|  |  |  |
| --- | --- | --- |
| **Univerzita Hradec Králové**  **Fakulta informatiky a managementu**  **Název katedry** | | |
| **Název bakalářské/diplomové práce**  (podtitul práce)  Bakalářská/Diplomová práce | | |
| Autor: Jméno, příjmení  Studijní obor: Studijní obor | | |
| Vedoucí práce: Titul, jméno, příjmení  Odborný konzultant: Titul, jméno, příjmení  Pracoviště | | |
| Hradec Králové | měsíc rok | |
| Prohlášení:  Prohlašuji, že jsem bakalářskou/diplomovou práci zpracoval/zpracovala samostatně a s použitím uvedené literatury. | | | |
| V Hradci Králové dne 4.1.2022 | | *vlastnoruční podpis*  Jméno a Příjmení | |
| Poděkování:  Děkuji vedoucímu bakalářské/diplomové práce titul, jméno, příjmení za metodické vedení práce a…. | | | |

Anotace

Text anotace – shrnutí cíle, významu práce a výsledky v ní dosažené. Délka minimálně 100 a maximálně 200 slov.

Annotation

Title: Název práce v anglickém jazyce

Anotace v anglickém jazyce. Délka minimálně 100 a maximálně 200 slov.

Obsah

[1 Úvod 1](#_Toc89591035)

[2 Cíl práce 2](#_Toc89591036)

[3 Metodika zpracování 3](#_Toc89591037)

[4 Teoretická část 4](#_Toc89591038)

[4.1 Vizualizace 4](#_Toc89591039)

[4.1.1 Vizualizace obecně 4](#_Toc89591040)

[4.1.2 Principy datové vizualizace 5](#_Toc89591041)

[4.1.3 Barvy v datové vizualizaci 5](#_Toc89591042)

[4.1.4 Anatomie grafu 7](#_Toc89591043)

[4.1.5 Nejčastější typy grafů 8](#_Toc89591044)

[4.1.6 Časté chyby při vizualizaci 14](#_Toc89591045)

[4.2 Jazyk Python 18](#_Toc89591046)

[4.3 Knihovny jazyka Python pro zpracování dat 18](#_Toc89591047)

[4.3.1 NumPy 18](#_Toc89591048)

[4.3.2 SciPy 19](#_Toc89591049)

[4.3.3 pandas 19](#_Toc89591050)

[4.4 Vizualizační knihovny jazyka Python 19](#_Toc89591051)

[4.4.1 Matplotlib 19](#_Toc89591052)

[4.4.2 Seaborn 20](#_Toc89591053)

[4.4.3 Bokeh 20](#_Toc89591054)

[4.4.4 Plotly 21](#_Toc89591055)

[4.4.5 Holoviews 21](#_Toc89591056)

[4.4.6 Pygal 22](#_Toc89591057)

[4.4.7 MidiTime 22](#_Toc89591058)

[4.4.8 Geoplotlib 22](#_Toc89591059)

[4.4.9 WordCloud 23](#_Toc89591060)

[5 Metodika zpracování 23](#_Toc89591061)

[5.1 Hodnocení knihoven 23](#_Toc89591062)

[6 Praktická část 24](#_Toc89591063)

[6.1 Způsoby získávání dat pomocí jazyka Python 24](#_Toc89591064)

[6.1.1 Načtení dat ze souboru 24](#_Toc89591065)

[6.1.2 Načtení dat z webové stránky 26](#_Toc89591066)

[6.1.3 Načtení dat z databáze 26](#_Toc89591067)

[6.1.4 Načtení dat pomocí API 26](#_Toc89591068)

[6.1.5 Načítání dat v reálném čase 26](#_Toc89591069)

[6.2 Práce s načtenými daty 26](#_Toc89591070)

[6.3 Porovnání vizualizačních knihoven 26](#_Toc89591071)

[6.3.1 Matplotlib 26](#_Toc89591072)

[6.3.2 Seaborn 30](#_Toc89591073)

[6.3.3 Bokeh 33](#_Toc89591074)

[6.3.4 Plotly 33](#_Toc89591075)

[6.3.5 Holoviews 33](#_Toc89591076)

[6.3.6 Pygal 33](#_Toc89591077)

[6.4 Ukázkové úlohy 33](#_Toc89591078)

[6.4.1 Podřazená podkapitola 34](#_Toc89591079)

[7 Shrnutí výsledků 35](#_Toc89591080)

[8 Závěry a doporučení 36](#_Toc89591081)

[9 Seznam použité literatury 37](#_Toc89591082)

[10 Přílohy 41](#_Toc89591083)

Seznam obrázků

[Obr. 1 Název obrázku/grafu/fotografie. 3](#_Toc348517265)

Seznam tabulek

[Tabulka 1 Název tabulky. 3](#_Toc348517268)

# Úvod

Zde vysvětlit problémovou situaci a otázky, které se budou v bakalářské/diplomové práci řešit.

# Cíl práce

Smysl a účel, výzkumné otázky.

# Metodika zpracování

Cíle, hypotézy/ výzkumné otázky, způsob hledání odpovědí na výzkumné otázky včetně metodiky vlastního výzkumu/šetření, literární rešerše.

# Teoretická část

## Vizualizace

### Vizualizace obecně

Definice pojmu „vizualizace dat/informací“ není až tak jednoznačná a jednoduchá, jak by se na první pohled mohlo zdát.

Kniha „Lecture Notes in Computer Science“ z roku 1970 popisuje, že „*Vizualizace informací využívá počítačové grafiky a interakce, aby napomohla lidem v řešení problémů*“ [1]. Vizualizace však nemusí nutně sloužit pouze k tomuto účelu, může najít uplatnění například ve výuce matematiky, kde „*Matematické koncepty, jako čísla, funkce, nebo vektory […] nejsou součástí intuitivního chápání, tak jako reálné fyzické objekty*“ [2]. Vizualizací je však možné pochopení těchto konceptů studentům usnadnit. „*Informační vizualizace nabízí možnost snáze a lépe ukázat klasické vizuální reprezentace matematických formátů, ale také je obohatit o prvky pohybu a interaktivity […] informační vizualizace tedy plní didaktické funkce nezbytné pro výuku matematiky.*“ [2]

Vizualizace může plnit i dekorativní roli, Lorène Fauvelle v článku „Data visualization: definition, examples, tools, advice“ popisuje „Datové umění“ jako nejvyšší formu vizualizace kde „*Získávání informací a postřehů již nestačí, výsledek musí být také vizuálně atraktivní…* “ [3] v rozhovoru s Nicholasem Rougeux popisuje, že „*Datový umělec se nesnaží pouze informovat, ale hlavně vyvolat emoce*“. [4]

Většina zdrojů se však shoduje na dvou základních konceptech „redukce“ a „prostoru“, které popisuje i Lev Manovich v článku „What is Visualization?“: „*Vizualizace informací využívá jednoduchých grafických elementů jako zástupce reálných objektů a vazeb mezi nimi – nezáleží na tom, jestli se jedná o lidi, ceny na burze, příjmy států, nezaměstnanost, nebo cokoliv jiného. Skrze tyto grafické elementy informační vizualizace odhaluje vzorce a struktury v datech. Nicméně, cenou za tuto možnost je extrémní úroveň schematizace. Zahazujeme 99 % toho, co je na objektech specifické, ve snaze nalezení vzorců na zbývajícím 1 % charakteristik.*“ [7]

„*Co mají všechny vizualizační techniky společné, kromě redukce? Všechny využívají prostoru (umístění, velikosti, tvaru a v poslední době i zakřivení a pohybu) pro reprezentaci klíčových rozdílů v datech a zobrazení nejdůležitějších vzorců a vztahů. […] Ostatní, méně důležité vlastnosti objektů jsou pak reprezentovány jinými vizuálními prostředky – odstínem, barvou, vzorem stínování nebo i průhledností.* “ [7]

### Principy datové vizualizace

Základním cílem vizualizace je zjednodušit porozumění datům, ať už hmatatelným (jako jsou počty produktů ve skladu, hladiny řek v čase, nebo rozmístění zastávek autobusu), ale i čistě abstraktním, které nemají fyzickou podobu a jsou tak složitější k pochopení. Aby bylo možné lépe vytvářet grafy zobrazující data srozumitelným způsobem, je nejprve nutné pochopit jakým způsobem člověk zpracovává zrakové vjemy. K tomuto účelu může dle Encyclopedia of Human-Computer Interaction využít zákonů Gestaltismu (česky také „Tvarové psychologie“). Konkrétně jsou při vytváření vizualizací uplatnitelné následující principy:

**Princip blízkosti** – Objekty nacházející se blízko u sebe jsou vnímány jako skupina

**Princip podobnosti** – Objekty stejného, nebo podobného tvaru, barvy či jiných vlastností jsou vnímány jako skupina

**Princip ohrady** – Objekty ohraničené čarou, nebo na stejnobarevném poradí jsou považovány za skupinu

**Princip uzavření** – Vnímání tvaru jako celku i v případě, že obsahuje mezery

**Princip návaznosti** – Objekty zarovnané do čáry/křivky jsou považovány za jeden celek

**Princip spojitosti** – Všechny objekty spojené čarou jsou považovány za jeden celek [18]

Využitím těchto principů lze čtenáře upozornit na konkrétní závislosti nalezené v grafu, ale také vyhledat a upravit části vizualizace, které by mohly vést k nepravdivým závěrům.

### Barvy v datové vizualizaci

Dalším důležitým aspektem vizualizace jsou použité barvy a barevné palety. Ačkoliv mnoho zdrojů se shoduje, že účelem barev není estetika, i ta je pro mnohé uživatele aspektem pro volbu specifických barev v grafu. [20]

Barevné palety lze rozdělit do třech základních skupin. Každá z těchto skupin je vhodná pro vizualizaci jiného typu dat. Sekvenční palety se obvykle skládají z různých odstínů jedné barvy, které se liší svým jasem. Taková paleta je vhodná pro data, která jsou nějakým způsobem příbuzná, protože dle principu podobnosti z kapitoly 4.1.2. budou vnímána jako jedna skupina. [19] Zvláštní podmnožinou sekvenčních palet jsou takzvané „pro lidské vnímání uniformní“ (ang. „perceptually uniform“) barevné palety, kde není využita pouze jedna barva, ale několik barev s proměnlivým jasem tak, aby kontrast mezi stejně vzdálenými hodnotami byl stejný ve všech částech palety a předešlo se tak ztrátě informací, nebo naopak nalezení zdánlivých anomálií v částech s vyšším vnímaným kontrastem. [21]



Obr. 1 Pro lidské vnímání uniformní sekvenční palety knihovny Matplotlib

Zdroj: <https://matplotlib.org/stable/tutorials/colors/colormaps.html>

Dalším typem jsou divergující (rozcházející se) barevné palety, které přechází z jedné kontrastní barvy do druhé přes „inflexní bod“ v jejich středu (obvykle se jedná o velmi světlou barvu, často bílou). Tato barevná paleta je vhodná pro data, která se nějakým způsobem dělí na dvě hlavní skupiny (například kladné a záporné hodnoty, nebo hodnoty, které se dají prezentovat jako positivní a negativní). [19] [20]

Posledním typem jsou kategorické (někdy také kvalitativní, či nominální) barevné palety, používané pro data, které nemají žádné souvislosti a jsou navrženy tak, aby je nebylo možné vnímat jako posloupnost. [20] Ačkoliv některé z těchto palet mohou být velmi obsáhlé, pro zachování přehlednosti se doporučuje nepoužívat více než 8 různých kategorií. [19]

Při výběru palety je také nutné myslet na to, jak budou barvy na čtenáře působit. Dle průzkumu popsaném v práci „Affective Color in Visualization“ je možné rozdělit barevné palety do skupin podle pocitu, který vyvolávají. Konkrétně uklidňující palety se skládají převážně ze světlých, chladných barev s nízkou sytostí (například různé odstíny modré a zelené). Naopak vzrušující se vyznačují vysokou sytostí a využívají hlavně teplých barev, jako je červená, žlutá, nebo oranžová. Asi nejdůležitějším aspektem je, jestli daná vizualizace celkově působí positivním, nebo negativním dojmem. Jako positivní jsou často vnímány syté barvy, často obsahující odstíny modré, zelené, ale i žluté či oranžové, naopak negativně jsou vnímány tmavé barvy, hlavně odstíny červené, hnědé a šedé. [22]



Obr. 2 Skupiny barev dle asociovaných pocitů

Zdroj: Lyn Bartram, Abhisekh Partra a Maureen Stone: Affective Color in Visualization

### Anatomie grafu

V této kapitole budou popsány základní součásti grafu, společné pro většinu typů. Elementy specifické pro určité typy grafů budou popsány v následující kapitole.



Obr. 3 Obecný graf

Zdroj: vlastní zpracování – Knihovna Matplotlib

Nadpis grafu (1) může být umístěn kdekoliv na ploše grafu, zpravidla se však nachází nad zobrazovanou oblastí grafu. Osa Y, nejčastěji svislá, (3) zobrazuje hodnoty závislé na ose X (2). U některých grafů však hodnoty na osách nemusí být závislé. Nadpisy os (4 a 5) mohou poskytnout čtenáři více informací o zobrazovaných datech, jako například jednotky. Legenda (6) slouží k popsání barev, které byly přiřazeny zobrazovaným datovým řadám. Mřížka (7) prodlužuje značky os pro snadnější určení hodnot jednotlivých datových bodů (8).

### Nejčastější typy grafů

Existuje mnoho způsobů, jak rozdělit typy grafů do skupin, ať už podle grafických elementů které využívají (body, čáry, plochy), oborů ve kterých se převážně využívají (finance, věda, zpravodajství), v této kapitole budou grafy rozděleny do skupin dle jejich funkce, stejně, jako je rozděluje Mike Yi v příručce How to Choose the Right Data Visualization [23], nebo Severino Ribecca v projektu Data Visualisation Catalogue [24].

#### Grafy vyjadřující změnu hodnoty v čase

Častým využitím datové vizualizace je porovnat jednu, nebo více proměnných, které se mění v čase. Pro takové případy je vhodné využít bodový, spojnicový, nebo plošný graf v závislosti na počtu měření a sledovaných veličin. Plošný graf je vhodnější pro vyjádření změny v čase, nebo rozdílu mezi několika hodnotami, spojnicový naopak více zvýrazňuje konkrétní hodnoty [24] [25]



Obr. 4 Ukázka grafů vyjadřujících změnu v čase

Zdroj: vlastní zpracování – Knihovna Matplotlib

Pro vizualizaci změny za delší časová období, lze také využít sloupcový graf, kde každý sloupec vyjadřuje součet, nebo průměr za dané období. V případě, že je nutné předat více informací o nasbíraných datech, je také možné využít krabicový graf. [23]



Obr. 5 Ukázka využití sloupcového a krabicového grafu pro vyjádření změny v čase

Zdroj: vlastní zpracování – Knihovna Matplotlib

#### Grafy vyjadřující poměr hodnot

Dalším velmi častým úkolem vizualizace je zobrazení poměru různých kategorií, nebo jejich rozložení v rámci celku. Asi nejčastějšími příklady grafů využívajících kategorií jsou sloupcový a koláčový graf, kde koláčový také ukazuje poměr vůči celku, zatímco u sloupcového obvykle není tato informace příliš patrná. Mezi další specifické druhy těchto grafů patří například takzvaný „donut“, což je koláčový graf s chybějícím středem. Takové uspořádání dovoluje umístit popisky do středu a vytvořit tak kompaktnější vizualizaci, zároveň napomáhá při porovnávání více grafů tohoto typu, protože kladou menší důraz na plochu. [24] Některé zdroje však nedoporučují využívat koláčových grafů a „donutů“ a tvrdí, že pro lidské vnímání je složité překládat úhly na hodnoty. [25]

Detailnějším grafem pro zobrazení poměrů hodnot, které tvoří určitou stromovou hierarchii (například rozložení dat v souborovém systému, nebo zisky z různých odvětví průmyslu) je takzvaná stromová mapa:



Obr. 6 Ukázka využití stromové mapy pro zobrazení souborového systému

Zdroj: vlastní zpracování – knihovna Pygal

#### Grafy vyjadřující rozdělení hodnot

Pro vizualizaci rozdělení hodnot lze využít již zmíněné sloupcové grafy, ale také jim velmi podobné histogramy. V některých případech jsou tyto typy grafů téměř identické, ale v případě, že histogram obsahuje pouze numerické hodnoty, může u některých histogramů být proměnlivá šířka sloupce v závislosti na velikosti intervalů (někdy také nazývaných „třídy“), nebo může být zobrazen jako plynulá křivka.



Obr. 7 a 8 Histogram v podobě sloupcového grafu s kategoriemi intervalů (červený) a histogram s proměnlivou šířkou sloupce (modrý)

Zdroj: vlastní zpracování – knihovna Pygal

Dalším již zmiňovaným grafem sloužícím k vizualizaci rozložení je krabicový graf, nejčastěji používaný ve statistice. Jednotlivé jeho části ukazují medián, kvartily, horní a dolní mez a případné odlehlé hodnoty. [24]



Obr. 9 Krabicový graf doplněný o popis součástí

Zdroj: vlastní zpracování – knihovna Matplotlib

Zajímavou kombinací krabicového grafu s křivkou ukazující hustotu pravděpodobností hodnot je takzvaný houslový graf/diagram. Stejně jako krabicový graf zobrazuje medián a kvartily, pomocí značek ve středu „houslí“. V některých případech nemusí být symetrický, pokud data mohou být dále dělena do dvou kategorií. Z tohoto grafu je také dobře porovnatelné rozložení hodnot mezi více skupinami.



Obr. 10 Houslový graf s nesymetrickým rozdělením podle skupin

Zdroj: vlastní zpracování – knihovna Seaborn

### Časté chyby při vizualizaci

#### Nevhodné použití spojnicového grafu

Při menším množství měření je vhodnější použít bodový graf, protože spojnicový graf by mohl zkreslovat průběh dat, jak je vidět na následujícím příkladu:



Obr. X Ukázka zavádějícího spojnicového grafu

Zdroj: vlastní zpracování – Knihovna Matplotlib

Při měření nebyly zaznamenány hodnoty pro 2,3 a 4 časovou jednotku. Při vizualizaci jako spojnicový graf je spojena hodnota 1 v čase 1 a hodnota 2 v čase 5 (červená čára) a zdá se hodnota lineárně rostla v tomto období. Ve skutečnosti však hodnota zůstala po první 4 jednotky času konstantě (zelená čára). V takových případech je lepší použít bodový graf, který zobrazuje pouze naměřené hodnoty a nevytváří zdání kontinuálního měření. Pro spojení takových bodů je vhodnější vypočítat spojnici trendu, která bude lépe odpovídat realitě:

Obr. X Ukázka využití spojnice trendu

Zdroj: vlastní zpracování – Microsoft Excel

#### Nevhodná volba barevné palety

Volba barevné palety je velmi důležitým krokem při tvorbě jakékoliv vizualizace. Jak již bylo zmíněno v předchozích kapitolách, tato volba může pomáhat s nalezením závislostí a vlastností zobrazovaných dat, zvýraznit nejdůležitější poznatky čtenáři, nebo dokonce přidat emocionální prvek. Je tedy zároveň velmi snadné vybrat barevnou paletu, která bude pro daný typ vizualizace nevhodná, bude zakrývat důležité informace, nebo působit rušivým dojmem. Asi nejčastěji používanou barevnou paletou nevhodnou pro vizualizaci většiny dat je duhová paleta. Tato paleta s sebou nese hned několik problémů: Pořadí barev je sice všeobecně známé, ale nelze intuitivně rozlišit vyšší a nižší hodnoty. Druhým problémem je vnímání rozdílu mezi barvami, například odstíny světle modré působí dojmem „rychlejší“ změny než odstíny zelené. Posledním důležitým problémem, který není zdaleka omezený na duhovou paletu, je složité vnímání takových palet pro čtenáře se zhoršeným barvocitem. [26] Navzdory těmto chybám jsou duhové barevné palety často součástí vizualizačních programů a knihoven.



Obr. X Porovnání barevných palet

Zdroj: vlastní zpracování – Knihovna Matplotlib

Grafy na obrázku X jsou vytvořeny na základě stejných dat (10000 hodnot normálního rozdělení pro osu X a x + náhodná hodnota normálního rozdělení pro osu Y), ale využívají různé barevné palety. Horní graf využívá duhovou paletu „gist\_rainbow“, zatímco dolní využívá pro lidské vnímání uniformní paletu „inferno“, obě dostupné v knihovně Matplotlib. Červená barva pro nulové hodnoty horního grafu působí velmi rušivým dojmem. Odstíny modré a zelené ve středu grafu vytváří zdání skupin s rozdílnými vlastnostmi na základě principu podobnosti popsaného v kapitole 4.1.2. ačkoliv dle dolního grafu je patrné, že zde dochází k plynulému zvyšování četnosti hodnot, bez výrazného zlomu, jak by horní graf naznačoval.

## Jazyk Python

Jazyk Python je interpretovaný objektově orientovaný programovací jazyk, který v dnešní době patří mezi jeden z nejpoužívanějších. V listopadu roku 2021 byl na prvním místě žebříčků PYPL a TIOBE index [27] [28], hodnotící programovací jazyky podle počtu kurzů pro daný jazyk vyhledaných online a počtu programátorů daný jazyk využívající. Využití nachází ve vědě a výzkumu, výuce, webových aplikacích, ale i při vývoji jiných aplikací pro testování a management. [30]

Jeho popularita se zakládá hlavně na jeho jednoduchosti, dobré čitelnosti kódu, přenositelnosti programů mezi platformami, velkému množství rozšiřujících knihoven a snadné integraci s komponentami v jiných jazycích. Nevýhodou jazyku Python je především jeho rychlost, která je oproti kompilovaným jazykům nižší, což však nemusí být problémem při mnoha využitích tohoto jazyka. Problém s rychlostí lze také řešit oddělením náročnějších částí programu do kompilovaných rozšíření, následně volaných z hlavního interpretovaného programu. [29] Další možnou nevýhodou může být pro některé programátory nezvyklá syntaxe využívající odsazení kódu a konce řádků

## Knihovny jazyka Python pro zpracování dat

### NumPy

Asi nejčastěji používanou rozšiřující knihovnou jazyka Python je NumPy. Jedná se o knihovnu přidávající podporu n-dimenzionálních polí a práce s nimi. Na rozdíl od seznamů jazyka Python mají tato pole definovanou velikost a datový typ, tento rozdíl je způsoben kompilovaným kódem jazyka C, který NumPy využívá pro velmi rychlé výpočty s velkými objemy dat. Právě díky této rychlosti se tato knihovna stala velmi oblíbeným nástrojem pro vědecké výpočty a je využívána velkým množstvím dalších knihoven. Kromě základní manipulace n-dimenzionálních polí a výpočtů s nimi nabízí tato knihovna také efektivní algoritmy pro jejich třídění, výpočty lineární algebry, statistické operace, diskrétní Fourierovy transformace i generování náhodných rozložení. Pro složitější operace je možné využít nadstavby SciPy. [31] [29]

### SciPy

SciPy je rozšířením knihovny NumPy, zaměřujícím se převážně na efektivní implemetaci složitějších matematických algoritmů v oblastech lineární algebry (včetně řešení rovnic), statistiky, optimalizačních úloh, interpolace, výpočtu určitých integrálů. Krom toho umí SciPy také pracovat se signály a obrazovými daty (filtrování, konvoluce, gradientní operátory atd.) a nabízí i možnost práce se soubory programu MATLAB. [32]

### pandas

Často využívanou knihovnou při tvorbě vizualizací je knihovna pandas, sloužící k manipulaci s tabulkovými datovými strukturami pomocí objektů Series (pro jednodimenzionální data stejného typu) a DataFrame (pro obecná tabulková data). Dovoluje snadno přidávat a odebírat sloupce dat, kombinovat datové soubory nebo vytvářet výběry na základě podmínek. Velmi praktická je také možnost vytvářet objekty DataFrame přímo ze souborů a databází, konverze textových záznamů na datové typy vyjadřující čas a práce s časovými řadami. Knihovna také disponuje schopností vytvářet vizualizace přímo z objektů DataFrame pomocí knihovny Matplotlib, častěji je však DataFrame použit jako zdroj dat pro jiné vizualizační knihovny. [33]

## Vizualizační knihovny jazyka Python

### Matplotlib

Pravděpodobně nejznámější vizualizační knihovna jazyka Python byla vytvořena okolo roku 2003 Johnem Huterem byla původně určena k vizualizaci dat zaznamenaných elektrokortikografií při výzkumu epilepsie. Laboratoř, ve které John Hunter pracoval, měla v té době pouze jednu licenci na softwarový balíček pro analýzu dat, o který se museli všichni výzkumníci dělit. Hunter se tedy rozhodl vytvořit náhradu v prostředí MATLAB. Tato aplikace však nebyla ideální pro vizualizaci dat z mnoha zdrojů (kromě elektrokortikografie například i EEG a magnetické rezonance) uložených na několika serverech. Hunter proto začal vyvíjet novou aplikaci v jazyce Python „EEG viewer and analyzer“, která se postupem času změnila v dnešní knihovnu matplotlib. [5]

V současné době se jedná o volně dostupný open-source projekt, sloužící k tvorbě statických, animovaných i interaktivních vizualizací nejčastěji ve 2D (knihovna však podporuje i trojrozměrné vizualizace). [6]

### Seaborn

Seaborn vznikl jako řešení nejčastějších nedostatků knihovny matplotlib, konkrétně defaultních nastavení vizualizace před verzí 2.0, která byla založena na vizualizacích MATLABu, nižší úrovně API u matplotlibu, která často měla za následek nadbytečný kód a složitost využití datových struktur knihovny pandas. [9]

Seaborn buduje na základech knihovny matplotlib, ke které poskytuje vysokoúrovňové rozhraní, také dokáže využívat datové struktury knihovny pandas. Seaborn na základě specifikace typu grafu zvládne automaticky provázat hodnoty v datech s vizuálními atributy, jako je barva, velikost a styl, propočítat statistické transformace a doplnit ke grafu informativní štítky a legendu. Díky tomu, že Seaborn dokáže vytvořit kompletní vizualizace jedním voláním funkce s minimálním počtem argumentů, je ideálním nástrojem pro explorační analýzu dat. [8]

### Bokeh

Stejně jako matplotlib je knihovna Bokeh open source projektem, finančně podporovaným neziskovou organizací NumFOCUS, která se zaměřuje na financování a propagaci, open source nástrojů ve vědě a výzkumu. [10]

Oproti knihovnám matplotlib a seaborn však Bokeh nevytváří kompletní grafy jednou funkcí, místo toho nabízí širokou škálu nástrojů pro manipulaci s elementy vektorové grafiky (nazývanými „glyphs“), ze kterých lze následně vytvářet jednotlivé vrstvy vizualizace. To dává uživateli mnohem větší kontrolu nad vzhledem finální vizualizace, za cenu větší složitosti kódu, potřebného na její vytvoření. Bokeh dále dovoluje do vizualizace přidávat interaktivní elementy, jako slidery, tlačítka nebo drop-down menu. Knihovna také umožňuje vytvoření „Bokeh serveru“, který zajišťuje možnost streamování dat a složitější uživatelské interakce.

Knihovna se vlastně skládá ze dvou, Bokeh pro Python – sloužící k vytváření vizualizací a BokehJS – knihovna jazyka JavaScript sloužící k renderování vizualizace a zajištění interaktivity ve webovém prohlížeči. [11]

### Plotly

Další open source knihovnou pro datovou vizualizaci je Plotly. Tato knihovna však není omezena pouze na jeden jazyk, lze ji používat v jazycích Python, R, Julia, MATLAB a existují i projekty zpřístupňující ji v jazyce Java a jazycích frameworku .NET. Implementace v jazyce Python je však její zdaleka nejoblíbenější variantou a existují zde dvě možnosti využití knihovny. Jednodušší vysokoúrovňový Plotly Express, který s minimem kódu vytváří kompletní vizualizace a složitější modul Plotly Graphic Objects, který však dovoluje mnohem větší kontrolu nad vytvářenou vizualizací a je pomocí něho možné vytvořit některé typy grafů, nepodporované modulem Express. [12]

Existují také komerční produkty založené na této knihovně, Dash Enterprise - platforma pro analýzu trhu, datové vědy a výzkum v oblasti umělé inteligence a Chart Studio Enterprise – nástroj pro rychlou tvorbu vizualizací přímo ze souborů nebo databází a jejich vkládání do webových stránek. [12]

### Holoviews

Na rozdíl od předchozích vizualizačních knihoven, není knihovna Holoviews schopna sama vizualizovat data a je v tomto ohledu závislá na knihovnách Matplotlib, Bokeh a Plotly. Místo toho se Holoviews soustředí na co možná nejjednodušší manipulaci s grafy jako objekty, které jsou až do chvíle zobrazení nezávislé na jakékoliv vizualizační knihovně. Tyto objekty je také možné skládat do složitějších vizualizací pomocí seznamů a jednoduchých operátorů jako + (pro rozložení grafů vedle sebe) nebo \* (pro překrytí grafů). Práce s daty k vizualizaci je také zjednodušena, u většiny typů grafů stačí data popsat a předat konstruktoru objektu grafu informace o veličinách, které mají být vizualizovány. Dle popisu knihovny tato zjednodušení dovolují uživateli „*soustředit se na to, jakým způsobem chce data prozkoumat a co se snaží ukázat namísto samotného procesu tvorby grafu*“ [13]

### Pygal

Knihovna Pygal je poslední z častěji používaných vizualizačních knihoven jazyka Python. Zaměřuje se převážně na jednoduchou tvorbu interaktivních vizualizací, ideálních pro vložení do webových stránek díky exportu do formátu vektorové grafiky. Nabízí 14 různých typů grafů, včetně geografických vizualizací a relativně široké možnosti přizpůsobení včetně vestavěných stylů a možnosti definovat vlastní styly. [14]

### MidiTime

Knihovna MidiTime neslouží ke klasické vizualizaci dat, naopak se zabývá oborem, který by se dal nazvat „sonifikace dat“. Jedná se o vyjádření dat rozložených v čase jako zvuk (v případě této knihovny jde o formát MIDI, který je možné využít v syntetizátorech a jiných elektronických nástrojích). Knihovnu vytvořil Michael Corey v roce 2015 za účelem znázornění počtu zemětřesení v americkém státě Oklahoma pro rádiové zpravodajství. [15] Takové znázornění dat by mohlo najít využití nejen pro média, jako je rádio nebo podcast, ale mohlo by přispět i k přístupnosti informací na webových stránkách pro lidi se zrakovým postižením nebo dokonce ke tvorbě hudby.

### Geoplotlib

Ačkoliv některé z předcházejících knihoven podporují zobrazování dat na mapě, Geoplotlib se specializuje pouze na zobrazování geografických dat využívající OpenStreetMap jako podklad a OpenGL pro rychlé renderování grafických elementů na mapový podklad. Knihovna podporuje několik různých typů grafů jako body, teplotní mapy, histogramy, nebo i vykreslení tvarů v podobě geografických shapefiles a geojson. [16]

### WordCloud

Poslední vizualizační knihovna se zabývá spíše estetickou než informativní vizualizací, konkrétně v podobě takzvaného Word cloudu (v češtině někdy také „Slovního mraku“). Jedná se o vizualizaci založenou na četnosti slov v textu. Čím častěji se slovo v textu vyskytuje, tím větší bude ve výsledném „mraku“. Kromě velikosti slov dovoluje knihovna WordCloud manipulovat s barvou slov (ať už na základě četnosti, délky, či jiného uživatelem definovaného pravidla) a s celkovým rozvržením „mraku“ pomocí masky založené na libovolném obrázku. [17]

# Metodika zpracování

## Hodnocení knihoven

Aby bylo možné využité knihovny snáze porovnávat, budou jejich funkcionality hodnoceny dle následující tabulky:

Tabulka 1 Ukázková tabulka hodnocení

|  |  |
| --- | --- |
| Závislosti knihovny | Seznam knihoven, které jsou nutné pro fungovaní dané vizualizační knihovny |
| Podporované vstupní formáty | Formáty a datové struktury, které je knihovna schopná zpracovat |
| Podporované výstupní formáty | Formáty, ve kterých je možné vytvořené vizualizace ukládat |
| Poskytované typy grafů | Které typy grafů knihovna poskytuje? Existují nějaké často používané grafy, které v této knihovně chybí? |
| Poskytované možnosti přizpůsobení | Jakým způsobem lze vytvořené vizualizace upravovat? |
| Poskytované možnosti interaktivity | Jakým způsobem může uživatel s vytvořenou vizualizací interagovat |
| Jednoduchost použití | Jaká je složitost potřebného kódu pro vytvoření a přizpůsobení vizualizace? |
| Přehlednost a atraktivita výchozích nastavení | Jak přehledná a vizuálně atraktivní je výsledná vizualizace při minimálním použití přizpůsobení? |

Pro každou obecnou vizualizační knihovnu bude vytvořeno několik skriptů podle společného vzoru (projekt Templates). Budou otestovány podporované vstupní formáty, způsoby exportu vizualizace, nabízené možnosti přizpůsobení a hodnocení schopnosti rozlišit větší množství dat v jednom grafu. Aby bylo možné knihovny objektivněji porovnávat, budou tyto aspekty testovány pomocí často používaných typů grafů podporovaných všemi knihovnami, tedy graf sloupcový, spojnicový a bodový.

# Praktická část

## Způsoby získávání dat pomocí jazyka Python

V následujících kapitolách jsou popsány některé ze způsobů získání dat pro vizualizaci v jazyce Python a knihovny, které lze k tomuto účelu využit.

### Načtení dat ze souboru

Jedním ze způsobů získání dat pro vizualizaci je jejich načtení přímo z pevného disku počítače. V této kapitole je popsána práce alespoň s některými z nejčastějších typů souborů.

#### Sobory typu CSV

Sobory s příponou .csv (comma-separated values) jsou jednoduchým způsobem, jak uchovávat data organizovaná do sloupců a jsou hojně využívány například pro zveřejňování takzvaných „Otevřených dat“. Pro jejich načtení a zpracování je možné využít modul csv jazyka Python. Pomocí csv.reader() je vytvořen objekt, přes který lze iterovat v cyklu. Každá iterace vrací jeden řádek CSV souboru ve formátu seznamu (pokud soubor obsahuje hlavičku, nachází se na prvním místě a je s ní pracováno jako s jakýmkoliv jiným řádkem souboru). Následné zpracování dat je tedy poměrně náročné na množství potřebného kódu a při velkém množství dat i na rychlost.

Snazším a rychlejším způsobem práce se soubory ve formátu CSV je využití knihovny pandas. Pomocí volání pd.read\_csv() lze ze souboru vytvořit DataFrame, se kterým lze dál snadno manipulovat. Není nutné soubor otevírat pomocí funkce open(), pro načtení stačí cesta. Důležitým rozdílem oproti csv modulu jazyka Python je nutnost specifikovat separátor pomocí sep='<znak>', pokud je pro oddělení hodnot použit jiný znak než čárka.

#### Bitmapové soubory

Pro načtení obrazových dat ve formátech jako jsou .png, .jpg, apod. je možné použít knihovnu Python Imaging Library, respektive její aktuálně používanou verzi Pillow. Po načtení pomocí PIL.Image.open() získáme objekt typu Image, který lze dále zpracovávat pomocí knihovny Pillow. Dostupné jsou funkce jako ořezávání, změna velikosti, geometrické transformace a kombinace více souborů. Asi nejdůležitějšími jsou však metoda .convert() pro konverzi reprezentací pixelů mezi RGB, CMYK nebo převedení na odstíny šedi a možnost převést obrazová data na vícerozměrné pole knihovny NumPy pomocí np.array(). Tvar a uspořádání pole závisí na zvolené reprezentaci pixelů. V případě více kanálů jsou pixely polem hodnot, zatímco u obrazových dat konvertovaných na odstíny šedi jsou pixely tvořeny jedinou hodnotou. Takto vytvořená pole je možné dále zpracovávat manuálně, nebo využít knihoven jako je například SciPy. Zpracovaná data lze převést zpět na Image objekt pomocí PIL.Image.fromarray() a následně uložit, nebo vizualizovat pomocí vhodné knihovny, například Matplotlib (pro data ve formě odstínů šedi lze využít pseudobarev specifikací ColorMap).

#### Zvukové soubory

Vizualizace zvuku může být praktická v některých oblastech výzkumu, ale může sloužit i k čistě estetickým účelům. Knihovna Librosa je jednou z možných knihoven pro práci se zvukovými soubory. Kromě jejich načtení jako pole knihovny NumPy pomocí librosa.load() umí také provádět operace s načtenými daty, jako krátkodobé Fourierovy transformace, nebo dokonce generovat nové signály. Krom toho lze pomocí knihovny Librosa vizualizovat načtený zvuk jako průběh signálu v čase, nebo spektrogram. Vizualizace jsou založeny na knihovně Matplotlib a lze je provést pomocí librosa.display.waveshow() a .specshow().

Krom této knihovny lze také využít SciPy, konkrétně scipy.io.wavfile.read() který je schopen přečíst soubory ve formátu .wav a vrátit ho také jako NumPy pole. Výběr knihovny záleží převážně na způsobu dalšího zpracování dat.

### Načtení dat z webové stránky

Hojně využívaným nástrojem pro práci s obsahem webových stránek (především výběr a zpracování dat, někdy také „web scraping“) je knihovna BeautifulSoup. Díky ní lze snadno procházet strukturu webové stránky, vyhledávat specifické elementy, třídy a identifikátory a extrahovat z nich text, nebo atributy (například atribut href v případě elementu odkazu může být užitečný pro další automatické procházení stránky).

V případě načítání tabulky z webové stránky není nutné parsovat jednotlivé řádky a sloupce, stačí tabulku načíst pomocí BeautifulSoup a předat ji v podobě řetězce metodě pd.read\_html() knihovny pandas, která z tabulky vygeneruje DataFrame. Pokud má tabulka hlavičku, pandas se pokusí tyto řetězce dosadit jako názvy sloupců.

### Načtení dat z databáze

Způsoby získání dat z databáze se pochopitelně liší dle použitých technologií. Pro ukázku byly zvolena MySQL databáze, jako reprezentace relační databáze a nerelační (NoSQL) MongoDB.

**MySQL**

Pro interakci s libovolnou databází je nejprve nutné získat odpovídající ovladač. V případě MySQL se jedná o oficiální connector dostupný z webových stránek MySQL. Dalším krokem po importu ovladače je vytvoření spojení s databází, zde pomocí mysql.connector.connect(), kde je nutné doplnit parametry s adresou, uživatelským účtem a databází. Dotazy na databázi jsou pak vytvářeny jako string, do kterého lze doplnit zástupné symboly %s pro pozdější doplnění parametrů, například z uživatelského vstupu. Hodnoty načítané z databáze jsou vraceny jako uspořádané n-tice (Tuple) představující jeden řádek, popřípadě jejich seznam.

**MongoDB**

Pro připojení k MongoDB databázi je vhodné využít ovladače pymongo. Podobně jako u MySQL je prvním krokem vytvoření spojení s databází, zde pomocí pymongo.MongoClient(), kde argumentem je „connection string“ obsahující adresu, uživatelské jméno a databázi k otevření. Každá databáze MongoDB se může skládat z mnoha kolekcí. Výběr databáze i kolekce lze provádět buď pomocí tečkové notace (client.nazev\_databaze a nazev\_databaze.nazev\_kolekce), nebo pomocí (client["nazev\_datanaze"]) v případě, že název by nebylo možné zapsat do kódu. Operace s kolekcemi se pak provádí voláním odpovídajících funkcí, jako .find(), .insert\_one(). Záznamy z databáze vrácené po vyhledání se chovají jako slovníky, lze tedy snadno přistupovat k jejich atributům.

### Načtení dat pomocí API

Pro dotazy na Application Programming Interface (API) přes http protokol slouží modul jazyka Python requests. Získání dat pak probíhá pomocí metody requests.get() kde argumentem je url dotazu (pro tvorbu je vždy nutné konzultovat dokumentaci dotazovaného rozhraní). Vrácen je objekt Response, který nese data, a některé informace jako kódování, čas potřebný pro dokončení dotazu a asi nejdůležitěji stavový kód (200 v případě úspěšného dokončení). Aplikační rozhraní s architekturou REST, vrací odpovědi v podobě JSON nebo vzácněji XML dokumentu. K datům ve formátu JSON lze pak v jazyce Python snadno přistupovat pomocí indexů hodnot, protože se chová jako slovník.

## Práce s načtenými daty

## Porovnání vizualizačních knihoven

### Matplotlib

#### Závislosti

Knihovna Matplotlib má následující závislosti:

* FreeType –sloužící k renderování fontů
* libpng –pro manipulaci se soubory typu png
* NumPy – knihovna pro práci s daty ve formátu vícerozměrných polí
* cycler – objekt pro nekonečné cyklické procházení seznamů
* dateutil – rozšíření DateTime modulu jazyka Python
* kiwi – rychlé řešení soustav rovnic a nerovnic

#### Vstupní formáty

Matplotlib zvládne data zpracovávat jako standardní seznamy a uspořádané n-tice (tuple) jazyka Python i pole knihovny Numpy. Objekty typu DataFrame z knihovny Pandas zvládne zpracovat přímo pouze pro některé typy grafů, u jiných (například u sloupcového grafu), lze volat metodu .plot() tohoto objektu, která vrátí objekt Axes, který lze snadno vložit jako podgraf do vytvářené vizualizace. Pro vytvoření grafu nelze přímo použít slovník jazyka Python, je nutné jeho části předat jako seznamy.

#### Výstupní formáty

Knihovna Matplotlib zvládne vytvořené vizualizace zobrazovat do okna, které nabízí uživateli určitou míru interaktivity (přibližování grafů, pohyb s osami, úprava podgrafů), export do souboru lze provést přímo z kódu voláním metody plt.savefig(), nebo z již zmiňovaného okna přes souborový dialog. Matplotlib nabízí široký výběr formátů, od často používaných jako jsou png, jpeg, svg, pdg až po vzácnější formáty, například raw, rgba a tiff.

#### Typy grafů

Podporovány jsou nejrůznější typy dvourozměrných vizualizací od jednoduchých grafů po vizualizace založené na obrazových datech, ale i specializované typy grafů jako je Sankeyův diagram. Vytvářet lze i velké množství trojrozměrných vizualizací. V případě, že nabízené typy vizualizací nejsou dostačující, lze vytvářet i vlastní skládáním existujících vizualizací a manipulací se základními geometrickými tvary.

#### Přizpůsobení vizualizací

Možnosti přizpůsobení vizualizací jsou velmi hluboké, od rychlého přepnutí celkového stylu grafu, přes jednoduché změny velikosti a barvy objektů při vytváření grafu, až po možnost jednotlivě manipulovat s libovolným elementem hotové vizualizace. Je možné snadno měnit barvu a velikost elementů v závislosti na hodnotách, díky speciálním argumentům a tzv. „colormaps“, které definují gradient barev, ze kterého lze vybírat na základě zobrazovaných dat.

Možnosti přizpůsobení mohou být někdy mírně nekonzistentní ve způsobu jejich použití, například při vytváření sloupcového grafu, lze nastavit barvy jednoduchým doplněním seznamu barev do argumentu „color“, stejným způsobem však nelze použít seznam šrafování pro argument „hatch“, ten přijímá pouze jeden typ šrafování. Pro použití různých šrafování je nutné vytvořit více grafů v jednom diagramu, nebo přistupovat přímo k vlastnost „hatch“ u jednotlivých grafických elementů grafu.





Obr. X a Y Možnosti rozlišení hodnot v knihovně Matplotlib

Zdroj: vlastní zpracování

#### Interaktivita

Jak již bylo zmíněno určitá míra interaktivity je možná pomocí zobrazení v okně knihovny Matplotlib. Pro hlubší interaktivitu je možné využít událostí, které dovolují detekovat klikání, pohyb a tažení myši a reagovat na ně. Je také možné opakovaně volat „animační funkci“ pozměňující zobrazovaná data a vytvářet tak animované grafy.

#### Jednoduchost použití

Knihovna Matplotlib je velmi jednoduchá na použití, k vytvoření vizualizace obvykle není třeba více než jen několik řádků kódu. Až na vzácné případy, jako například šrafování popsané v dřívější kapitole, se způsob volání metod a dosazování argumentů řídí jednoduchými a snadno pochopitelnými pravidly. Ke knihovně je dostupná obsáhlá a velmi detailně zpracovaná dokumentace popisující všechny její součásti, od grafů jako celku až po vlastnosti a metody jednotlivých částí grafu a grafických elementů. V případě, že by informace v dokumentaci nebyly dostačující, je díky rozšířenosti této knihovny velmi snadné je dohledat na jiných webových stránkách.

#### Výchozí nastavení

Výchozí nastavení knihovny pro většinu typů grafů je dostačující k nalezení některých závislostí mezi zobrazovanými daty, ale v mnoha případech nejsou příliš atraktivní pro čtenáře, vzhled lze však snadno změnit výběrem z mnoha zabudovaných stylů.





Obr. X a Y Výchozí vzhled grafů knihovny Matplotlib

Zdroj: vlastní zpracování

#### Shrnutí

|  |  |
| --- | --- |
| Závislosti knihovny | FreeType, libpng, NumPy, cycler, Dateutil, kiwi |
| Podporované vstupní formáty | Seznam, tuple, Numpy pole, Pandas DataFrame (u některých grafů) |
| Podporované výstupní formáty | png, jpeg, svg, pdf, pgf, ps, tiff, raw, rgba |
| Poskytované typy grafů | Velmi široký výběr 2D a 3D grafů, možnost tvorby vlastních vizualizací |
| Poskytované možnosti přizpůsobení | Široké možnosti úprav všech elementů, colormaps, předem definované styly, anotace a legendy |
| Poskytované možnosti interaktivity | Handlery pro události myši a klávesnice, periodicky volané funkce pro animace |
| Jednoduchost použití | Snadno pochopitelný zápis kódu, velmi dobře zpracovaná dokumentace |
| Přehlednost a atraktivita výchozích nastavení | Výchozí vizualizace nejsou příliš atraktivní, nicméně jsou pro mnoho účelů dostačující |

### Seaborn

#### Závislosti

Seaborne je vizualizační knihovna, založená na knihovně Matplotlib, má tedy i všechny její závislosti. Krom těchto knihoven dále vyžaduje:

* pandas – knihovna pro manipulaci s daty ve formě tabulek (kapitola 4.3.3)
* SciPy – rozšíření knihovny NumPy (kapitola 4.3.2)

#### Vstupní formáty

Oproti knihovně Matplotlib zvládne Seaborn lépe pracovat s daty ve formátu slovníků a DataFrame knihovny pandas. Pro tyto vstupy se při volání metody vytvářející vizualizaci daný objekt dosadí do parametru „data“ a definují se klíče (v případu slovníku) nebo názvy sloupců (u DataFrame) do parametrů „x“ a „y“ pro definici os. U některých typů vizualizací lze dále využít „hue“ a „size“ pro vyjádření dalších dat pomocí odstínu a velikosti grafických elementů. Zvláštností je, že uspořádané n-tice (Tuple) nelze přímo využít při volání vizualizační metody a musí být přetypovány na seznam, ačkoliv tak lze učinit v knihovně Matplotlib.

#### Výstupní formáty

Export vizualizací je založen na exportu z knihovny Matplotlib voláním plt.savefig(). Formáty exportu jsou tedy identické s touto knihovnou.

#### Typy grafů

Seaborn umožňuje generovat mřížky vizualizací třech základních typů: vztahu více proměnných (metoda .relplot()), rozložení hodnot (meotda .displot()) a hodnoty rozložené do kategorií (metoda .catplot()). Krom toho lze využít metod, které vytváří jeden specifický typ grafu. Jedním z často využívaných grafů, který v této knihovně chybí je koláčový graf a jeho varianty.



Obr. X Vizualizace pomocí FacetGrid knihovny Seaborn. Firmy jsou děleny do řádků dle burzy a sloupců dle kategorie

Zdroj: vlastní zpracování

#### Přizpůsobení vizualizací

Pro přizpůsobení vytvářených vizualizací lze využít všech parametrů a funkcí dostupných v knihovně Matplotlib. Seaborn samotný navíc umožňuje rychlý výběr vzhledů a okolo 180 barevných palet s možností definice vlastních a úpravy existujících.

#### Interaktivita

Seaborn nerozšiřuje interaktivitu knihovny Matplotlib, jsou tedy dostupné stejné funkce a události.

#### Jednoduchost použití

Seaborn zjednodušuje práci hlavně s objekty typu DataFrame a umožňuje rychle a jednoduše porovnávat obsáhlé soubory data pomocí FacetGrid vizualizací, které lze vytvořit jediným řádkem kódu. Díky základu v knihovně Matplotlib je způsob tvorby vizualizací snadno pochopitelný, pokud uživatel již má zkušenosti s touto knihovnou.

#### Výchozí nastavení

Výchozí vzhled Seaborn připomíná knihovnu Matplotlib. Pro lépe vypadající vizualizace lze zvolit styl voláním .set\_theme(). Pokud není specifikován parametr, je nastaven „výchozí“ vzhled knihovny Seaborn, který se až do verze 0.8 automaticky aplikoval při importu knihovny.

#### Shrnutí

|  |  |
| --- | --- |
| Závislosti knihovny | SciPy, pandas, závislosti knihovny Matplotlib |
| Podporované vstupní formáty | Seznam, slovník, NumPy pole, Pandas DataFrame, |
| Podporované výstupní formáty | png, jpeg, svg, pdf, pgf, ps, tiff, raw, rgba |
| Poskytované typy grafů | Většina základních 2D grafů kromě koláčového |
| Poskytované možnosti přizpůsobení | Stejné jako u knihovny Matplotlib, rozšířené o palety a styly |
| Poskytované možnosti interaktivity | Stejné jako Matplotlib |
| Jednoduchost použití | Stejné jako Matplotlib. Snadno pochopitelná a přehledná dokumentace |
| Přehlednost a atraktivita výchozích nastavení | Díky jednoduché aplikaci stylů je atraktivnější než u knihovny Matplotlib |

### Bokeh

#### Závislosti

K použití knihovny Bokeh jsou vyžadovány následující závislosti:

* Jinja2 – knihovna pro vytváření šablon
* NumPy – knihovna pro práci s daty ve formátu vícerozměrných polí
* packaging – knihovna nástrojů pro balíčky jazyka Python
* pillow – knihovna pro práci s obrazovými formáty
* PyYAML – parser serializačního jazyka YAML
* tornado – framework pro webové aplikace
* typing\_extensions – zpřístupnění modulu typing pro starší verze jazyka Python

#### Vstupní formáty

Knihovna Bokeh bez problému zpracovává nejrůznější vstupní formáty, ať už přímým zadáním dat ve formě seznamů, polí a n-tic, nebo využitím pojmenovaných dat a argumentu source, kterým lze předat knihovně slovník, DataFrame, nebo i objekt ColumnDataSource, který knihovna nabízí. Využitím tohoto nového objektu lze snadno sdílet data mezi grafy, přidávat i odebírat sloupce, či aktualizovat data v reálném čase.

#### Výstupní formáty

Jak již bylo zmíněno v předchozích kapitolách, Bokeh se skládá ze dvou části, vizualizační knihovny pro jazyk Python a BokehJS, který zajišťuje renderování a interaktivitu v prohlížeči. Hlavním výstupním formátem této knihovny jsou tedy soubory .html. Možný je také export do formátů .png a .svg, který však ke svému fungování potřebuje dodatečné závislosti v podobě knihovny selenium a ovladače (např. geckodriver prohlížeče Firefox).

Vytvořené vizualizace lze také spustit v podobě serveru, což dovoluje značně rozšířit interaktivitu vizualizací.

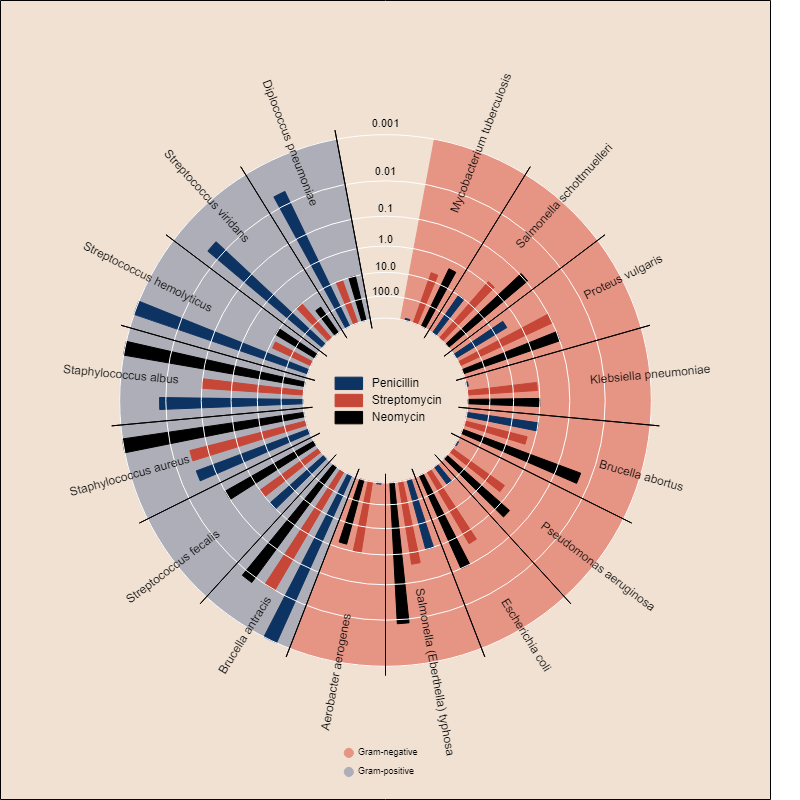
#### Typy grafů

V rámci objektu Figure lze volat funkce pro vytvoření konkrétních vizualizací, jako sloupcového, spojnicového, či bodového grafu. Hlavním způsobem tvorby vizualizací pomocí knihovny Bokeh je však manipulace základních grafických elementů (například pro vytvoření koláčového grafu je nutno využít grafického elementu „Wedge“ a výpočtu správných úhlů na základě dat. Obdobně pro tvorbu histogramu by bylo použito elementů „Quad“, nebo „Patch“).

#### Přizpůsobení vizualizací

Díky možnosti libovolně manipulovat s grafickými elementy vizualizace dovoluje knihovna Bokeh velmi široké možnosti přizpůsobení. Pokud je však důraz kladen na rychlost tvorby grafů, je lepší volbou využít předpřipravených stylů a palet (lze i definovat vlastní pro použití ve více vizualizacích, či v celé organizaci).

Pro vyjádření více proměnných ve vizualizaci jsou k dispozici funkce jako factor\_cmap() nebo factor\_hatch(), dovolující přiřadit barvu či typ šrafování různým hodnotám.



Obr. X Pomocí knihovny Bokeh lze vytvářet velmi složité vizualizace

Zdroj: Dokumentace knihovny Bokeh [11]

#### Interaktivita

Pro vytváření interaktivních vizualizací je možné přidat do výsledné stránky nejrůznější „widgety“, jako tlačítka, slidery, textová pole, nebo menu. V případě, že jsou tyto komponenty nevyhovující, lze také vkládat vlastní widgety vytvořené v jazyce JavaScript.

#### Jednoduchost použití

Oproti předcházejícím knihovnám Matplotlib a seaborn je Bokeh mnohem složitější a vyžaduje větší množství kódu ke tvorbě vizualizace. Ke každé součásti knihovny je však k dispozici rozsáhlá dokumentace včetně ukázek. Všechny objekty a funkce mají také vyplněné detailní informace o jejich správném použití, které je možné zobrazit přímo ve vývojovém prostředí. V některých případech jsou tyto příklady použití zahrnuty i do chybových zpráv, je tedy jednoduché opravit chybu vzniklou špatným použitím některé z funkcí.

#### Výchozí nastavení

Výchozí nastavení na rozdíl od předchozích knihoven žádným způsobem nezasahuje do barev (a to ani po použití předpřipraveného stylu), jsou tedy značně nepřehledné, pokud je takto vizualizováno více elementů v jednom grafu. Vždy je nutné doplnit elementům barvy manuálně, či je přiřadit dle existující palety.



Obr. X Výchozí nastavení knihovny Bokeh může být nepřehledné

Zdroj: vlastní zpracování

#### Shrnutí

|  |  |
| --- | --- |
| Závislosti knihovny | Jinja2, NumPy, packaging, pillow, PyYAML, tornado, typing\_extensions |
| Podporované vstupní formáty | Seznam, n-tice, slovník, NumPy pole, Pandas DataFrame, ColumnDataSource |
| Podporované výstupní formáty | .html, .png, .svg |
| Poskytované typy grafů | Spojnicový, plošný, sloupcový, bodový + tvorba vlastních vizualizací |
| Poskytované možnosti přizpůsobení | Palety, styly, možnost úpravy libovolného elementu |
| Poskytované možnosti interaktivity | Možnost vložení widgetů i definice vlastních |
| Jednoduchost použití | Mnohem složitější než Matplotlib, či seaborn |
| Přehlednost a atraktivita výchozích nastavení | Vždy nutné definovat alespoň barvy pro přehlednost grafu |

### Plotly

### Holoviews

### Pygal

## Ukázkové úlohy

Vlastní text práce.

Tabulka 1 Název tabulky.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Zdroj: citace zdroje, nebo autor, vlastní zpracování

### Podřazená podkapitola

Vlastní text práce.



Obr. 1 Název obrázku/grafu/fotografie.

Zdroj: citace zdroje, nebo autor, vlastní zpracování

# Shrnutí výsledků

Souhrn vlastních výsledků získaných v průběhu řešení problému.

# Závěry a doporučení

Kritická diskuze nad výsledky, ke kterým autor dospěl (soulad výsled-ků  literaturou či předpoklady; výsledky a okolnosti, které zvláště ovlivnily předkládanou práci atd.). Je vhodné naznačit i případné další (popř. alternativní) možnosti zkoumání dané problematiky a otevřené problémy pro další studium.

# Seznam použité literatury

1. PURCHASE, Helen et al. Theoretical Foundations of Information Visualization. In: *Lecture Notes In Computer Science*. 4950. 1970, s. 46–64. ISBN 978-3-540-70955-8. DOI: [10.1007/978-3-540-70956-5\_3](https://doi.org/10.1007/978-3-540-70956-5_3)
2. G. CHIAPPINI a R.M. BOTTINO. *Visualisation in Teaching-Learning Mathematics: The Role of the Computer* [online]. Dostupné z: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.3360&rep=rep1&type=pdf>
3. LORÈNE FAUVELLE. *Data visualization: definition, examples, tools, advice [guide 2021]* [online]. 2020 [cit. 03.04.2021]. Dostupné z: <https://www.intotheminds.com/blog/en/data-visualization/>
4. INTOTHEMINDS. *What is a data artist? | with Nicholas Rougeux* [online]. 2020 [cit. 16.09.2021]. Dostupné z: <https://www.youtube.com/watch?v=k4D9qgVb17Q>
5. JOHN HUNTER a MICHAEL DROETTBOOM. The Architecture of Open Source Applications (Volume 2): matplotlib. [cit. 16.09.2021]. Dostupné z: <http://aosabook.org/en/matplotlib.html>
6. JOHN HUNTER a DARREN DALE. Overview — Matplotlib 3.4.3 documentation. 2021 [cit. 08.09.2021]. Dostupné z: <https://matplotlib.org/stable/index.html>
7. MANOVICH, Lev. What is visualisation? *Visual Studies*. Routledge, 2011, roč. 26, č. 1, s. 36–49. ISSN 1472-586X. DOI: [10.1080/1472586X.2011.548488](https://doi.org/10.1080/1472586X.2011.548488)
8. WASKOM, Michael. seaborn: statistical data visualization. *Journal of Open Source Software*. 2021, roč. 6, č. 60, s. 3021. ISSN 2475-9066. DOI: [10.21105/joss.03021](https://doi.org/10.21105/joss.03021)
9. VANDERPLAS, Jake. *Python Data Science Handbook: Essential Tools for Working with Data*. 1st edition. vyd. Sebastopol, CA: O’Reilly Media, 2016. ISBN 978-1-4919-1205-8.
10. Sponsored Projects | pandas, NumPy, Matplotlib, Jupyter, + more. In: *NumFOCUS* [online] [cit. 19.09.2021]. Dostupné z: <https://numfocus.org/sponsored-projects>
11. BOKEH CONTRIBUTORS. Bokeh documentation. 2021 [cit. 08.09.2021]. Dostupné z: <https://docs.bokeh.org/en/latest/index.html>
12. Plotly Open Source Graphing Libraries. [cit. 19.09.2021]. Dostupné z: <https://plotly.com/api/>
13. Welcome to HoloViews! — HoloViews 1.14.5 documentation. [cit. 17.10.2021]. Dostupné z: <https://holoviews.org/getting_started/index.html>
14. FLORIAN MOUNIER. Pygal — pygal 2.0.0 documentation. 2016 [cit. 08.09.2021]. Dostupné z: <http://www.pygal.org/en/stable/>
15. COREY, Michael. Turn your data into sound using our new MIDITime library. In: *Reveal* [online] [cit. 17.10.2021]. Dostupné z: <http://revealnews.org/blog/turn-your-data-into-sound-using-our-new-miditime-library/>
16. CUTTONE, Andrea. Geoplotlib - documentation. 5. 9. 2021 [cit. 08.09.2021]. Dostupné z: <https://github.com/andrea-cuttone/geoplotlib>
17. MUELLER, Andreas. *word\_cloud* [online]. 2021 [cit. 24.10.2021]. Dostupné z: <https://github.com/amueller/word_cloud>
18. *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* [online] [cit. 30.10.2021]. Dostupné z: <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed>
19. ASEEM KASHYAP. 8 Rules for optimal use of color in data visualization. *Medium* [online]. 28. 12. 2020 [cit. 09.09.2021]. Dostupné z: <https://towardsdatascience.com/8-rules-for-optimal-use-of-color-in-data-visualization-b283ae1fc1e2>
20. MORELAND, Kenneth. *Diverging Color Maps for Scientific Visualization*. Berlin, Heidelberg: Springer, 2009. Lecture Notes in Computer Science. ISBN 978-3-642-10520-3. DOI: [10.1007/978-3-642-10520-3\_9](https://doi.org/10.1007/978-3-642-10520-3_9)
21. KOVESI, Peter. Good Colour Maps: How to Design Them. *arXiv:1509.03700 [cs]* [online]. 2015 [cit. 30.10.2021]. Dostupné z: <http://arxiv.org/abs/1509.03700>
22. BARTRAM, Lyn, Abhisekh PATRA a Maureen STONE. Affective Color in Visualization. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* [online]. New York, NY, USA: Association for Computing Machinery, 2017, s. 1364–1374 [cit. 09.09.2021]. ISBN 978-1-4503-4655-9. Dostupné z: <https://doi.org/10.1145/3025453.3026041>
23. YI, Mike. How to Choose the Right Data Visualization [online]. 27.2.2020 [cit. 31.09.2021]. Dostupné z: <https://cdn2.hubspot.net/hubfs/392937/How-To-Choose-The-Right-Data-Visualization%20(1).pdf>
24. SEVERINO RIBECCA. The Data Visualisation Catalogue. [cit. 05.04.2021]. Dostupné z: <https://datavizcatalogue.com/>
25. CONOR HEALY a YAN HOLTZ. From data to Viz | Find the graphic you need. [cit. 10.04.2021]. Dostupné z: <https://www.data-to-viz.com/>
26. MORELAND, Kenneth. Why We Use Bad Color Maps and What You Can Do About It. *Electronic Imaging*. 2016, roč. 2016, č. 16, s. 1–6. ISSN 2470-1173. DOI: [10.2352/ISSN.2470-1173.2016.16.HVEI-133](https://doi.org/10.2352/ISSN.2470-1173.2016.16.HVEI-133)
27. index | TIOBE - The Software Quality Company. [cit. 14.11.2021]. Dostupné z: <https://www.tiobe.com/tiobe-index/>
28. PYPL PopularitY of Programming Language index. [cit. 14.11.2021]. Dostupné z: <https://pypl.github.io/PYPL.html>
29. LUTZ, Mark. *Learning Python: Powerful Object-Oriented Programming*. O’Reilly Media, Inc., 2013. ISBN 978-1-4493-5569-2.
30. Applications for Python. In: *Python.org* [online] [cit. 14.11.2021]. Dostupné z: <https://www.python.org/about/apps/>
31. THE NUMPY COMMUNITY. NumPy v1.21 Manual. [cit. 08.09.2021]. Dostupné z: <https://numpy.org/doc/stable/>
32. THE SCIPY COMMUNITY. SciPy documentation — SciPy v1.8.0 Manual. [cit. 20.11.2021]. Dostupné z: <https://scipy.github.io/devdocs/index.html>
33. THE PANDAS DEVELOPMENT TEAM. Getting started — pandas 1.3.4 documentation. [cit. 20.11.2021]. Dostupné z: <https://pandas.pydata.org/docs/getting_started/index.html>
34. A
35. A
36. A
37. A
38. a
39. a

# Přílohy

*Oskenované zadání práce*