**ДИПЛОМНАЯ РАБОТА**

ПО ТЕМЕ:

**СРАВНЕНИЕ РАЗЛИЧНЫХ БИБЛИОТЕК ДЛЯ ВИЗУАЛИЗАЦИИ ДАННЫХ:  
MATPLOTLIB, SEABORN И PLOTLY**

Автор: Минькович Александр Владимирович

СИМФЕРОПОЛЬ – 2024

**ОГЛАВЛЕНИЕ**

|  |  |  |
| --- | --- | --- |
| 1. | **Введение**………………………………………………………………… | 3 |
| 1.1. | Обоснование выбора темы…………………………………………….. | 3 |
| 1.2. | Определение цели и задач исследования……………………………... | 4 |
| 1.3. | Основные понятия и определения…………………………………….. | 4 |
| 2. | **Инструменты визуализация данных**………………………………... | 7 |
| 2.1. | Обзор библиотеки Matplotlib…………………………………………... | 7 |
| 2.2. | Обзор библиотеки Seaborn…………………………………………….. | 8 |
| 2.3. | Обзор библиотеки Plotly……………………………………………….. | 9 |
| 3. | **Визуализация одномерных данных**………………………………… | 11 |
| 4. | **Визуализация двухмерных данных**……………………………........ | 15 |
| 5. | **Визуализация трехмерных данных**…………………………………. | 19 |
| 6. | **Сравнительный анализ использования библиотек**…………….… | 23 |
| 6.1. | Критерии анализа……………………………………………………….. | 23 |
| 6.2. | Анализ…………………………………………………………………… | 23 |
| 6.3. | Оценка времени работы кода………………………………………….. | 24 |
| 6.4. | Рекомендации по использованию библиотек………………………… | 25 |
| 7. | **Заключение**…………………………………...………………………… | 27 |
| 7.1. | Обзор выполненной работы……………………………………………. | 27 |
| 7.2. | Дальнейшие планы……………………………………………………… | 27 |

# ВВЕДЕНИЕ

Визуализация дает вам ответы на вопросы, о наличии которых вы и не подозревали[[1]](#footnote-1)

*Бен Шнейдерман*

* 1. **Обоснование выбора темы**

Визуализация данных - это графическое представление информации и данных. Используя визуальные элементы, такие как диаграммы, графические изображения и карты и другие инструменты визуализации данных мы получаем доступный способ увидеть и понять тенденции, выбросы и закономерности в данных. Этот мощный инструмент позволяет пользователям быстро выявлять закономерности, тенденции и корреляции в данных, что делает его незаменимым помощником в различных областях, включая бизнес, науку и образование.

Пользователям Python предлагается большой выбор различных инструментов визуализации. Однако лишь некоторые из этих инструментов используют метод Python и включают интерактивные диаграммы. Конкретные примеры помогут увидеть различия между Matplotlib, Seaborn и Plotly. Выбор правильного типа диаграммы имеет решающее значение для эффективной передачи данных. Разные диаграммы служат разным целям и могут освещать различные аспекты ваших данных.

Эффективный графический визуализатор можно охарактеризовать следующим образом:

* показывает или визуализирует данные очень четко и понятным способом.
* побуждает пользователей сравнивать различные фрагменты данных.
* тесно объединяет статистические и вербальные описания набора данных.
* привлекает интерес пользователя, фокусирует его разум и удерживает внимание на информации, поскольку человеческий мозг, как правило, больше фокусируется на визуальных данных, чем на письменных.
* помогает определить область, требующую большего внимания и улучшения.
* делает действия пользователя рациональнее и эффективнее.
* требуется меньше времени для понимания и анализа информации, чем текстовые данные.

Актуальность и безальтернативность использования инструментов визуализации, их постоянное развитие определило мой выбор темы дипломной работы «Сравнение различных библиотек для визуализации данных: Matplotlib, Seaborn и Plotly»

* 1. **Определение цели и задач исследования**

**Цель исследования**: осуществить сравнительный анализ библиотек для визуализации данных (Matplotlib, Seaborn и Plotly), чтобы оценить их эффективность и рациональность использования.

**Задачи исследования**:

1. Обзор библиотек для визуализации данных: Провести анализ существующих инструментов визуализации данных.
2. Разработка методики сравнения: Разработать метод объективного сравнения эффективности графиков, определить критерии сравнения и набрать минимальный набор данных для первичной проверки.
3. Проведение тестирования: оценить эффективность создания графиков библиотек для визуализации данных и провести их тестирование.
4. Написание дипломной работы: Составить дипломную работу, включающую в себя введение, методологию и результаты исследования, анализ результатов, выводы и рекомендации, обзор литературы.

Цели и задачи исследования направлены на получение практических результатов, которые позволят выработать рекомендации по применению библиотек для визуализации данных.

* 1. **Основные понятия и определения**

[**Двумерные данные**](https://translated.turbopages.org/proxy_u/en-ru.ru.fe1dd789-6724b08a-2758bd27-74722d776562/https/www.geeksforgeeks.org/bivariate-analysis/) **(Two-dimensional data)** — это данные по каждой из двух переменных, где каждое значение одной из переменных сопряжено со значением другой переменной.

**Визуализация данных** — Графическое представление любой информации или данных.

**Гистограмма** — способ представления данных в виде столбчатой диаграммы, состоящей из прямоугольников одинаковой ширины, высота которых указывает на частоту появления определённого параметра в заданном диапазоне

**Дашборд** — это информационная панель, которая получает данные из других систем и отображает их в понятном виде. На дашбордах используют текст, графики, диаграммы и другие средства визуализации. статическими график

панели мониторинга

**Инструменты визуализации данных** - это программные платформы, которые предоставляют информацию в визуальном формате, таком как График, диаграмма и т.д., чтобы сделать ее понятной и полезной.

**Категориальная переменная** — это переменная, принимающая значения из некоторого ограниченного набора категорий. Обычно она связана с неисчисляемыми атрибутами, такими как названия (товаров, услуг и др.), имена людей, исходы событий (да/нет) и т.д.

**Круговая диаграмма** — это схема в виде круга, разделённого центральными углами на части, или сектора

**Набор данных** (data set или dataset) — это коллекция данных, которая касается определённой темы или отрасли.

**Одномерные данные (One-dimensional data)** — это тип данных, в которых каждое наблюдение или точка данных соответствует одной переменной.

**Столбчатая диаграмма** — диаграмма, представленная прямоугольными зонами (столбцами), высоты или длины которых пропорциональны величинам, которые они отображают

**Трехмерные данные (three-dimensional data)** — это цифровые данные о пространственных объектах, включающие сведения об их местоположении, форме и свойствах, представленные в трёхмерном виде и трёхмерной координатно-временной системе.

**Трехмерная графика**, или 3D-графика, — одно из направлений в области CGI — компьютерной генерации изображений. Это техника, которая позволяет создавать графические объекты с эффектом глубины, что делает их похожими на объекты реального мира.

**API (Application programming interface)** — это программный интерфейс приложений, набор инструкций, который позволяет разным приложениям общаться между собой.

**Axes** - это объект, который содержит все прямоугольники, линии и т.д.

**CGI (Computer-Generated Imagery)** — изображение, созданное компьютером. Этот вид графики используется в фильмах при создании статических и динамических объектов: зданий, автомобилей, персонажей, ландшафтов и т.д. Также CGI применяется в дизайне и рекламе.

**DataFrame** это двумерная структура данных, похожая на таблицу в SQL или на лист в Excel.

**figure** — это визуальное представление, которое включает в себя график.

**GUI (graphical user interface)** это тип интерфейса, который позволяет пользователям взаимодействовать с компьютером посредством визуальных элементов, таких как значки, кнопки и окна

**JavaScript Object Notation (JSON)** — это текстовый формат обмена данными, основанный на JavaScript.

**MATLAB** — это высокоуровневый интерпретируемый язык программирования вместе с пакетом прикладных программ и интегрированной средой для разработки, выполнения инженерных и математических расчётов, работы с матричными базами данных и визуализации.

**Matplotlib** — это библиотека для визуализации данных в Python.

**Pandas** — это программная библиотека, написанная на языке Python, которая используется для анализа структурированных данных, размещенных в таблицах (панелях).

**Plotly** это библиотека визуализации данных с открытым исходным кодом, предоставляющая широкий спектр возможностей интерактивного построения графиков на языке Python

**Seaborn** — это библиотека для создания статистических графиков на Python

1. **ИНСТРУМЕНТЫ ВИЗУАЛИЗАЦИЯ ДАННЫХ**
   1. **Обзор библиотеки Matplotlib.**

****

**Matplotlib**  —  это пакет для построения двухмерных и трехмерных графиков и визуализации данных, написанный на языке Python. Это самый популярный и часто используемый пакет инструментов для построения графиков в сообществе Python. Его интерактивную среду можно применять на различных платформах. Matplotlib  —  это библиотека Python, используемая в скриптах Python, оболочках Python и IPython, ноутбуках Jupyter, серверах веб-приложений и во многих других случаях.

Графики библиотеки могут быть встроены в приложения, использующие Tkinter, GTK+, wxPython, Qt и другие наборы инструментов графического пользовательского интерфейса GUI (graphical user interface). С помощью Matplotlib можно создавать различные типы визуализации, включая[[2]](#footnote-2):

* **попарные данные** (pairwise data): plot(x, y), scatter(x, y), bar(x, height), stem(x, y), fill\_between(x, y1, y2), stackplot(x, y), stairs(values).
* **статистические распределения** (statistical distributions): hist(x), boxplot(X), errorbar(x, y, yerr, xerr), violinplot(D), eventplot(D), hist2d(x, y), hexbin(x, y, C), pie(x), ecdf(x).
* **данные, привязанные к сетке** (gridded data): imshow(Z), pcolormesh(X, Y, Z), contour(X, Y, Z), contourf(X, Y, Z), barbs(X, Y, U, V), quiver(X, Y, U, V), streamplot(X, Y, U, V).
* **данные с неправильной привязкой к сетке** (irregularly gridded data): tricontour(x, y, z), tricontourf(x, y, z), tripcolor(x, y, z), triplot(x, y).
* **трехмерные и объемные данные** (3D and volumetric data): bar3d(x, y, z, dx, dy, dz), plot(xs, ys, zs), quiver(X, Y, Z, U, V, W), scatter(xs, ys, zs), stem(x, y, z), plot\_surface(X, Y, Z), plot\_trisurf(x, y, z), voxels([x, y, z], filled), plot\_wireframe(X, Y, Z).

При построении возможно указать оси координат, сетку, добавить аннотации, использовать логарифмическую шкалу или полярные координаты. Созданные изображения могут быть легко сохранены, в частности, в популярные форматы (JPEG, PNG и др.).

В Matplotlib все объекты организованы в единую иерархию[[3]](#footnote-3):

* На вершине иерархии находится [конечный автомат](https://ru.ruwiki.ru/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82) (модель, число возможных внутренних состояний которой конечно), предоставляемый модулем matplotlib.pyplot. На данном уровне все функции, используемые для рисования диаграмм (линии, рисунки, текст и т.д.) применяются к текущему изображению.
* На уровне ниже располагается объектно-ориентированный интерфейс, используемый для создания изображений, где объектами являются само изображение, оси координат, графические примитивы и т.д.
  1. **Обзор библиотеки Seaborn.**

**Seaborn** - это библиотека визуализации данных на Python, которая упрощает процесс создания сложных визуализаций. Она построена поверх библиотеки Matplotlib и является расширением Matplotlib. Она позволяет создавать более привлекательные и простые для восприятия графики, которые удобнее строить и демонстрировать.

В Seaborn используются следующие типы графиков[[4]](#footnote-4):

**для визуализации распределения метрических переменных**: distplot, jointplot, rugplot, kdeplot.

**для визуализации относительного распределения между парами переменных**: PairGrid, pairplot, FacetGrid.

**для визуализации категориальных данных**: factorplot, boxplot, violinplot, stripplot, swarmplot, barplot, countplot.

**матричные графики**: Heatmaps

В Seaborn многие настройки по умолчанию уже оптимизированы для создания более привлекательных и информативных графиков. Кроме того, Seaborn может строить сложные графики с множественными переменными, что особенно полезно для исследования связей в многомерных данных.

Пользователям предоставляется возможность настраивать стиль графика, цветовую палитру и другие эстетические аспекты для создания профессионально выглядящих диаграмм Seaborn предлагает пять предустановленных тем: white grid, dark grid, white, dark и ticks, каждая из которых подходит для разных приложений, а также для личных предпочтений[[5]](#footnote-5).

Библиотека позволяет напрямую передавать табличные данные pandas DataFrame в функции построения графиков, что значительно ускоряет процесс визуализации.

С помощью Seaborn можно:

* вычислять и визуализировать корреляции — соотношения между разными данными.
* анализировать и сравнивать данные между собой, строить графики по этому анализу.
* создавать графики, подходящие для демонстрации визуализации анализа тем, кто не слишком глубоко разбирается в данных.
  1. **Обзор библиотеки Plotly.**

Plotly  —  это бесплатная графическая библиотека, позволяющая создавать визуализации данных. Plotly  —  это веб-приложение для визуализации данных, основанное на JavaScript-библиотеке Plotly (plotly.py/plotly.js).

У Plotly есть несколько функций, которые отличают его от других графических библиотек[[6]](#footnote-6):

* по умолчанию он интерактивен.
* графики хранятся в JavaScript Object Notation (JSON) data format чтобы их можно было читать с помощью скриптов других языков программирования, таких как R, Julia, MATLAB и т. д.
* вектор экспорта для печати и публикации.
* графики можно экспортировать в различные растровые и векторные форматы изображений и встраивать в Интернет.

Plotly Express является рекомендуемым способом визуализации данных и, в отличии от создания графических объектов напрямую, занимает несколько строчек кода. Это высокоуровневый API[[7]](#footnote-7) для визуализации данных. С помощью него можно создавать графические объекты вызовом соответствующих функций.

Графические объекты Plotly (часто сокращенно Plotly go) - это низкоуровневый интерфейс, обеспечивающий более детальный контроль над созданием графика и настройкой. В отличие от Plotly Express, который абстрагирует множество деталей, Plotly Graph Objects позволяет пользователям создавать графики, вручную указывая каждый компонент.

Этот подход подходит для пользователей, которым нужен детальный контроль над своими визуализациями или которые работают с более сложными требованиями к построению графиков.

В Plotly Express есть 30 функций для создания фигур[[8]](#footnote-8). Для большинства Data Science проектов их достаточно. Причём API для этих функций очень простой, достаточно разобраться с их аргументами, которые для каждого типа фигуры могут отличаться.

Plotly Express содержит функции для построения:

* **Базовые графики**: scatter, line, area, bar, funnel, timeline.
* **Части целого**: pie, sunburst, treemap, funnel\_area.
* **1D распределения**: histogram, box, violin, strip.
* **2D распределения**: density\_heatmap, density\_contour.
* **С матричным входом**: imshow.
* **3D-графики**: scatter\_3d, line\_3d.
* **Многомерные**: scatter\_matrix, parallel\_coordinates, parallel\_categories.
* **Тайловые карты**: scatter\_mapbox, line\_mapbox, choropleth\_mapbox, density\_mapbox.
* **Контурные карты**: scatter\_geo, line\_geo, choropleth.
* **Графики в полярной системе координат**: scatter\_polar, line\_polar, bar\_polar.
* **Тернарные диаграммы**: scatter\_ternary, line\_ternary.

Plotly предоставляет полный набор опций для настройки графиков. Пользователи могут управлять различными аспектами — цветовые палитры, шрифты, оси, аннотации, легенды, заголовки и макет — для создания визуально привлекательных и информативных визуализаций.

Plotly позволяет создавать интерактивные графики и диаграммы, которые реагируют на действия пользователя, такие как наведение курсора, щелчок и масштабирование. Такая интерактивность расширяет возможности исследования и понимания данных.

Plotly предлагает онлайн-платформу Plotly Chart Studio, где пользователи могут создавать, редактировать и делиться графиками. Он обеспечивает централизованное место для хранения визуализаций и совместной работы над ними, что упрощает обмен информацией с другими.

1. **ВИЗУАЛИЗАЦИЯ ОДНОМЕРНЫХ ДАННЫХ**

Одномерные данные (One-dimensional data) - этот тип данных состоит из одной переменной. Таким образом, анализ одномерных данных является простейшей формой анализа, поскольку информация имеет дело только с одной величиной, которая изменяется. В нем не рассматриваются причины или взаимосвязи, и основная цель анализа - описать данные и найти существующие в них закономерности.

В качестве примера рассматриваем набор данных **train.csv**. Train.csv — это обучающая выборка в наборе данных Titanic на платформе Kaggle. Каждая строчка набора данных содержит следующие поля:

Pclass — класс пассажира (1 — высший, 2 — средний, 3 — низший);

Name — имя;

Sex — пол;

Age — возраст;

SibSp — количество братьев, сестёр, сводных братьев, сводных сестёр, супругов на борту «Титаника»;

Parch — количество родителей, детей (в том числе приёмных) на борту «Титаника»;

Ticket — номер билета;

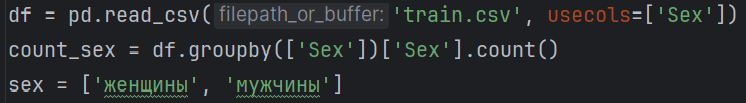
Fare — плата за проезд;

Cabin — каюта;

Embarked — порт посадки (C — Шербур; Q — Квинстаун; S — Саутгемптон).

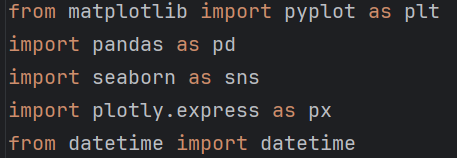
Хотя набор данных многомерен по своей природе, для одномерного анализа мы рассматриваем одно поле Sex.

Набор данных, первоначально представленный в формате .csv, загружается в DataFrame df с помощью функции pd.read\_csv( ) pandas . Затем он отображает DataFrame df. Делаем выборку по этому полю и подсчитываем количество значений male и female.



Для построения графиков будет использован модуль **dp\_one\_dimensional\_data.py**

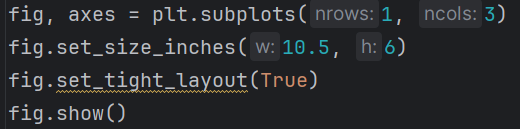
Для анализа используем круговую диаграмму, используемую для отображения только одного ряда данных. Площадь срезов круга представляет собой процентное соотношение частей данных. Сектора называются клиньями. Его можно создать с помощью метода **pie()**, который есть во всех библиотеках. Для построения графиков, необходимо сделать импорт:



Всё пространство (figure) включает в себя график (Axes), который, в свою очередь, состоит из таких элементов, как границы (Spines), сетки (Grid), описания (Legend), самого графика (Line) и других.

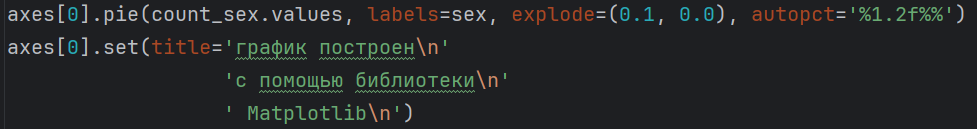
Графики отражают зависимость, при этом элементы графика — оси, подписи и др. — открыты к настройке.

Базовый элемент figure может содержать в себе как один график, так и несколько. Мною, например, будут созданы три графика в одном окне:



Функция subplots() возвращает объект Figure и объект Axes. Объект Figure — это базовый объект для всех графиков. Объект Axes — это объект, который содержит все прямоугольники, линии и т.д. С помощью Axes мы и будем рисовать круговую диаграмму. Вызов **axes[0].pie** — это и есть построение круговой диаграммы в **Matplotlib.**

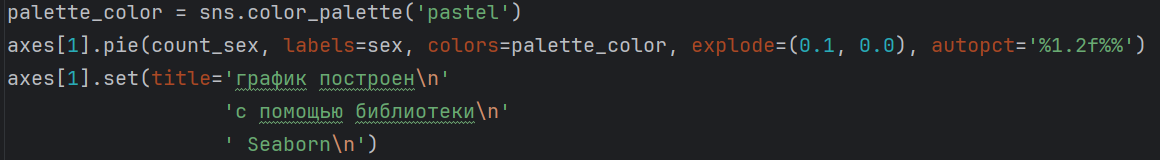
Синтаксис: *Axes.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=0, radius=1, counterclock=True, wedgeprops=None, textprops=None, center=(0, 0), frame=False, rotatelabels=False, \*, normalize=True, hatch=None, data=None)*



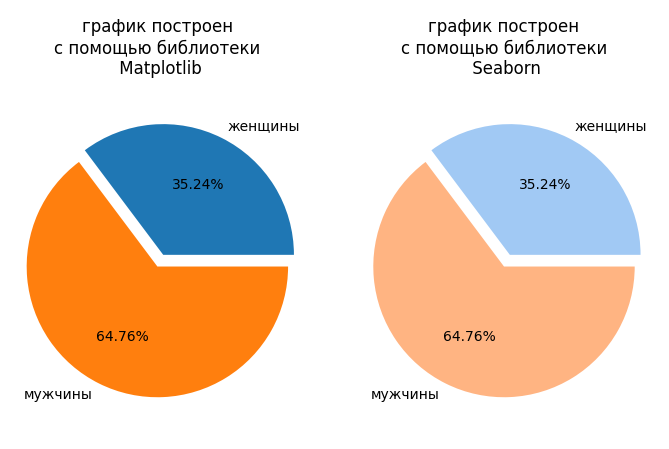
Параметр explode разделяет часть диаграммы, а colors определяет цвет каждого клина. Функция autopct настраивает отображение текста, а функции legend и title повышают удобочитаемость и эстетичность диаграммы.

Теперь обратимся к библиотеке **Seaborn**.

Чтобы создать круговую диаграмму с помощью Seaborn в Python, мы должны использовать атрибут **pie()** в Matplotlib и палитру цветов в Seaborn. Нам нужно передать входные данные и палитру цветов для создания круговой диаграммы[[9]](#footnote-9). Передаем палитру цветов в виде строки внутри атрибута color\_palette().



Результат получился таким:



Для создания круговой диаграммы с использованием библиотеки **Plotly** нужно передать в функцию **pie()** фрейм данных или массив значений и меток.

Синтаксис:*plotly.express.pie(data\_frame= Нет, names= Нет, values= Нет, color= Нет, color\_discrete\_sequence= Нет, color\_discrete\_map={}, hover\_name= Нет, hover\_data= Нет, custom\_data = Нет, labels ={}, title= Нет, template = Нет, width= Нет, height= Нет, opacity = Нет, hole = Нет)*

В px.pie данные, отображаемые по секторам pie, задаются в виде значений. Метки секторов задаются в виде имен. Аргументы width и height используются для задания ширины и высоты круговой диаграммы в пикселях. С помощью аргумента title, задаем заголовок круговой диаграммы. По умолчанию функция pie() присваивает каждой ячейке свой цвет и создает легенду.



В результате выполнения кода получается следующая диаграмма.

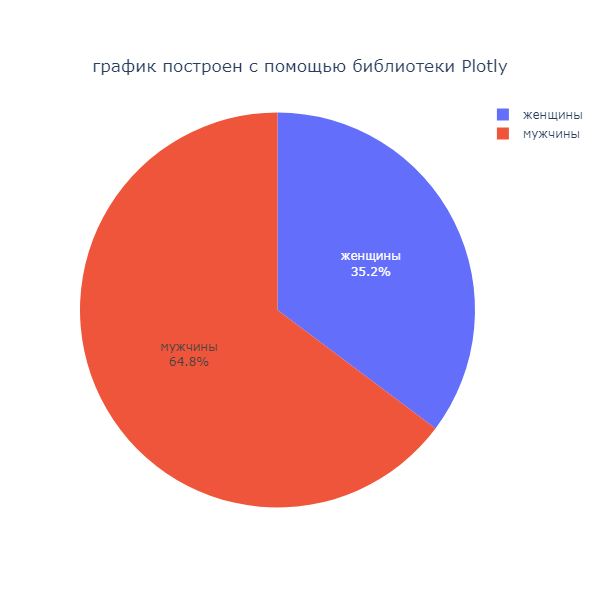
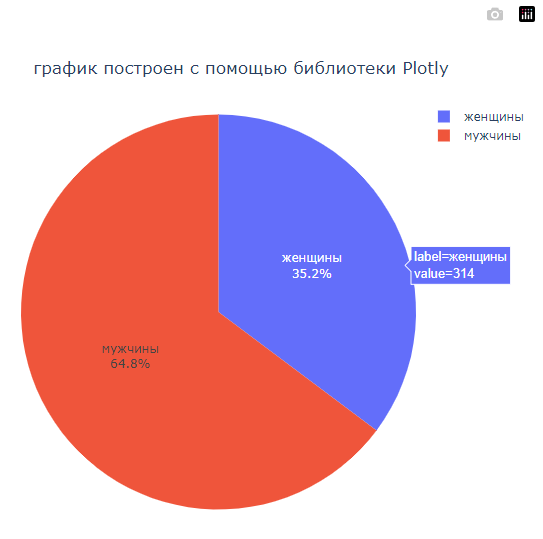


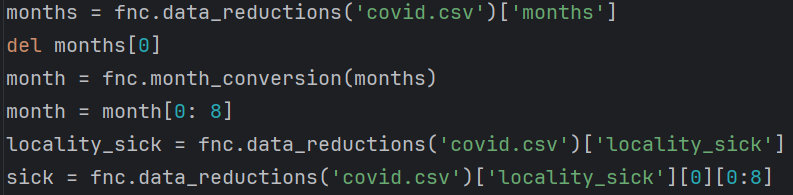
Диаграмма интерактивная, если навести на нее курсор, то можно ее приближать и удалять, выделять участки, по наведению курсора на точку получать подробную информацию, возвращать картинку в исходное положение, а при необходимости «скриншотить» и сохранять как файл.



1. **ВИЗУАЛИЗАЦИЯ ДВУХМЕРНЫХ ДАННЫХ**

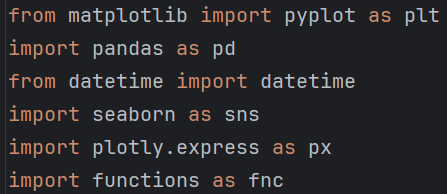
[Двумерные данные](https://translated.turbopages.org/proxy_u/en-ru.ru.fe1dd789-6724b08a-2758bd27-74722d776562/https/www.geeksforgeeks.org/bivariate-analysis/) (Two-dimensional data), этот такой тип данных, который включает в себя две разные переменные. Анализ данных этого типа имеет дело с причинами и взаимосвязями, и анализ проводится для выяснения взаимосвязи между двумя переменными.

Для анализа библиотек будет использован набор данных covid.csv. Этот набор представляет собой статистику заболеваемости коронавирусом по районам Владимирской области[[10]](#footnote-10) ежедневно с 6.05.2020 г. по 20.08.2021 г. Для анализа необходимо две переменные: первая **month** (список месяцев), вторая **sick** (общее количество заболевших за месяц). Эти данные выделяются из набора данных в модуле **functions.py** c помощью функции **data\_reductions()**. Берем первую строку для г. Владимира



Для построения графиков будет использован модуль **dp\_two\_dimensional\_data.py**

Для визуализации категориальных данных хорошо подходят столбчатые диаграммы (гистограммы). Осуществляем импорт:



В библиотеке **Matplotlib** для построения столбчатой диаграммы используют функцию: *bar()* – для построения вертикальной диаграммы.

Синтаксис[[11]](#footnote-11): *matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, align='center', data=None, \*\*kwargs)*

Основные параметры:

* + *x*: набор величин (*x* координаты столбцов)
  + *height*: скалярная величина или набор величин (Высоты столбцов)
  + *width*: скалярная величина, массив или *optional* (Ширина столбцов)
  + *bottom*: скалярная величина, массив или *optional (y* координата базы)
  + *align : {‘center’, ‘edge’}, optional,* значение по умолчанию*: ‘center’* (Выравнивание по координате *x)*.

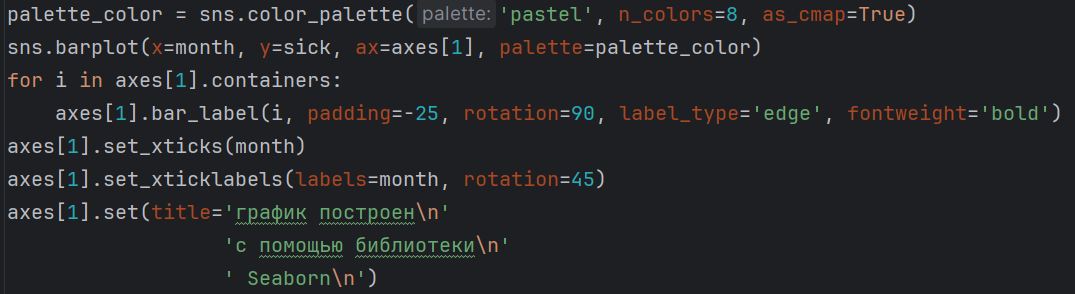
Дополнительные параметры:

* + *color*: скалярная величина, массив или *optional* (Цвет столбцов диаграммы)
  + *edgecolor*: скалярная величина, массив или *optional* (Цвет границы столбцов)
  + *linewidth*: скалярная величина, массив или *optional* (Ширина границы)
  + *tick\_label*: *str*, массив или *optional* (Метки для столбца)

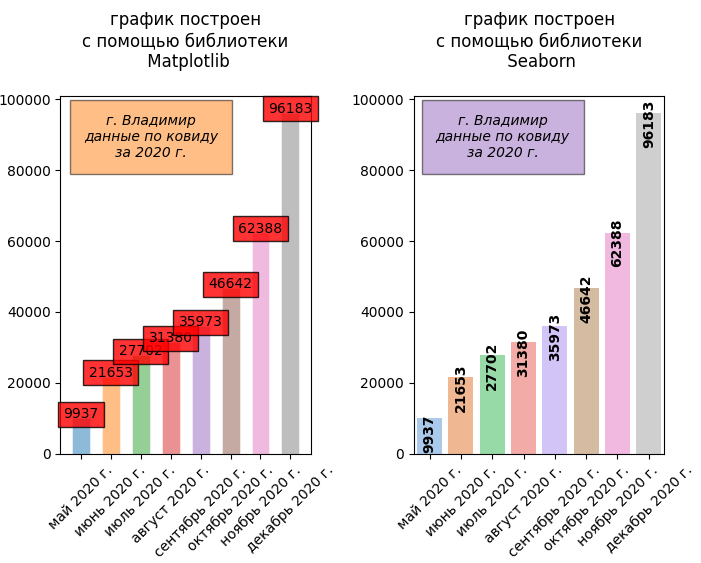


Чтобы нарисовать гистограмму в **Seaborn** нужно вызвать функцию **barplot()**, и передать ей категориальные и числовые переменные, которые нужно визуализировать

Синтаксис[[12]](#footnote-12): seaborn.barplot(*data=None*, *\**, *x=None*, *y=None*, *hue=None*, *order=None*, *hue\_order=None*, *estimator='mean'*, *errorbar=('ci'*, *95)*, *n\_boot=1000*, *seed=None*, *units=None*, *weights=None*, *orient=None*, *color=None*, *palette=None*, *saturation=0.75*, *fill=True*, *hue\_norm=None*, *width=0.8*, *dodge='auto'*, *gap=0*, *log\_scale=None*, *native\_scale=False*, *formatter=None*, *legend='auto'*, *capsize=0*, *err\_kws=None*, *ci=<deprecated>*, *errcolor=<deprecated>*, *errwidth=<deprecated>*, *ax=None*, *\*\*kwargs*)

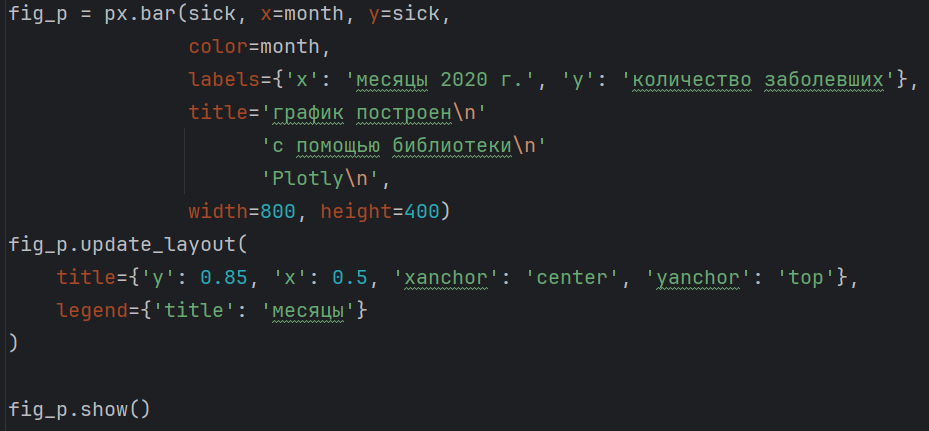


Результат построения графиков следующий:

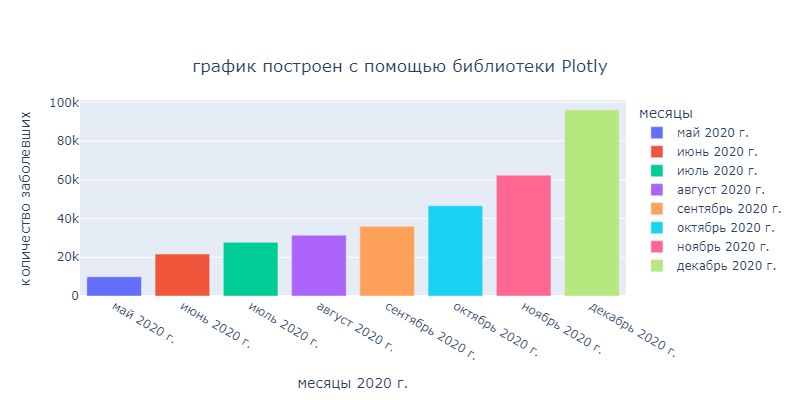


**Plotly** поддерживает подмодуль для создания гистограммы Plotly Express. Это простой в использовании высокоуровневый интерфейс Plotly, который работает с различными типами данных и создает простые в оформлении рисунки.

Синтаксис[[13]](#footnote-13):*plotly.express.bar(data\_frame=None, x=None, y=None, color=None, facet\_row=None, facet\_col=None, facet\_col\_wrap=0, hover\_name=None, hover\_data=None, custom\_data=None, text=None, error\_x=None, error\_x\_minus=None, error\_y=None, error\_y\_minus=None, animation\_frame=None, animation\_group=None, category\_orders={}, labels={}, color\_discrete\_sequence=None, color\_discrete\_map={}, color\_continuous\_scale=None, range\_color=None, color\_continuous\_midpoint=None, opacity=None, orientation=None, barmode=’relative’, log\_x=False, log\_y=False, range\_x=None, range\_y=None, title=None, template=None, width=None, height=None)*



В итоге получаем гистограмму:

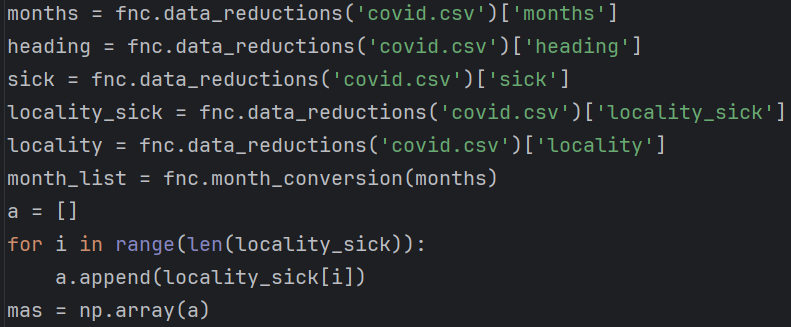


При наведении на один из столбцов будет отображено дополнительно значение месяца и количество заболевших в этом месяце. А также по цвету можно определить, какой месяц анализируется.

1. **ВИЗУАЛИЗАЦИЯ ТРЕХМЕРНЫХ ДАННЫХ**

Трехмерные данные (three-dimensional data). Трехмерный визуальный анализ позволяет анализировать данные в трехмерном пространстве, например, строить трехмерное изображение последовательностей исходных данных (наблюдений) для одной или нескольких выбранных переменных.

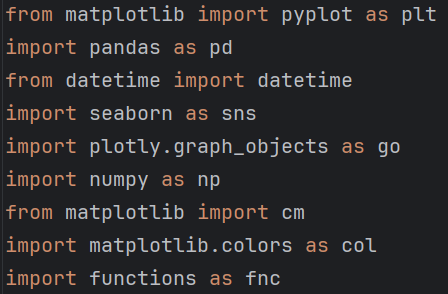
Для анализа библиотек будет использован тот же набор данных covid.csv, что был использован в предыдущей главе. Этот набор статистики заболеваемости коронавирусом по районам Владимирской области[[14]](#footnote-14) Для анализа возьмем три переменные: первая **month\_list** (список из 16 месяцев), вторая **locality\_sick** (общее количество заболевших за месяц по каждому населенному пункту) и третья **locality** (18 населенных пунктов). Эти данные так же выделяются из набора данных в модуле **functions.py** c помощью функции **data\_reductions()**. В процессе подготовки данных словарь **locality\_sick** преобразовываем в **mas** двухмерный массив (np.array).



Для построения графиков будет использован модуль **dp\_three\_dimensional\_data.py**

Для полноценной проверки не удалось использовать однотипные графики, к тому же Seaborn не предназначена для работы с трехмерной визуализацией или интерактивными графиками. В Matplotlib будем создавать 3D-график[[15]](#footnote-15), который имеет 2D-гистограммы, спроецированные на плоскости y=0, y= 1 и т.д. Для Seaborn повторим график Matplotlib, но используем методы библиотеки для создания цвета. Для проверки Plotly создадим 3D-график поверхности.

Выполним импорт:

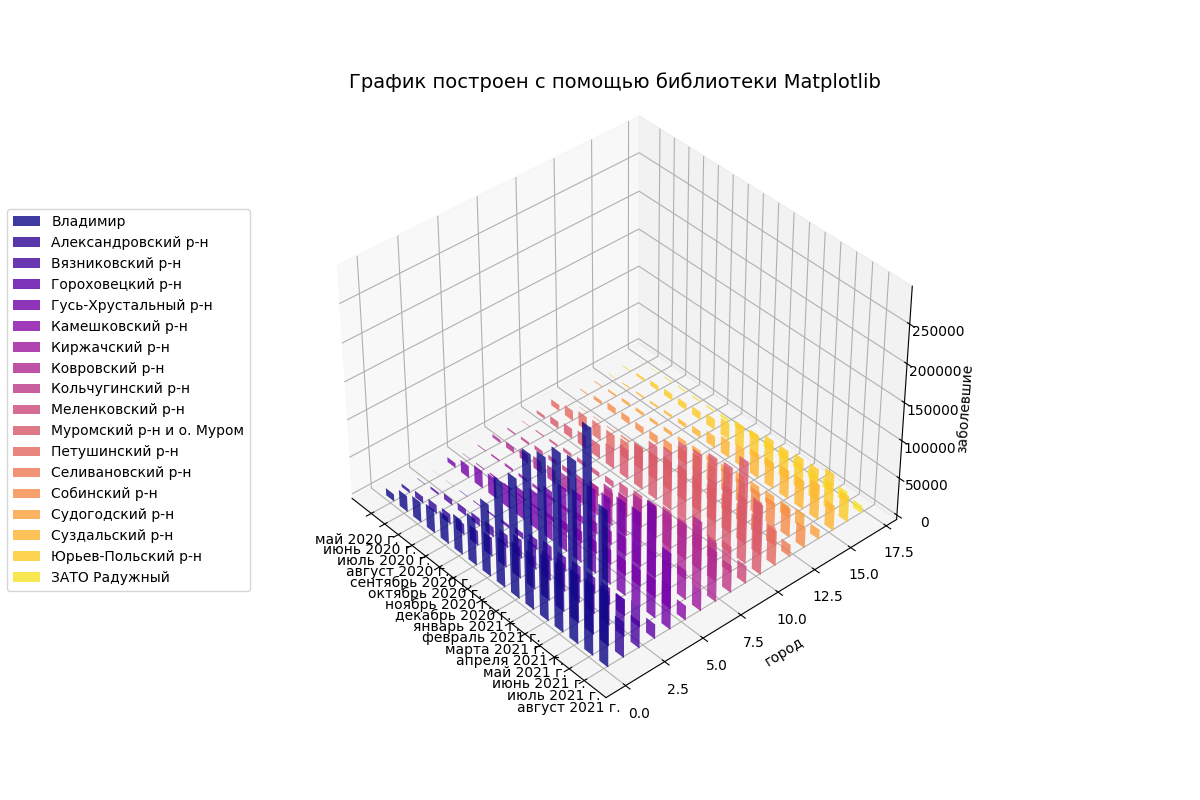


Для проверки библиотеки Matplotlib создадим фигуру и вспомогательный график для нашего 3D-графика. Мы будем использовать метод **add\_subplot()** для создания 3D-проекции. Далее настроим гистограммы: создадим массив цветов и используем метод **bar()** для построения гистограмм. Мы установим для параметра zdir значение 'y', чтобы проецировать гистограммы на плоскости оси y. Мы также установим параметр alpha равным 0.8, чтобы настроить прозрачность полос. Теперь мы настроим оси 3D-графика. Мы установим метки для осей x, y и z, используя методы set\_xlabel(), set\_ylabel() и set\_zlabel() соответственно.

Чтобы отобразить график используем метод show().



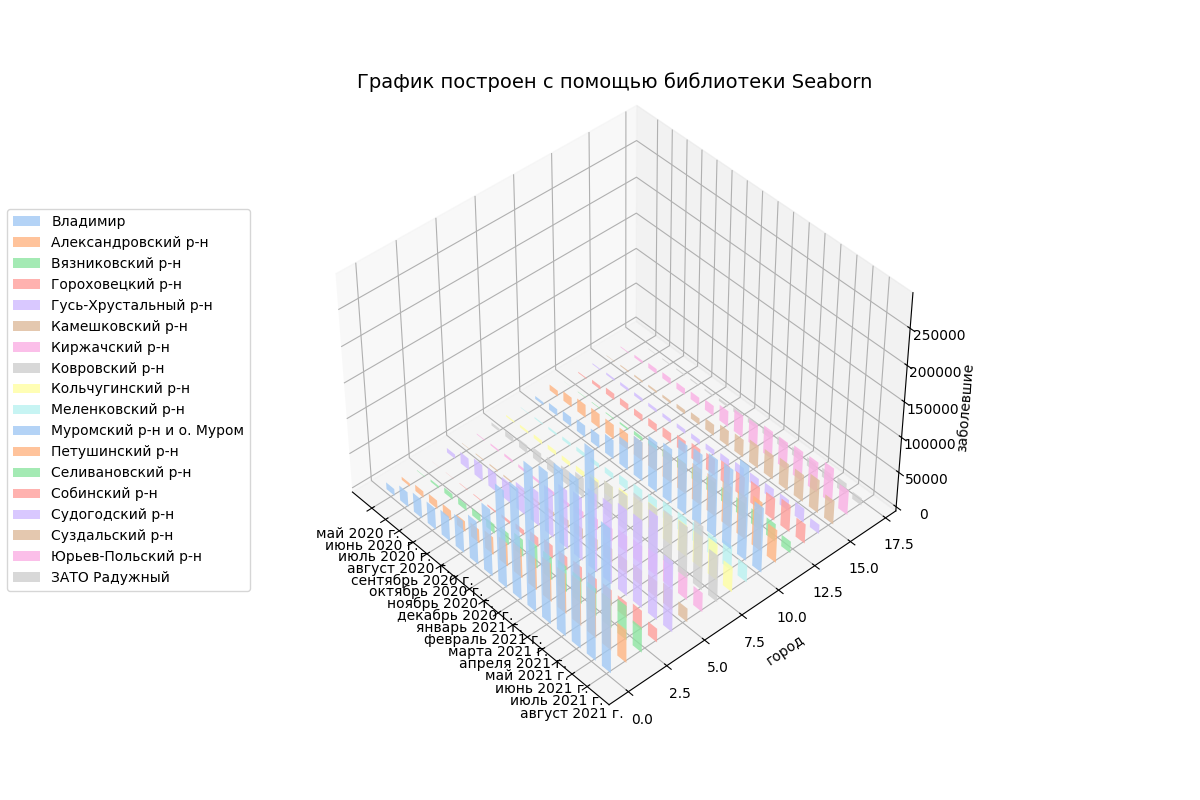
В итоге получаем график.



В Seaborn нет встроенной библиотеки, содержащей функции 3d-печати. Она является расширением библиотеки matplotlib и опирается на ее 3D-графику. Таким образом, мы можем стилизовать 3d-графику Matplotlib с помощью Seaborn. Используем функцию библиотеки **color\_palette()**, которую можно использовать для раскрашивания графика. Используя палитру, мы можем создать разные цвета.



График получился следующего вида:



Теперь перейдем к созданию 3D-графика поверхности с помощью **Plotly** с использованием go.Figure и go.Surface. Для этого нужно использовать метод create\_3d\_surface из графических объектов Plotly (plotly.graph\_objects)

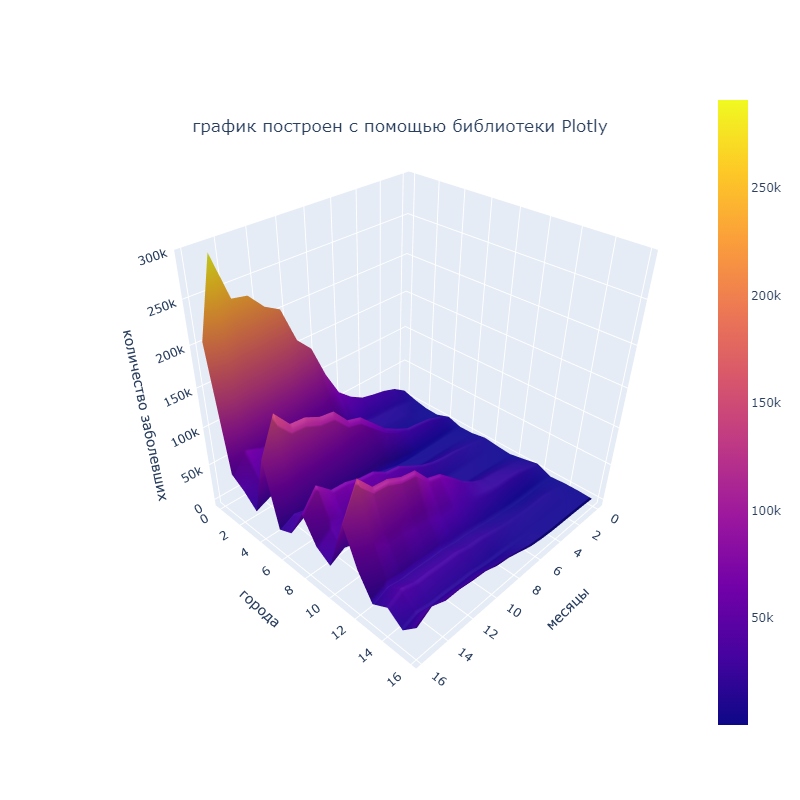
Необходимо подготовить данные. Для генерации координат для графика поверхности используем функцию linspace(). Получаем координаты x, y.

Создаем график. Для этого создаем объект Figure, используя go.Figure(), а затем добавляем к нему Surface. Аргументам x, y и z заданы значения **month\_list** (16 месяцев), **locality** (18 населенных пунктов) и **locality\_sick** (общее количество заболевших за месяц по каждому населенному пункту) соответственно.

Мы обновляем макет рисунка, используя fig.update\_layout(), включая title и scene.

Выводим график с помощью функции fig.show().





1. **СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИСПОЛЬЗОВАНИЯ БИБЛИОТЕК**
   1. **Критерии анализа**

Рассмотрим библиотеки Matplotlib, Seaborn и Plotly, уделяя особое внимание следующим показателям:

* + 1. Интерактивность.
    2. Синтаксис и гибкость.
    3. Тип данных и визуализации.
    4. Оценка времени работы кода.
  1. **Анализ**

**Библиотека Matplotlib**:

* позволяет быстро строить диаграммы и графики разных видов, настраивать их оформление.
* поддерживает API для интеграции графиков в разработанные приложения.
* умеет форматировать диаграммы и графики для более простого восприятия.

С помощью Matplotlib можно:

* строить 2D-фигуры.
* формировать на основе данных линейные, точечные, столбчатые, круговые и другие диаграммы.
* рисовать контурные графики.
* формировать поля векторов и спектрограммы.
* быстро встраивать визуализацию в сервисы, программы и приложения.

**Библиотека Seaborn**:

* имеет инструменты для исследования и анализа данных перед визуализацией.
* поддерживает данные разных форматов.
* имеет широкие возможности для настройки внешнего вида графиков и создания сложных визуализаций.

С помощью Seaborn можно:

* вычислять и визуализировать корреляции (соотношения между разными данными).
* анализировать и сравнивать данные между собой, строить графики по этому анализу.
* создавать графики, подходящие для демонстрации визуализации анализа тем, кто не слишком глубоко разбирается в данных.

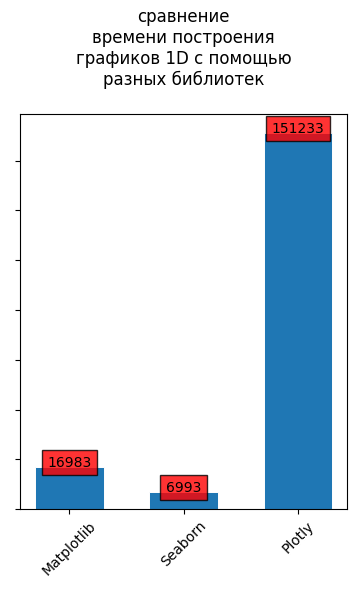
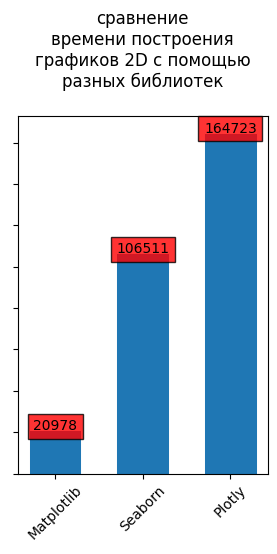
**Библиотека Plotly**:

* поддерживает трёхмерные визуализации и их продвинутые настройки.
* позволяет экспортировать результаты анализа в особом формате — JSON. Его удобно открывать в других приложениях.
* обладает одним из самых широких списков поддерживаемых диаграмм.
* умеет отправлять данные в облачные сервисы, чтобы работать там с ними дальше.
* на основе этой библиотеки существует ещё одна, Dash — она позволяет строить интерактивные дашборды для демонстрации данных.

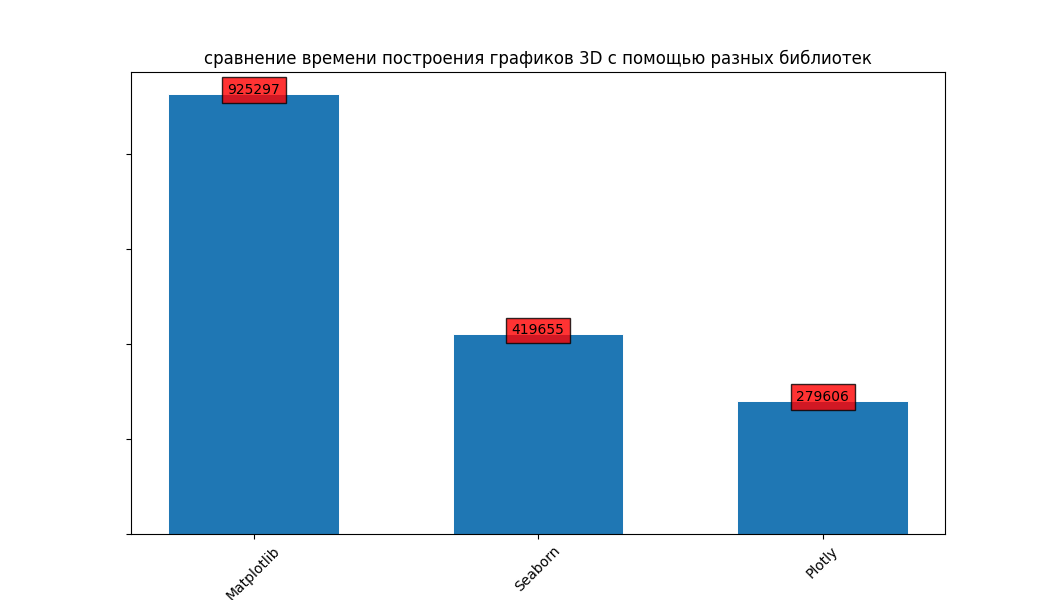
С помощью Plotly можно:

* строить любые обычные диаграммы и графики: круговые, Ганта, древовидные.
* формировать научные карты: тепловые, контурные, логарифмические, с полями векторов.
* строить финансовые графики.
  1. **Оценка времени работы кода**.

При построении графиков с использованием одномерных данных быстрее всего была построена круговая диаграмма с помощью библиотеки **Seaborn** **(6993 мкс)**, в три раза медленнее построил **Matplotlib (16983 мкс)** и замыкает ряд **Plotly** **(151233 мкс)**.

При построении графиков с использованием двухмерных данных быстрее всего был построен столбчатый график с помощью библиотеки **Matplotlib (20978 мкс)**, в четыре раза медленнее построил **Seaborn** **(106511 мкс)** и замыкает ряд **Plotly** **(164723 мкс)**.

При построении графиков с использованием трехмерных данных быстрее всего был построен 3D-график поверхности с помощью библиотеки **Plotly** **(279606 мкс)**, в полтора раза медленнее построил **Seaborn** **(419655 мкс)** и медленнее всего **Matplotlib (925297 мкс)**.



* 1. **Рекомендации по использованию библиотек**

В результате выполнения дипломной работы и анализа ее результатов получены выводы позволяющие сделать следующие рекомендации[[16]](#footnote-16).

Matplotlib лучше всего использовать, в следующих случаях:

* если надо выполнить быстрый анализ данных.
* если необходимо создать базовую визуализацию, чтобы можно было быстро увидеть тренд данных.
* если хочется создать более интегрированную и тщательную визуализацию. если надо создать свой собственный код и иметь представление о том, что на самом деле визуализируется в конце.
* если нужен подробный контроль над внешним видом графика, широкие возможности настройки или при работе со статическими графиками
* когда нужно создавать сложные графики с несколькими осями, вложенными графиками или пользовательскими аннотациями, Matplotlib предоставляет необходимый детальный контроль.
* если надо делать научные и инженерные графики: его способность обрабатывать научные данные, логарифмические шкалы и специализированные типы графиков делает его идеальным для технических областей.
* если необходимо создавать графики высокого разрешения для статей, презентаций или отчетов.

Seaborn лучше всего использовать, в следующих случаях:

* если надо быстро создать простые визуализации.
* функции Seaborn предназначены для быстрой визуализации взаимосвязей между переменными в наборе данных.
* если нужна статистическая графика: Seaborn предлагает множество типов графиков, адаптированных для статистического анализа, таких как графики распределения, категориальные графики и графики регрессии.
* если надо создать визуально приятные графики с минимальным количеством кода.
* если необходимо обрабатывать многомерные данные и создавать информативные визуализации, такие как парные графики и фасетные сетки.
* если надо работать с фреймами данных Pandas, упрощая подготовку данных и построение графиков.

Plotly лучше всего использовать, в следующих случаях:

* если необходимо создавать интерактивные визуализации и панели мониторинга.
* если необходимо обеспечить большую доступность данных.
* если необходимо создание интерактивных карт и географических визуализаций.
* если необходимо обрабатывать большие наборы данных, с высокой производительностью, позволяя исследовать и визуализировать их.
* если надо привлечь аудиторию с помощью интерактивных визуальных элементов.
* если надо экспортировать визуализации в различные форматы, включая статические изображения и HTML-файлы.

1. **ЗАКЛЮЧЕНИЕ**
   1. **Обзор выполненной работы**

Сравнение различных библиотек для визуализации данных (Matplotlib, Seaborn и Plotly) в результате проделанной работы было проведено.

Была изучена литература и материалы сайтов, осуществлен обзор библиотек для визуализации данных. Дана краткая характеристика каждой из рассматриваемых библиотек. Проанализированы существующие инструменты визуализации данных, изучены типы графиков.

Разработана методика сравнения. Принято, что для объективного сравнения эффективности графиков, требуется написание программы, задачей которой будет создание графиков с использованием тестируемых библиотек для разного типа данных.

Программа написана. Для ее реализации используется 5 модулей и 2 набора данных. При исполнении программы графики создаются.

Проведено тестирование библиотек для визуализации данных. Оценена эффективность создания графиков при выполнении разработанной программы.

Проведен анализ полученных результатов, позволивший выработать практические рекомендации по применению библиотек для визуализации данных.

* 1. **Дальнейшие планы**

Вопрос визуализации данных крайне интересен и востребован при создании практических приложений, в связи с чем планирую продолжить изучение данной темы.

Планируется добавить программе интерфейс, развить направление трехмерной визуализации, использовать различные наборы данных для тестирования.

1. «Shneiderman's Eight Golden Rules of Interface Design». Retrieved December 4, 2015. [↑](#footnote-ref-1)
2. Matplotlib 3.9.2 documentation. https://matplotlib.org [↑](#footnote-ref-2)
3. Devpractice Team. Библиотека Matplotlib. - devpractice.ru. 2019. - 100 с.: ил. [↑](#footnote-ref-3)
4. Seaborn 0/13/2 documentation, https://seaborn.pydata.org [↑](#footnote-ref-4)
5. https://pythonim.ru/libraries/biblioteka-seaborn-v-python [↑](#footnote-ref-5)
6. https://pythonim.ru/libraries/biblioteka-plotly-python [↑](#footnote-ref-6)
7. https://python-school.ru [↑](#footnote-ref-7)
8. Python API reference for Plotly – 5.24.1 documentatition, https://plotly.com/python-api-referen [↑](#footnote-ref-8)
9. https://pieriantraining.com/seaborn-pie-chart-a-tutorial-for-data-visualization/ [↑](#footnote-ref-9)
10. https://hubofdata.ru/dataset/vladimir\_covid-19 [↑](#footnote-ref-10)
11. https://python-lab.ru/metod-matplotlib-pyplot-bar [↑](#footnote-ref-11)
12. https://seaborn.pydata.org/generated/seaborn.barplot.html [↑](#footnote-ref-12)
13. https://www.geeksforgeeks.org/bar-chart-using-plotly-in-python/ [↑](#footnote-ref-13)
14. https://hubofdata.ru/dataset/vladimir\_covid-19 [↑](#footnote-ref-14)
15. Create 2D bar graphs in different planes, https://matplotlib.org/stable/gallery/mplot3d/bars3d.html [↑](#footnote-ref-15)
16. https://translated.turbopages.org/proxy\_u/en-ru.ru.ee8d4fd6-6724bd90-8eab70e2-74722d776562/https/www.geeksforgeeks.org/python-data-visualization-tutorial/ [↑](#footnote-ref-16)