**Дипломная работа по теме:**

Сравнение различных библиотек для визуализации данных:

Matplotlib, Seaborn и Plotly

Автор: Зацаринный Сергей Юрьевич

**Оглавление дипломной работы:**

1. Введение.

Анализ современных подходов и инструментов в области визуализации данных

Обоснование выбора использования Python для приложений визуализации данных

2. Выбор библиотек Python

Краткий обзор библиотек Python

3. Детальный разбор библиотек визуализации с фрагментами кода и картинками

Представление Matplotlib

Представление Seaborn

Представление Plotly

4. Обоснование критериев выбора библиотек визуализации

Библиотека Matplotlib

Библиотека Seaborn

Библиотека Plotly

5. Анализ требований к приложениям визуализации данных

Анализ библиотеки Matplotlib

Анализ библиотеки Seaborn

Анализ библиотеки Plotly

6. Сравнительный анализ применения библиотек Matplotlib, Seaborn, Plotly

7. Заключение.

8. Источники

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### **1. Введение**

Визуализация данных играет ключевую роль в современной аналитике, предоставляя мощный инструмент для интерпретации сложной информации. Кроме того визуализация данных представляет собой динамичный и увлекательный аспект науки о данных, охватывающий процесс отображения значительных массивов данных для выявления скрытых взаимосвязей. Графики, диаграммы и интерактивные визуализации облегчают донесение результатов и помогают исследователям и бизнесу принимать обоснованные решения, выявлять тенденции и представлять результаты анализа.

Язык программирования Python занимает лидирующие позиции среди языков программирования, используемых для анализа данных, благодаря богатой экосистеме библиотек и широкому сообществу разработчиков. Python предлагает множество библиотек, предназначенных для визуализации данных. Особенностью языка является возможность создания как статических, интерактивных и трехмерных графиков с использованием популярных библиотек визуализации, таких как Altair, Bokeh, Ggplot, Matplotlib, Plotly, Seaborn.

Цель данной работы – провести сравнительный анализ этих библиотек, выявить их преимущества и недостатки, а также предложить рекомендации по выбору подходящего инструмента в зависимости от задач визуализации.

#### **2. Выбор библиотек Python**

Язык программирования Python предлагает множество библиотек, предназначенных для визуализации данных, что может затруднить выбор наиболее подходящей для конкретной задачи. Среди множества библиотек визуализации на Python можно выделить наиболее популярные и широко используемые:

**2.1 Altair**

Altair, библиотека Python для декларативной статистической визуализации данных, основанная на языках визуализации Vega и Vega-Lite.

Плюсы:

Код минимальный и последовательный, создает красивые и эффективные визуализации. Термин “декларативный” означает, что в процессе построения любого графика от пользователя требуется только указать связи между столбцами данных и соответствующими каналами кодирования, такими как ось x, ось y, цвет и так далее. Остальные тонкости построения графика автоматически регулируются системой.

Минусы:

Не создает интерактивных визуализаций, только статистические визуализации.

Ограниченный набор основных графиков, например boxplot, отсутствуют в Altair.

**2.2 Bokeh**

Bokeh - это библиотека Python для создания интерактивных визуализаций для современных веб-браузеров. Она помогает создавать красивые графики - от простых графиков до сложных приборных панелей с потоковыми наборами данных. С помощью Bokeh вы можете создавать визуализации на основе JavaScript без написания какого-либо JavaScript.

Плюсы:

Bokeh позволяет строить сложные статистические графики быстро и с помощью простых команд

Bokeh предоставляет вывод в различных средах, таких как html, блокнот и сервер

Bokeh может преобразовывать визуализации, написанные на других библиотеках

Bokeh имеет набор макетов и различных вариантов стилизации для визуализации

Минусы:

Как и любая другая новая библиотека с открытым исходным кодом, Bokeh находится в стадии активного развития, что может сделать код, который вы пишете сегодня, непригодным для повторного использования в будущем.

**2.2 Matplotlib**

Matplotlib - это библиотека двумерного черчения на языке Python, которая позволяет создавать рисунки типографского качества в различных печатных форматах и интерактивных средах на разных платформах. Matplotlib можно использовать в скриптах Python, оболочках Python и IPython, блокноте Jupyter, серверах веб-приложений.

Плюсы:

Matplotlib поддерживает широкий спектр популярных графиков, включая двумерные и трехмерные диаграммы, гистограммы, спектры мощности, графики ошибок и диаграммы рассеяния.

Matplotlib является универсальным инструментом визуализации для всевозможных данных.

Matplotlib позволяет пользователям контролировать различные параметры визуализации на нескольких уровнях программы.

Matplotlib имеет расширения программы для создания визуализаций данных в специфических исследованиях, к примеру графики в криволинейных координатах.

Matplotlib позволяет создавать анимированные графики.

Matplotlib имеет наработанные методы применения команд для создания визуализаций, доступных в галерее рисунков и графиков, созданных пользователями библиотеки.

Минусы:

Matplotlib требует отдельного изучения и освоения возможностей программы.

**2.3 Plotly**

Plotly семейство графических библиотек с открытым исходным кодом для Python, R, Julia, Javascript, ggplot2, F#, MATLAB® и Dash. Здесь будет рассмотрена Plotly Python библиотека.

Построенная на базе JavaScript-библиотеки Plotly (plotly.js), библиотека plotly позволяет пользователям Python создавать красивые интерактивные веб-визуализации, которые можно отображать в блокнотах Jupyter, сохранять в отдельные HTML-файлы или использовать в качестве части веб-приложений, построенных на чистом Python, с помощью Dash.

Плюсы:

Plotly позволяет создавать интерактивные визуализации, построенные с использованием D3.js, даже не зная D3.js.

Plotly позволяет легко делиться онлайн с несколькими людьми.

Plotly также может использоваться людьми без технического образования для создания интерактивных графиков путем загрузки данных и использования графического интерфейса Plotly.

Plotly позволяет встраивать интерактивные графики в проекты или веб-сайты с помощью iframes или html.

Plotly позволяет создавать интерактивные графики простым и быстрым способом.

Минусы:

В Plotly отсутствует публикация статических документов с высококачественными векторными изображениями.

В Plotly есть ограничение на количество цветовых палитр для создаваемых рисунков.

Plotly с открытым исходным кодом ограниченная версия по сравнению с коммерческой версией и возможности библиотеки зависят от поддержки пользователей.

Plotly предполагает полноценное использование с помощью специального интерфейса Dash.

**2.3 Seaborn**

Seaborn - это библиотека для создания статистических графиков на Python. Интерфейс библиотеки ориентирован на использование библиотеки Matplotlib и интегрирован со структурами данных Pandas.

Плюсы:

Seaborn имеет функциональный интерфейс для построения сложных графиков.

Seaborn имеет встроенные темы для улучшения эстетики картинок.

Seaborn имеет встроенные функции работы с массивами данных и внутренне выполняет необходимое семантическое отображение и статистическое агрегирование для создания информативных графиков.

Seaborn позволяет создавать многослойные статистические графики и распределения.

Минусы:

Seaborn имеет специфическое ограничение, поскольку предпочтителен для создания визуализации статистических данных.

#### **3. Детальный разбор библиотек**

Остановимся на библиотеках Matplotlib, Plotly, Seaborn. Данные библиотеки были выбраны для анализа из-за их уникальных возможностей, популярности в сообществе и разнообразия задач, которые они решают.

##### **3.1. Matplotlib**

Matplotlib - это библиотека Python, которая облегчает создание двумерных графиков для массивов данных. Хотя она была разработана как средство эмуляции графических команд языка MATLAB, она не зависит от MATLAB и может быть использована объектно-ориентированной среде Python. Хотя Matplotlib в основном написан на чистом Python, он широко использует NumPy и другой код расширения, чтобы обеспечить оптимальную производительность даже для больших массивов. Релизы библиотеки Matplotlib доступны в виде бинарных пакетов дистрибутива для macOS, Windows и Linux на <https://pypi.org/project/matplotlib>. Есть сторонние дистрибуторы и компиляция из источников. Инсталляция Matplotlib с помощью pip:

python -m pip install -U pip

python -m pip install -U matplotlib

Можно приступать к созданию визуализаций:

import matplotlib.pyplot as plt

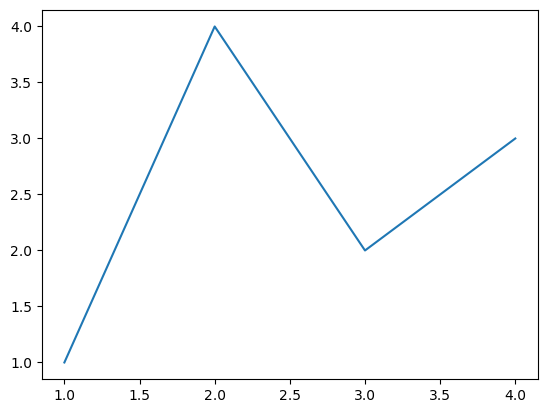
import numpy as np

Самый простой способ создания рисунка с осями - это использование pyplot.subplots. Затем можно использовать Axes.plot, чтобы нарисовать некоторые данные на осях, и show, чтобы отобразить рисунок:

fig, [ax](https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.html#matplotlib.axes.Axes) = [plt.subplots](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html#matplotlib.pyplot.subplots)() # Создаем фигуру с координатами

ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Передаем данные по координатам

plt.show() # Печать фигуры



С помощью Matplotlib можно отобразить данные на “фигурах” (например, окнах, виджетах Jupyter и т.д.), каждый из которых может содержать одну или несколько осей - областей, где точки могут быть заданы в терминах координат x-y (или theta-r в полярном графике, x-y-z в 3D графике и т.д.). Фигура это основной объект Matplotlib отслеживает все дочерние оси, группу "специальных художников” Artists (заголовки, легенды фигур, цветные полосы и т. д.) и даже вложенные под-фигуры.

Обычно фигура создается с помощью одной из следующих функций:

fig = plt.figure() # пустая фигура без координат

fig, [ax](https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.html#matplotlib.axes.Axes) = plt.subplots() # фигура с координатами

fig, [axs](https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) = plt.subplots(2, 2) # фигура с сеткой координат 2х2

# фигура с одной координатой слева и двумя координатами справа:

fig, [axs](https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) = plt.subplot\_mosaic([['left', 'right\_top'],

['left', 'right\_bottom']])

subplots() и subplot\_mosaic - это удобные функции, которые дополнительно создают объекты координаты внутри фигуры, но можно вручную добавить координаты позже.

Координаты “Axes” это объект, прикрепленный к рисунку, который содержит область для построения данных и обычно включает два (или три в случае 3D) объекта Axi, которые обеспечивают метки и тики для масштабирования данных в Axes. Каждый Axes также имеет заголовок (задается через set\_title()), x-метку (задается через set\_xlabel()) и y-метку (задается через set\_ylabel()). Методы манипулирования Axes являются основным интерфейсом для настройки большинства частей графика (добавление данных, управление масштабами и пределами осей, добавление меток и т. д.).

Axis эти объекты задают масштаб и границы, генерируют тики (метки на оси) и тикли (строки, обозначающие тики). Расположение тиков определяется объектом Locator, а строки меток форматируются Formatter. Сочетание правильного Locator и Formatter дает очень тонкий контроль над расположением и метками галочек.

Artists по сути, все, что видно на фигуре, является артистом (даже объекты Figure, Axes и Axis). Сюда входят текстовые объекты, объекты Line2D, объекты коллекций, объекты Patch и т. д. При рендеринге фигуры все художники перерисовываются на холст. Большинство “художников” привязаны к одной оси; такой “художник” не может быть общим для нескольких осей или перемещаться с одной на другую.

Типы входных данных для функций построения. Функции построения графиков ожидают в качестве входных данных numpy.array или numpy.ma.masked\_array, или объекты, которые можно передать в numpy.asarray. Классы, похожие на массивы, такие как объекты данных pandas и numpy.matrix, могут работать не так, как предполагалось. Обычно принято преобразовывать их в объекты numpy.array перед построением графика.

Преобразование матрицы numpy.matrix в numpy.array:

b = np.matrix([[1, 2], [3, 4]])

b\_asarray = np.asarray(b)

Большинство методов также анализируют индексируемые в строку объекты, такие как dict, структурированный массив numpy или pandas.DataFrame. Matplotlib позволяет указывать ключевой аргумент data и строить графики, передавая строки, соответствующие переменным x и y.

from matplotlib import pyplot as plt

import numpy as np

np.random.seed(19680801) # запускаем генератор случайных чисел

data = {'a': np.arange(50),

'c': np.random.randint(0, 50, 50),

'd': np.random.randn(50)}

data['b'] = data['a'] + 10 \* np.random.randn(50)

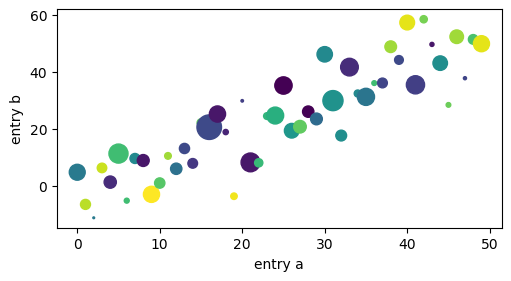
data['d'] = np.abs(data['d']) \* 100

fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')

ax.scatter('a', 'b', c='c', s='d', data=data)

ax.set\_xlabel('entry a')

ax.set\_ylabel('entry b')



Если нужно строить одни и те же графики снова и снова с разными наборами данных или контейнер Matplotlib, можно использовать рекомендуемую ниже сигнатурную функцию.

from matplotlib import pyplot as plt

import numpy as np

def my\_plotter(ax, data1, data2, param\_dict):# Вспомогательная функция

out = ax.plot(data1, data2, \*\*param\_dict)

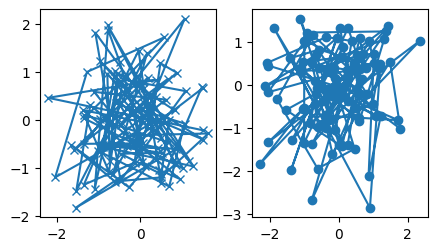
return out

data1, data2, data3, data4 = np.random.randn(4, 100) # 4 произвольных набора данных

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(5, 2.7))

my\_plotter(ax1, data1, data2, {'marker': 'x'})

my\_plotter(ax2, data3, data4, {'marker': 'o'})



Стили Artists, большинство методов черчения имеют опции стилизации для Artists, доступные либо при вызове метода черчения, либо из "сеттера" Artist. В приведенном ниже графике вручную установлен цвет, ширина линии и стиль линии для Artist, созданных методом plot, а стиль второй линии установлен уже после этого с помощью set\_linestyle.

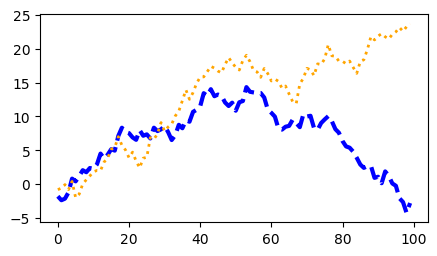
fig, ax = plt.subplots(figsize=(5, 2.7))

x = np.arange(len(data1))

ax.plot(x, np.cumsum(data1), color='blue', linewidth=3, linestyle='--')

l, = ax.plot(x, np.cumsum(data2), color='orange', linewidth=2)

l.set\_linestyle(':')



Matplotlib имеет очень гибкий набор цветов, который подходит для большинства Artists. Ширина линий, стили линий и размеры маркеров, ширина линий обычно указывается в типографских пунктах (1 пт = 1/72 дюйма) и доступна для Artists, имеющих строчные линии. Аналогично, строчные линии могут иметь стиль линии.

Здесь рассмотрена часть руководства пользователя под названием Quick start guide, полное руководство находится по адресу https://matplotlib.org/stable/users/index.html

##### **3.2. Seaborn**

Seaborn, библиотека визуализации данных на базе Python, упрощает создание привлекательных статистических графиков, предоставляя удобный интерфейс поверх matplotlib. Его расширенные функции построения графиков, включая jointplot, pairplot и heatmap, позволяют аналитикам выявлять сложные закономерности и взаимосвязи в данных, что делает его незаменимым инструментом в арсенале исследователя данных.

Официальные релизы Seaborn можно установить из PyPI <https://pypi.org/project/seaborn>

pip install seaborn

Базовый вызов pip установит Seaborn и, при необходимости, его обязательные зависимости. Можно включить дополнительные зависимости, которые дают доступ к некоторым расширенным возможностям.

Рекомендуется после установки seaborn протестировать его, загрузить и построить один из наборов данных (параметры тела пингвинов) для примера:

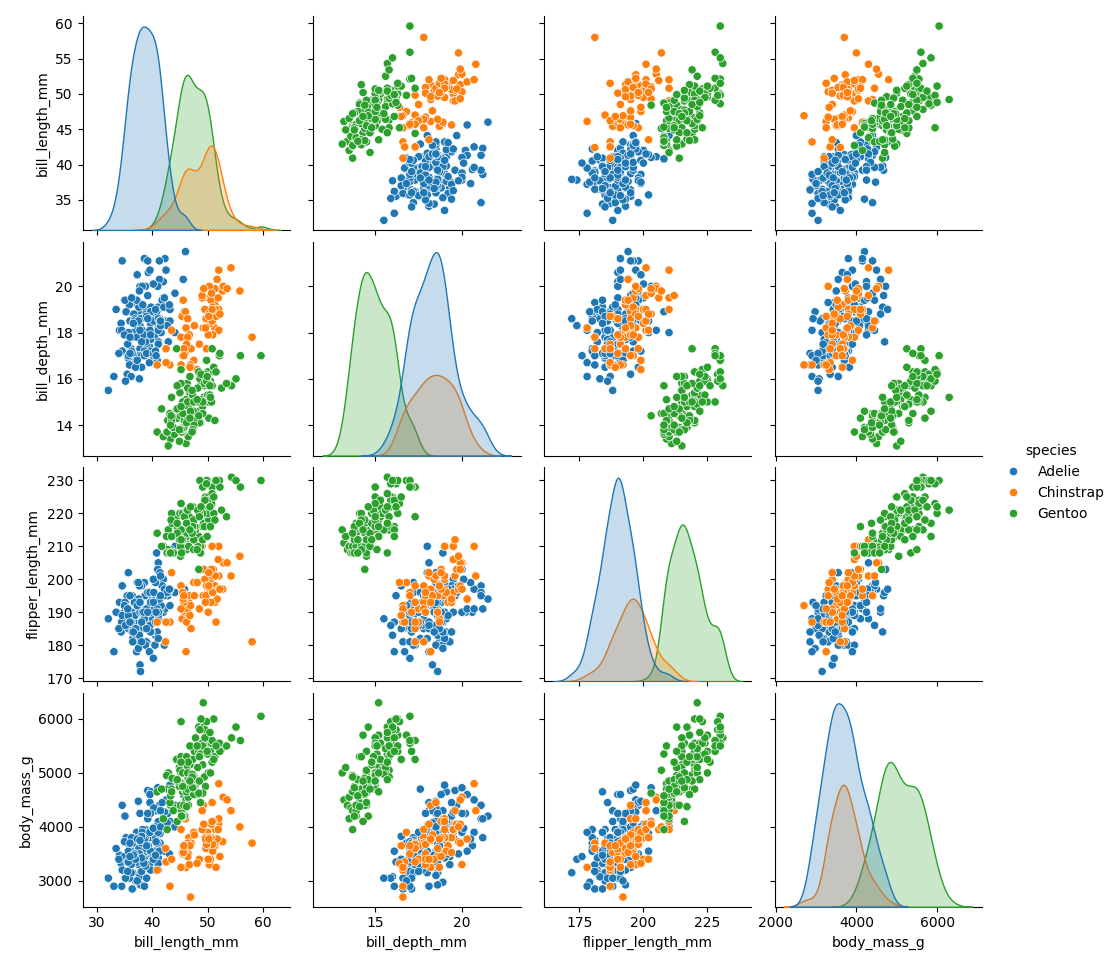
import seaborn as sns

import matplotlib.pyplot as plt

df = sns.load\_dataset("penguins")

sns.pairplot(df, hue="species")

plt.show()



Тестовый пример показывает, что Seaborn функции отображения картинок передает библиотеке Matplotlib.

Рассмотрим пример построения графика из руководства <https://seaborn.pydata.org/tutorial/introduction.html>

import seaborn as sns# импорт библиотеки seaborn

import matplotlib.pyplot as plt# импорт библиотеки matplotlib

sns.set\_theme()# применяем тему картинок по умолчанию

tips = sns.load\_dataset("tips")# загружаем набор данных

# создаем визуализацию

sns.relplot(

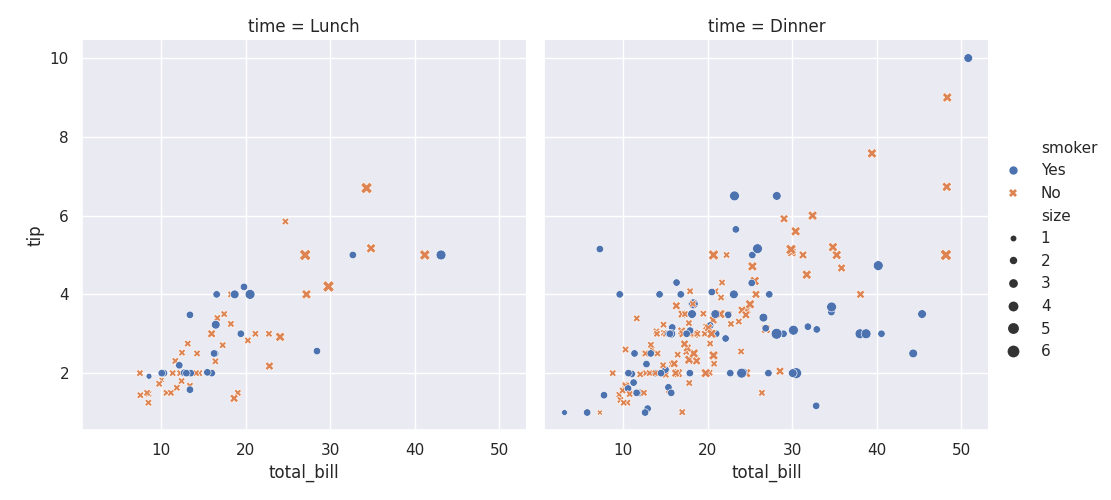
data=tips,

x="total\_bill", y="tip", col="time",

hue="smoker", style="smoker", size="size",

)

plt.show()# печатаем визуализацию



Этот график показывает взаимосвязь между пятью переменными в наборе данных tips с помощью одного вызова функции seaborn relplot(). Определены только имена переменных и их роли в графике. В отличие от прямого использования matplotlib, не было необходимости указывать атрибуты элементов графика в виде значений цвета или кодов маркеров. Seaborn выполнил перевод значений в дата-фрейме в аргументы, понятные matplotlib. Такой декларативный подход позволяет сосредоточиться на интерпретации графиков, а не на деталях управления matplotlib.

Сделаем столбчатую диаграмму, показывающую среднюю сумму чаевых, получаемых официантом каждый день. Код Seaborn выглядит так:

import matplotlib.pyplot as plt

import seaborn as sns

tips = sns.load\_dataset("tips")

(

sns.barplot(

data=tips, x="day", y="tip",

estimator="mean", errorbar=None,

)

.set(title="Daily Tips ($)")

)

plt.show()

Seaborn позволяет получить к наборов данных образцов Seaborn онлайн-доступ с помощью функции load\_dataset(). Можно увидеть список свободно доступных файлов в их репозитории GitHub. Чтобы получить тот, который нужен, сообщить load\_dataset() имя файла, содержащего интересующий набор данных, и он будет загружен в pandas DataFrame для использования.

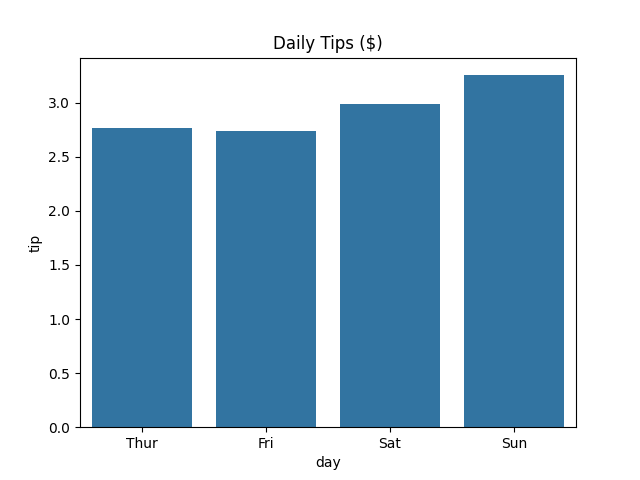
Фактическая столбчатая диаграмма создается с помощью barplot() функции seaborn. Указаны data=tipsDataFrame, который будет использоваться, а также функции построить столбцы day и tip из него. Они содержат день получения чаевых и сумму чаевых соответственно.

Важный момент, заключается в том, что barplot() функция seaborn, как и все функции построения графиков seaborn, может интуитивно понимать pandas DataFrames. Чтобы указать столбец данных для использования, вы передается имя столбца как строка. Нет необходимости писать код pandas для идентификации каждой серии, которую нужно построить.

Параметр estimator="mean" сообщает seaborn, что нужно построить график средних y значений для каждой категории x. Это означает, что график будет показывать средние чаевые за каждый день. Можно быстро настроить его, чтобы не использовать общие статистические функции, такие как sum, max, min, и median, а estimator="mean" это значение по умолчанию. График также будет показывать планки погрешностей по умолчанию. Задав errorbar=None , можно их отключить.

Функция barplot() создаст график, используя переданные ей параметры, и пометит каждую ось именем столбца данных. После barplot() завершения она возвращает Axes объект matplotlib, содержащий график. Чтобы задать графику заголовок, нужно вызвать метод Axes объекта .set() и передать ему нужный заголовок.

Функция load\_dataset(), в отличие от read\_csv(), автоматически преобразует столбцы строк в Categorical тип данных pandas. Далее создается график, с помощью Matplotlib.



Функциональный интерфейс seaborn содержит набор функций построения графиков для создания различных типов графиков. Пример использования функции barplot() выше. Функциональный интерфейс классифицирует свои функции построения графиков по нескольким общим типам.

Функции построения графиков являются частью классического функционального интерфейса Seaborn. Более современный способ использования Seaborn — это использование так называемого интерфейса объектов. Это обеспечивает декларативный синтаксис, то есть вы определяете, что хотите, используя различные объекты, а затем позволяете seaborn объединять их в ваш график. Это приводит к более последовательному подходу к созданию графиков, что делает интерфейс более простым для изучения. Он также скрывает базовую функциональность Matplotlib даже больше, чем функции построения графиков.

Когда строится график с использованием объектов Seaborn, первым используемым объектом является Plot. Этот объект ссылается на DataFrame, данные которого являются источником визуализации. При использовании интерфейса объектов seaborn принято импортировать его в Python с псевдонимом so. Чтобы создать Plot объект, вызывается его конструктор и передается DataFrame, содержащий данные, и имена столбцов, содержащих данные Series, которые нужно отобразить. У Plot объекта теперь есть данные для работы. Объект Plot содержит свой собственный .show() метод для его отображения.

График данных нигде не виден. Это потому, что объект Plot является только фоном для графика. Чтобы увидеть какой-то контент, нужно создать его, добавив один или несколько Mark объектов к Plot объекту. Mark объект является базовым классом целого ряда подклассов, каждый из которых представляет отдельную часть визуализации данных.

Следует отметить, что Plot объект может быть повторно использован для ряда различных графиков. Например, если назначить Plot объект переменной, такой как temperatures = so.Plot(data=crossings ...), можно впоследствии повторно использовать тот же объект и создавать различные типы графиков, добавляя в него разное содержимое.

Интерфейс объектов seaborn разработан, чтобы предоставить более интуитивный и расширяемый способ визуализации данных. Достигается это за счет модульности.

Интерфейс объектов также позволяет создавать более сложные графики без необходимости использования более сложного кода для этого.

Интерфейс объектов позволяет создавать несколько под-участков, внося небольшие изменения в существующий код, но не усложняя его. С объектами нет необходимости начинать с самого начала с совершенно другой функцией.

API объектов все еще разрабатывается. Разработчики Seaborn намереваются сделать API объектов своим будущим. Благодаря своим двойным API и своей основе в Matplotlib, Seaborn позволяет создавать широкий спектр различных графиков.

##### **3.3. Plotly**

Библиотека Plotly для построения графиков на Python позволяет создавать интерактивные графики издательского качества в режиме онлайн. Plotly выделяется среди остальных библиотек благодаря интерактивным возможностям. Инсталляция:

pip install plotly==5.24.1

Этот пакет содержит все необходимое для записи рисунков в автономные HTML-файлы.

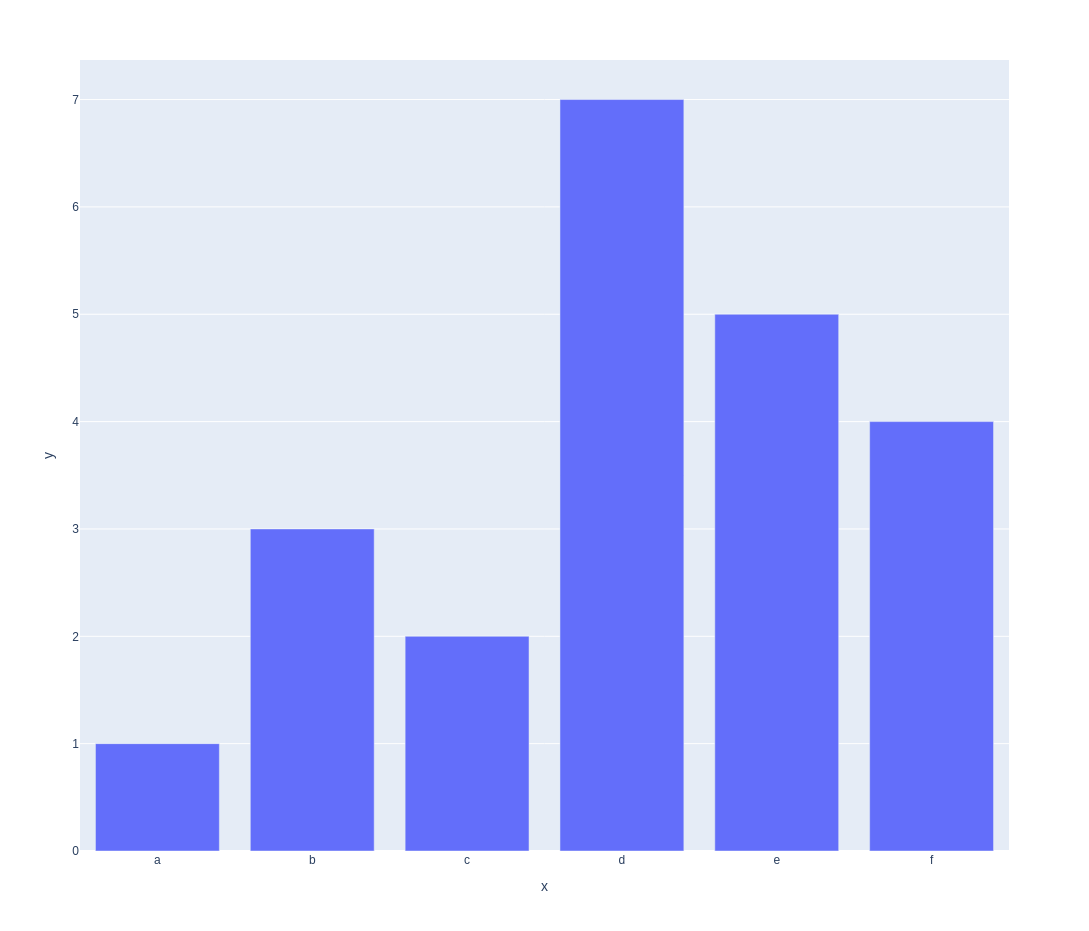
Попробуем создать график:

import plotly.express as px

fig = px.bar(x=["a", "b", "c", "d", "e", "f"], y=[1, 3, 2, 7, 5, 4])

fig.write\_html('first\_figure.html', auto\_open=True)

Созданная гистограмма публикуется в виде страницы браузера.



Пакет Plotly Python предназначен для создания, манипулирования и визуализации графических фигур (т.е. графиков, диаграмм, карт и схем), представленных структурами данных, также называемыми фигурами. Процесс рендеринга использует JavaScript-библиотеку Plotly.js под капотом, хотя разработчикам Python, использующим этот модуль, очень редко приходится взаимодействовать с Javascript-библиотекой напрямую, если вообще приходится. Фигуры могут быть представлены в Python либо как dicts, либо как экземпляры класса plotly.graph\_objects.Figure, и перед передачей в Plotly.js они сериализуются как текст в JavaScript Object Notation (JSON).

Plotly поставляется с несколькими модулями для создания визуализаций, то есть предоставляет выбор, как его использовать.

express: Высокоуровневый интерфейс для создания быстрых визуализаций. Это обертка вокруг модуля Plotly graph\_objects.

graph\_objects: Низкоуровневый интерфейс для работы с фигурами, трассами и макетами. figure\_factory: Фабрики фигур - это специальные функции для создания очень специфических типов графиков.

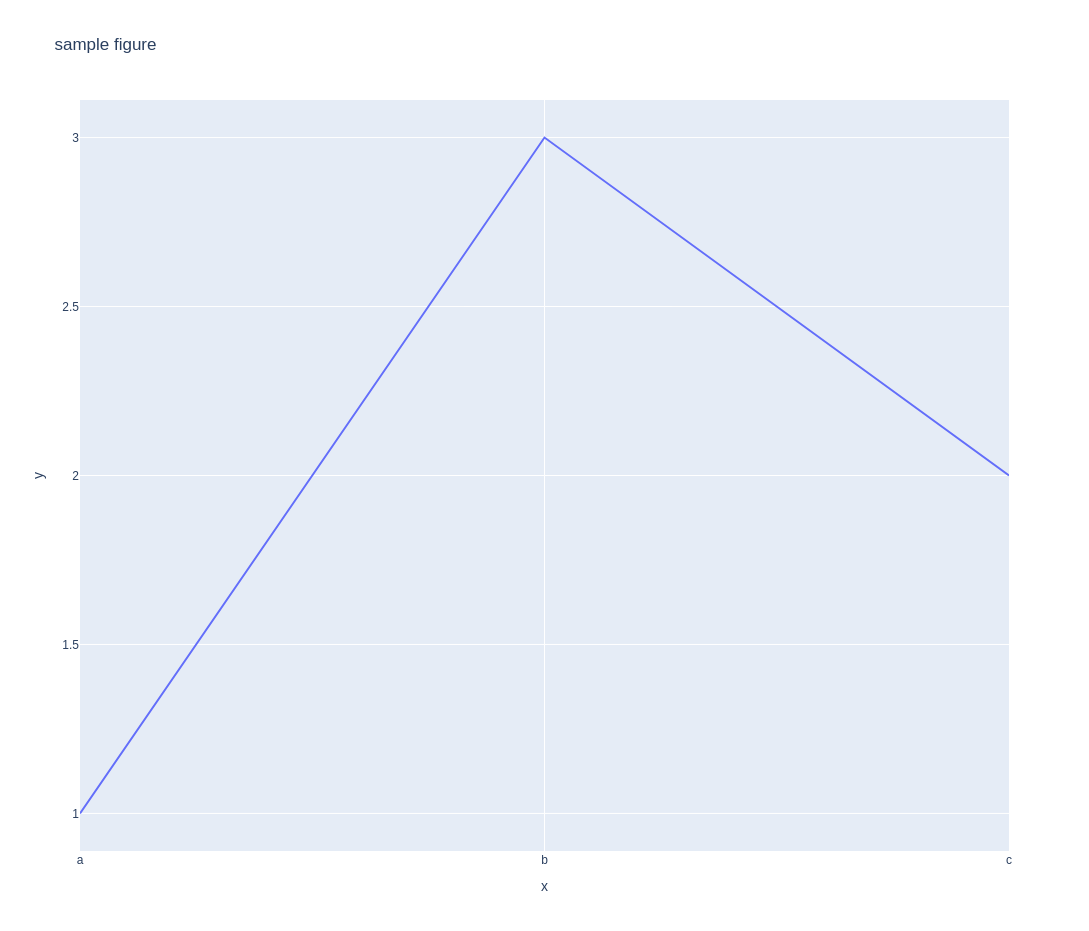
Просмотреть структуру данных, лежащую в основе любого объекта plotly.graph\_objects.Figure, включая те, которые возвращает Plotly Express, можно с помощью print(fig) или, в JupyterLab, с помощью специального рендеринга fig.show("json"). Фигуры также поддерживают методы fig.to\_dict() и fig.to\_json().

import plotly.express as px

fig = px.line(x=["a","b","c"], y=[1,3,2], title="sample figure")

print(fig)

fig.show()



Рисунки представлены в виде деревьев с именованными узлами, называемыми "атрибутами". Корневой узел дерева имеет три атрибута верхнего уровня: данные, макет и рамки. При работе с объектом plotly.graph\_objects.Figure атрибуты могут быть заданы либо непосредственно с помощью атрибутов объекта Python, например fig.layout.title.font.family="Open Sans", либо с помощью методов обновления и "магических подчеркиваний", например fig.update\_layout(title\_font\_family="Open Sans").

При построении фигуры нет необходимости заполнять каждый атрибут каждого объекта. Во время рендеринга слой JavaScript будет вычислять значения по умолчанию для каждого требуемого не указанного атрибута. Слой JavaScript будет игнорировать неизвестные атрибуты или неправильно сформированные значения, хотя модуль plotly.graph\_objects обеспечивает проверку значений атрибутов на стороне Python. Если присутствует ключ layout.template (по умолчанию), то значения по умолчанию сначала будут взяты из содержимого шаблона, и только если они отсутствуют, слой JavaScript выведет дальнейшие значения по умолчанию. Встроенный шаблон можно отключить, задав layout.template="none".

Plotly поддерживает более 40 уникальных типов графиков, охватывающих широкий спектр статистических, финансовых, географических, научных и трехмерных сценариев использования. Некоторые из них описаны ниже.

Box-and-whisker plot - график отображения распределения количественных данных на основе пятибалльной сводки (минимум, первый квартиль (Q1), медиана (Q2), третий квартиль (Q3) и максимум). Коробка простирается от значений квартиля Q1 до Q3, а усы - от краев коробки до 1,5\*IQR. IQR = (Q3 - Q1)

Histogram - гистограмма, точное представление распределения числовых данных.

Pie Chart - круговая диаграмма используется для представления числовой пропорции данных в виде кругового графика. Вся площадь диаграммы представляет 100% данных, длина дуги каждого среза представляет собой относительную процентную часть от целого.

3D scatter plots - трехмерная диаграмма рассеяния использует точки для представления значений трех различных числовых переменных. Она очень полезна для наблюдения взаимосвязи между тремя числовыми переменными.

Heatmap - тепловая карта, двумерное графическое представление данных, в котором значения матрицы представлены различными оттенками цветов.

Violin Plot - скрипичные графики похожи на бокс-графики, за исключением того, что они также показывают плотность вероятности данных при различных значениях. Другими словами, скрипичный график - это комбинация боксового графика и графика плотности.

Word Cloud - Облако слов, метод визуализации для представления частоты слов в заданном сегменте текста. Размер слова показывает, насколько часто оно встречается в тексте. Чем больше размер, тем больше важность (частота), а чем меньше размер, тем меньше важность (частота).

Plotly позволяет быстро создавать графики буквально в одну строку, публиковать их наиболее удобным способом, легко изменять вид опубликованных графиков для получения узкой выборки. В Plotly есть возможность углубиться в детали графика и настроить презентацию графика.

#### **4. Обоснование критериев выбора библиотек**

Визуализация данных - это процесс поиска, интерпретации и сравнения данных таким образом, чтобы они могли более четко передавать сложные идеи, облегчая тем самым выявление и анализ логических закономерностей. Визуализация данных важна для многих аналитических задач, включая сводки данных, анализ тестовых данных и анализ результатов моделирования. В Python есть множество библиотек, которые предоставляют полезные инструменты для извлечения информации из данных. Самые популярные из них - Matplotlib, Seaborn, Plotly.

Теперь рассмотрим факторы, которые следует учитывать при выборе пакета визуализации данных на Python.

Категории визуализаций: выбор библиотеки визуализации данных, которая способна генерировать конкретную визуализацию.

Формат вывода: выбор библиотеки визуализации данных, способную генерировать визуальные изображения в формате, совместимом с задачей, например SVG, PDF или PNG.

Формат ввода: выбор библиотеки визуализации данных, совместимую с конкретным типом данных, например структурированными или неструктурированными данными.

Совместимость: выбор библиотеки визуализации данных, способную работать с используемыми веб-браузерами.

Кривая обучения: некоторые библиотеки визуализации данных демонстрируют значительно более доступный процесс обучения по сравнению с другими. Наука о данных по своей сути сложна, и рекомендуется использовать библиотеку с минимальной кривой обучения.

Получив представление об определении библиотеки визуализации данных и факторах, которые следует учитывать при ее выборе, можно изучить множество библиотек, доступных для Python.

Matplotlib - это фундаментальный компонент набора инструментов визуализации данных Python, предлагающий основу с открытым исходным кодом для наглядного отображения сложных закономерностей в осмысленном виде. Matplotlib предоставляет пользователям разнообразный выбор графиков, инструменты настройки и множество функций для создания широкого спектра визуальных представлений. Коллекция предлагает необходимые ресурсы для создания линейных графиков, подчеркивающих тенденции, гистограмм для проведения сравнений и диаграмм рассеяния, подчеркивающих связи между данными. Matplotlib позволяет создавать многопанельные графики, которые позволяют более глубоко исследовать сложные наборы данных. Кроме того, с помощью модуля анимации Matplotlib разработчики могут создавать динамические визуальные эффекты, которые эффективно демонстрируют временные изменения и динамику изменения данных.

Модуль анимации в Matplotlib является примечательной особенностью, поскольку он позволяет создавать динамические изображения, которые эффективно демонстрируют временные колебания и динамику данных. Благодаря широкому распространению и исчерпывающей документации он стал предпочтительным инструментом как для начинающих, так и для опытных пользователей в области науки о данных и анализа. Более того, его интеграция с другими библиотеками Python повышает его полезность и удобство использования.

Seaborn, библиотека Python, расширяющая Matplotlib, повышает точность визуализации данных. В то время как Matplotlib обеспечивает прочную основу, Seaborn превосходит ее в повышении визуальной привлекательности статистических графиков. Seaborn упрощает процесс создания сложных графиков, эффективно передающих статистические данные, благодаря удобному интерфейсу. Seaborn обладает заметным преимуществом в плане возможностей создания легко интерпретируемых графиков с меньшим количеством кодирования. Библиотека отлично справляется с созданием сложных графиков, таких как тепловые карты, раскрывающие закономерности данных, и парные графики, идеально подходящие для демонстрации корреляции между многими переменными. Seaborn легко интегрируется со структурами данных Pandas, упрощая визуализацию данных на Python и ориентируясь на пользователей любого уровня подготовки, от новичков до экспертов.

Seaborn умеет создавать визуально привлекательные графики, которые эффективно передают статистическую информацию. Интерфейс разработан таким образом, чтобы быть понятным и удобным для пользователей, а значит, для создания сложных визуализаций требуется меньше кодирования. Seaborn - очень подходящий вариант для пользователей, независимо от их уровня подготовки, которые хотят усовершенствовать свои таланты в визуализации данных.

Plotly, известная библиотека визуализации данных Python, высоко ценится за свою универсальность, расширяя возможности Python по построению графиков в онлайн-режиме. Коллекция включает в себя разнообразные виды графиков, начиная от простых линейных диаграмм и заканчивая сложными 3D-представлениями. Plotly отличается тем, что делает упор на программную интерактивность, позволяя разработчикам вовлекать аудиторию и распространять динамические визуализации. Процедура создания интерактивных веб-графиков с помощью Plotly довольно проста. Графики Plotly можно интегрировать в онлайн-приложения, чтобы пользователи могли беспрепятственно работать с данными. Такая функциональность особенно удобна при обмене результатами с разнообразной аудиторией или при работе в команде, которая часто корректирует и реструктурирует информацию. Способность Plotly генерировать динамические и веб-совместимые визуальные эффекты делает его мощным инструментом для передачи информации. API разработан таким образом, чтобы быть понятным и простым для пользователей, что упрощает создание сложных графиков. Эта библиотека чрезвычайно полезна для команд, которым требуется совместная работа и обмен интерактивными визуализациями данных.

Подводя итог, можно сказать, что эти библиотеки Python для визуализации данных являются отличными вариантами для создания визуально привлекательных и глубоких визуализаций данных. Каждый вариант обладает своими преимуществами и достоинствами, что позволяет выбрать тот, который лучше всего подходит для визуализации ваших данных или проекта.

Выбор подходящей библиотеки зависит от задач, стоящих перед пользователем. **6.**

**5. Анализ требований к приложениям визуализации данных**

##### 

##### **5.1. Библиотека Matplotlib**

Используется в проектах, где требуется высокая кастомизация, например, научные исследования и публикации. Основные требования:

* Поддержка сложных графиков.
* Совместимость с форматом публикаций (PDF, PNG).

##### **5.2. Библиотека Seaborn**

Лучше всего подходит для аналитиков данных, где важна визуализация статистики. Основные требования:

* Лёгкость работы с большими массивами данных.
* Поддержка готовых шаблонов графиков.

##### **5.3. Библиотека Plotly**

Идеально для веб-приложений и дашбордов. Основные требования:

* Интерактивность графиков.
* Возможность публикации в вебе.

#### **6. Сравнительный анализ применения библиотек Matplotlib, Seaborn и Plotly**

Для выполнения сравнительного анализа библиотек визуализации данных определены ключевые критерии: функциональность, простота использования, гибкость настройки, интерактивность, а также производительность.

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### 

##### **Таблица сравнения**

| **Критерий** | **Matplotlib** | **Seaborn** | **Plotly** |
| --- | --- | --- | --- |
| **Функциональность** | Высокая, поддержка сложных графиков | Высокая, ориентирована на статистику | Высокая, поддержка 3D и интерактивности |
| **Простота** | Низкая, сложный синтаксис | Средняя, интуитивный интерфейс | Средняя, требует изучения структуры |
| **Гибкость** | Очень высокая | Средняя | Высокая |
| **Интерактивность** | Отсутствует | Отсутствует | Очень высокая |
| **Производительность** | Высокая | Высокая | Средняя, зависит от объёма данных |

##### **Обсуждение результатов**

1. **Matplotlib:**Эта библиотека идеально подходит для задач, требующих полной настройки и использования сложных графиков. Однако она требует большего времени на освоение из-за своей низкоуровневой структуры.
2. **Seaborn:**Более простой в использовании инструмент, подходящий для статистического анализа. Однако возможности кастомизации ограничены по сравнению с Matplotlib.
3. **Plotly:**Лидер в интерактивной визуализации. Эта библиотека особенно полезна для создания веб-дэшбордов и приложений, но её производительность снижается с увеличением объёма данных.

#### 

#### **7. Заключение**

На основании проведённого анализа можно сделать следующие выводы:

* Для задач, где требуется максимальная гибкость и сложность графиков, рекомендуется использовать **Matplotlib**.
* Если акцент делается на статистическую визуализацию и простоту, то оптимальным выбором будет **Seaborn**.
* Для интерактивных графиков и приложений с элементами динамики наиболее подходит **Plotly**.

Каждая из рассмотренных библиотек имеет свои сильные и слабые стороны. Выбор инструмента зависит от конкретных задач, требуемого уровня детализации и целевой аудитории.

#### **8. Источники**

* Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*.
* Waskom, M. (2021). *Seaborn: Statistical Data Visualization*.
* Plotly Technologies Inc. (2020). *Plotly Python User Guide*.
* McKinney, W. (2010). *Data Structures for Statistical Computing in Python*.