EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/B/2025/36146475540420100

Porovnanie štatistických metód v programovacích jazykoch R a Python

Bakalárska práca

2025 **Benjamín Kolárik**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Porovnanie štatistických metód v programovacích jazykoch R a Python

Bakalárska práca

Študijný program: Hospodárska Informatika

Študijný odbor: Ekonómia a manažment

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: Ing. Pavol Sojka, PhD.

Bratislava 2025 **Benjamín Kolárik**

Čestné vyhlásenie:

Čestne vyhlasujem, že som túto záverečnú prácu vypracoval samostatne na základe vlastných poznatkov, a že som uviedol všetku použitú literatúru.

Dátum:

.....

Benjamín Kolárik

Poďakovanie:

Chcel by som sa poďakovať môjmu školiteľovi Ing. Pavlovi Sojkovi, PhD. za podporu, pomoc a rady pri písaní tejto práce.

ABSTRAKT

KOLÁRIK, Benjamín: *Porovnanie štatistických metód v programovacích jazykoch R a Python* [Bakalárska práca]. - Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. - Vedúci práce: Ing. Pavol Sojka, PhD. Stupeň odbornej kvalifikácie: bakalár. Bratislava. FHI EU, 2025. 69 s.

Cieľom tejto bakalárskej práce je porovnať možnosti implementácie a výpočtovú efektivitu dvoch základných štatistických metód – lineárnej regresie a analýzy rozptylu v programovacích jazykoch R a Python. Teoretická časť sa venuje prehľadu základných princípov oboch metód a ich využitiu v analytickej praxi. Približuje dostupné knižnice v jednotlivých jazykoch a stručne predstavuje ich základné možnosti a využitie.

Praktická časť je zameraná na implementáciu oboch metód formou skriptov. Skripty boli testované na viacerých počítačoch s rôznym hardvérom a operačnými systémami. Pre každú kombináciu metódy, jazyka a veľkosti dát boli spustené viackrát, pričom sa meral čas ich vykonania. Získané výsledky boli spriemerované a porovnané medzi jednotlivými jazykmi a metódami. Ukázalo sa, že Python bol rýchlejší pri výpočte lineárnej regresie, zatiaľ čo R dosiahlo lepšie výsledky pri analýze rozptylu. Práca poukazuje na to, že výber jazyka by mal byť ovplyvnený konkrétnymi požiadavkami projektu, znalosťami používateľa a dostupnými výpočtovými prostriedkami.

Kľúčové slová:
R, Python, lineárna regresia, ANOVA, dátová analýza, štatistika, exekučný čas, grafy

ABSTRACT

KOLÁRIK, Benjamín: *Comparison of Statistical Methods in Programming Languages R and Python* – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Advisor: Ing. Pavol Sojka, PhD. Bratislava: FHI EU, 2025. 69 p.

The aim of this bachelor's thesis is to compare the implementation options and computational efficiency of two fundamental statistical methods – linear regression and analysis of variance (ANOVA) – in the programming languages R and Python. The theoretical part provides an overview of the basic principles of both methods and their applications in analytical practice. It also briefly introduces the available libraries in each language and summarizes their main functionalities and use cases.

The practical part focuses on the implementation of both methods in the form of scripts. These scripts were tested on multiple computers with different hardware configurations and operating systems. For each combination of method, language, and dataset size, the scripts were executed multiple times, and execution time was measured. The collected results were averaged and compared across languages and methods. The findings showed that Python was faster in linear regression computations, while R achieved better results in the analysis of variance. The thesis highlights that the choice of language should be influenced by the specific requirements of the project, the user's expertise, and the available computational resources.

Keywords:
R, Python, linear regression, ANOVA, data analysis, statistics, execution time, graphs

Zoznam tabuliek:

Zoznam obrázkov:

Zoznam grafov:

Zoznam skratiek:

ANOVA – analýza rozptylu (z angl. Analysis of Variance)
API – rozhranie pre programovanie aplikácii (z angl. Application Programming Interface)
CSV – hodnoty oddelené číslom (z angl. Comma-seperated values)
GB – gigabyte
IDE – vývojové prostredie (Integrated Development Enviroment)
IQR – medzikvartilové rozpätie (Interquartile range)
LR – lineárna regresia (z angl. Linear Regression)
MANOVA - Multiple Analysis of Variance
ML – strojové učenie (z angl. Machine Learning)
OLS – metóda najmenších štvorcov (z angl. Ordinary Least Squares)
OOP – objektovo orientované programovanie (z angl. Object-oriented programming)
OS – operačný systém
PC – počítač (z angl. Personal Computer)
SQL – Structured Query Language
UMA - Unified Memory Architecture

Obsah

Úvod

V súčasnosti sa štatistické metódy stávajú čoraz dôležitejším nástrojom pri riešení praktických problémov naprieč rôznymi oblasťami – od ekonomiky a financovania, cez zdravotníctvo a prírodné vedy, až po oblasť umelej inteligencie a strojového učenia. V rámci týchto disciplín zohráva kľúčovú úlohu najmä lineárna regresia, ktorá umožňuje modelovanie vzťahu medzi premennými, a analýza rozptylu (ANOVA), ktorá slúži na porovnanie stredných hodnôt medzi rôznymi skupinami. Tieto metódy patria medzi základné nástroje štatistického modelovania a analytickej práce s dátami.

Pri ich praktickej aplikácii sa vo veľkej miere využívajú dva programovacie jazyky – R a Python. Oba jazyky disponujú rozsiahlym ekosystémom knižníc, ktoré umožňujú efektívnu prácu s dátami, ich vizualizáciu, modelovanie a interpretáciu výsledkov. Kým jazyk R je dlhodobo etablovaný v akademickej sfére a špecializuje sa predovšetkým na štatistické výpočty, Python je známy svojou univerzálnosťou a dominanciou v oblasti dátovej vedy a vývoja aplikácií. Vzhľadom na rozdiely v architektúre, filozofii a dostupných nástrojoch predstavujú tieto dva jazyky zaujímavý predmet porovnania z hľadiska štatistickej efektivity a použiteľnosti.

Cieľom tejto bakalárskej práce je preto porovnať implementáciu dvoch vybraných štatistických metód – jednoduchej lineárnej regresie a analýzy rozptylu – v prostredí programovacích jazykov R a Python. Porovnanie je realizované nielen z pohľadu syntaktickej náročnosti a čitateľnosti kódu, ale aj z pohľadu výpočtovej efektivity, presnosti výstupov a možností vizualizácie. V praktickej časti sú vytvorené vlastné datasety rôznych veľkostí a sú testované skripty v oboch jazykoch na rôznych konfiguráciách výpočtovej techniky. Výsledky sú následne spriemerované, analyzované a interpretované.

1. Využitie Pythonu a R v štatistickej analýze

V súčasnosti sú Python a R dvoma najpopulárnejšími programovacími jazykmi na štatistickú analýzu a prácu s dátami. Obe technológie majú svoje špecifiká a silné stránky, vďaka čomu sú široko používané v akademickom i priemyselnom prostredí. [7]

Python sa vyznačuje univerzálnosťou a je obľúbený najmä v oblastiach strojového učenia (ML) a dátovej vedy. Na druhej strane, R bolo špecificky navrhnuté pre štatistickú analýzu a vizualizáciu dát. [7] Táto kapitola sa zameriava na prehľad oboch programovacích jazykov, ich využitie v štatistickom modelovaní, dostupné knižnice a výhody a nevýhody, ktoré každý jazyk prináša.

1.1. Úvod do Pythonu

Programovací jazyk Python bol vyvinutý v roku 1989 Guido van Rossumom, holandským programátorom ako hobby projekt, ktorý bol postavený na filozofii, že programovanie má byť zábávne. Je deriváciou rôznych programovacích jazykov ako je C, C++, ABC a ďalších skriptovacích jazykov. [15, 31]

Prvé oficiálne vydanie Pythonu 1.0 bolo 26. januára 1994 a obsahovalo základy objektovo orientovaného programovania, nové dátové štruktúry a koncept odchytávania chýb počas behu programu. V syntaxi sa kladie dôraz na jednoduchosť a čitateľnosť kódu. Python 2.0 bol následne vydaný v roku 2000 a verzia 3.0 v roku 2008, ktorá je zároveň aj najnovšou, dodnes využívanou verziou. [15, 31]

Postupom rokov sa jazyk stal čoraz viac populárnym, vznikali nové voľne dostupné knižnice pre rôzne odvetvia ako je machine learning, dátová analýza, web development. Vďaka jeho širokému využitiu dnes patrí medzi najpoužívanejšie a najžiadanejšie programovacie jazyky na trhu. [15, 31]

Python je jedným z najpoužívanejších programovacích jazykov v oblasti machine learningu a dátovej vedy. Oproti jazykom ako C++ a Java má lepšiu čitateľnosť a jednoduchú syntax, širokú komunitu používateľov s rozsiahlou dokumentáciou a rôznych návodov či tutoriálov, čo uľahčuje učenie a riešenie problémov – aj za cenu pomalšej exekučnej rýchlosti. [24]

Knižnice ako TensorFlow a PyTorch, ktoré vyvinuli Google resp. Meta sú základnými nástrojmi na machine a deep learning. Sú známe svojou flexibilitou a schopnosť škálovania od jednoduchých modelov po komplexné neurónové siete. [15, 24]

1.1.1. Knižnice pre štatistickú analýzu a manipuláciu s dátami

Pandas je jedna z najpoužívanejších knižníc na prácu s dátami, poskytuje výkonné nástroje na manipuláciu, analýzu a čistenie dát vo formáte tabuliek (DataFrame) alebo sérií (Series). Dáta sa dajú načítavať z rôznych zdrojov, ako sú CSV, Excel, SQL dopytmi z databázy alebo JSON súborov. [9] Na jej používanie musíme knižnicu lokálne nainštalovať a na začiatku kódu importovať, napr „import pandas as pd“. Pd v tomto prípade slúži ako skratka pre funkcie danej knižnice.

Medzi základné funkcie patrí:

Kategória	Funkcia	Popis
Načítanie a ukladanie dát	pd.read_csv("data.csv")	Načítanie CSV súboru do DataFrame
	df.to_csv("output.csv", index=False)	Uloženie DataFrame do CSV
Základné operácie	df.head(n)	Zobrazí prvých *n* riadkov
	df.tail(n)	Zobrazí posledných *n* riadkov
	df.info()	Zobrazí informácie o DataFrame
	df.describe()	Štatistický prehľad číselných stĺpcov
Výber a filtrovanie	df["stĺpec"]	Výber jedného stĺpca
	df[["stĺpec1", "stĺpec2"]]	Výber viacerých stĺpcov
	df.loc[riadok, "stĺpec"]	Výber hodnoty na konkrétnej pozícii
	df[df["stĺpec"] > hodnota]	Filtrovanie na základe podmienky
Manipulácia s dátami	df["nový_stĺpec"] = df["stĺpec1"] + df["stĺpec2"]	Pridanie nového stĺpca
	df.drop("stĺpec", axis=1, inplace=True)	Odstránenie stĺpca
	df.rename(columns={"stary": "novy"}, inplace=True)	Premenovanie stĺpca
Práca s chýbajúcimi hodnotami	df.isnull().sum()	Počet chýbajúcich hodnôt v stĺpcoch
	df.dropna()	Odstránenie riadkov s chýbajúcimi hodnotami
	df.fillna(hodnota, inplace=True)	Nahradenie chýbajúcich hodnôt

Tabuľka č. 1: Základne funkcie Pandas

(Zdroj: Vlastné spracovanie podľa: [9, 32])

SciPy predstavuje knižnicu na numerické výpočty, ktorá rozširuje funkcionalitu NumPy o pokročilé matematické a štatistické nástroje [26]. V praktickej bola využitá na **výpočet F-value a p-value.**

NumPy slúži na prácu s číselnými udajmi a viacrozmernými poľami. Poskytuje efektívne dátové štruktúry a funkcie na rýchle matematické operácie, čím je základom pre knižnice ako pandas, scipy alebo scikit-learn. [17] Pred jej použitím ju musíme importovať (import numpy as np), rovnako ako pre pandas a aj pre zvyšné knižnice.

Kategória	Funkcia	Popis
Vytvorenie NumPy poľa	np.array([1, 2, 3])	Vytvorenie 1D poľa (vektora)
	np.array([[1, 2], [3, 4]])	Vytvorenie 2D matice
	np.zeros((3,3))	Vytvorí maticu 3×3 plnú núl
	np.ones((2,2))	Vytvorí maticu 2×2 plnú jednotiek
	np.eye(3)	Vytvorí 3×3 jednotkovú maticu
	np.arange(0, 10, 2)	Pole čísel od 0 do 10 s krokom 2
Základné operácie	np.mean(arr)	Priemer hodnôt v poli
	np.sum(arr)	Súčet prvkov v poli
	np.min(arr)	Najmenšia hodnota v poli
	np.max(arr)	Najväčšia hodnota v poli
	np.std(arr)	Štandardná odchýlka
Matematické operácie	np.dot(A, B)	Skalárny súčin dvoch matíc
	np.transpose(A)	Transponuje maticu
	np.linalg.inv(A)	Inverzná matica
	np.linalg.det(A)	Determinant matice
	np.linalg.eig(A)	Vlastné čísla a vlastné vektory matice

Tabuľka č. 2: Základné funkcie NumPy

(Zdroj: Vlastné spracovanie podľa: [17])

Statsmodels je výkonná knižnica na štatistické modelovanie, testovanie hypotéz a analýzu údajov. Poskytuje pokročilé metódy pre regresnú analýzu, časové rady a štatistické testy. [28]

```python
import statsmodels.api as sm
#Generovanie náhodných dát
np.random.seed(42)
X = np.random.rand(100, 1)
y = 2 + 3 * X + np.random.rand(100, 1)

#Pridanie konštanty (intercept)
X = sm.add_constant(X)

#Lineárna regresia metódou najmenších štvorcov
model = sm.OLS(y, X).fit()
print(model.summary())
```

Obrázok č. 1: Implementácia Statsmodels

(Zdroj: vlastné spracovanie podľa: [28])

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.901
Model:                            OLS   Adj. R-squared:                  0.900
Method:                 Least Squares   F-statistic:                     889.4
Date:                Fri, 14 Mar 2025   Prob (F-statistic):           5.86e-51
Time:                        17:43:00   Log-Likelihood:                -18.613
No. Observations:                 100   AIC:                             41.23
Df Residuals:                      98   BIC:                             46.44
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.5136      0.055     45.482      0.000       2.404       2.623
x1             2.9665      0.099     29.822      0.000       2.769       3.164
==============================================================================
Omnibus:                       51.507   Durbin-Watson:                   1.965
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                6.965
Skew:                           0.059   Prob(JB):                       0.0307
Kurtosis:                       1.712   Cond. No.                         4.18
==============================================================================
```

Obrázok č. 2: Výstup Statsmodels

(Zdroj: vlastné spracovanie podľa: [28])

Openpyxl dokáže podobne ako knižnica **Pandas** čítať, zapisovať a upravovať súbory vo formáte .xlsx (Excel). Podporuje rôzne funkcie, ako prácu s hárkami, bunkami, formátovaním, grafmi, tabuľkami a inými objektmi v Excel dokumentoch. [18]

Scikit-learn sa používa pre machine learning, poskytuje jednoduché a efektívne nástroje na modelovanie a analýzu dát. Je postavená na knižniciach SciPy, NumPy a matplotlib. [25]

Matplotlib je najpopulárnejšia knižnica na vizualizáciu dát. Je veľmi flexibilná a umožnuje vytvárať širokú škálu grafov a vizualizácií – od jednoduchých grafov až po komplexné interaktívne vizualizácie. Taktiež umožňuje mnoho možností formátovania, ako je farba, štýl čiary, legenda či popisky. [16]

Kategória	Funkcia	Popis
Základné grafy	matplotlib.pyplot.plot()	Vytvorenie čiarového grafu
	matplotlib.pyplot.scatter()	Vytvorenie scatter (bodového) grafu
	matplotlib.pyplot.bar()	Vytvorenie stĺpcového grafu
	matplotlib.pyplot.hist()	Vytvorenie histogramu
Prispôsobenie grafu	plt.title()	Nastavenie názvu grafu
	plt.xlabel()	Nastavenie popisku osi x
	plt.ylabel()	Nastavenie popisku osi y
	plt.legend()	Zobrazenie legendy
Nastavenie farieb	plt.plot(..., color='blue')	Nastavenie farby grafu
	plt.bar(..., color='green')	Nastavenie farby stĺpcového grafu
Rozmery a rozloženie	plt.figure(figsize=(10, 6))	Nastavenie veľkosti obrázka
	plt.subplots_adjust()	Nastavenie rozloženia grafu
Uloženie grafu	plt.savefig('graf.png')	Uloženie grafu do súboru
Zobrazenie grafu	plt.show()	Zobrazenie grafu
Vytváranie podgrafov	plt.subplot()	Vytvorenie podgrafu v jednom obrázku

Tabuľka č. 3: Základné funkcie Matplotlib

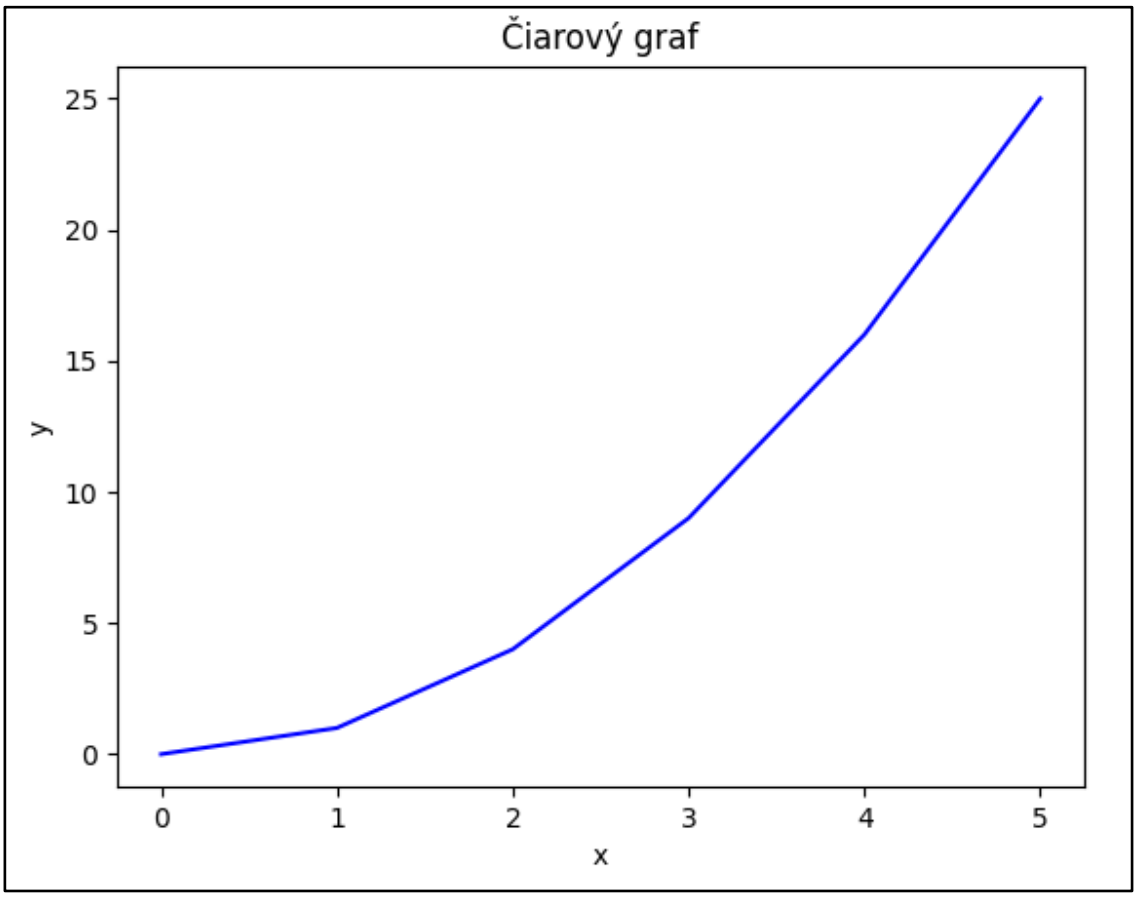
(Zdroj: Vlastné spracovanie podľa: [16])

```python
import matplotlib.pyplot as plt
#Vizualizácia dát
x = [0, 1, 2, 3, 4, 5]
y = [0, 1, 4, 9, 16, 25]

plt.plot( *args: x, y, label='y = x^2', color='blue')
plt.title('Čiarový graf')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Obrázok č. 3: Implementácia Matplotlib

(*Zdroj: vlastné spracovanie podľa: [16]*)



Graf č. 1: Čiarový graf vytvorený pomocou Matplotlib

(Zdroj: vlastné spracovanie podľa: [16])

Seaborn je postavená na základoch **matplotlib**, s ktorou sa výborne dopĺňa a rozširuje ju. Má jednoduchú syntax a prednastavené témy (ktoré sa môžu personalizovať), vďaka ktorým sa dajú rýchlo vytvárať štatistické grafy, ako sú boxploty, pairploty, scatter ploty atď. [27]

```
import seaborn as sns
#Vizualizácia dát
data = np.random.rand(10, 12)

#Vytvorenie heatmapy
sns.heatmap(data, cmap='coolwarm')
plt.title('Heatmapa')
plt.show()
```

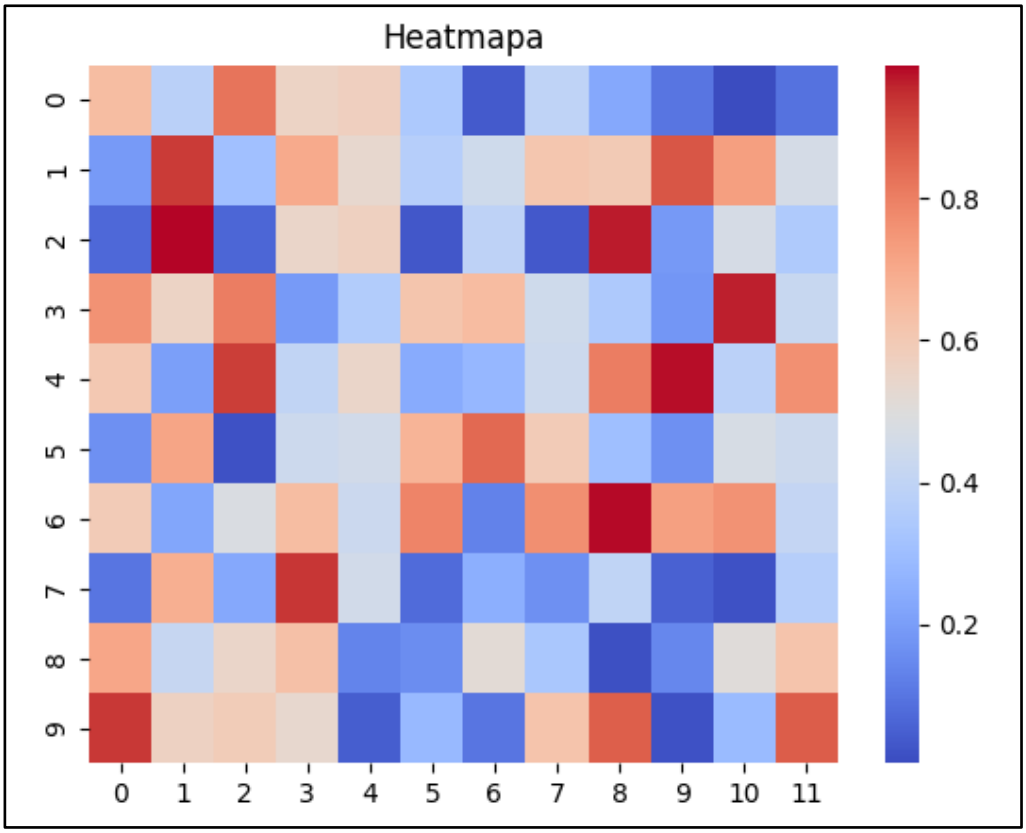
Obrázok č. 4: Implementácia Seaborn

(Zdroj: vlastné spracovanie podľa: [27])

Kategória	**Funkcia**	**Popis**
Základné grafy	seaborn.lineplot()	Čiarový graf (line plot)
	seaborn.barplot()	Stĺpcový graf (bar plot)
	seaborn.scatterplot()	Bodový graf (scatter plot)
	seaborn.histplot()	Histogram
	seaborn.boxplot()	Boxplot (box and whisker plot)
	seaborn.violinplot()	Violinplot
Štatistické grafy	seaborn.regplot()	Regresný graf
	seaborn.heatmap()	Heatmap (teplotná mapa)
	seaborn.pairplot()	Párový graf (pair plot)
FacetGrid	seaborn.FacetGrid()	Vytvorenie facet gridu pre viaceré grafy
Témy a štýly grafov	seaborn.set_style()	Nastavenie štýlu grafov
	seaborn.set_palette()	Nastavenie palety farieb
Prispôsobenie grafov	seaborn.set_context()	Nastavenie kontextu (napr. pre publikácie)

Tabuľka č. 4: Základné funkcie Seaborn

(Zdroj: Vlastné spracovanie podľa: [27])



Graf č. 2: Heatmapa vytvorená pomocou knižnice Seaborn

(Zdroj: vlastné spracovanie podľa: [27])

1.1.2. Silné a slabé stránky Pythonu pre štatistické modelovanie

Silné stránky Pythonu pre štatistické modelovanie:

1. **Bohatá podpora knižníc:**

   Ekosystém zahŕňa množstvo nástrojov a knižníc určených na štatistické analýzy a modelovanie, ako sú NumPy, SciPy, pandas, statsmodels, scikit-learn, TensorFlow a PyTorch. Tieto knižnice umožňujú efektívne vykonávanie rôznych štatistických analýz, regresií, strojového učenia a vizualizácií. [29]

2. **Jednoduchá a čitateľná syntax:**

   Zrozumiteľnosť kódu a čitateľná syntax patria medzi hlavné prednosti. Vďaka tomu je vhodný aj pre používateľov bez rozsiahlych programátorských skúseností. [29]

3. **Flexibilita a univerzálnosť:**

   Vďaka svojej flexibilite je vhodný pre širokú škálu úloh, od základných štatistických analýz až po komplexné aplikácie v oblasti strojového učenia, analýzy veľkých dát a umelej inteligencie. [24, 29]

4. **Integrácia s inými nástrojmi:**

   Bezproblémovo komunikuje s databázovými systémami, webovými rozhraniami aj API službami, čo výrazne uľahčuje nasadenie modelov do produkčného prostredia alebo ich integráciu do komplexnejších aplikácií. [14]

5. **Pokročilá vizualizácia dát:**

   Knižnice ako *Matplotlib*, *Seaborn* a *Plotly* poskytujú široké možnosti pre tvorbu pokročilých vizualizácií dát a výsledkov štatistických modelov, čo je kľúčové pre správnu interpretáciu a prezentáciu analýz. [1]

Slabé stránky Pythonu pre štatistické modelovanie:

1. **Nižší výkon v porovnaní s kompilovanými jazykmi:**

   Ako interpretovaný jazyk môže byť pomalší než niektoré kompilované jazyky, ako je C alebo C++, čo môže ovplyvniť efektivitu pri práci s veľmi veľkými dátovými súbormi alebo pri výpočtovo náročných operáciách. Avšak, knižnice ako Numpy alebo Pandas využívajú optimalizované algoritmy v jazyku C, čo zlepšuje výkon. [14, 29]

2. **Obmedzená podpora pre špecializované štatistické metódy:**

   Hoci je pokrytie väčšiny bežne používaných metód výborné, aplikácia niektorých špecializovaných techník – napríklad pokročilých Bayesovských modelov – môže byť obmedzená. V takých prípadoch je vhodnejšou voľbou špecializovaný nástroj ako R. [10]

3. **Zložitejšia práca s pokročilými štatistickými modelmi**

   Samotné nastavenie a konfigurácia Pythonu môže byť zložitejšia ako napríklad R, špeciálne, ak sa jedná o komplexné dátové analýzy alebo úlohy strojového učenia. [10]

1.2. R – história a špecializácia na štatistické výpočty

R je open-source programovací jazyk, ktorý sa špecializuje na štatistické výpočty a vizualizáciu dát. Bol vyvinutý v 90. rokoch minulého storočia programátorom Rossom Ihakom. Snahou bolo vytvoriť nástroj na analýzu dát, ktorý by bol flexibilný a prístupný výskumníkom bez väčších programátorských skúseností. [19]

R bolo vytvorené ako alternatíva ku jazyku S, ktorý sa už používal v štatistike a výskume, ale bol komerčne licencovaný. Syntax oboch jazykov je podobná, ale R je rozšírené o rozsiahly balík knižníc, vďaka ktorým sa dá využiť aj v oblasti machine learningu. V súčasnosti je jazyk jedným z najpoužívanejších v oblastiach ako štatistika, bioinformatika, ekonómia a ďalších, kde je analýza a vizualizácia dát kľúčová. [19]

Vyznačuje sa bohatým ekosystémom knižníc a balíkov, ktoré pokrývajú široké spektrum analytických úloh – od základných štatistických testov a regresných modelov až po pokročilé metódy machine learningu a analýzu big data. Taktiež obsahuje podporu pre tvorbu vizualizácií. [19]

1.2.1. Hlavné balíky/knižnice v R pre štatistickú analýzu

Stats je základný balík v jazyku R (už je súčasťou štandardnej distribúcie), ktorý obsahuje množstvo nástrojov pre štatistické výpočty a analýzu dát, ako aj pre implementáciu rôznych štatistických modelov. [23]

Funkcia	Popis	Príklad použitia
lm()	Vykonáva lineárnu regresiu (jednoduchú aj viacnásobnú).	lm(y ~ x1 + x2, data = dataset)
aov()	Vykonáva analýzu rozptylov (ANOVA) na základe faktorových modelov.	aov(y ~ factor, data = dataset)
t.test()	Vykonáva t-test na porovnanie dvoch skupín.	t.test(x, y)
cor()	Vypočíta koreláciu medzi dvoma alebo viacerými premennými.	cor(x, y)
filter()	Aplikuje filtračné operácie na časové série a signály.	filter(x, filter)
anova()	Vykonáva analýzu rozptylov medzi rôznymi modelmi.	anova(model1, model2)

Tabuľka č. 5: Základné funkcie Stats

(Zdroj: Vlastné spracovanie podľa: [23])

```r
#Generovanie dát pre lineárnu regresiu
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
y <- c(2, 4, 5, 7, 8, 10, 12, 14, 16, 18)

#Model lineárnej regresie
model <- lm(y ~ x)
summary(model)

#Generovanie dvoch vektorov
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)

#Výpočet korelácie
correlation <- cor(x, y)
correlation
```

Obrázok č. 5: Impelemtácia Stats

(Zdroj: vlastné spracovanie podľa: [23])

Tidyr tvorí súčasť tidyverse, čo je kolekcia knižníc navrhnutých na efektívne spracovanie, analýzu a vizualizáciu dát. Zameriava sa predovšetkým na manipuláciu s DataFrame a pretransformovanie dát do formátu, ktorý je vhodný na analýzu a vizualizáciu. Hlavným cieľom knižnice je teda uľahčiť proces čistenia a prípravy dát. [5, 11]

Funkcia	Popis	Príklad použitia
pivot_longer()	Pretransformuje dáta z širokého formátu do dlhého formátu (long format).	pivot_longer(df, cols = starts_with("score"))
pivot_wider()	Pretransformuje dáta z dlhého formátu do širokého formátu (wide format).	pivot_wider(df, names_from = "variable", values_from = "value")
separate()	Rozdelí jednu stĺpec na viacero stĺpcov podľa určitého oddelovača.	separate(df, col = "date", into = c("year", "month", "day"))
unite()	Spojí viaceré stĺpce do jedného stĺpca.	unite(df, "full_name", first_name, last_name, sep = " ")
fill()	Vyplní chýbajúce hodnoty v dátach na základe predchádzajúcich alebo nasledujúcich hodnôt.	fill(df, column_name, .direction = "down")
drop_na()	Odstráni riadky obsahujúce NA hodnoty v jednom alebo viacerých stĺpcoch.	drop_na(df, column_name)

Tabuľka č. 6: Základné funkcie Tidyr

(Zdroj: Vlastné spracovanie podľa: [5, 11])

Readxl je knižnica na načítavanie údajov zo súborov vo formáte excel (.xls a xlsx) do prostredia R - podobne ako knižnica **openxyl** v Pythone, avšak ako aj z názvu vyplýva, bez možnosti zápisu dát do Excelu. [30]

Dplyr, súčasť balíka tidyverse, slúži na manipuláciu s dátami. Knižnica poskytuje efektívne funkcie na filtrovanie, triedenie a modifikovanie DataFrame a iných dátových štruktúr. Použitie knižnice je vhodné, pokiaľ pracujeme s veľkými datasetmi. [20, 33]

Funkcia	**Popis**	**Príklad použitia**
select()	Vyberie určité stĺpce z DataFrame.	select(df, name, score)
filter()	Filtruje riadky na základe podmienok.	filter(df, score > 90)
arrange()	Usporiadanie dát podľa hodnoty v stĺpci (vzostupne alebo zostupne).	arrange(df, score)
mutate()	Vytvorí nový stĺpec alebo modifikuje existujúci stĺpec na základe výpočtu.	mutate(df, score_scaled = score / 100)
summarise()	Vypíše základné údaje (súčet, priemer, min, max, atď.) na základe vybraných stĺpcov.	summarise(df, avg_score = mean(score))
group_by()	Skupinuje dáta podľa určitého stĺpca alebo viacerých stĺpcov.	group_by(df, group)
rename()	Premenuje stĺpce v dátovom rámci.	rename(df, new_name = old_name)
bind_rows()	Zlúči (spojí) dva dátové rámce do jedného (pridá riadky).	bind_rows(df1, df2)
bind_cols()	Zlúči dva dátové rámce do jedného (pridá stĺpce).	bind_cols(df1, df2)

Tabuľka č. 7: Základné funkcie Dplyr

(Zdroj: Vlastné spracovanie podľa: [20, 33])

Ggplot2 slúži na vizualizáciu dát. Táto knižnica je taktiež súčasťou **tidyverse** a je založená na „gramatike grafov“. Podporuje vytváranie jednoduchých ale aj komplexných grafov, napr. Histogramy, boxploty, heatmapy atď. [21]

Každý graf pozostáva zo základných komponentov (nemusia byť všetky definované):

- **Data**: vstupná dátová množina, z ktorej budú čerpané údaje,
- **Aesthetics (aes):** určuje, ako budú údaje zobrazené (farby, tvary, atď.),
- **Geometries (geom):** definuje typ grafu (čiarový geom_line(), bodový geom_point()...),

- **Statistics (stat):** špecifikuje transformáciu údajov pred ich vykreslením,
- **Faceting**: umožňuje rozdelenie grafu do viacerých panelov,
- **Coordinates (coord):** poskytuje nástroje na manipuláciu s osami pomocou rôznych typov koordinátov,
- **Themes**: zmena farby pozadia, fontu atď. [4, 21]

Funkcia	Typ grafu	Popis	Príklad použitia
ggplot()	Základná funkcia	Vytvára základný graf, kde sa definuje dátová množina a estetické vlastnosti (napr. osy, farby).	ggplot(data, aes(x = var1, y = var2))
geom_point()	Bodový graf (scatter)	Vytvára bodový graf, kde každý bod reprezentuje dvojicu hodnôt (x, y).	geom_point()
geom_line()	Čiarový graf	Vytvára čiarový graf, ktorý spája body čiarami, zvyčajne sa používa na zobrazenie trendov.	geom_line()
geom_histogram()	Histogram	Vytvára histogram, ktorý ukazuje rozdelenie jednej kvantitatívnej premennej.	geom_histogram(binwidth = 0.1)
geom_boxplot()	Krabicový graf (box)	Vytvára boxplot (krabicový graf), ktorý ukazuje rozdelenie dát pomocou kvartilov a extrémnych hodnôt.	geom_boxplot()
geom_bar()	Stĺpcový graf (bar)	Vytvára stĺpcový graf, zobrazuje rozdelenie údajov alebo frekvenciu kategórií.	geom_bar(stat = "identity")
aes()	Estetické vlastnosti	Určuje, ako budú premenné mapované na estetické vlastnosti grafu (napr. osy, farby).	aes(x = var1, y = var2, color = var3)
facet_wrap()	Rozdelenie grafu	Rozdelí graf do viacerých panelov na základe hodnoty kategórií.	facet_wrap(~category)
labs()	Popisy grafu	Slúži na pridanie alebo úpravu názvov grafu, osí, popisov.	labs(title = "Graf", x = "X-osa", y = "Y-osa")

Tabuľka č. 8: Základné funkcie Ggplot2

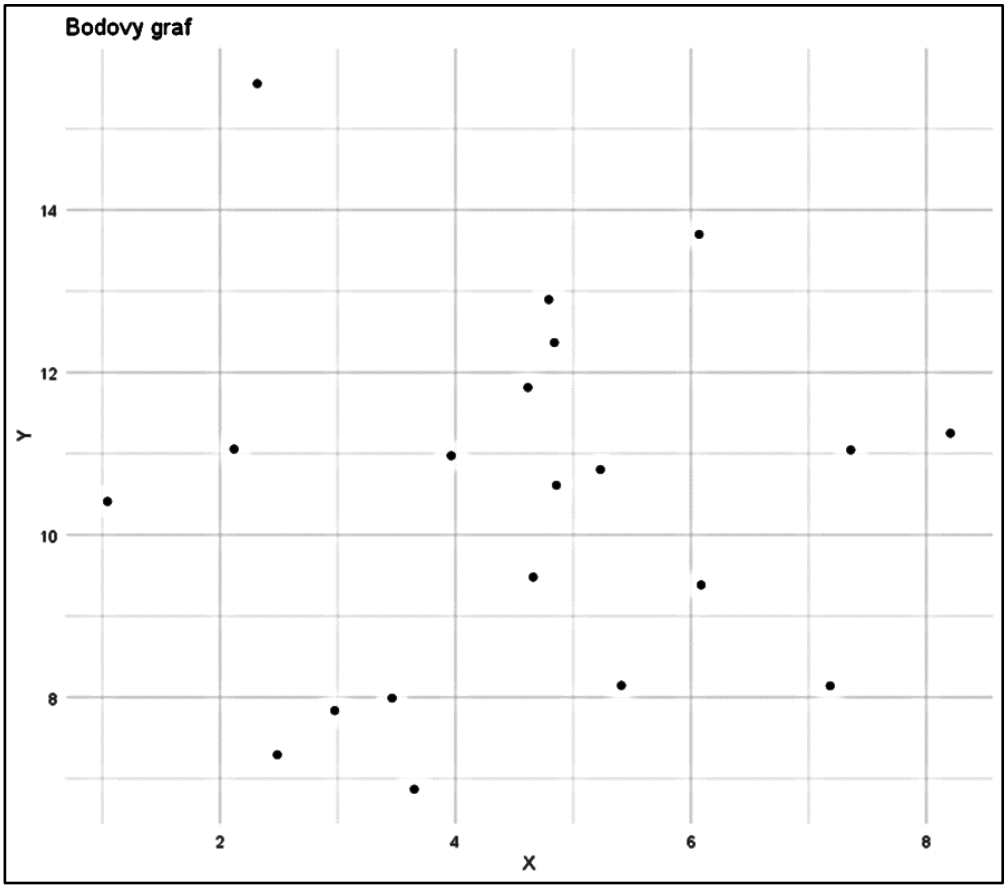
(Zdroj: Vlastné spracovanie podľa: [21])

```r
x <- rnorm( n: 20, mean = 5, sd = 2) # 20 náhodných čísel s priemerom 5 a smerodajnou odchýlkou 2
y <- rnorm( n: 20, mean = 10, sd = 3) # 20 náhodných čísel s priemerom 10 a smerodajnou odchýlkou 3
df <- data.frame(x, y)

library(ggplot2)
ggplot(df, aes(x = x, y = y)) + # Vytvorenie grafu
  geom_point() + # Bodový graf
  labs(title = "Bodovy graf", x = "X", y = "Y") + # Názvy osí
  theme_minimal() # Minimalistický dizajn
```

Obrázok č. 6: Implementácia Ggplot2

(Zdroj: vlastné spracovanie podľa: [21])



Graf č. 3: Bodový graf vytvorený pomocou ggplot2

(Zdroj: vlastné spracovanie podľa: [21])

1.3. Výhody a nevýhody jazyka R v porovnaní s Pythonom

Výhody:

1. **Silná podpora pre štatistiku a dátovú analýzu:**

   Tento jazyk bol od začiatku navrhnutý primárne pre účely štatistického spracovania a vizualizácie údajov. Disponuje množstvom vstavaných štatistických funkcií a pokročilých analytických metód, ktoré sú v Pythone dostupné len prostredníctvom externých knižníc. Vyznačuje sa tiež rozsiahlym ekosystémom balíkov špecificky zameraných na komplexné štatistické modelovanie. [2]

2. **Vysoký počet štatistických knižníc:**

   Dostupné balíky pokrývajú široké spektrum analytických potrieb – od vizualizácie (*ggplot2*), cez manipuláciu s dátami (*dplyr*) až po tvorbu interaktívnych aplikácií (*shiny*). Táto infraštruktúra je výnimočne dobre optimalizovaná práve pre štatistické a analytické úlohy. [6]

3. **Integrovaná podpora pre vizualizáciu dát:**

   Medzi najvýraznejšie prednosti patrí vizualizácia údajov, najmä vďaka knižnici *ggplot2*, ktorá je odbornou komunitou považovaná za jeden z najvyspelejších nástrojov svojho druhu. Okrem nej sú k dispozícii aj ďalšie nástroje, ako *lattice* či *plotly*, ktoré podporujú tvorbu sofistikovaných grafov. [33]

4. **Silná komunita a podpora pre akademické a výskumné prostredie:**

   R je obľúbený v akademických a výskumných kruhoch, najmä v oblastiach ako je bioštatistika, ekonometria, ekonómia, a iné oblasti aplikovanej matematiky a štatistiky. Mnoho štatistických modelov a techník, ktoré sú v R implementované, sa používajú vo vedeckých publikáciách. [2, 10]

5. **Špecializované knižnice pre štatistiku a analýzu dát:**

   Syntax je navrhnutá s dôrazom na jednoduchosť pri vykonávaní štatistických analýz. Základné testy alebo modely je možné realizovať s minimálnym objemom kódu. [2, 10]

Nevýhody:

1. **Ťažšia integrácia do produkčných systémov:**

   Na rozdiel od Pythonu, ktorý je známy svojou schopnosťou integrovať sa do produkčných systémov, R nie je až tak flexibilné v prípade potreby implementácie modelov do širších aplikácií. Python má silnejšiu podporu pre webové aplikácie, databázy, API a automatizáciu. [14, 29]

2. **Strmšia krivka učenia:**

   Pre začiatočníkov je lepšia voľba Python, jeho syntax je intuitívnejšia a čitateľnejšia. Zatiaľ čo R je špecializované na štatistiku a vyžaduje pochopenie jeho unikátneho spôsobu práce s dátami, Python má jednoduchšiu logiku, ktorá sa dá ľahšie osvojiť aj pre ľudí bez programátorského zázemia. [2]

3. **Menej univerzálny jazyk:**

   Hoci je tento jazyk výnimočne silný v oblasti štatistiky a vizualizácie, jeho využiteľnosť mimo tejto domény je obmedzená. V oblastiach ako vývoj webových aplikácií, spracovanie rozsiahlych dát alebo pokročilé modely strojového učenia, je oproti Pythonu menej flexibilný a rozšírený. [2]

1.4. Jednoduchá lineárna regresia

Lineárna regresia patrí k najjednoduchším typom regresnej analýzy a patrí k najpoužívanejším v analytickej praxi. Jej cieľom je nájsť takú funkciu, ktorá čo najpresnejšie popíše lineárnu závislosť medzi vysvetľovanou (závislou) premennou a vysvetľujúcimi (závislými) premmenými. Výsledná funkcia potom pomáha k pochopeniu vzťahov medzi rôznymi faktormi vplývajúcimi na skúmaný jav, napr. vplyv rozmeru bytu na cenu bytu, ako aj predikciu výsledkov v budúcnosti. [13]

V súčasnosti sa lineárna regresia aplikuje v rôznych oblastiach:

- Ekonomika a financie: predikcia cien akcií na základe makroekonomických ukazovateľov, analýza dopytu a ponuky,
- Marketing: meranie vplyvu reklám na predaj produktov/služieb,
- Športová analýza: Predikcia výkonu športovcov na základe tréningových údajov,
- Výroba a priemysel: Optimalizácia procesov na základe produkčných parametrov,
- Sociálne vedy: Skúmanie vplyvu vzdelania na výšku príjmu [3, 12].

1.4.1. Definícia a matematická formulácia

Jednoduchá lineárna regresia je štatistický model, ktorý opisuje vzťah medzi jednou nezávislou premennou (X) a závislou premennou (Y) pomocou lineárnej funkcie, ktorá je matematicky vyjadrená nasledovne:

$$y = \beta_0 + \beta_1 x_i + \varepsilon_i, \text{ pričom i = 1, 2 ... n}$$

Kde:

- y - závislá (odhadovaná) premenná,
- x_i – i-ta hodnota nezávislej premennej,
- β_0, β_1 - neznáme parametre regresného modelu,
- ε_i náhodná chyba i-teho pozorovania,
- n – počet pozorovaní [13].

Základná hypotéza (test významnosti regresného koeficientu β_1):

- Nulová hypotéza (H_0) : Neexistuje vzťah medzi X a Y (regresný koeficient β_1 je nulový)

$$H_0: \beta_1 = 0$$

Znamená, že nezávislá premenná X nemá žiadny štatistický významný vplyv na závislú premennú Y)

- Alternatívna hypotéza (H_1): Existuje vzťah medzi X a Y (regresný koeficient β_1 je nenulový)

$$H_1: \beta_1 \neq 0$$

Znamená, že nezávislá premenná ma štatisticky významný vplyv na Y [13].

Predpoklady JLR

Pre správnosť modelu musia byť splnené nasledujúce predpoklady:

1. Linearita modelu – predpokladá sa, že vzťah medzi nezávislou a závislou premennou je lineárny, čo znamená, že zmena X spôsobí priamo úmernú zmeni v Y,
2. Nezávislosť pozorovaní – hodnota jedného pozorovania by nemala ovplyvňovať hodnoty iných pozorovaní,
3. Homoskedasticita – variabilita reziduí (chýb) sa so zmenou vysvetľujúcej premennej nemení. Tento predpoklad však nie je reálne splniť, ak sa variabilita vysvetľovanej premennej so zmenami vysvetľujúcej premennej mení. V takomto prípade hovoríme o heteroskedasticitkom modeli, ktorý môže viesť k nespoľahlivým odhadom parametrov,
4. Normalita reziduí – rozdelenie reziduí by malo byť približne normálne, čo znamená, že by sme mali na grafe vidieť normálne rozdelenie s hodnotou rozptylu blízkou k nule [13].

Metóda najmenších štvorcov (OLS)

Najčastejšie používanou metódou na odhad parametrov β_0 a β_1 je metóda najmenších štvorcov (Ordinary Least Squares - OLS). Jej cieľom je minimalizovať sumu štvorcov medzi skutočnými a odhadovanými hodnotami Y a teda získať také odhady parametrov β_0 a β_1, aby sa skutočné hodnoty závislej premennej čo najmenej odchyľovali od vyrovnávajúcej regresnej priamky. [13]

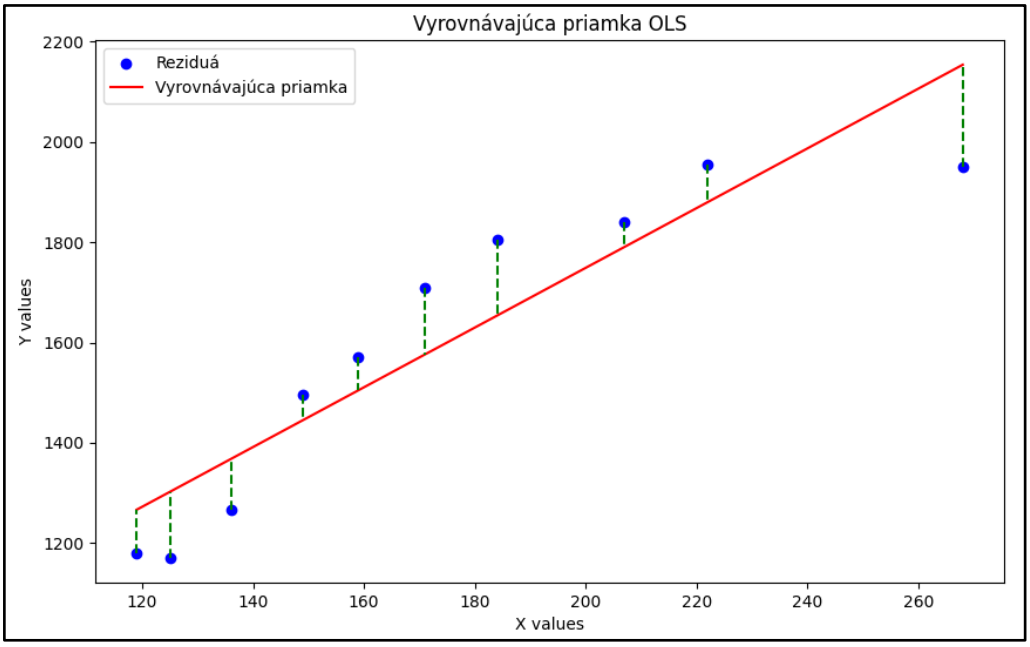
$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} e_i^2 \rightarrow min$$

Vyrovnávajúca regresná priamka:

$$\hat{y}_i = b_0 + b_1 x_i$$

Kde:

- $\hat{y}_i$ – i-ta vyrovnaná (odhadnutá) hodnota závislej premennej Y,
- x_i – i-ta hodnota nezávislej premennej,
- b_0 – bodový odhad parametra β_0,
- b_1 – bodový odhad parametra β_1[13].



Graf č. 4: Vyrovnávajúca priamka OLS

(*Zdroj: vlastné spracovanie)*

Interpretovanie výstupov

Výstup regresnej analýzy obsahuje:

- Odhady parametrov β_0, β_1 – určujú, ako sa hodnota závislej premennej mení pri zmene nezávislej premennej
- P-value – overenie významnosti modelu aj významnosti nezávislej premennej ,
- r^2 (koeficient determinácie) – dosahuje hodnoty $\{0, 1\}$, vyjadruje, aká časť variability je vysvetlená modelom, zvyšná je vysvetlená činiteľmi nezaradenými do modelu a náhodnými vplyvmi,
- F-stat – overenie celkovej významnosti modelu [13].

1.5. Analýza rozptylu

Analýza rozptylu, známa aj ako ANOVA (z angl**. An**alysis **o**f **Va**riance) sa v praxi používa ako samostatná technika alebo postup, ktorý umožňuje analyzovať zdroje variability v štatistických modeloch. Posudzuje vplyv jednej alebo niekoľkých nezávislých premenných (nazývaných faktory) na kvantitatívnu závislú premennú. Jej podstatou je zistiť, či existujú štatisticky významné rozdiely medzi strednými hodnotami závislej premennej v súboroch, ktoré boli vytvorené pre jednotlivé úrovne (kategórie, obmeny) nezávislej premennej. Existuje aj MANOVA (Multiple Analysis of Variance), ktorá skúma súčasné pôsobenie jedného alebo viacerých faktorov na dve alebo viac nezávislých premenných. [13]

V tejto práci sa zameriame na jednorozmernú jednofaktorovú analýzu rozptylu, ktorá skúma vplyv iba jedného faktora na závislú premennú. Je zovšeobecnením t-testu, ktorým sa overuje zhoda stredných hodnôt dvoch základných súborov $\mu_1 = \mu_2$.[13]

Základným princípom ANOVY je rozklad variability v dátach na 2 skupiny:

- Variabilita medzi skupinami – rozdiely medzi priemermi jednotlivých skupín,
- Variabilita vnútri skupín – rozdiely medzi jednotlivými hodnotami v rámci skupín. [13]

Matematicky testuje hypotézu pomocou porovnania týchto 2 zložiek variability:

$$F = \frac{MSA}{MSE}$$

Kde:

- F-hodnota – testovacia štatistika, ktorá sa porovnáva s kritickou hodnotou F-rozdelenia,
- MSA – priemerný rozptyl medzi skupinami,
- MSE – priemerný rozptyl v rámci skupín [13].

Základná hypotéza analýzy rozptylu:

- Nulová hypotéza (H_0): Medzi skupinami neexistuje žiadny rozdiel v priemeroch,

$$\mu_1 = \mu_2 = \mu_3 = \mu_k$$

- Alternatívna hypotéza (H_1): Aspoň jeden priemer je rozdielny [13].

Analýza rozptylu sa využíva vo viacerých oblastiach, ako napríklad:

- Medicína a farmácia: Porovnanie účinnosti rôznych liekov na rovnaké ochorenie,
- Sociálne vedy: Vplyv rôznych vyučovacích metód na výsledky študentov,
- Ekonomika: Analýza predajných výkonov rôznych marketingových stratégií,
- Poľnohospodárstvo: Vplyv rôznych typov zavlažovania na rast rastlín [8].

Predpoklady ANOVA:

1. Nezávislosť pozorovaní – hodnoty by nemali byť medzi sebou závislé,
2. Homoskedasticita - zhodnosť rozptylov náhodnej premennej vo všetkých porovnánavných základných súboroch,
3. Normalita - normálne rozdelenie pravdepodobnosti náhodnej premennej vo všetkých porovnávaniach [13].

Výstup ANOVA analýzy obsahuje:

- **F-hodnotu** – čím je vyššia, tým väčšie sú rozdiely medzi skupinami,
- **P-hodnotu** – ak je menšia ako 0.05 (resp. 1 – α), zamietame nulovú a prijímame alternatívnu hypotézu, a to že aspoň jedna skupina sa významne líši [13].

1.6. Vizualizácia výsledkov v Pythone a R

Vizualizácia je neoddeliteľnou súčasťou štatistického modelovania a dátovej analýzy, pomáha pochopiť vzťahy medzi premennými, analyzovať výstupy modelov a overiť predpoklady rôznych metód. V našom prípade budeme skúmať možnosti implementácie grafických výstupov a pre lineárnu regresiu a analýzu rozptylu (ANOVA). [13]

1.6.1. Vizualizácia lineárnej regresie

Vizualizácia lineárnej regresie pomáha pochopiť vzťah medzi závislou a nezávislou premennou, ako aj identifikovať potenciálne problémy, napr. heteroskedasticitu, nenormálne rozdelenie reziduí. [13]

Scatter plot (bodový graf) – zobrazuje vzťah medzi nezávislou premennou (X) a závislou premennou (Y) spolu s regresnou priamkou

- V Pythone: knižnica Seaborn obsahuje funkciu sns.regplot(), ktorá automaticky vykreslí bodový graf aj s regresnou priamkou [27].
- V R knižnica ggplot2 umožnuje vytvoriť graf pomocou geom_point(), pričom regresnú priamku možno pridať pomocou geom_smooth(method = "lm") [21]

Residual plot (graf rezíduí) – znázorňuje rozdiel medzi odhadovanými a skutočnými hodnotami (rozdiel rezíduí), pomáha odhaliť heteroskedasticitu

- V Pythone: Seaborn ponúka funkciu sns.residplot() [27].
- V R: Diagnostické grafy regresie, vrátane grafu reziduí, sú dostupné cez funkciu plot(lm_model), konkrétne which = 1 parameter [22, 33].

Histogram alebo Q-Q plot reziduí – slúži na kontrolu normálneho rozdelenia rezíduí, v Q-Q plote by mali body ležať na diagonálnej čiare

- V Pythone: statsmodels.graphics.gofplots.qqplot() vizualizuje Q-Q plot, funkcia sns.histplot() histogram [27, 28].
- V R: Q-Q plot je dostupný v base R v rámci funkcie plot(lm_model), konkrétne which = 2 parameter [22, 33].

1.6.2. Vizualizácia analýzy rozptylu

Boxplot – porovnáva rozloženie hodnôt medzi viacerými skupinami

- V Pythone: funkcia sns.boxplot() umožňuje porovnávať viaceré kategórie vedľa seba [27].
- V R: ggplot2 vytvoriť boxploty pomocou geom_boxplot() [21].

Bar plot (stĺpcový graf) s priemermi a intervalmi spoľahlivosti – vizualizuje rozdiely medzi skupinami

- V Pythone: funkcia sns.barplot() zobrazuje priemery skupín a pomocou argumentu „ci“ sa pridávajú intervaly spoľahlivosti[27].
- V R: graf je možné vytvoriť pomocou geom_bar(stat=“summary“, fun = “mean“), v kombinácii s geom_errorbar() pre zobrazenie intervalov spoľahlivosti [21]

Residual plot (graf rezíduí) – rovnako ako pri LR znázorňuje rozdiel medzi odhanovanými a skutočnými hodnotami

Typ Grafu	**Python (Matplotlib, Seaborn, Statsmodels)**	**R (ggplot2, base R)**
Scatter plot s regresnou priamkou (LR)	sns.regplot()	geom_point() + geom_smooth(method="lm")
Reziduálny graf (LR)	sns.residplot()	plot(lm_model, which = 1)
Histogram reziduí (LR)	plt.hist(residuals)	hist(residuals)
Q-Q plot reziduí (LR)	statsmodels.graphics.gofplots.qqplot(residuals, fit=True)	plot(lm_model, which = 2)
Boxplot (ANOVA)	sns.boxplot()	geom_boxplot()
Bar plot (ANOVA)	sns.barplot()	geom_bar(stat="summary", fun="mean")
Residual plot (ANOVA)	sns.residplot()	plot(aov_model, which = 1)

Tabuľka č. 9: Základné funkcie grafov

(Zdroj: Vlastné spracovanie podľa: [21, 22, 23, 27, 28])

2. Cieľ práce, metodika práce a metódy skúmania

V praktickej časti porovnáme implementáciu lineárnej regresie a ANOVY v Pythone a Rku z nasledovných hľadísk:

- Náročnosť implementácie
- Výkon a rýchlosť výpočtov
- Presnosť výsledkov
- Vizualizácia modelov

2.1. Postup testovania

Na overenie teoretických poznatkov budú vykonané nasledovné postupy:

1. Porovnanie implementácie lineárnej regresie a ANOVY v Pythone a R
   - Použitie rovnakých datasetov v oboch jazykoch.
   - Implementácia jednoduchej lineárnej regresie (lm(), statsmodels).
   - Implementácia analýzy rozptylu (aov(), scipy.stats.f_oneway()).
   - V Pythone aj vlastná implementácia riešenia bez použitia knižníc Statsmodels, SciPy a Scikit.
2. Testovanie výkonu a rýchlosti výpočtov
   - Meranie času vykonania regresie a ANOVY na rovnakých datasetoch.
   - Testovanie na rôznych počítačoch s rôznym OS a hardvérom.
3. Porovnanie výsledkov a vizualizácia dát
   - Overenie ekvivalencie výstupov z oboch jazykov.
   - Vizualizácia výsledkov pomocou ggplot2 a Seaborn s Matplotlib.

Datasety budú vytvorené v rôznych veľkostiach s počtom pozorovaní 100, 1 000, 10 000 a 100 000 s využitím náhodného generátora z knižnice random v jazyku Python.

3. Výsledky práce

Táto kapitola sa zameriava na implementáciu lineárnej regresie a analýzy rozptylu v programovacích jazykoch R a Python, ako aj na následné porovnanie týchto implementácií z viacerých hľadísk. V úvode je opísaný proces prípravy vývojového prostredia vrátane inštalácie potrebného softvéru, knižníc a konfigurácie nástrojov pre oba jazyky. Opísané sú metódy generovania údajov a formát exportu. Ďalej sa kapitola venuje spôsobu merania výpočtovej výkonnosti, vrátane postupu zaznamenávania času vykonania skriptov a porovnania rýchlosti na rôznych typoch výpočtovej techniky.

3.1. Príprava vývojového prostredia

Na začiatok je potrebné nainštalovať Python. V čase písania tejto práce bola najnovšia verzia 3.13.1, ktorú je možné stiahnuť zo stránky https://www.python.org/downloads/release/python-3131/. Po stiahnutí Pythonu je nutné nainštalovať aj R, konkrétne verziu 4.4.3: https://cran.r-project.org/bin/windows/base/old/4.4.3/.

Ako vývojové prostredie bude použitý **PyCharm** od spoločnosti Jetbrains. Toto IDE (Integrated Development Enviroment) poskytuje podporu pre rôzne pluginy, ktoré budú v projekte využívané, vrátane **R extension**, ktorý zabezpečuje používanie oboch jazykov v rovnakom prostredí.

Pre zabezpečenie izolácie projektu bolo použité virtuálne prostredie. Aj keď nie je nevyhnutné, jeho použitím sa udržiavajú konzistentné verzie knižníc a zamedzuje konfliktom medzi rôznymi projektami. PyCharm ponúka možnosť jeho vytvorenia pri zakladaní alebo importovaní projektu.

Po vytvorení projektu v PyCharm je potrebné nainštalovať plugin R Extension for IntelliJ, ktorý poskytuje plnú podporu pre prácu s jazykom R. Tento plugin umožňuje plynulú integráciu jazyka R do vývojového prostredia a napodobňuje funkcionalitu natívneho prostredia R – RStudio. Taktiež je nutné nastaviť cesty k interpreterom oboch jazykov (napr. C:\Program Files\R\R-4.4.3\bin\R.exe) v projekte.

Finálna verzia projektu sa nachádza na GitHube, čo je online platforma na zdieľanie a ukladanie kódu, ktorá využíva systém Git. Odkaz na repozitár je

https://github.com/BenjaminKolarik/bc. Obsahuje súbor requirements.txt, ktorý definuje všetky potrebné knižnice pre Python spolu s ich verziami - po importovaní projektu sa automaticky stiahnu. Taktiež sa v ňom nachádza R skript install_packages.R, v ktorom je zoznam(vektor) potrebných knižníc.

```
required_packages <- c("readxl", "dplyr", "tidyr", "ggplot2", "openxlsx", "car", "lmtest")

is_installed <- function(pkg) {
  is.element(pkg, installed.packages()[, "Package"])
}


for (pkg in required_packages) {
  if (!is_installed(pkg)) {
    install.packages(pkg, dependencies = TRUE)
  }
}
```

Tento skript postupne iteruje cez každý prvok zoznamu knižníc, a ak daná knižnica nie je nainštalovaná, automaticky ju doinštaluje.

3.2. Generovanie dát

Keďže cieľom tejto práce nie je dospieť k reálnym štatistickým záverom na základe konkrétnych dát, ale porovnať efektívnosť a implementáciu metód, boli zvolené umelo vytvorené dáta. Táto voľba umožňuje kontrolu nad premennými, rozsahom dát a ich štruktúrou. Pre oba druhy štatistického modelovania boli vytvorené Python skripty.

```
import random as rd
import pandas as pd
import os
import numpy as np

def generate_data(num_samples, excel_file, nan_probability=0.02):
    random_numbers_x = []
    for _ in range(num_samples):
        if rd.random() < nan_probability:
            random_numbers_x.append(np.nan)
        else:
            random_numbers_x.append(rd.randint(1, 100))

    random_numbers_y = []
    for x in random_numbers_x:
        if pd.isna(x) or rd.random() < nan_probability:
            random_numbers_y.append(np.nan)
        else:
            random_numbers_y.append(x + 10 + rd.randint(0, 15))

    x_nan_count = sum(pd.isna(x) for x in random_numbers_x)
    y_nan_count = sum(pd.isna(y) for y in random_numbers_y)
    total_nan_count = x_nan_count + y_nan_count
```

```
    print(f"Dataset {excel_file}: NaN count - x: {x_nan_count} ({x_nan_count /
num_samples:.2%}), "           f"y: {y_nan_count} ({y_nan_count / num_samples:.2%}), "
f"total: {total_nan_count} ({total_nan_count / (num_samples * 2):.2%})")

    data = {'y': random_numbers_y, 'x': random_numbers_x}
    excel_writer(excel_file, data)


def excel_writer(excel_file, data, output_dir = '../../input/LR'):
    df = pd.DataFrame(data)

    os.makedirs(output_dir, exist_ok=True)
    file_path = os.path.join(output_dir, excel_file)

    df.to_excel(file_path, index=False)
    print(f"Data written to {file_path}")
```

Táto funkcia berie ako vstupné parametre počet pozorovaní, názov Excel súboru a šancu na výskyt nulovej hodnoty – tá je vopred zadefinovaná s hodnotou 0,02. Pre každé pozorovanie sa náhodne vyberá cele číslo medzi 1 a 100 ako hodnota pre premennú X. Funkcia rd.random() generuje náhodnú hodnotu od 0 po 1. Ak je táto hodnota menšia ako 0,02, do zoznamu sa nezapíše nič. Rovnakým princípom sa generujú aj hodnoty premennej Y, avšak s podmienkou naviac – ak hodnota x je žiadna, bude hodnota y taktiež nič. Ak ani jedna podmienka nie je splnená, premenná Y sa počíta ako premenná x + 10 + náhodné číslo od 0 do 15 – toto nám zabezpečí, že Y bude vždy vačší ako X. Posledným krokom je zapísanie dát do 2 Excelových stĺpcov – X a Y.

```
import random as rd
import pandas as pd
import os
import numpy as np

def generate_anova_data(num_samples_per_group, num_groups, excel_file, nan_probability=0.02):

    data = {'group': [], 'value': []}

    for group in range(1, num_groups + 1):
        group_mean = 50 + group * 5

        for _ in range(num_samples_per_group):
            data['group'].append(f"Group_{group}")

            if rd.random() < nan_probability:
                data['value'].append(np.nan)
            else:
                value = round(group_mean + rd.normalvariate(0, 5))
                data['value'].append(value)

    nan_count = sum(pd.isna(v) for v in data['value'])
    total_samples = num_samples_per_group * num_groups

    print(f"Dataset {excel_file}: NaN count - {nan_count} ({nan_count / total_samples:.2%})")

    excel_writer(excel_file, data)

def excel_writer(excel_file, data, output_dir='../../input/ANOVA'):
    df = pd.DataFrame(data)
```

```
    os.makedirs(output_dir, exist_ok=True)
    file_path = os.path.join(output_dir, excel_file)

    df.to_excel(file_path, index=False)
    print(f"Data written to {file_path}")

excel_file="ANOVA_extra_large.xlsx")
```

Funkcia na generovanie dát pre ANOVU funguje na veľmi podobnom princípe ako funkcia na generovanie dát pre lineárnu regresiu. Ako vstupné parametre berie počet pozorovaní na jednu skupinu, počet skupín a názov Excel súboru, do ktorého sa budú uložené výsledky. Na dočasné uchovávanie dát vytvorí slovník s kľúčmi group a value. Pre každú skupinu sa vytvorí skupinový priemer, ku ktorému sa postupne pridáva hodnota 5. Následne sa pre každú hodnotu v skupine vygeneruje hodnota, ktorá je súčtom skupinového priemeru a náhodného čísla zo štandardného normálneho rozdelenia s priemerom 0 a štandardnou odchýlkou 5. S pravdepodobnosťou 0,02 sa však hodnota nemusí zapísať a zostane prázdna. Na záver sa celý slovník uloží do dvoch Excelových stĺpcov: Group a Value.

Funkcie sa dajú zavolať nasledovne:

- LR: generate_data(100, excel_file="LR_100.xlsx")
- ANOVA: generate_anova_data(num_samples_per_group=1000, num_groups=10, excel_file="ANOVA_large.xlsx")

3.3. Implementácia Lineárnej regresie

3.3.1. Python

Viacero knižníc v Pythone umožňuje výpočet lineárnej regresie, a to konkrétne Statsmodels, SciPy a Scikit-learn. V tejto implementácii sa využijú všetky knižnice, ako aj vlastné riešenie bez ich použitia. Kódy sa budú líšiť predovšetkým funkciou perform_regression, zatiaľ čo ostatné pomocné funkcie budú prevažne rovnaké, s výnimkou odlišných parametrov, ktoré je potrebné prispôsobiť špecifikám jednotlivých knižníc.

Najskôr si predstavíme pomocné funkcie:

```
def data_load(file_path):

    data = pd.read_excel(file_path)
    data = data[['y', 'x']].dropna()
    return data

def evaluate_model(model, x, y):

    y_pred = model.predict(x)
    mse = mean_squared_error(y, y_pred)
    print(f"Model Evaluation:\nR-squared: {model.rsquared:.4f}\nAdjusted R-squared:
{model.rsquared_adj:.4f}\nMSE: {mse:.4f}")
```

Funkcia data_load zoberie cestu k Excel súboru, vymaže všetky riadky, v ktorých chýba hodnota a vráti data vo forbe typu DataFrame. Funkcia Evaluate_model vypočíta odhadované hodnoty modelu, MSE a vypíše do konzoly koeficient determinácie, upravený koeficient determinácie a MSE.

```
def test_assumptions(data, slope, intercept):
    x = data['x']
    y = data['y']
    y_pred = intercept + slope * x     residuals = y - y_pred

    bp_test = het_breuschpagan(residuals, sm.add_constant(x))
    print(f"\nBreusch-Pagan test: p-value = {bp_test[1]:.4f}")

    shapiro_test = shapiro(residuals)
    print(f"\n\n\nShapiro-Wilk test: p-value = {shapiro_test.pvalue:.4f}")

    dw_test = durbin_watson(residuals)
    print(f"\n\n\nDurbin-Watson test: statistic = {dw_test:.4f}")
```

Test_assumptions vykoná testy na kontrolu predpokladov lineárnej regresie, ako sú normalita reziduálov, linearita, homoskedasticita a nezávislosť reziduálov. Výsledky týchto testov sú následne vypísané do konzoly.

```python
def plot_regression(data, model, output_dir):  1 usage  benjamin_kolarik
    x = data[['x']]
    y_pred = model.predict(x)

    plt.figure(figsize=(10, 6))
    sns.regplot(x=data['x'], y=data['y'], line_kws={'color': 'red'})
    plt.title('Linear Regression')
    plt.savefig(output_dir + 'linear_regression.png')
    plt.close()

    residuals = data['y'] - y_pred
    plt.figure(figsize=(10, 6))
    sm.qqplot(residuals, line='s')
    plt.title('Q-Q Plot of Residuals')
    plt.savefig(output_dir + 'qq_residuals.png')
    plt.close()

    plt.figure(figsize=(10, 6))
    sns.residplot(x=data['x'], y=residuals, lowess=True, line_kws={'color': 'red', 'lw': 1})
    plt.xlabel('x')
    plt.ylabel('Residuals')
    plt.title('Residual Plot')
    plt.savefig(output_dir + 'residual_plot.png')
    plt.close()
```

Obrázok č. 7: Pomocná funkcia plot_regression

(Zdroj: vlastné spracovanie)

Funkcia plot_regression vykreslí tri rôzne diagnostické grafy s využitím knižníc Matplotlib a Seaborn. Tieto grafy zahŕňajú scatter plot s regresnou priamkou, Q-Q plot reziduí a reziduálny graf. Všetky grafy sa následne uložia do priečinka, ktorý je zadaný používateľom.

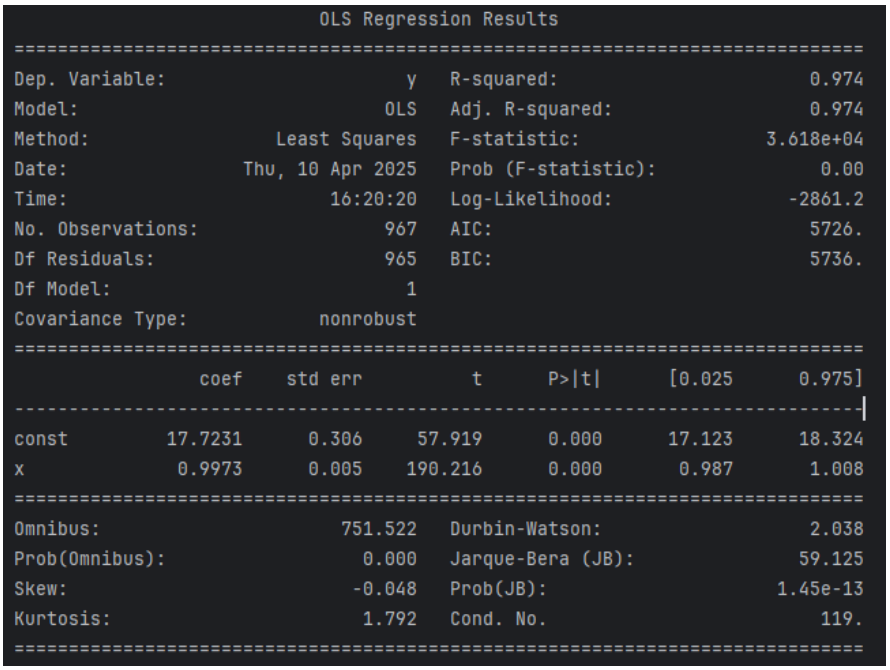
```python
def perform_regression(data):

    x = data['x']
    y = data['y']

    x = sm.add_constant(x))
    model = sm.OLS(y, x).fit()

    return model
```

Na výpočet lineárnej regresie pomocou Statsmodels stačia efektívne 4 riadky kódu. To zahŕňa určenie premenných X a Y, pridanie konštatny 1 do zoznamu x a následný výpočet parametrov modelu pomocou metódy najmenších štvorcov.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.974
Model:                            OLS   Adj. R-squared:                  0.974
Method:                 Least Squares   F-statistic:                 3.618e+04
Date:                Thu, 10 Apr 2025   Prob (F-statistic):               0.00
Time:                        16:20:20   Log-Likelihood:                -2861.2
No. Observations:                 967   AIC:                             5726.
Df Residuals:                     965   BIC:                             5736.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         17.7231      0.306     57.919      0.000      17.123      18.324
x              0.9973      0.005    190.216      0.000       0.987       1.008
==============================================================================
Omnibus:                      751.522   Durbin-Watson:                   2.038
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               59.125
Skew:                          -0.048   Prob(JB):                     1.45e-13
Kurtosis:                       1.792   Cond. No.                         119.
==============================================================================
```

Obrázok č. 8: Výstup lineárnej regresie pomocou Statsmodels

(*Zdroj: vlastné spracovanie)*

Výstup z knižnice Statsmodels je komplexný a zahŕňa všetky kľúčové hodnoty, ako sú F-štatistika, odhadnuté parametre regresnej priamky, intervalové odhady parametrov, p-hodnoty, ale aj štatistiky potrebné na detekciu prítomnosti autokorelácie.

```python
def perform_regression(data):
    x = data['x']
    y = data['y']
    slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)

    return slope, intercept, r_value, p_value, std_err
```

Implementácia pomocou knižnice Scipy je na tom horšie. Funkcia linregress vracia len parametre regresnej priamky, koeficient korelácie, p-value a štandardnú odchýlku, teda omnoho menej údajov, ako Statsmodels.

```python
def perform_regression(data):
    x = data[['x']]
    y = data['y']
    model = LinearRegression().fit(x, y)
    return model
```

Knižnica scikit ponúka najmenej možností zo všetkých knižníc, vracia iba hodnoty parametrov regresnej priamky. Ponúka ešte dodatočné metódy výpočtov, ako koeficient determinácie a odhadované hodnoty.

Ďalšou možnosťou je implementácia vlastného riešenia, ktoré umožňuje presne definovať požadované výpočty a údaje. V tomto prístupe sa vypočítajú všetky medzivýpočty, vrátane tých, ktoré nie sú priamo potrebné pre výstup modelu, pričom všetky tieto hodnoty sa ukladajú do Excelovej tabuľky. Tento prístup poskytuje flexibilitu v prispôsobovaní výstupov podľa potreby, vrátane výstupu na konzolu.

Okrem toho sa ponúka voľba medzi dvoma programovacími paradigmami: procedurálnou a objektovo-orientovanou paradigmou. Pre každú sme vytvorili samostatné implementácie.

OOP môže byť na začiatku vnímaná ako zložitejšia než procedurálny prístup, keďže vyžaduje pochopenie abstraktnejších konceptov, ako sú triedy, objekty a dedičnosť. Na rozdiel od priameho sekvenčného toku procedurálneho programovania, OOP kladie dôraz na štruktúru, opätovné použitie a modelovanie reality. V ďalšej analýze preukážeme, ktorý z týchto prístupov je v danom prípade efektívnejší a rýchlejší.

```
Breusch-Pagan test: p-value = 0.7559
Shapiro-Wilk test: p-value = 0.0000
Durbin-Watson test: statistic = 2.0384

Linear Regression Summary:
Equation: y = 17.7231 + 0.9973x
R-squared: 0.9740
P-value: 0.0000
Significant: Yes
RMSE: 4.6690
MAE: 4.0445
F-statistic: 36182.1285

Data exported to ../../output/LR/LR_base/excel\linear_regression_results.xlsx
```

Obrázok č. 9: Výstup vlastnej implemtácie LR

(Zdroj: vlastné spracovanie)

3.3.2. R

Narozdiel od Pythonu, jazyk R ponúka možnost lineárnej regresie už vo svojom základnom balíku Base R, čo znamená, že knižnice tretích strán už nemuseli túto metódu implementovať znovu.

```r
data <- read_excel("input/LR/LR_100.xlsx")

data_clean <- data %>% drop_na(x, y)

model <- lm(y ~ x, data = data_clean)

coefficients <- coef(model)
print("Regression Coefficients:")
print(coefficients)

summary_result <- summary(model)
print(summary_result)

r_squared <- summary_result$r.squared adj_r_squared <- summary_result$adj.r.squared
print(paste("R-squared:", r_squared))
print(paste("Adjusted R-squared:", adj_r_squared))
```

Implementácia v R je rovnako jednoduchá. Stačí importovať dáta, očistiť ich od nulových hodnôt a následne pomocou funkcie lm() vypočítať parametre modelu.

```
Call:
lm(formula = y ~ x, data = data_clean)

Residuals:
    Min      1Q  Median      3Q     Max
-7.7204 -3.6876  0.3915  4.2865  7.5444

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 17.723096   0.306000   57.92   <2e-16 ***
x            0.997270   0.005243  190.22   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.669 on 965 degrees of freedom
Multiple R-squared:  0.974, Adjusted R-squared:  0.974
F-statistic: 3.618e+04 on 1 and 965 DF,  p-value: < 2.2e-16
```

Obrázok č. 10: Výstup summary_result

(Zdroj: vlastné spracovanie)

Ako je z výstupu zrejmé, poskytuje komplexné informácie vrátane štatistickej významnosti parametrov pre rôzne hladiny významnosti, počtu stupňov voľnosti a podrobných údajov o reziduáloch.

Rovnako ako v prostredí Python, aj v R je možné kód rozšíriť o vizualizácie a testy predpokladov lineárnej regresie. Na ich využitie je však potrebné importovať knižnice ggplot2 a lmtest.

```r
regression_plot <- ggplot(data_clean, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE) +
  labs(title = "Linear Regression",
       x = "Independent Variable",
       y = "Dependent Variable") +
  theme_minimal()
ggsave(file.path(output_dir, "linear_regression.jpeg"), plot = regression_plot, width = 10, height = 6, dpi = 300)

residual_plot <- ggplot(data_clean, aes(x = model$fitted.values, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  labs(title = "Residual Plot",
       x = "Fitted Values",
       y = "Residuals") +
  theme_minimal()
ggsave(file.path(output_dir, "residual_plot.jpeg"), plot = residual_plot, width = 10, height = 6, dpi = 300)

qq_plot <- ggplot(data_clean, aes(sample = residuals)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Q-Q Plot of Residuals") +
  theme_minimal()
ggsave(file.path(output_dir, "qq_plot.jpeg"), plot = qq_plot, width = 10, height = 6, dpi = 300)

residual_histogram <- ggplot(data_clean, aes(x = residuals)) +
  geom_histogram(aes(y = after_stat(density)), bins = 30, fill = "blue", alpha = 0.7) +
  geom_density(color = "red") +
  labs(title = "Histogram of Residuals",
       x = "Residuals",
       y = "Density") +
  theme_minimal()
ggsave(file.path(output_dir, "residual_histogram.jpeg"), plot = residual_histogram, width = 10, height = 6, dpi = 300)
```

Obrázok č. 11: Implementácia grafov LR v R

(Zdroj: vlastné spracovanie)

```r
bp_test <- bptest(model)
print(bp_test)

shapiro_test <- shapiro.test(residuals)
print(shapiro_test)

dw_test <- dwtest(model)
print(dw_test)
```

3.3.3. Porovnanie LR

Zložitosť:

- Všetky použité knižnice sa vyznačovali jednoduchým použitím, pričom základné analýzy bolo možné realizovať v rámci desiatich riadkov kódu. Tento fakt výrazne uľahčuje rýchlu prvotnú implementáciu a základnú analytickú prácu.

Výstupy:

- Knižnice Base R a Statsmodels poskytujú komplexné výstupy zahŕňajúce široké spektrum detailných metrík, čo je kľúčové pre dôkladnú analýzu modelu. Naopak, knižnice SciPy a Scikit-learn ponúkajú skromnejšie výstupy, ktoré sa zameriavajú predovšetkým na základné parametre modelu a menej rozsiahle diagnostické informácie.

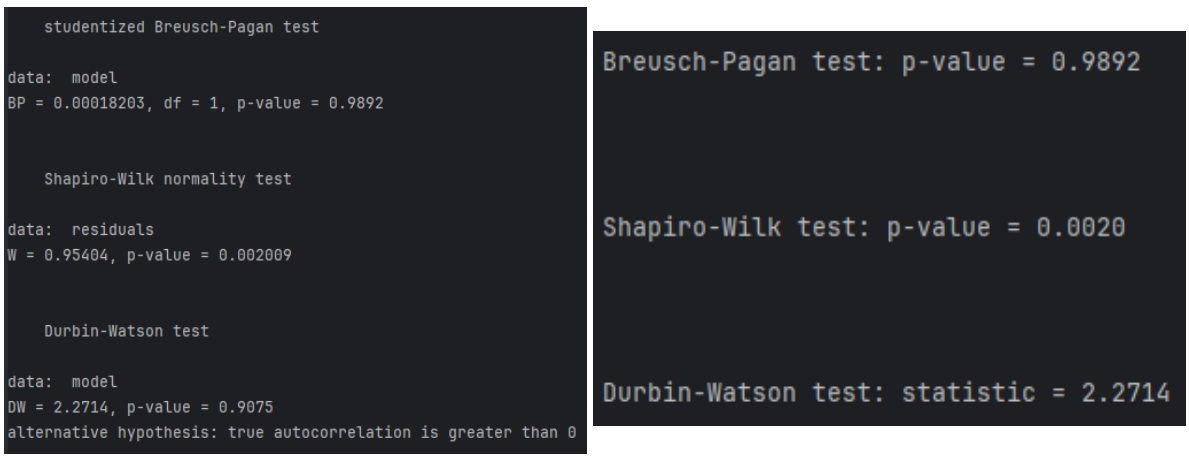
Flexibilita:

- Rozšírenie o nové funkcie alebo dodatočné výpočty je relatívne jednoduché, vyžaduje však zapojenie ďalších knižníc.

Presnosť výsledkov:

- Získané výsledky boli prakticky totožné, s výnimkou miernych odchýlok spôsobených rozdielnym zaokrúhľovaním. Tento rozdiel je však zanedbateľný a nemá vplyv na presnosť výpočtov. Koeficienty, p-hodnoty a ďalšie kľúčové štatistické ukazovatele boli konzistentné naprieč všetkými použitými prístupmi.

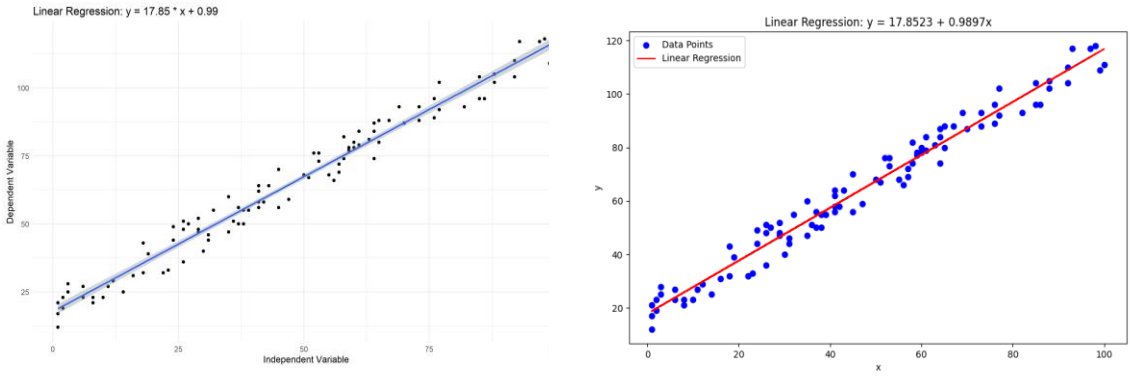
Vizualizácie:



Obrázok č. 12: Porovnanie výstupov na predpoklady modelov

(Zdroj: vlastné spracovanie)

Ako vyplýva z výstupu, prostredie R poskytuje podrobnejšie výsledky. Obsahuje nielen samotný koeficient testu, ale aj p-value, ktorá slúži na rozhodovanie o prijatí/zamietnutí nulovej hypotézy. Naopak, v prípade Pythonu Durbin-Watsonov test vracia iba hodnotu koeficientu, zatiaľ čo pri ostatných testoch je vrátená hodnota p-value.



Obrázok č. 13: Porovnanie grafov lineárnej regresie

(*Zdroj: vlastné spracovanie)*

Grafické výstupy sú takmer identické, v predvolených nastaveniach má Python (vpravo) zobrazené väčšie body v porovnaní s R, čo však predstavuje detail, ktorý je možné jednoducho upraviť. Z oboch grafov je zrejmý rozdiel medzi odhadovanými hodnotami (priamka) a skutočnými hodnotami (body). Knižnica ggplot2 v jazyku R však ponúka širšie možnosti prispôsobenia vizualizácií.

3.4. Implementácia Analýzy rozptylu

3.4.1. Python

Implementáciu ANOVY ponúkajú 2 knižnice: Statsmodels a SciPy. Rovnako ako pri LR, kód sa skladá z hlavnej funkcie perform_anova a viacerých pomocných funckii, ktoré zabezpečujú načítanie a úpravu vstupných dát, tvorbu grafov a testov na predpoklady modelu.

```
def graph(data):  1 usage  benjamin_kolarik *

    plt.figure(figsize=(10, 6))
    sns.boxplot(x='group', y='value', data=data)
    plt.title('Boxplot of Values by Group')
    plt.xlabel('Groups')
    plt.ylabel('Values')
    plt.savefig('../../output/ANOVA/ANOVA_scipy/boxplot.png')
    plt.close()

    group_means = data.groupby('group')['value'].mean().reset_index()

    plt.figure(figsize=(10, 6))
    sns.barplot(x='group', y='value', hue='group', data=group_means, palette='viridis')
    plt.title('Barplot of Values by Group')
    plt.xlabel('Groups')
    plt.ylabel('Mean Values')
    plt.savefig('../../output/ANOVA/ANOVA_scipy/barplot.png')
    plt.close()

    group_stats = data.groupby('group')['value'].agg(['mean', 'std', 'count']).reset_index()
    print("\nGroup Statistics:")
    print(group_stats)

    return group_stats
```

Obrázok č. 14: Pomocné funkcie ANOVA

(*Zdroj: vlastné spracovanie)*

Pomocou knižníc Matplotlib a Seaborn je možné vykresliť boxploty na porovnanie distribúcií dát vrátane kvantilov medzi skupinami, ako aj stĺpcový graf priemerov jednotlivých skupín. Pred samotnou vizualizáciou je potrebné transformovať pôvodné dáta do požadovanej štruktúry.

```
Group Statistics:
     group       mean       std  count
0  Group_1  54.974790  5.240750    119
1  Group_2  59.311475  4.853357    122
2  Group_3  65.008130  5.136651    123
3  Group_4  70.058333  4.802478    120
4  Group_5  74.806452  5.257093    124
5  Group_6  79.620968  5.133332    124
6  Group_7  85.219512  4.878082    123
7  Group_8  89.600000  4.986368    120
```

Obrázok č. 15: Výstup funkcie graph

(Zdroj: vlastné spracovanie)

Funkcia zároveň zoskupuje dáta podľa jednotlivých skupín a vypočítava priemer, štandardnú odchýlku a počet pozorovaní. Výsledné hodnoty vracia vo forme objektu typu DataFrame a súčasne ich vypisuje do konzoly.

```
def test_assumptions(data_frame, group_col, value_col):
    model = ols(f'{value_col} ~ C({group_col})', data=data_frame).fit()
    residuals = model.resid

    shapiro_test = stats.shapiro(residuals)
    print(f"\nShapiro-Wilk test for normality: p-value = {shapiro_test.pvalue:.4f}")

        groups = [data_frame[data_frame[group_col] == group][value_col] for group in
data_frame[group_col].unique()]
    levene_test = stats.levene(*groups)
    print(f"Levene's test for homogeneity of variances: p-value = {levene_test.pvalue:.4f}")

    tukey = pairwise_tukeyhsd(data_frame['value'], data_frame['group'], alpha = 0.05)
    print("\nPost-hoc analysis (Tukey's HSD):")
    print(tukey)
```

Na výpočty testov predpokladov modelu boli použité knižničné funkcie zo Statsmodels, ktoré kontrolujú normalitu reziduálov a homogenitu rozptylov. Tukeyho HSD test párovo porovnáva skupiny a testuje rozdiely medzi ich priemermi, čím určuje, medzi ktorými skupinami existuje štatisticky významný rozdiel.

```
def perform_anova(data):

    groups = [data[data['group'] == group]['value'] for group in data['group'].unique()]
    print(groups, "\n", print(type(groups)))
    f_stat, p_value = stats.f_oneway(*groups)
    print(f"SciPy ANOVA results: F-statistic = {f_stat:.4f}, p-value = {p_value:.4f}")

    return f_stat, p_value
```

Pred samotným výpočtom je potrebné dáta pretransformovať na model lineárnej regresie. Následne je možné pomocou funkcie anova_lm získať hodnoty potrebné na výpočet ANOVY.

```
Statsmodels ANOVA results:
                 sum_sq     df          F  PR(>F)
C(group)  127309.021333    7.0  716.16121     0.0
Residual   24557.085334  967.0        NaN     NaN
```

Obrázok č. 16: Výstup Statsmodels

(Zdroj: vlastné spracovanie)

Výsledný výstup funkcie obsahuje všetky potrebné hodnoty na určenie štatistickej významnosti modelu: sumu štvorcov, počet stupňov voľnosti (údaje, pomocou ktorých sa dá vypočítať F-štatistika manuálne) a samotnú F-štatistiku.

```python
def perform_anova(data):

    groups = [data[data['group'] == group]['value'] for group in data['group'].unique()]
    print(groups, "\n", print(type(groups)))
    f_stat, p_value = stats.f_oneway(*groups)
    print(f"SciPy ANOVA results: F-statistic = {f_stat:.4f}, p-value = {p_value:.4f}")

    return f_stat, p_value
```

Prístup k výpočtu v porovnaní so Statsmodels je odlišný, pretože dáta je najskôr potrebné pretransformovať do X-rozmerného zoznamu dátových typov Series, kde X predstavuje počet pozorovaných skupín. Funkcia f_oneway následne vráti iba hodnotu F-štatistiky a p-value. Tieto výsledky sú postačujúce na základné posúdenie štatistickej významnosti rozdielov medzi skupinami, avšak neposkytujú hlbší pohľad na dáta alebo model.

Podobne ako pri lineárnej regresii boli vytvorené aj vlastné kódy na výpočet ANOVY – jeden s procedurálnou paradigmou a druhý v objektovo-orientovanom prístupe, pričom všetky medzivýpočty sú zaznamenané v Exceli. Porovnaniu týchto vlastných prístupov s knižničnými metódami sa bude venovať nasledujúca podkapitola.

3.4.2. R

R poskytuje funkciu na výpočet ANOVY priamo v základnom balíku Base R, čím eliminuje potrebu importu ďalších externých knižníc.

```
data <- read_excel("input/ANOVA/ANOVA_small.xlsx")
data_clean <- data %>% drop_na(value)

anova_result <- aov(value ~ group, data = data_clean)

summary_result <- summary(anova_result)
```

Na výpočet v jazyku R postačujú iba štyri riadky kódu, keďže dáta sú už pripravené v požadovanej štruktúre.

```
> summary_result
             Df Sum Sq Mean Sq F value Pr(>F)
group         7 127309   18187   716.2 <2e-16 ***
Residuals   967  24557      25
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Obrázok č. 17: Výstup funkcie aov()

(Zdroj: vlastné spracovanie)

Výstup sa podobá na ten zo Statsmodels, avšak je doplnený ešte o štatistickú významnosť "group" pre rôzne hladiny významnosti, čo je pri interpretácii výhodou.

```
boxplot <- ggplot(data, aes(x = group, y = value, fill = group)) +
  geom_boxplot() +
  theme_minimal() +
  labs(
    title = "Boxplot of Values by Group",
    x = "Groups",
    y = "Values"   ) +
  theme(legend.position = "none")
ggsave(file.path(output_dir, "boxplot.jpg"), plot = boxplot, width = 10, height = 6)

barplot <- ggplot(group_means, aes(x = group, y = mean, fill = group)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(
    title = "Barplot of Mean Values by Group",
    x = "Groups",
    y = "Mean Values"   ) +
  theme(legend.position = "none")
ggsave(file.path(output_dir, "barplot.jpg"), plot = barplot, width = 10, height = 6)
```

Implementácia grafov je rovnako jednoduchá, stačí len vedieť názvy stĺpcov z dát. S použitím knižnice ggplot2 jednoducho vizualizovať výsledky. Grafy sa automaticky ukladajú do vopred určeného priečinku vo formáte .jpg.

```
tukey_result <- TukeyHSD(anova_result)
print(tukey_result)

residuals <- residuals(anova_result)
shapiro_test <- shapiro.test(residuals)
print(shapiro_test)
levene_test <- car::leveneTest(value ~ group, data = data_clean)
print(levene_test)
```

Na testovanie predpokladov modelu sme použili knižnice lmtest a car, ktoré obsahujú implementácie testov Tukeyho HSD alebo Levenov test homogenity rozptylu.

3.4.3. Porovnanie ANOVA

Zložitosť:

- Implementácia analýzy rozptylu v jazyku R bola veľmi jednoduchá, keďže jazyk priamo podporuje štatistické modelovanie už v základnom balíku. V prípade Pythonu bolo potrebné vstupné dáta pred spracovaním upraviť – buď transformáciou pomocou metódy najmenších štvorcov, alebo zmenou dátového typu, čo zvyšuje zložitosť a môže spomaliť proces implementácie.

Výstupy:

- Knižnice ako Statsmodels (Python) a Base R poskytujú rozsiahle výstupy, ktoré obsahujú všetky kľúčové štatistické ukazovatele – súčet štvorcov, stupne voľnosti, F-štatistiku a p-hodnotu. Tieto údaje sú nevyhnutné pre dôkladnú interpretáciu modelu a hlbšiu analýzu výsledkov.
- Naopak, knižnica SciPy ponúka iba základný výstup, ktorý obsahuje F-štatistiku a p-hodnotu. Hoci to postačuje na rýchle vyhodnotenie štatistickej významnosti, neposkytuje detailnejší pohľad na štruktúru modelu.

Flexibilita:

- Rozšírenie analýzy o ďalšie výpočty alebo vizualizácie je v oboch jazykoch možné, avšak často si vyžaduje použitie doplnkových knižníc. V prípade Pythonu ide napríklad o statsmodels.stats.multicomp alebo seaborn, zatiaľ čo v R sú bežne využívané knižnice ako ggplot2 či car. Vďaka tomu je možné doplniť napríklad viacnásobné porovnania skupín, testy predpokladov alebo grafické výstupy.

Presnosť výsledkov:

- Rozdiely vo výsledkoch výpočtov medzi jednotlivými implementáciami boli minimálne. Napríklad R uvádza extrémne malé p-hodnoty vo forme „< 2e-16“, zatiaľ čo Python ich často interpretuje ako „0.00“. Tento rozdiel je len formálny a nemá vplyv na interpretáciu výsledku, keďže v oboch prípadoch ide o štatisticky významný výstup. Vo zvyšných parametroch sa výsledky odlišovali len v miere zaokrúhlenia.

Vizualizácia:

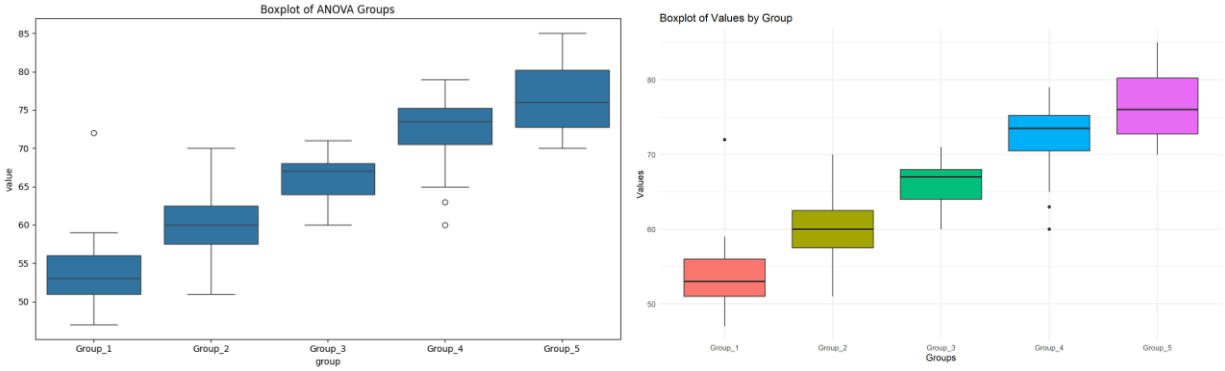
```
Post-hoc analysis (Tukey's HSD):
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
 group1  group2 meandiff p-adj   lower   upper  reject
----------------------------------------------------
Group_1 Group_2   6.3079 0.0009  2.0101 10.6057   True
Group_1 Group_3  12.0974    0.0  7.7995 16.3952   True
Group_1 Group_4    18.25    0.0 14.0076 22.4924   True
Group_1 Group_5    22.55    0.0 18.3076 26.7924   True
Group_2 Group_3   5.7895 0.0033  1.4369 10.1421   True
Group_2 Group_4  11.9421    0.0  7.6443 16.2399   True
Group_2 Group_5  16.2421    0.0 11.9443 20.5399   True
Group_3 Group_4   6.1526 0.0013  1.8548 10.4505   True
Group_3 Group_5  10.4526    0.0  6.1548 14.7505   True
Group_4 Group_5     4.3 0.0453  0.0576  8.5424   True
----------------------------------------------------
```

```
$group
                     diff        lwr       upr     p adj
Group_2-Group_1  6.307895  2.0100578 10.605732 0.0008760
Group_3-Group_1 12.097368  7.7995315 16.395205 0.0000000
Group_4-Group_1 18.250000 14.0076214 22.492379 0.0000000
Group_5-Group_1 22.550000 18.3076214 26.792379 0.0000000
Group_3-Group_2  5.789474  1.4368851 10.142062 0.0032807
Group_4-Group_2 11.942105  7.6442684 16.239942 0.0000000
Group_5-Group_2 16.242105 11.9442684 20.539942 0.0000000
Group_4-Group_3  6.152632  1.8547947 10.450468 0.0012506
Group_5-Group_3 10.452632  6.1547947 14.750468 0.0000000
Group_5-Group_4  4.300000  0.0576214  8.542379 0.0453365
```

Obrázok č. 18: Porovnanie výstupu Tukeyho testu

(Zdroj: vlastné spracovanie)

Výstup generovaný v jazyku Python (vľavo) bol v tomto prípade podrobnejší, keďže okrem základných štatistických ukazovateľov obsahoval aj doplnkovú interpretáciu výsledkov vo forme rozhodnutia o zamietnutí nulovej hypotézy na základe nastavenej hladiny významnosti a upravenej p-hodnoty (p-adj). Samotné numerické hodnoty vo výstupoch oboch jazykov boli identické, s výnimkou rozdielov spôsobených odlišným zaokrúhľovaním.



Obrázok č. 19: Porovnanie grafov ANOVA

(Zdroj: vlastné spracovanie)

Ako možno pozorovať, vizualizácie sú takmer totožné, pričom graf vytvorený v prostredí R (vpravo) je navyše farebne odlíšený, čo prispieva k lepšej prehľadnosti. Na oboch grafoch je znázornený medián, miera variability vyjadrená prostredníctvom medzikvartilového rozpätia (IQR) a tiež prítomnosť odľahlých hodnôt (outliers).

3.5. Benchmarking a hodnotenie výkonnosti

Táto podkapitola sa zameriava na porovnanie vytvorených skriptov z hľadiska časovej náročnosti výpočtov. V úvode sú predstavené konkrétne implementácie funkcií v jazykoch Python a R, ktoré slúžia na meranie doby vykonávania programu. Zaznamenané hodnoty sú ďalej využívané pri analýze výkonnosti jednotlivých prístupov. Následne je uvedený prehľad použitého hardvéru, na ktorom boli merania realizované, a na záver sú prezentované a porovnané výsledky získané v rámci testovania.

3.5.1. Príprava na testovanie

V Pythone sme vytvorili nový súbor s názvom execution_timer.py, v ktorom sme zadefinovali 3 funkcie: measure_execution_time, timed_input a append_execution_time.

```
def measure_execution_time(func, *args, **kwargs):

    start_time = time.time()
    result = func(*args, **kwargs)
    end_time = time.time()
    execution_time = end_time - start_time
    if result is not None:
        return result, end_time - start_time
    else:
        return execution_time

def timed_input(prompt=""):

    pause_time = time.time()
    user_input = input(prompt)
    resume_time = time.time()
    return user_input, resume_time - pause_time
```

Funkcia measure_execution_time akceptuje ako parameter ďalšiu funkciu, ktorej čas sa má merať (*args a **kwargs sú ľubovoľné argumenty, ktoré sa odovzdávajú funkcii func). Premenná na začiatku funkcie sa uloží čas spustenia, do premennej result sa uloží návratová hodnota funkcie posunutej ako argument a potom sa uloží čas ukončenia. Ak result neobsahuje nič (čo znamená, že funkcia nemala žiadnu návratovú hodnotu), measure_execution_time vráti len dobu exekučného času ako rozdiel konca a začiatku, inak vráti aj premennú result.

Timed_input obsahuje parameter prompt=“ “, ktorý reprezentuje text, ktorý sa používateľovi zobrazí ako výzva na konzole. Následne vypočíta dobu čakania na vstup od používateľa a vráti vstup s dobou čakania.

Posledná funkcia má povinné parametre execution_time, method a computer_name. K týmto údajom sa dopíše jazyk a čas zapísania dát a vytvorí sa dataframe. Následne prebehne kontrola, či už existuje Excelový súbor a ak áno, existujúce dáta spojí s novými a do príslušného hárku podľa parametra method ich zapíše. V prípade, že hárok alebo Excel neexistuje, vytvorí ich. V prípade chyby z niektorých krokov funkcie vypíše chybovú hlášku.

```python
def append_execution_time(execution_time, method, computer_name, excel_file):  benjamin_kolarik
    Path(os.path.dirname(excel_file)).mkdir(parents=True, exist_ok=True)

    new_data = pd.DataFrame({
        "Method": [method],
        "Computer": [computer_name],
        "Execution_time": [float(execution_time)],
        "Timestamp": [datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")],
        "Language": ["Python"]
    })

    try:
        if os.path.exists(excel_file):
            try:
                with pd.ExcelFile(excel_file) as xls:
                    sheet_dict = {sheet: pd.read_excel(excel_file, sheet_name=sheet)
                                  for sheet in xls.sheet_names}

                if method in sheet_dict:
                    sheet_dict[method] = pd.concat( objs: [sheet_dict[method], new_data], ignore_index=True)
                else:
                    sheet_dict[method] = new_data

                with pd.ExcelWriter(excel_file, engine="openpyxl") as writer:
                    for sheet_name, df in sheet_dict.items():
                        df.to_excel(writer, sheet_name=sheet_name, index=False)

            except Exception as e:
                print(f"Error reading existing Excel file: {str(e)}")
                print("Creating new Excel file instead...")
                with pd.ExcelWriter(excel_file, engine="openpyxl") as writer:
                    new_data.to_excel(writer, sheet_name=method, index=False)
        else:
            # Create new file
            with pd.ExcelWriter(excel_file, engine="openpyxl") as writer:
                new_data.to_excel(writer, sheet_name=method, index=False)

        print(f"Execution time {execution_time:.6f} seconds added to {excel_file} in sheet '{method}'")

    except Exception as e:
        print(f"Error appending execution time: {str(e)}")
        print(f"Execution time was {execution_time:.6f} seconds for method {method}")
```

Obrázok č. 20: Funkcia append_execution_time

(*Zdroj: vlastné spracovanie)*

```
if __name__ == "__main__":
    result = measure_execution_time(main)

    if isinstance(result, tuple):
        wait_time, execution_time = result
        print(f"\nTotal execution time: {execution_time:.6f} seconds")
        print(f"Waiting time: {wait_time:.6f} seconds")
        print(f"Active execution time: {execution_time - wait_time:.6f} seconds")

        append_execution_time(
            execution_time - wait_time,
            method="LR - statsmodels",
            computer_name="Windows Ryzen 9 5900x 32GB",
            excel_file="../../output/execution_times/h/moje/execution_times_python_small.xlsx"
)
    else:
        execution_time = result         print(f"\nTotal execution time: {execution_time:.6f}
seconds")

        append_execution_time(
            execution_time,
            method="LR - statsmodels",
            computer_name="Windows Ryzen 9 5900x 32GB",
            excel_file="../../output/execution_times/h/moje/execution_times_python_small.xlsx"
)
```

Takýmto spôsobom sa bude merať každý program, všetky postupné kroky a funkcie sa nachádzajú vo funkcii main().

V prostredí jazyka R ostáva relevantná už iba funkcia append_execution_time, keďže ostatné dve funkcie nie sú potrebné z dôvodu absencie interakcie s používateľom. Meranie exekučného času sa realizuje pomocou funkcie Sys.time(), ktorá je volaná na začiatku a na konci každého skriptu. Získané časové údaje sa ukladajú do samostatných premenných, pričom rozdiel medzi nimi predstavuje celkový čas vykonávania daného skriptu.

```
append_execution_time <- function(time_second, method_name = "LR", excel_file =
"output/execution_times/R/execution_times_R_small.xlsx", computer_name) {
  library(openxlsx)

 dir.create(dirname(excel_file), recursive = TRUE, showWarnings = FALSE)
  time_data <- data.frame(
    Method = method_name,
    Computer = computer_name,
    Execution_time = as.numeric(time_second),
    Timestamp = format(Sys.time(), "%Y-%m-%d %H:%M:%S"),
    Language = 'R'   )

  if(file.exists(excel_file)) {
    wb <- loadWorkbook(excel_file)
        if (method_name %in% names(wb)){
          existing_data <- read.xlsx(wb, sheet = method_name)
          updated_data <- rbind(existing_data, time_data)
          writeData(wb, sheet = method_name, updated_data, startRow = 1)
        } else {
          addWorksheet(wb, method_name)
          writeData(wb, sheet = method_name, time_data, startRow = 1)
        }
  } else{
    wb <- createWorkbook()
    addWorksheet(wb, method_name)
```

```
    writeData(wb, sheet = method_name, time_data, startRow = 1)
  }
    saveWorkbook(wb, excel_file, overwrite = TRUE)

  print(paste("Execution time data appended to", excel_file, "in sheet", method_name))
}
```

Funkcia funguje na rovnakom princípe ako jej verzia v Pythone, parametre má rovnaké, z nich vytvorí dataframe a skontroluje, či už Excel s daným hárkom existuje. Ak áno, nové dáta jednoducho spojí s pôvodnými a zapíše. Ak metóda alebo Excel súbor neexistuje, vytvorí ich.

```
start_time <- Sys.time()
end_time <- Sys.time()
execution_time <- end_time - start_time execution_time

append_execution_time(
    time_second = execution_time,
    method_name = "ANOVA",
    computer_name = "Windows Ryzen 9 5900x 32GB"
)
```

Oba programovacie jazyky zapisujú hodnoty do Excelu rovnakým spôsobom, ktorý je zobrazený v nasledujúcej tabuľke:

Method	Computer	Execution_time	Timestamp	Language
LR_scikit	Windows Ryzen 9 5900x 32GB	0.493910313	2025-04-06 17:27:18	Python
LR_scikit	Windows Ryzen 9 5900x 32GB	0.491721392	2025-04-06 17:27:38	Python

Tabuľka č. 10: Príklad zapisovania údajov

(Zdroj: vlastné spracovanie)

3.5.2. Testovaný hardware

Na účely porovnania výpočtovej efektivity skriptov v jazykoch R a Python boli použité zariadenia s rôznou konfiguráciou hardvéru, operačných systémov a procesorovej architektúry. Cieľom bolo pokryť široké spektrum bežne dostupných zariadení, od výkonných desktopových počítačov až po prenosné notebooky s rôznym výkonom, aby bolo možné objektívne zhodnotiť, ako sa správa výkon skriptov v rôznych podmienkach.

Testované zariadenia zahŕňali:

- Výkonné desktopové PC s 12-jadrovým procesorom AMD Ryzen 9 5900X a 32 GB RAM, ktoré slúžilo ako referenčná jednotka s vysokým výkonom.
- Moderné notebooky so systémom Windows 11 a procesorom Intel i5 1340P (12-jadrový hybridný dizajn) a AMD Ryzen 5600H, ktoré reprezentujú aktuálnu generáciu pracovných zariadení.
- Starší notebook s Windows 10 a procesorom Intel i5 6200U, ktorého zaradenie do testovania poskytuje pohľad na výkon na zariadeniach s nižšou výpočtovou kapacitou.
- Notebook s operačným systémom Mac OS a čipom M1 reprezentuje odlišnú procesorovú architektúru – ARM s tradičnou x86. Zároveň je charakteristické využitím unifikovanej pamäťovej architektúry UMA, čo vplýva na spôsob spracovania dát a alokáciu zdrojov.

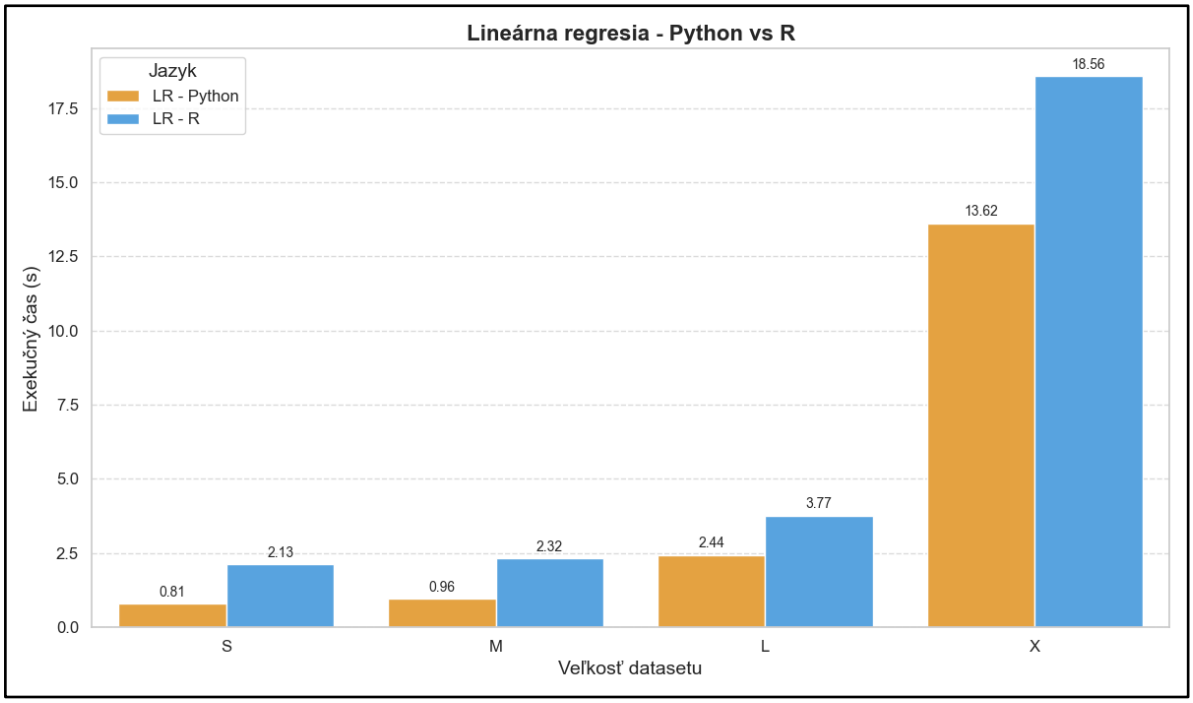
OS	Procesor	RAM
Windows 11	AMD Ryzen 9 5900x	32GB
Windows 11	Intel i5 1340p	32GB
Windows 10	Intel i5 6200U	12GB
Windows 11	AMD Ryzen 5 5600H	16GB
Mac OS	Apple M1	16GB

Tabuľka č. 11: Použitý hardware

(*Zdroj: vlastné spracovanie)*

3.5.3. Testovanie a analýza výsledkov

Testovanie bolo realizované systematicky na všetkých zariadeniach uvedených v kapitole *Testovaný hardvér*. Pre každú kombináciu programovacieho jazyka (Python alebo R), veľkosti vstupného datasetu (malý, stredný, veľký, extra veľký) a aplikovanej štatistickej metódy (napr. lineárna regresia, ANOVA – Statsmodels, ANOVA – SciPy) bol každý skript spustený 10-krát. Tento postup slúžil na minimalizáciu vplyvu náhodných výkyvov systémovej záťaže, ktoré by mohli ovplyvniť presnosť meraní. Všetky namerané časy boli zaznamenané a následne spriemerované, čím sa zabezpečilo získanie reprezentatívnej hodnoty výpočtovej náročnosti pre danú konfiguráciu. Výsledky meraní boli následne porovnané medzi jednotlivými zariadeniami, metódami a jazykmi s cieľom objektívne zhodnotiť výkonnosť jednotlivých riešení v rôznych podmienkach.



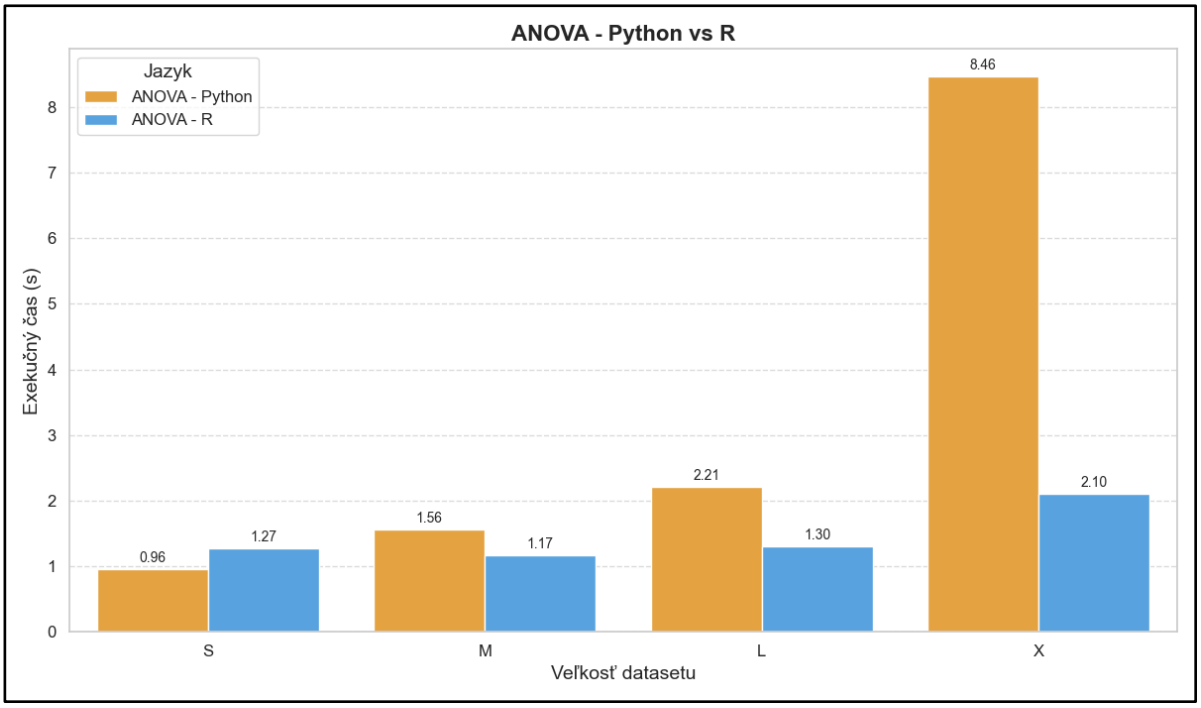
Graf č. 5: Porovnanie exekučného času lineárnej regresie Python vs R

(Zdroj: vlastné spracovanie)

Na grafe je znázornené porovnanie výpočtového času (v sekundách) pre lineárnu regresiu implementovanú v jazykoch Python a R pri rôznych veľkostiach datasetu. Časy Pythonu zodpovedajú priemeru riešení pomocou všetkých knižníc podporujúcich lineárnu regresiu, aj vlastnej implementácií.

Z grafu vyplýva, že pri menších datasetoch (S a M) sú rozdiely medzi jazykmi relatívne malé, no Python vykazuje o niečo rýchlejšie spracovanie. Pri veľkosti „S“ trval výpočet v Pythone len 0,81 sekundy, zatiaľ čo v R až 2,13 sekundy. Tento trend sa potvrdzuje aj pri veľkosti „M“ (0,96 s vs. 2,32 s).

Pri väčších datasetoch sa rozdiely postupne zvyšujú. Pri „L“ je výpočtový čas v Pythone 2,44 s, v R už 3,77 s. Najväčší rozdiel je však pri extra veľkom datasete („X“), kde Python dosiahol čas 13,62 sekundy, zatiaľ čo R až 18,56 sekundy. Avšak treba brať v úvahu aj fakt, že pri vlastnej implementácií so zápisom medzivýpočtov do Excelu bol priemerný čas naprieč zariadeniami 48,42 sekundy, čo pomerne výrazne zvyšuje priemer ostatných riešení.



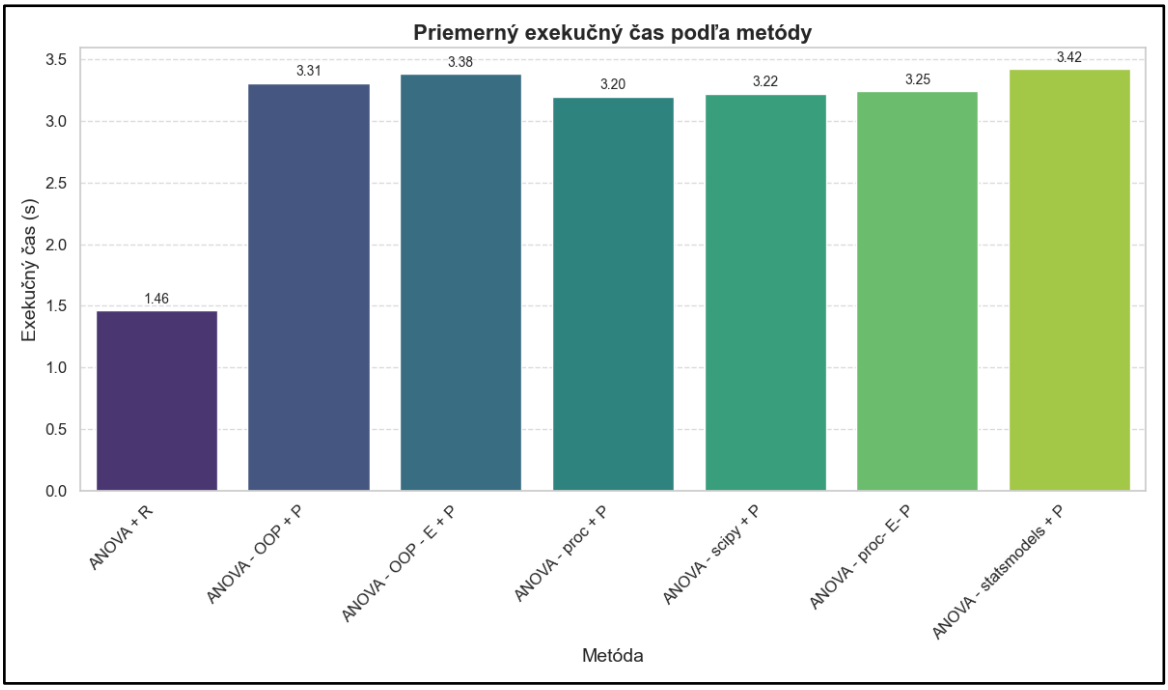
Graf č. 6: Porovnanie exekučného času analýzy rozptylu Python vs R

(Zdroj: vlastné spracovanie)

Na rozdiel od lineárnej regresie, v prípade ANOVA sa ukázalo, že jazyk R je pri spracovaní väčších datasetov výrazne efektívnejší. Pri menších datasetoch sú časy porovnateľné: pre veľkosť „S“ trvá výpočet v Pythone 0,96 sekundy, v R 1,27 sekundy. Pri veľkosti „M“ je naopak R mierne rýchlejšie (1,17 s vs. 1,56 s).

Výrazný rozdiel sa prejavuje už pri datasete „L“, kde výpočet v Pythone trvá 2,21 sekundy, zatiaľ čo v R iba 1,30 sekundy. Najväčší rozdiel je pri extra veľkom datasete („X“), kde výpočet v Pythone trvá až 8,46 sekundy, zatiaľ čo v R iba 2,10 sekundy. To predstavuje viac než štvornásobný rozdiel v prospech R.

Tieto výsledky naznačujú, že implementácia ANOVA v R je výrazne optimalizovanejšia a lepšie škáluje s rastúcim objemom dát. Naopak, implementácie v Pythone (napr. v SciPy alebo Statsmodels) môžu byť menej optimalizované alebo menej efektívne využívať pamäť a viac závisieť od interného pretypovania dát.



Graf č. 7: Rozdiel medzi jednotlivými implementáciami ANOVA

(*Zdroj: vlastné spracovanie)*

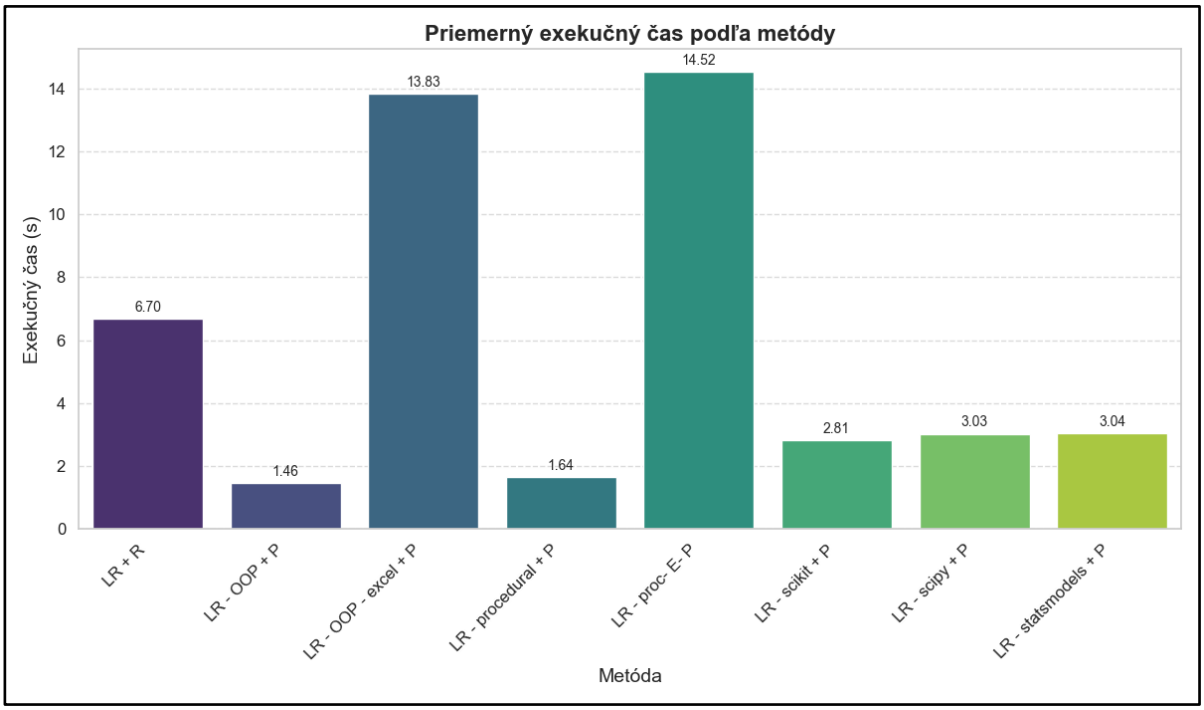
Graf zobrazuje priemerný exekučný čas pre rôzne implementácie jednofaktorovej analýzy rozptylu (ANOVA) v jazykoch Python a R, zosumarizované naprieč všetkými veľkosťami datasetov.

Z výsledkov je jasné, že najrýchlejšia implementácia ANOVY je v jazyku R, ktorá dosiahla priemerný čas iba 1,46 sekundy. Oproti tomu všetky testované implementácie v Pythone dosahujú v priemere viac než dvojnásobný čas.

Metódy Pythonu sú na tom časovo veľmi podobne – čas sa pohybuje medzi 3,20 až 3,42 sekundy. Najpomalšia bola implementácia pomocou knižnice Statsmodels (3,42 s), nasledovaná objektovo orientovaným prístupom a zápisom do Excelu OOP-E (3,38 s) a OOP (3,31 s). Mierne lepšie výsledky dosiahli knižnica SciPy a vlastné riešenie s procedurálnou paradigmou (3,22 s, resp. 3,20 s).

Tieto výsledky poukazujú na to, že R je efektívnejšie pri výpočte ANOVY nielen pri veľkých datasetoch, ale aj v priemere naprieč rôznymi veľkosťami datasetov. Rozdiely medzi jednotlivými Python implementáciami sú malé, no môžu byť dôležité pri opakovanom volaní funkcie na veľkom množstve dát. Treba tu však aj podotknúť, že hlavnú výhodu malo Rko vďaka lepšej optimalizácií funkcie na výpočet Tukeyho testu, tu Python stratil viac ako 1 sekundu.

Zároveň nám tento graf naznačuje, že pri výbere implementácie ANOVY v Pythone nezáleží zásadne na použitom prístupe (OOP vs. procedurálny), keďže výkonnostné rozdiely sú minimálne.



Graf č. 8: Rozdiel medzi jednotlivými implementáciami lineárnej regresie

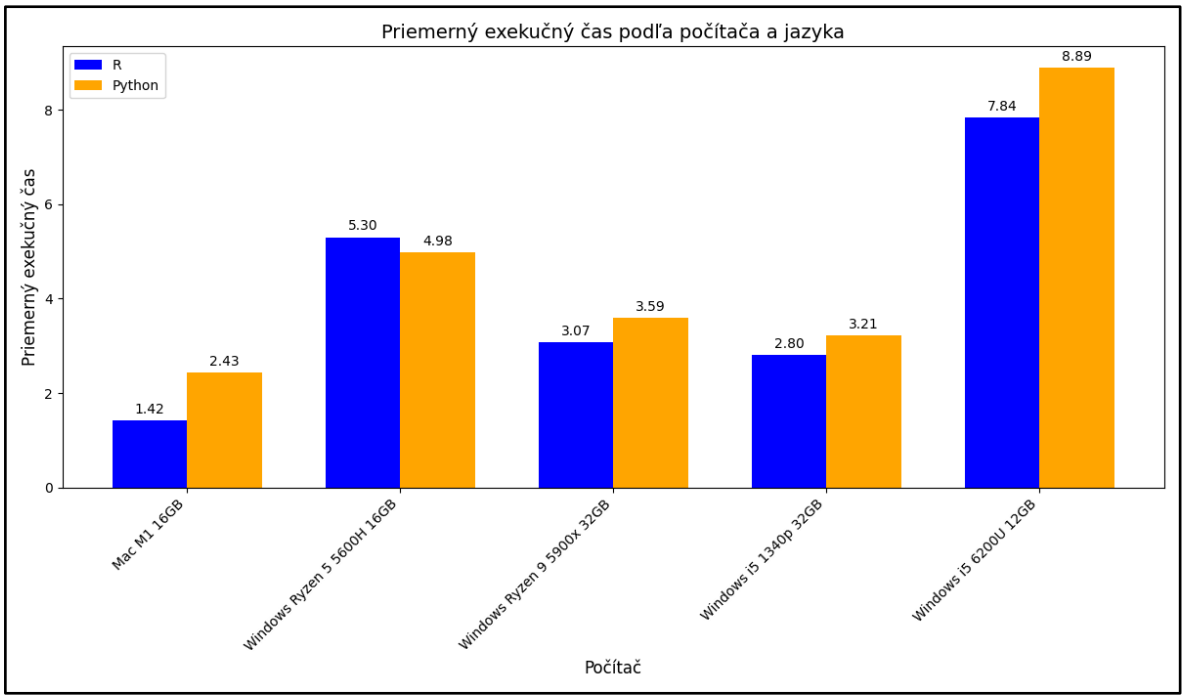
(*Zdroj: vlastné spracovanie)*

Najrýchlejšími implementáciami boli vlastné implementácie v Pythone, a to objektovo orientovaný prístup s priemerným časom iba 1,46 sekundy a tesne za ním procedurálny prístup s časom 1,64 sekundy.

Oproti tomu implementácie využívajúce štandardné dátovo-analytické knižnice Pythonu boli pomalšie, aj keď s veľmi podobnými výsledkami medzi sebou. Implementácia pomocou scikit dosiahla čas 2,81 s, nasledovaná SciPy s 3,03 s a Statsmodels s 3,04 s. Tieto časy sú približne dvojnásobkom vlastných implementácií. Môžeme si všimnúť, že sú zoradené aj podľa komplexnosti výstupov. Statsmodels, ktorá mala výstup najkomplexnejší sa umiestnila medzi knižnicami Pythonu na poslednom mieste, kým knižnica scikit s jednoduchým výstupom bola v priemere najrýchlejšia.

Implementácia v jazyku R bola v priemere 2-krát pomalšia ako knižniče Pythonu a až 4-krát pomalšia ako vlastné implementácie. Tento výsledok môže byť prekvapivý, najmä ak sa pozrieme na ANOVU, kde bolo Rko bezkonkurečne najrýchlejšie.

Z prvého pohľadu na graf je zrejmé, že zapisovanie medzivýpočtov do Excelu nie je veľmi optimalizované a kódu to výpočtovo uškodilo. Avšak v prípade extra veľkého datasetu (100 000 pozorovaní) sa doň zapísalo 95 574 * 15 údajov, takže sa nie je moc čomu čudovať.



Graf č. 9: Porovnanie systémov a jazykov

(Zdroj: vlastné spracovanie)

Z výsledkov je na prvý pohľad jasné, že najlepší výkon dosiahla konfigurácia Mac M1 (16GB). Na tomto systéme dosiahol jazyk R suverénne najkratší priemerný čas, iba 1,42 sekundy. Rovnako najlepší čas dosiahol v jazyku Python, a to s časom 2,43 sekundy, aj keď rozdiel už nebol taký veľký.

Na opačnom konci výkonnostného spektra sa nachádza systém Windows i5 6200U (16GB), reprezentujúci staršiu generáciu hardvéru. Tu boli výpočty výrazne najpomalšie pre oba jazyky, takmer dvojnásobne.

Ostatné tri konfigurácie (Windows Ryzen 5 5600H, Windows Ryzen 9 5900X, Windows i5 1340p) sa výkonnostne zaradili medzi tieto dva extrémy. Merania medzi jazykmi boli pomerne vyrovnané, avšak ako aj pri ostatných systémoch, R malo väčšinou navrch. Jedinou výnimkou, kde Python dosiahol lepší výsledok ako R, bola konfigurácia Windows Ryzen 5 5600H (16GB). Tu Python úlohu zvládol v priemere za 4,98 sekundy, zatiaľ čo R potreboval 5,30 sekundy. Rozdiel bol síce malý, ale pre Python išlo o jediné víťazstvo v tomto teste.

Záver

Cieľom tejto bakalárskej práce bolo porovnať efektivitu a praktickú použiteľnosť rôznych implementácií štatistických metód v jazykoch Python a R, pričom hlavná pozornosť bola venovaná metódam lineárnej regresie a jednofaktorovej analýzy rozptylu (ANOVA). V práci sme sa zamerali na porovnanie z hľadiska časovej náročnosti, presnosti výstupov, použiteľnosti knižníc, ako aj flexibility a možností rozšírenia. Súčasťou výskumu bolo aj testovanie na viacerých zariadeniach s rôznym hardvérom a operačnými systémami, aby bolo možné zohľadniť rôznorodosť výpočtového prostredia.

Z výsledkov vyplynulo, že lineárna regresia bola všeobecne rýchlejšie vykonaná v prostredí Pythonu, najmä pri väčších datasetoch. Naopak, pri metóde ANOVA sa výrazne lepšie darilo jazyku R, ktorý vykazoval nižší exekučný čas vo všetkých testovaných prípadoch – najmä pri extrémne veľkých datasetoch, kde bol rozdiel medzi R a Pythonom až štvornásobný. Tieto rozdiely môžu byť spôsobené jednak optimalizáciou jadra R pre štatistické výpočty, ako aj rôznou implementáciou a výpočtovými modelmi jednotlivých knižníc.

Z pohľadu použiteľnosti a komplexnosti knižníc sa ukázalo, že základné implementácie v oboch jazykoch sú relatívne jednoduché a vyžadujú minimum kódu pre základnú analýzu. V Pythone však častejšie nastávala potreba transformovať alebo upraviť dáta (napr. konverzia dátových typov), čo môže proces mierne spomaliť alebo skomplikovať. Knižnice ako Statsmodels, Scipy či Scikit poskytujú dostatok funkcií, avšak v porovnaní s R ponúkajú menej rozsiahle a menej štruktúrované výstupy, čo môže byť nevýhodou pre hĺbkovú analytickú prácu.

Pri porovnaní flexibility a možností rozšírenia sú oba jazyky dostatočne robustné, no Python vyniká svojou univerzálnosťou mimo štatistiky – je vhodný aj na tvorbu webových aplikácií či machine learning. R je naopak viac špecializovaný, no v oblasti štatistiky ponúka veľmi efektívne nástroje s množstvom dostupných balíkov.

Z pohľadu vizualizácie dát sú oba jazyky porovnateľné. R so svojimi knižnicami ako ggplot2 poskytuje veľmi elegantné a profesionálne výstupy, zatiaľ čo Python s matplotlib či seaborn ponúka flexibilitu a jednoduchosť. Výber medzi nimi závisí najmä od preferencie používateľa a cieľovej oblasti projektu.

Na záver možno konštatovať, že ani jeden z jazykov nie je jednoznačne lepší, avšak v konkrétnych prípadoch sa jeden môže javiť ako vhodnejší – napríklad R pri výpočtovo náročných štatistických operáciách, Python pri širšej integrácii s inými systémami a aplikáciami. Výber nástroja by teda mal byť podmienený konkrétnymi požiadavkami projektu, znalosťami používateľa a dostupnými výpočtovými prostriedkami.

Zoznam použitej literatúry

[1] A S, Ravikiran. *Data Visualization in Python* [online]. Dostupné z: https://www.simplilearn.com/tutorials/python-tutorial/data-visualization-in-python [cit. 15.4.2025]

[2] BASUMALLICK, Chiradeep. *R vs Python* [online]. Dostupné z: https://www.spiceworks.com/tech/devops/articles/r-vs-python/ [cit. 23.3.2025]

[3] BOBBITT, Zach. *Linear Regression Real-Life Examples* [online]. Dostupné z: https://www.statology.org/linear-regression-real-life-examples/ [cit. 17.3.2025]

[4] CORNELL UNIVERSITY. *Grammar of Graphics* [online]. Dostupné z: https://info5940.infosci.cornell.edu/notes/dataviz/grammar-of-graphics/ [cit. 25.3.2025]

[5] CRAN. *Tidyr PDF Manual* [online]. Dostupné z: https://cran.r-project.org/web/packages/tidyr/tidyr.pdf [cit. 22.3.2025]

[6] DANKO, Jakub – ŠAFR, Karel. *R snadno a Rychle 1: Úvod do jazyka.* Praha: Oeconomica, 2020, 161 s. ISBN 978-80-245-2380-4 Dostupné na internete: https://oeconomica.vse.cz/wp-content/uploads/Danko_Safr_R-snadno-a-rychle_1.pdf

[7] DATACAMP. *Top Programming Languages for Data Scientists in 2022* [online]. Dostupné z: https://www.datacamp.com/blog/top-programming-languages-for-data-scientists-in-2022 [cit. 22.3.2025]

[8] GEEKSFORGEEKS. *Applications of ANOVA in Real Life* [online]. Dostupné z: https://www.geeksforgeeks.org/applications-of-anova-in-real-life/ [cit. 3.4.2025]

[9] GEEKSFORGEEKS. *Pandas Tutorial* [online]. Dostupné z: https://www.geeksforgeeks.org/pandas-tutorial/ [cit. 30.3.2025]

[10] GEEKSFORGEEKS. *R vs Python in Data Science* [online]. Dostupné z: https://www.geeksforgeeks.org/r-vs-python-datascience/ [cit. 15.4.2025]

[11] GEEKSFORGEEKS. *Tidyr Package in R Programming* [online]. Dostupné z: https://www.geeksforgeeks.org/tidyr-package-in-r-programming/ [cit. 10.3.2025]

[12] HEROVIRED. *What is Linear Regression?* [online]. Dostupné z: https://herovired.com/learning-hub/blogs/what-is-linear-regression/ [cit. 17.3.2025]

[13] LABUDOVÁ, Viera – PACÁKOVÁ, Viera – SIPKOVÁ, Ľubica – ŠOLTÉS, Erik – VOJTKOVÁ, Mária. *Štatistické metódy pre ekonómov a manažérov*. Praha: Wolters Kluwer, 2021. 392 s. ISBN 978-80-571-0401-8.

[14] LIE HETLAND, Magnus. *Beginning Python: From Novice to Professional.* New York: Apress. 2005, 603 s., ISBN 978-15-905-9519-0

[15] MATHIEU. *History of Python Programming* [online]. Dostupné z: https://www.bocasay.com/history-python-programming/ [cit. 5.3.2025]

[16] MATPLOTLIB. *Plot Types* [online]. Dostupné z: https://matplotlib.org/stable/plot_types/index.html [cit. 30.3.2025]

[17] NUMPY. *Documentation* [online]. Dostupné z: https://numpy.org/doc/ [cit. 25.3.2025]

[18] OPENPYXL. *Documentation* [online]. Dostupné z: https://openpyxl.readthedocs.io/en/stable/ [cit. 28.3.2025]

[19] PENG, Roger. *R Programming for Data Science*. Lulu.com, 2016. 181 s. ISBN 978-13-650-5682-6.

[20] R PROJECT. *dplyr Vignette* [online]. Dostupné z: https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html [cit. 25.3.2025]

[21] RDOCUMENTATION. *ggplot2 Version 3.5.0* [online]. Dostupné z: https://www.rdocumentation.org/packages/ggplot2/versions/3.5.0 [cit. 10.3.2025]

[22] RDOCUMENTATION. *R Documentation* [online]. Dostupné z: https://www.rdocumentation.org/ [cit. 25.3.2025]

[23] RDOCUMENTATION. *stats Version 3.6.2* [online]. Dostupné z: https://www.rdocumentation.org/packages/stats/versions/3.6.2 [cit. 27.3.2025]

[24] REITER, Miroslav. *Python vs Java – Ktorý programovací jazyk sa naučíte rýchlejšie?* [online]. Dostupné na internete: https://www.vita.sk/blog/programovacie-jazyky/programovanie-python-vs-java-ktory-programovaci-jazyk-sa-naucite-rychlejsie/ [cit. 25.03.2025]

[25] SCIKIT-LEARN. *Scikit-learn Documentation* [online]. Dostupné z: https://scikit-learn.org/stable/ [cit. 25.3.2025]

[26] SCIPY. *SciPy Documentation* [online]. Dostupné z: https://docs.scipy.org/doc/scipy/ [cit. 28.3.2025]

[27] SEABORN. *Seaborn Documentation* [online]. Dostupné z: https://seaborn.pydata.org/ [cit. 28.3.2025]

[28] SEABOLD, Skipper – PERKTOLD, Josef**.** *Statsmodels: Econometric and statistical modeling with Python* [online]. In: Proceedings of the 9th Python in Science Conference. 2010 Dostupné z: https://www.statsmodels.org/stable/index.html [cit. 10.4.2025]

[29] SQUAREBOAT. *Advantages and Disadvantages of Python* [online]. Dostupné z: https://squareboat.com/blog/advantages-and-disadvantages-of-python [cit. 10.4.2025]

[30] TIDYVERSE. *Readxl Package* [online]. Dostupné z: https://readxl.tidyverse.org [cit. 27.3.2025]

[31] TUTORIALSPOINT. *Python - History* [online]. Dostupné z: https://www.tutorialspoint.com/python/python_history.htm [cit. 5.3.2025]

[32] W3SCHOOLS. *Pandas Tutorial* [online]. Dostupné z: https://www.w3schools.com/python/pandas/default.asp [cit. 30.3.2025]

[33] ŠAFR, Karel - DANKO, Jakub. *R snadno a Rychle 2: Vizualizace dat a programování.* Praha: Oeconomica, 2020, 174 s. ISBN 978-80-245-2381-1 Dostupné na internete: https://oeconomica.vse.cz/wp-content/uploads/Safr_Danko_R-snadno-a-rychle_2.pdf