데이터 시각화

안 화 수

데이터 시각화

- 데이터 시각화 방법
- matplotlib 그래프
- pandas 그래프
- seaborn 그래프

matplotlib 그래프

matplotlib 모듈

- **matplotlib 모듈**

matplotlib 모듈은 파이썬에서 데이터를 효과적으로 시각화하기 위해서 만든 라이버러리 이다.
matplotlib은 MATLAB(과학 및 공학 연산을 위한 소프트웨어)의 시각화 기능을 모델링해서 만들어졌다.
matplotlib 모듈을 이용하면 2차원 선 그래프(line), 산점도(scatter), 막대그래프(bar chart), 히스토그램(histogram), 파이 그래프(pie chart) 등을 그릴 수 있다.

matplotlib 모듈은 아나콘다가 설치 될 때 같이 설치 되므로, 따로 설치 하지 않아도 된다.

matplotlib 모듈

- matplotlib 모듈 설치

  c:₩> pip install matplotlib

- matplotlib 모듈 설치 확인

  c:₩> pip list

line graph

- **line_graph.py**

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4])
plt.show()
```

line graph

- **line_graph01.py**

```
import matplotlib.pyplot as plt

plt.plot([0,1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1,0])
plt.xlabel('x-axis')                    # x축 라벨
plt.ylabel('y-axis')                    # y축 라벨
plt.show()
```

line graph

- **x축, y축에 한글 라벨 설정**

- **line_graph_label.py**

```
import matplotlib.pyplot as plt
import matplotlib

x = [1,2,3,4,5]
y = [1,2,3,4,5]

# 한글 라벨 설정 :  '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

plt.xlabel('x축')        # x축 라벨 ( 한글 라벨 )
plt.ylabel('y축')        # y축 라벨
plt.plot(x,y)
plt.show()
```

line graph

- **line_graph02.py**

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5, 6]
y = [1, 4, 5, 8, 9, 5, 3]

plt.figure(figsize=(10,6))          # 크기 설정
plt.plot(x,y,color='green')         # 선색깔: green
plt.show()
```

line graph

- **line_graph03.py**

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [1,2,3,4,5]

# plt.plot(x,y)
plt.plot(x,y,'ro')                    # 'r-', 'g-', 'b-'
plt.show()                            # 'ro', 'go', 'bo'
                                      # 'rv', 'gv', 'bv'

# plot()은 b- 옵션이 기본값 : 파란색(b) 라인(-)이라는 뜻
# ro 옵션은 빨간색(r)으로 o표시를 의미함
# bv 옵션은 파란색(b)으로 v표시를 의미함
```

dashed line graph

- **점선 그래프 그리기**
  점선 : plot()함수에 linestyle='dashed' 추가

- **dashed_line_graph01.py**

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5, 6]
y = [1, 4, 5, 8, 9, 5, 3]

plt.figure(figsize=(10,6))
plt.plot(x,y,color='green', linestyle='dashed')
plt.show()
```

dashed line graph

- **점선 그래프 그리기**
  점선 : plot()함수에 linestyle='dashed' 추가
  마커 표시 : plot()함수에 marker = 'o' 추가

- **dashed_line_graph02.py**

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5, 6]
y = [1, 4, 5, 8, 9, 5, 3]

plt.figure(figsize=(10,6))
plt.plot(x,y,color='green', linestyle='dashed', marker='o')
plt.show()
```

dashed line graph

- **점선 그래프 그리기**

  점선 : plot()함수에 linestyle='dashed' 추가
  마커 표시 : plot()함수에 marker = 'o' 추가
  마커 색깔 : plot()함수에 markerfacecolor = 'red'
  마커 크기 : plot()함수에 markersize = 12

- **dashed_line_graph03.py**

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5, 6]
y = [1, 4, 5, 8, 9, 5, 3]

plt.figure(figsize=(10,6))
plt.plot(x,y,color='green', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=12)
plt.show()
```

산점도

산점도

산점도는 2개의 요소로 이루어진 데이터 집합의 관계 (예를들면, 키와 몸무게와의 관계, 공부시간과 시험점수와의 관계, 기온과 아이스크림 판매량과의 관계)를 시각화하는데 유용하다.

산점도를 그리기 위해서는 plot()함수 대신에 scatter()함수 사용함

산점도 형식

```
plt.scatter( x, y [, s = size, c = colors, marker = `marker_string `,
                    alpha = alpha_f ] )
```

s : 마커의 크기
c : 마커의 색깔
marker : 마커 모양
alpha : 투명도 : 0(완전 투명) ~ 1(완전 불투명)

옵션을 지정하지 않으면, 기본값 (s=40, c='b', marker='o', alpha=1)으로 지정

산점도

- **산점도**

  plot()함수 대신에 scatter()함수 사용함

- scatter_graph01.py

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

x = np.array([0,1,2,3,4,5,6,7,8,9])
y = np.array([9,8,7,9,8,3,2,4,3,4])

plt.figure(figsize=(10,6))
plt.scatter(x,y)
plt.show()
```

산점도

- **산점도**

  plot()함수 대신에 scatter()함수 사용함
  marker 의 모양 변경 : scatter()함수에 marker = '>' 추가
  marker의 크기 변경 : s = 50
  x의 값에 따라 y축 값의 색상을 바꾸는 colormap 추가
  scatter()함수에 c=colormap
  plt.colorbar()

- **scatter_graph02.py**

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

x = np.array([0,1,2,3,4,5,6,7,8,9])
y = np.array([9,8,7,9,8,3,2,4,3,4])

colormap = x

plt.figure(figsize=(10,6))
plt.scatter(x,y, s=50, c=colormap, marker='>')

plt.colorbar()
plt.show()
```

산점도

- **scatter_graph03.py (1 / 2)**

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

# '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

city = ['서울', '인천', '대전', '대구', '울산', '부산', '광주']

# 위도(latitude)와 경도(longitude)
lat  = [37.56, 37.45, 36.35, 35.87, 35.53, 35.18, 35.16]
lon = [126.97, 126.70, 127.38, 128.60, 129.31, 129.07, 126.85]

# 인구 밀도(명/km^2): 2017년 통계청 자료
pop_den = [16154, 2751, 2839, 2790, 1099, 4454, 2995]
```

산점도

- **scatter_graph03.py (2 / 2)**

```
size = np.array(pop_den) * 0.2                  # 마커의 크기 지정
colors = ['r', 'g', 'b', 'c', 'm', 'k', 'y']    # 마커의 컬러 지정

plt.scatter(lon, lat, s=size, c=colors, alpha=0.5)
plt.xlabel('경도(longitude)')
plt.ylabel('위도(latitude)')
plt.title('지역별 인구 밀도(2017)')

for x, y, name in zip(lon, lat, city):
    plt.text(x, y, name)                        # 위도 경도에 맞게 도시 이름 출력

plt.show()
```

막대 그래프

막대 그래프

막대그래프는 값을 막대의 높이를 나타내므로 여러 항목의 수량이 많고 적음을 한눈에 파악할 수 있다. 따라서 여러 항목의 데이터를 서로 비교할 때 주로 이용한다.

막대 그래프 형식

plt.bar(x, height [, width = width_f, color = colors, tick_label = tick_labels, align = 'center' (기본) , label = labels])

x : height와 길이가 일치하는 데이터로 x축에 표시될 위치를 지정
height : 시각화 하고자 하는 막대 그래프 데이터
width : [0, 1]사이의 실수를 지정해 막대의 폭을 조절
width 옵션을 입력하지 않으면 기본값인 0.8이 입력
color : fmt옵션의 컬러지정 약어로 막대그래프의 색을 지정
tick_label : 막대 그래프 x축의 tick 라벨 이름을 지정 (기본값은 숫자 라벨)
align : 막대 그래프의 위치를 가운데로 할지(center) 한쪽으로 치우치게 할지(edge) 설정 (기본값은 center)
label : 범례에 사용될 문자열을 지정

막대 그래프

막대 그래프 예제

헬스클럽 회원 4명이 운동을 시작하기 전에 측정한 윗몸 일으키기 횟수와 운동 한달 후에 측정한 윗몸 일으키기 횟수를 측정한 데이터를 막대 그래프로 시각화

<운동 하기 전과 운동 한 달 후의 윗몸일으키기 횟수>

회원 ID	운동 시작 전	운동 한 달 후
m_01	27	30
m_02	35	38
m_03	40	42
m_04	33	37

막대 그래프

- **막대 그래프 예제**

bar_graph.py (1 / 3)

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

# '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

member_IDs = ['m_01', 'm_02', 'm_03', 'm_04']      # 회원 ID
before_ex = [27, 35, 40, 33]                       # 운동 시작 전
after_ex = [30, 38, 42, 37]                        # 운동 한 달 후

# 1. 기본 값으로 막대 그래프 출력
n_data = len(member_IDs)        # 회원이 네 명이므로 전체 데이터 수는 4
index = np.arange(n_data)       # NumPy를 이용해 배열 생성 (0, 1, 2, 3)
plt.bar(index, before_ex)       # bar(x,y)에서 x=index, height=before_ex 로 지정
plt.show()
```

막대 그래프

- **막대 그래프 예제**

bar_graph.py (2 / 3)

```
# 2. tick_label : x축 tick 라벨 설정 (회원 ID로 설정)
plt.bar(index, before_ex, tick_label = member_IDs)
plt.show()

# 3. color : 막대 그래프의 색깔 설정
colors=['r', 'g', 'b', 'm']       # m : 자홍색(magenta)
plt.bar(index, before_ex, color = colors, tick_label = member_IDs)
plt.show()

# 4. width : 막대 그래프의 폭 설정 ( 기본값 : 0.8 )
plt.bar(index, before_ex, tick_label = member_IDs, width = 0.6)
plt.show()

# 5. plt.barh() :  수평 막대 그래프
colors=['r', 'g', 'b', 'm']
plt.barh(index, before_ex, color = colors, tick_label = member_IDs)
plt.show()
```

막대 그래프

- **막대 그래프 예제**

bar_graph.py (3 / 3)

```
# 6. 최종 그래프 출력
barWidth = 0.4      # c:청록색(cyan), m:자홍색(magenta)
plt.bar(index, before_ex, color='c', align='edge', width = barWidth,
        label='before')                                  # 운동전 그래프
plt.bar(index + barWidth, after_ex , color='m', align='edge',
        width = barWidth, label='after')            # 운동후 그래프

plt.xticks(index + barWidth, member_IDs)
plt.legend()
plt.xlabel('회원 ID')
plt.ylabel('윗몸일으키기 횟수')
plt.title('운동 시작 전과 후의 근지구력(복근) 변화 비교')
plt.show()
```

히스토그램

히스토그램

히스토그램(histogram)은 데이터를 정해진 간격으로 나눈 후 그 간격 안에 들어간 데이터 개수를 막대로 표시한 그래프로, 데이터가 어떤 분포를 갖는지를 볼 때 주로 사용한다.
즉, 히스토그램은 도수 분포표를 막대 그래프로 시각화한 것이다.
히스토그램은 주로 통계 분야에서 데이터가 어떻게 분포하는지 알아볼 때 많이 사용한다.

히스토그램 형식

plt.hist (x [,bins = bins_n 혹은 'auto'])

x : 변량 데이터
bins : 계급의 개수 (기본값은 10)
bins = 'auto' 가 입력되면, x에 맞게 자동으로 bins에 값이 들어간다.

도수 분포표

❖ 도수 분포표의 용어

- **변량(variate) :** 자료를 측정해 숫자로 표시한 것
  ex) 점수, 키, 몸무게, 판매량, 시간 등

- **계급(class) :** 변량을 정해진 간격으로 나눈 구간
  ex) 시험점수를 60~70, 70~80, 80~90, 90~100점 구간으로 나눔

- **계급의 간격(class width)** : 계급을 나눈 크기 (기본값은 10)

- **도수(frequency) :** 나눠진 계급에 속하는 변량의 수
  ex) 각 계급에서 발생한 수

- **도수 분포표(frequency distribution table) :** 계급에 도수를 표시한 것

도수 분포표

도수 분포표 예제

히스토그램을 그리기 위해서는 우선 도수 분포표에 대해서 이해해야 한다.
어떤 학급에서 수학 시험 결과를 이용해 도수 분포표를 만드는 과정

1. 변량 생성

학생 25명의 수학 시험 결과

76, 82, 84, 83, 90, 86, 85, 92, 72, 71, 100, 87, 81, 76, 94, 78, 81, 60, 79, 69, 74, 87, 82, 68, 79

2. 계급 간격 설정 및 계급 생성 – 계급간격 : 5, 계급 : 8개

변량 중 가장 작은 숫자가 60이고, 가장 큰 숫자가 100이므로 60에서 100까지 일정 간격(여기서는 5로 설정)으로 나눠서 8개의 계급으로 설정

60~65, 65~70, 70~75, 75~80, 80~85, 85~90, 90~95, 95~100

도수 분포표

❖ 도수 분포표 예제

3. 계급별 도수 확인 및 도수 분포표 만들기

각 계급에 몇 개의 변량이 들어 있는지를 확인해 계급별로 도수를 구한다.
각 도수를 시각적으로 쉽게 볼 수 있도록 마크(O)로 도수를 표시할 수 있다.

<도수 분포표>

계급	도수	마크로 표시
60(이상) ~ 65(미만)	1	O
65 ~ 70	2	OO
70 ~ 75	3	OOO
75 ~ 80	5	OOOOO
80 ~ 85	6	OOOOOO
85 ~ 90	4	OOOO
90 ~ 95	3	OOO
95 ~ 100	1	O

히스토그램

- **히스토그램 예제**
  25명 학생들의 수학점수를 히스토그램으로 시각화

histogram.py

```
import matplotlib.pyplot as plt
import matplotlib

# '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

# 25명 학생들의 수학성적
math = [76, 82, 84, 83, 90, 86, 85, 92, 72, 71, 100, 87, 81, 76, 94, 78, 81, 60, 79, 69, 74, 87, 82,
        68, 79]

# 기본적으로 변량을 10개의 계급으로 나눠서 출력
plt.hist(math)
plt.show()

# 60에서 100까지 5간격으로 8개의 계급으로 나눠서 출력
plt.hist(math, bins= 8)
plt.xlabel('시험 점수')
plt.ylabel('도수(frequency)')
plt.title('수학 시험의 히스토그램')
plt.grid()
plt.show()
```

파이 그래프

- **파이 그래프**

파이(pie) 그래프는 원안에 데이터의 각 항목이 차지하는 비율만큼 부채꼴의 크기를 갖는 영역으로 이루어진 그래프이다.
파이 그래프는 전체 데이터에서 각 항목이 차지한 비율을 비교할 때 많이 사용한다.

파이 그래프

- **파이 그래프 형식**

```
plt.pie(x [, labels = label_seq ,                    #부채꼴에 출력되는 문자열
          autopct = '비율 표시 형식(ex: %.1f)' ,
          shadow = False(기본) 혹은 True ,           # 그림자효과
          explode = explode_seq ,                    # 부채꼴부분이 원에서 돌출
          counterclock = True(기본) 혹은 False ,     #부채꼴이 그려지는 순서
                                                     #기본값은 반시계방향(True)
          startangle = 각도 (기본은 0 )              #처음 부채꼴부분이 그려지는
                                                     # 각도(기본값은 0)
```

파이 그래프

- **파이 그래프 예제**

pie_graph.py (1 / 2)

```
import matplotlib.pyplot as plt
import matplotlib

# '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

fruit = ['사과', '바나나', '딸기', '오렌지', '포도']
result = [7, 6, 3, 2, 2]

# 1.파이 그래프가 타원모양으로 출력
plt.pie(result)
plt.show()

# 2.파이 그래프의 너비와 높이를 1대 1로 출력(원모양) : figsize=(5,5)
plt.figure(figsize=(5,5))
plt.pie(result)
plt.show()
```

파이 그래프

- **파이 그래프 예제**

pie_graph.py (2 / 2)

```
# 3.라벨과 비율 추가
plt.figure(figsize=(5,5))
plt.pie(result, labels= fruit, autopct='%.1f%%')
plt.show()

# 4.각도 90도에서 시작해서 시계방향으로 설정
plt.figure(figsize=(5,5))
plt.pie(result, labels= fruit, autopct='%.1f%%', startangle=90, counterclock = False)
plt.show()

# 5.그림자 효과 추가하고 사과 부채꼴만 강조
explode_value = (0.1, 0, 0, 0, 0)
plt.figure(figsize=(5,5))
plt.pie(result, labels= fruit, autopct='%.1f%%', startangle=90, counterclock = False,
                    explode=explode_value, shadow=True)
plt.show()
```

pandas 그래프

pandas 그래프

- **pandas 그래프 구조**

  pandas의 Series나 DataFrame으로 생성한 데이터가 있을 경우에는
  pandas 모듈로 그래프를 그릴 수 있다.

  Series_data . plot([kind = ‘ graph_kind ’] [,option])

  DataFrame_data . plot([x=label 혹은 potion , y=label 혹은 potions ,]
  [kind = ‘ graph_kind ‘] [, option])

  ex)
  Series_data . plot (kind = ‘ line ‘) : 선그래프

  kind 옵션을 생략하면 기본적으로 선 그래프가 그려진다.

pandas 그래프

- pandas 그래프 종류

kind 옵션	의 미
line	선 그래프(기본)
bar	수직 막대 그래프
barh	수평 막대 그래프
scatter	산점도(DataFrame 데이터만 가능)
hist	히스토그램
pie	파이 그래프

pandas 그래프

- pandas 그래프 그리기

DataFrame_data . plot(kind = ' line ')
DataFrame_data . plot . line()

DataFrame_data . plot(kind = ' bar ')
DataFrame_data . plot . bar()

DataFrame_data . plot(kind = ' pie ')
DataFrame_data . plot . pie()

DataFrame_data . plot . hist()

DataFrame_data . plot . scatter()

pandas 선 그래프

❖ **pandas 선그래프**

pandas_line01.py

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

s = pd.Series([1,2,3,4,5,6,7,8,9,10])
print(s)

s.plot()
plt.show()
```

pandas 선 그래프

❖ **pandas 선그래프**

pandas_line02.py

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

# x축에 index값 날짜출력, y축에 Series 데이터 출력
s = pd.Series([1,2,3,4,5,6,7,8,9,10],
                    index = pd.date_range('2019-01-01', periods=10))
print(s)

s.plot()
plt.show()

s.plot(grid=True)            # 격자모양 추가함
plt.show()
```

pandas 막대 그래프

- **pandas_bar01.py**

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


# 데이터 세트 만들기 (3개의 난수로 이루어진 10개의 배열)
data_set = np.random.rand(10,3)
print(data_set)

# pandas의 데이터프레임(DataFrame) 생성
df = pd.DataFrame(data_set, columns = ['A','B','C'])
print(df)

# 수직 막대 그래프 그리기
df.plot(kind='bar')
plt.show()

# 수평 막대 그래프 그리기 (horizontal)
df.plot(kind='barh')
plt.show()

# 영역 그래프 그리기
df.plot(kind='area', stacked=False)
plt.show()
```

pandas 막대 그래프

- **pandas_bar02.py**

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

# '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

grade_num = [5, 14, 12, 3]
students = ['A', 'B', 'C', 'D']

# columns 값 Student 가 범례로 출력
df_grade = pd.DataFrame(grade_num, index=students, columns = ['Student'])
print(df_grade)

grade_bar = df_grade.plot.bar(grid = True)
grade_bar.set_xlabel("학점")
grade_bar.set_ylabel("학생수")
grade_bar.set_title("학점별 학생 수 막대 그래프")
plt.show()
```

pandas 산점도

❖ **pandas_scatter.py**

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

# '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

temperature = [25.2, 27.4, 22.9, 26.2, 29.5, 33.1, 30.4, 36.1, 34.4, 29.1]
Ice_cream_sales = [236500, 357500, 203500, 365200, 446600, 574200,
                   453200, 675400, 598400, 463100]

dict_data = {'기온' : temperature,  '아이스크림 판매량' : Ice_cream_sales}
df_ice_cream = pd.DataFrame(dict_data, columns=['기온', '아이스크림 판매량'])
print(df_ice_cream)

df_ice_cream.plot.scatter(x='기온', y='아이스크림 판매량',
                          grid=True, title='최고 기온과 아이스크림 판매량')
plt.show()
```

pandas 히스토그램

- **pandas_hist.py**
  25명 학생들의 수학점수를 히스토그램으로 시각화

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

# '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

math = [76,82,84,83,90,86,85,92,72,71,100,87,81,76,94,78,81,60,79,69,74,87,82,68,79]
df_math = pd.DataFrame(math, columns = ['Student'])

# 옵션 bins는 계급의 갯수를 의미하며, 기본값은 10이다.
math_hist = df_math.plot.hist(bins=8, grid = True)
math_hist.set_xlabel("시험 점수")
math_hist.set_ylabel("도수(frequency)")
math_hist.set_title("수학 시험의 히스토그램")

plt.show()
```

pandas 파이 그래프

- **pandas_pie01.py**

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


# 5개의 난수로 이루어진 데이터 생성
data = np.random.rand(5)
print(data)

# pandas 시리즈형 데이터로 변환
s = pd.Series(data, index=['a','b','c','d','e'], name='series')
print(s)

# pie 그래프 출력
s.plot(kind='pie', autopct='%.2f', figsize=(7,7))
plt.show()
```

pandas 파이 그래프

- **pandas_pie02.py (1 / 2)**

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

# '맑은 고딕'으로 설정
matplotlib.rcParams['font.family'] = 'Malgun Gothic'

fruit = ['사과', '바나나', '딸기', '오렌지', '포도']
result = [7, 6, 3, 2, 2]

df_fruit = pd.Series(result, index = fruit, name = '선택한 학생수')
print(df_fruit)

df_fruit.plot.pie()
plt.show()
```

pandas 파이 그래프

- **pandas_pie02.py (2 / 2)**

```
explode_value = (0.1, 0, 0, 0, 0)                    # pie 간격설정
fruit_pie = df_fruit.plot.pie(figsize=(5, 5), autopct='%.1f%%',
                              startangle=90, counterclock = False,
                              explode=explode_value, shadow=True, table=True)
fruit_pie.set_ylabel("")                              # 불필요한 y축 라벨 제거
fruit_pie.set_title("과일 선호도 조사 결과")

# 그래프를 이미지 파일로 저장. dpi는 200으로 설정
plt.savefig('saveFruit.png', dpi = 200)
plt.show()
```

seaborn 그래프

Seaborn

- **seaborn 모듈**

  seaborn은 matplotlib의 기능과 스타일을 확장한 파이썬 시각화 도구의 고급 버전이다. 그리고 matplotlib으로 표현하기 힘든, 좀더 정교한 그래프를 지원하고 있다.

- seaborn으로 표현 가능한 그래프
  - 산점도
  - 히스토그램
  - 커널 밀도 그래프
  - 히트맵
  - 막대 그래프
  - 빈도 그래프
  - 박스 플롯
  - 바이올린 그래프
  - 조인트 그래프

Titanic Dataset

Titanic Dataset

seaborn 라이브러리에서 제공되는 'titanic' 데이터셋을 가져와서 다양한 그래프로 출력 해보자.

```
import   seaborn   as   sns

titanic = sns.load_dataset('titanic')
```

titanic 데이터셋에는 탑승객 891명의 정보가 담겨져 있다.

index : 891 (0 ~ 890)
columns : 15 columns

Titanic Dataset

- **Titanic Dataset**

seaborn_dataset.py

```
# 라이브러리 불러오기
import seaborn as sns

# titanic 데이터셋 가져오기
titanic = sns.load_dataset('titanic')
print(titanic)                                  # [891 rows x 15 columns]

# 출력할 행.열의 개수 늘리기 : 15개 컬럼, 891행 데이터 출력
pd.set_option('display.max_columns', 15)
pd.set_option('display.max_rows', 891)
print(titanic)

# titanic 데이터셋 살펴보기
print(titanic.head())                           # 앞에서 5개의 데이터 출력
print('₩n')
print(titanic.info())                           # 자료형 확인
```

Titanic Dataset

- **Titanic Dataset**

	survived	pclass	sex	age	...	deck	embark_town	alive	alone
0	0	3	male	22.0	...	NaN	Southampton	no	False
1	1	1	female	38.0	...	C	Cherbourg	yes	False
2	1	3	female	26.0	...	NaN	Southampton	yes	True
3	1	1	female	35.0	...	C	Southampton	yes	False
4	0	3	male	35.0	...	NaN	Southampton	no	True
..	...	...	...	...	...	...	...	...	...
886	0	2	male	...		NaN	Southampton	no	True
887	1	1	female	...		B	Southampton	yes	True
888	0	3	female	...		NaN	Southampton	no	False
889	1	1	male	...		C	Cherbourg	yes	True
890	0	3	male	...		NaN	Queenstown	no	True

[891 rows x 15 columns]

산점도

- **회귀선이 있는 산점도**

seaborn 모듈로 산점도를 그리기 위해서는 regplot() 함수를 이용한다.

```
sns.regplot ( x = 'age' ,          # x축 변수
              y = 'fare' ,         # y축 변수
              data = titanic ,     # 데이터
              ax = ax1 ,           # 1번째 그래프
              fit_reg = True )     # 회귀선 표시
```

산점도

- **회귀선이 있는 산점도**

seaborn_regplot.py (1/2)

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('darkgrid')

# 그래프 객체 생성 (figure에 2개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 5))          # 그래프 크기 설정
ax1 = fig.add_subplot(1, 2, 1)             # 1행 2열 - 1번째 그래프
ax2 = fig.add_subplot(1, 2, 2)             # 1행 2열 - 2번째 그래프
```

산점도

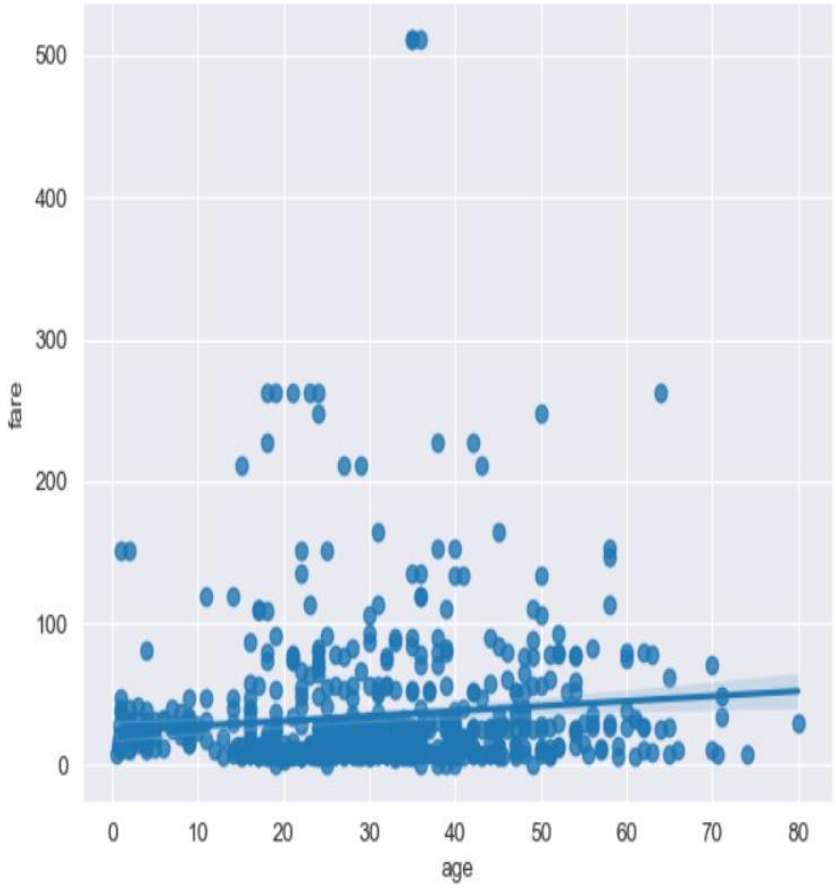
- 회귀선이 있는 산점도

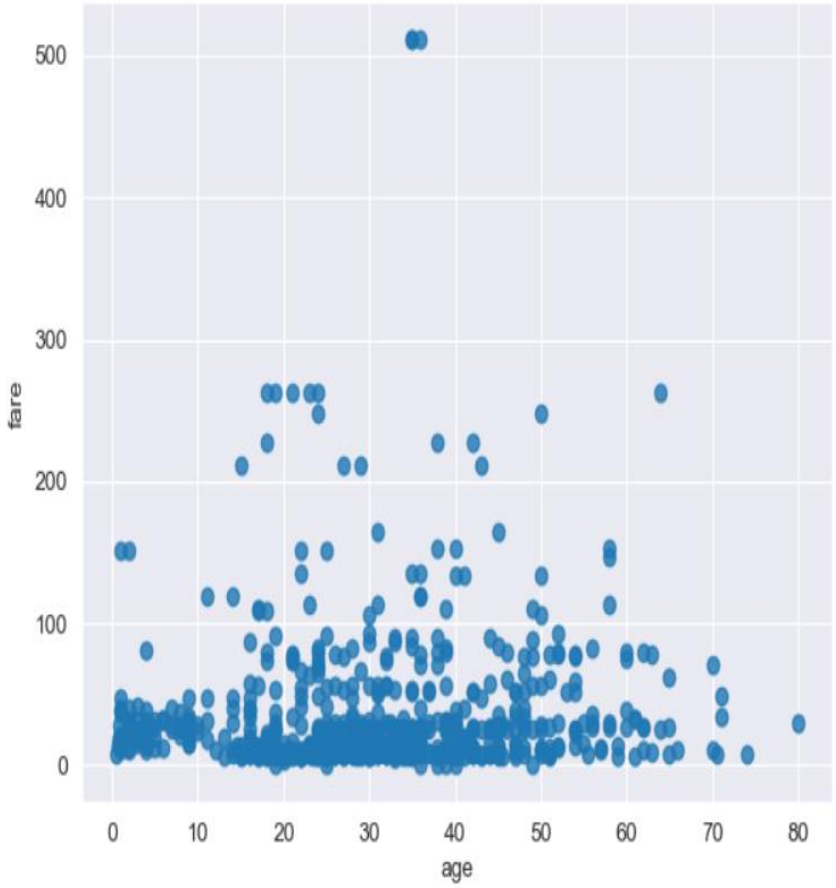
seaborn_regplot.py (2/2)

```
# 산점도 그래프 그리기 - 선형회귀선 표시(fit_reg=True)
sns.regplot(x='age',                    # x축 변수
            y='fare',                   # y축 변수
            data=titanic,               # 데이터
            ax=ax1,                     # axe 객체 - 1번째 그래프
            fit_reg=True)               # 회귀선 표시 (기본값)

# 산점도 그래프 그리기 - 선형회귀선 미표시(fit_reg=False)
sns.regplot(x='age',                    # x축 변수
            y='fare',                   # y축 변수
            data=titanic,               # 데이터
            ax=ax2,                     # axe 객체 - 2번째 그래프
            fit_reg=False)              # 회귀선 미표시

plt.show()
```

산점도

- 회귀선이 있는 산점도





산점도

범주형 데이터의 산점도

범주형 변수(class)에 들어 있는 각 범주별 데이터의 분포를 시각화하는 방법이다. 범주형 데이터의 산점도를 그릴때에는 stripplot() 함수와 swarmplot() 함수를 사용한다. swarmplot()함수는 데이터의 분산까지 고려하고, 데이터 포인트가 서로 중복되지 않도록 그래프를 그린다.
즉, 데이터가 퍼져있는 정도를 입체적으로 볼 수 있다.

```
 # 이산형 변수의 분포 - 데이터 분산 미고려
sns.stripplot(x="class",                    # x축 변수
              y="age",                      # y축 변수
              data=titanic,                 # 데이터셋 - 데이터프레임
              ax=ax1)                       # axe 객체 - 1번째 그래프

# 이산형 변수의 분포 - 데이터 분산 고려 (중복 X)
sns.swarmplot(x="class",                    # x축 변수
              y="age",                      # y축 변수
              data=titanic,                 # 데이터셋 - 데이터프레임
              ax=ax2)                       # axe 객체 - 2번째 그래프
```

산점도

- **범주형 데이터의 산점도**

seaborn_scatter.py (1/2)

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('whitegrid')

# 그래프 객체 생성 (figure에 2개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 5))          # 그래프 크기 설정
ax1 = fig.add_subplot(1, 2, 1)             # 1행 2열 - 1번째 그래프
ax2 = fig.add_subplot(1, 2, 2)             # 1행 2열 - 2번째 그래프
```

산점도

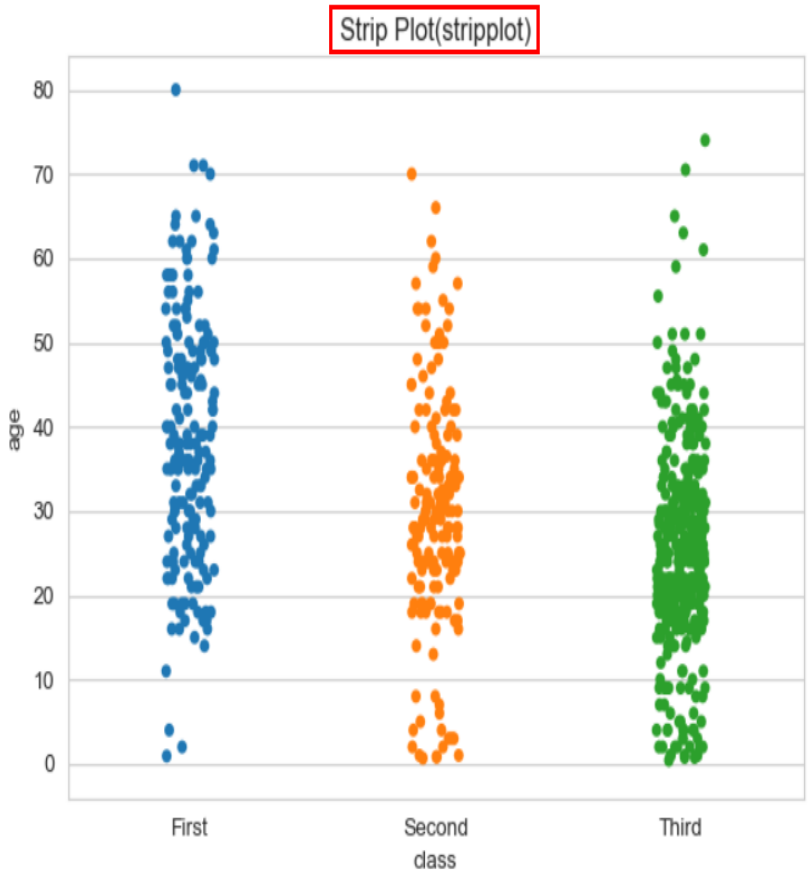
- **범주형 데이터의 산점도**

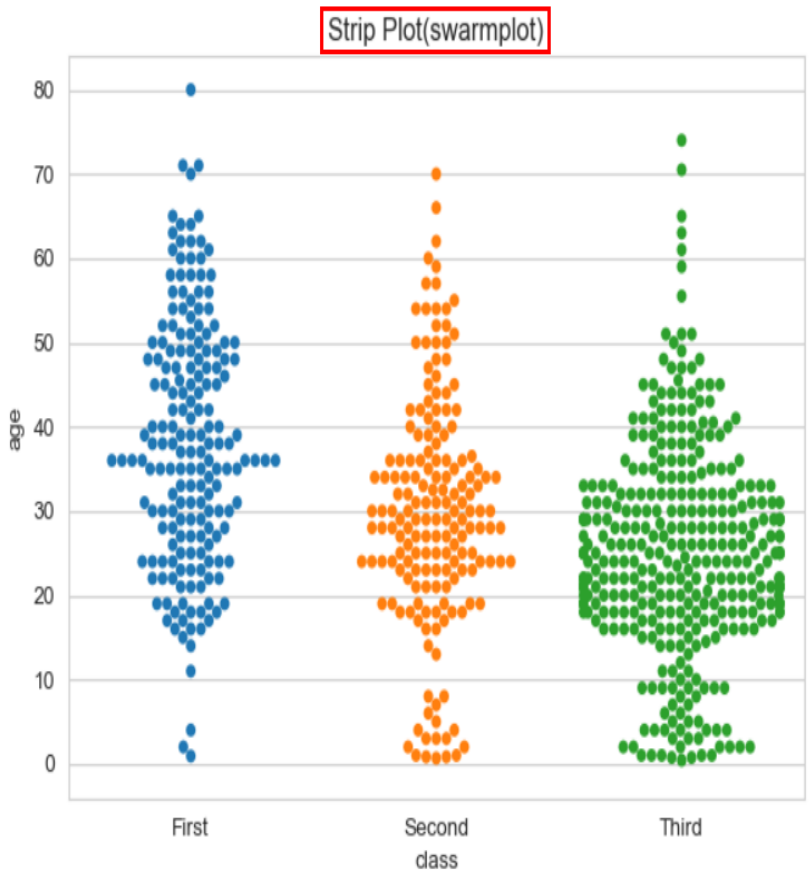
seaborn_scatter.py (2/2)

```
# 이산형 변수의 분포 - 데이터 분산 미고려
sns.stripplot(x="class",                    # x축 변수
              y="age",                      # y축 변수
              data=titanic,                 # 데이터셋 - 데이터프레임
              ax=ax1)                       # axe 객체 - 1번째 그래프

# 이산형 변수의 분포 - 데이터 분산 고려 (중복 X)
sns.swarmplot(x="class",                    # x축 변수
              y="age",                      # y축 변수
              data=titanic,                 # 데이터셋 - 데이터프레임
              ax=ax2)                       # axe 객체 - 2번째 그래프

# 차트 제목 표시
ax1.set_title('stripplot')
ax2.set_title('swarmplot')

plt.show()
```

산점도

- 범주형 데이터의 산점도





히스토그램

히스토그램과 커널 밀도 함수

하나의 변수 데이터의 분포를 확인할 때, 주로 히스토그램과 커널 밀도 함수로 시각화하며, seaborn 모듈로 히스토그램과 커널 밀도 함수를 그리기 위해서는 distplot() 함수를 이용한다.

커널 밀도 함수는 그래프와 x축 사이의 면적이 1이 되도록 그리는 밀도 분포함수이다.

```
# 기본값 : 히스토그램 + 커널밀도함수
sns.distplot(titanic['fare'], ax=ax1)

# hist=False : 커널밀도함수
sns.distplot(titanic['fare'], hist=False, ax=ax2)

# kde=False : 히스토그램
sns.distplot(titanic['fare'], kde=False, ax=ax3)
```

히스토그램

- **히스토그램과 커널 밀도 함수**

seaborn_distplot.py (1/2)

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('darkgrid')

# 그래프 객체 생성 (figure에 3개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 5))                    # 그래프 크기 설정
ax1 = fig.add_subplot(1, 3, 1)                       # 1행 3열 - 1번째 그래프
ax2 = fig.add_subplot(1, 3, 2)                       # 1행 3열 - 2번째 그래프
ax3 = fig.add_subplot(1, 3, 3)                       # 1행 3열 - 3번째 그래프
```

히스토그램

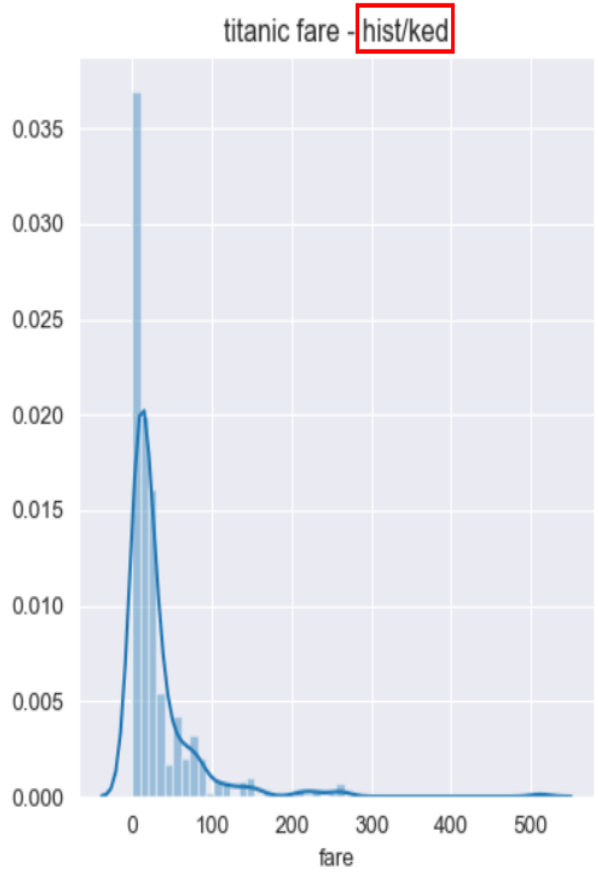
- **히스토그램과 커널 밀도 함수**

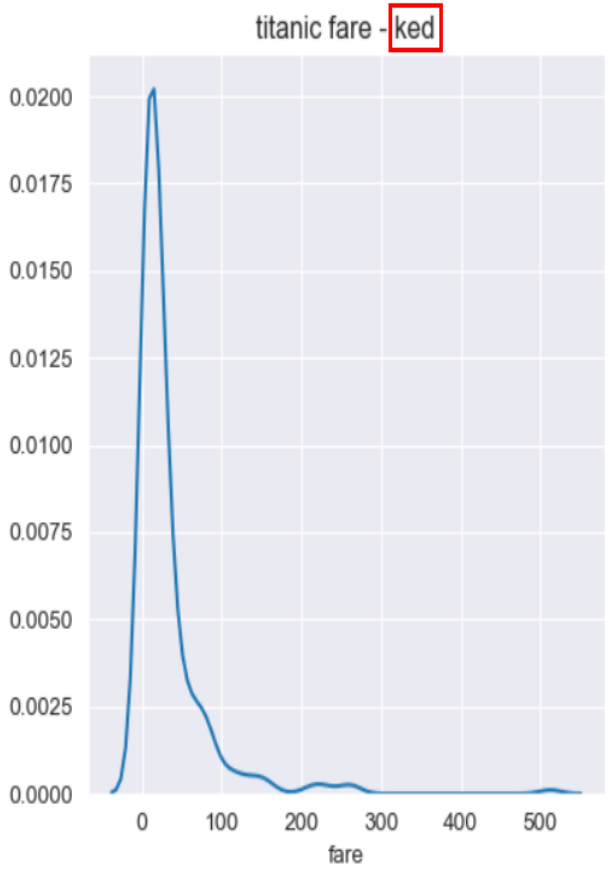
seaborn_distplot.py (2/2)

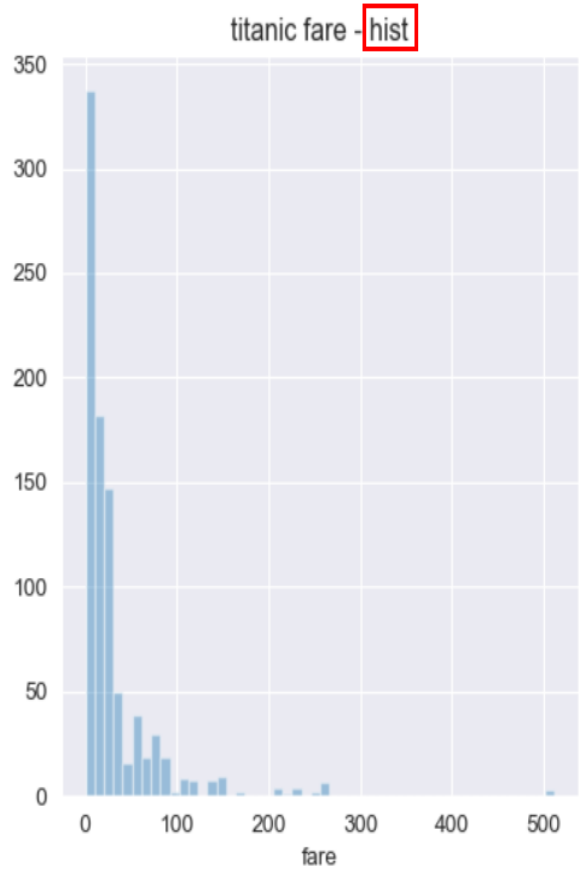
```
# 기본값 : 히스토그램 + 커널밀도함수
sns.distplot(titanic['fare'], ax=ax1)

# hist=False : 커널밀도함수
sns.distplot(titanic['fare'], hist=False, ax=ax2)

# kde=False : 히스토그램
sns.distplot(titanic['fare'], kde=False, ax=ax3)

# 차트 제목 표시
ax1.set_title('titanic fare - hist/ked')     # 히스토그램 / 커널밀도함수
ax2.set_title('titanic fare - ked')          # 커널밀도함수
ax3.set_title('titanic fare - hist')         # 히스토그램

plt.show()
```

히스토그램

- 히스토그램과 커널 밀도 함수의 결과를 보면 타이타닉의 운임(fare)의 분포는 대부분 100달러 미만에 집중되어 있다.







히트맵

히트맵

seaborn 라이브러리는 히트맵(heatmap)을 그리는 heatmap() 함수를 제공한다.
히트맵은 2개의 범주형 변수를 각각 x, y축에 놓고 데이터를 매트릭스 형태로 분류한다.

```
# 피벗테이블로 범주형 변수를 각각 행, 열로 재구분하여 데이터프레임을 생성함
# aggfunc='size' 옵션은 데이터 값의 크기를 기준으로 집계한다는 의미
table = titanic.pivot_table(index=['sex'], columns=['class'], aggfunc='size')

# 히트맵 그리기
sns.heatmap(table,                      # 데이터프레임
            annot=True, fmt='d',        # 데이터 값 표시 여부, 정수형 포맷
            cmap='YlGnBu',              # 컬러 맵
            linewidth=.5,               # 구분 선
            cbar=False)                 # 컬러 바 표시 여부
```

히트맵

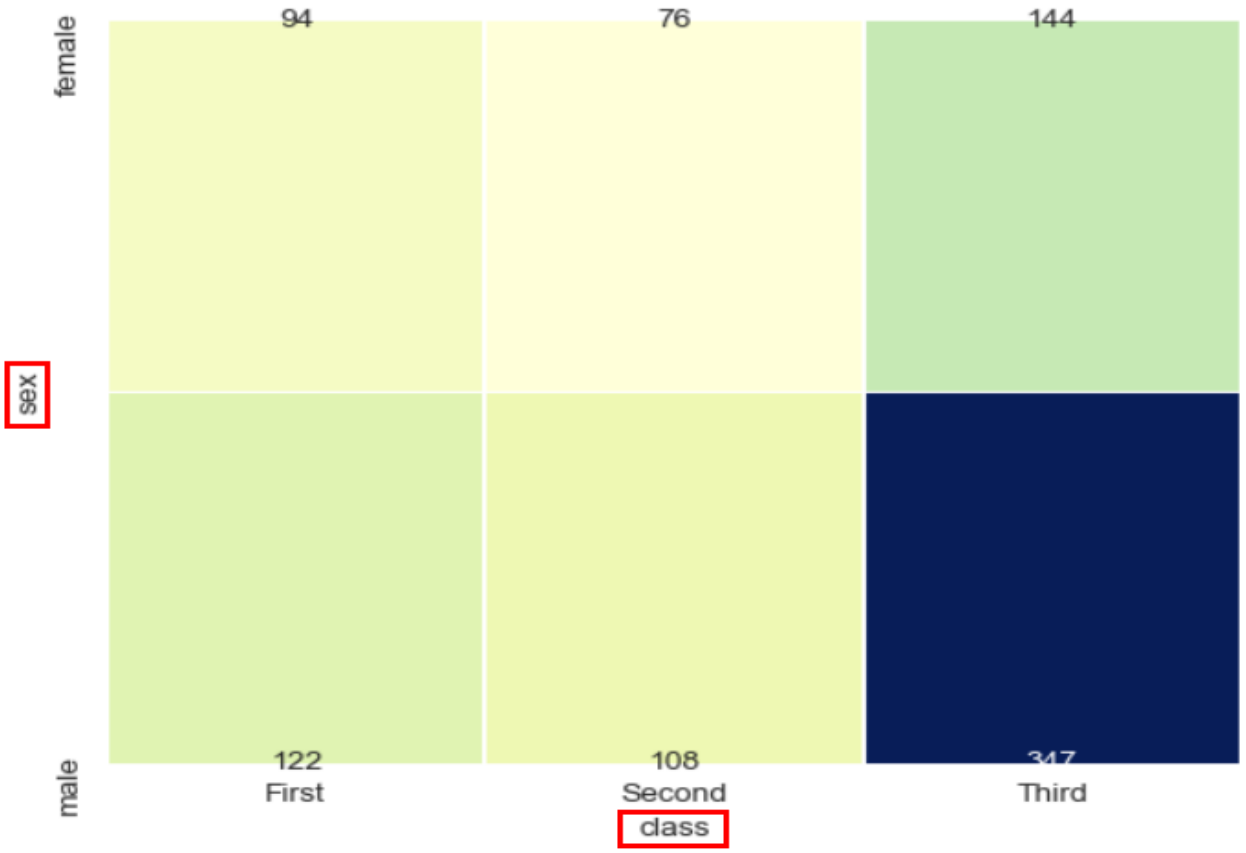
❖ 히트맵

seaborn_heatmap.py

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

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('darkgrid')

# 피벗테이블로 범주형 변수를 각각 행, 열로 재구분하여 데이터프레임을 생성함
# aggfunc='size' 옵션은 데이터 값의 크기를 기준으로 집계한다는 의미
table = titanic.pivot_table(index=['sex'], columns=['class'], aggfunc='size')

# 히트맵 그리기
sns.heatmap(table,                          # 데이터프레임
            annot=True, fmt='d',            # 데이터 값 표시 여부, 정수형 포맷
            cmap='YlGnBu',                  # 컬러 맵
            linewidth=.5,                   # 구분 선 두께
            cbar=False)                     # 컬러 바 표시 여부

plt.show()
```

히트맵

- 히트맵의 결과를 보면 남자(male) 승객수가 여자(female) 승객수보다 많은 편이다.
- 특히 3등석 남자 승객수가 여자 승객수 보다 압도적으로 많다는 것을 알 수 있다.



막대그래프

막대 그래프

seaborn 라이브러리는 막대 그래프를 그리기 위해 barplot() 함수를 제공한다.

```
# x축, y축에 변수 할당
sns.barplot(x='sex', y='survived', data=titanic, ax=ax1)

# x축, y축에 변수 할당하고, hue 옵션 추가하여 누적 출력순으로 출력
sns.barplot(x='sex', y='survived', hue='class', data=titanic, ax=ax2)

# x축, y축에 변수 할당하고, dodge=False 옵션으로 1개의 막대 그래프로 출력
sns.barplot(x='sex', y='survived', hue='class', dodge=False, data=titanic, ax=ax3)
```

막대그래프

막대 그래프

seaborn_bar.py (1/2)

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('whitegrid')

# 그래프 객체 생성 (figure에 3개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 5))                    # 그래프 크기 설정
ax1 = fig.add_subplot(1, 3, 1)                       # 1행 3열 - 1번째 그래프
ax2 = fig.add_subplot(1, 3, 2)                       # 1행 3열 - 2번째 그래프
ax3 = fig.add_subplot(1, 3, 3)                       # 1행 3열 - 3번째 그래프
```

막대그래프

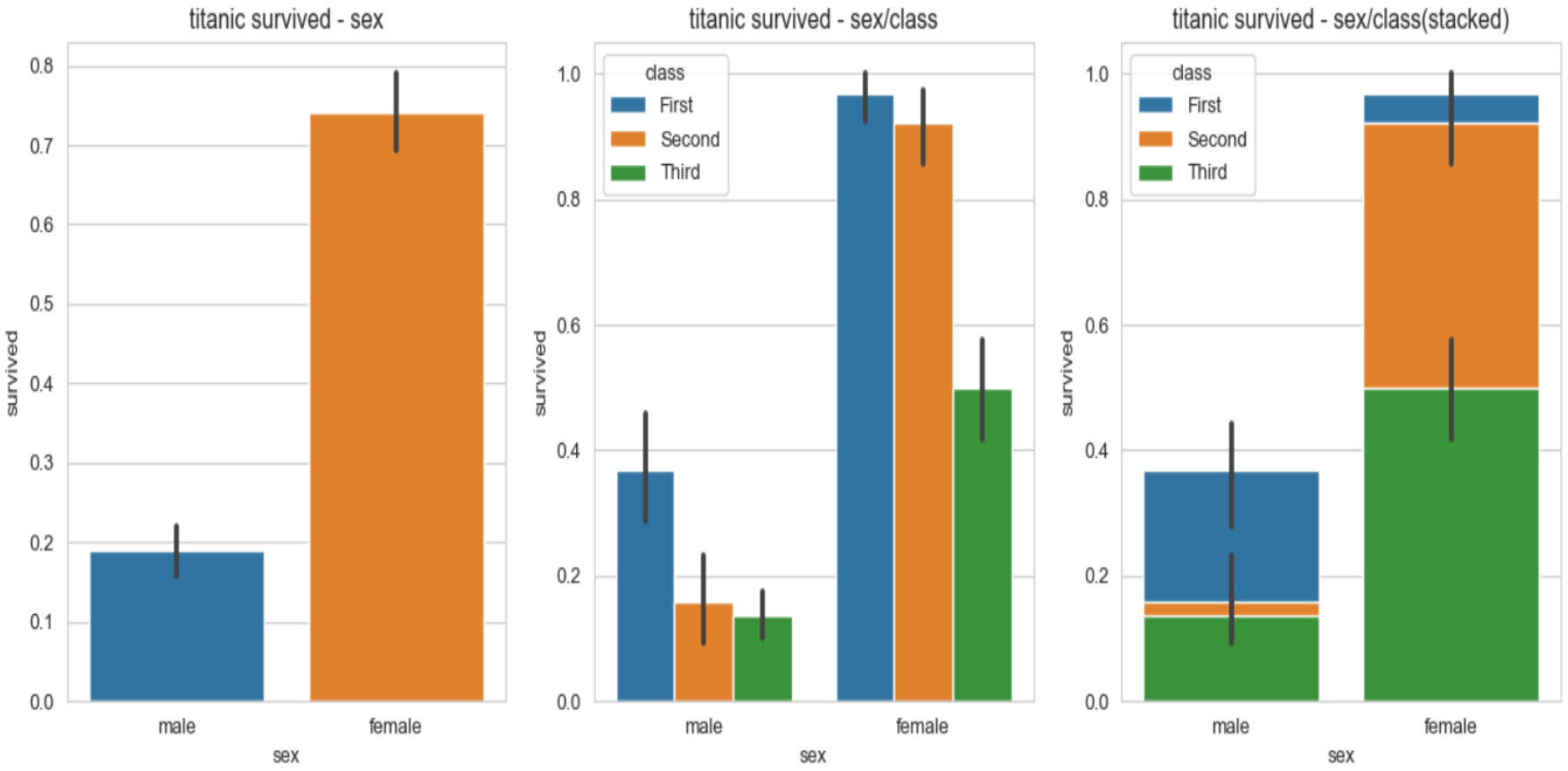
막대 그래프

seaborn_bar.py (2/2)

```
# x축, y축에 변수 할당
sns.barplot(x='sex', y='survived', data=titanic, ax=ax1)

# x축, y축에 변수 할당하고, hue 옵션 추가하여 누적 출력순으로 출력
sns.barplot(x='sex', y='survived', hue='class', data=titanic, ax=ax2)

# x축, y축에 변수 할당하고, dodge=False 옵션으로 1개의 막대그래프로 출력
sns.barplot(x='sex', y='survived', hue='class', dodge=False, data=titanic, ax=ax3)

# 차트 제목 표시
ax1.set_title('titanic survived - sex')
ax2.set_title('titanic survived - sex/class')
ax3.set_title('titanic survived - sex/class(stacked)')

plt.show()
```

막대그래프

- 생존자는 남성 보다는 여성 승객이 훨씬 많다는 것을 알수 있다.
- 남성 생존자는 1등석이 많고, 여성 생존자는 1등석,2등석 승객이 많다는 것을 알 수 있다.



빈도 막대그래프

❖ 빈도 막대그래프

seaborn 라이브러리는 빈도 막대그래프를 그리기 위해 countplot() 함수를 제공한다.

```
# 기본값 : palette='Set1' 로 색상 설정
sns.countplot(x='class', palette='Set1', data=titanic, ax=ax1)

# hue 옵션에 'who' 추가  : who(man, woman, child) 각각 빈도 막대그래프 출력
sns.countplot(x='class', hue='who', palette='Set2', data=titanic, ax=ax2)

# dodge=False 옵션 추가 (축 방향으로 분리하지 않고 누적 그래프 출력)
sns.countplot(x='class', hue='who', palette='Set3', dodge=False, data=titanic, ax=ax3)
```

빈도 막대그래프

- **빈도 막대그래프**

seaborn_count.py (1/2)

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('whitegrid')

# 그래프 객체 생성 (figure에 3개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 5))              # 그래프 크기 설정
ax1 = fig.add_subplot(1, 3, 1)                 # 1행 3열 - 1번째 그래프
ax2 = fig.add_subplot(1, 3, 2)                 # 1행 3열 - 2번째 그래프
ax3 = fig.add_subplot(1, 3, 3)                 # 1행 3열 - 3번째 그래프
```

빈도 막대그래프

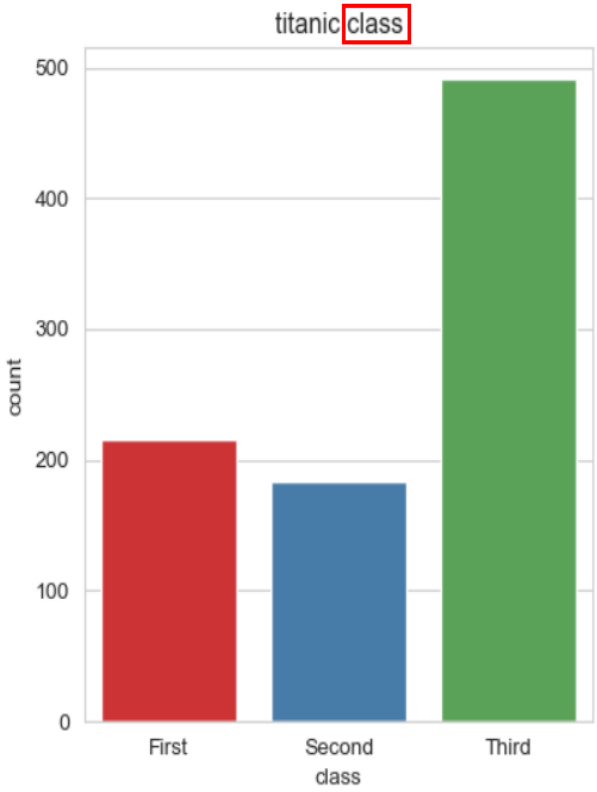
빈도 막대그래프

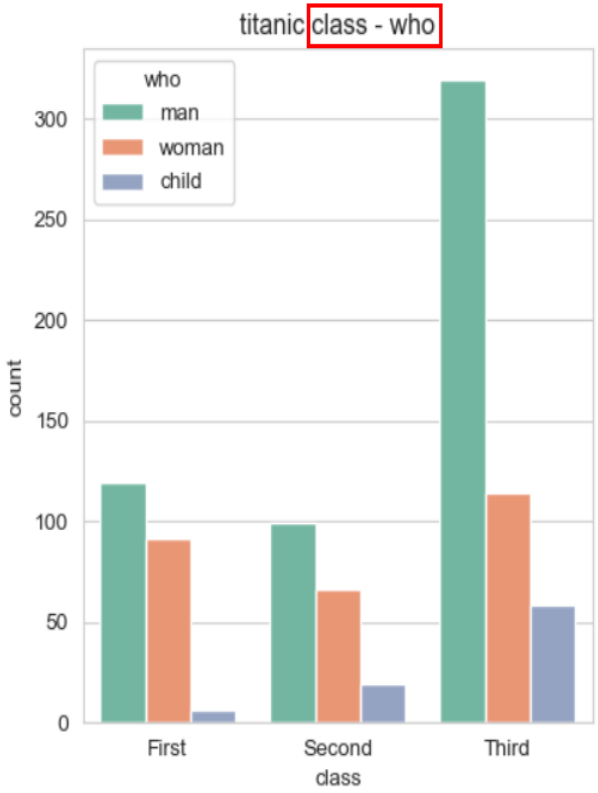
seaborn_count.py (2/2)

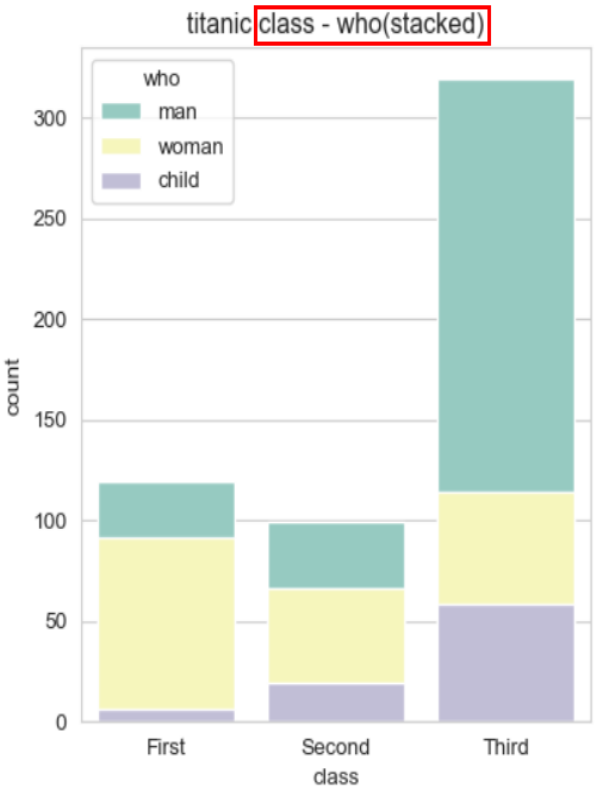
```
# 기본값 : palette='Set1' 로 색상 설정
sns.countplot(x='class', palette='Set1', data=titanic, ax=ax1)

# hue 옵션에 'who' 추가  : who(man, woman, child) 각각 빈도 막대그래프 출력
sns.countplot(x='class', hue='who', palette='Set2', data=titanic, ax=ax2)

# dodge=False 옵션 추가 (축 방향으로 분리하지 않고 누적 그래프 출력)
sns.countplot(x='class', hue='who', palette='Set3', dodge=False, data=titanic, ax=ax3)

# 차트 제목 표시
ax1.set_title('titanic class')
ax2.set_title('titanic class - who')
ax3.set_title('titanic class - who(stacked)')

plt.show()
```

빈도 막대그래프

- 타이타닉 승객은 1,2등석 보다 3등석이 훨씬 많은 것을 알 수 있다.
- 타이타닉 승객중 3등석에 남성의 비율이 높은 것을 알 수 있다.
- 타이타닉 좌석중 1등석은 여성의 비율이 높고, 3등석은 남성의 비율이 상대적으로 높은 것을 알 수 있다.







박스플롯 / 바이올린그래프

박스플롯 / 바이올린 그래프

박스플롯은 범주형 데이터 분포와 주요 통계 지표를 함께 제공한다.
박스플롯 만으로는 데이터가 퍼져있는 분산의 정도를 정확하게 알기 어렵기 때문에
커널 밀도함수를 y축 방향에 추가하여 바이올린 그래프로 출력할 수 있다.
박스플롯은 boxplot() 함수로 그릴수 있고, 바이올린 그래프는 violinplot() 함수로 그릴수 있다.
다음은 타이타닉 생존자 분포를 박스플롯과 바이올린 그래프로 출력 해보자

```
# 박스 그래프 - 기본값
sns.boxplot(x='alive', y='age', data=titanic, ax=ax1)

# 박스 그래프 - hue = 'sex' 변수를 추가하여 남녀 데이터를 구분하여 출력
sns.boxplot(x='alive', y='age', hue='sex', data=titanic, ax=ax2)

# 바이올린 그래프 - 기본값
sns.violinplot(x='alive', y='age', data=titanic, ax=ax3)

# 바이올린 그래프 - hue = 'sex' 변수를 추가하여 남녀 데이터를 구분하여 출력
sns.violinplot(x='alive', y='age', hue='sex', data=titanic, ax=ax4)
```

박스플롯 / 바이올린그래프

박스플롯 / 바이올린 그래프

seaborn_box_violin.py (1/2)

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('whitegrid')

# 그래프 객체 생성 (figure에 4개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 10))                    # 그래프 크기 설정
ax1 = fig.add_subplot(2, 2, 1)                        # 2행 2열 - 1번째 그래프
ax2 = fig.add_subplot(2, 2, 2)                        # 2행 2열 - 2번째 그래프
ax3 = fig.add_subplot(2, 2, 3)                        # 2행 2열 - 3번째 그래프
ax4 = fig.add_subplot(2, 2, 4)                        # 2행 2열 - 4번째 그래프
```

박스플롯 / 바이올린그래프

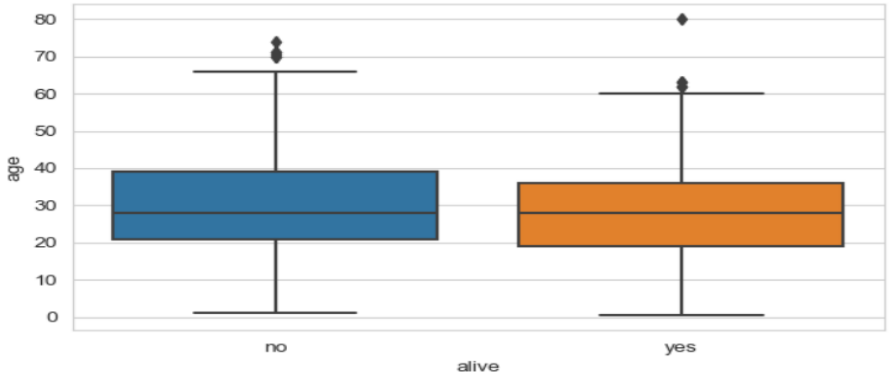
박스플롯 / 바이올린 그래프

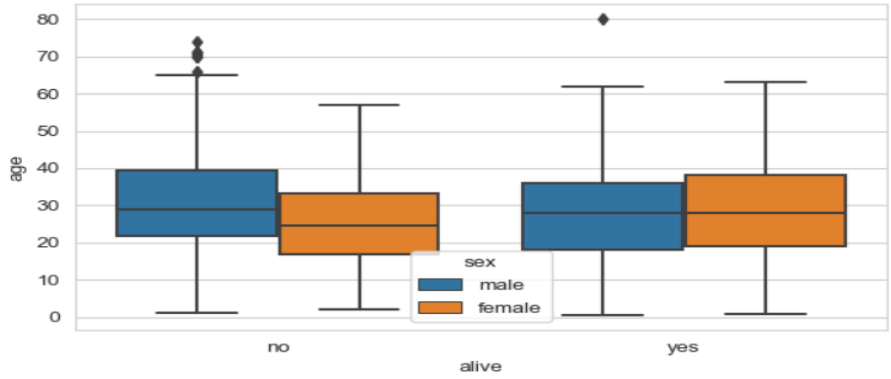
seaborn_box_violin.py (2/2)

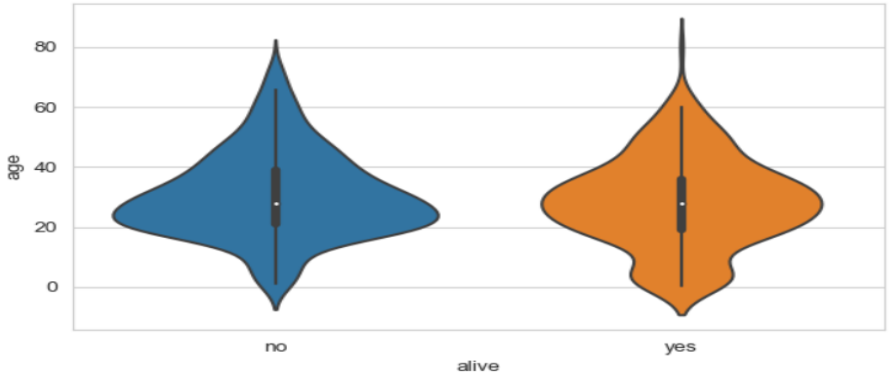
```
# 박스 그래프 - 기본값
sns.boxplot(x='alive', y='age', data=titanic, ax=ax1)

# 박스 그래프 - hue = 'sex' 변수를 추가하여 남녀 데이터를 구분하여 출력
sns.boxplot(x='alive', y='age', hue='sex', data=titanic, ax=ax2)

# 바이올린 그래프 - 기본값
sns.violinplot(x='alive', y='age', data=titanic, ax=ax3)

# 바이올린 그래프 - hue = 'sex' 변수를 추가하여 남녀 데이터를 구분하여 출력
sns.violinplot(x='alive', y='age', hue='sex', data=titanic, ax=ax4)

plt.show()
```

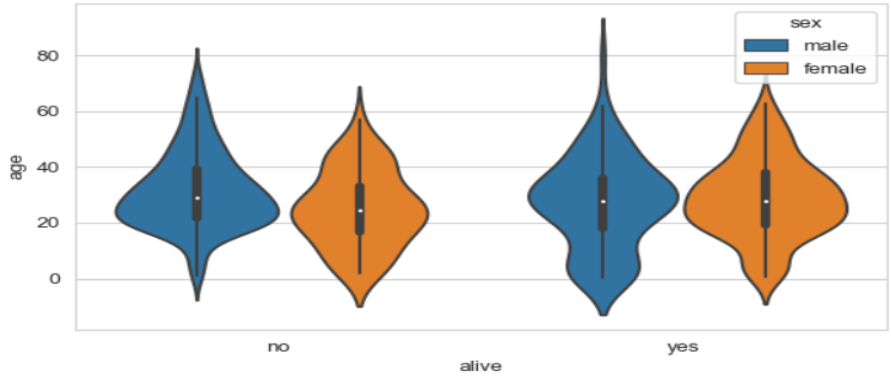
박스플롯 / 바이올린그래프

- 박스플롯으로 보면 타이타닉 생존자는 남자는 20~30대가 많고, 여자는20~40대가 많은 것을 알 수 있다.
- 바이올린 그래프로 타이타닉 생존자는 남자는 30대가 가장 많이 분포해 있고, 여자는 20~30대가 가장 많이 분포해 있는 것을 알 수 있다.



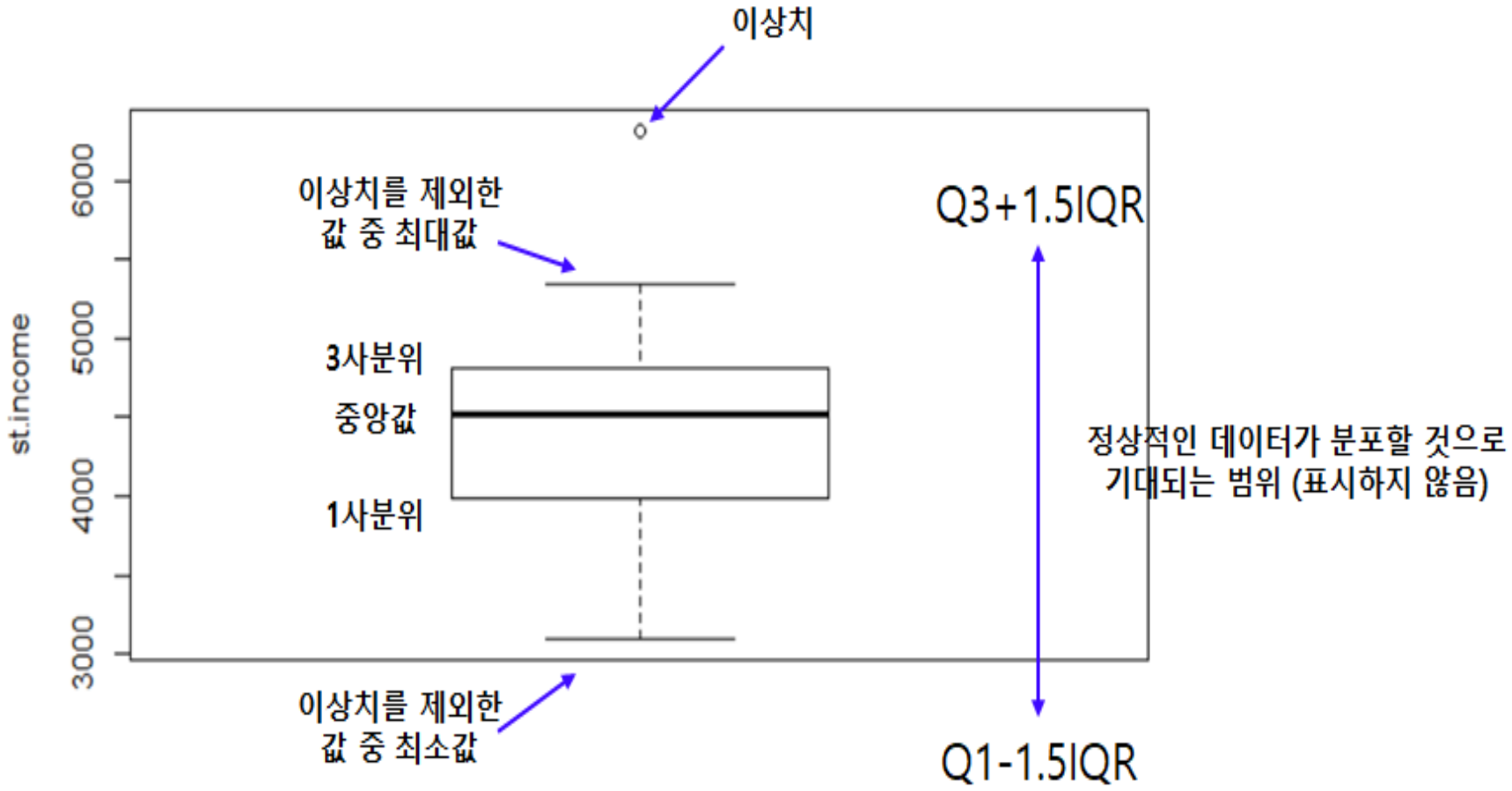






박스플롯 / 바이올린그래프

- 박스플롯



조인트 그래프

조인트 그래프

조인트 그래프는 jointplot() 함수로 산점도를 기본적으로 표시하고, x-y축에 각 변수에 대한 히스토그램을 동시에 보여주는 그래프 이다. 따라서 두 변수의 관계와 데이터가 분산되어 있는 정도를 한눈에 파악하기 좋은 그래프 이다.
예제에서는 산점도(기본값), 회귀선 추가(kind='reg'), 육각 산점도(kind='hex'), 커널밀집 그래프(kind='kde') 순으로 조인트 그래프를 그려서 차이를 비교해보자.

```
# 조인트 그래프 - 산점도(기본값)
j1 = sns.jointplot(x='fare', y='age', data=titanic)

# 조인트 그래프 - 회귀선 : kind='reg'
j2 = sns.jointplot(x='fare', y='age', kind='reg', data=titanic)

# 조인트 그래프 - 육각 그래프 : kind='hex'
j3 = sns.jointplot(x='fare', y='age', kind='hex', data=titanic)

# 조인트 그래프 - 커럴 밀집 그래프 : kind='kde'
j4 = sns.jointplot(x='fare', y='age', kind='kde', data=titanic)
```

조인트 그래프

❖ 조인트 그래프

seaborn_joint.py (1/2)

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('whitegrid')
```

조인트 그래프

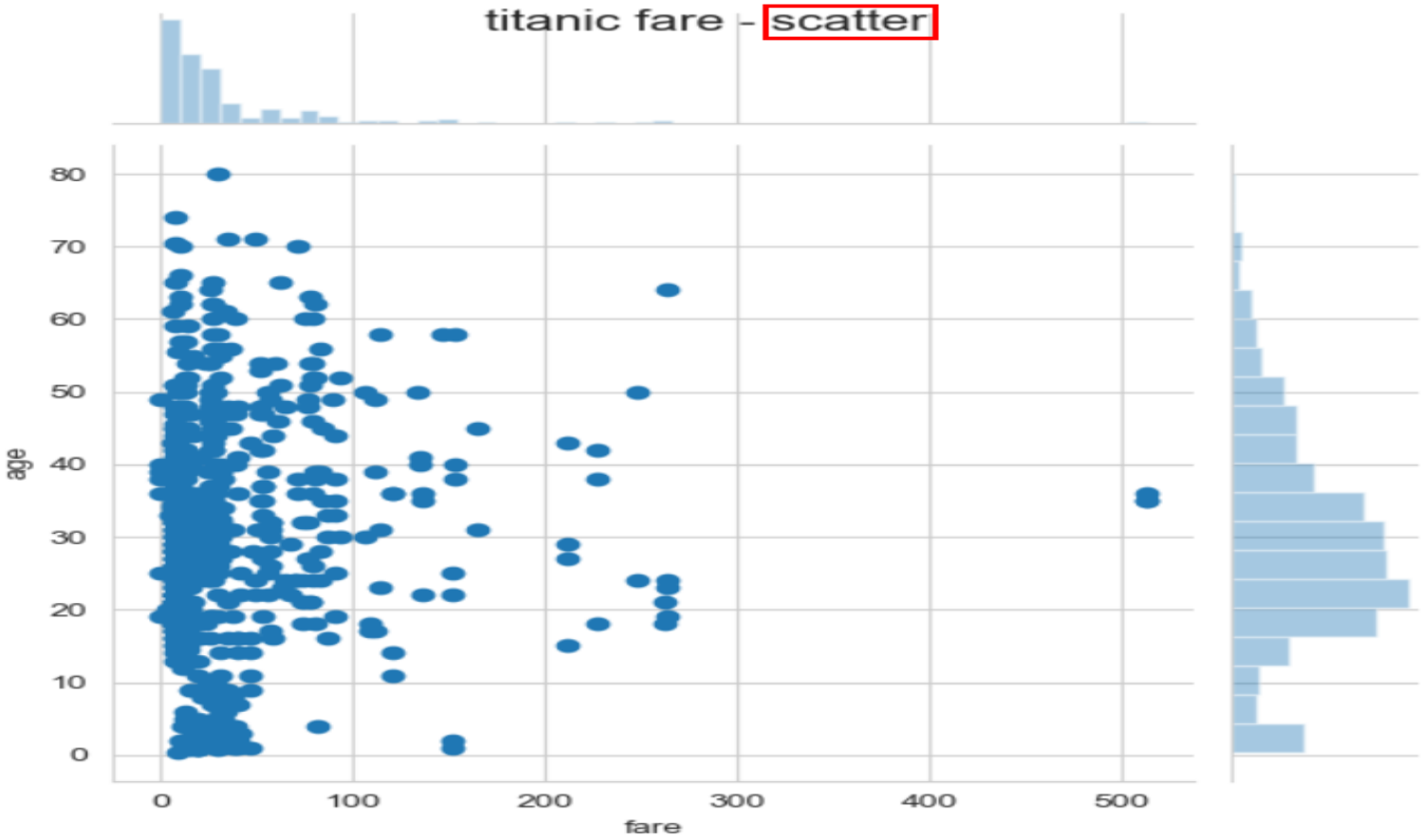
- **조인트 그래프**

seaborn_joint.py (2/2)

```
# 조인트 그래프 - 산점도(기본값)
j1 = sns.jointplot(x='fare', y='age', data=titanic)

# 조인트 그래프 - 회귀선 : kind='reg'
j2 = sns.jointplot(x='fare', y='age', kind='reg', data=titanic)

# 조인트 그래프 - 육각 그래프 : kind='hex'
j3 = sns.jointplot(x='fare', y='age', kind='hex', data=titanic)

# 조인트 그래프 - 커널 밀집 그래프 : kind='kde'
j4 = sns.jointplot(x='fare', y='age', kind='kde', data=titanic)

# 차트 제목 표시
j1.fig.suptitle('titanic fare - scatter', size=15)
j2.fig.suptitle('titanic fare - reg', size=15)
j3.fig.suptitle('titanic fare - hex', size=15)
j4.fig.suptitle('titanic fare - kde', size=15)

plt.show()
```

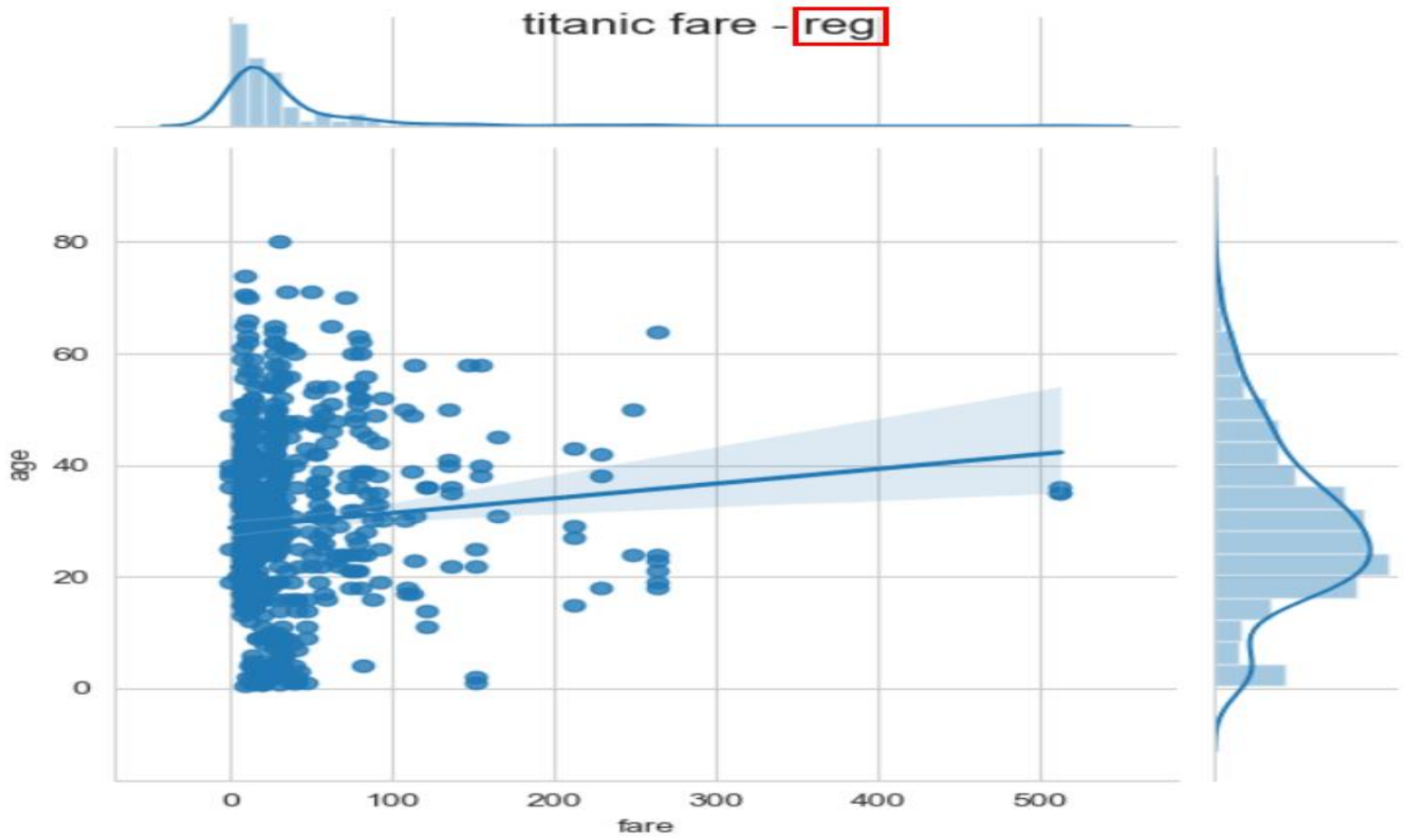
조인트 그래프

- 조인트 그래프 – 산점도



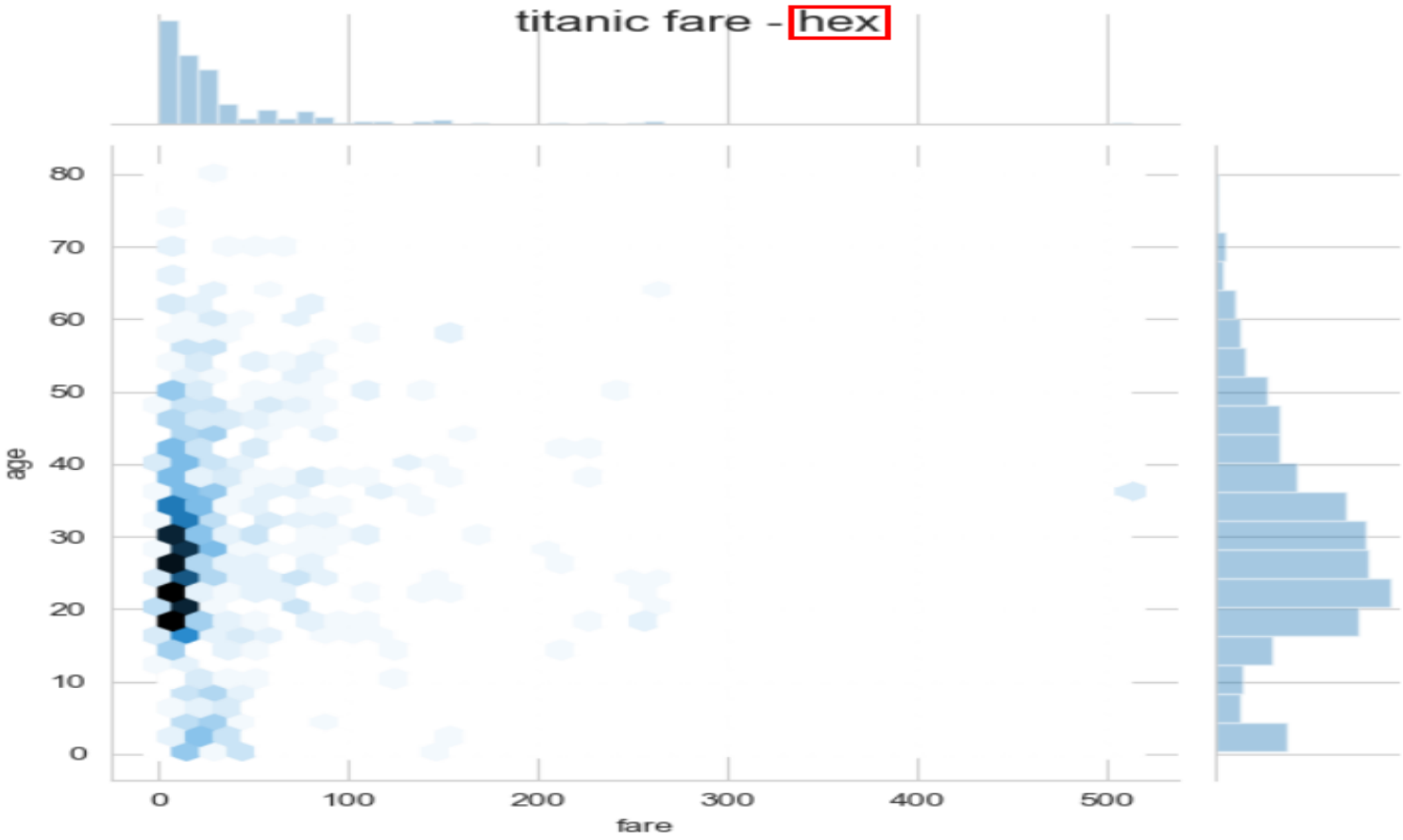
조인트 그래프

- 조인트 그래프 – 산점도 + 회귀선 추가



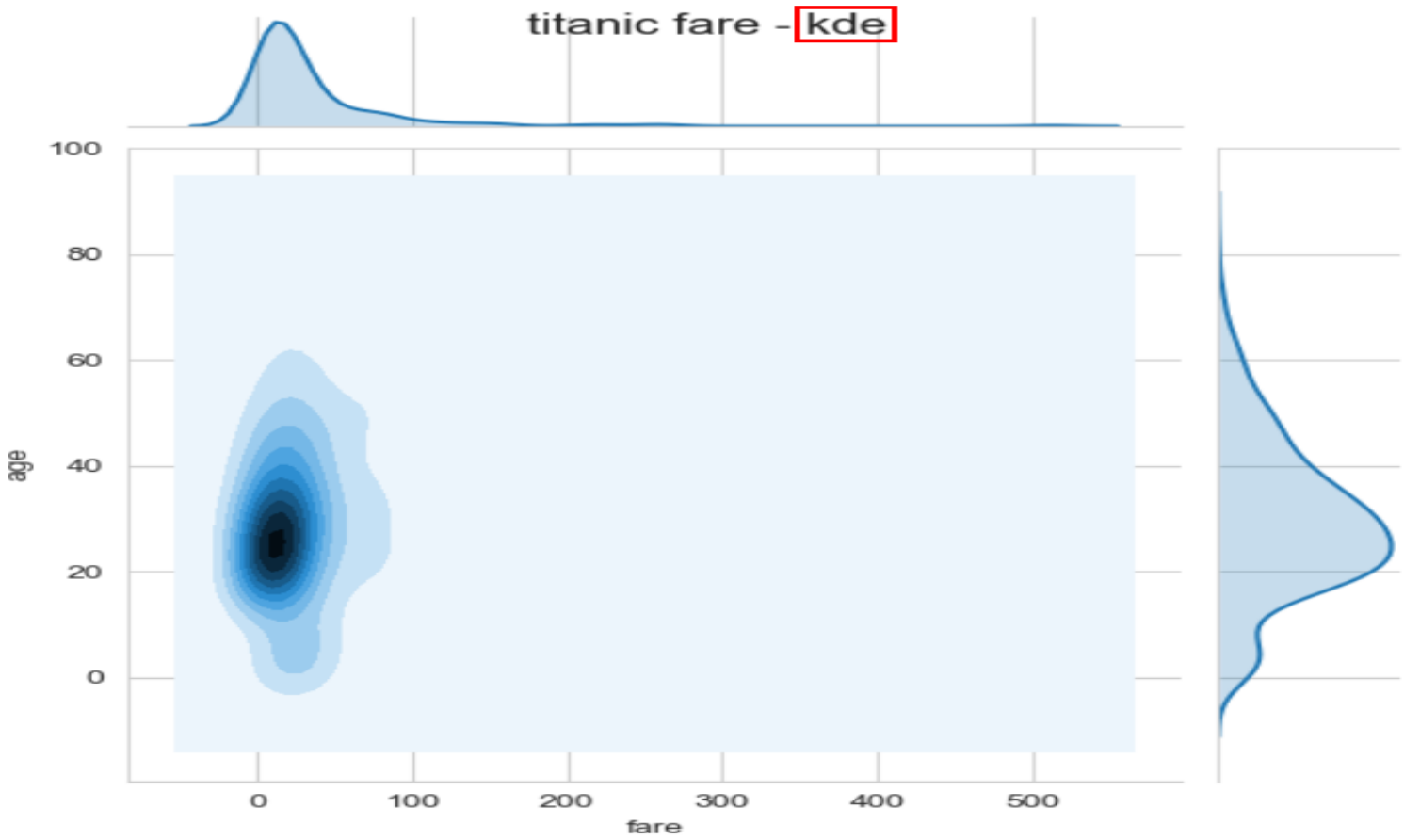
조인트 그래프

- 조인트 그래프 – 육각 산점도



조인트 그래프

- 조인트 그래프 – 커널밀집 그래프



조건을 적용한 그래프

조건을 적용하여 화면을 그리드로 분할한 그래프

FacetGrid() 함수는 행, 열 방향으로 서로 다른 조건을 적용하여 여러 개의 서브플롯 그래프를 그려주는 역할을 한다. 그리고 각 서브플롯에 적용할 그래프 종류는 map() 함수를 이용하여 그리드 객체에 전달한다.

다음 예제는 열 방향으로는 'who' 열의 탑승객 구분(man, woman, child) 값으로 구분하고, 행 방향으로는 'survived' 열의 구조 여부(구조 survived=1, 구조실패 survived=0) 값으로 구분하여 2행 3열 모양의 그리드로 그래프를 출력한다.
각 조건에 맞는 탑승객을 구분하여, 'age' 열의 나이를 기준으로 히스토그램을 그려보자

```
# 조건에 따라 그리드 나누기 : who(man, woman, child) , survived (0 or 1)
g = sns.FacetGrid(data=titanic, col='who', row='survived')

# 그래프 적용하기 : 히스토그램
g = g.map(plt.hist, 'age')
```

조건을 적용한 그래프

- **조건을 적용하여 화면을 그리드로 분할한 그래프**

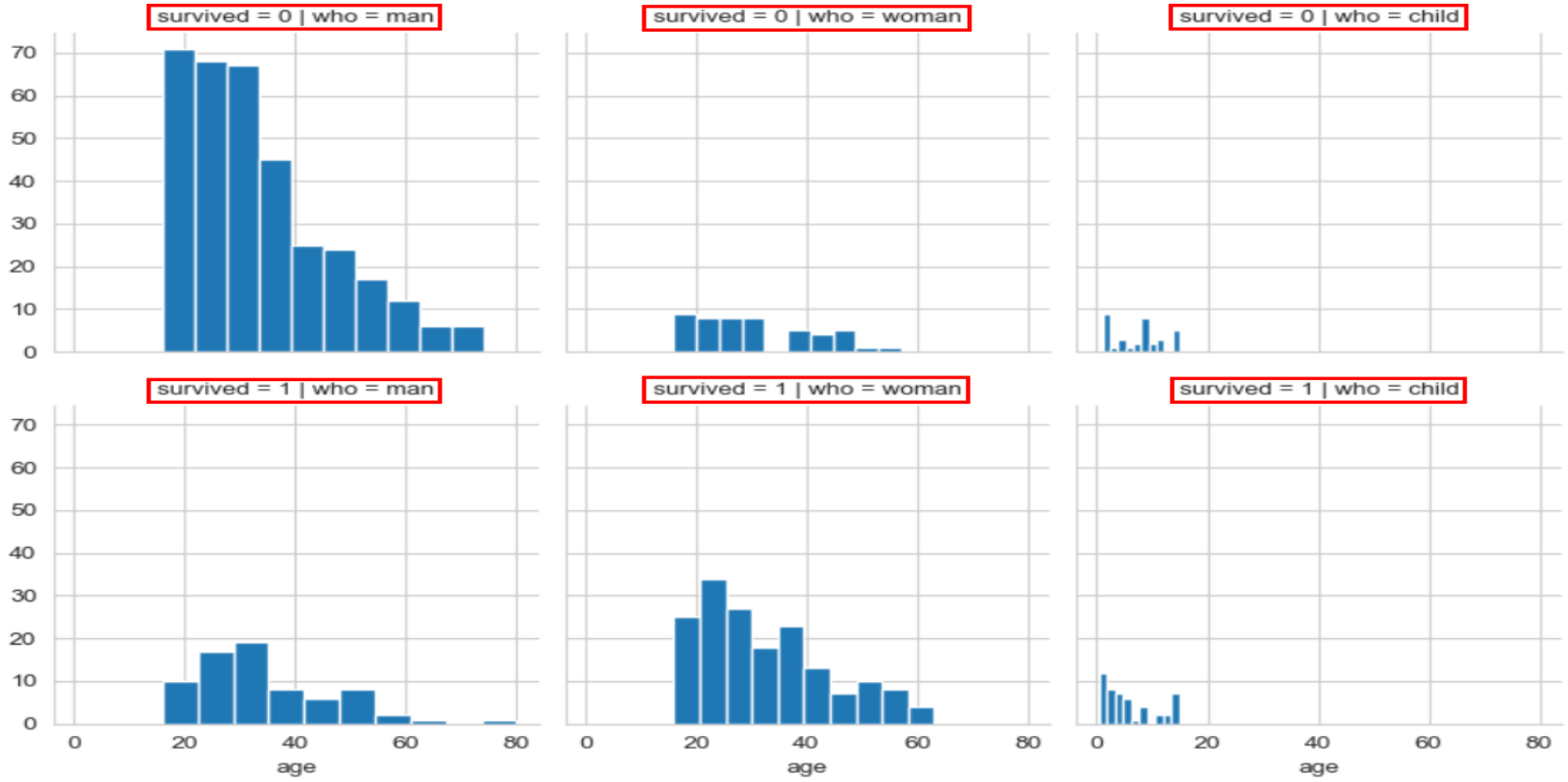
seaborn_facetgrid.py

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('whitegrid')

# 조건에 따라 그리드 나누기 : who(man, woman, child) , survived (0 or 1)
g = sns.FacetGrid(data=titanic, col='who', row='survived')

# 그래프 적용하기 : 히스토그램
g = g.map(plt.hist, 'age')

plt.show()
```

조건을 적용한 그래프

- **조건을 적용하여 화면을 그리드로 분할한 그래프**

남성에 비해서 여성 생존자가 상대적으로 많은 편이고, 성인 중에서는 활동성이 좋은 20 ~ 40대의 생존자가 많은 것으로 나타난다.



데이터 분포 그래프

데이터 분포 그래프

pairplot() 함수는 인자로 전달되는 데이터프레임의 열을 두개씩 짝을 지을 수 있는 모든 조합에 대해서 그래프로 표현한다. 그래프를 그리기 위해서는 데이터의 개수만큼 짝의 개수만큼 화면을 그리드로 나눈다.

다음에 예제는 3개의 열을 사용하기 때문에 3행 3열 크기로 모두 9개의 그리드를 만든다. 각 그리드에 두 변수 간의 관계를 나타내는 그래프를 하나씩 그린다.
같은 변수끼리 짝을 이루는 대각선 방향으로는 히스토그램을 그리고 서로 다른 변수 간에는 산점도를 그린다.

```
# titanic 데이터셋 중에서 분석 데이터 선택하기
titanic_pair = titanic[['age','pclass', 'fare']]

# 조건에 따라 그리드 나누기 : 3행 3열 그리드로 출력
g = sns.pairplot(titanic_pair)
```

데이터 분포 그래프

- **데이터 분포 그래프**

seaborn_pairplot.py

```
# 라이브러리 불러오기
import matplotlib.pyplot as plt
import seaborn as sns

# Seaborn 제공 데이터셋 가져오기
titanic = sns.load_dataset('titanic')

# 스타일 테마 설정 (5가지: darkgrid, whitegrid, dark, white, ticks)
sns.set_style('whitegrid')

# titanic 데이터셋 중에서 분석 데이터 선택하기
titanic_pair = titanic[['age','pclass', 'fare']]

# 조건에 따라 그리드 나누기 : 3행 3열 그리드로 출력
g = sns.pairplot(titanic_pair)

plt.show()
```

데이터 분포 그래프

- **데이터 분포 그래프**

  같은 변수끼리 짝을 이루는 대각선 방향은 히스토그램을 그리고 서로 다른 변수 간에는 산점도로 출력된다.