Python
CH12: Matplotlib

전종준 교수

서울시립대학교 통계학과/통계데이터사이언스대학원

Contents

- 산점도 그리기

- 컬러와 시각화

- 2D 시각화

산점도 그리기

강의 개요: matplotlib 기초부터 실전까지

- **1. matplotlib 소개와 기본 구조**
  - pyplot과 객체 지향 방식
  - Figure와 Axes의 관계 이해
- **2. 기본 그래프 그리기**
  - plot, scatter 함수 사용법
  - 선/점의 스타일 지정 (색상, 마커, 라벨 등)
- **3. 시각적 구성 요소 다루기**
  - 축 레이블, 제목, 범례(legend), 눈금 설정
  - colorbar, colormap, layout 정렬
- **4. 2차원 데이터 시각화**
  - meshgrid를 활용한 등고선 시각화
  - contour, contourf, pcolormesh 응용
- **5. 마무리 및 실습**
  - 시각화 조합 실습 및 종합 예제
  - 코드 따라하며 시각적 직관 키우기

matplotlib이란?

- 파이썬에서 사용하는 대표적인 데이터 시각화 라이브
- 가장 많이 사용하는 모듈: `pyplot`

기본 구조 예시:

```
1 import matplotlib.pyplot as plt
2
3 fig, ax = plt.subplots()
4 ax.plot([1, 2, 3], [1, 4, 9])
5 plt.show()
```

- `Figure`: 전체 캔버스
- `Axes`: 실제 그래프가 그려지는 영역
- plt.show(): 그래픽 객체를 화면에 보여줌

기본 플롯: plot과 scatter

- `plot()`: 선 그래프(Line Plot)를 그릴 때 사용
- `scatter()`: 점 단위로 데이터를 표현할 때 사용
- `fig`, `ax`는 matplotlib의 핵심 구조:
  - `fig` (`Figure`): 전체 그래프 캔버스
  - `ax` (`Axes`): 실제 데이터가 그려지는 영역
  - 한 Figure 안에 여러 Axes 배치 가능

선 그래프 예시:

```
1  import matplotlib.pyplot as plt
2
3  x = [1, 2, 3, 4]
4  y = [1, 4, 9, 16]
5
6  fig, ax = plt.subplots() # fig: 전체 도화지, ax: 그래프 영역
7  ax.plot(x, y)
8  ax.set_title("선 그래프 예제")
```

객체 지향 방식 예시

```
fig, ax = plt.subplots()  # Figure와 Axes 생성
ax.plot([1, 2, 3], [1, 4, 9])
ax.set_xlabel("X축")
ax.set_ylabel("Y축")
ax.set_title("간단한 그래프")
plt.show()
```

기본 플롯: plot과 scatter

산점도 예시:

```
fig, ax = plt.subplots()
ax.scatter(x, y)
ax.set_title("산점도 예제")
```

fig와 ax의 속성 설정하기

- `ax.set_title()`: 그래프 제목 설정
- `ax.set_xlabel()`, `set_ylabel()`: 축 이름 지정
- `ax.set_xticks()`, `set_yticks()`: 눈금 위치와 라벨 설정
- `ax.margins()`: 그래프 여백 설정

예시 코드:

```
fig, ax = plt.subplots()
x = [1, 2, 3, 4]
y = [1, 4, 9, 16]
labels = ['A', 'B', 'C', 'D']

ax.plot(x, y)
ax.set_title("제목: 예제 그래프", fontsize=14)
ax.set_xlabel("x 축", fontsize=12)
ax.set_ylabel("y 축", fontsize=12)
ax.set_xticks(x, labels, rotation='vertical')  # 눈금 라벨 회전
ax.margins(x=0.2, y=0.2)  # 여백 설정
```

Figure(fig)의 속성 설정하기

- `figsize`: 그래프의 전체 크기 조정 (단위: 인치)
- `facecolor`: 캔버스 배경 색상 설정
- `fig.colorbar()`: 컬러바 추가 (시각화 강조)

예시 코드:

```
fig, ax = plt.subplots(figsize=(8, 4), facecolor='lightgray')

x = [1, 2, 3, 4]
y = [1, 4, 9, 16]
sc = ax.scatter(x, y, c=y, cmap='viridis')

fig.colorbar(sc)  # 컬러바는 fig에 추가
ax.set_title("fig 크기와 컬러바 예제")
```

여러 그래프의 레이아웃 설정하기

- `plt.subplots(rows, cols)`로 여러 그래프를 생성할 수 있음
- `layout="constrained"` 옵션을 사용하면 간격을 자동 조정
- 각 `ax`에 개별 그래프를 그릴 수 있음

예시: 2행 2열 subplot 생성

```
fig, axs = plt.subplots(2, 2, layout="constrained")

for ax in axs.flat:
    ax.plot([1, 2, 3], [1, 4, 9])
    ax.set_title("소형 그래프")
```

참고:

- `axs.flat`을 사용하면 2D 배열을 1D로 순회할 수 있어요.
- 각 `ax`는 개별 그래프 영역이므로 따로 설정이 가능합니다.

plot과 scatter에서 자주 쓰는 옵션들

- `color` 또는 `c`: 색상 지정 (이름, RGB, 컬러맵 등)
- `linestyle` 또는 `ls`: 선의 스타일 (`-`, `--`, `:` 등)
- `marker`: 데이터 포인트의 모양 (`o`, `s`, `.` 등)
- `linewidth` 또는 `lw`: 선의 두께
- `s`: (scatter 전용) 점의 크기

plot 예시:

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3], [1, 4, 9],
        color='green', linestyle='--',
        marker='o', linewidth=2)
```

scatter 예시:

```
fig, ax = plt.subplots()
ax.scatter([1, 2, 3], [1, 4, 9],
           color='red', marker='^', s=100)
```

실습 문제 1

- x = [1, 2, 3, 4, 5], y = [1, 4, 9, 16, 25] 데이터를 이용하여:
  1. 산점도를 그리고
  2. 제목과 축 라벨을 추가하고
  3. 빨간색 삼각형 마커 사용

실제 데이터 산점도 예제

- pandas로 CSV 파일을 불러온 뒤
- matplotlib로 산점도를 그려봅니다.
- 예시 데이터: seaborn의 penguins.csv

```python
import pandas as pd
import matplotlib.pyplot as plt

# CSV 파일 불러오기 (예: 펭귄 데이터)
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/penguins.csv"
df = pd.read_csv(url)
# bill_length_mm vs bill_depth_mm 산점도
fig, ax = plt.subplots()
ax.scatter(df["bill_length_mm"], df["bill_depth_mm"],
           color="teal", alpha=0.6)
ax.set_xlabel("Bill Length (mm)")
ax.set_ylabel("Bill Depth (mm)")
ax.set_title("Penguin Beak Size")
plt.show()
```

mtcars 데이터 산점도

- `mtcars` 데이터셋을 불러와 `mpg`와 `hp` 관계를 시각화
- `pandas`로 불러온 후 `matplotlib`로 그립니다.

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

# mtcars 데이터 불러오기
url = "https://gist.githubusercontent.com/seankross/a412dfbd88b3db70b74b/raw/mtcars.csv"
df = pd.read_csv(url)
# mpg vs hp 산점도
fig, ax = plt.subplots()
ax.scatter(df["hp"], df["mpg"], color="purple", alpha=0.7)
ax.set_xlabel("Horsepower (hp)")
ax.set_ylabel("Miles per Gallon (mpg)")
ax.set_title("Fuel Efficiency vs Engine Power")
plt.show()
```

iris 데이터 산점도

- seaborn에서 제공하는 유명한 붓꽃 데이터
- 꽃잎 길이(petal_length) vs 꽃잎 폭(petal_width)를 시각화

```
import seaborn as sns
import matplotlib.pyplot as plt
# iris 데이터 불러오기
iris = sns.load_dataset("iris")
# 산점도
fig, ax = plt.subplots()
ax.scatter(iris["petal_length"], iris["petal_width"], alpha=0.7)
ax.set_xlabel("Petal Length (cm)")
ax.set_ylabel("Petal Width (cm)")
ax.set_title("Iris Petal Size")
plt.show()
```

정리

- `plt.plot()`, `ax.plot()`, `ax.scatter()` 기본 사용법
- Figure와 Axes의 개념
- 제목, 라벨, 범례, 눈금 설정 방법
- 다음 시간: 색상과 스타일 심화

컬러와 시각화

색상 표현 방법과 컬러맵 사용

- `color=` 옵션은 다양한 방식으로 색상을 지정할 수 있음:
  - 색상명 (예: `'red'`, `'blue'`)
  - RGB 문자열 (예: `'#FF0000'`)
  - RGB 튜플 (예: `(1.0, 0.0, 0.0)`)
- 컬러맵(`cmap`)은 값의 크기에 따라 색을 자동으로 매핑

색상명 → RGB 튜플 변환 예시:

```
1  import matplotlib.colors as mcolors
2  mcolors.to_rgb('red')   # 결과: (1.0, 0.0, 0.0)
```

컬러맵을 활용한 예시 (hot):

```
1  import numpy as np
2  x = np.linspace(0, 1, 100)
3  fig, ax = plt.subplots()
4  sc = ax.scatter(x, x, c=x, cmap='hot')  # 컬러맵 적용
5  fig.colorbar(sc)
```

colorbar 사용법

- `fig.colorbar()`는 그래프 옆에 색상의 기준 눈금을 추가해주는 함수
- `cmap`과 함께 사용되어 색상의 값 범위를 시각적으로 알려줌
- 입력으로는 컬러 정보가 포함된 시각화 객체 (`scatter`, `contourf`, `pcolormesh` 등)를 넘겨줌

예시: scatter와 colorbar 함께 사용하기

```
x = np.linspace(0, 1, 100)
y = np.cos(2 * np.pi * x)
fig, ax = plt.subplots()
sc = ax.scatter(x, y, c=y, cmap='viridis')
fig.colorbar(sc)  # scatter 객체를 넘겨줌
```

주의: `fig.colorbar(...)`는 `fig`에 적용해야 함

Colormap 적용

- c 옵션: 각 데이터 점의 색을 나타내는 값 (리스트나 배열)
- cmap: c 값에 따라 색상을 매핑할 색상표 지정
- 값이 클수록 색상표 상의 '고온' 쪽으로, 작을수록 '저온' 쪽으로 매핑

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 x = np.linspace(0, 10, 50)
4 y = np.sin(x)
5 fig, ax = plt.subplots()
6 # c = y : y값에 따라 색상 변경
7 scatter = ax.scatter(x, y, c=y, cmap="plasma")
8 fig.colorbar(scatter, ax=ax)
9 plt.show()
```

astype("category")

- **목적:** 범주형(categorical) 자료형으로 변환
- 범주형 자료형의 특징:
  1. 데이터가 제한된 **범주(Category)** 값만 가짐
  2. 내부적으로는 숫자 코드로 저장되어 메모리 효율 ↑
  3. 순서가 없는 범주형(예: 색상)과 순서 있는 범주형(예: 등급) 모두 표현 가능
- Colormap 적용을 위해 문자열 범주를 숫자로 매핑할 때 필수

```
import pandas as pd
df = pd.DataFrame({"species": ["setosa", "versicolor", "virginica"]})
# 문자열 -> 범주형 변환
cat_var = df["species"].astype("category")
print(cat_var)
print(cat_var.cat.categories)  # 카테고리 목록
species_codes = cat_var.cat.codes
print(species_codes) # 매핑된 숫자 (알파벳 순서)
```

astype("category")

```
import pandas as pd
df = pd.DataFrame({
  "species": ["setosa", "versicolor", "virginica", "setosa"]
})
# 원하는 순서 지정
order = ["virginica", "setosa", "versicolor"]
df["species"] = pd.Categorical(df["species"], categories=order, ordered=True)
codes = df["species"].cat.codes
print(codes)
# virginica -> 0, setosa -> 1, versicolor -> 2
```

범주형 변수에 Colormap 적용

- species는 문자열로 되어 있어 Colormap에 바로 사용 불가
- .astype("category").cat.codes로 정수 코드로 변환
- 각 코드 값이 Colormap에서 색상으로 매핑됨

```
import seaborn as sns
import matplotlib.pyplot as plt
  # iris 데이터 불러오기
df = sns.load_dataset("iris")
# 범주형 변수를 코드로 변환
codes = df["species"].astype("category").cat.codes
fig, ax = plt.subplots()
scatter = ax.scatter(df["petal_length"], df["petal_width"],
                     c=codes, cmap="Set1", s=50, alpha=0.8)
ax.set_xlabel("Petal Length (cm)")
ax.set_ylabel("Petal Width (cm)")
ax.set_title("Iris Petal Size by Species")
plt.show()
```

범례(legend) 사용하기

- `ax.legend()`를 사용하면 그래프에 범례(legend)를 추가할 수 있음
- 각 데이터 시리즈에 `label`을 지정해줘야 범례에 표시
- `loc=` 옵션으로 범례의 위치를 지정할 수 있음

예시 코드:

```
x = [1, 2, 3, 4]
y1 = [1, 4, 9, 16]
y2 = [1, 2, 4, 8]

fig, ax = plt.subplots()
ax.plot(x, y1, label="제곱")
ax.plot(x, y2, label="지수", linestyle='--')
ax.legend(loc='upper left')  # 범례 위치 지정
```

Tip:

- `label`은 시각적 요소를 설명하는 텍스트
- `legend()`는 마지막에 한 번만 호출

mtcars: 실린더 수별 색상 구분 산점도

- `cyl`을 범주형으로 변환 후 코드(`0, 1, 2`) 생성
- `c`에 코드 값 전달 → `cmap`에서 자동 색상 매핑
- 범례는 카테고리 이름과 colormap 색상으로 수동 생성

mtcars: 실린더 수별 색상 구분 산점도

```python
import pandas as pd
import matplotlib.pyplot as plt
# 데이터 불러오기
url = "https://gist.githubusercontent.com/seankross/a412dfbd88b3db70b74b/raw/mtcars.csv"
df = pd.read_csv(url)
# cyl → 범주형 코드 변환
codes = df["cyl"].astype("category").cat.codes
fig, ax = plt.subplots()
ax.scatter(df["hp"], df["mpg"], c=codes, cmap="Set1", s=80, alpha=0.8)
ax.set_xlabel("Horsepower (hp)")
ax.set_ylabel("Miles per Gallon (mpg)")
ax.set_title("Fuel Efficiency by Cylinder Count")
plt.show()
```

2D 시각화

2D 데이터 시각화 개념

- 2차원 좌표 (x, y)와 해당 위치의 값 z를 색상으로 표현
- 대표적인 방법:
  - 등고선 플롯 (`contour`, `contourf`)
  - 색상 매핑 (`pcolormesh`, `imshow`)
- 과학·공학 데이터 분석, 지도 제작, 이미지 처리 등에 활용

meshgrid의 원리

- 1D 좌표 배열 x, y를 받아서 2D 좌표 행렬을 생성
- x는 열 방향으로 복제, y는 행 방향으로 복제
- 반환값: (X좌표 배열, Y좌표 배열) (둘 다 2차원)

meshgrid 예시와 반환값

```
1 import numpy as np
2 x = np.linspace(0, 2, 3)  # [0. , 1. , 2.]
3 y = np.linspace(10, 30, 3)  # [10., 20., 30.]
4 xs, ys = np.meshgrid(x, y)
5 print("xs:\n", xs)
6 print("ys:\n", ys)
```

간단한 2D 함수 시각화

```
import matplotlib.pyplot as plt
x = np.linspace(-2, 2, 100)
y = np.linspace(-2, 2, 100)
xs, ys = np.meshgrid(x, y)
# 2D 함수
zs = np.sin(xs**2) + np.cos(ys**2)
plt.imshow(zs, extent=[-2, 2, -2, 2], origin='lower')
plt.colorbar()
plt.show()
```

Colorbar 개념

- 색상과 수치 값의 관계를 시각적으로 표시
- Colormap 사용 시 필수적인 보조 도구
- `fig.colorbar(mappable)` 로 추가
- 해석 시 색상 범위와 실제 값 범위를 혼동하지 않도록 주의

Colorbar 값 범위 제한 (imshow)

- vmin, vmax로 색상 매핑 범위를 제한
- 해당 범위 밖의 값은 Colormap 양 끝 색상으로 표시

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-2, 2, 100)
y = np.linspace(-2, 2, 100)
xs, ys = np.meshgrid(x, y)
# 2D 함수
zs = np.sin(xs**2) + np.cos(ys**2)
plt.imshow(zs, extent=[-2, 2, -2, 2],
            origin='lower', vmin=-0.5, vmax=0.5,  # 색상 범위 제한
            cmap='viridis')
plt.colorbar(label="Function Value")
plt.title("Color range: -0.5 ~ 0.5")
plt.show()
```

등고선 시각화: contour와 contourf

- `contour()`: 선으로 등고선(동일한 높이)을 그림
- `contourf()`: 영역을 색으로 채운 등고선을 그림
- `clabel()`: 등고선 선 위에 수치를 표시

예시 코드:

```
import numpy as np
x = np.linspace(-1, 1, 100)
y = np.linspace(-1, 1, 100)
xs, ys = np.meshgrid(x, y)
zs = np.sqrt(np.sin(xs**2) + np.cos(ys**2))

fig, ax = plt.subplots()
c = ax.contour(x, y, zs, colors='black', linewidths=0.5)
ax.clabel(c)  # 선 위에 수치 표시

cf = ax.contourf(x, y, zs, cmap='plasma')  # 색 채움
fig.colorbar(cf)  # 색 기준 눈금
```

linspace와 meshgrid: 2차원 데이터를 만들기

- `np.linspace(start, stop, num)`: 일정한 간격으로 값을 생성
- `np.meshgrid()`: 1D 좌표 배열을 2D 좌표 그리드로 변경
- 2차원 그래프(등고선 등)를 그릴 때 좌표 격자 정보가 필요

예시 코드:

```
1  import numpy as np
2
3  x = np.linspace(-1, 1, 5)
4  y = np.linspace(-1, 1, 5)
5
6  xs, ys = np.meshgrid(x, y)
7  print(xs.shape)  # (5, 5)
8  print(ys.shape)  # (5, 5)
```

결과: 각 `(x, y)` 좌표쌍을 이루는 2차원 배열을 생성

contour의 작동 방식

- `contour(x, y, z)`는 3차원 함수 $z = f(x, y)$의 등고선을 2D에 그림
- `x, y`: meshgrid로 만든 2차원 좌표
- `z`: 각 위치에서의 함수값 (x, y → z)
- 동일한 높이(z)를 가지는 점들을 선으로 연결하여 보여줌

z값 만들기 예시:

```
1 zs = np.sqrt(np.sin(xs**2) + np.cos(ys**2))
```

이제 준비된 (x, y, z)를 `contour()`에 넘기면, 함수의 등고선을 시각화할 수 있음

등고선 시각화 예제: contour와 contourf

- `contour()`: 선만 그리기
- `contourf()`: 색으로 채운 등고선
- `clabel()`: 등고선 위에 수치 라벨 추가

```
fig, ax = plt.subplots()

c = ax.contour(x, y, zs, colors='black', linewidths=0.5)
ax.clabel(c)  # 선 위에 수치 표시

cf = ax.contourf(x, y, zs, cmap='plasma')  # 색상 채우기
fig.colorbar(cf)
ax.set_title("contour & contourf 예제")
```

contour 입력을 위한 (x, y, z)의 조건

- `contour(x, y, z)`를 사용하려면 **좌표쌍(x, y)**과 **함수값 z**가 아래 조건을 만족해야 해요:

1. `x`, `y`는 2차원 좌표 그리드여
   - 보통 `np.meshgrid()`로 생성해요
   - `x.shape == y.shape == z.shape`
2. `z[i, j]`는 `x[i, j]`, `y[i, j]` 위치에서의 함수값
3. 2D 배열의 구조가 맞지 않으면 에러가 발생하거나 잘못된 그림이 그려짐

예시:

```
x = np.linspace(-1, 1, 100)
y = np.linspace(-1, 1, 100)
xs, ys = np.meshgrid(x, y)       # xs.shape == ys.shape == (100, 100)
zs = np.sin(xs**2 + ys**2)       # z도 같은 크기로 계산
```

contour 입력값의 조건: 좌표 정렬까지 포함

- `contour(x, y, z)`를 사용하기 위해서는 다음 조건을 만족

1. shape 조건

- `x.shape == y.shape == z.shape`
- 보통 `np.meshgrid`로 생성된 2차원 배열

2. 정렬 조건 (중요!)

- `x[0, :]` 또는 `x[:, 0]` 은 **오름차순**
- `y[:, 0]` 또는 `y[0, :]` 도 **오름차순**
- 등고선은 이 정렬 순서를 기준으로 계산되기 때문임

예시:

```
x = np.linspace(-1, 1, 100)     # 정렬된 1D 배열
y = np.linspace(-1, 1, 100)
xs, ys = np.meshgrid(x, y)      # 2D 격자 생성
zs = np.sin(xs**2 + ys**2)
```

pcolormesh로 부드럽게 색상 시각화하기

- `pcolormesh(x, y, z)`는 각 (x, y) 영역을 색으로 채우는 격자 시각화
- `contourf`보다 시각적으로 더 부드럽고 빠르게 표현
- 색상 기준은 `vmin`, `vmax`, `cmap`으로 조정 가능

예시:

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

x = np.linspace(-1, 1, 100)
y = np.linspace(-1, 1, 100)
xs, ys = np.meshgrid(x, y)
zs = np.sin(xs**2 + ys**2)

fig, ax = plt.subplots(figsize=(8, 4))
pc = ax.pcolormesh(xs, ys, zs, cmap='plasma',
                   vmin=0.0, vmax=1.0)

fig.colorbar(pc)
ax.set_title("pcolormesh로 부드러운 색상 표현")
```

pcolormesh 위에 contour 겹쳐 그리기

- `pcolormesh()`로 색상으로 표현
- `contour()`로 같은 데이터를 선으로 덧씌울 수 있음
- `clabel()`을 사용하면 등고선 라벨도 함께 표시할 수 있음

예시:

```
fig, ax = plt.subplots(figsize=(8, 4))
# 부드러운 색상 배경
pc = ax.pcolormesh(xs, ys, zs, cmap='plasma', vmin=0.0, vmax=1.0)
fig.colorbar(pc)
# 흰색 등고선 겹치기
c2 = ax.contour(xs, ys, zs, linewidths=0.5, colors='white')
ax.clabel(c2)  # 등고선 라벨 추가
ax.set_title("pcolormesh + contour 조합 시각화")
```

실제 수치지도 데이터 불러오기

- 인터넷에서 DEM(디지털 표고 모델) 또는 해양 깊이 데이터 사용

```
!pip install srtm.py
import numpy as np
import matplotlib.pyplot as plt
import srtm
min_lon, max_lon = 127.0, 128.0 # # 1) 관심 영역(작게): 경도[127.0, 128.0], 위도[36.0, 37.
min_lat, max_lat = 36.0, 37.0
nx, ny = 300, 300 # 2) 격자 해상도 설정(너무 크면 느림)
lons = np.linspace(min_lon, max_lon, nx)
lats = np.linspace(min_lat, max_lat, ny)
# 3) SRTM 데이터 핸들러
elev = srtm.get_data()  # 필요 타일 자동 다운로드
# 4) DEM 격자 생성 (행: 위도, 열: 경도)
Z = np.empty((ny, nx), dtype=float)
for i, la in enumerate(lats):
    for j, lo in enumerate(lons):
        h = elev.get_elevation(la, lo)
        Z[i, j] = np.nan if h is None else h
```

수치지도 pcolormesh 시각화

```
# 5) 2D 시각화
# 5) 2D 시각화
plt.figure(figsize=(7,5))
lon_grid, lat_grid = np.meshgrid(lons, lats)
# 5) pcolormesh 시각화
plt.figure(figsize=(7,5))
pc = plt.pcolormesh(
    lon_grid, lat_grid, Z,
    cmap="terrain", shading="auto"
)
plt.contour(lon_grid, lat_grid, Z, levels=5, c="white")
plt.colorbar(pc, label="Elevation (m)")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.title("SRTM DEM with pcolormesh")
```