주요 차트

사용 라이브러리

- `Matplotlib`
  - Python에서 가장 기본적이고 널리 사용되는 시각화 라이브러리
  - 저수준(low-level) 라이브러리로, 플롯의 모든 요소를 세밀하게 제어할 수 있는 강력한 유연성을 제공
  - 거의 모든 종류의 정적(static) 2D 그래프를 제작 가능
  - `pyplot` 모듈이 주로 사용되며, 보통 `plt`라는 별칭으로 임포트
- `Seaborn`
  - `Matplotlib`을 기반으로 만들어진 고수준(high-level) 라이브러리
  - 더 적은 코드로 더 아름답고 통계적으로 의미 있는 그래프를 그릴 수 있도록 설계
  - `Pandas` 데이터프레임과 매우 잘 호환되며, `hue`, `col`, `row`와 같은 인자를 통해 복잡한 조건에 따른 다차원적인 데이터를 시각화하는 데 매우 강력
  - 보통 `sns`라는 별칭으로 임포트

그래프 종류

- **라인 플롯 (Line Plot)**: 시간의 흐름이나 순서에 따른 데이터의 변화 추세를 보여주는 데 사용됩니다. (e.g., 주가 변동, 월별 매출액)
- **히스토그램 (Histogram)**: 하나의 수치형 변수의 분포를 파악하기 위해, 데이터의 범위를 여러 구간(bin)으로 나누고 각 구간에 속하는 데이터의 빈도를 막대로 나타냅니다.
- **박스 플롯 (Box Plot)**: 하나의 수치형 변수의 분포를 사분위수(최소값, 1사분위수, 중앙값, 3사분위수, 최대값)를 이용하여 요약하고, 이상치를 탐지하는 데 사용됩니다. 여러 그룹 간의 분포를 비교하는 데 매우 효과적입니다.
- **산점도 (Scatter Plot)**: 두 수치형 변수 간의 관계(상관관계, 분포, 군집 등)를 파악하기 위해 각 데이터 포인트를 2차원 평면에 점으로 나타냅니다.
- **바 차트 (Bar Chart)**: 범주형 데이터의 각 항목에 대한 값(크기, 빈도)을 막대의 길이로 비교하여 보여줍니다. (e.g., 국가별 인구, 상품별 판매량)
- **파이 차트 (Pie Chart)**: 전체에 대한 각 범주의 **비율(구성 비중)**을 원 모양으로 나타내는 데 사용됩니다. 각 범주는 파이의 조각(slice)으로 표현되며, 조각의 크기는 해당 범주가 전체에서 차지하는 비율을 의미합니다. (e.g., 시장 점유율, 설문조사 응답 비율)
- **히트맵 (Heatmap)**: 2차원 행렬 형태의 데이터에서 각 셀의 값을 색상의 농도로 표현하는 시각화 방법입니다. 변수 간의 상관계수 행렬이나 혼동 행렬(confusion matrix)을 시각화하는 데 주로 사용됩니다.
- **페어 플롯 (Pair Plot)**: 데이터프레임에 있는 여러 수치형 변수들에 대해, 모든 변수 쌍 간의 산점도(off-diagonal)와 각 변수 자체의 분포(diagonal, 보통 히스토그램이나 KDE)를 한 번에 그려주는 강력한 탐색적 분석 도구입니다.

예제 데이터 생성

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

# Seaborn 내장 데이터셋 로드
tips = sns.load_dataset('tips')
iris = sns.load_dataset('iris')

# 라인 플롯을 위한 시계열 데이터 생성
time_series = pd.DataFrame({
    'date': pd.to_datetime(pd.date_range('2023-01-01', periods=12, freq='M')),
    'sales': np.random.randint(100, 500, size=12)
})
```

라인 플롯 (Line Plot)

- **용도**: 시간/순서에 따른 데이터 변화 추세 확인.
- **코드 예시**

```
plt.figure(figsize=(8, 4))

# seaborn으로 작성 시
sns.lineplot(x='date', y='sales', data=time_series, marker='o')

# seaborn 없이 작성 시 x, y 축 명칭 작성해야함
plt.plot(time_series['date'], time_series['sales'], marker='o')
plt.xlabel('date')
plt.ylabel('sales')

# 제목 추가 및 x축 꾸미기
plt.title('Monthly Sales Trend')
plt.xticks(rotation=45)
plt.show()
```

- **xticks**: x축 값 꾸미기
  - `rotation`: 기울기
  - `color`: r, g, b, c, m, y, k, w
  - `linestyle`: 'solid', 'dashed', 'dashdot', 'dotted', '-' , '--' , '-.' , ':'
  - `marker`

marker	description
"."	point
","	pixel

marker	description
"o"	circle
"v"	triangle_down
"^"	triangle_up
"<"	triangle_left
">"	triangle_right

히스토그램 (Histogram)

- **용도**: 단일 수치형 변수의 분포(빈도) 확인.
- **파라미터**
  - `bins`: int, 나눌 구간의 수
  - `edgecolor`: str, bar의 윤곽선 색깔을 지정
  - `range`: tuple, 표현해줄 값의 범위를 지정
  - `kde`: boolean, 밀도 함수 표시 여부
- `seaborn.distplot` **코드 예시**

```
plt.figure(figsize=(8, 5))
sns.distplot(tips['total_bill'], bins=20, kde=True)  # hist + kde
plt.title('Distribution of Total Bill')
plt.show()

# kde 함수만 별도로 그릴 때는 아래와 같이 작성 가능
sns.kdeplot(tips['total_bill'])
sns.kdeplot(x='total_bill', data = tips)
```

- **해석**
  - `total_bill` 데이터는 10~20달러 사이에 가장 많이 분포
  - 오른쪽으로 긴 꼬리를 갖는(right-skewed) 분포
- `matplot.pyplot.hist` **코드 예시**

```
from scipy.stats import gaussian_kde

plt.figure(figsize=(8, 5))

# (1) 히스토그램
plt.hist(tips['total_bill'], bins=20, density=True, alpha=0.6,
color='skyblue', range=(0, 60))#, edgecolor='black')

# (2) KDE (밀도 추정 곡선)
kde = gaussian_kde(tips['total_bill'])  # 데이터 기반 KDE 모델 생성
x = np.linspace(tips['total_bill'].min(), tips['total_bill'].max(), 200)  #
구간 생성
plt.plot(x, kde(x), color='b', linewidth=1)
```

```
# (3) 제목
plt.title('Distribution of Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Density')

plt.show()
```

- `plt.hist`로 반환되는 값은 튜플이며, ([빈도수], [구간 값]) 형태로 반환됨

박스 플롯 (Box Plot)

- **용도**: 데이터 분포 요약 및 그룹 간 분포 비교.
- **주의 사항**: 반드시 결측치 제외해야 그래프가 그려짐.
- **박스 해석**
  - 박스 플롯은 데이터의 사분위수를 시각화합니다.
  - 박스(box): IQR(Q1 ~ Q3) 범위
    - 박스 하단: 제1사분위수(Q1, 25%)
    - 박스의 선: 중앙값(Median, Q2, 50%)
    - 박스 상단: 제3사분위수(Q3, 75%)
  - 수염(whisker): Q1-1.5×IQR, Q3+1.5×IQR 범위 내의 최소·최대값
- `seaborn.boxplot` **코드 예시**

```
plt.figure(figsize=(8, 5))
# hue를 통해 각 값을 추가로 분리해서 그릴 수 있다.
sns.boxplot(x='day', y='total_bill', hue='smoker', data=tips)
plt.title('Total Bill Distribution by Day and Smoker')
plt.show()
```

- `seaborn.boxplot` **해석**
  - 주말(Sat, Sun)에 계산 금액이 평일보다 높은 경향 존재
  - 각 요일별로 흡연자와 비흡연자의 계산 금액 분포를 비교 가능
- `matplot.pyplot.boxplot` **코드 예시**:
  - `hue` 구현 시 for문을 통해 하나씩 그려야해서 pass
  - 관련 코드는 `practice.ipynb` 파일에 작성 (참고)

```
plt.figure(figsize=(8, 5))

# day별 total_bill 값 모아주기
data_to_plot = [tips[tips['day'] == day]['total_bill'] for day in
sorted(tips['day'].unique())]

# 박스플롯 그리기
plt.boxplot(data_to_plot, labels=sorted(tips['day'].unique()),
patch_artist=True)

plt.xlabel("day")
plt.ylabel("total_bill")
```

```
plt.title("Total Bill Distribution by Day")
plt.show()
```

산점도 (Scatter Plot)

- **용도**: 두 수치형 변수 간의 관계 파악.
- `seaborn.scatterplot` **코드 예시**

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='total_bill', y='tip', hue='time', size='size', data=tips,
alpha=0.7)
plt.title('Total Bill vs Tip')
plt.show()
```

- `seaborn.scatterplot` **해석**
  - `total_bill`이 증가할수록 `tip`도 증가하는 양의 상관관계 확인 가능
  - `hue`로 점심/저녁을, `size`로 인원수를 추가하여 다차원적인 정보를 표현 가능
- `matplot.pyplot.scatter` **코드 예시**
  - `hue`와 `size` 구현 시 for문 루프 돌면서 확인해주면 가능

```
plt.figure(figsize=(8, 6))

# 색상을 time에 따라 다르게 지정
colors = {'Lunch': 'skyblue', 'Dinner': 'orange'}

# 산점도 직접 반복해서 그리기
for time in tips['time'].unique():
    subset = tips[tips['time'] == time]
    plt.scatter(
        subset['total_bill'],
        subset['tip'],
        s=subset['size'] * 20,   # size 열을 마커 크기로 반영 (배율 조정)
        alpha=0.7,
        label=time,
        color=colors[time]
    )

plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.title("Total Bill vs Tip")

plt.legend(title="Time")
plt.show()
```

`seaborn` 제공 번외 산점도 그래프

- `sns.jointplot()` : 산점도와 각각의 히스토그램을 함께 보여줌

```
sns.jointplot(x='total_bill', y='tip', data=tips)
plt.show()
```

- `sns.regplot()` : 직선의 관계를 보여주며, 95% 신뢰구간을 함께 나타냄

```
sns.regplot(x='total_bill', y='tip', data=tips)
plt.show()
```

바 차트 (Bar Chart)

- **용도**: 범주별 데이터 크기 비교.
- `seaborn.barplot` **코드 예시**

```
plt.figure(figsize=(8, 5))
# barplot은 기본적으로 각 범주에 대한 평균값(추정치)과 신뢰구간을 보여줌
sns.barplot(x='day', y='total_bill', data=tips, palette='viridis')
plt.title('Average Total Bill by Day')
plt.show()

# 단순히 빈도를 세고 싶을 때는 countplot 사용
sns.countplot(x='day', data=tips, palette='pastel')
plt.title('Count of Records by Day')
plt.show()
```

- `seaborn.barplot` **해석**
  - 막대(bar): 각 값의 평균값
  - 세로 선(error bar): 해당 평균의 신뢰구간(CI, confidence interval)
    - 기본값은 95% 신뢰구간
    - 다른 통계와 에러바가 크게 겹칠 경우, 통계적 큰 차이가 없을 가능성이 높음
    - 다른 통계와 에러바가 거의 안 겹칠 경우, 통계적 차이가 있을 가능성이 높음
  - 신뢰구간(세로선)이 짧을수록 해당 평균 값에 대한 신뢰도가 높다.
  - 데이터가 많을수록, 편차가 적을수록 신뢰구간은 좁아진다.
- `matplot.pyplot.bar` **코드 예시**

```
plt.figure(figsize=(8, 5))

# day별 평균 total_bill 계산
mean_values = tips.groupby('day')['total_bill'].mean()

plt.bar(mean_values.index, mean_values.values)
plt.title('Average Total Bill by Day')
plt.xlabel('day')
plt.ylabel('total_bill')
plt.show()
```

```
# day별 빈도 계산
count_values = tips['day'].value_counts().sort_index()

plt.bar(count_values.index, count_values.values,
color=plt.cm.Pastel1(range(len(count_values))))
plt.title('Count of Records by Day')
plt.xlabel('day')
plt.ylabel('count')
plt.show()
```

파이 차트 (Pie Chart)

- **용도**: 범주형 데이터의 비율 시각화.
- **특징**
  - 각 범주가 전체에서 차지하는 비율을 원형 조각으로 표현
  - 전체 합이 100%라는 점에서 비율 비교에 직관적
  - 많은 범주가 있을 경우 시각적으로 복잡해질 수 있음
- `matplotlib.pyplot.pie` **코드 예시**

```
plt.figure(figsize=(7, 7))

# day별 빈도 계산
day_counts = tips['day'].value_counts()

# 파이 차트 그리기
plt.pie(
    day_counts.values,
    labels=day_counts.index,
    autopct='%1.1f%%',   # 각 조각의 비율 표시
    startangle=90,       # 12시 방향에서 시작
    counterclock=False,  # 시계 방향으로 회전
    shadow=True,         # 그림자 추가
    # 중심으로 부터 띄워둘 간격 (x 길이만큼 작성해줘야함)
    explode = [0.05, 0.05, 0.05, 0.05],
    colors=plt.cm.Pastel1(range(len(day_counts)))  # 색상 지정
)

plt.title('Proportion of Records by Day')
plt.show()
```

- `matplotlib.pyplot.pie` **해석**
  - 각 조각: 범주별 데이터 비율
  - `autopct`를 통해 백분율 표시 가능
  - `startangle`을 조정하면 시각적 시작 위치 변경 가능
  - 색상 지정으로 시각적 구분 가능
- `seaborn`**에는 직접적인 Pie Chart 함수 없음**

히트맵 (Heatmap)

- **용도**: 2D 행렬 데이터의 값 시각화 (상관계수, 혼동 행렬 등).
- **파라미터**
  - `annot`: boolean, 각 셀에 숫자(상관계수) 표기 여부
  - `fmt`: '.3f' 등 숫자 포맷, 소수점 3자리까지 표기
  - `cmap`: 칼라맵
    - "RdYlBu_r", "coolwarm"
  - `vmin`: int, 값의 최소값(-1)
  - `vmax`: int, 값의 최대값(1)
- `seaborn.heatmap` **코드 예시**

```
# iris 데이터의 수치형 변수 간 상관계수 계산
corr = iris[['sepal_length', 'sepal_width', 'petal_length',
'petal_width']].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Correlation Matrix of Iris Features')
plt.show()
```

- `seaborn.heatmap` **해석**
  - `petal_length`와 `petal_width`는 0.96으로 매우 강한 양의 상관관계 확인 가능
  - `sepal_width`와 `petal_length`는 -0.43으로 음의 상관관계 확인 가능
- `matplot.pyplot.boxplot` **코드 예시**는 `practice.ipynb` 파일에 작성 (참고)

페어 플롯 (Pair Plot)

- **용도**: 여러 변수 쌍 간의 관계를 한 번에 탐색.
- **주의사항**: 변수 및 데이터가 많다면, ①시간이 많이 걸리고 ②확인하기 어려움
- `seaborn.pairplot` **코드 예시**

```
# hue='species'를 통해 품종별로 색상을 다르게 표시, palette는 색상 팔레트
sns.pairplot(iris, hue='species', palette='husl', markers=['o', 's', 'D'])
plt.suptitle('Pair Plot of Iris Dataset', y=1.02)
plt.show()
```

- `seaborn.pairplot` **해석**
  - 대각선(diagonal)에는 각 변수의 분포(KDE)가 그려짐
  - 그 외의 칸(off-diagonal)에는 두 변수 간의 산점도가 그려짐
  - `petal_length`와 `petal_width`의 산점도를 보면,
    - 품종(species)에 따라 명확하게 군집이 형성되는 것을 한눈에 파악 가능
    - 두 변수가 품종을 분류하는 데 매우 중요한 역할을 할 것임을 시사