netýká jen těch, pro něž je statistika vedlejší obor. Jedna respondentka z řad učitelů odpověděla na dotaz, proč by ocenila nápovědu a interpretaci (kdo by to ostatně měl potřebovat méně), že některé metody používá jen výjimečně a tudíž i ona potřebuje v tomto komplexním oboru občas poradit. Nejedná se zjevně o anomálii, tento fakt nějakým způsobem zmiňovali i ostatní pedagogové - je to stejné, jako by specialista na ORL měl umět pohotově vyndat žlučník. Mimochodem, biomedicinské obory statistickou analýzu hojně využívají, přitom se jedná o podobně vzdálený obor, jako jsou třeba sociologie či psychologie, jehož studenti si rovněž musí analytické dovednosti osvojit. Dle mé osobní zkušenosti vedou obavy ze statistiky u studentů dvou posledně zmíněných oborů často k málo využitému potenciálu

dat, které pro svá šetření sbírají. Stejnou zkušenost mám také z marketingu, kde i na úrovni výzkumných agentur jen zřídkakdy narazíme na komplexnější statistické testování – zde je častou příčinou to, že klienti sami výsledkům statistické analýzy příliš nerozumí a proto je podobná činnost nadbytečná.

Často se hovoří o tom, že v datech (firemních) dlí vysoká hodnota – toto tvrzení je však neúplné. Sama hodnota je totiž podmíněná nejen kvalitními daty (více než jejich množstvím), ale zároveň jejich výtěžnou analýzou, interpretací a zejména využitím. Pokud nebudou majitelé těchto dat schopni z nich vytěžit nosné informace, je hodnota podkladů sporná. Jiným příkladem budiž sáhodlouhý dotazník, ze kterého analytik zpracuje jen grafy a tabulky, aniž by se jakkoliv zabýval vnitřní dynamikou dat[44]. Toto je jedna z běžných příčin nevyužitého potenciálu dat. Bylo by však nespravedlivé přisuzovat chybu uživatelům. Naopak, nehledejme viníka, ale ptejme se stejně jako ve Sprintu[45]: *„jak bychom mohli pomoci uživatelům, kteří mají problémy se statistickou analýzou*?“

Jednou z možných odpovědí je koncept aplikace Retusa GUI. Jak bylo popsáno v části o strategii, budoucí úspěch produktu tkví do značné míry v tom, jak dobře dokážeme řešit skutečné problémy našich zákazníků. Pokud je problémem naší uvažované cílové skupiny to, že statistice zkrátka tolik nerozumí a potřebují v ní vést, pak je užší odpovědí na jejich potřeby aplikace, která je celým procesem nejrůznějších uživatelských případů provede jako mentor. Tento přístup by teoreticky mohl oživit výzkumné zájmy v řadě nestatistických oborů, které mají tendenci se buďto statistické analýze vyhýbat úplně, nebo je mezi jejich představiteli jen málo těch, kteří dokáží pomocí profesionálně provedené analýzy vyzdvihnout výsledky své práce vysoko nad průměr.

Tato práce dokazuje, že při použití nejdostupnějších nástrojů (Node.js a internetový prohlížeč) lze bez vstupních investic – vyjma lidské práce – vytvořit nástroj, který má potenciál pomocí pár stovek řádků potlačit některé běžné potíže uživatelů, popsaných výše. Celý rámec je překvapivě jednoduchý – stačí pozorně naslouchat těmto uživatelům a krok po kroku (doslova řádek po řádku) se přibližovat k tomu, aby pro ně byla statistická analýza nikoliv strašákem, ale naopak vítaným pomocníkem. Z technického hlediska se nejedná o nic neortodoxního ani nedosažitelného, dříve rozebraná dokumentace aplikací ukazuje, jak lze dílčí komponenty i celkový systém řešit.

Pomyslnou hozenou rukavicí předkládané práce je tedy otázka, zdali my, designéři a vývojáři statistických softwarů (mohu-li se mezi ně nyní neskromně řadit), dáváme našich zákazníkům to, co skutečně potřebují, co řeší jejich problémy. Je pravděpodobné, že na trhu existuje příležitost v podobě milionů uživatelů, kteří se statistické analýze vyhýbají, protože na ně doposud – a to i za přispění existujících aplikací – působí jako těžko osvojitelná věda, určená jen pro vybrané. Přitom smysl statistické analýzy je opačný – ne věci komplikovat, ale naopak je co nejvíce zjednodušit. Pokud je tento předpoklad správný, může být misí Retusy ***zpřístupňovat statistickou analýzu komukoliv a kdekoliv.***

[44] Tento problém je, dle mého soudu, dnes častější i kvůli tomu, že je možné přes online dotazníková řešení získat poměrně pohodlně a rychle odpovědi od desítek i stovek respondentů. V dobách, kde se dotazování realizovalo výhradně formou CAPI nebo CATI byl častěji dáván důraz na pečlivější přípravu celého výzkumu a větší vytěžitelnost dat, s ohledem na respondenty, tazatele a konec konců i na náklady.

[45] Nejedná se zde o řečnickou otázku, nýbrž o jeden z klíčových nástrojů metodik Sprint i Design Thinking, známý v angličtině jako „How might we…“.

Seznam použité literatury

1. **Kačerová, Eva.** Příběh statistiky. *Český statistický úřad | ČSÚ.* [Online] 2014. [Citace: 12. dubna 2023.] https://www.czso.cz/documents/10180/20540505/32018414.pdf/5001140e-9277-4551-845e-b1ca5a977284?version=1.2.
2. **Hájková, Petra a Veselý, Jan.** Lean Canvas - Online kurz za dobrý skutek. [Online] 2015. [Citace: 27. března 2023.] https://www.leancanvas.cz/.
3. **Yadolah, Dodge a Hand, David J.** What should future statistical software look like? *Computational Statistics & Data Analysis.* 1991, Sv. 12, 3.
4. **The VSNi Team.** A history of statistical computing. *VSNi | Data Science & Statistics Software for Biosciences.* [Online] 27. dubna 2021. [Citace: 26. března 2023.] https://vsni.co.uk/blogs/evolution-of-statistical-computing.
5. **Ozgur, Ceyhun, a další.** Selection of Statistical Software for Data Scientists. *Journal of Modern Applied Statistical Methods.* 2017, Sv. 16, 1.
6. **Muenchen, Robert A.** The Popularity of Data Science Software. *R Language Training & Data Science Market Share Analysis.* [Online] 2014. [Citace: 27. března 2023.] https://r4stats.com/articles/popularity/.
7. **Mauadi, Emad, a další.** Trends in the Usage of Statistical Software and Their Associated Study Designs in Health Sciences Research: A Bibliometric Analysis. *Cureus.* 2021, Sv. 13, 1.
8. **Mazouchová, Aneta, Jedličková, Tereza a Hlaváčová, Lucie.** Statistics Teaching Practice at Czech Universities with Emphasis on Statistical Software. *Journal on Efficiency and Responsibility in Education and Science.* 2021, Sv. 14, ISSN 2336-744X.
9. **Muenchen, Bob a MacKinnon, Sean.** Is Scholarly Use of R Beating SPSS Already? *R Language Training & Data Science Market Share Analysis.* [Online] 15. července 2019. [Citace: 26. března 2023.] https://r4stats.com/2019/07/15/is-scholarly-use-of-r-use-beating-spss-already/.
10. **Google.** V8 JavaScript engine. *V8 JavaScript engine.* [Online] Google. [Citace: 21. prosince 2022.] https://v8.dev/.
11. **The OpenJS Foundation.** Node.js. *Node.js.* [Online] The OpenJS Foundation. [Citace: 21. prosince 2022.] https://nodejs.org/en/.
12. **W3Techs.** Usage Statistics and Market Share of Node.js, December 2022. *W3Techs - extensive and reliable web technology surveys.* [Online] W3Techs, 2022. [Citace: 21. prosince 2022.] https://w3techs.com/technologies/details/ws-nodejs.
13. **Koďousková, Barbora.** Proč k vývoji webových aplikací použít technologii NodeJS? *WEB & MOBILE DEVELOPMENT AGENCY.* [Online] Rascasone, 13. dubna 2021. [Citace: 21. prosince 2022.] https://www.rascasone.com/cs/blog/node-js-architektura-moduly-npm.
14. **Foundation, The OpenJS.** Introduction to Node.js. *nodejs.dev.* [Online] The OpenJS Foundation;. [Citace: 21. prosince 2022.] The OpenJS Foundation;. https://nodejs.dev/en/learn/.
15. **Brown, Ethan.** *Web Development with Node and Express: Leveraging the JavaScript Stack.* Sebastopol : O'Reilly Media, 2014. ISBN 978-14-91949-30-6.
16. **Sapegin, Artem.** Webpack. [Online] https://webpack.js.org/. [Citace: 14. února 2023.] https://webpack.js.org/.
17. **MacWright, Tom.** Simple Statistics. *Simple Statistics.* [Online] 2023. [Citace: 27. 3 2023.] http://simple-statistics.github.io/.
18. **Norris, Trevor a Williams, Matthew.** jStat - JavaScript Statistical Library. *github.com.* [Online] Microsoft, 2023. [Citace: 21. 3 2023.] https://github.com/jstat/jstat.
19. **Davies, Jason.** GitHub - jasondavies/science.js: Scientific and statistical computing in JavaScript. *GitHub.* [Online] Microsoft, 15. září 2015. [Citace: 27. března 2023.] https://github.com/jasondavies/science.js.
20. **Provoost, Pieter.** GitHub - pieterprovoost/jerzy: JavaScript statistics library. *GitHub.* [Online] Microsoft, 9. října 2020. [Citace: 27. března 2023.] https://github.com/pieterprovoost/jerzy.

21. **Planitzer, Matthias.** statistics.js - Documentation. *GitHub.io.* [Online] Microsoft, 2017. [Citace: 27. března 2023.]
22. **Hagelbäck, Johan.** EZ Statistics. [Online] 28. dubna 2022. [Citace: 27. března 2023.] http://aiguy.freecluster.eu/ez_statistics/.
23. **Blank, Steve a Dorf, Bob.** *The Startup Owner's Manual. The Step-by-Step Guidee for Building a Great Company.* Pascadero : K&S Ranch Press, 2012. ISBN 978-0-9849993-0-9.
24. **Libby, Alex a Wellman, Dan.** *jQuery UI 1.10: The User Interface Library for jQuery.* Birmingham : Packt Publishing, 2013. ISBN 978-1-78216-220-9.
25. **Mužík, Pavel.** Dokumentace. *github.com.* [Online] [Citace: 30. března 2023.] https://github.com/muzikp/retusa/blob/main/docs/cs-CZ.md.
26. **Hendl, Jan.** *Přehled statistických metod zpracování dat. Analýza a metaanalýza dat.* Praha : Portál, 2004. ISBN 80-7178-20-1.
27. **Marek, Luboš, a další.** *Statistika v příkladech.* Praha : Kamil Mařík - Professional Publishing, 2015. ISBN 978-80-7431-153-6.
28. **Zaiontz, Charles.** Real Statistics Using Excel. *Real Statistics Using Excel.* [Online] 2023. [Citace: 26. března 2023.] https://real-statistics.com/.
29. **Glen, Stephanie.** Welcome to Statistics How To! *StatisticsHowTo.com: Elementary Statistics for the rest of us!* [Online] [Citace: 26. března 2023.] https://www.statisticshowto.com/.
30. **Bootstrap Team.** The most popular HTML, CSS, and JS library in the world. [Online] [Citace: 14. února 2023.] https://getbootstrap.com/.
31. **Edge, James.** *Lean: An Essential Guide to Lean Startup, Lean Six Sigma, Lean Analytics, Lean Enterprise, Lean Manufacturing, Agile Project Management, Kanban and Scrum.* místo neznámé : CreateSpace Independent Publishing Platform, 2018. ISBN 978-1729813263.
32. **Dolza, Marcelo.** *iziToast.* [Online] 2023. [Citace: 14. února 2023.] https://izitoast.marcelodolza.com/.
33. **Wen, Zhixin.** An extended table to the integration with some of the most widely used CSS frameworks. (Supports Bootstrap, Semantic UI, Bulma, Material Design, Foundation). *Bootstrap Table.* [Online] 2019. [Citace: 30. března 2023.] https://bootstrap-table.com/.
34. **ChartJS.** *Chart.js | Open source HTML5 Charts for your website.* [Online] [Citace: 11. února 2023.] https://www.chartjs.org/.
35. **Brown, Tim.** *Change by Design, Revised and Updated: How Design Thinking Transforms Organizations and Inspires Innovation.* místo neznámé : Harper Business, 2019. ISBN 978-0062856623.
36. **LUMA Institute.** *Innovating for People Handbook of Human-Centered Design Methods.* Pittsburg : LUMA Institute, 2012. ISBN 978-0-9857509-0-9.
37. **Knapp, Jake, Zeratsky, John a Kowitz, Braden.** *Sprint: Jak vyřešit velké problémy a otestovat nové myšlenky v pouhých pěti dnech.* Brno : Jan Melvil Publishing, 2017. ISBN 978-80-7555-023-1.
38. **Hendl, Jan.** *Kvalitativní výzkum: základní metody a aplikace.* Praha : Portál, 2005. ISBN 80-7367-040-2.
39. **Poland, Stephen.** *Founder's Pocket Guide: Founder Equity Splits.* místo neznámé : 1x1 Media, 2016. ISBN 978-1-938162-09-1.
40. **Poland, Stephen R.** *Founder's Pocket Guide: Startup Valuation. 2nd Edition.* místo neznámé : 1x1 Media, 2017. ISBN 978-1-938162-04-6.
41. **Guillebeau, Chris.** *Startup za pakatel. Objevte způsob, jak pracovat na sebe a živit se tím, co vás baví.* Brno : Jan Melvil Publishing, 2015. ISBN 978-80-87270-59-2.
42. **Korolov, Sergiy.** 5 Lean Canvas Examples of Multi-Billion Startups. *Railsware Blog.* [Online] Railsware Solutions FZ, 8. března 2023. [Citace: 9. dubna 2023.] https://railsware.com/blog/5-lean-canvas-examples/.
43. **Pirátská strana.** MIT licence - Pirati.CZ. *Pirati.CZ.* [Online] Pirátská strana, 1. února 2023. [Citace: 17. března 2023.] https://wiki.pirati.cz/kci/mit_licence.

44. **GitHub.** Licensing a repository. *github.com.* [Online] Microsoft. [Citace: 17. března 2023.] https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/licensing-a-repository.
45. **ČVUT.** FAQ - Autorské právo. *cvut.cz.* [Online] České vysoké učení technické, 27. 11 2018. [Citace: 17. 3 2023.] http://knihovna.cvut.cz/seminare-a-vyuka/ochrana-dusevniho-vlastnictvi/faq-autorske-pravo?start=2.
46. **Dimitrov, Dimitre.** *Software Project Estimation: Intelligent Forecasting, Project Control, and Client Relationship Management.* New York : Apress, 2020. ISBN 978-1-4842-5025-9.
47. **TOPOLOVÁ, Ivana, Markéta KUBÁLKOVÁ a Tomáš KUBÁLEK.** *Manažerská informatika. Textový procesor Microsoft Word 2016.* Praha : Nakladatelství Oeconomica, 2017. 978-80-245-2198-5.
48. **KUBÁLEK, Tomáš a Markéta KUBÁLKOVÁ.** *Program pro tvorbu diagramů Microsoft Visio 2013.* Praha : Nakladatelství Oeconomica, 2015. 978-80-245-2104-6.

Seznam obrázků

Seznam tabulek