데이터 다루기 (4)

프롤로그: Data Analysis란? & 왜 시각화인가

GOAL

- Data Analysis의 개념과 역할을 이해한다.
- 시각화가 데이터 작업의 어디에, 왜 필요한지 설명할 수 있다.
- Matplotlib/Seaborn으로 기본 시각화의 틀을 잡을 수 있다.

목차

Data Analysis란?

> **질문에 답하기 위해** 데이터를 정리·요약·탐색하고,
> 패턴과 이상을 발견해 **의사결정에 도움**을 주는 일련의 활동

- 데이터 수집 → 정리/정제(결측·이상치) → 탐색(EDA) → 요약/설명 → 전달(리포트/대시보드)
- 핵심: **문맥 있는 해석**과 **명확한 커뮤니케이션**

예시로 이해하기

- 마케팅: “어떤 캠페인이 전환을 잘 냈나?” → 지표 비교, A/B 분석
- 운영: “재고는 어느 요일에 부족해지나?” → 추세·계절성 시각화
- 의료: “어떤 특징이 당뇨와 관련 있나?” → 변수 분포·상관 탐색

> 공통점: **질문→증거(데이터)→설명(그래프/표)**

Analysis vs Science vs Engineering

구분	초점	산출물	비유
Data Analysis	설명/요약/인사이트	리포트·대시보드	*지도 읽기*
Data Science	예측/추론/실험	모델·실험결과	*내비게이션 만들기*
Data Engineering	수집/저장/전달	파이프라인·플랫폼	*도로 깔기*

> Pandas를 쓰면 보통 **Analysis(탐색·정제)**를 하고,
> 필요하면 **Science(모델링)**로 확장,
> 대규모 운영은 **Engineering**이 받쳐 준다.

데이터 워크플로 & 시각화의 자리

1. 문제·가설 정의
2. 데이터 수집·적재(Engineering 비중)
3. **정제/전처리(결측/이상/형 변환)**
4. **탐색적 데이터 분석(EDA)** ← ***여기서 시각화 필수***
5. 모델링/검증(Science)
6. **결과 전달(스토리텔링 시각화)**

> 시각화는 **초기 EDA**와 **최종 커뮤니케이션**에서 모두 중요

왜 시각화인가? (3가지 이유)

1. **발견(Discovery)**: 표로는 안 보이는 **패턴/이상치/군집**을 눈으로 즉시 파악
2. **검증(Validation)**: 가정(정규성, 선형성 등)과 전처리 효과를 **빠르게 확인**
3. **전달(Communication)**: 비전문가도 **한눈에 이해**하도록 스토리화

> 비유: “원시 신호(표)”를 “그림(그래프)”로 번역해 **두뇌 친화적**으로 보여준다

시각화를 안 하면?

- 평균만 보고 **왜곡**을 놓치기 쉽다(꼬리·이상치·다봉형 분포).
- 상관/관계가 **숫자 한 줄**로 축약되어 맥락을 잃는다.
- 설득과 공유가 어려워 **의사결정 지연**.

시각화 기본 원칙

- 질문과 대상에 **맞는 차트 유형** 선택(단변량/이변량/다변량)
- 축 범위와 스케일(로그/리밋)을 **정직하게** 표현
- 범례/라벨/단위를 **명확히**
- **색은 의미**를 위해서만(장식 최소화)
- 샘플 수와 결측·이상치를 **명시/표현**

흔한 함정(주의)

- 잘못된 축 절단으로 **차이 과장**
- 범주형을 **임의 수치**로 인코딩해 **순서 착시** 유발
- 과도한 겹침/투명도 → 해석 어려움
- 다중 비교/표본 편향을 무시한 **확증편향**

어떤 그래프를 고를까?

- **단변량(숫자형)**: 히스토그램, KDE, 박스/바이올린
- **단변량(범주형)**: 카운트플롯, 막대(요약통계)
- **이변량(숫자-숫자)**: 산점도(+회귀선), 2D KDE/hexbin
- **이변량(범주-숫자)**: 박스/바이올린/막대(CI)
- **다변량**: 상관행렬, pairplot, 페싯(facet) 그리드

> 아래부터는 **Matplotlib** ↔ **Seaborn** 비교로 실습

Matplotlib vs Seaborn 한눈 비교

항목	Matplotlib	Seaborn
철학	저수준(모든 요소 직접 제어)	고수준(통계적 시각화 추상화)
강점	정교한 커스터마이징, 복잡한 레이아웃	빠른 인사이트, 일관된 스타일, 범주형 지원
전략	**Seaborn으로 초안** → Matplotlib로 **미세 조정**	

Matplotlib

큰 그림(객체 지향 방식)

- Figure (도화지) 안에 Axes(그래프 판)
- 그 위에 선/막대/텍스트 같은 Artist가 얹힘

자주 쓰는 기본 그래프

- 선 그래프: ax.plot(x, y) — 추이/시간
- 산점도: ax.scatter(x, y) — 두 변수 관계
- 막대/가로막대: ax.bar(x, h), ax.barh(y, w) — 범주 비교
- 히스토그램: ax.hist(x, bins=30) — 분포
- 박스플롯: ax.boxplot(data) — 사분위/이상치

예시 데이터

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

# (1) Line/Scatter용 연속형 데이터
x_line = np.linspace(0, 10, 30)          # 0~10 사이 균등 간격 30점
y_line = np.sin(x_line) + 0.2 * np.random.randn(len(x_line))  # 사인 + 잡음

x_scatter = np.linspace(0, 10, 60)       # 산점도 X: 60점
y_scatter = 0.7 * x_scatter + 1.5 + np.random.randn(len(x_scatter)) * 1.2  # 선형관계 + 잡음

# (2) Bar용 범주형 요약 데이터
categories = ["A", "B", "C", "D"]        # 막대 카테고리 라벨
values = [23, 17, 35, 29]                # 각 카테고리의 값

# (3) Hist/Box용 분포 데이터
samples_norm = np.random.normal(loc=0, scale=1, size=400)     # 정규분포 표본 400개
box_a = np.random.normal(0.0, 1.0, 200)  # 그룹 A 분포
box_b = np.random.normal(1.5, 0.8, 200)  # 그룹 B 분포(평균↑, 분산↓)
box_c = np.random.normal(3.0, 1.2, 200)  # 그룹 C 분포(평균↑, 분산↑)
```

Line plot

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)  # 도화지(fig)와 좌표(ax) 생성
ax.plot(x_line, y_line, marker='o', linestyle='--')            # 선 그래프: 점·점선 스타일
ax.set_title("Line Plot (Matplotlib)")         # 제목
ax.set_xlabel("X")                              # X축 라벨
ax.set_ylabel("Y")                              # Y축 라벨
ax.grid(True, linestyle=':')                    # 격자 표시(점선)
fig.tight_layout()                              # 여백 자동 조정
plt.show()                                      # 화면에 표시
```

Scatter plot

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)                    # 새로운 캔버스
ax.scatter(x_scatter, y_scatter)                                   # 산점도
ax.set_title("Scatter Plot (Matplotlib)")      # 제목
ax.set_xlabel("X")                              # X축 라벨
ax.set_ylabel("Y")                              # Y축 라벨
ax.grid(True, linestyle=':')                    # 격자
fig.tight_layout()                              # 여백 정돈
plt.show()                                      # 표시
```

Bar plot

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)                    # 새 캔버스
ax.bar(categories, values)                                         # 범주형 막대 그래프
ax.set_title("Bar Plot (Matplotlib)")          # 제목
ax.set_xlabel("Category")                        # X축 라벨(범주)
ax.set_ylabel("Value")                           # Y축 라벨(값)
ax.grid(True, axis='y', linestyle=':')           # y축 방향 격자만
fig.tight_layout()                               # 여백 정돈
plt.show()                                       # 표시
```

Histogram

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)                    # 새 캔버스
ax.hist(samples_norm, bins=20)                                     # 히스토그램(빈=20)
ax.set_title("Histogram (Matplotlib)")      # 제목
ax.set_xlabel("Value")                          # 값 축 라벨
ax.set_ylabel("Count")                          # 빈도 축 라벨
ax.grid(True, axis='y', linestyle=':')          # y축 격자
fig.tight_layout()                              # 여백 정돈
plt.show()                                      # 표시
```

Box plot

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)                # 새 캔버스
ax.boxplot([box_a, box_b, box_c], labels=["Group A", "Group B", "Group C"])  # 박스플롯
ax.set_title("Box Plot (Matplotlib)")        # 제목
ax.set_xlabel("Group")                         # 그룹 라벨
ax.set_ylabel("Value")                         # 값 라벨
ax.grid(True, axis='y', linestyle=':')         # y축 격자
fig.tight_layout()                              # 여백 정돈
plt.show()
```

Seaborn

- Matplotlib 위에서 돌아가는 고수준 통계 시각화
- tidy DataFrame에 최적 (열=변수, 행=관측치)
- 색상 팔레트/Facet/통계요약이 기본 내장

line plot

```
df_line = pd.DataFrame({"x": x_line, "y": y_line})

fig, ax = plt.subplots(figsize=(6, 4), dpi=120)
sns.lineplot(data=df_line, x="x", y="y", marker="o", ax=ax)
ax.set_title("Line Plot (Seaborn)")
ax.grid(True, linestyle=":")
fig.tight_layout()
plt.show()
```

Scatter plot

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)
sns.scatterplot(data=df_scatter, x="x", y="y", ax=ax)
ax.set_title("Scatter Plot (Seaborn)")
ax.grid(True, linestyle=":")
fig.tight_layout()
plt.show()
```

Bar plot

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)
sns.barplot(data=df_bar, x="category", y="value", ax=ax)
ax.set_title("Bar Plot (Seaborn)")
ax.grid(True, axis='y', linestyle=":")
fig.tight_layout()
plt.show()
```

Histogram

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)
sns.histplot(data=df_hist, x="value", bins=20, kde=True, ax=ax)
ax.set_title("Histogram (Seaborn)")
ax.grid(True, axis='y', linestyle=":")
fig.tight_layout()
plt.show()
```

Box plot

```
fig, ax = plt.subplots(figsize=(6, 4), dpi=120)
sns.boxplot(data=df_box, x="group", y="value", ax=ax)
ax.set_title("Box Plot (Seaborn)")
ax.grid(True, axis='y', linestyle=":")
fig.tight_layout()
plt.show()
```

오픈소스 데이터

```
import seaborn as sns
iris = sns.load_dataset("iris")
tips = sns.load_dataset("tips")
penguins = sns.load_dataset("penguins")
```

- **iris**: 수치+품종(분류 예시)
- **tips**: 범주×수치(요약/비교)
- **penguins**: 결측 포함(현실성↑)

단변량 예: 히스토그램(iris·sepal_length)

- 변수 한 개만 놓고 그 분포와 특징을 요약·파악하는 분석
- 중심(평균·중앙값), 퍼짐(분산·표준편차·IQR), 모양(왜도·꼬리), 이상치를 볼 때 사용

Matplotlib

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.hist(iris["sepal_length"], bins=20, alpha=0.8)
ax.set_title("Iris · Sepal Length (Histogram)")
ax.set_xlabel("sepal_length"); ax.set_ylabel("count")
plt.show()
```

Seaborn

```
sns.histplot(iris, x="sepal_length", bins=20, kde=True)
plt.title("Iris · Sepal Length (Hist + KDE)"); plt.show()
```

이변량 예: 산점도(iris·sepal↔petal, hue=species)

- 변수 두 개 사이의 **관계(연관·패턴)**를 살피는 분석
- 선형/비선형 관계, 상관계수, 집단 간 차이, 교차비율을 봄

Matplotlib

```
fig, ax = plt.subplots()
ax.scatter(iris["sepal_length"], iris["petal_length"], alpha=0.7)
ax.set_title("Sepal vs Petal Length")
ax.set_xlabel("sepal_length"); ax.set_ylabel("petal_length")
plt.show()
```

Seaborn

```
sns.scatterplot(iris, x="sepal_length", y="petal_length",
                hue="species", style="species", alpha=0.85)
plt.title("Sepal vs Petal Length · by Species"); plt.show()
```

상관행렬(iris 수치형)

```
num = iris.select_dtypes("number")
corr = num.corr()

# Seaborn
sns.heatmap(corr, annot=True, fmt=".2f", square=True)
plt.title("Correlation Matrix (Iris)"); plt.show()
```

> 높은 상관쌍은 **중복 정보·예측력** 평가에 단서

실습

- 당뇨병 데이터의 결측치를 채우고 시각화 하기