Seaborn

```
pip install seaborn
```

Візуалізація є важливим аспектом аналізу та інтерпретації даних, оскільки вона дозволяє легко розуміти складні набори даних. Вона допомагає виявляти патерни, зв'язки та тенденції, які можуть бути непомітними, просто аналізуючи сиру інформацію. Останніми роками Python став однією з найпопулярніших мов програмування для аналізу даних завдяки своєму великому набору бібліотек та фреймворків.

Бібліотеки візуалізації в Python дозволяють користувачам створювати зрозумілі та інтерактивні графіки, які ефективно комунікують інсайти широкій аудиторії. Деякі з популярних бібліотек та фреймворків для візуалізації в Python включають Matplotlib, Plotly, Bokeh та Seaborn. Кожна з цих бібліотек має свої унікальні функціональні можливості, які відповідають певним потребам.

У цьому посібнику ми сфокусуємося на Seaborn, популярній бібліотеці візуалізації даних в Python, яка надає простий у використанні інтерфейс для створення інформативних статистичних графіків.

Що таке Seaborn?

Побудований на основі Matplotlib, Seaborn - це відома бібліотека Python для візуалізації даних, яка пропонує користувачу зручний інтерфейс для створення привабливих та інформативних статистичних графіків. Вона розроблена для роботи з об'єктами даних Pandas, що дозволяє легко та ефективно візуалізувати та досліджувати дані.

Seaborn пропонує різноманітні потужні інструменти для візуалізації даних, включаючи точкові діаграми, лінійні графіки, стовпчикові діаграми, теплові карти та багато іншого. Вона також надає підтримку для високорівневого статистичного аналізу, такого як регресійний аналіз, графіки розподілу та категоріальні графіки.

Головна перевага Seaborn полягає в його здатності генерувати привабливі графіки з мінімальними зусиллями. Вона надає набір тем та палітр кольорів

за замовчуванням, які можна легко налаштувати під свої вподобання. Крім того, Seaborn пропонує набір вбудованих статистичних функцій, які дозволяють користувачам легко виконувати складний статистичний аналіз з використанням їх візуалізацій.

Ще одна помітна особливість Seaborn полягає в його здатності створювати складні багатошарові графіки. За допомогою Seaborn користувачі можуть створювати сітки графіків, які дозволяють легко порівнювати кілька змінних або підмножини даних. Це робить його ідеальним інструментом для дослідження даних та презентаційної роботи.

Seaborn - потужна та гнучка бібліотека візуалізації даних в Python, яка надає простий у використанні інтерфейс для створення інформативних та естетично привабливих статистичних графіків. Вона нада

є набір інструментів для візуалізації даних, включаючи високорівневий статистичний аналіз, та дозволяє легко створювати складні багатошарові графіки.

Seaborn проти Matplotlib

Дві найпопулярніші бібліотеки візуалізації даних у Python - це Matplotlib і Seaborn. Хоча обидві бібліотеки призначені для створення високоякісних графіків і візуалізацій, вони мають кілька ключових відмінностей, які роблять їх краще підходящими для різних випадків використання.

Одна з основних відмінностей між Matplotlib і Seaborn полягає в їх фокусі. Matplotlib - це низькорівнева бібліотека для побудови графіків, яка надає широкий спектр інструментів для створення висококастомізованих візуалізацій. Вона є дуже гнучкою бібліотекою, що дозволяє користувачам створювати практично будь-який тип графіку. Ця гнучкість вимагає поглибленого вивчення та більш розгорнутого коду.

Seaborn, з іншого боку, є високорівневим інтерфейсом для створення статистичних графіків. Вона побудована на основі Matplotlib і надає простий, зрозумілий інтерфейс для створення типових статистичних графіків. Seaborn призначений для роботи з об'єктами даних Pandas, що спрощує створення візуалізацій з мінімальним кодом. Вона також надає набір вбудованих статистичних функцій, що дозволяє користувачам легко виконувати складний статистичний аналіз з їх візуалізаціями.

Ще одна важлива відмінність між Matplotlib і Seaborn полягає в їх стилях за замовчуванням та палітрах кольорів. Matplotlib надає обмежений набір стилів та палітр кольорів за замовчуванням, що вимагає вручну налаштувати графіки для досягнення бажаного вигляду. Seaborn, з іншого боку, пропонує ряд стилів та палітр кольорів за замовчуванням, які оптимізовані для різних типів даних та візуалізацій. Це дозволяє користувачам легко створювати привабливі візуалізації з мінімальним налаштуванням.

Хоча обидві бібліотеки мають свої переваги та недоліки, Seaborn, як правило, краще підходить для створення статистичних графіків та дослідження даних, тоді як Matplotlib краще підходить для створення високоякісних графіків для презентацій та публікацій. Проте варто зауважити, що Seaborn побудована на основі Matplotlib, і обидві бібліотеки можуть використовуватись разом для створення складних та високоякісних візуалізацій, використовуючи переваги обох бібліотек.

Ви можете детальніше ознайомитись з Matplotlib у нашому посібнику «Вступ до створення графіків з Matplotlib у Python».

Matplotlib і Seaborn - це потужні бібліотеки візуалізації даних у Python з різними перевагами та обмеженнями. Розуміння відмінностей між цими двома бібліотеками може допомогти користувачам вибрати потрібний інструмент для конкретних вимог до візуалізації даних.

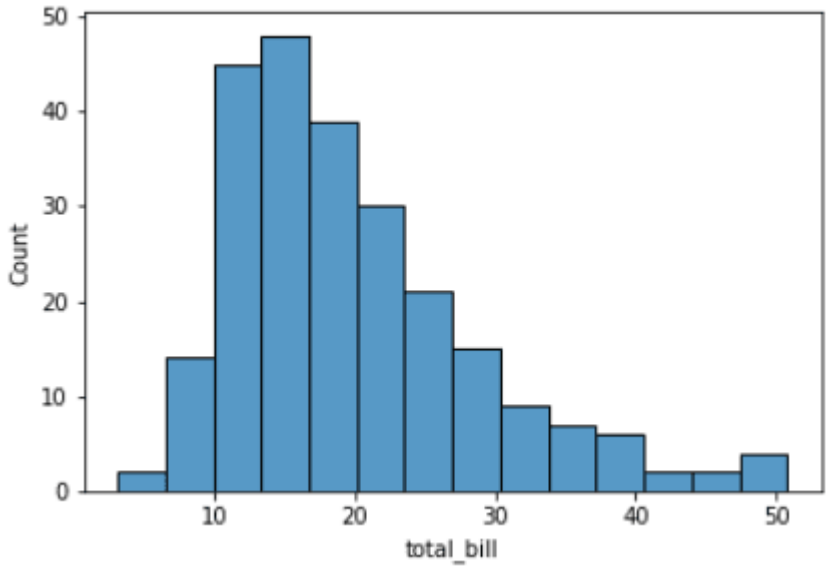
Набори даних

Seaborn надає кілька вбудованих наборів даних, які можна використовувати для візуалізації даних та статистичного аналізу. Ці набори даних зберігаються у форматі pandas dataframes, що полегшує їх використання з функціями побудови графіків Seaborn.

Один з найпоширеніших наборів даних, який використовується у всіх офіційних прикладах Seaborn, називається набором даних "tips"; він містить інформацію про чайові, що даються у ресторанах. Ось приклад завантаження та візуалізації набору даних "tips" у Seaborn:

```
import seaborn as sns

# Load the Tips dataset
tips = sns.load_dataset("tips")
```

```
# Create a histogram of the total bill amounts
sns.histplot(data=tips, x="total_bill")
```



Важливим є те, що візуалізація може не працювати на деяких системах. Тому для того щоб гарантувати виведення зображення треба використовувати **matplotlib**

Наразі, основна інформація полягає в тому, що Seaborn має багато вибіркових наборів даних у форматі pandas DataFrames, які легко використовувати для практики ваших навичок візуалізації. Ось ще один приклад завантаження набору даних `exercise`:

```
import seaborn as sns

# Load the exercise dataset
exercise = sns.load_dataset("exercise")

# check the head
exercise.head()
```

	Unnamed: 0	id	diet	pulse	time	kind
0	0	1	low fat	85	1 min	rest
1	1	1	low fat	85	15 min	rest
2	2	1	low fat	88	30 min	rest
3	3	2	low fat	90	1 min	rest
4	4	2	low fat	92	15 min	rest

Типи графіків у Seaborn

Seaborn надає широкий спектр типів графіків, які можна використовувати для візуалізації даних та дослідження даних. Загалом, будь-яка візуалізація може належати до однієї з трьох категорій.

Одновимірна - лише x (містить лише одну ось інформації)
Двовимірна - x та y (містить дві осі інформації)
Тривимірна - x, y, z (містить три осі інформації)

Ось деякі з найбільш поширених типів графіків у Seaborn:

- Графік розсіювання (Scatter Plot). Графік розсіювання використовується для візуалізації залежності між двома змінними. Функція scatterplot() у Seaborn надає простий спосіб створення графіків розсіювання.
- Графік ліній (Line Plot). Графік ліній використовується для візуалізації тенденції зміни змінної з плином часу. Функція lineplot() у Seaborn надає простий спосіб створення графіків ліній.
- Гістограма (Histogram). Гістограма використовується для візуалізації розподілу змінної. Функція histplot() у Seaborn надає простий спосіб створення гістограм.
- Діаграма "ящик з вусами" (Box Plot). Діаграма "ящик з вусами" використовується для візуалізації розподілу змінної. Функція boxplot() у Seaborn надає простий спосіб створення діаграм "ящик з вусами".
- Діаграма "скрипка" (Violin Plot). Діаграма "скрипка" схожа на діаграму "ящик з вусами", але надає більш детальний вигляд розподілу даних.

Функція violinplot() у Seaborn надає простий спосіб створення діаграм "скрипка".

- Теплова карта (Heatmap). Теплова карта використовується для візуалізації кореляції між різними змінними. Функція heatmap() у Seaborn надає простий спосіб створення теплових карт.
- Графік пар (Pairplot). Графік пар використовується для візуалізації залежності між кількома змінними. Функція pairplot() у Seaborn надає простий спосіб створення графіків пар.

Тепер ми розглянемо приклади та детальні пояснення для кожного з цих типів графіків у наступному розділі цього посібника.

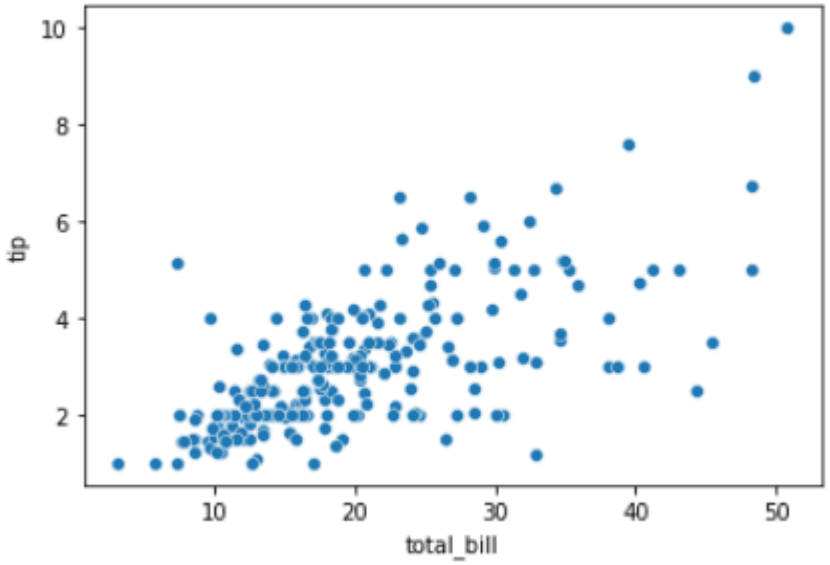
Приклади використання Seaborn

Графіки розсіювання у Seaborn

Графіки розсіювання використовуються для візуалізації залежності між двома неперервними змінними. Кожна точка на графіку представляє окремий значення даних, а положення точки на вісі x та y відображає значення двох змінних.

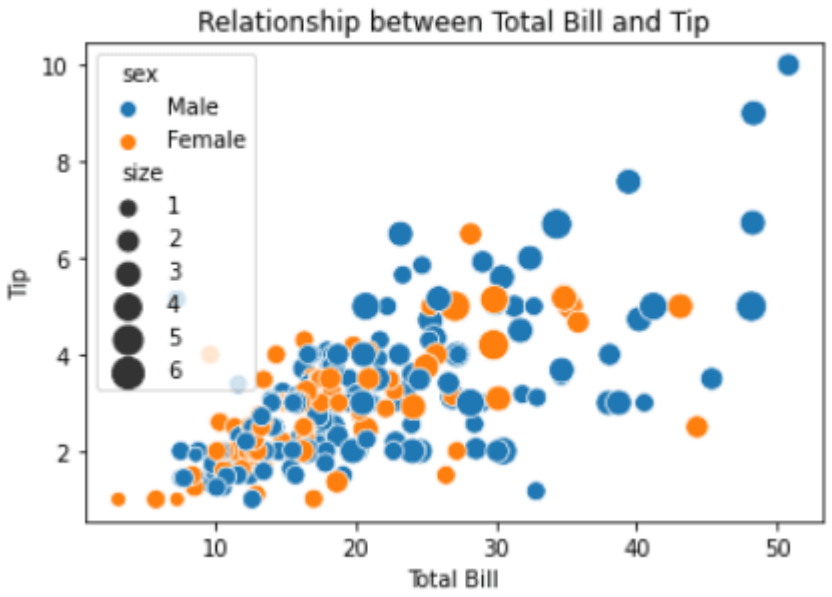
Графік можна налаштувати з використанням різних кольорів та маркерів, щоб виділити різні групи точок даних. У Seaborn графіки розсіювання можна створити за допомогою функції scatterplot().

```
import seaborn as sns

tips = sns.load_dataset("tips")

sns.scatterplot(x="total_bill", y="tip", data=tips)
```



А можна ще налаштувати розміри та кольори

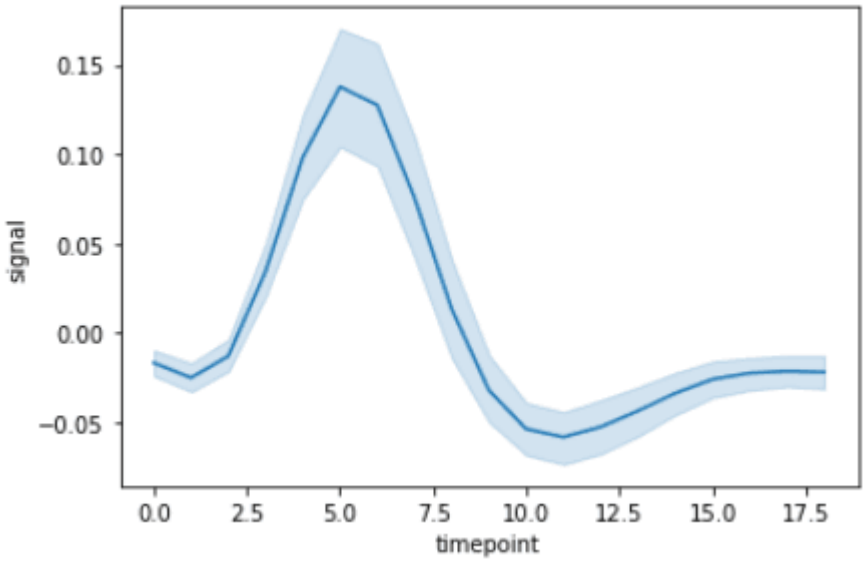
```
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset("tips")

# customize the scatter plot
sns.scatterplot(x="total_bill", y="tip", hue="sex", size="s
ize", sizes=(50, 200), data=tips)

# add labels and title
plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.title("Relationship between Total Bill and Tip")

# display the plot
plt.show()
```



Seaborn графіки ліній

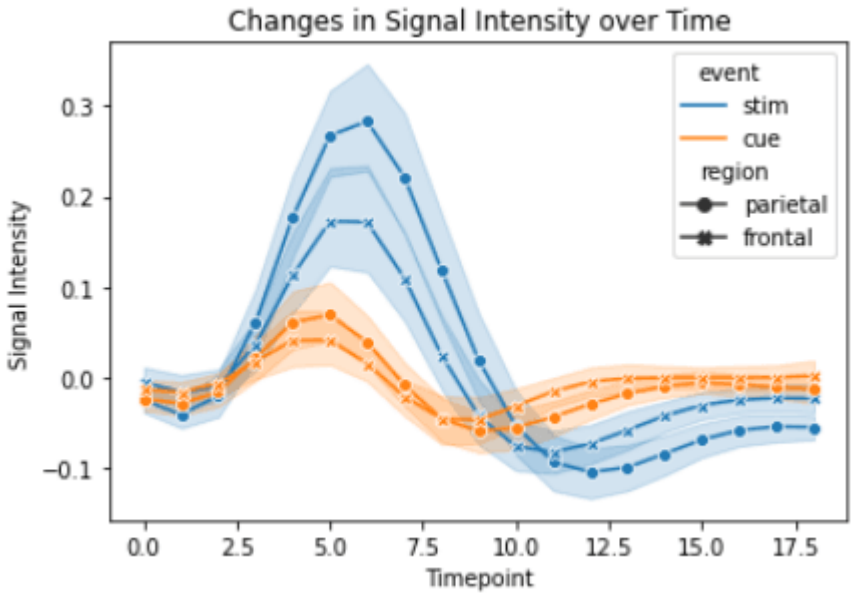
Графіки ліній використовуються для візуалізації тенденцій у даних з плином часу або інших неперервних змінних. У графіку ліній кожна точка даних з'єднана лінією, утворюючи плавну криву. У Seaborn графіки ліній можна створити за допомогою функції lineplot().

```
import seaborn as sns

fmri = sns.load_dataset("fmri")

sns.lineplot(x="timepoint", y="signal", data=fmri)
```



Ми можемо легко налаштувати це, використовуючи стовпці "event" та "region" з набору даних.

```
import seaborn as sns

import matplotlib.pyplot as plt

fmri = sns.load_dataset("fmri")

# customize the line plot
sns.lineplot(x="timepoint", y="signal", hue="event", style
="region", markers=True, dashes=False, data=fmri)

# add labels and title
plt.xlabel("Timepoint")
plt.ylabel("Signal Intensity")
plt.title("Changes in Signal Intensity over Time")

# display the plot
plt.show()
```

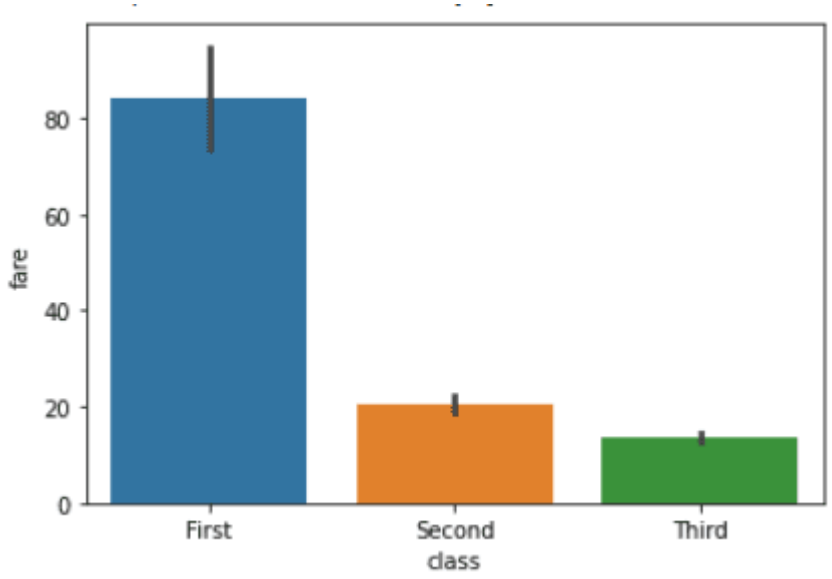


Знову ж таки, ми використали бібліотеку `seaborn` для створення простого графіку ліній та використали бібліотеку `matplotlib` для налаштування та поліпшення цього графіку ліній. Ви можете детальніше ознайомитися з графіками ліній у Seaborn у нашому окремому посібнику.

Графіки стовпчиків у Seaborn

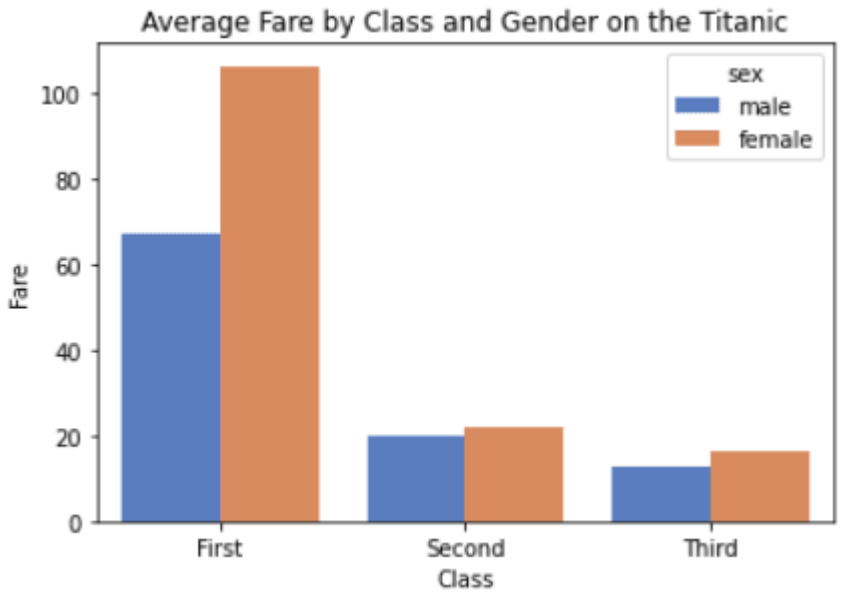
Графіки стовпчиків використовуються для візуалізації залежності між категоріальною змінною та неперервною змінною. У графіку стовпчиків кожен стовпець представляє середнє значення або медіану (або будь-яке інше агрегування) неперервної змінної для кожної категорії. У Seaborn графіки стовпчиків можна створити за допомогою функції barplot().

```
import seaborn as sns

titanic = sns.load_dataset("titanic")

sns.barplot(x="class", y="fare", data=titanic)
```



Давайте налаштуємо цей графік, включивши стовпець "sex" з набору даних.

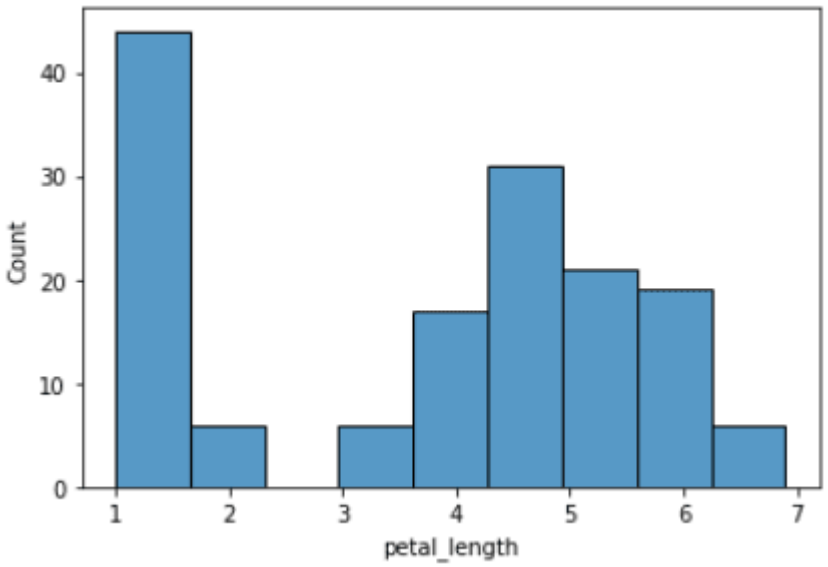
```
import seaborn as sns
import matplotlib.pyplot as plt

titanic = sns.load_dataset("titanic")

# customize the bar plot
sns.barplot(x="class", y="fare", hue="sex", ci=None, palett
e="muted", data=titanic)

# add labels and title
plt.xlabel("Class")
plt.ylabel("Fare")
plt.title("Average Fare by Class and Gender on the Titani
c")


# display the plot
plt.show()
```



Seaborn гістограми

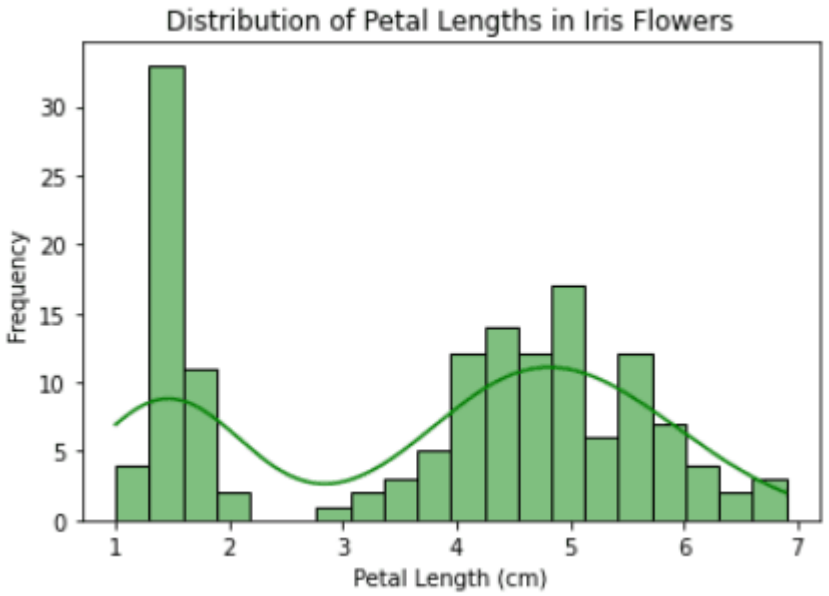
Гістограми візуалізують розподіл неперервної змінної. У гістограмі дані розбиваються на категорії (біни), а висота кожного біна представляє частоту або кількість точок даних, що потрапляють у цей бін. У Seaborn гістограми можна створити за допомогою функції histplot().

```
import seaborn as sns

iris = sns.load_dataset("iris")

sns.histplot(x="petal_length", data=iris)
```



Customizing a histogram

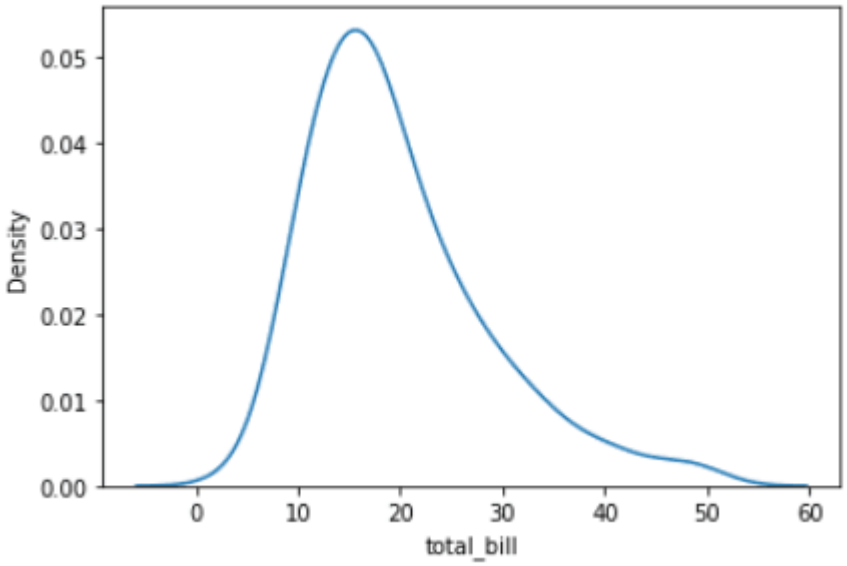
```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")

# customize the histogram
sns.histplot(data=iris, x="petal_length", bins=20, kde=Tru
e, color="green")

# add labels and title
plt.xlabel("Petal Length (cm)")
plt.ylabel("Frequency")
plt.title("Distribution of Petal Lengths in Iris Flowers")

# display the plot
plt.show()
```



Seaborn графіки щільності

Графіки щільності, також відомі як графіки щільності ядра, є типом візуалізації дистрибуції неперервної змінної. Вони схожі на гістограми, але замість того, щоб представляти дані у вигляді стовпців, графіки щільності використовують гладку криву для оцінки щільності даних. У Seaborn графіки щільності можна створити за допомогою функції kdeplot().

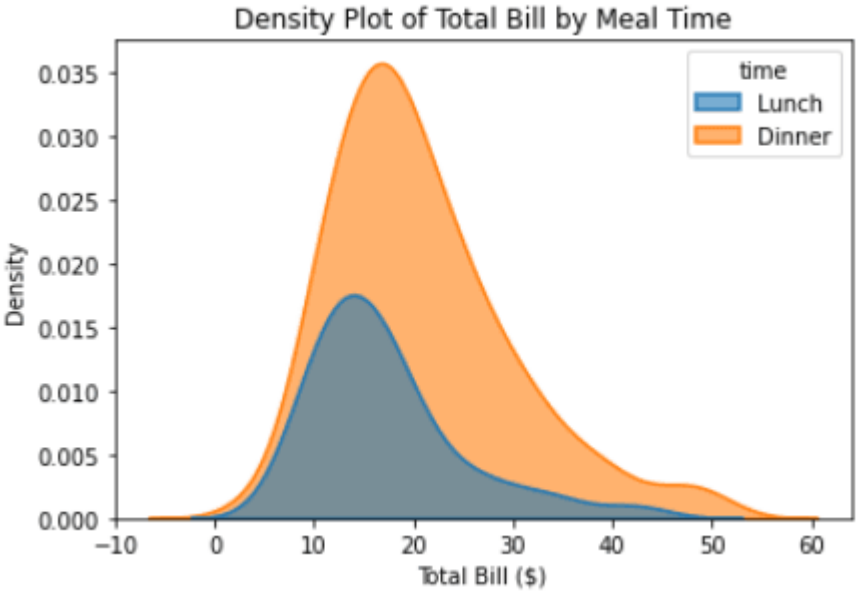
```
import seaborn as sns

tips = sns.load_dataset("tips")

sns.kdeplot(data=tips, x="total_bill")
```



Зробимо краще

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

# Load the "tips" dataset from Seaborn
tips = sns.load_dataset("tips")

# Create a density plot of the "total_bill" column from the
"tips" dataset
# We use the "hue" parameter to differentiate between "lunc
h" and "dinner" meal #times

# We use the "fill" parameter to fill the area under the cu
rve
# We adjust the "alpha" and "linewidth" parameters to make
the plot more visuall#y appealing

sns.kdeplot(data=tips, x="total_bill", hue="time", fill=Tru
e, alpha=0.6, linewidth=1.5)

# Add a title and labels to the plot using Matplotlib
plt.title("Density Plot of Total Bill by Meal Time")
plt.xlabel("Total Bill ($)")
plt.ylabel("Density")
```

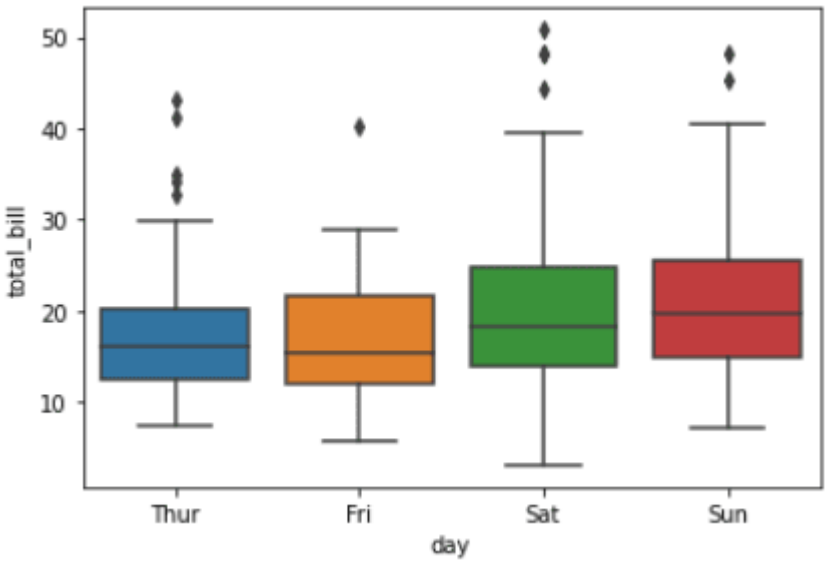
```
# Show the plot
plt.show()
```



Seaborn графіки "ящик з вусами"

Графіки "ящик з вусами" є типом візуалізації, який показує розподіл даних. Вони часто використовуються для порівняння розподілу однієї або декількох змінних в різних категоріях.

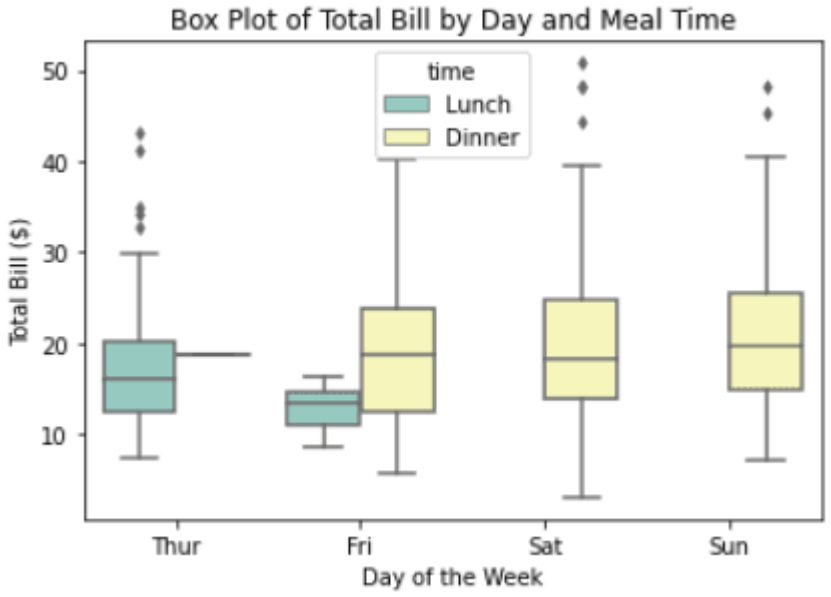
```
import seaborn as sns

tips = sns.load_dataset("tips")

sns.boxplot(x="day", y="total_bill", data=tips)
```



Налаштуйте графік "ящик з вусами", включивши стовпець "time" з набору даних.

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

# load the tips dataset from Seaborn
tips = sns.load_dataset("tips")

# create a box plot of total bill by day and meal time, usi
ng the "hue" paramete#r to differentiate between lunch and
dinner
# customize the color scheme using the "palette" parameter

# adjust the linewidth and fliersize parameters to make the
plot more visually a#ppealing

sns.boxplot(x="day", y="total_bill", hue="time", data=tips,
palette="Set3", linewidth=1.5, fliersize=4)

# add a title, xlabel, and ylabel to the plot using Matplot
lib functions
plt.title("Box Plot of Total Bill by Day and Meal Time")
plt.xlabel("Day of the Week")
plt.ylabel("Total Bill ($)")
```

```
# display the plot
plt.show()
```



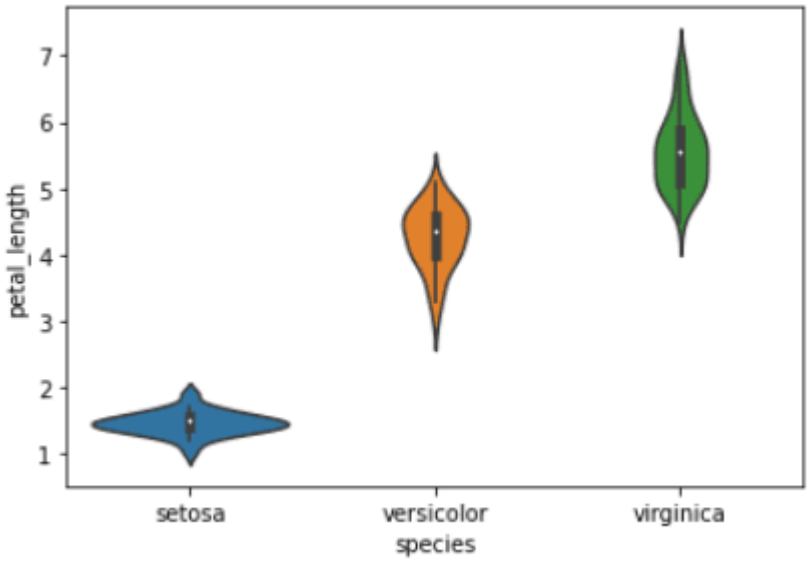
Seaborn графіки "скрипкових діаграм"

Графік "скрипкових діаграм" є типом візуалізації даних, який поєднує аспекти як графіків "ящик з вусами", так і графіків щільності. Він відображає оцінку щільності даних, зазвичай згладжену за допомогою ядерного оцінювача щільності, разом з міжквартильним розмахом (IQR) та медіаною у формі, схожій на графік "ящик з вусами".

Ширина скрипки представляє оцінку щільності, де ширші частини вказують на вищу щільність, а IQR та медіана показані як біла крапка та лінія всередині скрипки.

```
import seaborn as sns

# load the iris dataset from Seaborn
iris = sns.load_dataset("iris")


# create a violin plot of petal length by species
sns.violinplot(x="species", y="petal_length", data=iris)
```

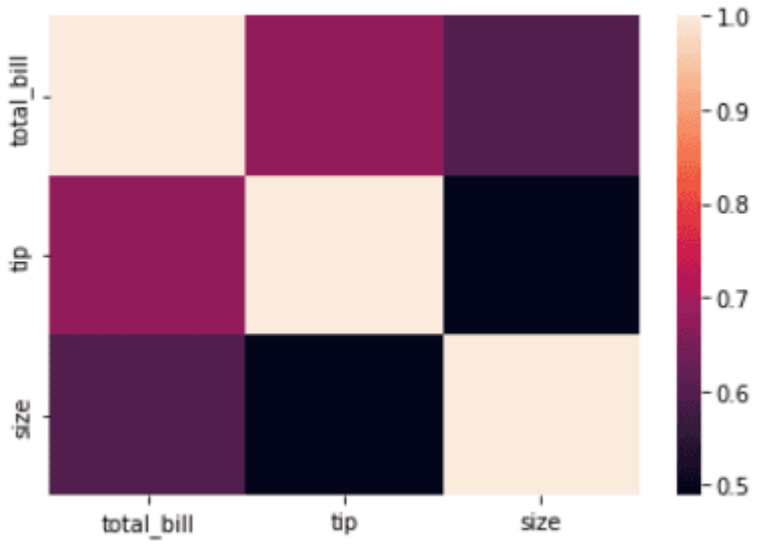
```
# display the plot
plt.show()
```



Seaborn графіки "теплових карт"

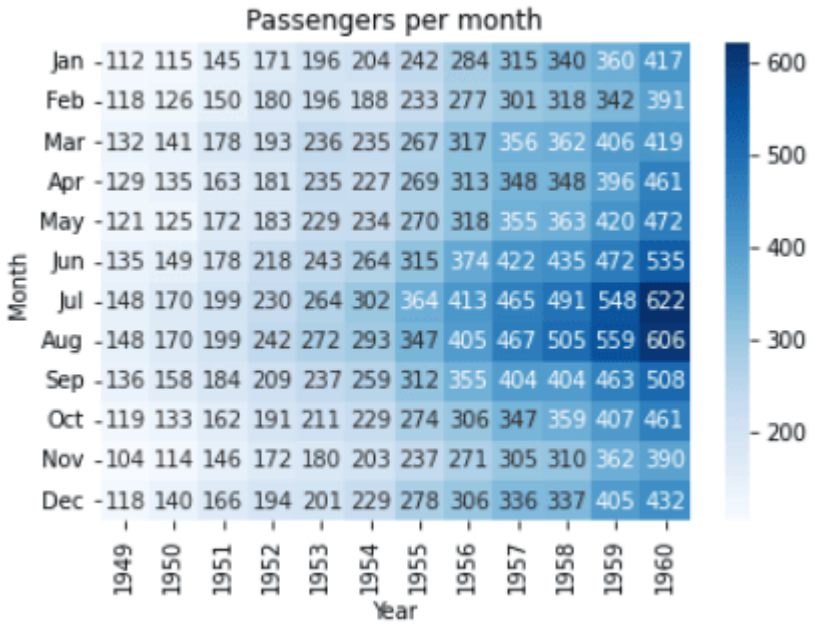
Теплова карта є графічним зображенням даних, яке використовує кольори для зображення значення змінної у двовимірному просторі. Теплові карти часто використовуються для візуалізації кореляції між різними змінними в наборі даних.

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

# Load the dataset
tips = sns.load_dataset('tips')

# Create a heatmap of the correlation between variables
corr = tips.corr()
sns.heatmap(corr)

# Show the plot
plt.show()
```



Ще один приклад теплової карти за допомогою набору даних `flights`.

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

# Load the dataset
flights = sns.load_dataset('flights')


# Pivot the data
flights = flights.pivot('month', 'year', 'passengers')

# Create a heatmap
sns.heatmap(flights, cmap='Blues', annot=True, fmt='d')

# Set the title and axis labels
plt.title('Passengers per month')
plt.xlabel('Year')
plt.ylabel('Month')

# Show the plot
plt.show()
```



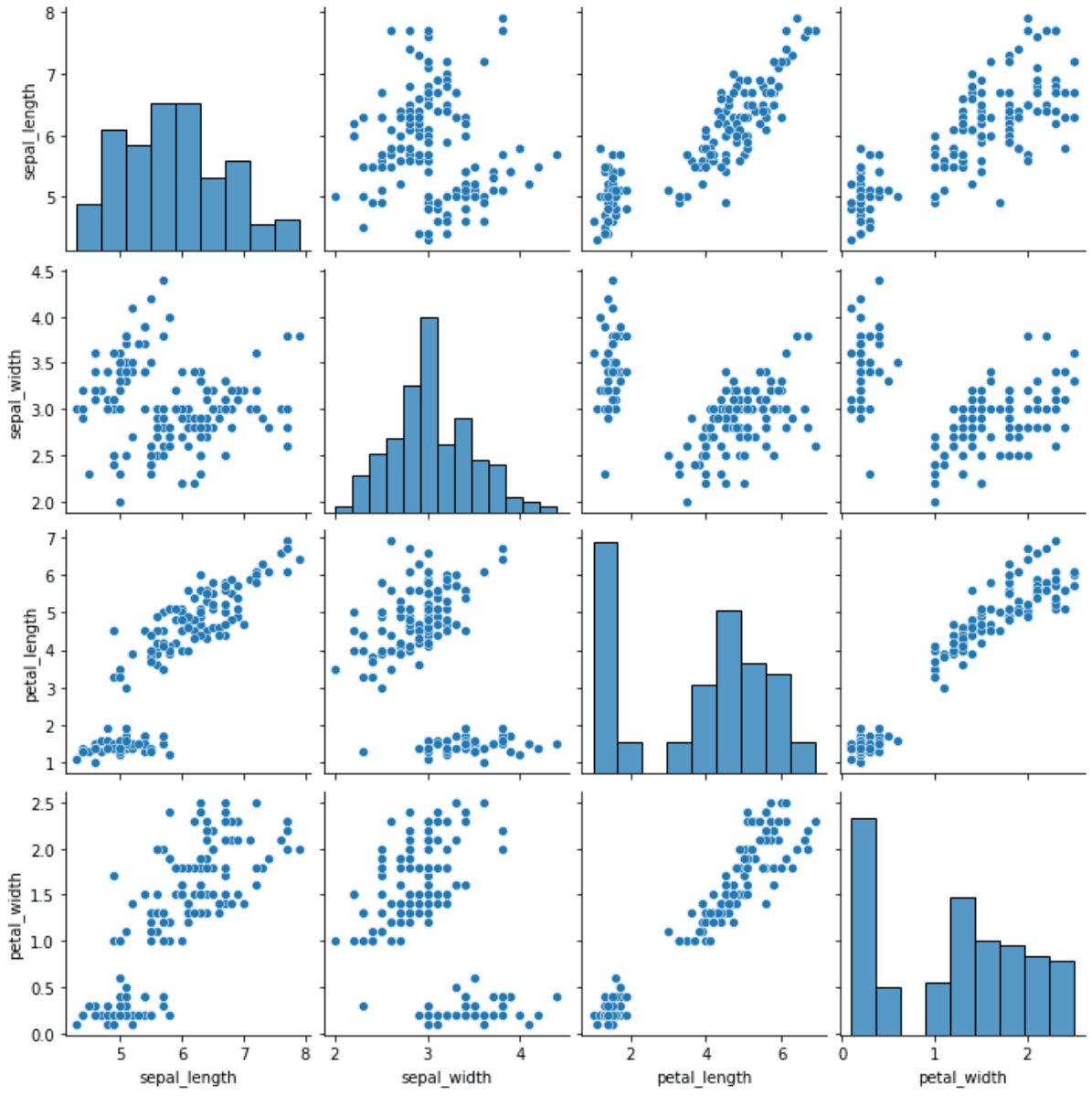
У цьому прикладі ми використовуємо набір даних `flights` з бібліотеки `seaborn`. Ми перетворюємо дані за допомогою методу .pivot(), щоб підготувати їх для представлення у вигляді теплової карти. Потім ми створюємо теплову карту за допомогою функції sns.heatmap() і передаємо перетворений набір даних `flights` як аргумент.

Seaborn графіки парами

Графіки парами є типом візуалізації, в якому кілька попарних графіків розсіювання відображаються у форматі матриці. Кожен графік розсіювання показує взаємозв'язок між двома змінними, тоді як графіки на діагоналі показують розподіл окремих змінних.

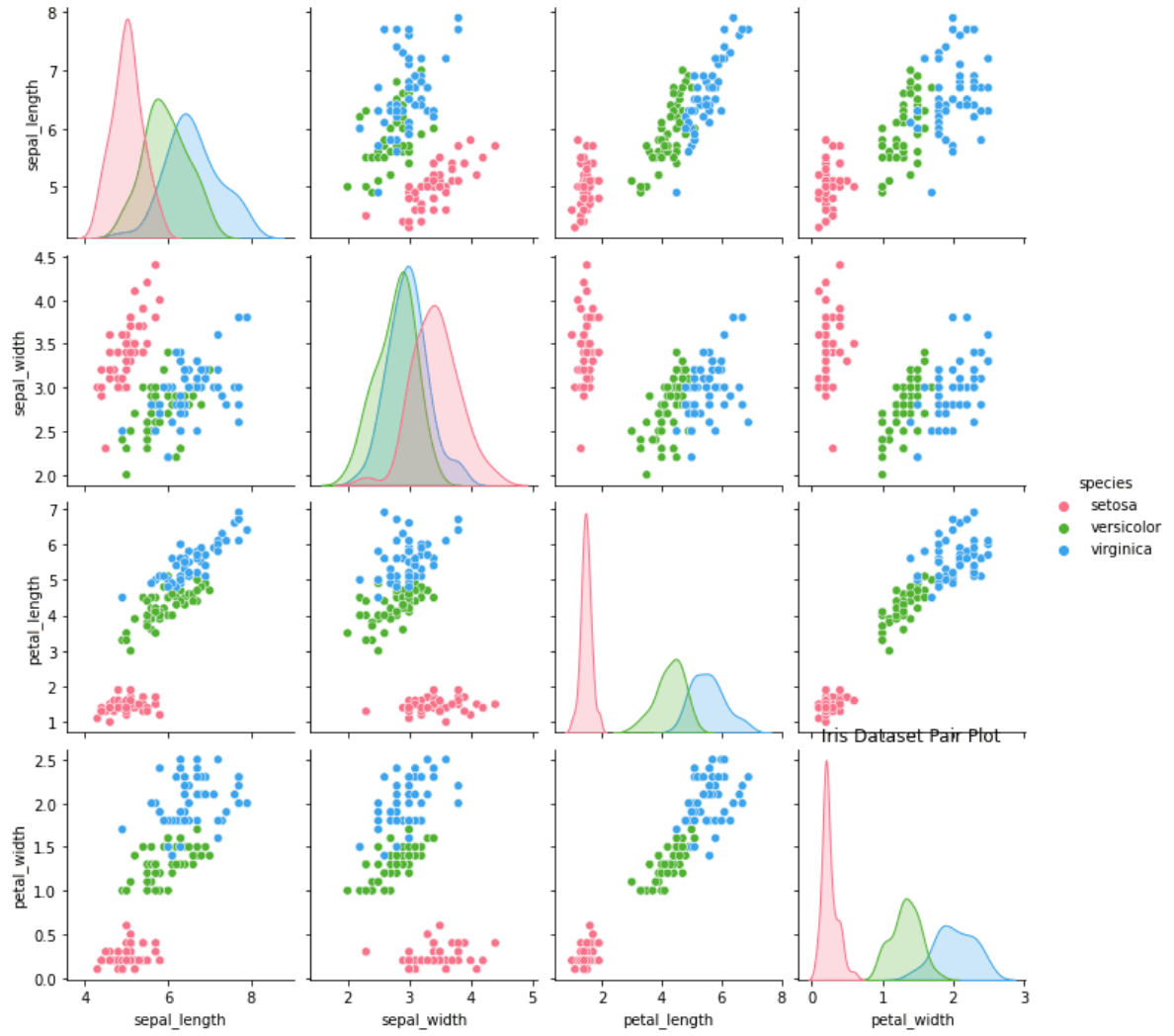
```
import seaborn as sns


# Load iris dataset
iris = sns.load_dataset("iris")

# Create pair plot
sns.pairplot(data=iris)
```

```
# Show plot
plt.show()
```




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

# Load iris dataset
iris = sns.load_dataset("iris")
```

```
# Create pair plot with custom settings
sns.pairplot(data=iris, hue="species", diag_kind="kde", pal
ette="husl")

# Set title
plt.title("Iris Dataset Pair Plot")

# Show plot
plt.show()
```



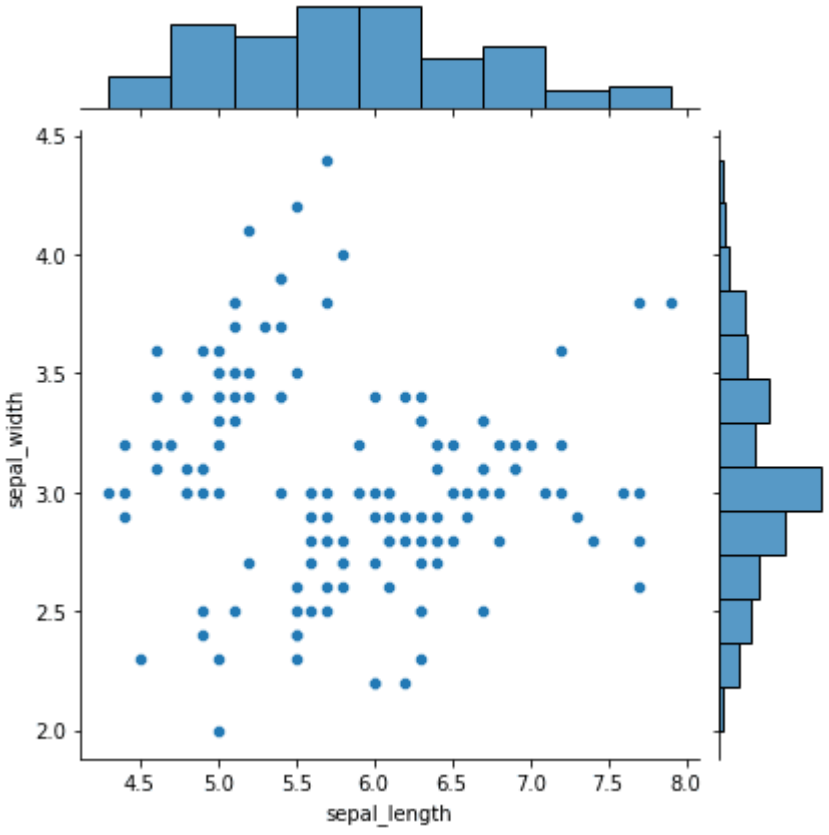
Seaborn графіки спільних розподілів

Графік спільних розподілів - це потужний метод візуалізації у бібліотеці seaborn, який поєднує два різні графіки в одному: графік розсіювання та

гістограму. Графік розсіювання показує зв'язок між двома змінними, тоді як гістограма відображає розподіл кожної окремої змінної. Це дозволяє провести більш комплексний аналіз даних, оскільки воно відображає кореляцію між двома змінними та їхні індивідуальні розподіли.

Ось простий приклад побудови графіка спільних розподілів за допомогою набору даних iris:

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

# load iris dataset
iris = sns.load_dataset("iris")

# plot a joint plot of sepal length and sepal width
sns.jointplot(x="sepal_length", y="sepal_width", data=iris)

# display the plot
plt.show()
```



Seaborn сітки з фасетами

FacetGrid є потужним інструментом в бібліотеці seaborn, який дозволяє візуалізувати розподіл однієї змінної, а також взаємозв'язок між двома змінними на різних рівнях додаткових категоріальних змінних.

FacetGrid створює сітку підграфіків на основі унікальних значень вказаної категоріальної змінної.

```
import seaborn as sns


# load the tips dataset
tips = sns.load_dataset('tips')

# create a FacetGrid for day vs total_bill
g = sns.FacetGrid(tips, col="day")

# plot histogram for total_bill in each day
g.map(sns.histplot, "total_bill")
```

Наводимо красоту

Колір

Нижче наведений приклад того, як ви можете змінити палітру кольорів у вашому графіку з використанням seaborn:

```
import seaborn as sns

# Change the color palette
sns.set_palette("Set2")

# Create a plot
sns.scatterplot(data=df, x="x", y="y", hue="category")
```

```
# Show the plot
plt.show()
```

У цьому прикладі використовується `sns.set_palette("Set2")`, щоб змінити палітру кольорів на "Set2". Ви також можете використовувати інші вбудовані палітри або власні налаштування кольорів, передавши список кольорів у `sns.set_palette()`.

Seaborn має вбудовані палітри кольорів, які можна використовувати для налаштування зовнішнього вигляду графіків. Ось кілька прикладів вбудованих палітр:

1. `"deep"`: глибокі темні відтінки
2. `"muted"`: приглушені відтінки
3. `"bright"`: яскраві відтінки
4. `"pastel"`: пастельні відтінки
5. `"dark"`: темні відтінки
6. `"colorblind"`: палітра для людей з вадами зору

Ви можете використовувати ці палітри, передаючи їх імена у функцію `sns.set_palette()` або в інших відповідних функціях seaborn. Крім того, ви можете створити власні палітри, передаючи список кольорів у вигляді кодів або назв колірних мап у функцію `sns.set_palette()`.

Розмір

Щоб змінити розмір фігури на графіках у seaborn, ви можете використовувати функцію `plt.figure(figsize=(ширина, висота))` з модулю `matplotlib.pyplot`. Ось приклад:

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

# Встановлюємо розмір фігури
plt.figure(figsize=(8, 6))

# Створюємо графік у seaborn
sns.scatterplot(x='x', y='y', data=data)
```

```
# Відображаємо графік
plt.show()
```

У цьому прикладі `figsize=(8, 6)` встановлює ширину 8 дюймів і висоту 6 дюймів для графіка. Змінюйте ці значення залежно від бажаного розміру фігури.

Підписи

Додавання анотацій може допомогти зробити ваші візуалізації більш зрозумілими. Ось приклад, як їх додати:

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

# Створюємо графік у seaborn
sns.scatterplot(x='x', y='y', data=data)

# Додаємо анотацію до точки
plt.annotate('Точка 1', (x1, y1), textcoords="offset point
s", xytext=(5,5), ha='center')

# Додаємо анотацію до іншої точки
plt.annotate('Точка 2', (x2, y2), textcoords="offset point
s", xytext=(5,-10), ha='center')

# Відображаємо графік
plt.show()
```

У цьому прикладі `plt.annotate()` використовується для додавання анотацій до відповідних точок на графіку. Ви можете змінювати текст анотації, координати точки (`(x, y)`), положення тексту відносно точки (`xytext`), а також визначати інші параметри за необхідності.

Найкращі практики для візуалізації з Seaborn

Вибирайте відповідний тип графіку для ваших даних

Seaborn надає широкий спектр типів графіків, кожен з яких призначений для різних типів даних та аналізу. Важливо вибрати відповідний тип графіку для ваших даних, щоб ефективно передати ваші висновки. Наприклад, для візуалізації зв'язку між двома неперервними змінними може бути більш підходящим графік розсіювання, тоді як для візуалізації категоріальних даних може бути більш підходящим графік стовпчиків.

Використовуйте кольори ефективно

Колір може бути потужним інструментом для візуалізації даних, але важливо використовувати його ефективно. Уникайте використання занадто багатьох кольорів або яскравих кольорів, оскільки це може ускладнити читання візуалізації. Замість цього використовуйте кольори, щоб виділити важливу інформацію або групувати подібні точки даних.

Позначайте осі та використовуйте зрозумілі заголовки

Маркування та заголовки є важливими для ефективної візуалізації даних. Переконайтеся, що ваші осі позначені чітко і надайте описовий заголовок для вашої візуалізації. Це допоможе вашій аудиторії зрозуміти повідомлення, яке ви намагаєтеся передати.

Враховуйте аудиторію

При створенні візуалізацій важливо враховувати аудиторію та повідомлення, яке ви намагаєтеся передати. Якщо ваша аудиторія не володіє технічними знаннями, використовуйте зрозумілу та лаконічну мову, уникайте технічного жаргону та надайте зрозумілі пояснення будь-яких статистичних концепцій.

Використовуйте відповідний статистичний аналіз

Seaborn надає ряд статистичних функцій, які можна використовувати для аналізу даних. При виборі статистичної функції переконайтеся, що вона найбільш підходить для ваших даних та дослідницького питання.

Налаштовуйте ваші візуалізації

У Seaborn ви знайдете великий набір параметрів, які можна налаштувати для покращення ваших візуалізацій. Експериментуйте з різними шрифтами, стилями та кольорами, щоб знайти той, який найкраще передає ваше повідомлення.

Порівняння Seaborn з іншими бібліотеками візуалізації

Seaborn проти Matplotlib

Seaborn побудована на основі Matplotlib і надає більш високорівневий інтерфейс для створення статистичних графіків. В той час як Matplotlib - загального призначення бібліотека для побудови графіків, Seaborn спеціально призначена для статистичної візуалізації даних.

Seaborn має декілька переваг порівняно з Matplotlib, включаючи простий синтаксис для створення складних графіків, вбудовану підтримку статистичних візуалізацій та астетично привабливі настройки за замовчуванням, які можна легко налаштувати.

Крім того, Seaborn пропонує декілька спеціалізованих типів графіків, які недоступні в Matplotlib, таких як графіки скрипок та графіки зграї.

Seaborn проти Pandas

Pandas - потужна бібліотека для маніпулювання даними в Python, яка пропонує ряд функціональності для роботи з структурованими даними. Хоча Pandas пропонує базові можливості побудови графіків через метод DataFrame.plot(), Seaborn надає більш розширену функціональність візуалізації, яка спеціально призначена для статистичних даних.

Функції Seaborn оптимізовані для роботи з структурами даних Pandas, що дозволяє легко створювати широкий спектр інформативних візуалізацій безпосередньо з об'єктів даних Pandas.

Seaborn також пропонує спеціалізовані типи графіків, такі як сітки фасетів та парні графіки, які недоступні в Pandas.

Seaborn проти Plotly

Plotly - це веб-орієнтована бібліотека візуалізації даних, яка пропонує інтерактивні та спільну візуалізацію даних.

Хоча Seaborn в першу чергу спрямована на створення статичних візуалізацій, Plotly пропонує більш інтерактивні та динамічні візуалізації, які можуть бути використані в веб-додатках або поширені в Інтернеті. Plotly також пропонує кілька спеціалізованих типів графіків, які недоступні в Seaborn, таких як графіки контурів та 3D-графіки поверхонь.

Однак Seaborn має простіший синтаксис та легшу настройку для створення статичних візуалізацій, що робить її кращим вибором для певних типів проектів.

Тренажер №1

Завдання: Аналіз даних пасажирів Титаніка

Датасет: Титанік

Опис завдання:
Використовуючи датасет Титанік, виконайте аналіз розміщення пасажирів на палубах з подальшою візуалізацією результатів. Зосередьтеся на статевому та віковому розподілі пасажирів на різних палубах.

Кроки виконання:

1. Завантажте датасет Титанік, якщо він ще не завантажений.
2. Використовуючи Seaborn або будь-яку іншу бібліотеку для візуалізації, побудуйте стовпчасту діаграму, що показує розподіл кількості пасажирів за палубами.
3. Розділіть пасажирів на палубах за статтю (чоловіки та жінки) та побудуйте стовпчасту діаграму для кожної статі, що показує розподіл пасажирів за палубами.
4. Розділіть пасажирів на палубах за віком (наприклад, діти, дорослі) та побудуйте стовпчасту діаграму для кожного вікового діапазону, що показує розподіл пасажирів за палубами.
5. Додайте підписи осей та назви графіків, щоб зробити їх більш інформативними та зрозумілими.

Це завдання дозволить вам проаналізувати розподіл пасажирів Титаніка за палубами з урахуванням їх статі та віку. Він допоможе вам зрозуміти, як ці фактори вплинули на розташування пасажирів на кораблі.

Якщо вам потрібні конкретні кодові приклади або додаткові підказки для розв'язання цього завдання, будь ласка, повідомте мене.

Рішення

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

# Завантаження набору даних "titanic"
titanic_df = sns.load_dataset("titanic")
```

```
# Побудова стовпчастої діаграми розподілу пасажирів за палубами
sns.countplot(x='deck', data=titanic_df)
plt.title('Distribution of Passengers by Deck')
plt.xlabel('Deck')
plt.ylabel('Count')
plt.show()

# Побудова стовпчастих діаграм розподілу пасажирів за палубами з поділом на стать
sns.countplot(x='deck', hue='sex', data=titanic_df)
plt.title('Distribution of Passengers by Deck and Sex')
plt.xlabel('Deck')
plt.ylabel('Count')
plt.legend(title='Sex')
plt.show()

# Побудова стовпчастих діаграм розподілу пасажирів за палубами з поділом за віком
sns.countplot(x='deck', hue='age', data=titanic_df)
plt.title('Distribution of Passengers by Deck and Age')
plt.xlabel('Deck')
plt.ylabel('Count')
plt.legend(title='Age')
plt.show()
```

В цьому прикладі ми використовуємо функцію `load_dataset()` з бібліотеки Seaborn для завантаження набору даних "titanic". Потім ми використовуємо функцію `countplot()` для побудови стовпчастих діаграм з розподілом пасажирів за палубами, розділеними за статтю (`sex`) та віком (`age`).

Зверніть увагу, що в даному прикладі використовується стовпчик `'deck'` для розподілу за палубами. В оригінальному датасеті Титанік, палуба позначена стовпчиком `'deck'`. Якщо ви працюєте з іншим датасетом Титанік, переконайтеся, що ви використовуєте відповідний стовпчик для побудови графіків.

Тренажер №2

Припустимо, у нас є набір даних, що містить інформацію про вартість житлових квартир у різних районах міста. Завдання полягає в аналізі розподілу цін на житло в різних районах і відображенні цієї інформації за допомогою графіку ящик з вусами.

Ось приклад вирішення цього завдання:

Створити набір даних, що містить інформацію про вартість житлових квартир у різних районах міста.

```
import pandas as pd

# Створення набору даних
data = {
    'Район': ['Район А', 'Район Б', 'Район В', 'Район А',
'Район Б', 'Район В'],
    'Ціна': [100000, 120000, 90000, 110000, 95000, 115000]
}

df = pd.DataFrame(data)
```

Використати графік ящик з вусами для відображення розподілу цін на житло в різних районах.

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

# Побудова графіку ящик з вусами
sns.boxplot(x='Район', y='Ціна', data=df)
plt.title('Розподіл цін на житло за районами')
plt.xlabel('Район')
plt.ylabel('Ціна')
plt.show()
```

У цьому прикладі ми спочатку створюємо набір даних з допомогою словника `data` і конвертуємо його в об'єкт DataFrame. Далі ми використовуємо функцію `boxplot()` з бібліотеки Seaborn для побудови графіку ящик з вусами. У змінній `x` ми вказуємо стовпець, за яким будемо групувати дані (у цьому випадку - "Район"), а в змінній `y` - стовпець зі значеннями, які будуть відображені на вісі `y` (у цьому випадку - "Ціна").

Не забудьте імпортувати необхідні бібліотеки `pandas`, `seaborn` та `matplotlib.pyplot`, а також викликати функцію `plt.show()`, щоб показати графік.

Тренажер №3

sales_data.csv

1. Завантажте набір даних про продажі продуктів у магазині. Це може бути CSV-файл або набір даних з розподіленими колонками.
2. Використовуючи Seaborn, побудуйте діаграми, що відображають:
   - Розподіл продажів за категоріями продуктів за допомогою стовпчикової діаграми (`bar plot`).
   - Залежність кількості продажів від ціни продукту за допомогою точкового графіку (`scatter plot`).
   - Розподіл продажів за регіонами за допомогою графіку "вусатих ящиків" (`box plot`).
3. Додайте на графіки необхідні підписи осей, заголовки та легенди, щоб зробити їх більш інформативними та зрозумілими.
4. Врахуйте рекомендації щодо використання кольорів, лінійок та інших налаштувань, щоб покращити візуальне враження ваших графіків.
5. Запишіть ваші графіки у файл або відобразіть їх безпосередньо в програмі, залежно від ваших потреб.

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

# Зчитуємо дані з CSV-файлу
data = pd.read_csv('sales_data.csv')

# Розподіл продажів за категоріями продуктів (стовпчикова д
іаграма)
plt.figure(figsize=(10, 6))
```

```
sns.countplot(x='Category', data=data)
plt.title('Розподіл продажів за категоріями продуктів')
plt.xlabel('Категорія')
plt.ylabel('Кількість продажів')
plt.xticks(rotation=45)
plt.show()

# Залежність кількості продажів від ціни продукту (точковий
графік)
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Price', y='Sales', data=data)
plt.title('Залежність кількості продажів від ціни продукт
y')
plt.xlabel('Ціна продукту')
plt.ylabel('Кількість продажів')
plt.show()

# Розподіл продажів за регіонами (графік "вусатих ящиків")
plt.figure(figsize=(10, 6))
sns.boxplot(x='Category', y='Sales', data=data)
plt.title('Розподіл продажів за категоріями продуктів')
plt.xlabel('Категорія')
plt.ylabel('Кількість продажів')
plt.xticks(rotation=45)
plt.show()
```

Додатки

https://drive.google.com/file/d/1kkgMhVIx99Qe2Q0x5bbLX1Sq5pRg4jzJ/view?usp=share_link

All datasets

https://github.com/mwaskom/seaborn-data