Python
CH13: ggplot2

전종준 교수

서울시립대학교 통계학과/통계데이터사이언스대학원

aes의 이해

강의 개요

- 목표: Grammar of Graphics 철학과 `plotnine` 문법 습득
- 데이터: **iris**, **mtcars**, **California housing**
- 산점도/막대/박스/히스토그램/스무딩/Facet/Grid/주석/저장
- 결과물: 데이터로 “스토리”를 만드는 리포트용 그래프

Grammar of Graphics 핵심

- **데이터(Data)**: `pandas DataFrame` 형식, 각 열은 변수, 각 행은 관측치
- **미적 매핑(aes)**: 변수 → 시각 속성(축, 색상, 모양, 크기 등) 매핑 예) `aes(x='hp', y='mpg', color='factor(cyl)')`
- **기하 객체(geom)**: 데이터 표현 방식 예) 점(`geom_point`), 선(`geom_line`), 막대(`geom_bar`)
- **통계 변환(stat)**: 데이터를 그래프용으로 변환 예) 빈도 계산(`stat_count`), 회귀선(`stat_smooth`)
- **위치(position)**: 겹침 방지, 막대 위치 조정 예) `position_dodge`, `position_stack`
- **스케일(scale)**: 데이터 값 → 화면 값 변환 예) 색상 팔레트, 로그 변환, 범례 라벨 지정
- **테마(theme)**: 축/격자/폰트/배경 등 스타일 제어
- **Facet**: 하나의 플롯을 여러 소그룹으로 분할 예) `facet_wrap(' Species')`, `facet_grid('gear cyl')`

실습 데이터셋 소개

- **iris** (앤더슨, 피셔 데이터)
  - 변수: `Sepal_Length`, `Sepal_Width`, `Petal_Length`, `Petal_Width`, `Species`
  - 3종의 붓꽃(Setosa, Versicolor, Virginica), 각 종별 50개 관측치
  - **활용:** 품종별 길이·너비 관계 시각화, 분류 경향 파악
- **mtcars** (Motor Trend 잡지 1974년 데이터)
  - 변수: `mpg`, `cyl`, `disp`, `hp`, `drat`, `wt`, `qsec`, `gear`, `carb` 등
  - 자동차 32종의 연비, 엔진 크기, 기어 수, 카뷰레터 수 등 특성
  - **활용:** 마력-연비 관계, 실린더·기어별 성능 비교
- **California housing** (미국 인구조사 기반, 1990년)
  - 변수: `MedInc`, `HouseAge`, `AveRooms`, `AveOccup`, `Latitude`, `Longitude`, `MedHouseVal` 등
  - 가구 단위의 중위 소득, 평균 방 개수, 위도·경도, 중위 주택가치 포함
  - **활용:** 소득-주택가치 상관 분석, 지역별 가격 분포 시각화

환경 준비

```
# 설치 (필요 시)
# pip install plotnine pandas numpy scikit-learn statsmodels

import pandas as pd
import numpy as np
from plotnine import *
import statsmodels.api as sm
from sklearn.datasets import fetch_california_housing
```

첫 코드: ggplot 기본 골격

```
# iris 데이터 불러오기
# seaborn을 쓰지 않고 공개 CSV를 사용하거나, 예제로 직접 구성 가능하지만
# 여기서는 statsmodels의 R 데이터셋 경로 활용
iris = sm.datasets.get_rdataset('iris').data
# 컬럼명 통일(대문자 -> 밑줄)
iris.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']

# ggplot 기본 골격: ggplot(data, aes(...)) + geom_*
p = (ggplot(iris, aes('Sepal_Length', 'Petal_Length', color='Species'))
      + geom_point()
      + labs(title='Iris: Sepal vs Petal Length', x='Sepal Length', y='Petal Length'))
p
```

aes() 매핑 개념

- `x`, `y`: 필수 축 매핑
- `color`, `fill`, `size`, `shape`, `alpha`: 시각 속성 매핑
- 데이터 컬럼명을 문자열로 지정: `aes('x', 'y', color='col')`

aes() 시각 속성 — color

- 점, 선, 텍스트 등의 **윤곽선 색상**을 데이터 값에 따라 매핑
- 연속형 → 그라데이션, 범주형 → 색상 팔레트로 표현
- geom_point, geom_line, geom_text 등 대부분에서 사용 가능

```
from plotnine import *
import statsmodels.api as sm

iris = sm.datasets.get_rdataset('iris').data
iris.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']

(ggplot(iris, aes('Sepal_Length', 'Petal_Length', color='Species'))
 + geom_point(size=2)
 + labs(title='color: 품종별 점 색상 매핑'))
```

aes() 시각 속성 — fill

- 막대, 박스플롯, 도형의 **채움 색상**을 데이터 값에 따라 매핑
- 범주형 데이터로 그룹 색을 쉽게 표현 가능
- geom_bar, geom_boxplot, geom_area 등에서 주로 사용

```
(ggplot(iris, aes('Species', 'Petal_Length', fill='Species'))
 + geom_boxplot()
 + labs(title='fill: 품종별 채움 색상 매핑'))
```

aes() 시각 속성 — size

- 점이나 선의 **크기**를 데이터 값에 비례시켜 표현
- 연속형 → 크기 비례, 범주형 → 그룹별 크기 매핑 가능
- 시각적 비중 차이를 줄 때 유용

```
(ggplot(iris, aes('Sepal_Length', 'Petal_Length',
                  size='Sepal_Width', color='Species'))
 + geom_point(alpha=0.7)
 + labs(title='size: 꽃받침 너비에 따른 점 크기 변화'))
```

aes() 시각 속성 — shape

- 점의 **모양**을 데이터 값에 따라 구분
- 범주형 변수에서 그룹을 구분하는 데 효과적
- 색상 구분이 어려운 경우, shape로 차별화 가능

```
(ggplot(iris, aes('Sepal_Length', 'Petal_Length',
                  shape='Species', color='Species'))
 + geom_point(size=2)
 + labs(title='shape: 품종별 점 모양 구분'))
```

aes() 시각 속성 — alpha

- 점, 선, 면의 **투명도**를 데이터 값에 따라 매핑
- 값 범위: 0(완전 투명) 1(불투명)
- 겹침이 많을 때 밀도나 분포를 보기 쉽게 함

```
(ggplot(iris, aes('Sepal_Length', 'Petal_Length',
                  alpha='Sepal_Width', color='Species'))
 + geom_point(size=2)
 + labs(title='alpha: 꽃받침 너비에 따른 투명도 변화'))
```

geom_methods

geom_point() — iris

```
(ggplot(iris, aes('Sepal_Length','Petal_Length', color='Species'))
 + geom_point(alpha=0.8)
 + labs(title='산점도: 종별 패턴 관찰')
)
```

- **데이터:** `iris` — 붓꽃 품종별 꽃받침 길이(Sepal_Length)와 꽃잎 길이(Petal_Length)
- **매핑:**
  - `x` = 꽃받침 길이
  - `y` = 꽃잎 길이
  - `color` = 품종(Species) — 범주형 색상 구분
- **geom_point:** 각 관측치를 점으로 표시
- **alpha=0.8:** 점의 투명도를 80%로 설정 → 겹침 시 밀도 파악 용이

geom_line() — mtcars 예시

```
mtcars = sm.datasets.get_rdataset('mtcars').data.reset_index()
mtcars.rename(columns={'index':'model'}, inplace=True)
# 임의 정렬 인덱스를 "시간" 축처럼 사용
mtcars['idx'] = range(len(mtcars))
(ggplot(mtcars, aes('idx','mpg', group=1))
  + geom_line() + geom_point()
  + labs(title='mtcars mpg (임의 인덱스 순서)', x='Index', y='MPG'))
```

geom_bar() — housing 구간 빈도

```
cal = fetch_california_housing(as_frame=True).frame
# 주택가치 구간화
cal['MedHouseVal_bin'] = pd.qcut(cal['MedHouseVal'], q=10)
(ggplot(cal, aes('MedHouseVal_bin'))
 + geom_bar()
 + labs(title='주택가치 분위수 구간 빈도', x='MedHouseVal bin', y='Count')
 + theme(axis_text_x=element_text(rotation=45, ha='right')))
```

geom_bar() — housing 구간 빈도 (설명)

- **데이터:** `California housing` — 캘리포니아 지역별 주택 데이터(1990년 인구조사 기반)
- **전처리:**
  - `MedHouseVal` = 중위 주택 가치(단위: $100,000)
  - `pd.qcut(..., q=10)` — 전체 데이터를 10개의 **분위수 구간**으로 나눔
  - 구간(bin)은 범주형 데이터로 변환되어 `x`축에 사용
- **geom_bar:** 각 구간(bin)에 속하는 관측치 개수(빈도) 표시
- **labs:** - `x` = 주택가치 구간 - `y` = 빈도(Count)
- **theme:** x축 라벨이 겹치지 않도록 45도 회전
- **결과 해석:** - 특정 가격 구간(중하위·중위권)에 주택이 몰려 있음 - 상위 가격대(오른쪽 구간)는 상대적으로 관측치가 적음 → 주택가 분포의 비대칭성 확인

geom_histogram() — 분포 이해

```
(ggplot(cal, aes('MedHouseVal'))
 + geom_histogram(bins=40)
 + labs(title='주택가치 분포', x='Median House Value'))
```

```
  # 상대빈도 히스토그램
  (ggplot(cal, aes('MedHouseVal', y=after_stat('density')))
   + geom_histogram(bins=40, fill='skyblue', color='black', alpha=0.7)
  + labs(title='주택가치 분포 (상대빈도)', x='Median House Value', y='Density'))
```

geom_boxplot() — 그룹 비교

```
(ggplot(iris, aes('Species', 'Petal_Length', fill='Species'))
 + geom_boxplot()
 + labs(title='품종별 꽃잎 길이 비교'))
```

첫 등장: geom 구조와 파라미터

- 공통 구조: `geom_xxx(mapping=None, data=None, stat, position, ...)`
- `mapping`은 상위 `aes`를 덮어쓸 수 있음
- `alpha`, `size`, `color`, `fill` 등 미적 속성

레이어 개념과 중첩

```
(ggplot(iris, aes('Sepal_Length','Petal_Length', color='Species'))
 + geom_point()
 + geom_smooth(se=False)  # 추세형태만 확인
 + labs(title='레이어 중첩: 점 + 스무딩(선만)'))
```

Visualization with statistics

geom_smooth() — 추세선과 신뢰구간

- **역할:** 데이터의 추세를 시각적으로 보여주는 **스무딩 곡선**을 그림
- **기본 동작:** x, y의 관계를 비모수 방식(loess) 또는 회귀(linear, glm 등)로 적합
- **주요 매개변수:**
  - `method`: 스무딩 방법 지정 (`'loess'`, `'lm'`, `'glm'` 등)
  - `se`: 신뢰구간 표시 여부 (기본 `True`)
  - `span`: loess 곡선의 민감도 조절 (0 1, 작을수록 곡선이 세부 변화를 더 반영)
  - `color`, `linetype`, `size`: 선 스타일
- **활용:** 산점도 위에 추세선을 겹쳐 데이터 패턴과 경향성을 직관적으로 파악

stat_smooth() — 회귀선

```
(ggplot(mtcars, aes('hp','mpg'))
 + geom_point()
 + stat_smooth(method='lm')  # 선형회귀선 + 신뢰구간
 + labs(title='마력 vs 연비: 선형회귀'))
```

stat_summary() — 요약 통계 시각화

- **역할:** 데이터의 요약 통계(평균, 중앙값, 표준편차 등)를 계산하여 시각적으로 표시
- **기본 원리:** `fun_y` (또는 `fun_data`)로 계산 함수 지정 후, 결과를 지정한 `geom` 형태로 그림
- **주요 매개변수:**
  - `fun_y`: y축 값을 계산하는 함수 (예: `np.mean`, `np.median`)
  - `fun_data`: 평균 ± 표준편차/표준오차 같이 여러 값을 반환하는 함수
  - `geom`: 출력 형태 지정 (예: `'point'`, `'line'`, `'bar'`, `'crossbar'`)
  - `color`, `size`, `width`: 스타일 조정
- **활용:** 그룹별 평균선, 요약 막대, 오차 막대 등 빠르게 표현 가능

stat_summary() — 요약선

```
(ggplot(mtcars, aes('factor(cyl)','mpg'))
 + geom_point(position=position_jitter(width=0.1, height=0))
 + stat_summary(fun_y=np.mean, geom='crossbar', width=0.5)
 + labs(title='실린더 수별 MPG 평균(크로스바)'))
```

색상/채우기 스케일

```
(ggplot(iris, aes('Sepal_Length','Petal_Length', color='Species'))
 + geom_point(size=2)
 + scale_color_manual(values=['#1b9e77','#d95f02','#7570b3'])
 + labs(title='수동 색상 스케일 예'))
```

변환과 범례 수정

```
(ggplot(mtcars, aes('hp','mpg', color='factor(gear)'))
 + geom_point()
 + scale_x_log10()
 + labs(color='Gear', title='x축 로그 변환 + 범례 제목 변경'))
```

테마: 기본 스타일

```
(ggplot(iris, aes('Sepal_Length','Petal_Length', color='Species'))
 + geom_point()
 + theme_minimal()
 + labs(title='theme_minimal 예시'))
```

theme() 세부 조정

```
(ggplot(iris, aes('Sepal_Length','Petal_Length', color='Species'))
 + geom_point()
 + theme(
     legend_position='bottom',
     axis_title=element_text(size=12),
     plot_title=element_text(weight='bold'))
 + labs(title='테마 세부 커스터마이즈'))
```

facet_wrap() — iris 품종별

```
(ggplot(iris, aes('Sepal_Length','Sepal_Width'))
 + geom_point()
 + facet_wrap('~Species', ncol=3)
 + labs(title='facet_wrap: 품종별 산점도'))
```

facet_grid() — mtcars 기어×실린더

```
(ggplot(mtcars, aes('hp','mpg'))
 + geom_point()
 + facet_grid('gear~cyl')   # 행=기어, 열=실린더
 + labs(title='facet_grid: 기어×실린더'))
```

Facet와 데이터 형식 요구사항

- Facet 변수는 **열로 존재**해야 함(범주형 권장)
- 결측/희소 카테고리 처리 필요
- **long vs wide** 형식 이해가 중요

long vs wide 형식 (개념)

- **wide**: 변수별 열이 가로로 펼쳐진 형태(예: 각 측정치가 별도 열)
- **long**: `variable, value` 열로 녹인 형태 — ggplot 계열에 사용
- Facet/색상/모양 등으로 변수 간 비교가 용이

pandas.melt() — wide → long 변환

- **역할:** 열(column) 형태로 펼쳐진 여러 변수를 **행(row)** 형태로 녹여(long format) 저장
- **구문:**

```
DataFrame.melt(id_vars=고정열, value_vars=변환대상열,
               var_name='새변수명', value_name='값변수명')
```

- **코드 해석:**
  - `id_vars=['Species']`: 변환하지 않고 그대로 유지할 열 (품종)
  - `value_vars=[...]`: long 형식으로 변환할 열 목록 (`Sepal_Length`, `Sepal_Width`, `Petal_Length`, `Petal_Width`)
  - `var_name='Measure'`: 원래 열 이름이 저장될 새 변수명
  - `value_name='Value'`: 각 측정값이 저장될 새 변수명
- **결과:** - `Species`, `Measure`, `Value` 3개의 열로 구성된 long format 데이터 - long format은 ggplot 계열의 facet, 색상, 모양 매핑에 유리

첫 코드: wide→long 변환 + facet 적용

```
# iris를 long으로 변환 (Sepal/Petal, Length/Width)
iris_long = (iris
  .melt(id_vars=['Species'],
        value_vars=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width'],
        var_name='Measure', value_name='Value'))
# 파생 변수: 부위, 길이/너비
iris_long['Part'] = iris_long['Measure'].str.split('_').str[0]    # Sepal/Petal
iris_long['Dim']  = iris_long['Measure'].str.split('_').str[1]    # Length/Width

(ggplot(iris_long, aes('Species','Value', fill='Species'))
 + geom_boxplot()
 + facet_grid('Part~Dim')  # 행=Sepal/Petal, 열=Length/Width
 + labs(title='iris wide→long 변환 후 facet_grid'))
```

groupby/summarize + 시각화

```
mt_sum = (mtcars.assign(cyl=mtcars['cyl'].astype(int))
                .groupby('cyl', as_index=False)['mpg'].mean())
(ggplot(mt_sum, aes('factor(cyl)','mpg'))
 + geom_col()
 + labs(title='실린더 수별 평균 MPG', x='cyl', y='mean MPG'))
```

housing: 소득 구간별 주택가치 평균

```
cal['MedInc_bin'] = pd.qcut(cal['MedInc'], q=10)
cal_mean = (cal.groupby('MedInc_bin', as_index=False)['MedHouseVal'].mean())
(ggplot(cal_mean, aes('MedInc_bin','MedHouseVal'))
 + geom_col()
 + theme(axis_text_x=element_text(rotation=45, ha='right'))
 + labs(title='소득 분위수별 중위 주택가치', x='소득 분위', y='중위 주택가치'))
```