



ВИЗУАЛИЗАЦИЯ ВРЕМЕННЫХ РЯДОВ И ГЕОГРАФИЧЕСКИХ ДАННЫХ

- ❖ • Использование библиотеки Seaborn для визуализации временных рядов.
- ❖ • Создание интерактивных графиков с помощью библиотеки Plotly.
- ❖ • Визуализация географических данных на карте с использованием библиотеки GeoPandas.

SEABORN

Seaborn — это библиотека для улучшенной визуализации данных на основе *Matplotlib*. Она упрощает создание сложных графиков и добавляет стиль по умолчанию, что делает графики более привлекательными и информативными.

Seaborn предоставляет функции для построения тепловых карт, парных графиков и других сложных визуализаций, которые помогают лучше понять структуру данных.

SEABORN

- ✓ Seaborn — это библиотека для создания статистических графиков на Python. Она основывается на `matplotlib` и тесно взаимодействует со структурами данных `pandas`.
- ✓ Архитектура Seaborn позволяет вам быстро изучить и понять свои данные.
- ✓ Seaborn захватывает целые фреймы данных или массивы, в которых содержатся все ваши данные, и выполняет все внутренние функции, нужные для семантического маппинга и статистической агрегации для преобразования данных в информативные графики.
- ✓ Она абстрагирует сложность, позволяя вам проектировать графики в соответствии с вашими нуждами.

ВОЗМОЖНОСТИ БИБЛИОТЕКИ SEABORN

Графики для категориальных данных	«Ящик с усами» (box plot) — <code>sns.boxplot</code> ; уровневый график (violin plot) — <code>sns.violinplot</code> ; столбчатые графики (bar plot) — <code>sns.barplot</code> ; графики точек (point plot) — <code>sns.pointplot</code>
Сравнение множественных групп	Парные графики (pair plots), которые показывают взаимосвязи между всеми парами переменных; тепловые карты (heatmaps) для визуализации матриц корреляции
Кастомизация графиков	Возможность изменять стили и цветовые палитры с помощью встроенных функций — лёгкая настройка заголовков, меток осей и других элементов графиков
Многомерная визуализация	Графики с несколькими переменными, поддерживающие различные параметры, включая размер и цвет точек. Упрощённая визуализация сложных данных через использование FacetGrid и PairGrid
Интеграция с Pandas	Упрощённая работа с DataFrame для создания графиков напрямую из табличных данных
Поддержка временных рядов	Визуализация временных данных с помощью линейных графиков и других подходящих типов графиков
Тематические графики	Возможность создания графиков с использованием различных тем — например, dark, white grid и др.
Анализ данных	Визуализация результатов статистических тестов и моделей

ПРЕИМУЩЕСТВА И НЕДОСТАТКИ БИБЛИОТЕКИ SEABORN

Преимущества	Недостатки
Удобство использования. Поскольку Seaborn была написана на основе библиотеки визуализации данных Matplotlib, их довольно просто использовать вместе	Ограниченные возможности. Поскольку Seaborn построена на базе Matplotlib, её возможности ограничены функциональностью этой библиотеки
Доступность для начинающих пользователей. Seaborn предоставляет высокоуровневый интерфейс для создания сложных визуализаций с минимальным количеством кода	Производительность. Для очень больших наборов данных Seaborn может работать медленнее по сравнению с другими библиотеками, так как она использует сложные визуализации
Статистическая визуализация. Библиотека включает в себя встроенные функции для статистической визуализации (например, регрессионные графики, графики распределения и тепловые карты)	
Эстетика графиков. Благодаря современным цветовым палитрам и стилям Seaborn создаёт более привлекательные и информативные графики, чем Matplotlib	
Гибкость и расширяемость. Пользователи могут настраивать графики, добавлять элементы и комбинировать различные типы визуализаций	

ОСНОВНЫЕ ФУНКЦИИ SEABORN

1. Установить библиотеку с помощью **pip**:

```
pip install seaborn
```

2. Загрузка набора данных

```
import seaborn as sns
# Загрузка набора данных tips
tips = sns.load_dataset("tips")
# Отображение первых 5 строк набора данных
tips.head()
```

1. Другой способ подключения:

```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

ОСНОВНЫЕ ФУНКЦИИ SEABORN

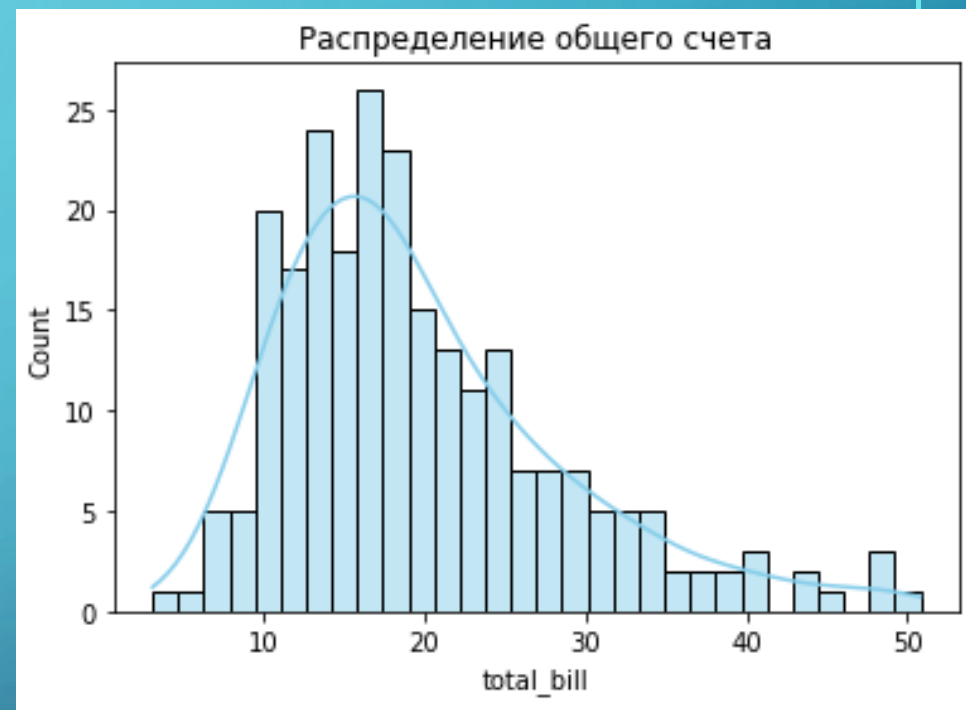
- ✓ Гистограммы позволяют визуализировать распределение переменной
- ```
sns.histplot()
```

Пример создания гистограммы для общего счета:

```
import seaborn as sns
import matplotlib.pyplot as plt
Создание гистограммы для отображения распределения общего счета
sns.histplot(data=tips, x="total_bill", bins=30, kde=True, color='skyblue')
Добавление заголовка к графику
plt.title("Распределение общего счета")
Отображение графика
plt.show()
```

Параметры:

- `data=tips` — указываем набор данных.
- `x="total_bill"` — ось X будет представлять общую сумму счета.
- `bins=30` — количество корзин (или интервалов) для гистограммы (можно настроить по желанию).
- `kde=True` — добавляет линию ядерной оценки плотности (KDE) для лучшего понимания распределения.
- `color='skyblue'` — настройка цветовой палитры графика



# ОСНОВНЫЕ ФУНКЦИИ SEABORN

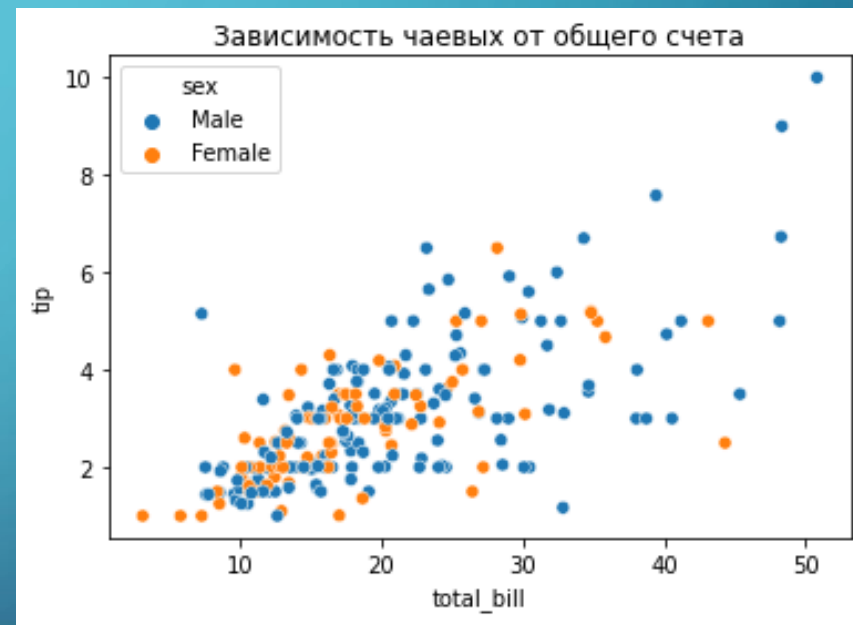
✓ Диаграмма рассеяния (Scatter Plot) используются для визуализации взаимосвязи между двумя `sns.scatterplot()`

Пример создания диаграммы рассеяния для отображения зависимости чаевых от общего

```
Создание диаграммы рассеяния для отображения зависимости чаевых от общего с
чета
sns.scatterplot(x="total_bill", y="tip", hue="sex", data=tips)
plt.title("Зависимость чаевых от общего счета")
plt.show()
```

Параметры:

`x="total_bill"` — ось X будет представлять общую сумму счета.  
`y="tip"` — ось Y будет представлять сумму чаевых.  
`data=tips` — указываем набор данных.  
`hue="sex"` — разбиваем данные по дополнительной категории.





# ОСНОВНЫЕ ФУНКЦИИ SEABORN

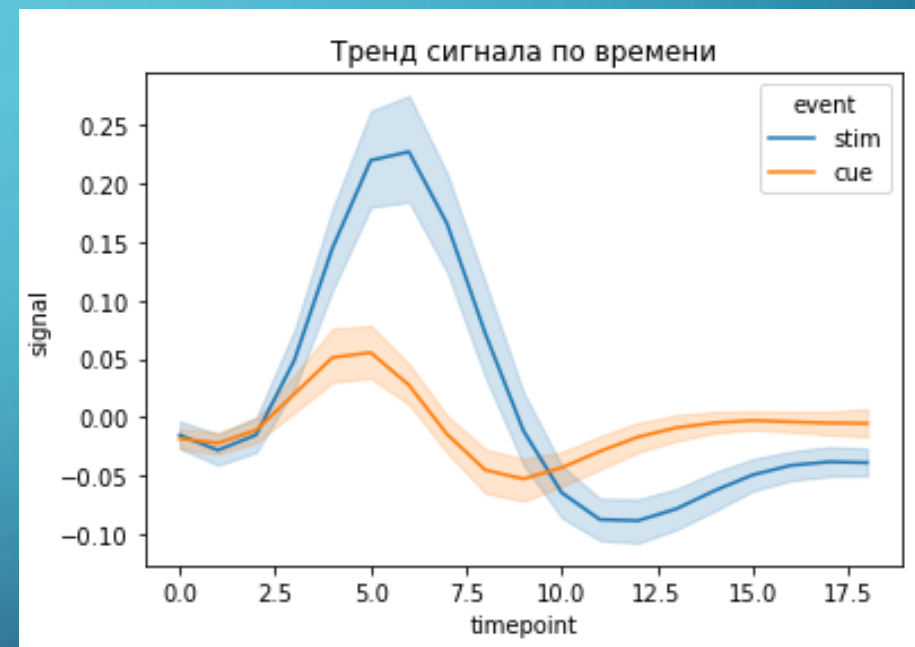
✓ Линейный график (Line Plot) помогает визуализировать тренды во времени: `sns.lineplot()`

Пример создания линейного графика для отображения тренда сигнала по времени:

```
Загрузка набора данных fmri
fmri = sns.load_dataset("fmri")
Создание линейного графика для отображения тренда сигнала по времени
sns.lineplot(x="timepoint", y="signal", hue="event", data=fmri)
plt.title("Тренд сигнала по времени")
plt.show()
```

Параметры:

x="timepoint" — ось X будет представлять временные точки.  
y="signal" — ось Y будет представлять значение сигнала.  
data=fmri — указываем набор данных.  
hue="event" — разбить данные по категории событие (stim или не stim)



# ОСНОВНЫЕ ФУНКЦИИ SEABORN

- ✓ Столбчатая диаграмма (Bar Plot) показывает средние значения (или другого указанного агрегирования) для категориальных

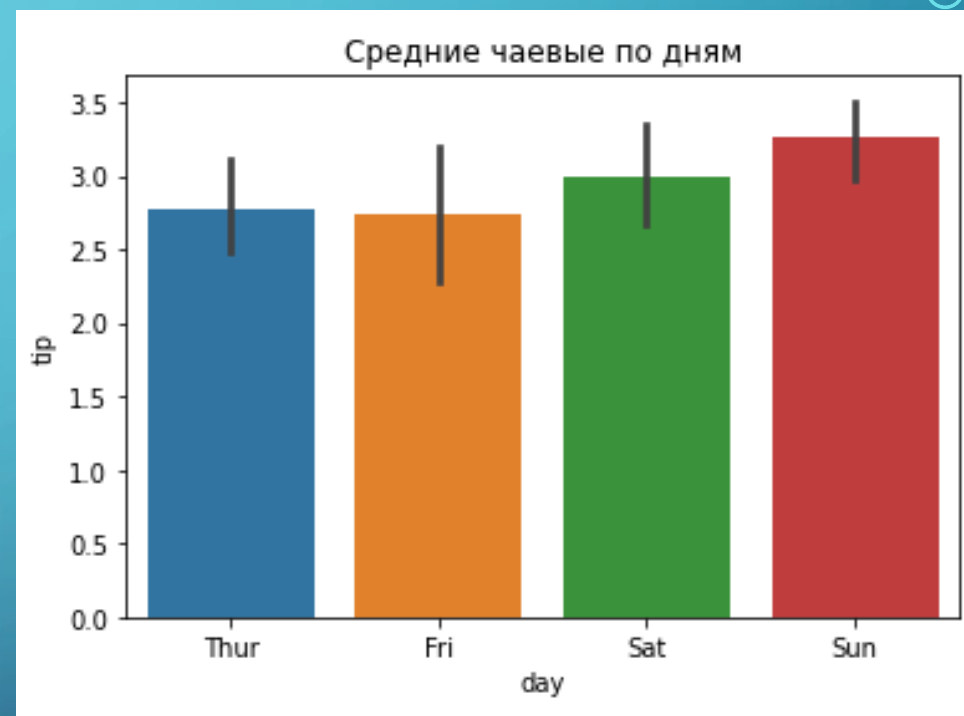
■ `sns.barplot()`

Пример вычисления средних чаевых (tip) для каждого дня недели (day):

```
import numpy as np
Создание столбчатой диаграммы для отображения средних чаевых по дням
sns.barplot(x="day", y="tip", data=tips, estimator=np.mean)
plt.title("Средние чаевые по дням")
plt.show()
```

Параметры:

- x: Указывает категориальную переменную (дни недели).
- y: Указывает числовую переменную (чаевые).
- data: Набор данных, который в данном случае — это tips.
- estimator: Этот параметр указывает функцию, используемую для агрегации значений. Здесь используется np.mean для вычисления средних чаевых.



Эта диаграмма позволяет быстро сравнить средние чаевые в разные дни, предоставляя информацию о поведении клиентов и тенденциях, связанных с чаевыми.

# ОСНОВНЫЕ ФУНКЦИИ SEABORN

- ✓ Коробчатая диаграмма (Box Plot) визуализирует распределение данных и выявляют в `sns.boxplot()`

Пример создания коробчатой диаграммы для отображения общего счета по дням:

```
Создание коробчатой диаграммы для отображения общего счета по дням
sns.boxplot(x="day", y="total_bill", hue="sex", data=tips)
plt.title("Коробчатая диаграмма общего счета по дням")
plt.show()
```

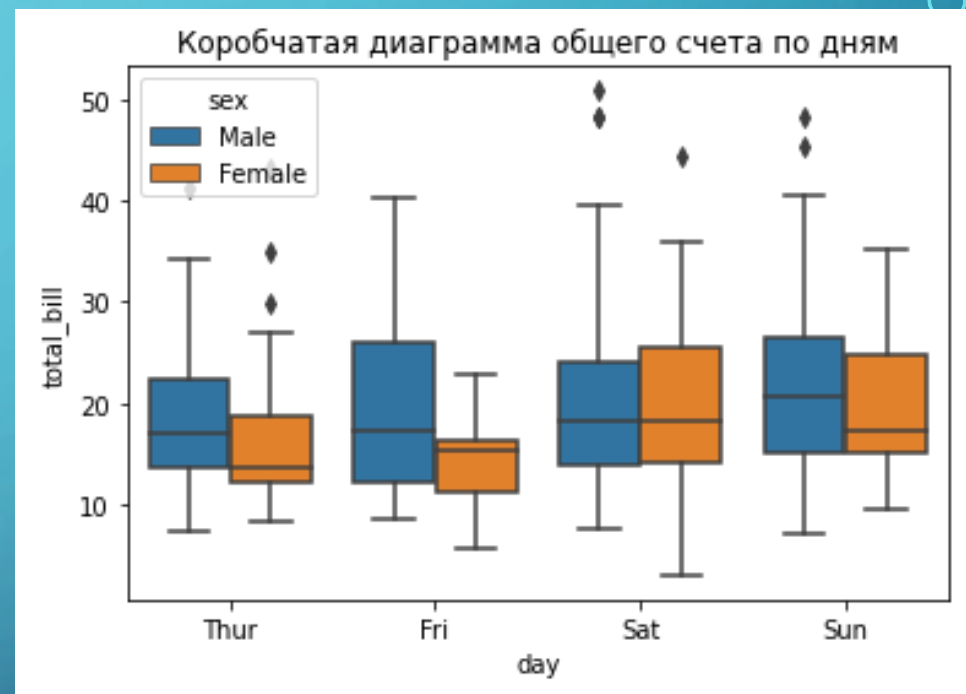
Параметры:

`x="day"` — ось X будет представлять дни недели.

`y="total_bill"` — ось Y будет представлять общий счет.

`data=tips` — указываем набор данных.

`hue="sex"` — разбиваем данные по дополнительной категории.



# ОСНОВНЫЕ ФУНКЦИИ SEABORN

- ✓ Тепловая карта (Heatmap) визуализирует матричные данные, например, корреляцию между переменными. Функция `sns.heatmap()`

Пример создания тепловой карты для отображения корреляционной матрицы:

```
Вычисление корреляционной матрицы
corr = tips.corr()
Создание тепловой карты для отображения корреляционной матрицы
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", square=True)
plt.title("Корреляционная матрица")
plt.show()
```

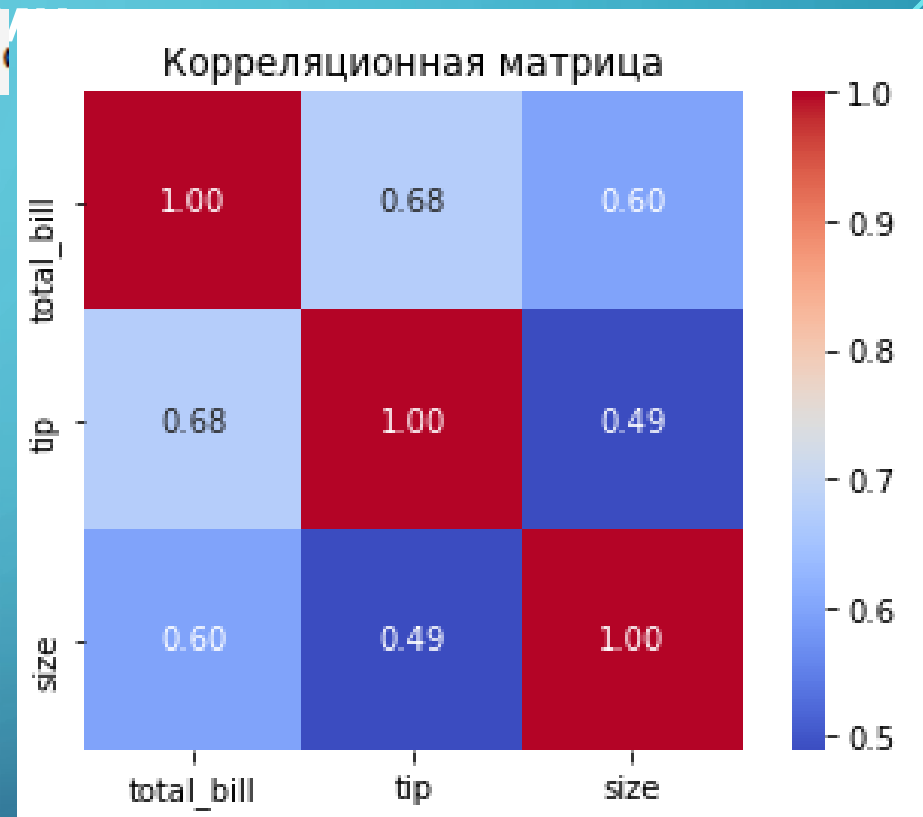
Параметры:

`annot=True`: Значения корреляции отображаются в ячейках, что облегчает интерпретацию графика.

`cmap='coolwarm'`: Используется цветовая палитра от холодных (синих) к теплым (красным) цветам, что визуально подчеркивает различия в корреляции.

`fmt=".2f"`: Форматирует значения до двух знаков после запятой, улучшая читаемость.

`square=True`: Обеспечивает квадратную форму ячеек, что делает тепловую карту более симметричной и эстетически приятной.



Тепловая карта позволяет быстро оценить взаимосвязи между переменными, где цвет ячеек указывает на силу и направление корреляции (положительная или отрицательная). Это помогает выявить значимые зависимости, которые могут быть полезны для дальнейшего анализа.

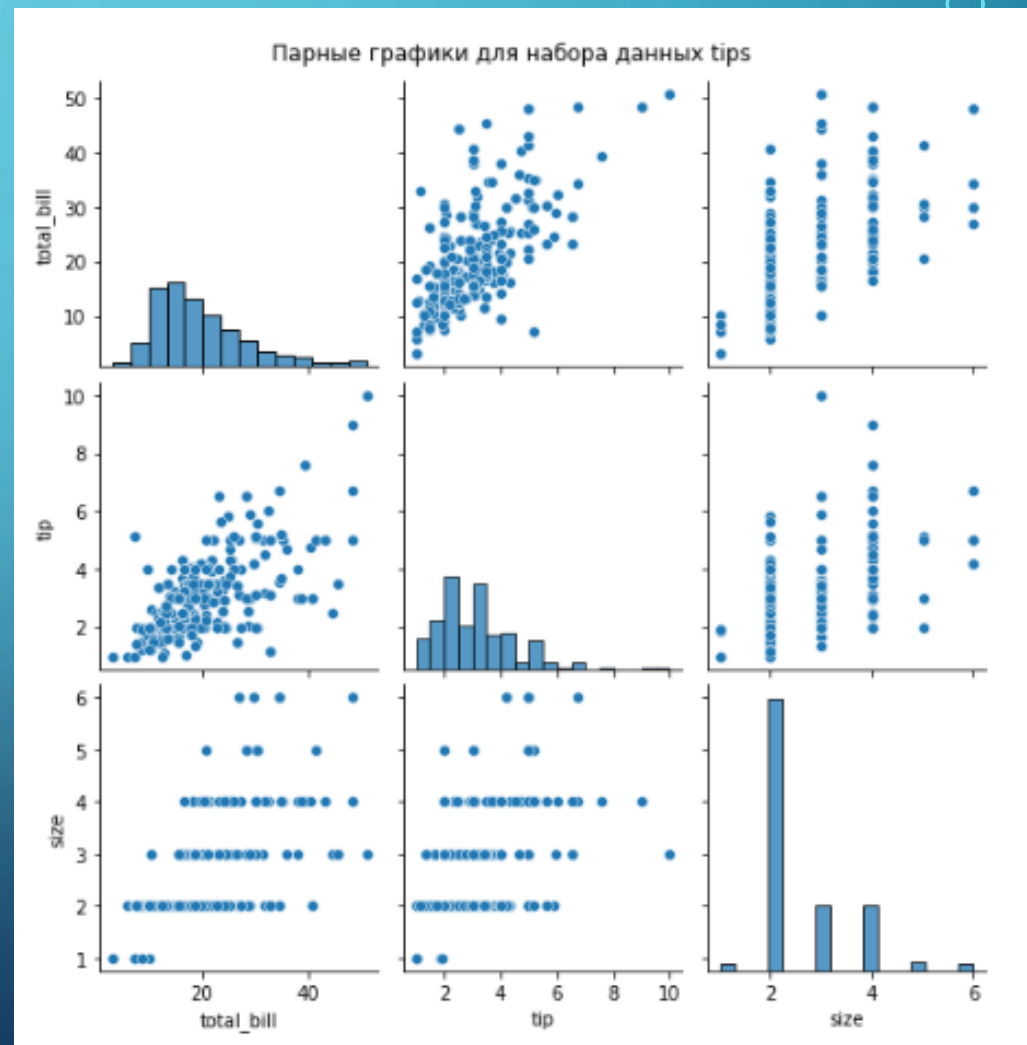
# ОСНОВНЫЕ ФУНКЦИИ SEABORN

- ✓ Парные графики (Pair Plot) визуализирует отношения между несколькими переменными `sns.pairplot()`:

Пример создания парных графиков для набора данных tips:

```
Создание парных графиков для набора данных tips
pair_plot = sns.pairplot(tips)
pair_plot.fig.suptitle("Парные графики для набора данных tips", y=1.02)
plt.show()
```

Функция `sns.pairplot()` генерирует матрицу графиков, где каждый график показывает зависимость между двумя переменными. Это позволяет быстро оценить потенциальные корреляции и паттерны в данных, а также выявить выбросы. Парные графики упрощают анализ многомерных данных, предоставляя возможность одновременно рассмотреть взаимосвязи между всеми числовыми переменными, что особенно полезно на начальных этапах анализа данных.





# ИНТЕРАКТИВНОСТИ С ПОМОЩЬЮ БИБЛИОТЕКИ PLOTLY

- ✓ Plotly — это многофункциональная библиотека для создания графиков, поддерживающая высокоуровневую интерактивность. Она идеально подходит для тех случаев, когда вы хотите, чтобы ваши графики были более интуитивно понятными и доступными для изучения. Plotly используется в разнообразных сферах: от бизнеса до научных исследований, предоставляя мощные инструменты визуализации данных в веб-браузере.
- ✓ На практике это может быть полезно, когда вам нужно представить данные лидерам, которым важны детали, или когда вы хотите создать интерактивную аналитическую панель. Например, в маркетинге интерактивные графики помогают легче анализировать клиентскую базу, а в науке — исследовать экспериментальные данные.
- ✓ Главное отличие Plotly от классических библиотек вроде Matplotlib — интерактивность. Каждый график, созданный с её помощью, по умолчанию поддерживает взаимодействие с пользователем: масштабирование областей, отображение значений при наведении курсора, переключение между слоями данных. Для трёхмерных визуализаций доступно вращение и изменение угла обзора. Такие возможности особенно ценны при работе с большими наборами данных, где важно быстро «погрузиться» в конкретный сегмент.
- ✓ Библиотека не ограничивается стандартными типами диаграмм. В арсенале Plotly — географические карты с наложением тепловых карт, 3D-поверхности, диаграммы Санки для визуализации потоков, кастомные SVG-фигуры и даже интеграция с алгоритмами машинного обучения через библиотеку SciKit-Learn. Отдельного внимания заслуживает поддержка временных рядов: автоматическое распознавание форматов дат и интуитивное масштабирование оси времени.

# PLOTLY

- ✓ Как и большинство замечательных вещей в Python, Plotly можно установить через pip. Откройте командную строку или терминал и выполните следующую команду:

```
pip install plotly
```

После установки, чтобы начать рисовать, импортируем необходимые библиотеки:

```
import plotly.express as px
import plotly.graph_objects as go
```

- ✓ Сохранение результатов  
HTML (интерактивность сохраняется):

```
fig.write_html("temperature.html")
```

Изображения (PNG, JPEG, SVG):

```
fig.write_image("temperature.png")
```

В коде чаще всего используются два модуля:

- plotly.express (сокращённо — px) — для быстрого создания стандартных графиков;
- plotly.graph\_objects (сокращённо — go) — для тонкой настройки элементов.

# PLOTLY -СТРУКТУРА ГРАФИКА: ДАННЫЕ И ОФОРМЛЕНИЕ

- ✓ Каждая визуализация в Plotly строится на двух компонентах:
  - Data — информация для отображения (координаты точек, столбцов и т. д.);
  - Layout — параметры оформления (заголовок, подписи осей, легенда).

Например, чтобы изменить цвет линии и добавить сетку, используется метод `update_layout()`:

```
fig.update_layout(
 plot_bgcolor='white',
 xaxis=dict(showgrid=True, gridcolor='lightgray'),
 yaxis=dict(showgrid=True, gridcolor='lightgray'),
 title_font=dict(size=20)
)
```

## ✓ Работа с Figure

Объект Figure — основа всех манипуляций. Через него добавляются новые данные (например, несколько линий на один график) и декоративные элементы:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=months, y=temperatures, name='2023'))
fig.add_trace(go.Scatter(x=months, y=[-3, -1, 5, 12], name='2024'))
fig.update_layout(title='Сравнение температур')
```

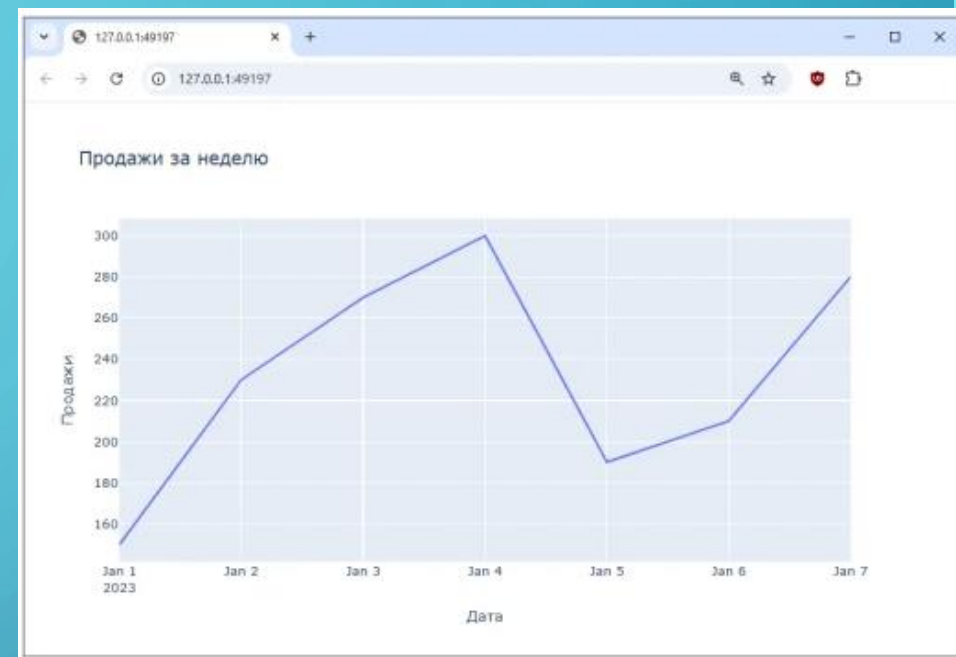
# PLOTLY

Простой пример — линейного графика. Этот код создаст линейный график, который можно будет изучать, приближать и перемещать. Благодаря интерактивности, вы сможете лучше сосредоточиться на интересующих вас областях данных:

```
import plotly.express as px
import pandas as pd

Пример данных
data = pd.DataFrame({
 "Дата": pd.date_range(start="2023-01-01", periods=7),
 "Продажи": [150, 230, 270, 300, 190, 210, 280]
})

Создание интерактивного графика
fig = px.line(data, x="Дата", y="Продажи", title="Продажи за неделю")
fig.show()
```

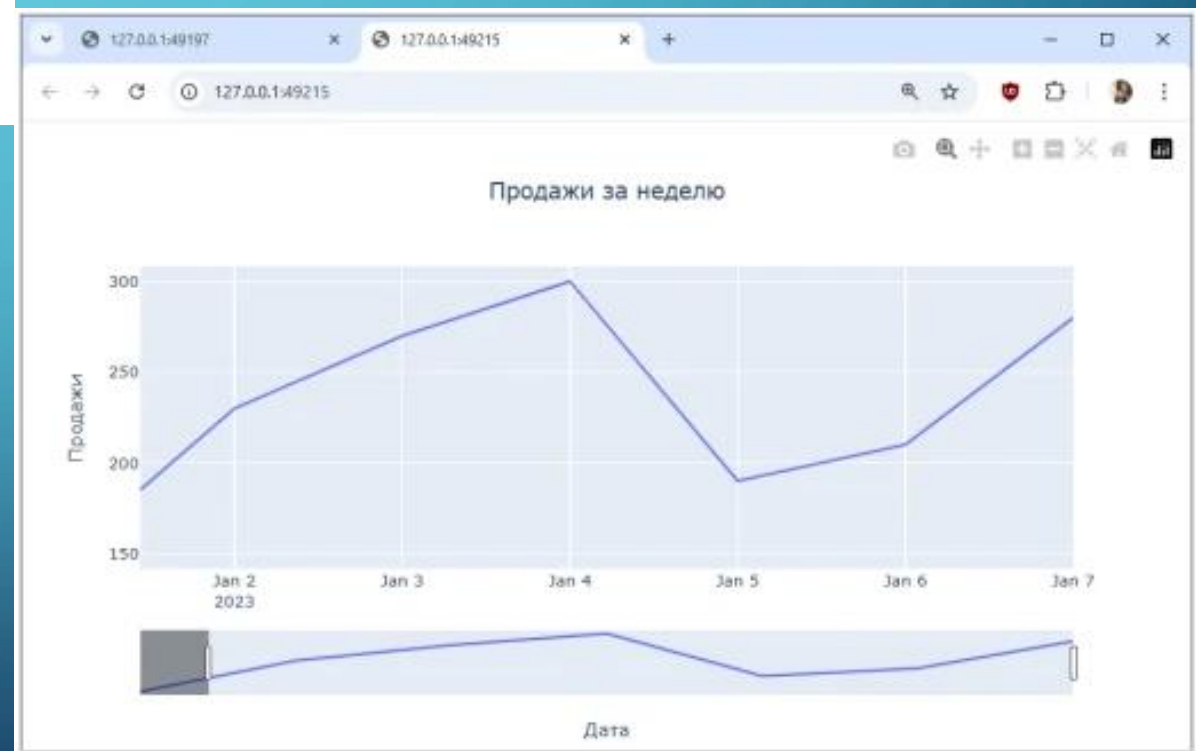


# PLOTLY

Добавим интерактивные элементы: включим возможность выделения данных, зума и панорамирования.

```
fig.update_layout(
 xaxis=dict(rangeslider=dict(visible=True)),
 title=dict(x=0.5) # Центрирование заголовка
)
fig.show()
```

Включили диапазонный ползунок (rangeslider) и выровняли заголовок по центру. Это делает ваш график более гибким и удобным для пользователей.

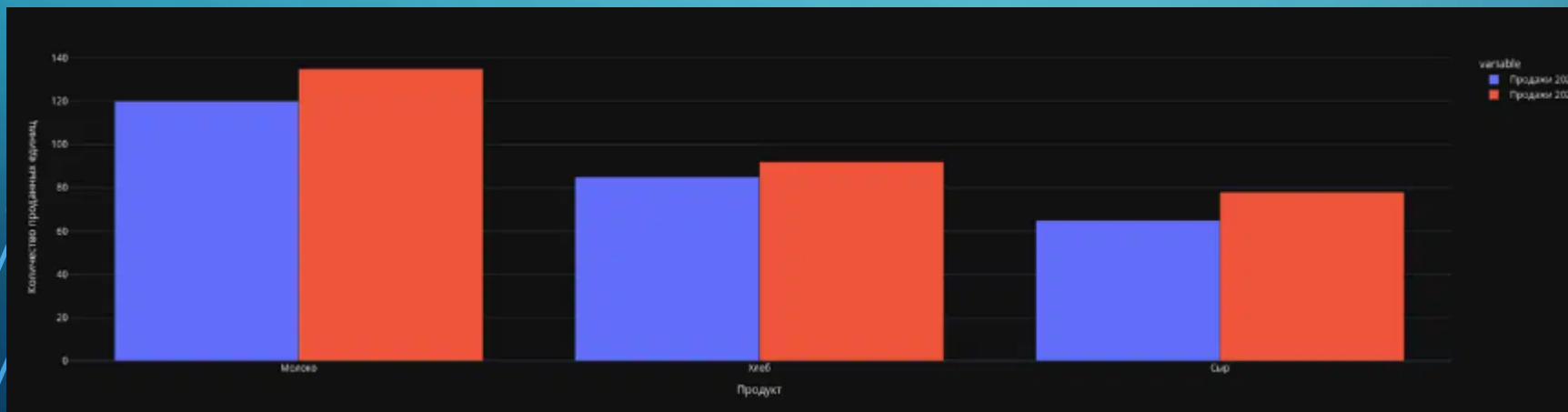




# PLOTLY - РАЗНЫЕ ТИПЫ ИНТЕРАКТИВНЫХ ГРАФИКОВ

Столбчатые диаграммы ( Bar() ) - используются для сравнения величин между категориями — например, для отображения рейтинга продуктов, результатов А/В-тестов или распределения бюджетов

```
data = {
 'Продукт': ['Молоко', 'Хлеб', 'Сыр'],
 'Продажи 2023': [120, 85, 65],
 'Продажи 2024': [135, 92, 78]
}
fig = px.bar(data, x='Продукт', y=['Продажи 2023', 'Продажи 2024'], barmode='group')
fig.update_layout(yaxis_title='Количество проданных единиц')
```

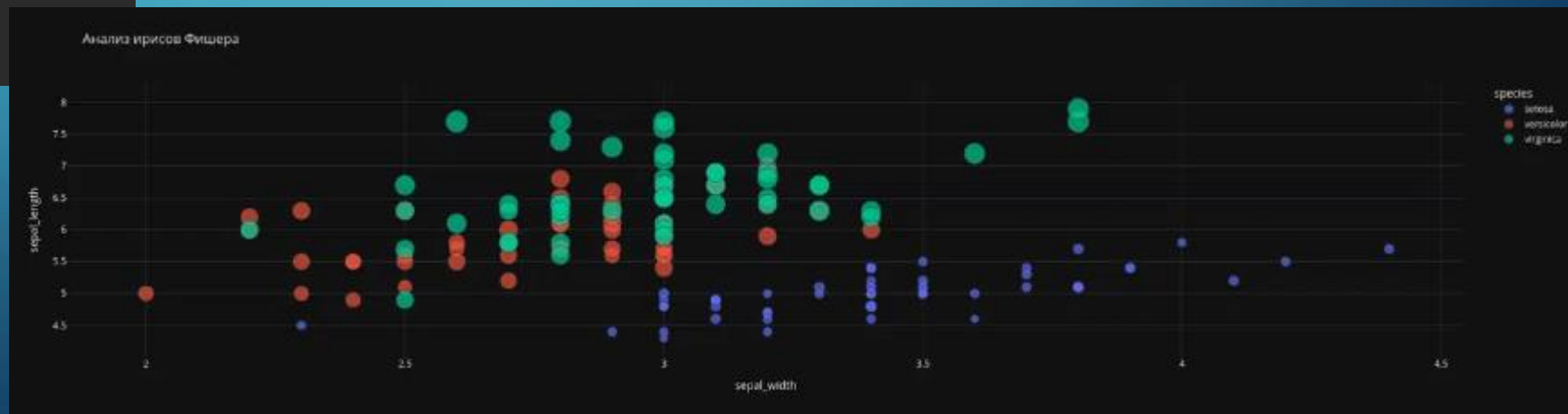


# PLOTLY - РАЗНЫЕ ТИПЫ ИНТЕРАКТИВНЫХ ГРАФИКОВ

Точечные диаграммы ( Scatter() )- используются для выявления корреляции между двумя переменными. Незаменимы при анализе взаимосвязей: рост расходов на рекламу и увеличение выручки, зависимость веса от роста и т. д.

```
df = px.data.iris()
fig = px.scatter(
 df
 x='sepal_width',
 y='sepal_length',
 color='species',
 size='petal_length',
 hover_data=['petal_width'],
 title='Анализ ирисов Фишера'
)
fig.show()
```

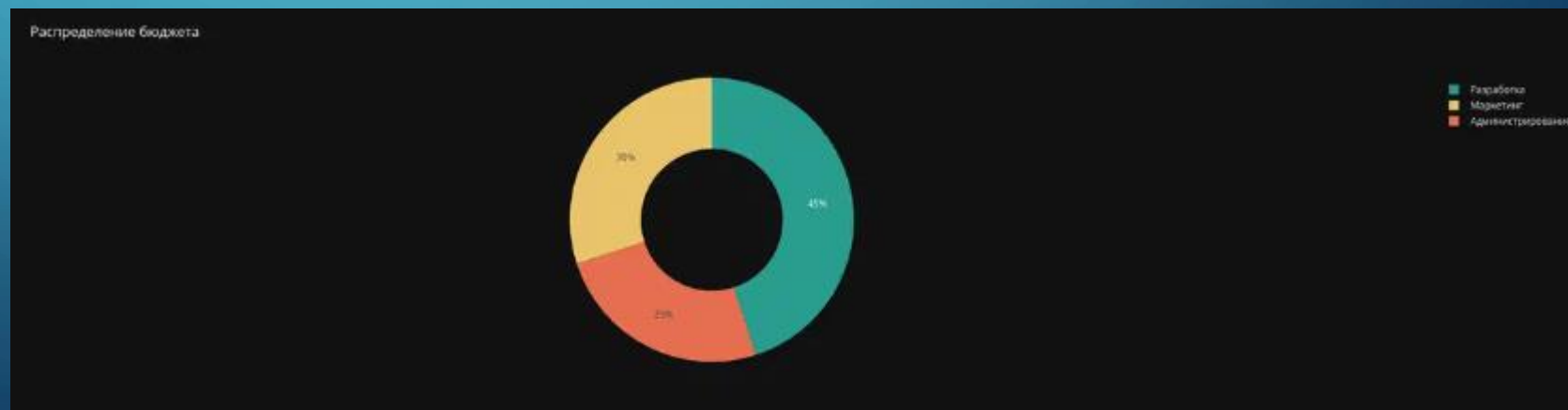
График распределения ирисов



# PLOTLY - РАЗНЫЕ ТИПЫ ИНТЕРАКТИВНЫХ ГРАФИКОВ

Круговые и кольцевые диаграммы ( Pie() ) - используются для демонстрации пропорций — например, доли рынка компаний, структуры расходов или распределения голосов на выборах.

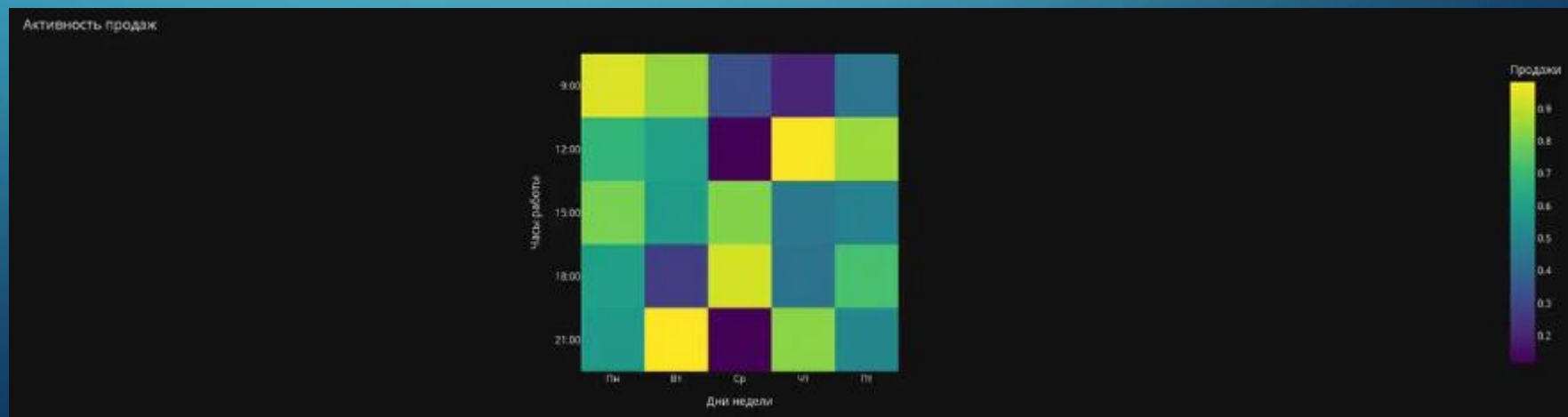
```
labels = ['Разработка', 'Маркетинг', 'Администрирование']
values = [45, 30, 25]
fig = go.Figure(go.Pie(
 labels=labels,
 values=values,
 hole=0.5, # Создаёт кольцо
 marker_colors=['#2A9D8F', '#E9C46A', '#E76F51']
))
fig.update_layout(title='Распределение бюджета')
```



# PLOTLY - РАЗНЫЕ ТИПЫ ИНТЕРАКТИВНЫХ ГРАФИКОВ

Тепловые карты (Heatmaps()) - используются для показа интенсивности значений через цветовые градиенты. Применяются для визуализации матриц корреляции, активности пользователей по времени суток или географического распределения показателей.

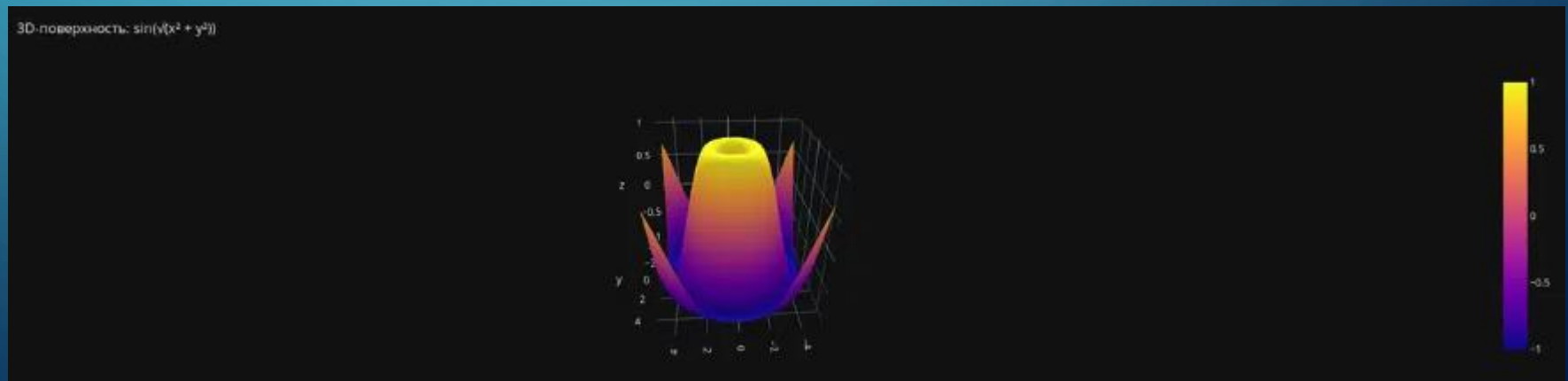
```
import numpy as np
matrix = np.random.rand(5, 5) # Случайные данные
fig = px.imshow(
 matrix,
 labels=dict(x="Дни недели", y="Часы работы", color="Продажи"),
 x=['Пн', 'Вт', 'Ср', 'Чт', 'Пт'],
 y=['9:00', '12:00', '15:00', '18:00', '21:00'],
 color_continuous_scale='Viridis'
)
fig.update_layout(title='Активность продаж')
```



# PLOTLY - РАЗНЫЕ ТИПЫ ИНТЕРАКТИВНЫХ ГРАФИКОВ

3D-графики (Surface()) - используются для показа интенсивности значений через цветовые градиенты. Применяются для визуализации матриц корреляции, активности пользователей по времени суток или географического распределения показателей.

```
import numpy as np
matrix = np.random.rand(5, 5) # Случайные данные
fig = px.imshow(
 matrix,
 labels=dict(x="Дни недели", y="Часы работы", color="Продажи"),
 x=['Пн', 'Вт', 'Ср', 'Чт', 'Пт'],
 y=['9:00', '12:00', '15:00', '18:00', '21:00'],
 color_continuous_scale='Viridis'
)
fig.update_layout(title='Активность продаж')
```





# PLOTLY - РАЗНЫЕ ТИПЫ ИНТЕРАКТИВНЫХ ГРАФИКОВ

Географические карты (scatter\_geo()). Plotly поддерживает несколько картографических проекций (*equiangular, orthographic, natural earth*) и позволяет

```
df_geo = px.data.gapminder().query("year == 2007")
fig = px.scatter_geo(
 df_geo,
 locations="iso_alpha",
 size="pop",
 color="continent",
 hover_name="country",
 projection="natural earth",
 title='Население стран (2007)'
)
```



# МАТPLOTLIB ИЛИ PLOTLY: ЧТО ВЫБРАТЬ?

## ✓ Интерактивность vs статика

- Plotly создан для интерактивных сценариев. Графики по умолчанию реагируют на действия пользователя — клики, прокрутку, наведение курсора. Это незаменимо при работе в Jupyter Notebook, создании дашбордов или презентаций, где важно «покрутить» данные.
- Matplotlib фокусируется на статичной визуализации. Её сильная сторона — подготовка графиков для печати или публикаций в научных журналах, где требуется точный контроль над каждым элементом (толщина линий, размеры шрифтов, DPI).

# МАТPLOTLIB ИЛИ PLOTLY: ЧТО ВЫБРАТЬ?

## ✓ Кастомизация: где больше свободы?

- Matplotlib выигрывает в глубине кастомизации. Библиотека позволяет менять буквально всё: от расстояния между штрихами на оси до формы маркеров в легенде. Plotly упрощает базовую настройку (цвета, заголовки, шрифты), но для нестандартных задач (например, совмещения карты и 3D-графика) может потребоваться работа с низкоуровневым API.

# МАТРОТЛІВ ІЛИ РЛОТЛІ: ЧТО ВЫБРАТЬ?

## ✓Производительность

- Matplotlib обрабатывает большие датасеты (миллионы точек) быстрее, особенно с использованием NumPy. Plotly требует больше ресурсов в силу интерактивности. Визуализация более 500 000 точек может «подтормаживать» в браузере.
- **Интеграция с другими инструментами**
- Plotly тесно связана с экосистемой веб-технологий. Графики легко встраиваются в Dash-приложения, React-проекты или WordPress-сайты через iframe.
- Matplotlib интегрируется с LaTeX для научных публикаций и лучше совместима с GUI-фреймворками вроде Tkinter или PyQt.

# МАТРОТЛІВ ІЛИ РЛОТЛІ: ЧТО ВЫБРАТЬ?

## ✓ Интеграция с другими инструментами

- Plotly тесно связана с экосистемой веб-технологий. Графики легко встраиваются в Dash-приложения, React-проекты или WordPress-сайты через `iframe`.
- Matplotlib интегрируется с LaTeX для научных публикаций и лучше совместима с GUI-фреймворками вроде Tkinter или PyQt.



# МАТPLOTLIB ИЛИ PLOTLY: ЧТО ВЫБРАТЬ?

✓ Когда что выбрать?

❖ Plotly подходит, если:

- нужны интерактивные отчёты или дашборды;
- данные требуют исследования «в движении» (зумирование, фильтрация);
- цель — поделиться визуализацией онлайн.

❖ Matplotlib предпочтительна, когда:

- требуется подготовить графики для печати (PDF, SVG);
- работа идёт с огромными массивами данных;
- нужен кастомный дизайн, выходящий за рамки стандартных шаблонов.

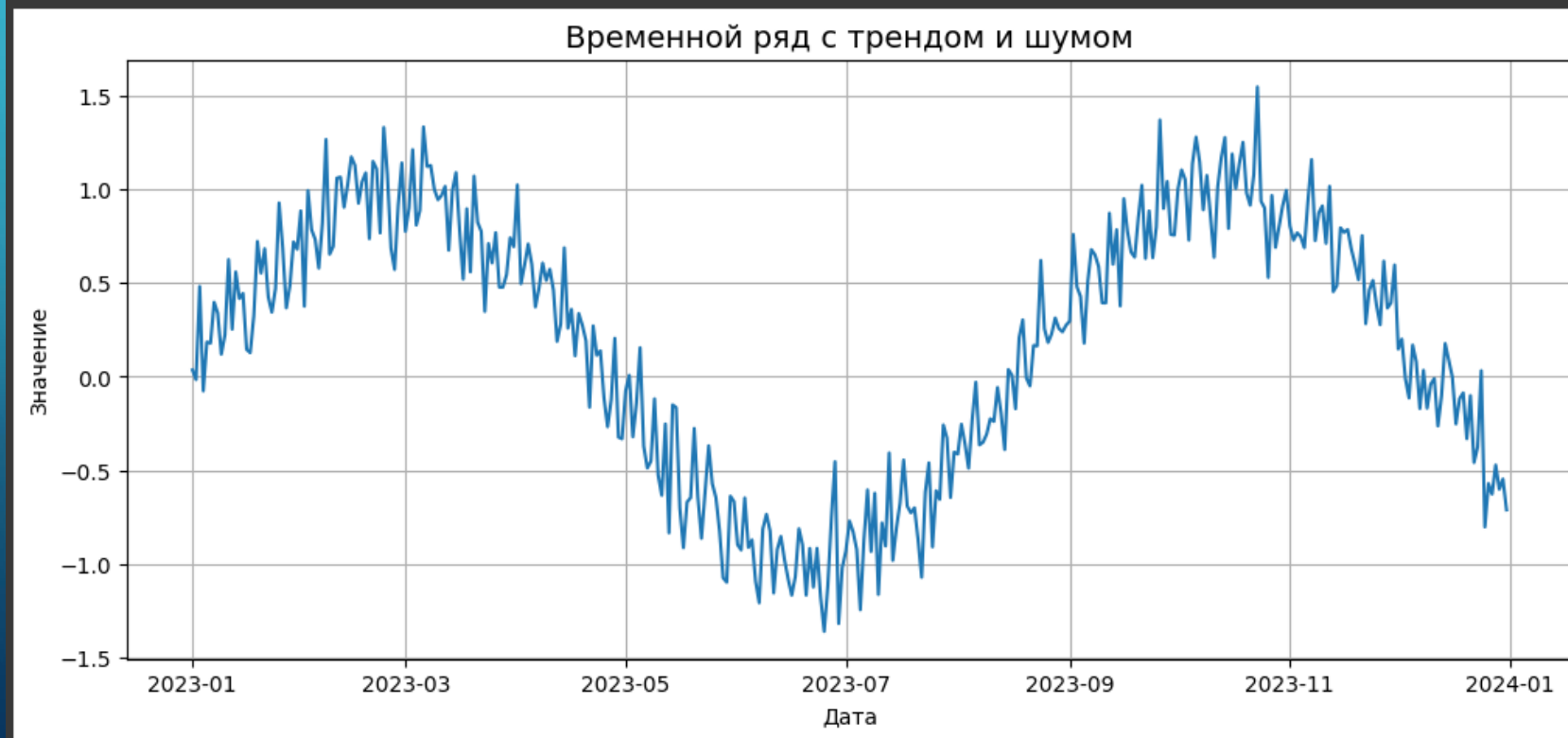
# МАТПЛОТЛІВ ИЛИ PLOTLY: ЧТО ВЫБРАТЬ?

- ✓ Можно ли совмещать? Да! Например:
  - использовать Matplotlib для предварительного анализа (быстрый просмотр распределения данных);
  - создавать финальные интерактивные графики в Plotly для презентации.
  - Главный тренд последних лет — смещение фокуса в сторону интерактивности. Однако это не отменяет важности Matplotlib: «под капотом» многих высокоуровневых библиотек (включая Plotly) до сих пор работают её механизмы.
- ✓ Идеальная стратегия — освоить оба инструмента, используя их сильные стороны. Для быстрого исследования данных и презентаций — Plotly, для тонкой настройки графиков и подготовки их к публикации — Matplotlib. Такой подход превращает визуализацию из рутины в искусство.

# ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

# ЛИНЕЙНЫЙ ГРАФИК ВРЕМЕННОГО РЯДА

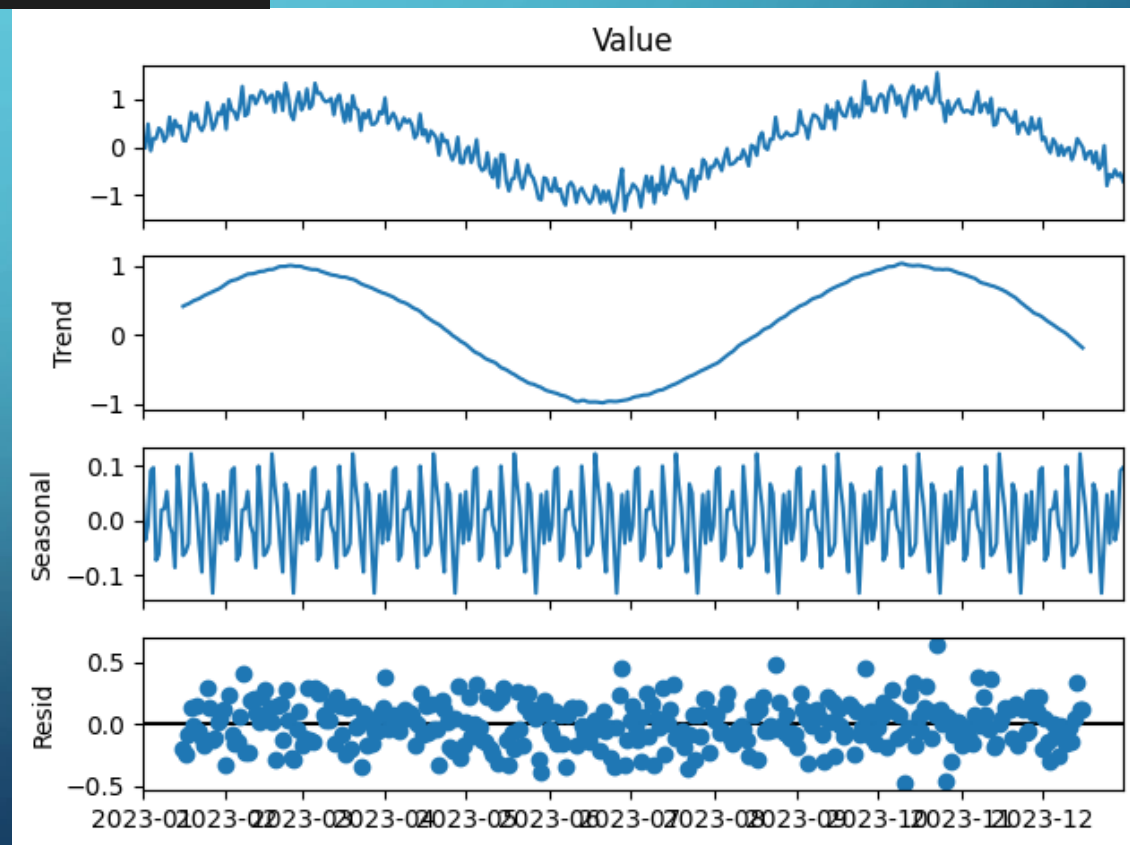
```
plt.figure(figsize=(12, 5))
sns.lineplot(data=data, x='Date', y='Value')
plt.title('Временной ряд с трендом и шумом', fontsize=14)
plt.xlabel('Дата')
plt.ylabel('Значение')
plt.grid(True)
plt.show()
```



# РАЗЛОЖЕНИЕ ВРЕМЕННОГО РЯДА

```
from statsmodels.tsa.seasonal import seasonal_decompose

Преобразование в формат временного ряда
data_ts = data.set_index('Date')
result = seasonal_decompose(data_ts['Value'], model='additive', period=30)
result.plot()
plt.show()
```





# ПОДГОТОВКА ДАННЫХ



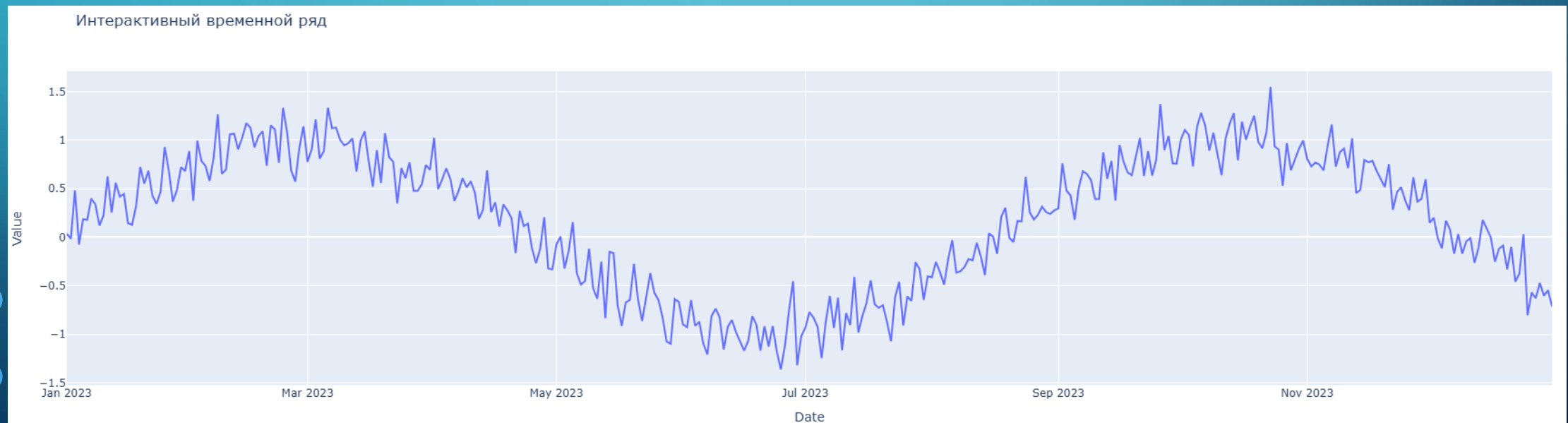
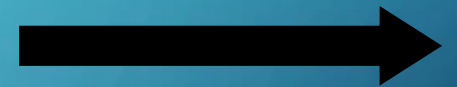
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

Создание временного ряда
date_range = pd.date_range(start='2023-01-01', periods=365, freq='D')
data = pd.DataFrame({
 'Date': date_range,
 'Value': np.sin(np.linspace(0, 10, 365)) + np.random.normal(0, 0.2, 365)
})
```

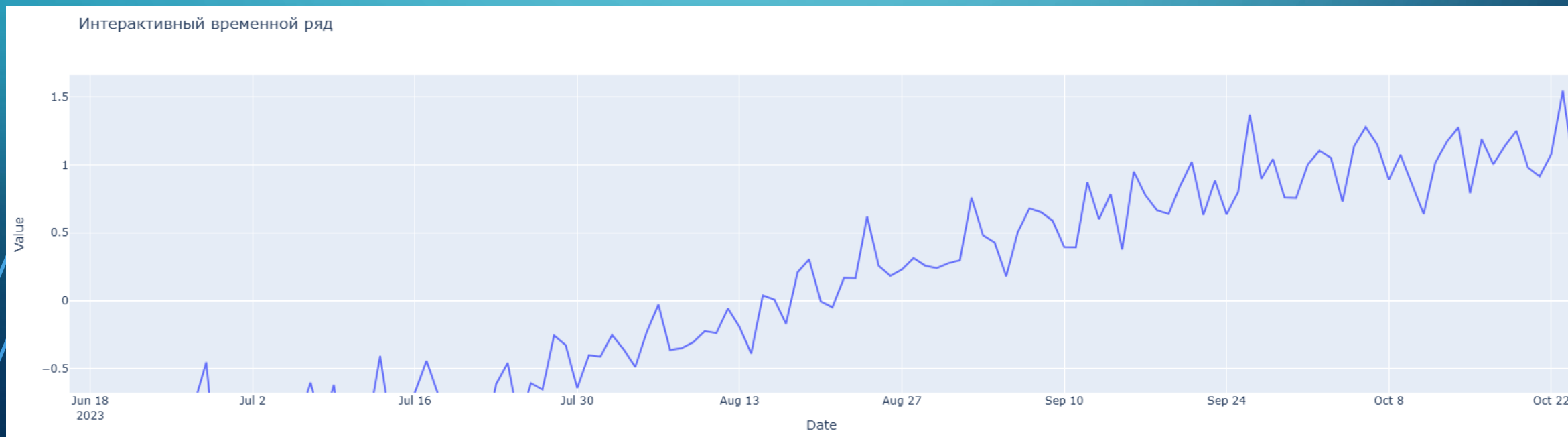
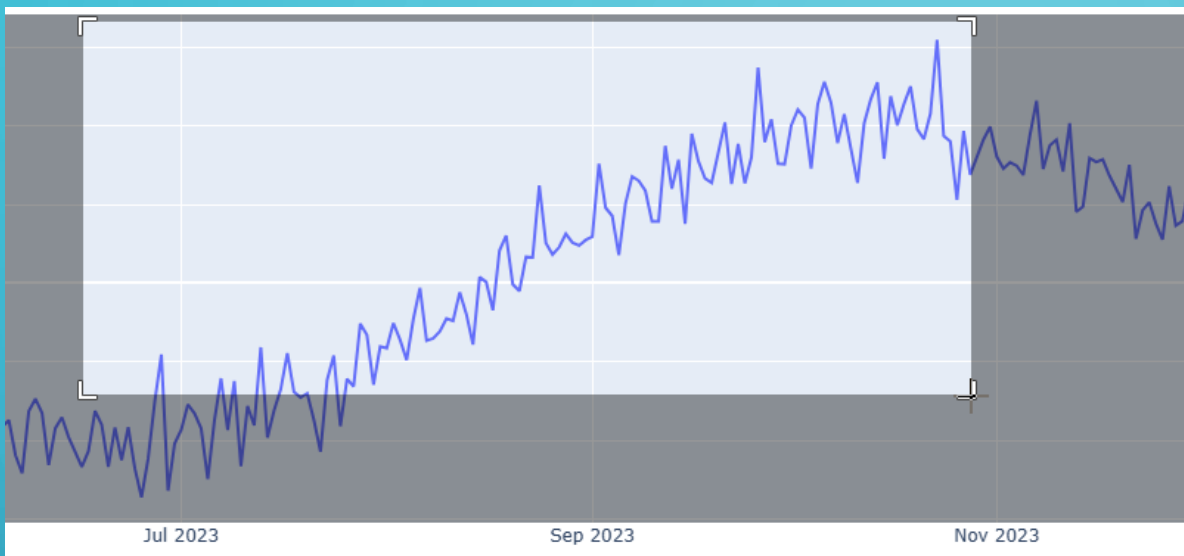
# ИНТЕРАКТИВНЫЕ ГРАФИКИ В PLOTLY

```
import plotly.express as px

Линейный график
fig = px.line(data, x='Date', y='Value',
 title='Интерактивный временной ряд')
fig.show()
```



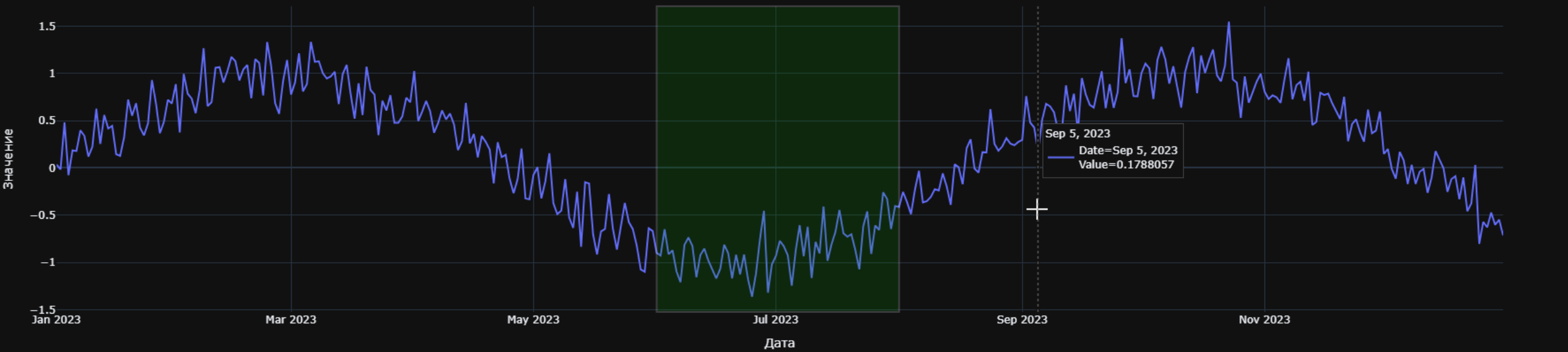
# ИНТЕРАКТИВНЫЕ ГРАФИКИ В PLOTLY



# ИНТЕРАКТИВНЫЙ КАСТОМИЗИРОВАННЫЙ ГРАФИК В PLOTLY

```
fig.update_layout(
 hovermode='x unified', # Отображение данных при наведении
 xaxis_title='Дата',
 yaxis_title='Значение',
 template='plotly_dark' # Тема оформления
)
fig.add_vrect(x0='2023-06-01', x1='2023-08-01', fillcolor='green', opacity=0.2) # Выделение периода
fig.show()
```

Интерактивный временной ряд

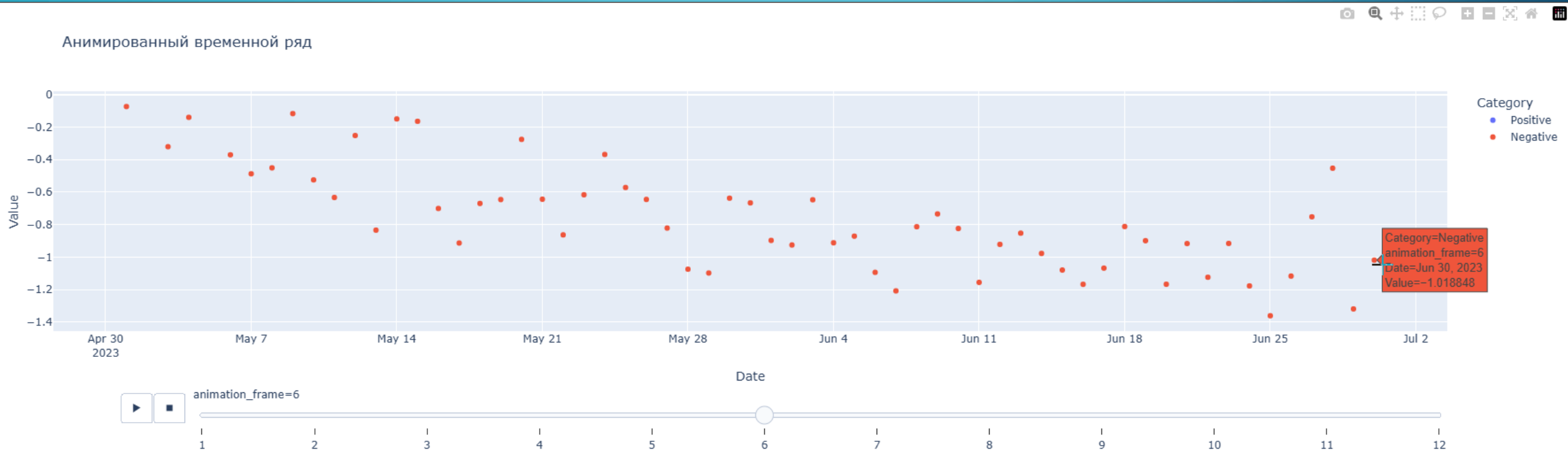


# ИНТЕРАКТИВНЫЙ АНИМИРОВАННЫЙ ГРАФИК В PLOTLY

```
Данные с категориями
data['Category'] = np.where(data['Value'] > 0, 'Positive', 'Negative')

fig = px.scatter(data, x='Date', y='Value', color='Category',
 animation_frame=data['Date'].dt.month, # Анимация по месяцам
 title='Анимированный временной ряд')

fig.show()
```





# ВИЗУАЛИЗАЦИЯ ГЕОГРАФИЧЕСКИХ ДАННЫХ В GEOPANDAS

```
import geopandas as gpd
import matplotlib.pyplot as plt

Загрузка актуальных данных
world = gpd.read_file("https://naciscdn.org/naturalearth/110m/cultural/ne_110m_admin_0_countries.zip")

Проверяем доступные столбцы
print("Доступные столбцы:", world.columns.tolist())

Используем правильные названия столбцов:
POP_EST - оценка населения
AREA_KM2 - площадь в км² (или используем geometry для расчета площади)
world['pop_density'] = world['POP_EST'] / (world['geometry'].area / 10**6) # Переводим площадь из кв.м в кв.км

Построение карты
fig, ax = plt.subplots(figsize=(15, 10))
world.plot(column='pop_density',
 cmap='OrRd',
 legend=True,
 ax=ax,
 legend_kwds={'label': "Плотность населения (чел/км²)"})
plt.title('Плотность населения стран мира', fontsize=16)
plt.axis('off')
plt.show()
```

# ВИЗУАЛИЗАЦИЯ ГЕОГРАФИЧЕСКИХ ДАННЫХ В GEORANDAS

