EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103006/B/2025/36146835375526148

MATEMATICKÉ A ŠTATISTICKÉ GRAFY V JAZYKU R PROSTREDNÍCSTVOM SHINY APLIKÁCIÍ

Bakalárska práca

2025 **Timea Blahutová**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

MATEMATICKÉ A ŠTATISTICKÉ GRAFY V JAZYKU R PROSTREDNÍCSTVOM SHINY APLIKÁCIÍ

Bakalárska práca

Študijný program: Hospodárska informatika

Študijný odbor: Ekonómia a manažment

Školiace pracovisko: Katedra matematiky a aktuárstva

Vedúci záverečnej práce: Mgr. František SLANINKA, PhD.

Bratislava 2025 **Timea Blahutová**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Timea Blahutová
Študijný program: hospodárska informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: ekonómia a manažment
Typ záverečnej práce: Bakalárska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Matematické a štatistické grafy jazyka R prostrednictvom Shiny aplikácií

Anotácia: Cieľom práce je analýza grafických možností jazyka R, analýza balíkov funkcií v R súvisiacich so zobrazovaním štatistických a matematických grafov, ako aj grafov z oblasti diskrétnej matematiky. Uvedené poznatky by mali byť následne využité na tvorbu dynamicky orientovaných Shiny aplikácií, ktoré umožňujú užívateľsky prívetivo riešiť problematiku zobrazenia grafov a vizualizácie dát. Autor môže uvedené poznatky prezentovať napríklad formou webovej stránky.

Vedúci: Mgr. František Slaninka, PhD.
Oponent: doc. Ing. Michal Páleš, PhD.
Katedra: KMA FHI - Katedra matematiky a aktuárstva
Vedúci katedry: doc. Ing. Michal Páleš, PhD.

Dátum zadania: 07.03.2024

Dátum schválenia: 11.03.2024

doc. Ing. Martin Mišút, CSc.
osoba zodpovedná za realizáciu študijného programu

Poďakovanie

Touto cestou by som chcela vyjadriť poďakovanie vedúcemu záverečnej práce pánovi Mgr. Františkovi Slaninkovi, PhD., za odborné vedenie, cenné rady, pripomienky a ochotu, ktorú mi počas písania práce venoval.

Čestné vyhlásenie

Čestne vyhlasujem, že som túto bakalársku prácu s názvom „ Matematické a štatistické grafy v jazyku R prostredníctvom Shiny aplikácií“ vypracovala samostatne pod vedením školiteľa Mgr. Františka Slaninku, PhD., s použitím nižšie uvedenej literatúry.

Bratislava, dňa 30.04.2025 .....

podpis autora

Abstrakt

BLAHUTOVÁ, Timea: *Matematické a štatistické grafy v jazyku R prostredníctvom Shiny aplikácií.* – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra matematiky a aktuárstva. – Vedúci záverečnej práce: Mgr. František Slaninka, PhD. – Bratislava: FHI EU, 2025, 53 s.

Cieľom bakalárskej práce je preskúmať možnosti vizualizácie matematických a štatistických grafov v programovacom jazyku R s využitím balíka Shiny na tvorbu interaktívnych webových aplikácií. Údaje pochádzajú z verejne dostupných webových databáz, ako aj z vlastných príkladov vytvorených pre účely tejto práce. Na ich vizualizáciu používame knižnice ggplot2, základnú funkciu plot a balík igraph.
Výsledkom práce je interaktívna aplikácia, ktorá umožňuje používateľovi vizuálne a intuitívne pracovať s rôznymi typmi grafov z oblasti matematiky, diskrétnej matematiky a štatistiky.
Pridaná hodnota práce spočíva v prepojení teoretických konceptov so zrozumiteľným praktickým nástrojom, ktorý môže slúžiť na edukáciu a prezentáciu údajov.

Kľúčové slová: jazyk R, Shiny, graf, vizualizácia údajov, štatistika, diskrétna matematika, aplikácia

Abstract

BLAHUTOVÁ, Timea: *Mathematical and Statistical Graphs in the R Language through Shiny Applications.* – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Mathematics and Actuarial Science. – Thesis supervisor: Mgr. František Slaninka, PhD. – Bratislava: FEI UE, 2025, 53 p.

The aim of this bachelor's thesis is to explore ways of visualizing mathematical and statistical graphs in the R programming language, using the Shiny package to build interactive web applications. The data originate from publicly accessible web repositories as well as from custom examples created specifically for this study. Their visualisation is carried out with the ggplot2 library, the base plot function, and the igraph package.

The outcome of the thesis is an interactive application that allows users to work visually and intuitively with various types of graphs in mathematics, discrete mathematics, and statistics. The added value of this work lies in linking theoretical concepts to a clear, practical tool that can serve both educational and data-presentation purposes.

Keywords: R language, Shiny, graph, data visualization, statistics, discrete mathematics, application

Obsah

Zoznam obrázkov a tabuliek

Úvod

Vizualizácia dát je podstatnou súčasťou analýzy údajov. Predstavuje dôležitý nástroj na spracovanie a prezentáciu informácií v rámci štatistických a matematických analýz. Správna interpretácia a znázornenie dát dokáže nielen uľahčiť pochopenie komplexných vzorcov a trendov, ale tiež zefektívniť rozhodovací proces v rôznych oblastiach, ako sú veda, školstvo alebo podnikanie. Jedným z často používaných nástrojov pri vizualizácii a analýze dát je programovací jazyk R.

Cieľom tejto bakalárskej práce je analyzovať grafické možnosti programovacieho jazyka R so zameraním na dostupné balíky určené na tvorbu štatistických, matematických a diskrétnych grafov. Získané poznatky budú následne aplikované pri vývoji dynamicky orientovaných Shiny aplikácií, ktoré umožnia používateľom jednoduchým a intuitívnym spôsobom pracovať s vizualizáciou dát a grafov.

V teoretickej časti práce si definujeme pojem graf, vysvetlíme jeho význam a základné vlastnosti. Následne si priblížime rôzne typy grafov, ktoré sa využívajú v oblasti matematiky a štatistiky. V metodickej časti opisujeme programovací jazyk R a balík Shiny, ktoré slúžia na tvorbu interaktívnych aplikácií.

V praktickej časti práce sa zameriame na tvorbu Shiny aplikácie v prostredí RStudio. Aplikácia je určená na vizualizáciu vybraných matematických, štatistických a diskrétnych grafov, pričom umožňuje používateľovi interaktívne pracovať s údajmi a sledovať ich grafické znázornenie.

1 Súčasný stav problematiky doma a v zahraničí

V úvodnej kapitole sa oboznámime so základnými pojmami z oblasti grafov, ktoré sa využívajú v matematike a štatistike. Podrobnejšie sa zameriame na rôzne typy grafov, s ktorými budeme pracovať v ďalších častiach tejto práce.

1.1 Graf

Graf predstavuje vizuálnu reprezentáciu súboru údajov, ktorý umožňuje prehľadné zhrnutie komplexných informácií a znázornenie vzťahov medzi rôznymi premennými. Používanie grafov je efektívnym spôsobom, ako demonštrovať trendy a prepojenia v dátach, čím sa uľahčuje ich interpretácia.

Pre mnohých ľudí je jednoduchšie porozumieť numerickým alebo zložitým údajom, ak sú prezentované vizuálnou formou, ako je napríklad graf. Tento prístup pomáha rýchlejšie spracovať informácie a získať lepší prehľad o vzťahoch v analyzovaných dátach. (Indeed Editorial Team, 2025)

Jedna z mnohých definícií uvádza, že „graf je kresba vytvorená podľa určitých dohodnutých a uznávaných pravidiel, ktorá znázorňuje kvantitatívne alebo kvalitatívne informácie.“ (Roubiček, 1967, p. 57)

Grafy nachádzajú široké uplatnenie v rôznych odboroch, ako sú matematika, bankovníctvo, informatika, biológia, sociológia, ekonómia, fyzika a mnohé ďalšie vedecké disciplíny. V informatike sa grafy využívajú na reprezentáciu dátových štruktúr, zatiaľ čo v matematike sú dôležitým nástrojom pri štúdiu teórie grafov. V sieťovej analýze umožňujú grafové algoritmy určiť najkratšiu cestu medzi bodmi a analyzovať spôsob prepojenia uzlov v grafe. V strojovom učení sa grafové modely uplatňujú pri klasifikácii, predikcii a zoskupovaní údajov. V biológii sa využívajú na znázornenie biologických sietí a analýzu vzťahov medzi proteínmi či sekvenciami DNA. (Brasseur, 2003)

1.2 Vlastnosti grafov

Grafy predstavujú súbor grafických prvkov, ktoré slúžia na vizuálne znázornenie údajov. Ich výhodou je prehľadnosť a jednoduchá interpretácia. Grafická reprezentácia údajov zohráva kľúčovú úlohu pri predbežnej analýze dát.

Každý graf sa skladá z viacerých komponentov, medzi ktoré patria súradnicová sústava, stupnica, grafická sieť, názov grafu a legenda.

Jedným z kľúčových prvkov pri výslednej úprave grafu je jeho názov, ktorý sa zvyčajne nachádza nad samotným grafom. Jeho úlohou je stručne a jednoznačne opísať, aké údaje graf znázorňuje a čo v ňom sledujeme.

Grafy sa najčastejšie zobrazujú na súradnicovej sústave, pričom najznámejšia je pravouhlá súradnicová sústava, ktorá pozostáva z osí, ktoré sú na seba kolmé. Horizontálna os sa označuje ako *x* a vertikálna os nesie označenie *y*. Každá os musí mať vyznačené jednotky, pričom jednotky môžu byť pre jednotlivé osi odlišné – napríklad jedna os môže zobrazovať metre, a druhá sekundy.

Stupnica je čiara s označenými bodmi, ktoré reprezentujú konkrétne hodnoty. Vzdialenosť medzi týmito bodmi sa nazýva grafický interval, zatiaľ čo rozdiel číselných hodnôt tvorí číselný interval. Grafická sieť napomáha lepšej orientácii v číselných hodnotách uvedených v grafe.

Údaje v grafe môžu byť vizualizované rôznymi spôsobmi, pričom môžu obsahovať aj poznámky, ktoré poskytujú doplňujúce informácie o konkrétnych hodnotách na grafe. Hodnoty môžu byť znázornené bodmi, geometrickými tvarmi alebo čiarami, pričom je možné využiť rozmanité farebné odtiene a vzory na ich rozlíšenie.

V prípadoch, keď graf zobrazuje viacero premenných, je vhodné použiť legendu, ktorá obsahuje ich zoznam. Legenda umožňuje rýchlu orientáciu v grafe, čím pomáha jednoznačne priradiť údaje k jednotlivým premenným. (Markechová et al., 2011)

1.3 Matematické grafy

Grafy slúžia na prehľadné znázornenie vzťahov a prepojení medzi objektami. V matematike nachádzajú uplatnenie v rôznych oblastiach, ako sú kombinatorika, teória čísel či algoritmy.

„Graf je sústava bodov (vrcholov) pospájaných čiarami (hranami). Tieto hrany môžu, ale nemusia mať predpísaný smer (orientáciu)“ (Bosák, 1980, p**. 7**).

Formálne je graf definovaný ako usporiadaná dvojica množín G=(V,E), kde V je množina vrcholov a E je množina hrán, ktoré prepájajú dvojice vrcholov.

Štúdium grafov patrí do oblasti matematiky zvanej teória grafov, ktorá sa zaoberá štúdium a vlastnosťami grafov, a často slúži na modelovanie reálnych problémov či štruktúr. Typickým príkladom je hľadanie najkratšej cesty v sieti, úloha obchodníka, či modelovanie

chemických zlúčenín. Je dôležitou súčasťou diskrétnej matematiky, ktorá sa zameriava na štúdium celých čísel a zahŕňa informatiku, pravdepodobnosť, algoritmy a teóriu čísel.

Poznáme tri typy grafov:

- Neorientovaný graf: hrany nemajú smer
- Orientovaný graf: hrany majú určený smer
- Zmiešaný graf: kombinácia orientovaných a neorientovaných hrán.

Na rozdiel od kartézskych grafov, ktoré pracujú s osami x a y, grafy v teórii grafov reprezentujú sieťové štruktúry a ich vlastnosti sa skúmajú pomocou kombinatorických metód a algoritmov.

Použitie grafov prináša výhodu najmä v tom, že umožňuje vizuálne a prehľadné znázornenie riešeného problému. Takéto grafické zobrazenie býva často pre používateľa zrozumiteľnejšie než matematická formulácia pomocou modelov. (Bosák, 1980)

1.3.1 Funkcie

Dôležitým pojmom v matematike je funkcia, ktorá definuje vzťah medzi závislou a nezávislou premennou. Funkcie sú základom rôznych matematických disciplín a nachádzajú široké uplatnenie v oblastiach, ako sú algebra, analýza, fyzika či ekonomika.

Už v minulosti si ľudia uvedomovali, že medzi prírodnými javmi existujú rôzne vzťahy. Napríklad, čím väčšiu korisť ulovili, tým viac ľudí mohli nasýtiť. V 18. storočí Leonhard Euler popísal premenlivé a konštantné veličiny a na ich základe definoval pojem funkcie. Funkciu vnímal ako výraz, ktorý je zostavený z premennej, čísel a konštánt.

V matematike je funkcia zobrazenie, ktoré každému prvku x z množiny A priradí práve jeden prvok y z množiny B, pričom platí, že $A \neq \emptyset$ *a* $B \neq \emptyset$. Tento vzťah sa bežne symbolizuje ako $y = f(x)$. To znamená, že $f(x)$ nemôže mať viac ako jednu hodnotu pre to isté x.

Funkciu môžeme definovať štyrmi spôsobmi:

1. Vymenovaním prvkov – funkciu vyjadrujeme ako množinu usporiadaných dvojíc
2. Tabuľkou – hodnota funkcie pre rôzne vstupy je zapísaná v tabuľkovej forme
3. Rovnicou – funkcia je definovaná algebraickým predpisom

4. Grafom – funkcia je vizualizovaná grafickým znázornením závislosti medzi premennými. (Gilbert, 2025)

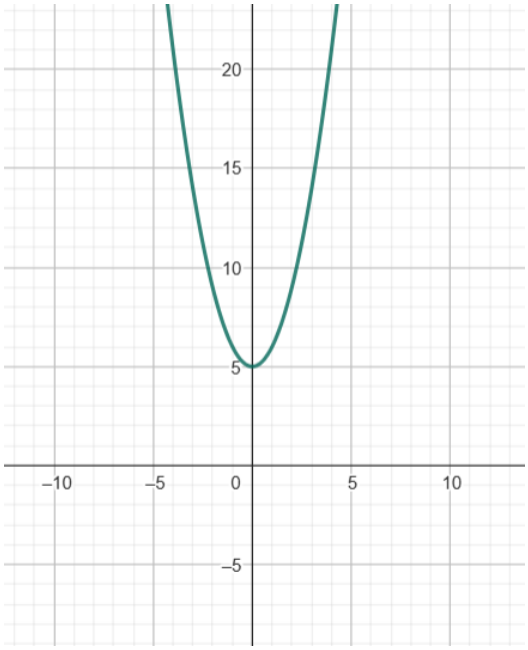
1.3.2 Graf funkcie

Funkciu môžeme reprezentovať pomocou grafov. Vizuálne informácie, ktoré grafy poskytujú, často uľahčujú pochopenie vzťahov medzi premennými.

Grafy zobrazujeme tak, že vstupné hodnoty (nezávislá premenná) sú umiestnené na horizontálnej osi (osi *x*) a výstupné hodnoty (závislá premenná) sa nachádzajú na vertikálnej osi (osi *y*).

Grafom funkcie *f* nazývame množinu všetkých bodov so súradnicami [*x; f(x)*], zostrojenú v pravouhlej súradnicovej sústave, pričom $x \in D(f)$. (Gilbert,, 2025)

Obrázok 1 Graf funkcie *f(x)=x^2+5* v programe GeoGebra



Zdroj: Vlastné spracovanie

1.4 Štatistika

Štatistika ako vedná disciplína zahŕňa súbor pracovných činností a postupov, ktorých cieľom je získavanie, spracovanie a vyhodnocovanie informácií o vlastnostiach hromadných javov a procesov. Táto postupnosť nadväzujúcich krokov sa označuje ako štatistické skúmanie, ktoré prebieha v troch základných etapách.

Prvou etapou je štatistické zisťovanie, ktoré slúži na získavanie údajov o hromadných javoch a procesoch. V súčasnosti existuje viacero metód štatistického zisťovania, ako sú ankety, dotazníky, výkazy, a ďalšie.

Druhou etapou je štatistické spracovanie údajov, ktoré zahŕňa súbor pracovných postupov od kontroly získaných údajov, ich usporiadania a prenosu až po prezentáciu v prehľadnej a názornej forme. Spracovanie údajov tiež zahŕňa výpočet opisných číselných charakteristík, ktoré umožňujú lepšie pochopenie analyzovaných dát.

Poslednou etapou je štatistická analýza, ktorá predstavuje záverečnú fázu štatistického skúmania. V tejto etape sa získané výsledky z predchádzajúcich fáz vyhodnocujú pomocou štatistických metód, následne sa porovnávajú a na ich základe sa formulujú závery. Táto etapa poskytuje dôležité podklady pre rozhodovanie v rôznych oblastiach, ako sú veda, ekonomika či sociálna politika. (Labudová, 2021)

Vizualizácia údajov zohráva významnú úlohu v štatistickom skúmaní, pretože umožňuje lepšie porozumieť analyzovaným informáciám nielen odborníkom, ktorí s nimi pracujú, ale aj širšiemu publiku, ktorému sú výsledky určené. Na zabezpečenie zrozumiteľnosti sa údaje často prezentujú v podobe tabuliek alebo grafov, čo umožňuje efektívne a prehľadné sprostredkovanie kľúčových zistení.

1.4.1 Štatistické pojmy

Štatistika sa zaoberá skúmaním hromadných javov. Existujú dva typy hromadných javov. Prvý typ je založený na potrebe veľkého počtu opakovaných pozorovaní konkrétnej vlastnosti objektu. Druhý, bežnejší typ, sa týka vlastnosti určitej množiny, ktorá obsahuje veľké množstvo prvkov, pričom každý z nich túto vlastnosť vykazuje v určitej miere.

Opakom hromadného javu je individuálny jav, pri ktorom sa sleduje jedna vlastnosť jedného prvku. Objekt, udalosť alebo proces, ktorý vykazuje individuálny jav, sa označuje ako štatistická jednotka. Štatistické jednotky musia mať presne identifikované:

- Priestorové hľadisko – určuje kde sa štatistické jednotky nachádzajú (počet obyvateľov v rôznych krajoch Slovenska)
- Časové hľadisko – definuje kedy boli údaje získané alebo k akému obdobiu sa vzťahujú (nezamestnanosť na Slovensku v roku 2024)
- Vecné hľadisko – určuje čo presne je predmetom skúmania (priemerný vek zamestnancov v určitej firme).

Štatistické jednotky, ktoré majú spoločné vlastnosti, tvoria štatistický súbor. Počet štatistických jednotiek v tomto súbore sa nazýva rozsah štatistického súboru.

Každá štatistická jednotka disponuje rôznymi vlastnosťami alebo atribútmi, ktoré sa nazývajú štatistické znaky. Štatistický znak predstavuje vonkajší merateľný prejav skúmanej vlastnosti štatistickej jednotky.

Podľa vyjadrenia hodnôt a spôsobu merania ich delíme na:

- Nominálne znaky
- Ordinálne znaky
- Metrické resp. kardinálne znaky

Nominálne znaky predstavujú kategórie vyjadrené slovnými hodnotami, ktoré medzi sebou nemožno porovnávať. Príkladmi sú farba očí, národnosť či pohlavie. Ordinálne znaky umožňujú zoradenie do určitej hierarchie podľa dôležitosti, avšak rozdiely medzi hodnotami nie sú presne určiteľné. Príkladom sú známky zo skúšky, dosiahnuté vzdelanie alebo pracovná pozícia. Metrické znaky sú číselné hodnoty, ktoré sa dajú vzájomne porovnávať a umožňujú vykonávať matematické operácie. Napríklad počet áut v rodine, výška osoby alebo mesačný príjem. (Labudová, 2021)

1.4.2 Typy štatistických grafov

Grafy predstavujú vizuálny spôsob spracovania štatistických údajov, vďaka čomu môžeme lepšie porozumieť vzájomným vzťahom medzi jednotlivými štatistickými prvkami. Ich hlavnou výhodou je zlepšenie prehľadnosti a jednoduchšia orientácia v prezentovaných informáciách.

Existuje široká škála grafických reprezentácií, pričom výber konkrétneho typu závisí od charakteru získaných údajov a od toho, aké informácie chceme zdôrazniť alebo vizualizovať. Každý druh grafu má svoje špecifiká, ktoré môžu byť v niektorých prípadoch výhodné, no v iných môžu predstavovať isté obmedzenia. V tejto práci sa zameriame na hlavné kategórie grafov, ktoré budú podrobnejšie opísané v nasledujúcich častiach.

Stĺpcový graf

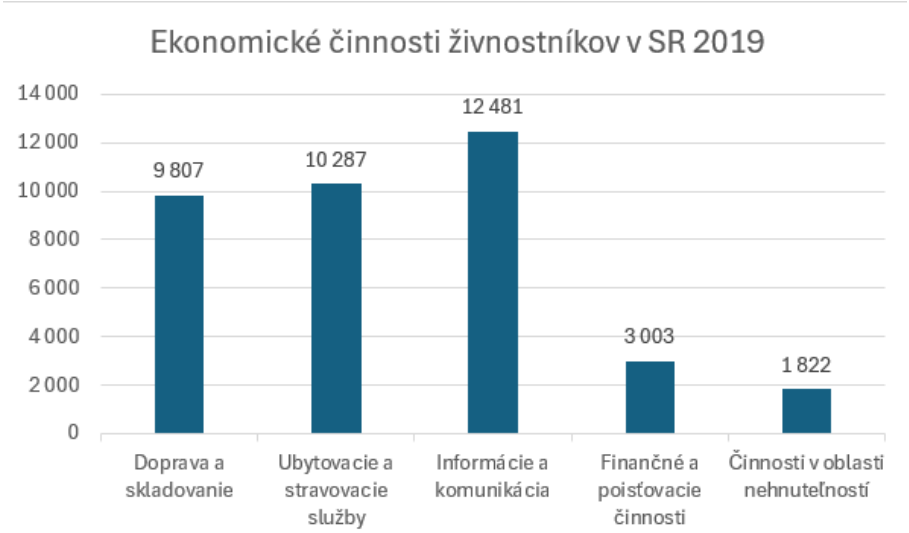
Stĺpcový graf patrí medzi grafické metódy vizualizácie údajov a je obzvlášť vhodný na znázornenie hodnôt kvalitatívnych premenných v určitých časových obdobiach alebo na vzájomné porovnanie viacerých premenných.

Tento typ grafu sa často využíva na prezentáciu vývoja hodnôt v čase (napríklad pri analýze trendov) alebo na ilustráciu rozdielov medzi jednotlivými kategóriami. Pri časových radoch umožňuje jasne sledovať zmeny v hodnotách premenných, zatiaľ čo v prípade kategorizovaných údajov poskytuje prehľadnú komparáciu medzi rôznymi skupinami.

Typická štruktúra stĺpcového grafu zahŕňa kategórie alebo časové údaje umiestnené na vodorovnej osi, pričom výška stĺpcov na zvislej osi vyjadruje hodnoty sledovaných premenných. Stĺpcový graf môže byť orientovaný vertikálne alebo horizontálne, pričom ich spoločnou vlastnosťou je, že dĺžka alebo výška stĺpcov zodpovedá hodnote danej premennej. Tento spôsob vizualizácie umožňuje jednoduchú interpretáciu dát a efektívne porovnávanie rôznych hodnôt. (Rosták, 2023)

Nižšie uvádzame stĺpcový graf, ktorý sme vytvorili v programe Microsoft Excel na základe údajov zo Štatistického úradu Slovenskej republiky. Graf znázorňuje počet živnostníkov v Slovenskej republike v roku 2019 podľa vybraných ekonomických činností. Tento stĺpcový graf umožňuje jednoduché porovnanie zastúpenia jednotlivých odvetví.

Obrázok 2 Stĺpcový graf Ekonomická činnosť živnostníkov v SR 2019



Zdroj: Vlastné spracovanie

Histogram

Pri analýze rozdelenia číselných (kardinálnych) premenných sa často využíva histogram. Tento typ grafu zobrazuje frekvenciu výskytu hodnôt v rámci daného rozdelenia. Vyzerá podobne ako stĺpcový graf, ale na rozdiel od stĺpcových grafov, ktoré sa používajú

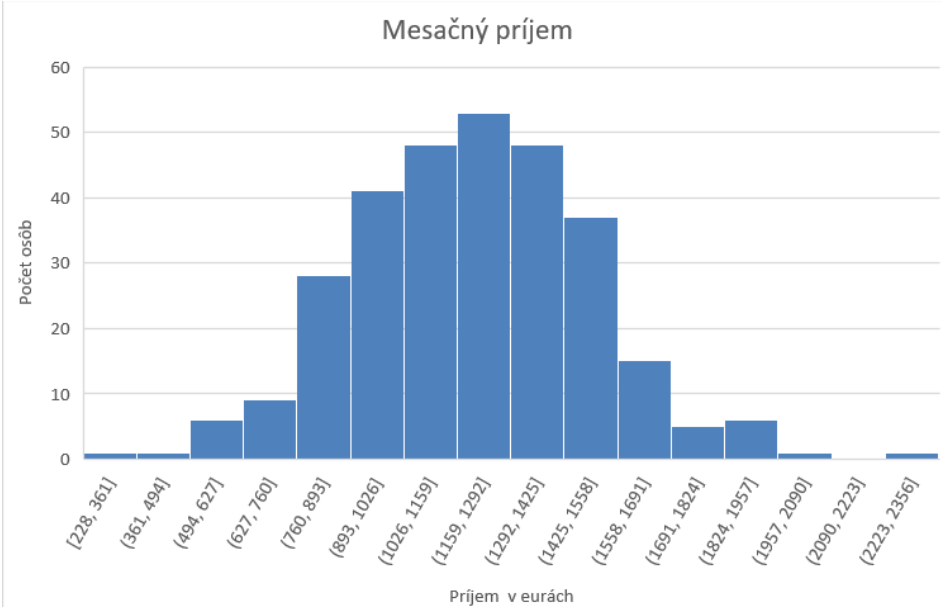
pre kategorické údaje, sú histogramy navrhnuté pre spojité údaje a zoskupujú ich do logických rozsahov.

Histogram pozostáva zo stĺpcov, ktoré reprezentujú rozsahy hodnôt (intervaly) a ich výška zodpovedá početnosti údajov v danom intervale. Vodorovná os môže obsahovať bodové alebo intervalové hodnoty, zatiaľ čo na zvislej osi sú uvedené frekvencie výskytu jednotlivých hodnôt.

Histogramy sú obzvlášť užitočné pri skúmaní rozloženia údajov, identifikácii symetrie, šikmosti alebo extrémnych hodnôt (outlierov). (Chen, 2022)

Uvádzame príklad histogramu vytvoreného v programe Microsoft Excel, ktorý znázorňuje rozdelenie mesačného príjmu medzi 300 respondentmi. Tento graf poskytuje prehľad o frekvencii jednotlivých intervalov príjmov a umožňuje identifikovať, v ktorých príjmových kategóriách sa respondenti najčastejšie nachádzajú. Údaje použité na tvorbu tohto grafu boli vygenerované umelou inteligenciou.

Obrázok 3 Histogram Mesačný príjem



Zdroj: Vlastné spracovanie

Koláčový graf

Koláčový graf je typ kruhového grafu, ktorý je rozdelený na jednotlivé výseky (sektory). Každý výsek znázorňuje určitú časť z celkového súboru údajov. Tento typ grafu je vhodný najmä na znázornenie nominálnych alebo ordinálnych kategórií údajov, pričom

poskytuje jasný prehľad o percentuálnych alebo proporcionálnych podieloch jednotlivých zložiek.

V praxi sa koláčový graf často používa na porovnávanie rôznych oblastí rastu v rámci podnikania, napríklad pri analýze zisku alebo podielov na trhu. Aby bol prehľadný a efektívny, odporúča sa používať ho na zobrazenie údajov pre 3 až 5 kategórií, pretože pri väčšom počte môže byť interpretácia výsledkov menej prehľadná. (YI, 2025)

Ako príklad uvádzame koláčový graf, ktorý znázorňuje percentuálne rozdelenie živnostníkov na Slovensku podľa vybraných ekonomických činností v roku 2019. Graf umožňuje porovnať podiel jednotlivých sektorov, ako sú informácie a komunikácia, ubytovacie a stravovacie služby, doprava a skladovanie, finančné činnosti a činnosti v oblasti nehnuteľností. Údaje boli prevzaté zo Štatistického úradu Slovenskej republiky.

Obrázok 4 Koláčový graf Ekonomická činnosť živnostníkov v SR 2019

Zdroj: Vlastné spracovanie

Krabicový graf (box plot)

Box plot (krabicový graf) je štatistická vizualizácia, ktorá slúži na zobrazenie rozloženia číselných údajov vrátane ich šikmosti, rozptylu a polohy. Graf využíva kvartily na znázornenie dát: základ tvorí obdĺžnik (krabica), ktorý reprezentuje rozsah od prvého po tretí kvartil. V jeho strede sa nachádza vodorovná čiara označujúca medián (druhý kvartil). Z krabice vedú na obe strany úsečky (tzv. „fúzy“), ktoré znázorňujú minimálnu a maximálnu hodnotu. Horný okraj krabice predstavuje tretí kvartil a dolný okraj prvý kvartil.

Krabicový graf sa používa na zobrazenie rozdelenia hodnôt číselných údajov, najmä ak ich chceme porovnať medzi viacerými skupinami. Sú zostavené tak, aby poskytovali všeobecné informácie o symetrii, odchýlke, rozptyle a odľahlých hodnotách skupiny údajov. (YI, 2023)

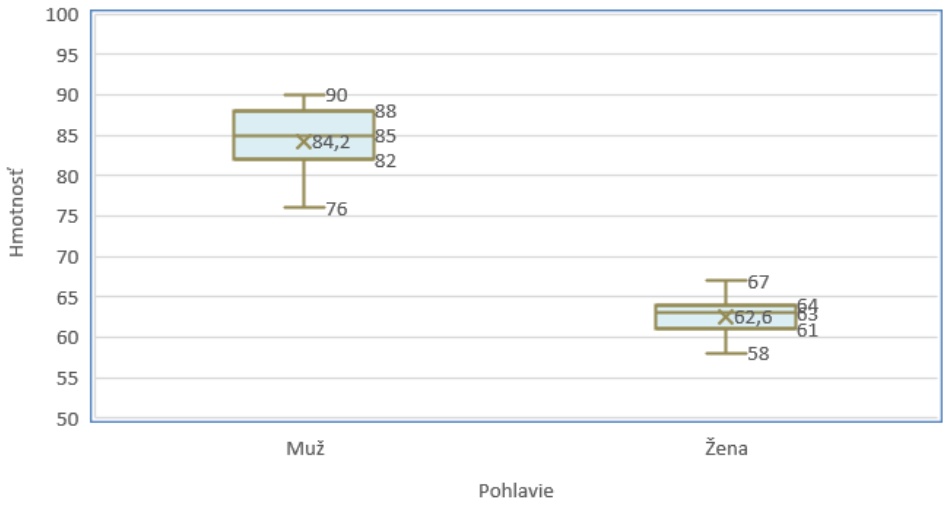
V programe Microsoft Excel sme vytvorili tabuľku s náhodnými údajmi o hmotnosti 6 mužov a 6 žien. Následne sme na základe týchto dát zostrojili boxplot, ktorý vizualizuje medián hmotnosti pre obe skupiny, spolu s minimálnymi a maximálnymi hodnotami. Tento graf nám umožňuje lepšie pochopiť rozloženie údajov a identifikovať prípadné extrémne hodnoty. Boxplot, ktorý porovnáva rozdelenie číselnej premennej medzi viacerými skupinami, sa označuje ako Multiple Box Plot.

Tabuľka 1 Hmotnosť

Pohlavie	Hmotnosť					
Žena	64	58	61	67	63	59
Muž	82	90	76	85	88	84

Zdroj: Vlastné spracovanie

Obrázok 5 Boxplot Hmotnosť



Zdroj: Vlastné spracovanie

2 Cieľ práce

Cieľom bakalárskej práce je riešiť problematiku zobrazenia matematických, štatistických a diskrétnych grafov v jazyku R prostredníctvom Shiny aplikácie. Vizualizácia údajov je zjednodušená pomocou grafov, ktoré slúžia ako univerzálne nástroje na prezentáciu dát.

Na splnenie stanoveného cieľa sa najskôr oboznámime s programovacím jazykom R, jeho vývojovým prostredím R studio a základnými princípmi, na ktorých je tento jazyk postavený. V tejto práci sa sústredíme na štyri kľúčové štatistické vizualizácie – stĺpcový graf, koláčový graf, boxplot a histogram. V rámci matematickej analýzy budeme vykresľovať grafy funkcií a v oblasti diskrétnej matematiky použijeme grafové modely s hranami na znázornenie vzťahov.

Na tento účel budeme pracovať s viacerými balíkmi, ktoré slúžia na tvorbu grafov – najmä s balíkom ggplot2 na vykresľovanie matematických a štatistických grafov a s balíkmi igraph a ggraph na vizualizáciu grafov z oblasti diskrétnej matematiky.

Na dosiahnutie stanoveného cieľa sme definovali nasledovné čiastkové ciele:

- Stručne opísať a predstaviť programovací jazyk R, vývojové prostredie RStudio a balík Shiny, ktoré sú nevyhnutné na vytvorenie efektívnej a interaktívnej aplikácie.
- Implementovať jednotlivé grafické zobrazenia tak, aby výsledná aplikácia bola funkčná, používateľsky prívetivá a intuitívna.
- Vysvetliť jednotlivé časti implementovaného kódu, ich účel a funkcionalitu pre lepšie pochopenie fungovania aplikácie.
- Demonštrovať správanie aplikácie na vybraných príkladoch.

3 Metodika práce a metódy skúmania

V tejto kapitole sa oboznámime s programovacím jazykom R, vývojovým prostredím RStudio a balíkom Shiny, ktorý umožňuje tvorbu interaktívnych webových aplikácií priamo v R a opíšeme ich základné vlastnosti.

3.1 Programovací jazyk R

Jazyk R je voľne šíriteľné prostredie, ktoré sa používa na štatistické výpočty, grafickú prezentáciu a vizualizáciu údajov. Je napísaný pre rôzne typy platforiem a používa sa na analýzu údajov v rôznych oblastiach ako napríklad vo výrobe, zdravotníctve, financiách, bio informatike a v mnohých iných odvetviach.

Z historického hľadiska je R implementáciou podobnou jazyku a prostrediu S, ktoré vyvinuli John Chambers a jeho kolegovia v Bell Laboratories. R je teda nástupcom jazyka S a zároveň súčasťou projektu GNU. Prvotná verzia jazyka R bola vydaná v roku 1995 Rossom Ihakom a Robertom Gentlemanom a na jeho vývoji pracoval tím R Development Core Team. Názov „R“ je teda odvodený od počiatočných písmen mien jeho dvoch hlavných vývojárov.

R je široko používaný v data science na analýzu údajov. Ponúka rozsiahlu sadu balíkov a knižníc slúžiacich na manipuláciu s údajmi, štatistické a lineárne modelovanie a grafickú vizualizáciu. Obsahuje veľké množstvo knižníc, ktoré sa dajú využiť na komplexnú štatistickú prácu vrátane implementácie regresných modelov, analýzu priestorových a časových radov a modelovanie trendov.

V akademickom prostredí je populárny vďaka svojim funkciám a skutočnosti, že je možné ho bezplatne stiahnuť v súlade s podmienkami všeobecnej verejnej licencie. Kompiluje a beží na rôznych platformách ako Unix, Linux, Windows a macOS.

Programovacie prostredie v jazyku R je založené na štandardnom rozhraní príkazového riadka, ktoré umožňuje používateľovi pracovať s údajmi efektívnym spôsobom. Používatelia tak môžu čítať údaje a načítať ich do pracovného priestoru, zadávať príkazy a prijímať výsledky. Príkazy môžu byť čokoľvek od jednoduchých matematických operátorov až po zložitejšie funkcie. (Robinoson, 2024)

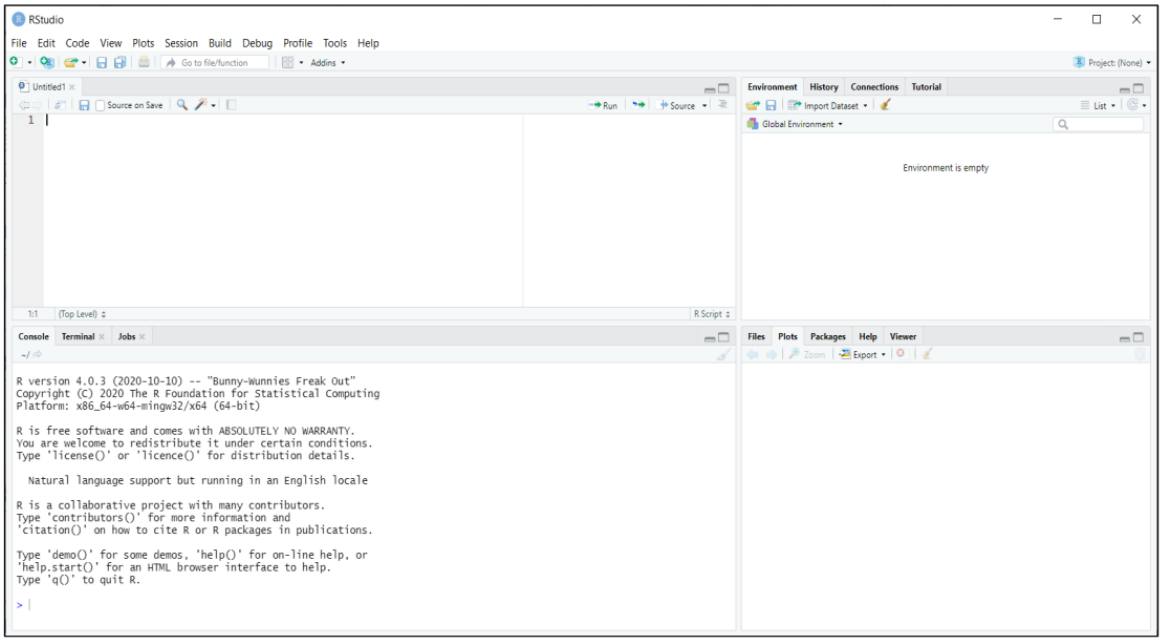
3.2 R studio

R studio je bezplatné integrované vývojové prostredie (IDE) s otvoreným zdrojovým kódom pre programovací jazyk R, ale aj pre ďalšie programovacie jazyky ako sú Python alebo SQL. Bolo vytvorené v roku 2011 spoločnosťou Posit.

RStudio uľahčuje prácu v jazyku R pomocou užitočných funkcií, ako sú napríklad dokončenie kódu, zvýrazňovanie syntaxu, náhľady tabuliek alebo grafov a ďalšie.

Pred spustením RStudia je dôležité mať nainštalovaný programovací jazyk R. Nasledujúci obrázok zobrazuje počítačové zobrazenie RStudia.

Obrázok 6 R studio



Zdroj: Vlastné spracovanie

RStudio sa skladá zo všeobecného menu a štyroch hlavných okien. Každé okno môže obsahovať niekoľko záložiek. Môžeme si skonfigurovať, kde bude ktoré okno, aké bude veľké a ktoré záložky budú v ktorom okne.

Hlavné časti RStudia:

- konzola (Console) – kód, ktorý tu napíšeme, sa v R hneď vyhodnotí a výsledky sa vypíšu do konzoly
- editor (Source) – poskytuje priestor na písanie R scriptu, ktorý môžeme uložiť a spúšťať opakovane, v editore môžeme mať otvorené ľubovoľné množstvo súborov rôznych typov (R script, R markdown, textové dokumenty a mnohé ďalšie)

- prehľad prostredia R (Environment) – zobrazuje všetky aktívne objekty (dáta, funkcie a pod.), ktoré žijú vo zvolenom prostredí v R (implicitne v globálnom prostredí), umožňuje tiež importovať niektoré typy dát
- súbory (Files) – zobrazuje súbory a adresáre, ktoré sú v aktuálnom projekte alebo adresári a umožňuje s nimi vykonávať základné operácie
- grafy (Plots) – zobrazuje grafy, ktoré sme v R vykreslili
- pomocník (Help) – zobrazuje dostupnú dokumentáciu k funkciám, dátam a knižniciam
- balíky (Packages) – zobrazuje zoznam dostupných a aktívnych knižníc; umožňuje tiež inštalovať nové knižnice a mazať staré knižnice
- história (History) – zobrazuje kód, ktorý ste spustili v minulosti v konzole, a zároveň umožňuje jeho uloženie, presunutie späť do konzoly alebo do editora, čím uľahčuje opätovné použitie a úpravu príkazov (Kvasnička, 2025)

3.3 Shiny aplikácie

Shiny je balík v R vyvinutý spoločnosťou RStudio v roku 2012 a patrí medzi najpopulárnejšie nástroje v tomto jazyku. V súčasnosti sa využíva v širokom spektre odvetví, podobne ako samotný jazyk R.

V akademickej sfére sa Shiny často používa ako vyučovací nástroj na vysvetľovanie štatistických metód. Vo farmaceutickom priemysle ho využívajú veľké spoločnosti na urýchlenie spolupráce medzi vedcami a analytikmi pri vývoji nových liekov. Okrem toho je Shiny široko aplikované aj v technologickom sektore, kde firmy zo Silicon Valley vytvárajú pomocou neho dynamické dashboardy pre pokročilú analytiku.

Balík Shiny umožňuje tvorbu webových aplikácií, ktoré dokážu dynamicky reagovať na používateľský vstup a zobrazovať dáta v reálnom čase. Je navrhnutý tak, aby nevyžadoval znalosť webových technológií, ako sú HTML, CSS alebo JavaScript, keďže všetky potrebné prvky sú priamo implementované v jazyku R.

Aplikácie vytvorené v Shiny sú založené na reaktívnom programovaní, čo znamená, že sú navrhnuté tak, aby automaticky reagovali na zmeny v údajoch alebo používateľských vstupoch. Pri každej zmene vstupu dochádza k okamžitej aktualizácii výstupov, čím sa

zabezpečuje plynulá interaktivita aplikácie s minimálnou odozvou. Tento prístup umožňuje efektívne spracovanie údajov a vizualizáciu v reálnom čase.

Anatómia Shiny aplikácie pozostáva z troch hlavných komponentov: užívateľského rozhrania (UI), servera a volacej funkcie, ktorá zabezpečuje prepojenie medzi rozhraním a serverovou časťou.

- Užívateľské rozhranie (UI) predstavuje grafickú časť aplikácie, ktorá určuje rozloženie a vzhľad. Obsahuje rôzne interaktívne prvky, známe ako widgety, medzi ktoré patria textové a číselné vstupy, zaškrtávacie políčka, tlačidlá, rozbaľovacie zoznamy a mnoho ďalších ovládacích prvkov. Napríklad pri testovaní hypotéz býva bežné rozvrhnutie, kde bočný panel slúži na výber parametrov testu a hlavný panel zobrazuje výsledky a vizualizácie.
- Serverová časť je zodpovedná za spracovanie vstupov z UI a na základe logiky definovanej vývojárom vytvára výstupy, ktoré sa následne zobrazujú v užívateľskom rozhraní.
- Prepojenie UI a servera sa nedeje automaticky. Na ich synchronizáciu Shiny využíva volaciu funkciu, ktorá umožňuje interakciu medzi komponentami a zabezpečuje spustenie aplikácie.

V niektorých prípadoch je možné UI a server spracovávať aj samostatne v oddelených skriptoch, pričom volacia funkcia nie je explicitne obsiahnutá v žiadnom z nich. Pri spustení aplikácie sa však automaticky aktivuje na pozadí, čím sa zabezpečí vzájomné prepojenie. Tento spôsob umožňuje lepšiu modularitu a flexibilitu pri vývoji Shiny aplikácií. (Wickham, 2024)

4 Výsledky práce

V našej práci sa zameriame na tvorbu interaktívnej webovej aplikácie pomocou jazyka R a balíku Shiny. Vytvorená aplikácia umožní používateľovi interaktívne zobrazovať a analyzovať dáta prostredníctvom grafov z oblastí matematiky, štatistiky a diskrétnej matematiky. Používateľ bude mať možnosť meniť rôzne vstupné parametre, pričom tieto zmeny sa automaticky prejavia v aktualizácii vizualizovaných údajov.

4.1 Využité nástroje a balíky jazyka R

Najskôr je potrebné nainštalovať programovací jazyk R spolu s vývojovým prostredím RStudio. Obe sú bezplatne dostupné online, pričom R je možné stiahnuť z webovej stránky https://cran.rstudio.com/ a RStudio z https://posit.co/download/rstudio.

Na vytvorenie aplikácie boli použité nasledujúce knižnice jazyka R, ktoré umožňujú vytvárať interaktívne a vizuálne používateľské rozhranie a taktiež umožňujú prezentovať, spracovávať a analyzovať údaje:

- library(shiny) – knižnica určená na tvorbu interaktívnych webových aplikácií.
- library(ggplot2) – knižnica na tvorbu pokročilých štatistických grafov.
- library(igraph)– knižnica určená na tvorbu, manipuláciu a analýzu diskrétnych matematických grafov.
- library(ggraph)– knižnica zameraná na vizualizáciu diskrétnych matematických grafov vytvorených prostredníctvom igraph.
- library(tidyr) – knižnica na usporiadanie dát, predovšetkým transformáciu dátových štruktúr medzi tzv. dlhým (long) a širokým (wide) formátom.
- library(dplyr)– knižnica, ktorá poskytuje intuitívne a efektívne nástroje na manipuláciu a transformáciu dát.

Pre správne fungovanie aplikácie je potrebné nainštalovať všetky vyššie uvedené R balíky. Inštaláciu týchto balíkov zabezpečíme pomocou funkcie *install.packages()*, pričom pre každý balík zadáme samostatný príkaz. Ukážka inštalačných príkazov vyzerá nasledovne:

Obrázok 7 Inštalácia balíkov v R

```
install.packages("shiny")
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("igraph")
install.packages("ggraph")
```

Zdroj: Vlastné spracovanie

Po úspešnej inštalácii balíkov je potrebné ich načítať do prostredia pomocou funkcie *library()*. Na obrázku nižšie je znázornený príklad načítania všetkých potrebných knižníc, ktoré sú využité pri tvorbe aplikácie:

Obrázok 8 Načítanie knižníc

```
library(shiny)
library(ggplot2)
library(dplyr)
library(tidyr)
library(igraph)
library(ggraph)
```

Zdroj: Vlastné spracovanie

4.2 Užívateľské rozhranie

Konkrétna funkcia v balíku Shiny, ktorá vytvára užívateľské rozhranie je najmä funkcia *fluidPage()*, ktorá zabezpečuje automatické prispôsobenie veľkosti okna prehliadača používateľa. Táto funkcia pozostáva z niekoľkých častí ako napríklad CSS štýly, ktoré upravujú vizuálny vzhľad aplikácie. V úvode kódu *(tags$head())* sú pomocou HTML definované základné štýly, ktoré určujú vzhľad aplikácie ako napríklad:

- farba pozadia (#f9f9f9 – svetlá sivá),
- farba hlavného nadpisu a titulkov (darkblue – tmavomodrá),
- vlastný štýl tlačidiel, ktorý kombinuje zelené pozadie s bielym písmom a jednoduchý moderný dizajn.

Obrázok 9 CSS štýly

```
ui <- fluidPage(
  tags$head(
    tags$style(HTML("
      body { background-color: #f9f9f9; }
      h1, .title { color: darkblue; }
      .custom-button {
        background-color: #4CAF50;
        color: white;
        border: none;
        padding: 10px 20px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 4px 2px;
        cursor: pointer;
      }
    "))
```

Zdroj: Vlastné spracovanie

Rozloženie aplikácie je definované prostredníctvom funkcie *sidebarLayout()*, ktorá rozdeľuje stránku na dve hlavné časti:

- *sidebarPanel()* – bočný panel obsahujúci všetky ovládacie prvky potrebné pre zadávanie vstupných parametrov.
- *mainPanel()* – hlavný panel určený na zobrazenie výsledkov, teda grafov a ďalších výstupov generovaných aplikáciou.

Pri prvotnom spustení aplikácie si používateľ z rozbaľovacieho zoznamu v bočnom paneli môže vybrať typ grafu, ktorý chce vytvoriť. Tento zoznam je vytvorený pomocou funkcie *selectInput()*.

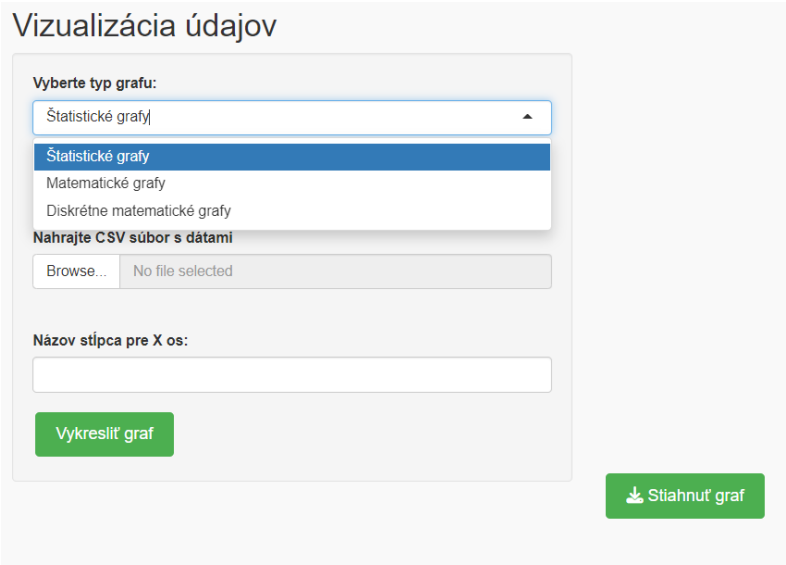
Obrázok 10 Ukážka kódu pre výber typu grafu

```
      selectInput("graphType", "Vyberte typ grafu:",
                  choices = c("Štatistické grafy" = "stat",
                              "Matematické grafy" = "math",
                              "Diskrétne matematické grafy" = "discrete")),
```

Zdroj: Vlastné spracovanie

V aplikácii táto časť kódu zabezpečuje grafické zobrazenie užívateľského rozhrania a vyzerá nasledovne:

Obrázok 11Výber typu grafu



Zdroj: Vlastné spracovanie

Po výbere typu grafu z rozbaľovacieho zoznamu používateľ zadáva ďalšie parametre, ktoré ovplyvnia výsledné zobrazenie grafu v hlavnom paneli. Na základe výberu sa prostredníctvom *conditionalPanel()* dynamicky zobrazujú ďalšie ovládacie prvky.

Pri výbere štatistických grafov používateľ môže zvoliť konkrétny typ štatistického grafu pomocou *selectInput()*, ktorý ponúka možnosti ako histogram, boxplot, stĺpcový a koláčový graf. Následne sa zobrazí možnosť nahrať CSV súbor s dátami cez *fileInput()*.

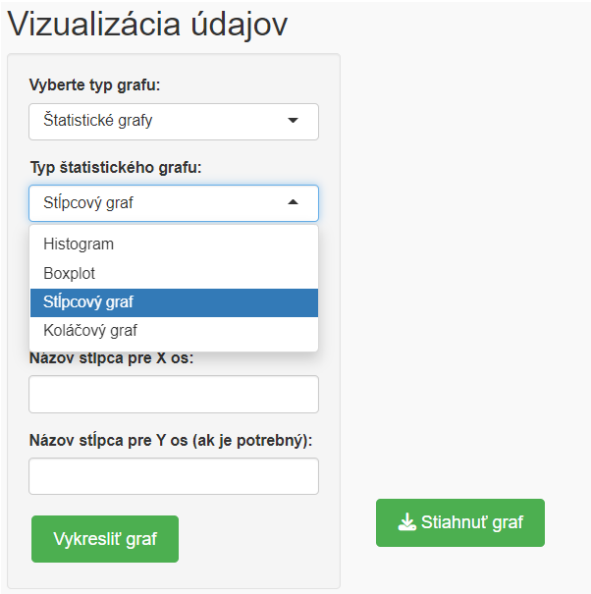
Obrázok 12 Ukážka kódu pre výber štatistických grafov

```
43      conditionalPanel(
44        condition = "input.graphType == 'stat'",
45        selectInput("statType", "Typ štatistického grafu:",
46                    choices = c("Histogram" = "histogram",
47                                "Boxplot" = "boxplot",
48                                "Stĺpcový graf" = "barplot",
49                                "Koláčový graf" = "piechart")),
50        fileInput("fileInput", "Nahrajte CSV súbor s dátami", accept = ".csv"),
```

Zdroj: Vlastné spracovanie

V aplikácii výber štatistických grafov vyzerá nasledovne:

Obrázok 13 Výber štatistických grafov



Zdroj: Vlastné spracovanie

Pri výbere „Boxplot“ sa zobrazí ovládací prvok vytvorený cez *uiOutput("boxplotColSelector")*. Tento prvok umožňuje používateľovi vybrať konkrétne stĺpce z načítaného datasetu, ktoré budú následne použité pri tvorbe boxplotu.

Obrázok 14 Ukážka kódu pri výbere stĺpcov Boxplot

```
51        # Dynamický výber stĺpcov (iba pre boxplot)
52        conditionalPanel(
53          condition = "input.statType == 'boxplot'",
54          uiOutput("boxplotColSelector")
55        ),
```

Zdroj: Vlastné spracovanie

Po výbere typu štatistického grafu sa vždy zobrazí textové pole pre zadanie názvu stĺpca pre os *x,* pomocou funkcie *textInput()*, ktorý určuje údaje zobrazené na horizontálnej osi. Vstup pre os *y* sa následne zobrazí iba vtedy, ak nie je zvolený histogram – pretože histogram automaticky generuje frekvencie.

Obrázok 15 Ukážka kódu pre zadávanie hodnôt štatistických grafov

```
56        # Vstup pre X os sa zobrazí vždy
57        textInput("xCol", "Názov stĺpca pre X os:", value = ""),
58        # Vstup pre Y os sa zobrazí len ak nie je histogram
59        conditionalPanel(
60          condition = "!(input.statType == 'histogram')",
61          textInput("yCol", "Názov stĺpca pre Y os (ak je potrebný):", value = "")
62        )
```

Zdroj: Vlastné spracovanie

Ak používateľ zvolí možnosť „Matematické grafy“, bočný panel sa dynamicky upraví a zobrazí vstupné prvky pre zadanie matematickej funkcie, nastavenie rozsahu hodnôt (minimálna a maximálna hodnota) a určenie kroku pre os *x*.

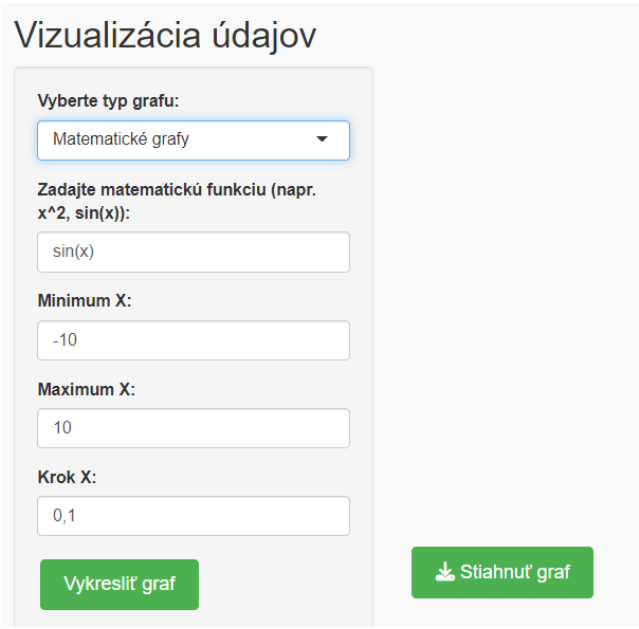
Obrázok 16 Ukážka kódu pre výber matematického grafu

```
65      conditionalPanel(
66        condition = "input.graphType == 'math'",
67        textInput("mathFunc", "Zadajte matematickú funkciu (napr. x^2, sin(x)):", value = "sin(x)"),
68        numericInput("xMin", "Minimum X:", value = -10),
69        numericInput("xMax", "Maximum X:", value = 10),
70        numericInput("xStep", "Krok X:", value = 0.1)
71      ),
```

Zdroj: Vlastné spracovanie

Pri výbere matematického grafu sa v aplikácii zobrazí nasledujúce rozhranie:

Obrázok 17 Výber matematický graf



Zdroj: Vlastné spracovanie

Pri zvolení diskrétneho matematického grafu sa v bočnom paneli dynamicky zobrazia dva vstupné prvky. Prvý z nich, vytvorený pomocou *textInput()*, umožňuje zadanie hrán grafu vo formáte, napríklad „1-2,2-3,3-1“, čím sa definujú prepojenia medzi uzlami. Druhý vstup, zabezpečený cez *selectInput()*, poskytuje možnosť výberu typu grafu – používateľ si môže zvoliť buď neorientovaný graf, kde hrany nie sú vybavené smerovými šípkami, alebo orientovaný graf, v ktorom hrany obsahujú šípky indikujúce smer prepojení.

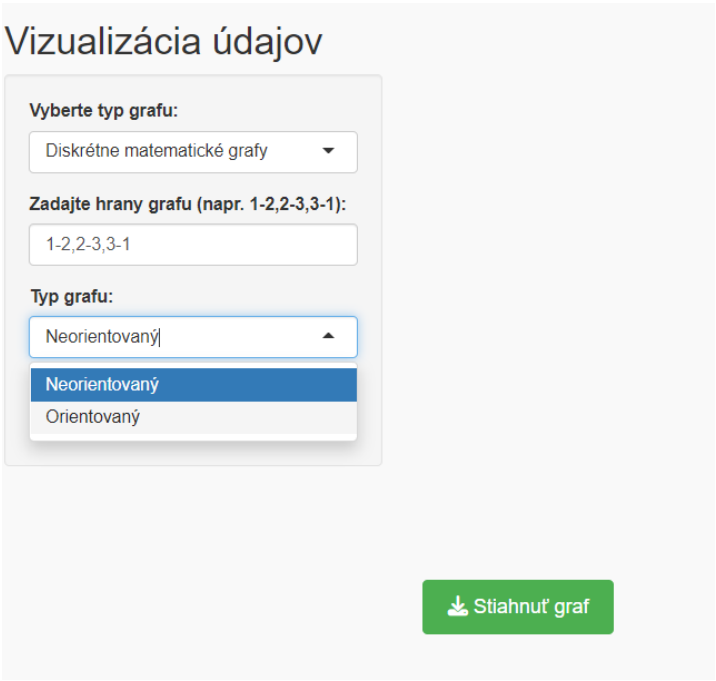
Obrázok 18 Ukážka kódu pre diskrétny matematický graf

```
73        conditionalPanel(
74          condition = "input.graphType == 'discrete'",
75          textInput("edges", "Zadajte hrany grafu (napr. 1-2,2-3,3-1):", value = "1-2,2-3,3-1"),
76          selectInput("discreteType", "Typ grafu:",
77                      choices = c("Neorientovaný" = "undirected",
78                                  "Orientovaný" = "directed"))
79        ),
```

Zdroj: Vlastné spracovanie

V aplikácii sa pri výbere grafu diskrétnej matematiky zobrazí nasledujúce rozhranie, cez ktoré môže používateľ upravovať parametre grafu.

Obrázok 19 Výber diskrétny graf



Zdroj: Vlastné spracovanie

Výsledné grafy sú zobrazené v hlavnom paneli pomocou funkcie *plotOutput()*. Používateľ môže výsledný graf vykresliť tlačidlom „Vykresliť graf“ (*actionButton()*) a následne ho stiahnuť pomocou tlačidla *downloadButton()*.

Obrázok 19 Vykreslenie a stiahnutie grafu

```
81          actionButton("generate", "Vykresliť graf", class = "custom-button")
82        ),
83        mainPanel(
84          plotOutput("plotOutput"),
85          downloadButton("downloadPlot", "Stiahnuť graf", class = "custom-button")
86        )
```

Zdroj: Vlastné spracovanie

4.3 Serverová časť

Táto časť kódu predstavuje serverovú logiku aplikácie, ktorá zabezpečuje načítanie údajov zo súboru, ktorý používateľ vloží cez grafické rozhranie.

Obrázok 20 Načítanie údajov

```
server <- function(input, output, session) {

  data_reactive <- reactive({
    req(input$fileInput)
    df <- read.csv(input$fileInput$datapath, sep = ";", dec = ",",
                   stringsAsFactors = FALSE, fileEncoding = "UTF-8")
    names(df) <- make.names(names(df))
    df
  })
```

Zdroj: Vlastné spracovanie

Po vložení CSV súboru aplikácia automaticky reaguje a spustí proces načítania dát. Používa sa funkcia *read.csv()*, ktorej parametre sú prispôsobené slovenskému formátu CSV súborov – ako oddeľovač stĺpcov je použitá bodkočiarka (sep = ";") a desatinné čísla sú označené čiarkou (dec = ",").

Po načítaní sú názvy stĺpcov upravené do štandardného formátu pomocou funkcie *make.names(),* ktorá odstráni prípadné medzery a špeciálne znaky, čím zaručí bezproblémovú manipuláciu s údajmi v ďalšom kroku. Výsledný upravený dátový rámec (df) je následne dostupný pre ďalšie spracovanie a vizualizáciu v aplikácii.

Obrázok 21 Reaktívne vykreslenie grafu

```
  myPlot <- reactive({
    input$generate
    isolate({
      if (input$graphType == "stat") {
        df <- data_reactive()
```

Zdroj: Vlastné spracovanie

Tento kód na obrázku 21 predstavuje reaktívnu funkciu *myPlot()*, ktorá sa spustí až po kliknutí používateľa na tlačidlo „Vykresliť graf“ (*input$generate*). Funkcia najprv izoluje reaktivitu, čím zaručuje, že sa výpočet spustí len explicitne po stlačení tlačidla. Ak si používateľ zvolí typ grafu ako „Štatistické grafy“, kód pristúpi k dátam načítaným predchádzajúcou funkciou (*data_reactive()*) a uloží ich do dátového rámca df. Tento rámec je potom pripravený na ďalšie spracovanie podľa konkrétneho typu štatistického grafu, ktorý používateľ zvolil (napríklad histogram, boxplot, stĺpcový graf alebo koláčový graf).

4.3.1 Histogram

V tejto časti sa zameriame na vizualizáciu histogramu, ktorý zobrazuje dáta rozdelené do intervalov s jasnými označeniami a frekvenciami výskytu v jednotlivých intervaloch.

Obrázok 22 Vytváranie frekvencie pomocou funkcie *pretty()*

```
116       if (input$statType == "histogram") {
117         req(input$xCol)
118         x <- as.numeric(as.character(df[[input$xCol]]))
119         req(!all(is.na(x)))
120
121         breaks_vec <- pretty(x, n = 8)
122         midpoints <- (head(breaks_vec, -1) + tail(breaks_vec, -1)) /
123         labels <- paste0(head(breaks_vec, -1), "-", tail(breaks_vec, -1))
124
```

Zdroj: Vlastné spracovanie

Najskôr kód overuje existenciu a platnosť vstupného stĺpca zadaného používateľom (*input$xCol*). Dáta z tohto stĺpca sú následne konvertované na numerické hodnoty, pričom ak sú všetky hodnoty neplatné (napríklad textové údaje), ďalšie spracovanie sa zastaví. Potom pomocou funkcie *pretty()* vytvára vhodné intervaly, ktoré sú ľahko interpretovateľné. Stredné hodnoty týchto intervalov sú použité na označenie intervalov na osi *x* (napríklad "10-20", "20-30").

Samotný histogram je vykreslený funkciou *geom_histogram()*, ktorá využíva tieto intervaly. Histogram je ďalej doplnený textovými popismi (*geom_text()*), ktoré udávajú početnosť (frekvenciu) údajov v každom intervale. Osi grafu sú upravené pomocou *scale_x_continuous()*, aby zodpovedali vytvoreným intervalom.

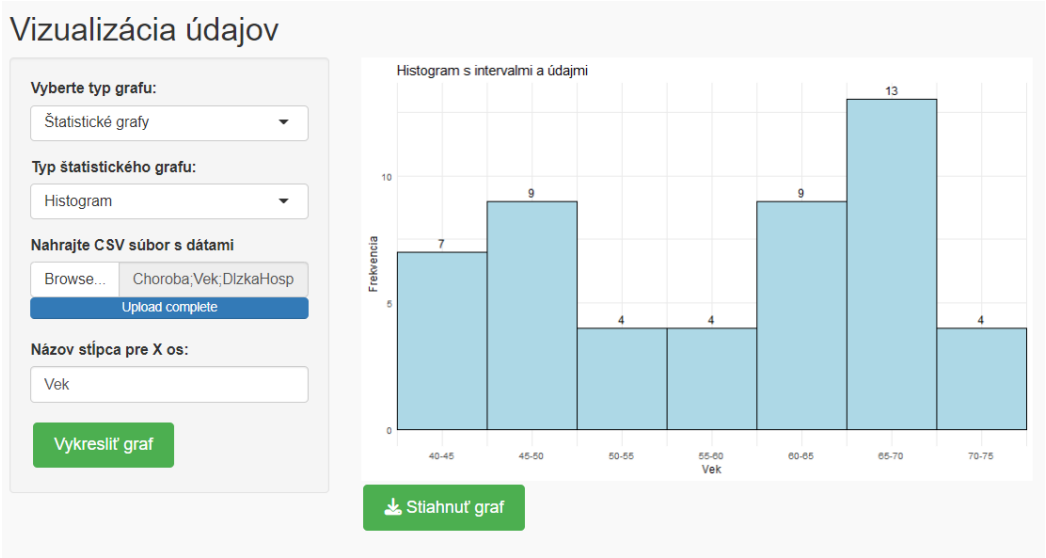
Obrázok 23 Vykreslenie histogramu pomocou geom_histogram()

```
125         p <- ggplot(df, aes(x = x)) +
126           geom_histogram(
127             breaks = breaks_vec,
128             boundary = min(breaks_vec),
129             fill = "lightblue",
130             color = "black"
131           ) +
132           geom_text(stat = "bin", breaks = breaks_vec, aes(label = after_stat(count)),
133                     vjust = -0.5, color = "black") +
134           scale_x_continuous(
135             breaks = midpoints,
136             labels = labels,
137             expand = c(0, 0)
138           ) +
139           labs(
140             title = "Histogram s intervalmi a údajmi",
141             x = input$xCol,
142             y = "Frekvencia"
143           ) +
144           theme_minimal()
145
146         return(p)
```

Zdroj: Vlastné spracovanie

Na demonštráciu sme využili vstupné údaje zo súboru CSV, ktorý obsahuje stĺpce Choroba, Vek a Dĺžka hospitalizácie. Pre vytvorenie histogramu sme sa zamerali na vizualizáciu údajov pre premennú vek. Na osi *x* sú zobrazené vekové intervaly pacientov, zatiaľ čo na osi *y* je vyjadrený počet výskytov (frekvencia) jednotlivých intervalov. Tento histogram poskytuje prehľad o vekovom rozložení hospitalizovaných osôb a umožňuje jednoduchú analýzu vekovej štruktúry v rámci dostupných údajov. Údaje pochádzajú z Národného centra zdravotníckych informácií.

Obrázok 24 Histogram Hospitalizácia (vek)



Zdroj: Vlastné spracovanie

4.3.2 Boxplot

Boxplot slúži na prehľadnú vizualizáciu a je veľmi užitočný na identifikáciu odľahlých hodnôt, rozsahu údajov a na zistenie základných štatistických charakteristík, ako sú kvartily a medián.

Najskôr sa overí, či používateľ v aplikácii označil aspoň jeden numerický stĺpec, ktorý chce analyzovať. Ak žiadny stĺpec nie je vybraný, graf sa nevytvorí a používateľ dostane vizuálne upozornenie s informáciou „Nie sú vybrané žiadne stĺpce.“

Obrázok 25 Výber stĺpcov pri Boxplot

```
149       } else if (input$statType == "boxplot") {
150         req(input$selectedNumericCols)
151         selected_cols <- input$selectedNumericCols
152         if (length(selected_cols) == 0) {
153           return(ggplot() + labs(title = "Nie sú vybrané žiadne stĺpce."))
154         }
```

Zdroj: Vlastné spracovanie

V prípade, že používateľ označil požadované numerické stĺpce, dôjde k transformácii pôvodných dát do vhodnej formy na vizualizáciu. Táto transformácia je zabezpečená pomocou funkcie *pivot_longer()*, ktorá údaje prevedie z tzv. „širokého formátu“, kde má každý stĺpec vlastnú premennú, do „dlhého formátu“. Takáto forma dát umožňuje vykreslenie viacerých boxplotov naraz v jednom spoločnom grafickom výstupe.

Obrázok 26 Transformácia dát

```
155          df_long <- df %>%
156            select(all_of(selected_cols)) %>%
157            pivot_longer(cols = everything(),
158                         names_to = "variable",
159                         values_to = "value")
160          data_summary <- df_long %>%
```

Zdroj: Vlastné spracovanie

Po transformácii dát nasleduje výpočet kľúčových štatistických charakteristík pre každý zo zvolených stĺpcov. Tieto štatistiky zahŕňajú:

- Minimum (najnižšia hodnota)
- Q1 (prvý kvartil – 25 % údajov je nižších ako táto hodnota)
- Medián (stredná hodnota – 50 % údajov je vyšších a 50 % nižších ako táto hodnota)
- Priemer (aritmetický priemer všetkých hodnôt)
- Q3 (tretí kvartil – 75 % údajov je nižších ako táto hodnota)
- Maximum (najvyššia hodnota)

Tieto údaje sú prehľadne uložené v dátovom rámci data_summary.

Obrázok 27 Výpočet štatistických charakteristík

```
162          summarise(
163            min_val    = min(value, na.rm = TRUE),
164            q1_val     = quantile(value, 0.25, na.rm = TRUE),
165            median_val = median(value, na.rm = TRUE),
166            mean_val   = mean(value, na.rm = TRUE),
167            q3_val     = quantile(value, 0.75, na.rm = TRUE),
168            max_val    = max(value, na.rm = TRUE)
169          )
```

Zdroj: Vlastné spracovanie

Samotné boxploty sú vytvorené pomocou funkcie *geom_boxplot()* z knižnice ggplot2, pričom každý boxplot zobrazuje rozpätie medzi prvým a tretím kvartilom (medzikvartilové rozpätie), medián a prípadne odľahlé hodnoty, ak sa také v údajoch

nachádzajú. Priemerná hodnota je v každom boxplote zvýraznená pomocou funkcie *stat_summary()* ako malý biely bod.

Pri zobrazení viacerých premenných sa pre prehľadnosť jednotlivé boxploty zobrazia v samostatných paneloch vedľa seba prostredníctvom funkcie *facet_wrap()*. Každý panel reprezentuje jeden vybraný stĺpec dát, pričom rozsah osi *y* je prispôsobený individuálne pre každý panel zvlášť, čo zabezpečuje dobrú čitateľnosť aj pri veľmi rozdielnych škálach údajov.

Obrázok 28 Vykreslenie Boxplotu

```
p <- ggplot(df_long, aes(x = "", y = value)) +
  geom_boxplot(fill = "lightgreen") +
  stat_summary(fun = mean, geom = "point", shape = 23,
               size = 3, fill = "white", color = "black") +
  facet_wrap(~ variable, scales = "free_y") +
  labs(title = "Boxplot pre vybrané stĺpce", x = "", y = "Hodnota") +
  theme_minimal() +
```

Zdroj: Vlastné spracovanie

Pre vyššiu informatívnosť a lepšiu orientáciu sú do grafu pridané ešte aj textové označenia štatistických charakteristík priamo vedľa jednotlivých boxplotov. Tieto popisy sú farebne rozlíšené nasledovne:

- Modrou farbou sú označené hodnoty minima, Q1, Q3 a maxima.
- Červenou farbou je označená hodnota mediánu.
- Tmavozelenou farbou je označená hodnota priemeru.

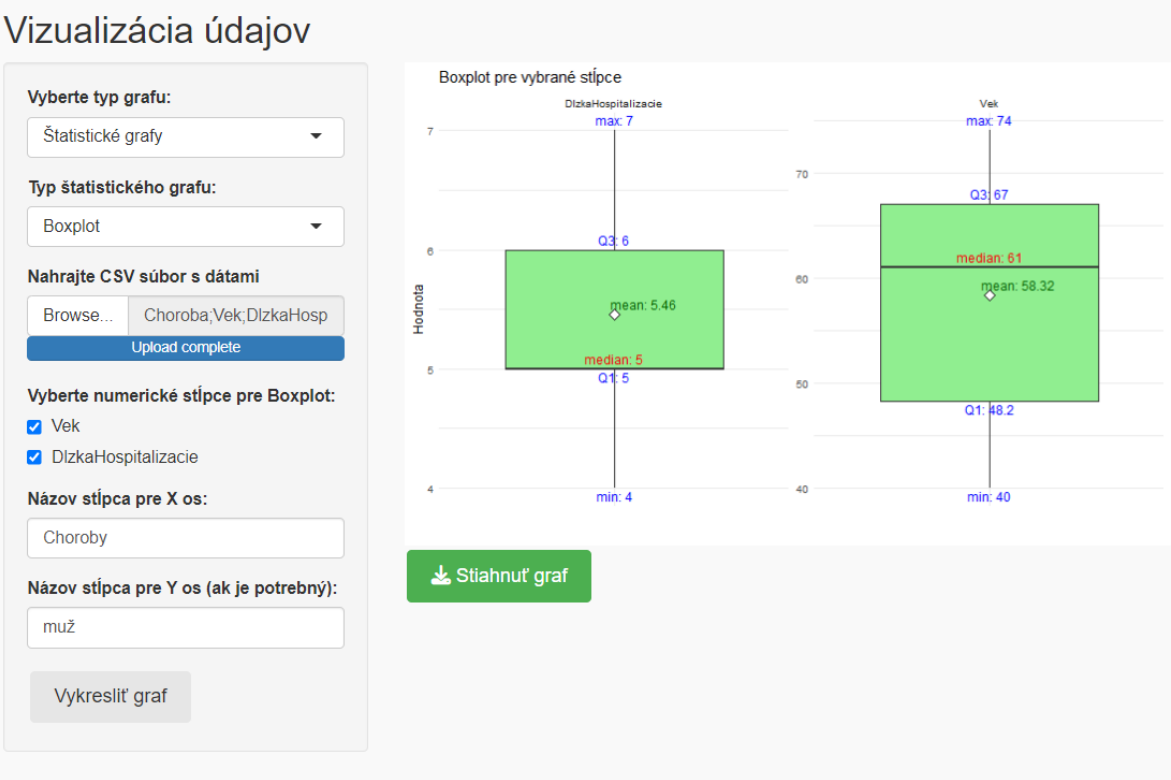
Obrázok 29 Popis štatistických charakteristík boxplotu

```
  geom_text(data = data_summary, aes(x = "", y = min_val,
                                     label = paste0("min: ", round(min_val, 1))),
            vjust = 1.2, color = "blue") +
  geom_text(data = data_summary, aes(x = "", y = q1_val,
                                     label = paste0("Q1: ", round(q1_val, 1))),
            vjust = 1.2, color = "blue") +
  geom_text(data = data_summary, aes(x = "", y = median_val,
                                     label = paste0("median: ", round(median_val, 1))),
            vjust = -0.5, color = "red") +
  geom_text(data = data_summary, aes(x = "", y = mean_val,
                                     label = paste0("mean: ", round(mean_val, 2))),
            vjust = -0.5, nudge_x = 0.1, color = "darkgreen") +
  geom_text(data = data_summary, aes(x = "", y = q3_val,
                                     label = paste0("Q3: ", round(q3_val, 1))),
            vjust = -0.5, color = "blue") +
  geom_text(data = data_summary, aes(x = "", y = max_val,
                                     label = paste0("max: ", round(max_val, 1))),
            vjust = -0.5, color = "blue")
return(p)
```

Zdroj: Vlastné spracovanie

Ako príklad vizualizácie údajov pomocou boxplotu sme použili rovnaký CSV súbor ako pri predchádzajúcom grafe. Boxplot v tomto prípade zobrazuje rozloženie veku pacientov a dĺžky ich hospitalizácie v závislosti od typu ochorenia. Z grafu možno vyčítať základné štatistické charakteristiky, ako sú medián, kvartily či prípadné odľahlé hodnoty, vďaka čomu získavame prehľad o variabilite údajov. Na základe analyzovaných údajov je priemerný vek pacientov hospitalizovaných s vybranými ochoreniami 58,32 roka, pričom medián veku dosahuje 61 rokov. Priemerná dĺžka hospitalizácie je 5,46 dňa s mediánom 5 dní.

Obrázok 30 Boxplot- Hospitalizácia



Zdroj: Vlastné spracovanie

4.3.3 Stĺpcový graf

Stĺpcový graf a umožňuje používateľovi flexibilne vizualizovať dve rôzne situácie v závislosti od typu vstupných údajov:

1. Numerické údaje:

   Ak používateľ zadá dva numerické stĺpce, napríklad „cena“ a „vek“, kód najskôr pretransformuje dáta zo širokého formátu do dlhého formátu pomocou funkcie *pivot_longer()*.

Po transformácii sa pre každý zo stĺpcov vypočíta priemerná hodnota. Tieto priemery sa následne zobrazia v stĺpcovom grafe, kde každý stĺpec reprezentuje jednu premennú („cena“, „vek“ a pod.). Stĺpce sú farebne odlíšené a nad každým stĺpcom je zobrazená konkrétna priemerná hodnota zaokrúhlená na jedno desatinné miesto.

Graf obsahuje popisné názvy osí a jednoduchý prehľadný dizajn zabezpečený použitím *theme_minimal()*.

Obrázok 31 Vykresľovanie numerických hodnôt stĺpcového grafu

```
197   } else if (input$statType == "barplot") {
198     req(input$xCol, input$yCol)
199     df <- data_reactive()
200     df[[input$yCol]] <- as.numeric(as.character(df[[input$yCol]]))
201
202     if (is.numeric(df[[input$xCol]]) && is.numeric(df[[input$yCol]])) {
203       df_long <- df %>%
204         pivot_longer(
205           cols = c(input$xCol, input$yCol),
206           names_to = "variable",
207           values_to = "value"
208         )
209       df_long$value <- as.numeric(as.character(df_long$value))
210       agg_df <- df_long %>%
211         group_by(variable) %>%
212         summarise(avg = mean(value, na.rm = TRUE), .groups = "drop")
213
214       p <- ggplot(agg_df, aes(x = variable, y = avg, fill = variable)) +
215         geom_col(width = 0.6, color = "black") +
216         geom_text(aes(label = round(avg, 1)), vjust = -0.5, size = 5) +
217         labs(title = "Priemerné hodnoty",
218              x = "Premenná",
219              y = "Priemerná hodnota") +
220         theme_minimal()
221
222       return(p)
```

Zdroj: Vlastné spracovanie

2. Kategoriálne údaje

Ak aspoň jeden zo zadaných stĺpcov nie je numerický (napríklad jeden stĺpec obsahuje kategórie ako „pes“, „mačka“), potom aplikácia považuje stĺpec zadaný v *xCol* za kategóriu. Tieto kategórie sú prevedené na faktor, čo umožňuje ich správne zobrazenie v grafe.

Následne sa vypočíta priemerná hodnota numerickej premennej (stĺpec zadaný ako *yCol*) pre každú z týchto kategórií. Vznikne tak graf, v ktorom na osi *x* sú kategórie (napríklad zvieratá), a na osi *y* sú príslušné priemerné hodnoty (napríklad priemerný počet bodov, počet výskytov, hodnoty testov a podobne). Každý stĺpec má nad sebou popis s presnou hodnotou priemeru, čo uľahčuje čítanie údajov.

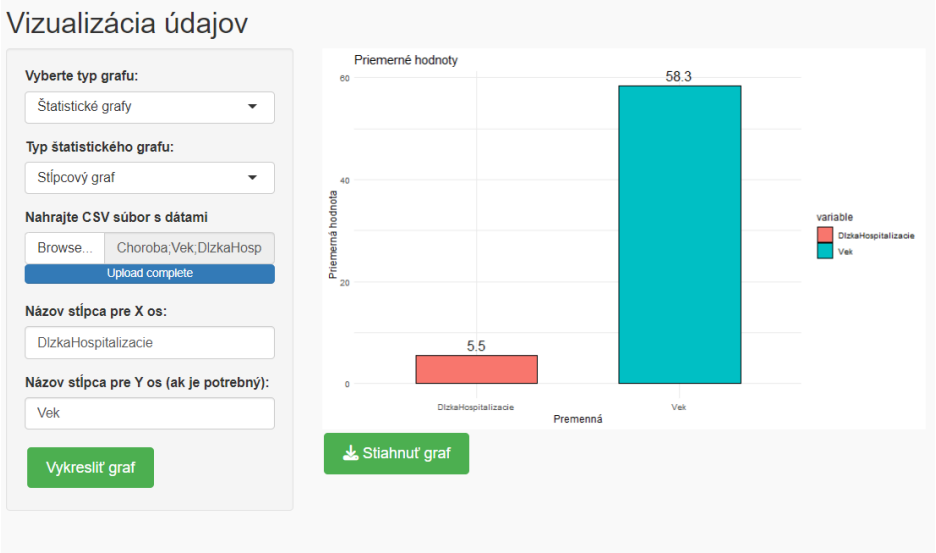
Obrázok 32 Vykresľovanie kategoriálnych údajov stĺpcového grafu

```
        } else {
          df[[input$xCol]] <- factor(df[[input$xCol]])
          agg_df <- df %>%
            group_by(.data[[input$xCol]]) %>%
            summarise(agg_value = mean(.data[[input$yCol]], na.rm = TRUE), .groups = "drop")

          p <- ggplot(agg_df, aes(x = .data[[input$xCol]], y = agg_value, fill = .data[[input$xCol]])) +
            geom_col(width = 0.6, color = "black") +
            geom_text(aes(label = round(agg_value, 1)), vjust = -0.5, size = 5) +
            labs(title = paste("Stĺpcový graf:", input$xCol, "vs. priemer", input$yCol),
                 x = input$xCol, y = "Priemer") +
            theme_minimal()

          return(p)
        }
```

Zdroj: Vlastné spracovanie

Na základe údajov z CSV súboru sme vytvorili stĺpcový graf, ktorý porovnáva priemerný vek hospitalizovaných pacientov a priemernú dĺžku ich hospitalizácie. Os *x* reprezentuje dve vybrané premenné – vek a dĺžka hospitalizácie, zatiaľ čo os *y* znázorňuje ich priemerné hodnoty. Ako vidíme z grafu, priemerný vek pacientov je približne 58,3 roka, zatiaľ čo priemerná dĺžka hospitalizácie dosahuje hodnotu 5,5 dňa. Údaje použité na tvorbu grafu pochádzajú z webovej stránky Národného centra zdravotníckych informácií.

Obrázok 33 Stĺpcový graf- Priemerné hodnoty



Zdroj: Vlastné spracovanie

Ako príklad pre zobrazenie kategoriálnych údajov sme použili CSV súbor obsahujúci dva stĺpce – „Choroby“ a „muž“, kde sú uvedené jednotlivé kategórie ochorení a počet hospitalizovaných mužov v rámci každej kategórie. Dataset bol spracovaný na základe údajov zo Štatistického výstupu Národného centra zdravotníckych informácií .

Obrázok 34Stĺpcový graf- Hospitalizácia

Zdroj: Vlastné spracovanie

4.3.4 Koláčový graf

Táto časť sa zaoberá generovaním koláčového grafu pomocou knižnice ggplot2 na základe dát, ktoré používateľ nahrá do aplikácie vo formáte CSV. Koláčový graf patrí medzi veľmi populárne spôsoby vizualizácie dát, pretože jasne a jednoducho ukazuje proporcie jednotlivých kategórií v rámci celku.

Obrázok 35 Ukážka kódu na načítanie dát

```
240       } else if (input$statType == "piechart") {
241         req(input$xCol, input$yCol)
242         df <- data_reactive()  # Načítame dáta
243         aggregated_data <- aggregate(df[[input$yCol]],
244                                      by = list(Category = df[[input$xCol]]),
245                                      FUN = sum)
246         names(aggregated_data)[2] <- input$yCol
247         aggregated_data$perc <- aggregated_data[[input$yCol]] / sum(aggregated_data[[input$yCol]]) * 100
```

Zdroj: Vlastné spracovanie

Najprv sa pomocou funkcie *req()* zabezpečí, že oba vstupné parametre (*input$xCol* a *input$yCol*) sú zadané používateľom. Tieto parametre určujú, ktoré stĺpce z nahraných dát sa použijú na tvorbu grafu. Konkrétne:

- *input$xCol* predstavuje stĺpec obsahujúci názvy jednotlivých kategórií, teda identifikátor každého segmentu koláča (napríklad kategórie produktov alebo druhy zvierat).
- *input$yCol* predstavuje stĺpec obsahujúci numerické hodnoty, ktoré určujú veľkosť jednotlivých segmentov v grafe (napríklad počty, množstvá alebo hodnoty daných kategórií).

Následne sa načítajú samotné dáta do dátového rámca df. Potom sa dáta agregujú pomocou funkcie *aggregate()* a následne vypočítame percentuálny podiel každej kategórie. Tento výpočet prebieha tak, že hodnotu každej kategórie vydelíme celkovou sumou všetkých hodnôt a výsledok vynásobíme číslom 100.

Obrázok 36 Ukážka kódu na vykreslenie koláčového grafu

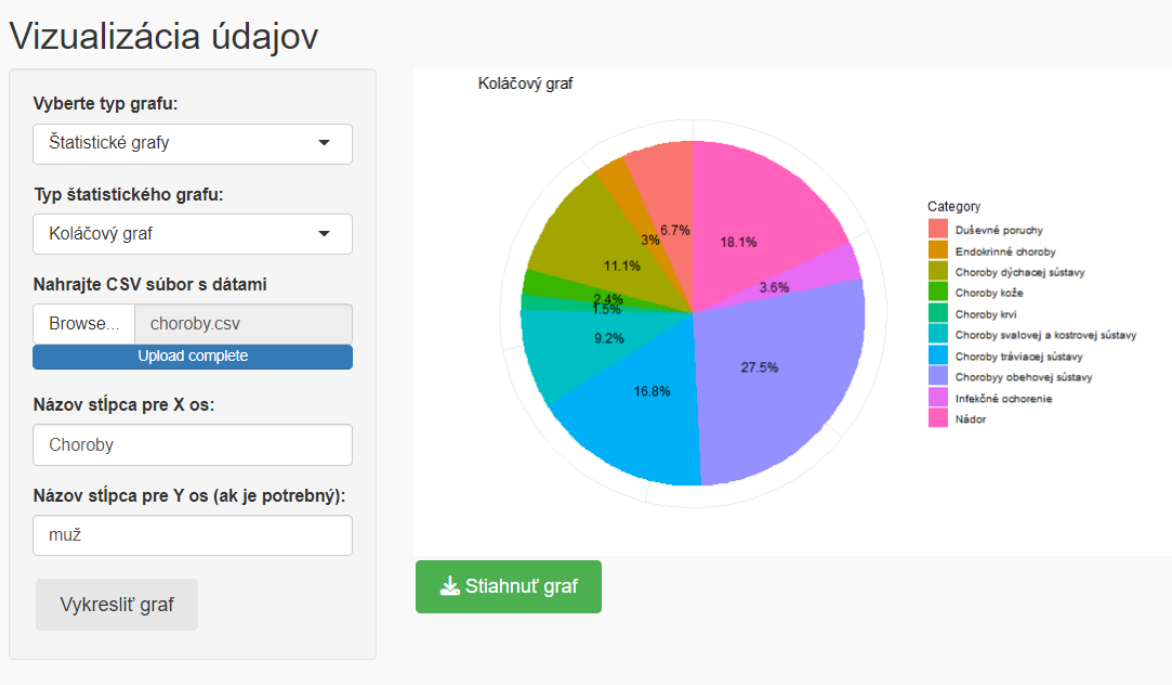
```
249           p <- ggplot(aggregated_data, aes(x = "", y = .data[[input$yCol]], fill = Category)) +
250             geom_bar(width = 1, stat = "identity") +
251             coord_polar("y", start = 0) +
252             geom_text(aes(label = paste0(round(perc, 1), "%")),
253                       position = position_stack(vjust = 0.5)) +
254             labs(title = "Koláčový graf", x = "", y = "") +
255             theme_minimal() +
256             theme(axis.text.x = element_blank())
257           return(p)
258         }
```

Zdroj: Vlastné spracovanie

Na vizualizáciu dát slúži knižnica ggplot2. Graf je vytvorený pomocou funkcie *geom_bar()*, pričom parameter stat = "identity" určuje, že sa použijú už vypočítané hodnoty namiesto počítania frekvencií. Aby sa dosiahol kruhový vzhľad grafu, použije sa funkcia *coord_polar("y", start = 0)*. Segmenty sú farebne odlíšené podľa kategórií.

Ako príklad uvádzame koláčový graf, ktorý znázorňuje rozdelenie počtu hospitalizácií mužov na Slovensku v roku 2022 podľa vybraných skupín diagnóz. Graf bol vytvorený na základe údajov zo Štatistického výstupu Národného centra zdravotníckych informácií a poskytuje prehľad o podiele jednotlivých kategórií ochorení, ako sú nádory, obehová sústava, tráviaca sústava či duševné poruchy. Vizualizácia umožňuje jednoduché porovnanie frekvencie výskytu jednotlivých typov diagnóz.

Obrázok 37 Koláčový graf- hospitalizácia



Zdroj: Vlastné spracovanie

4.3.5 Matematický graf

V tejto časti predstavíme implementáciu matematických grafov, ktoré umožňujú používateľovi vizualizovať priebeh matematických funkcií zadaných do aplikácie. Používateľ zadá matematickú funkciu, na základe ktorej kód vypočíta príslušné hodnoty a dynamicky vygeneruje graf, ktorý jasne zobrazuje priebeh funkcie v definovanom intervale.

Obrázok 38 Ukážka kódu pri matematických grafov

```
276     }else if (input$graphType == "math") {
277       req(input$mathFunc)
278       func <- function(x) eval(parse(text = input$mathFunc))
279       x_vals <- seq(input$xMin, input$xMax, by = input$xStep)
280       y_vals <- sapply(x_vals, func)
281
282       p <- ggplot(data.frame(x = x_vals, y = y_vals), aes(x = x, y = y)) +
283         geom_line(color = "blue") +
284         # Zvýraznenie súradnicovej osi 0,0
285         geom_vline(xintercept = 0, color = "black", size = 0.5) +
286         geom_hline(yintercept = 0, color = "black", size = 0.5) +
287         labs(title = paste("Matematický graf: y =", input$mathFunc), x = "X", y = "Y") +
288         theme_minimal()
289       return(p)
```

Zdroj: Vlastné spracovanie

Na začiatku kód kontroluje, či používateľ zadal matematickú funkciu pomocou *req(input$mathFunc)*. Táto kontrola zaručuje, že ďalší výpočet pokračuje iba vtedy, keď je zadaná potrebná vstupná hodnota. Potom sa vytvorí funkcia pomocou kombinácie funkcií *eval(parse(text = input$mathFunc))*. To umožní dynamicky interpretovať matematickú

funkciu zadanú používateľom ako textový reťazec. Napríklad používateľ môže zadať výraz ako „sin(x)“ alebo „x^2“.

Na generovanie bodov pre vykreslenie grafu sa použije funkcia *seq()*, ktorá vytvorí postupnosť hodnôt pre os *x*. Rozsah a hustotu tejto postupnosti zadáva používateľ prostredníctvom parametrov:

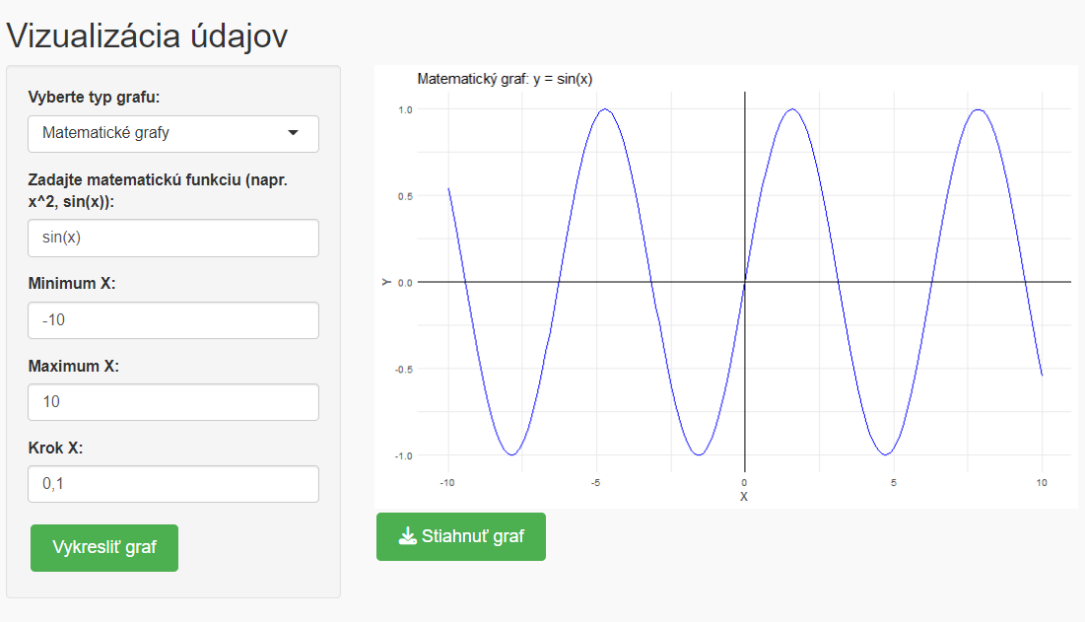
- input$xMin (minimum pre os *x*)
- input$xMax (maximum pre os *x*)
- input$xStep (krok medzi jednotlivými hodnotami)

Na základe vytvorenej postupnosti hodnôt osi *x* (*x_vals*) sú následne vypočítané príslušné hodnoty osi *y* (*y_vals*) aplikáciou používateľom definovanej matematickej funkcie na každú hodnotu z *x_vals*. Tieto výsledky sú potom spojené do dátového rámca, ktorý je vstupom pre funkciu *ggplot()*. Graf je vykreslený ako spojnicový graf (*geom_line()*).

Pomocou *labs()* sa k výslednému grafu pridávajú opisy osí a názov grafu, ktorý obsahuje explicitný zápis použitej matematickej funkcie, čo výrazne uľahčuje porozumenie výslednej vizualizácie.

Ako príklad pre zobrazenie matematickej funkcie sme zvolili graf funkcie *sin(x)*. V tomto prípade sú vstupné hodnoty pre *x* generované v definovanom intervale (od -10 do 10). Graf zobrazuje periodické oscilácie funkcie, pričom os *x* predstavuje hodnoty vstupnej premennej a os *y* udáva príslušné hodnoty funkcie *sin(x)*.

Obrázok 39 Funkcia *sin(x)*



Zdroj: Vlastné spracovanie

4.3.6 Diskrétny graf

V tejto časti aplikácie sa zameriavame na generovanie diskrétnych grafov. Používateľ má možnosť vizualizovať neorientované alebo orientované grafy pričom vstupom sú údaje o hranách, ktoré zadáva prostredníctvom aplikácie. Takto vytvorené grafy umožňujú prehľadnú analýzu vzťahov medzi uzlami, čo je užitočné najmä pri štúdiu sieťových štruktúr a komplexných prepojení medzi prvkami.

Obrázok 40 Ukážka kódu Diskrétny graf

```
290     } else if (input$graphType == "discrete") {
291       req(input$edges, input$discreteType)
292       edges <- strsplit(input$edges, ",")[[1]]
293       edges <- do.call(rbind, strsplit(edges, "-"))
```

Zdroj: Vlastné spracovanie

Na začiatku sa kontroluje, či používateľ zadal potrebné údaje pomocou funkcie *req(input$edges, input$discreteType)*. Hrany grafu zadáva používateľ ako textový reťazec v tvare napríklad „1-2,2-3,3-1“. Tento reťazec sa najprv rozdelí pomocou funkcie *strsplit()* na jednotlivé hrany podľa čiarky. Následne sú jednotlivé hrany rozdelené podľa pomlčky, čím vzniknú dvojice vrcholov, ktoré sú následne použité ako zoznam hrán pre vytvorenie grafu.

Graf je vytvorený pomocou funkcie *graph_from_edgelist()* z balíka igraph. Konkrétna realizácia grafu závisí od typu grafu, ktorý si používateľ zvolil:

- Neorientovaný graf: Hrany nemajú orientáciu, teda spojenia medzi vrcholmi sú obojsmerné.

Obrázok 41 Ukážka kódu Neorientovaný graf

```
295         if (input$discreteType == "undirected") {
296           g <- graph_from_edgelist(edges, directed = FALSE)
297           p <- ggraph(g, layout = "fr") +
298             geom_edge_link(edge_colour = "black") +
299             geom_node_point(color = "blue", size = 5) +
300             geom_node_text(aes(label = name), repel = TRUE) +
301             labs(title = "Neorientovaný graf") +
302             theme_minimal()
303           return(p)
```

Zdroj: Vlastné spracovanie

- Orientovaný graf: Hrany majú jasne definovaný smer, ktorý je znázornený šípkami pomocou parametra *arrow()* vo funkcii *geom_edge_link()*.

Obrázok 42 Ukážka kódu Orientovaný graf

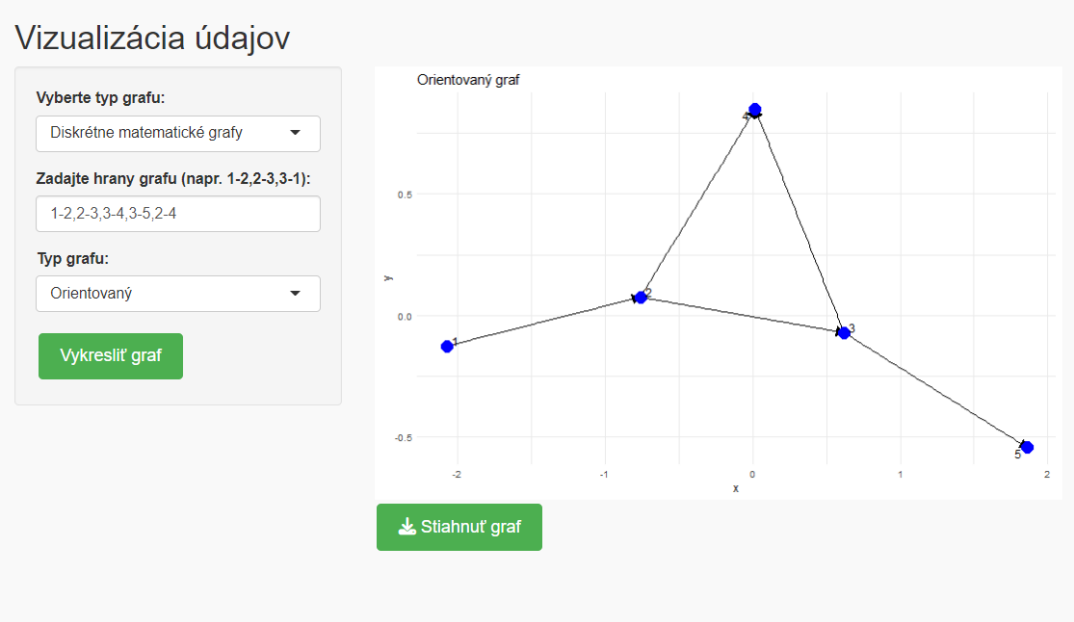
```
304         } else if (input$discreteType == "directed") {
305           g <- graph_from_edgelist(edges, directed = TRUE)
306           p <- ggraph(g, layout = "fr") +
307             geom_edge_link(arrow = arrow(length = unit(3, "mm"), type = "closed"),
308                            edge_colour = "black") +
309             geom_node_point(color = "blue", size = 5) +
310             geom_node_text(aes(label = name), repel = TRUE) +
311             labs(title = "Orientovaný graf") +
312             theme_minimal()
313           return(p)
```

Zdroj: Vlastné spracovanie

Všetky typy grafov sú vykreslené pomocou funkcie *ggraph()* s algoritmom layout = "fr", ktorý využíva algoritmus Fruchterman-Reingold na optimalizované usporiadanie vrcholov.

Ako príklad sme uviedli orientovaný graf, ktorý demonštruje funkčnosť našej aplikácie pre tvorbu diskrétnych grafov. Tento graf bol vytvorený na základe vstupných údajov o hranách, pričom uzly reprezentujú jednotlivé prvky siete a hrany zobrazujú prepojenia medzi uzlami a šípky indikujú smer týchto prepojení.

Obrázok 43 Graf diskrétnej matematiky



Zdroj: Vlastné spracovanie

4.3.7 Vykreslenie a sťahovanie grafu

Táto časť kódu zabezpečuje samotné vykreslenie a sťahovanie výsledných grafov v aplikácii.

Obrázok 44 Vykresľovanie a sťahovanie grafov

```
320  output$plotOutput <- renderPlot({
321    req(myPlot())
322    myPlot()
323  })
324
325  output$downloadPlot <- downloadHandler(
326    filename = function() {
327      paste("graf", Sys.Date(), ".png", sep = "")
328    },
329    content = function(file) {
330      ggsave(file, plot = myPlot(), device = "png", width = 8, height = 5)
331    }
332  )
333 }
```

Zdroj: Vlastné spracovanie

Najskôr sa výsledný graf generuje pomocou funkcie *renderPlot()*. Táto funkcia využíva reaktívnu funkciu *myPlot*(), ktorá bola definovaná v predchádzajúcej časti serverovej logiky. Funkcia *req(myPlot())* zabezpečí, že graf sa vykreslí až vtedy, keď je dostupný a platný výstup z *myPlot()*. Po úspešnom vygenerovaní je graf zobrazený priamo v hlavnom paneli aplikácie.

Okrem samotného vykreslenia grafu ponúka aplikácia možnosť tento graf aj stiahnuť. Na to slúži tlačidlo s funkciou *downloadHandler()*, ktoré je označené ako „Stiahnuť graf“. Používateľ si tak môže výsledok svojej práce jednoducho uložiť vo formáte PNG.

Obrázok 45 Spustenie Shiny aplikácie

```
334  shinyApp(ui = ui, server = server)
```

Zdroj: Vlastné spracovanie

Na záver kódu sa spustí samotná aplikácia pomocou funkcie *shinyApp()*, ktorá spája definované používateľské rozhranie (UI) a serverovú logiku (server). Výsledkom je kompletná interaktívna aplikácia pre vizualizáciu dát rôzneho charakteru.

Záver

Cieľom tejto bakalárskej práce bolo oboznámenie sa s programovacím jazykom R a definovanie základných pojmov z oblasti matematických funkcií, štatistiky a diskrétnych grafov, pričom získané poznatky sme následne aplikovali na vizualizáciu údajov pomocou Shiny aplikácie.

V praktickej časti sme sa zamerali na využitie dostupných balíkov jazyka R slúžiacich na vizualizáciu dát. Pre zobrazovanie matematických funkcií sme vytvárali grafy pomocou balíka ggplot2. Okrem priameho zadávania príkazov na tvorbu grafov sme implementovali aj dynamické ovládacie prvky v Shiny aplikácii, ktoré umožňujú interaktívne nastavovať parametre a prispôsobovať výsledné vizualizácie, čím zvyšujú užívateľskú prívetivosť a flexibilitu práce s dátami.

Na vykresľovanie základných štatistických grafov sme v našej práci využili opäť balík ggplot2. Štatistické súbory, s ktorými sme pracovali, sme získali z voľne dostupných datasetov alebo zo Štatistického úradu Slovenska a následne sme ich upravili pomocou balíkov dplyr a tidyr podľa potreby vizualizácie. Tieto úpravy umožnili efektívne zobraziť relevantné štatistické charakteristiky údajov, čo prispelo k prehľadnej interpretácii a analýze prezentovaných dát.

Získané poznatky prispievajú k lepšiemu pochopeniu možností vizualizácie dát v R a otvárajú priestor pre ďalší vývoj v tejto oblasti. Ako smer budúcej práce je možné uviesť rozšírenie funkcionalít Shiny aplikácie o ďalšie typy grafov a integráciu s pokročilými analytickými metódami, čo by mohlo prispieť k ešte efektívnejšej prezentácii a spracovaniu komplexných údajov.

Celkovo možno konštatovať, že táto práca presvedčivo ukazuje, aký široký potenciál ponúka jazyk R so svojimi balíkmi pri vytváraní interaktívnych vizualizačných riešení s reálnym využitím v akademickom výskume aj v praktických aplikáciách.

Zoznam použitej literatúry

Elektronické zdroje:

GeeksforGeeks. (2023). *Types of graphs in statistics*. https://www.geeksforgeeks.org/types-of-graphs-in-statistics/

Gilbert, L. (n.d.). *Function: Terms & graph examples*. Study.com. https://study.com/learn/lesson/function-terms-graph-examples-math.html

Chen, J. (2022). *Histogram definition*. Investopedia. https://www.investopedia.com/terms/h/histogram.asp

Indeed Editorial Team. (2025). *Presenting and arranging data: How to explain a graph*. Indeed Career Guide. https://www.indeed.com/career-advice/career-development/how-to-explain-a-graph

Kvasnička, M. (2025). *Úvod do R*. MUNI.cz. https://is.muni.cz/el/econ/podzim2016/BPE_AVED/um/64916157/uvod_do_R.html

Markechová, D., Stehlíková, B., & Tirpáková, A. (2011). *Štatistické metódy a ich aplikácie*. Univerzita Konštantína Filozofa v Nitre. https://www.researchgate.net/profile/Dagmar-Markechova/publication/313724644_Statistical_Methods_and_their_Applications/links/58a4293592851ce3473d7e0b/Statistical-Methods-and-their-Applications.pdf

Robinson, S. (2024, May 31). *What is R programming language? – Definition from WhatIs.com*. SearchBusinessAnalytics. https://www.techtarget.com/searchbusinessanalytics/definition/R-programming-language

Rostás, J. (2023). *Základy štatistiky*. Univerzita Komenského v Bratislave. http://hore.dnom.fmph.uniba.sk/~rostas/vyuka/1gk/ZS/Zaklady%20statistiky_v2.pdf

Wickham, H. (2024, April). Chapter 1: Your first shiny app. In *Mastering Shiny*. O'Reilly Media. https://mastering-shiny.org/basic-app.html

Yi, M. (2023). *A complete guide to pie charts*. Atlassian. https://www.atlassian.com/data/charts/pie-chart-complete-guide

Yi, M. (2024). *Box plot: A complete guide*. Atlassian. https://www.atlassian.com/data/charts/box-plot-complete-guide

Knižné zdroje:

Bosák, J. (1980). *Grafy a ich aplikácie* (2nd ed.). Alfa.

Brasseur, L. E. (2003). *Visualizing technical information: A cultural critique*. Baywood Publishing Company.

Labudová, V., Pacáková, V., Sipková, Ľ., Šoltés, E., & Vojtková, M. (2021). *Štatistické metódy pre ekonómov a manažérov*. Wolters Kluwer SR.

Roubiček, V. (1967). *Stručný statistický slovník pro hospodářské pracovníky*. Svoboda.