DOI: https://doi.org/10.36910/6775-2524-0560-2026-64-24
УДК 004.4:004.92
Повстяна Юлія Славомирівна, к.т.н., доцент
https://orcid.org/0000-0001-5426-4157
Сухонос Павло Петрович, аспірант
https://orcid.org/0009-0004-4557-702X
Луцький національний технічний університет, м. Луцьк, Україна

ЕКСПЕРИМЕНТАЛЬНЕ ОЦІНЮВАННЯ ПРОГРАМНИХ ЗАСОБІВ АВТОМАТИЗОВАНОЇ ВІЗУАЛІЗАЦІЇ ДАНИХ

Повстяна Ю. С., Сухонос П. П. Експериментальне оцінювання програмних засобів автоматизованої візуалізації даних. У статті представлено результати експериментального оцінювання сучасних програмних засобів автоматизованої візуалізації даних. Метою дослідження є визначення доцільності застосування бібліотек Matplotlib, Plotly, Bokeh та Altair у різних сценаріях оброблення і графічного подання структурованих даних. Для порівняння сформовано дев'ять тестових сценаріїв, що охоплювали лінійні, точкові та стовпчикові діаграми, а також набори різного обсягу і структури. Експеримент передбачав десятикратне повторення кожного вимірювання. Оцінювання здійснювалося за часом формування візуалізації, піковим обсягом додаткових Python-алокацій, розміром вихідного артефакту, рівнем інтерактивності, підтримуваними форматами та складністю програмної реалізації. Встановлено, що Plotly забезпечує найвищу швидкість створення інтерактивних графіків для великих числових наборів, Matplotlib формує найкомпактніші статичні файли, Bokeh демонструє збалансоване поєднання продуктивності та інтерактивності, а Altair є ефективною для декларативного створення типових діаграм невеликого й середнього обсягу. Доведено, що універсально найкращого засобу не існує, а його вибір має визначатися структурою даних, вимогами до результату та умовами використання. Окремо встановлено, що зі збільшенням кількості записів зростають час серіалізації та розмір інтерактивних HTML-артефактів, тому для великих таблиць доцільно попередньо застосовувати агрегацію, фільтрацію або вибіркове подання даних користувачеві у браузері. Практичне значення результатів полягає у формуванні рекомендацій щодо вибору бібліотеки для наукової графіки, інтерактивної аналітики та вебзастосунків. Запропонована методика забезпечує відтворюваність експерименту та може бути використана для подальшого оцінювання інших засобів візуалізації.
Ключові слова: автоматизована візуалізація даних, бібліотеки Python, програмні засоби візуалізації, продуктивність, інтерактивність, оброблення даних, графічне подання даних.

Povstiana Y., Sukhonos P. Experimental evaluation of software tools for automated data visualization. The article presents the results of an experimental evaluation of modern software tools for automated data visualization. The study aims to determine the suitability of the Matplotlib, Plotly, Bokeh, and Altair libraries for different scenarios involving the processing and graphical representation of structured data. Nine test scenarios were developed, covering line, scatter, and bar charts, as well as datasets of varying sizes and structures. Each measurement was repeated ten times. The evaluation criteria included visualization generation time, peak additional Python memory allocations, output artifact size, level of interactivity, supported formats, and implementation complexity. The results indicate that Plotly provides the highest generation speed for interactive visualizations of large numerical datasets, while Matplotlib produces the most compact static files. Bokeh offers a balanced combination of performance and interactivity, whereas Altair is effective for the declarative creation of standard visualizations based on small and medium-sized datasets. It was established that no single tool is universally optimal and that the choice should depend on the data structure, output requirements, and intended application. The findings also show that increasing the number of records results in longer serialization times and larger interactive HTML artifacts. Therefore, aggregation, filtering, or selective data presentation should be applied before visualizing large tables in a web browser. The practical significance of the results lies in providing recommendations for selecting appropriate libraries for scientific graphics, interactive analytics, and web applications. The proposed methodology ensures experiment reproducibility and can be applied to evaluate other visualization tools.
Keywords: automated data visualization, Python libraries, visualization software, performance, interactivity, data processing, graphical data representation.

Постановка наукової проблеми. Стрімке зростання обсягів цифрових даних і розширення сфер їх використання зумовлюють потребу в програмних засобах, здатних автоматизувати побудову наочних, інформативних і придатних до подальшого аналізу візуалізацій. Графічне подання даних застосовується в наукових дослідженнях, бізнес-аналітиці, моніторингових системах, освіті, медицині та інформаційних технологіях. При цьому ефективність візуалізації залежить не лише від правильності вибору типу графіка, а й від швидкості оброблення даних, обсягу використаної оперативної пам'яті, підтримки інтерактивної взаємодії, можливостей експорту та складності програмної реалізації.

Сучасні бібліотеки візуалізації пропонують широкий набір засобів для побудови статичних та інтерактивних графіків. Водночас вони суттєво відрізняються за внутрішньою архітектурою, способом опису візуалізації, підтримуваними форматами, рівнем автоматизації та вимогами до обчислювальних ресурсів. У сучасних дослідженнях наголошується, що програмні засоби візуальної аналітики відрізняються не лише функціональними можливостями, а й рівнем

автоматизації окремих етапів опрацювання та графічного подання даних [2]. Комплексний огляд наявних платформ і бібліотек також засвідчує відсутність універсального засобу, однаково ефективного для всіх типів даних і сценаріїв використання [9]. Один програмний засіб може забезпечувати високу швидкість створення статичних графіків, проте мати обмежені можливості інтерактивної взаємодії. Інший може пропонувати розвинену інтерактивність, але потребувати більше оперативної пам'яті або мати складнішу програмну реалізацію. Тому популярність бібліотеки чи кількість доступних типів діаграм не є достатніми критеріями для обґрунтованого вибору засобу автоматизованої візуалізації.

Проблема полягає у відсутності універсальних практичних рекомендацій щодо вибору програмного засобу залежно від обсягу та структури даних, типу візуалізації й вимог до інтерактивності. Наявні порівняння часто охоплюють лише окремі характеристики бібліотек, а результати тестування на невеликих наборах не відображають їхньої продуктивності під час опрацювання значних обсягів даних. Недостатньо дослідженим залишається співвідношення між швидкодією, функціональністю та складністю програмної реалізації.

Це зумовлює необхідність проведення відтворюваного експерименту, у якому різні засоби тестуються в однаковому програмно-апаратному середовищі, на тих самих наборах даних і типах візуалізацій.

Метою дослідження є експериментальне оцінювання Matplotlib, Plotly, Bokeh та Altair і визначення доцільності їх застосування в різних сценаріях. Об'єктом дослідження є процес автоматизованої візуалізації структурованих даних, а предметом – показники продуктивності, ресурсомісткості, інтерактивності, функціональності та складності реалізації досліджуваних бібліотек.

Наукова новизна полягає у розробці та експериментальній апробації методики багатокритеріального оцінювання програмних засобів автоматизованої візуалізації даних (Matplotlib, Plotly, Bokeh, Altair), що, на відміну від наявних підходів, враховує не лише швидкість формування графіків, а й пікове споживання пам'яті, розмір підсумкових файлів, складність реалізації та збереження функціональної інтерактивності у різних сценаріях навантаження.

Аналіз досліджень. Сучасні дослідження автоматизованої візуалізації охоплюють оцінювання програмних засобів, продуктивність вебвізуалізацій і застосування методів штучного інтелекту. Shakeel, Iram, Al-Aqrabi, Alsboui та Hill систематизували можливості сучасних платформ і визначили основні проблеми їх використання [9]. Khan, Rydow, Etemaditajbakhsh, Adamek та Armour дослідили продуктивність вебвізуалізації за умов опрацювання великих обсягів потокових даних [4].

Основний напрям сучасних праць пов'язаний з інтелектуальною автоматизацією. Wang, Chen, Wang та Qu розглянули застосування машинного навчання на різних етапах візуалізаційного процесу – від підготовки даних до побудови й оцінювання графічного представлення [11]. Wu, Wang, Shu, Moritz, Cui, Zhang, Zhang та Qu систематизували підходи штучного інтелекту до автоматичного створення, удосконалення та інтерпретації візуалізацій [12].

Водночас зазначені дослідження переважно зосереджені на методах інтелектуальної автоматизації, тоді як порівняння швидкодії, використання пам'яті, розміру артефактів та складності реалізації універсальних Python-бібліотек висвітлено недостатньо. Це обґрунтовує необхідність експериментального зіставлення Matplotlib, Plotly, Bokeh та Altair у стандартизованих сценаріях.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. У межах цього дослідження під автоматизованою візуалізацією даних розуміється процес програмного формування графічного представлення структурованих даних на основі попередньо визначеного сценарію візуалізації без ручного редагування отриманого графічного результату. Автоматизація передбачає виконання програмним засобом операцій побудови та налаштування візуалізації відповідно до заданих параметрів, тоді як вибір типу візуалізації та структури вхідних даних визначається експериментальним сценарієм.

Для експериментального порівняння обрано чотири бібліотеки мови Python: Matplotlib 3.10.8, Plotly 6.3.0, Bokeh 3.8.0 та Altair 5.5.0. Вибір саме цих засобів зумовлений тим, що вони представляють різні підходи до візуалізації:

- Matplotlib орієнтована передусім на статичну наукову графіку;
- Plotly забезпечує інтерактивні вебвізуалізації та підтримує HTML-публікацію;

– Bokeh призначена для інтерактивних графіків і вебзастосунків;
– Altair використовує декларативний підхід на основі специфікації Vega-Lite.

Усі бібліотеки інтегруються з NumPy та pandas, підтримують основні типи двовимірних діаграм і можуть застосовуватися в автоматизованих аналітичних процесах. Водночас вони відрізняються способом формування графічних об'єктів, механізмом серіалізації даних, рівнем інтерактивності та вимогами до ресурсів.

Для перевірки впливу обсягу і структури даних сформовано дев'ять експериментальних сценаріїв. Дані генерувалися засобами NumPy з фіксованим початковим станом генератора псевдовипадкових чисел (seed =`20260818`).

З метою забезпечення комплексності експерименту сформовано дев'ять сценаріїв, які відрізнялися структурою даних, кількістю елементів і типом графічного представлення. Характеристику сформованих експериментальних сценаріїв наведено в табл. 1.

Таблиця 1. Характеристика експериментальних сценаріїв

Група сценаріїв	**Структура даних**	**Кількість елементів**	**Тип візуалізації**
S1–S3	Послідовність значень із трендом і випадковим шумом	1 000; 10 000; 100 000	Лінійна діаграма
S4–S6	Дві пов'язані кількісні змінні	1 000; 10 000; 100 000	Точкова діаграма
S7–S9	Категоріальні значення та відповідні числові показники	20; 100; 500 категорій	Стовпчикова діаграма

Джерело: авторська розробка

Лінійні набори імітували часовий ряд із періодичною складовою та випадковим шумом. Точкові набори містили дві корельовані числові змінні. Категоріальні набори складалися з унікальних назв категорій та згенерованих цілих значень.

Методика вимірювання.

Експеримент проведено в середовищі Linux 6.18.35 на обчислювальній системі з дев'ятьма віртуальними ядрами Intel Xeon Platinum 8573C і 15,93 ГБ оперативної пам'яті. Використано Python 3.12.13, NumPy 2.3.5 і pandas 2.2.3. Порядок виконання 36 комбінацій «бібліотека – сценарій» було випадково перемішано.

Для кожної комбінації спочатку виконувався один прогрівальний запуск, який не включався до результатів. Після нього проводилося десять вимірюваних повторень. Загальна кількість зафіксованих вимірювань становила:

$$N = 4 \cdot 9 \cdot 10 = 360$$

Основним показником продуктивності визначено час *T*, необхідний для створення графічного об'єкта та його серіалізації в готовий нативний артефакт:

$$T = T_{finish} - T_{start}$$

Імпорт бібліотек і генерування тестових даних до показника не включалися. Для Matplotlib результат зберігався у форматі PNG. Для Plotly, Bokeh та Altair створювався повний HTML-документ із CDN-посиланням на відповідне JavaScript-середовище. Такий підхід відображає типові сценарії застосування бібліотек: підготовку статичної ілюстрації для Matplotlib та інтерактивної вебвізуалізації для інших засобів.

Додатково фіксувалися:
– піковий додатковий обсяг Python-алокацій M за допомогою модуля `tracemalloc`;
– розмір сформованого артефакту S;
– міжквартильний розмах часу виконання;
– рівень інтерактивності;
– підтримувані формати експорту;
– кількість рядків програмної реалізації.

Для зменшення впливу випадкових коливань основною узагальнювальною характеристикою обрано медіану:

$$\tilde{T} = \text{median}\,(\mathrm{T}_1, \mathrm{T}_2, \ldots, T_{10}).$$

Результати вимірювання швидкодії досліджуваних бібліотек для дев'яти експериментальних сценаріїв наведено в табл. 2. Менше значення медіанного часу відповідає вищій швидкості формування готової візуалізації.

Таблиця 2. Медіанний час формування візуалізації, мс

Тип графіка	Обсяг	Matplotlib	Plotly	Bokeh	Altair
Лінійний	1 000	170,5	45,4	93,5	**39,3**
Лінійний	10 000	204,6	**28,0**	142,5	334,2
Лінійний	100 000	244,1	**35,9**	366,8	3265,8
Точковий	1 000	199,1	**24,8**	112,2	45,3
Точковий	10 000	225,9	**26,4**	135,2	328,5
Точковий	100 000	227,1	**47,6**	351,0	3281,8
Стовпчиковий	20	310,2	22,0	159,3	**10,3**
Стовпчиковий	100	1090,8	20,7	110,2	**13,3**
Стовпчиковий	500	8261,2	23,3	101,7	**22,9**

Джерело: авторська розробка

Plotly продемонструвала найменший час у п'яти з дев'яти сценаріїв, а Altair - у чотирьох. Геометричне середнє медіанних значень за всіма сценаріями становило 29,1 мс для Plotly, 123,3 мс для Altair, 153,2 мс для Bokeh та 396,7 мс для Matplotlib.

Найбільш виразні відмінності між бібліотеками спостерігалися під час опрацювання наборів максимального обсягу. Порівняння медіанного часу формування відповідних візуалізацій подано на рис. 1.

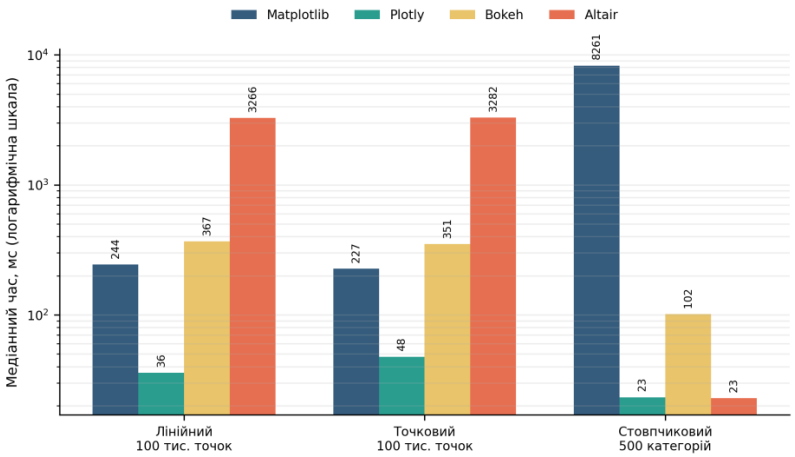

Рис. 1. Медіанний час формування візуалізацій у сценаріях максимального обсягу
Джерело: авторська розробка

Як видно з рис. 1, найменший час побудови лінійної та точкової діаграм зі 100 тис. точок продемонструвала бібліотека Plotly. Лінійний графік було сформовано за 35,9 мс, що у 6,8 раза швидше порівняно з Matplotlib, у 10,2 раза – з Bokeh та у 91 раз – з Altair. Для точкової діаграми час становив 47,6 мс, тоді як Matplotlib виконала цей сценарій за 227,1 мс, Bokeh – за 351,0 мс, Altair – за 3281,8 мс. Під час побудови стовпчикової діаграми з 500 категоріями близькі найкращі результати показали Altair і Plotly – 22,9 та 23,3 мс відповідно.

Найбільше зростання часу Altair пояснюється перетворенням усіх рядків таблиці на декларативну специфікацію Vega-Lite та включенням даних до HTML-документа. Якщо кількість точок збільшилася з 1 тис. до 100 тис., час формування лінійної діаграми Altair зріс у 83 рази, а точкової – у 72 рази.

Для визначення впливу обсягу даних на продуктивність було проаналізовано зміну часу формування лінійної діаграми за послідовного збільшення кількості точок від 1 тис. до 100 тис. Отриману залежність наведено на рис. 2.

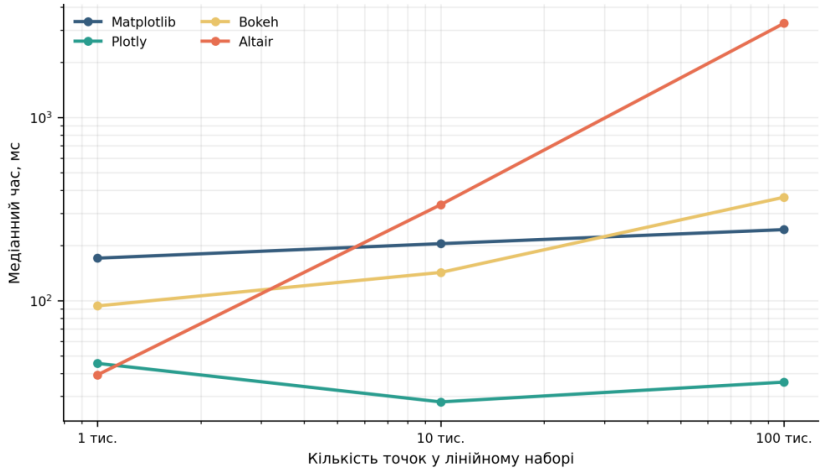

Рис. 2. Залежність часу формування лінійної візуалізації від кількості точок

Джерело: авторська розробка

Matplotlib показала відносно стабільний час для лінійних і точкових графіків. Збільшення кількості точок зі 1 тис. до 100 тис. спричинило зростання медіанного часу лінійної візуалізації лише з 170,5 до 244,1 мс. Водночас для стовпчикової діаграми з 500 унікальними категоріями час досяг 8261,2 мс. Причиною є створення великої кількості окремих графічних елементів і текстових підписів категоріальної осі. Отже, Matplotlib є придатною для числових масивів значного обсягу, але потребує попереднього агрегування категорій.

Найбільшу стабільність у сценарії лінійної діаграми зі 100 тис. точок продемонструвала Plotly: міжквартильний розмах становив 1,64 мс. Для Matplotlib цей показник дорівнював 9,56 мс, для Bokeh – 19,55 мс, для Altair – 688,79 мс. Висока варіативність Altair додатково свідчить про чутливість декларативної серіалізації до обсягу вбудованих даних.

Використання пам'яті та розмір результатів.

Окрім швидкодії, проаналізовано використання пам'яті та розмір сформованих файлів. Показник пам'яті характеризує піковий обсяг додаткових Python-алокацій під час створення артефакту, а не загальне споживання оперативної пам'яті процесом. Результати для сценаріїв із максимальною кількістю елементів наведено в табл. 3.

Таблиця 3. Пам'ять і розмір артефактів для максимальних сценаріїв

Сценарій	**Бібліотека**	**Пікова пам'ять, МБ**	**Розмір артефакту, КБ**
Лінійний, 100 тис.	Matplotlib	8,31	40,2
	Plotly	31,02	1865,2
	Bokeh	8,33	1180,4
	Altair	31,24	3859,9
Точковий, 100 тис.	Matplotlib	4,11	62,1
	Plotly	33,09	2662,9
	Bokeh	14,00	2007,4
	Altair	33,12	5203,0
Стовпчиковий, 500	Matplotlib	17,32	11,6
	Plotly	22,84	12,2
	Bokeh	0,53	11,1
	Altair	0,23	19,1

Джерело: авторська розробка

Matplotlib сформувала найкомпактніші результати для великих числових наборів: 40,2 КБ для лінійної та 62,1 КБ для точкової діаграми. Проте PNG не містить вихідних даних та інтерактивних функцій. HTML-артефакти включали значення, необхідні для відтворення графіка у браузері, тому їхній розмір зростав майже пропорційно кількості записів.

Найбільший файл – 5203 КБ – сформувала Altair для точкового набору зі 100 тис. записів. Аналогічний результат Plotly мав розмір 2663 КБ, Bokeh – 2007 КБ. Отже, використання інтерактивної візуалізації великих наборів без попередньої агрегації може істотно збільшувати обсяг даних, які передаються користувачеві.

Plotly характеризувалася значним постійним рівнем алокацій: 22,8-33,1 МБ залежно від сценарію. Altair використовувала найменше пам'яті для невеликих категоріальних наборів, однак для набору зі 100 тис. записів її потреба зросла до 31-33 МБ. Bokeh посіла проміжне положення та продемонструвала порівняно невеликі витрати пам'яті для стовпчикових діаграм.

Функціональність і складність реалізації.

Для остаточного вибору програмного засобу кількісні результати доповнено оцінюванням інтерактивності, підтримуваних форматів, складності реалізації та переважних сфер застосування. Узагальнені функціональні характеристики бібліотек наведено в табл. 4.

Таблиця 4. Функціональні характеристики досліджуваних засобів

Засіб	**Рядки коду**[1]	**Інтерактивність**	**Основні формати**	**Найдоцільніші сценарії**
Matplotlib	14	Обмежена; переважно статичний результат	PNG, JPEG, TIFF, SVG, PDF, EPS	Наукові статті, звіти, друкована графіка
Plotly	10	Висока: масштабування, наведення, панорамування, вибір даних	HTML, JSON; PNG, JPEG, WebP, SVG, PDF із додатковим модулем	Інтерактивна аналітика, вебпанелі, презентація даних
Bokeh	9	Висока; підтримка інструментів, подій і серверних callback-функцій	HTML, JSON; PNG і SVG із браузерним середовищем	Вебзастосунки, потокові й динамічні дані
Altair	9	Декларативні параметри, фільтри та виділення	HTML, JSON; PNG, SVG, PDF із додатковим конвертером	Швидке створення компактних графіків для малих і середніх даних

Джерело: авторська розробка

[1] Кількість рядків стандартизованої функції, яка формувала три типи графіків; імпорти та генерування дани х не враховано.

Перелік форматів узгоджується з офіційною документацією Matplotlib [6], Plotly [7], Bokeh [1] та Altair [10].

Найкоротшу програмну реалізацію в межах розробленого сценарію мали Bokeh та Altair. Декларативний синтаксис Altair дає змогу описувати відповідність між полями таблиці та візуальними каналами, не керуючи кожним графічним елементом окремо. Matplotlib потребувала більшої кількості команд, однак забезпечила найдетальніший контроль над оформленням статичної ілюстрації.

Обґрунтування та зіставлення результатів.

Отримані результати підтверджують, що поняття масштабованості візуалізації не може оцінюватися лише за однією характеристикою. У моделі Richer та співавторів масштабованість розглядається через співвідношення розміру задачі, доступних ресурсів і необхідних обчислювальних або візуальних зусиль. Проведене дослідження конкретизує цей підхід одночасним урахуванням часу, пам'яті, розміру результату та функціональності [8].

Висновок про обмежену масштабованість універсальних бібліотек для дуже великих точкових наборів узгоджується з дослідженням Jupyter Scatter, автори якого зазначають, що Matplotlib, Bokeh та Altair потребують спеціальної оптимізації під час роботи з мільйонами точок [5]. У проведеному експерименті відповідна тенденція проявилася вже на рівні 100 тис. записів через збільшення часу серіалізації та розміру HTML.

Результати оцінювання складності реалізації також відповідають висновкам порівняльного дослідження Python-бібліотек: високорівневі декларативні засоби, зокрема Altair, спрощують створення типової візуалізації, тоді як Matplotlib, Plotly та Bokeh забезпечують ширші можливості детального налаштування ціною більшої складності [3].

Таким чином, жодна з досліджених бібліотек не є оптимальною за всіма критеріями. Plotly доцільно використовувати, якщо пріоритетом є швидке формування інтерактивної візуалізації. Matplotlib є кращою для статичних наукових матеріалів і компактних графічних файлів. Bokeh доцільна для вебзастосунків, у яких потрібні події, callback-функції або оновлення даних. Altair є ефективною для автоматизованого створення типових діаграм невеликого та середнього обсягу, але перед візуалізацією великих таблиць потребує агрегації, фільтрації або зовнішнього зберігання даних.

Під час інтерпретації отриманих результатів слід ураховувати обмеження проведеного експерименту. Для Matplotlib створювався статичний PNG-файл, тоді як для Plotly, Bokeh та Altair

формувалися інтерактивні HTML-документи. Тому розмір артефакту характеризує типовий спосіб публікації результатів кожної бібліотеки, але не є прямим порівнянням однакових файлових форматів. Крім того, у межах експерименту не вимірювалися час браузерного відображення, швидкість мережевого завантаження та затримка інтерактивної взаємодії.

Слід ураховувати, що експеримент оцінював серверний етап створення артефакту. Швидкість браузерного відображення, час мережевого завантаження JavaScript, затримка реакції на дії користувача та продуктивність графічного процесора не вимірювалися. Крім того, результати стосуються базових двовимірних діаграм і не можуть без додаткової перевірки поширюватися на карти, тривимірну графіку та потокові дані.

Висновки та перспективи подальшого дослідження. Отримані результати дають підстави стверджувати, що ефективність програмних засобів автоматизованої візуалізації не можна визначати лише за швидкістю побудови графіків. Обґрунтований вибір бібліотеки має враховувати обсяг і структуру даних, тип візуалізації, використання пам'яті, розмір вихідного файлу, інтерактивність і складність програмної реалізації.

Експеримент підтвердив наявність компромісу між продуктивністю, компактністю результату та функціональністю. Plotly доцільно використовувати для швидкого формування інтерактивних візуалізацій, Matplotlib – для статичної наукової графіки, Bokeh – для вебзастосунків із динамічною взаємодією, а Altair – для декларативного створення типових діаграм невеликого та середнього обсягу. Для великих наборів даних важливого значення набуває попередня агрегація або фільтрація, оскільки безпосереднє включення всіх записів до інтерактивного HTML-документа збільшує час серіалізації та розмір результату.

Таким чином, практична доцільність програмного засобу визначається конкретним сценарієм його застосування, а сформована система експериментальних критеріїв дає змогу здійснювати такий вибір на основі кількісних і якісних показників.

Перспективи подальших досліджень полягають у перевірці отриманих закономірностей на реальних наборах даних обсягом понад один мільйон записів, а також у дослідженні потокових, просторових і тривимірних візуалізацій. Доцільно додатково оцінити швидкість браузерного відображення, затримку інтерактивної взаємодії та вплив апаратного прискорення.

Список бібліографічного опису

1. Bokeh. PNG and SVG Export. Bokeh Documentation. URL: https://docs.bokeh.org/en/latest/docs/user_guide/output/export.html (дата звернення: 17.08.2026).
2. Domova V., Vrotsou K. A Model for Types and Levels of Automation in Visual Analytics: A Survey, a Taxonomy, and Examples. *IEEE Transactions on Visualization and Computer Graphics*. 2023. Vol. 29, No. 8. P. 3550–3568. DOI: https://doi.org/10.1109/TVCG.2022.3163765.
3. Herda G., McNabb R. Python for Smarter Cities: Comparison of Python Libraries for Static and Interactive Visualisations of Large Vector Data. *arXiv*. 2022. URL: https://arxiv.org/abs/2202.13105 (дата звернення: 18.08.2026). DOI: https://doi.org/10.48550/arXiv.2202.13105.
4. Khan S., Rydow E., Etemaditajbakhsh S., Adamek K., Armour W. Web Performance Evaluation of High Volume Streaming Data Visualization. *IEEE Access*. 2023. Vol. 11. P. 15623–15636. DOI: https://doi.org/10.1109/ACCESS.2023.3245043.
5. Lekschas F., Manz T. Jupyter Scatter: Interactive Exploration of Large-Scale Datasets. *Journal of Open Source Software*. 2024. Vol. 9, No. 101. Art. 7059. DOI: https://doi.org/10.21105/joss.07059.
6. Matplotlib. Introduction to Figures. Matplotlib documentation. URL: https://matplotlib.org/stable/users/explain/figure/figure_intro.html (дата звернення: 16.08.2026).
7. Plotly. Interactive HTML Export in Python. Plotly Python Open Source Graphing Library. URL: https://plotly.com/python/interactive-html-export/ (дата звернення: 17.08.2026).
8. Richer G., Pister A., Abdelaal M., Fekete J.-D., Sedlmair M., Weiskopf D. Scalability in Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 2024. Vol. 30, No. 7. P. 3314–3330. DOI: https://doi.org/10.1109/TVCG.2022.3231230.
9. Shakeel H. M., Iram S., Al-Aqrabi H., Alsboui T., Hill R. A Comprehensive State-of-the-Art Survey on Data Visualization Tools: Research Developments, Challenges and Future Domain-Specific Visualization Framework. *IEEE Access*. 2022. Vol. 10. P. 96581–96601. DOI: https://doi.org/10.1109/ACCESS.2022.3205115.
10. Vega-Altair. Saving Altair Charts. Vega-Altair Documentation. URL: https://altair-viz.github.io/user_guide/saving_charts.html (дата звернення: 18.08.2026).
11. Wang Q., Chen Z.-T., Wang Y., Qu H. A Survey on ML4VIS: Applying Machine Learning Advances to Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 2022. Vol. 28, No. 12. P. 5134–5153. DOI: https://doi.org/10.1109/TVCG.2021.3106142.
12. Wu A., Wang Y., Shu X., Moritz D., Cui W., Zhang H., Zhang D., Qu H. AI4VIS: Survey on Artificial Intelligence Approaches for Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 2022. Vol. 28, No. 12. P. 5049–5070. DOI: https://doi.org/10.1109/TVCG.2021.3099002.

References

1. Bokeh. (n.d.). *PNG and SVG export***.** Bokeh documentation. Retrieved August 17, 2026, from https://docs.bokeh.org/en/latest/docs/user_guide/output/export.html
2. Domova, V., & Vrotsou, K. (2023). A model for types and levels of automation in visual analytics: A survey, a taxonomy, and examples. *IEEE Transactions on Visualization and Computer Graphics, 29*(8), 3550–3568. https://doi.org/10.1109/TVCG.2022.3163765
3. Herda, G., & McNabb, R. (2022). *Python for smarter cities: Comparison of Python libraries for static and interactive visualisations of large vector data* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2202.13105
4. Khan, S., Rydow, E., Etemaditajbakhsh, S., Adamek, K., & Armour, W. (2023). Web performance evaluation of high volume streaming data visualization. *IEEE Access, 11*, 15623–15636. https://doi.org/10.1109/ACCESS.2023.3245043
5. Lekschas, F., & Manz, T. (2024). Jupyter Scatter: Interactive exploration of large-scale datasets. *Journal of Open Source Software, 9*(101), Article 7059. https://doi.org/10.21105/joss.07059
6. Matplotlib. (n.d.). *Introduction to figures*. Matplotlib documentation. Retrieved August 16, 2026, from https://matplotlib.org/stable/users/explain/figure/figure_intro.html
7. Plotly. (n.d.). *Interactive HTML export in Python*. Plotly Python Open Source Graphing Library. Retrieved August 17, 2026, from https://plotly.com/python/interactive-html-export/
8. Richer, G., Pister, A., Abdelaal, M., Fekete, J.-D., Sedlmair, M., & Weiskopf, D. (2024). Scalability in visualization. *IEEE Transactions on Visualization and Computer Graphics, 30*(7), 3314–3330. https://doi.org/10.1109/TVCG.2022.3231230
9. Shakeel, H. M., Iram, S., Al-Aqrabi, H., Alsboui, T., & Hill, R. (2022). A comprehensive state-of-the-art survey on data visualization tools: Research developments, challenges and future domain-specific visualization framework. *IEEE Access, 10*, 96581–96601. https://doi.org/10.1109/ACCESS.2022.3205115
10. Vega-Altair. (n.d.). *Saving Altair charts*. Vega-Altair documentation. Retrieved August 18, 2026, from https://altair-viz.github.io/user_guide/saving_charts.html
11. Wang, Q., Chen, Z.-T., Wang, Y., & Qu, H. (2022). A survey on ML4VIS: Applying machine learning advances to data visualization. *IEEE Transactions on Visualization and Computer Graphics, 28*(12), 5134–5153. https://doi.org/10.1109/TVCG.2021.3106142
12. Wu, A., Wang, Y., Shu, X., Moritz, D., Cui, W., Zhang, H., Zhang, D., & Qu, H. (2022). AI4VIS: Survey on artificial intelligence approaches for data visualization. *IEEE Transactions on Visualization and Computer Graphics, 28*(12), 5049–5070. https://doi.org/10.1109/TVCG.2021.3099002

Історія статті:
Отримано: 17.06.2026 Доопрацьовано: 24.08.2026 Прийнято до друку: 23.09.2026 Опубліковано: 29.09.2026