데이터 시각화 - Seaborn

Seaborn

Seaborn은 Matplotlib의 기능과 스타일을 확장한 파이썬 시각화 도구의 고급 버전이다. 비교적 단순한 인터페이스를 제공한다.

먼저 Seaborn 라이브러리를 설치해야 한다.

(* 아나콘다 배포판을 사용하는 경우 기본으로 설치되기 때문에 추가 설치를 하지 않아도 된다.)

```
!pip install seaborn
```

Seaborn

```
import seaborn as sns
```

Seaborn 라이브러리를 임포트 할 때는 'SNS'라는 약칭을 주로 사용한다.

```
titanic = sns.load_dataset('titanic')
print(titanic.head())
print('\n')
print(titanic.info())
```

Seaborn 라이브러리에서 제공하는 'titanic' 데이터셋을 사용한다.

Seaborn의 load_dataset() 함수를 사용하여 데이터프레임으로 가져온다.

Seaborn

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	₩
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	

	who	adult_male	deck	embark_town	alive	alone
0	man	True	NaN	Southampton	no	False
1	woman	False	C	Cherbourg	yes	False
2	woman	False	NaN	Southampton	yes	True
3	woman	False	C	Southampton	yes	False
4	man	True	NaN	Southampton	no	True

Seaborn

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
None
```

모두 891명의 탑승객 정보가 담겨 있다.

휘귀선이 있는 산점도

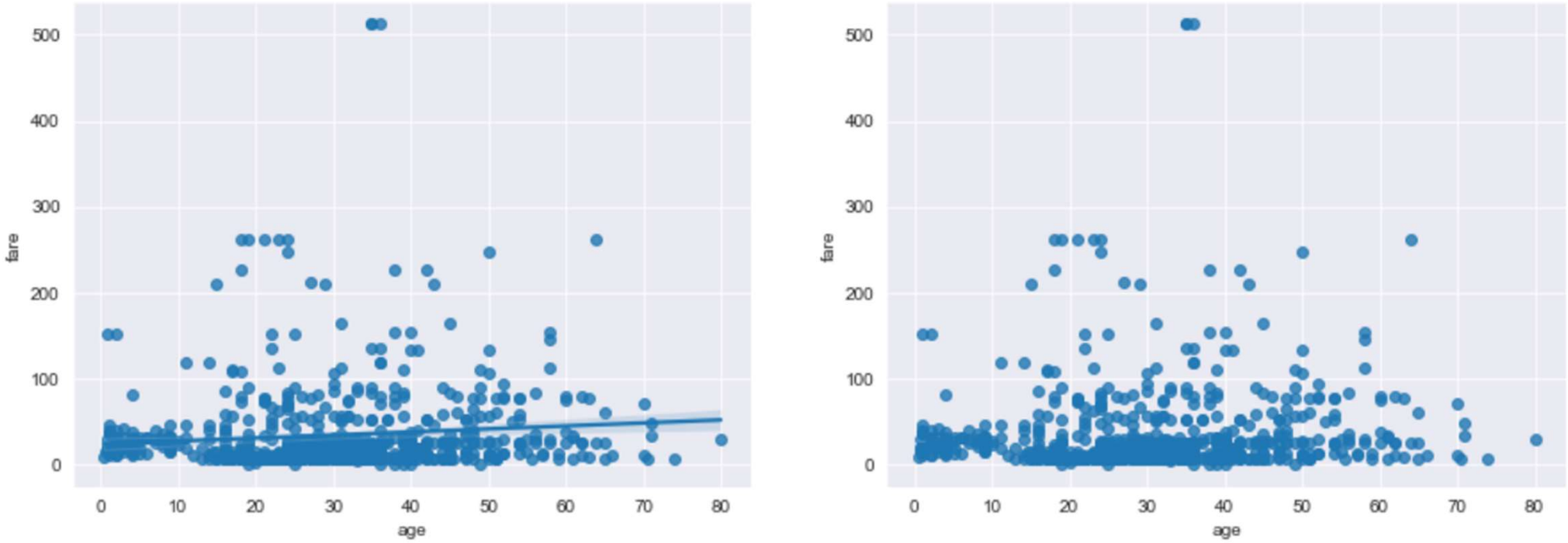
regplot() 함수는 서로 다른 2개의 연속 변수 사이의 산점도를 그리고 선형희귀분석에 의한 회귀선을 함께 나타낸다. Fit_reg=False 옵션을 설정하면 휘귀선을 안 보이게 할 수 있다.

```
# 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))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)
```

회귀선이 있는 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

# 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()
```

휘귀선이 있는 산점도



다음의 예제에서 왼쪽 그래프는 선형회귀선을 표시하고 오른쪽 그래프에는 표시하지 않는다.

히스토그램/커널 밀도 그래프

단변수(하나의 변수) 데이터의 분포를 확인할 때 distplot() 함수를 이용한다.
기본값으로 히스토그램과 커널 밀도 함수를 그래프로 출력한다.

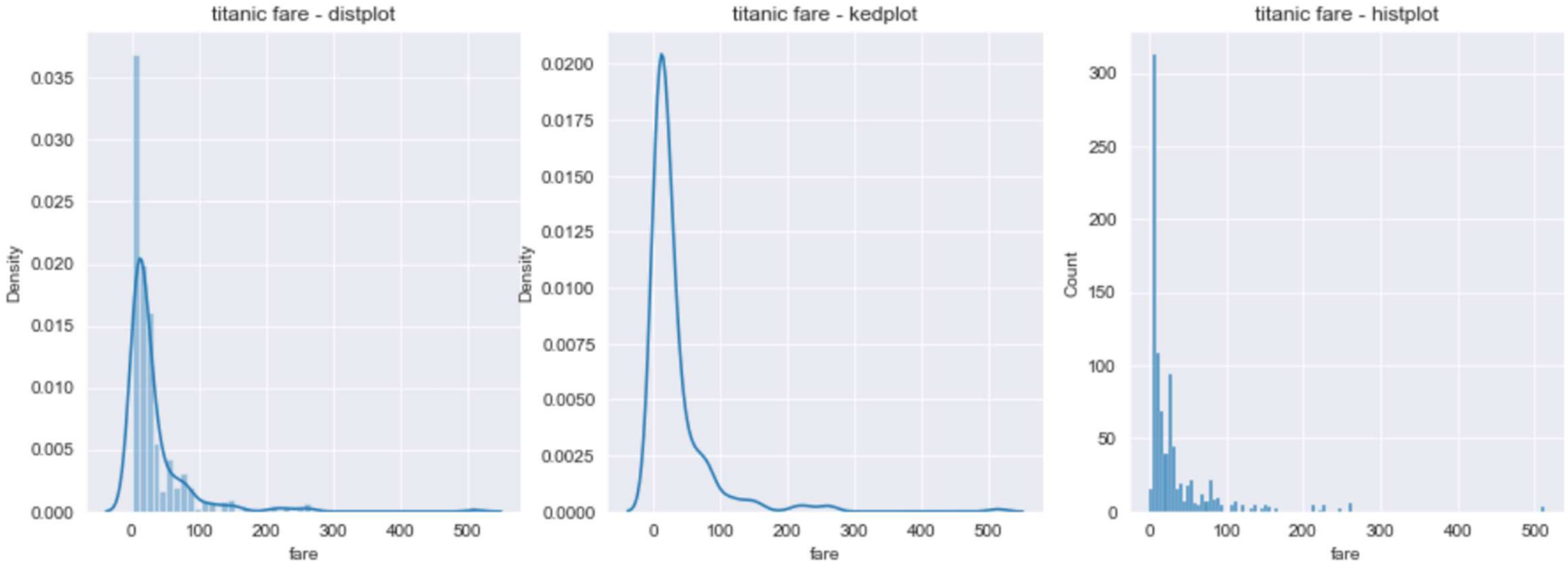
- 커널 밀도 함수는 그래프와 x축 사이의 면적이 1이 되도록 그리는 밀도 분포 함수이다.

```
# 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))
ax1 = fig.add_subplot(1, 3, 1)
ax2 = fig.add_subplot(1, 3, 2)
ax3 = fig.add_subplot(1, 3, 3)
```

히스토그램/커널 밀도 그래프

```
# distplot
sns.distplot(titanic['fare'], ax=ax1)

# kdeplot
sns.kdeplot(x='fare', data=titanic, ax=ax2)

# histplot
sns.histplot(x='fare', data=titanic,  ax=ax3)

# 차트 제목 표시
ax1.set_title('titanic fare - distplot')
ax2.set_title('titanic fare - kedplot')
ax3.set_title('titanic fare - histplot')

plt.show()
```

히스토그램/커널 밀도 그래프



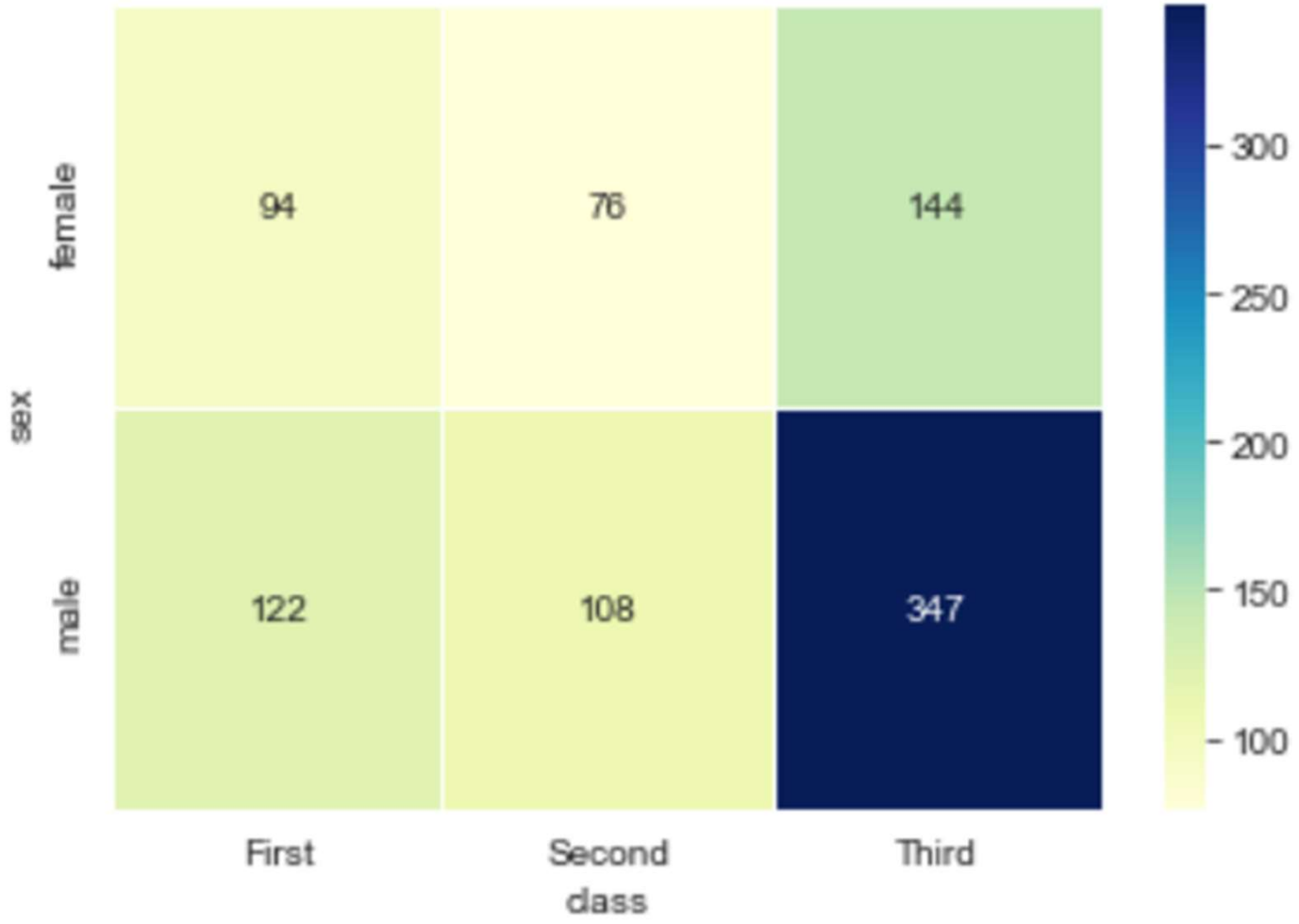
예제에서 타이타닉의 움임(‘fare’ 열)의 분포를 그리면 대부분 100달러 미만에 집중되어 있다.

히트맵

Seaborn 라이브러리는 히트맵(heatmap)을 그리는 heatmap() 메소드를 제공한다.
2개의 범주형 변수를 각각 x, y축에 놓고 데이터를 매트릭스 형태로 분류한다.
데이터프레임을 피벗테이블로 정리할 때 한 변수('sex' 열)를 행 인덱스로 나머지 변수('class' 열)를 열 이름으로 설정한다.
aggfunc = 'size' 옵션은 데이터 값의 크기를 기준으로 집계한다는 뜻이다.

```
# 라이브러리 불러오기 생략
# Seaborn 제공 데이터셋 가져오기 생략
# 스타일 테마 설정 생략

# 피벗테이블로 범주형 변수를 각각 행, 열로 재구분하여 정리
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=True)              # 컬러 바 표시 여부
plt.show()
```

히트맵



히트맵을 그려 보면 타이타닉호에는 여자(female) 승객보다 남자(male) 승객이 상대적으로 많은 편이다. 특히 3등석 남자 승객의 수가 압도적으로 많은 것을 알 수 있다.

범주형 데이터의 산점도

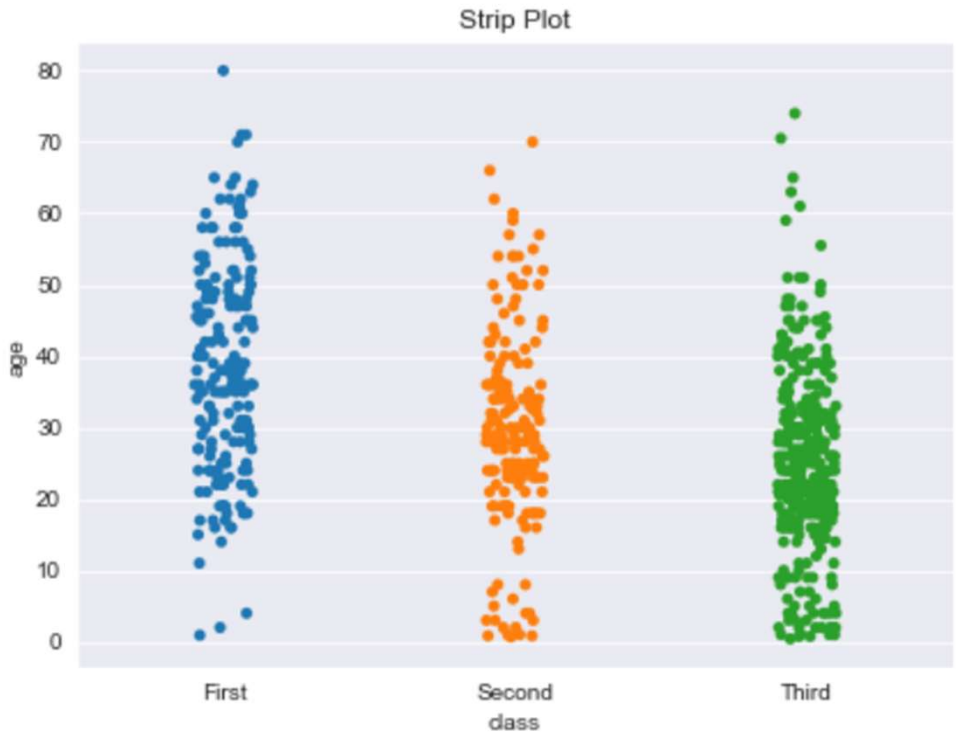
범주형 변수에 들어 있는 각 범주별 데이터의 분포를 확인하는 방법이다.
Stripplot() 함수와 swarmplot() 함수를 사용한다.
Swarmplot() 함수는 데이터의 분산까지 고려하여, 데이터 포인트가 서로 중복되지 않도록 그린다. 즉, 데이터가 퍼져 있는 정도를 입체적으로 볼 수 있다.

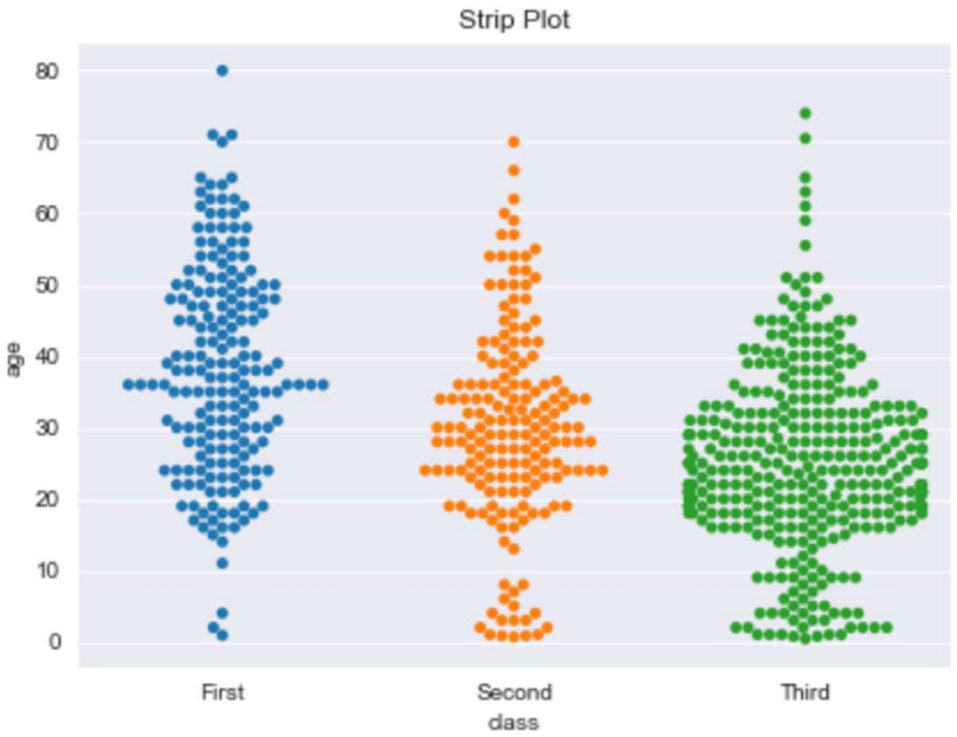
```
# 라이브러리 불러오기 생략
# Seaborn 제공 데이터셋 가져오기 생략
# 스타일 테마 설정 생략

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

범주형 데이터의 산점도

```
# 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('Strip Plot')
ax2.set_title('Strip Plot')

plt.show()
```

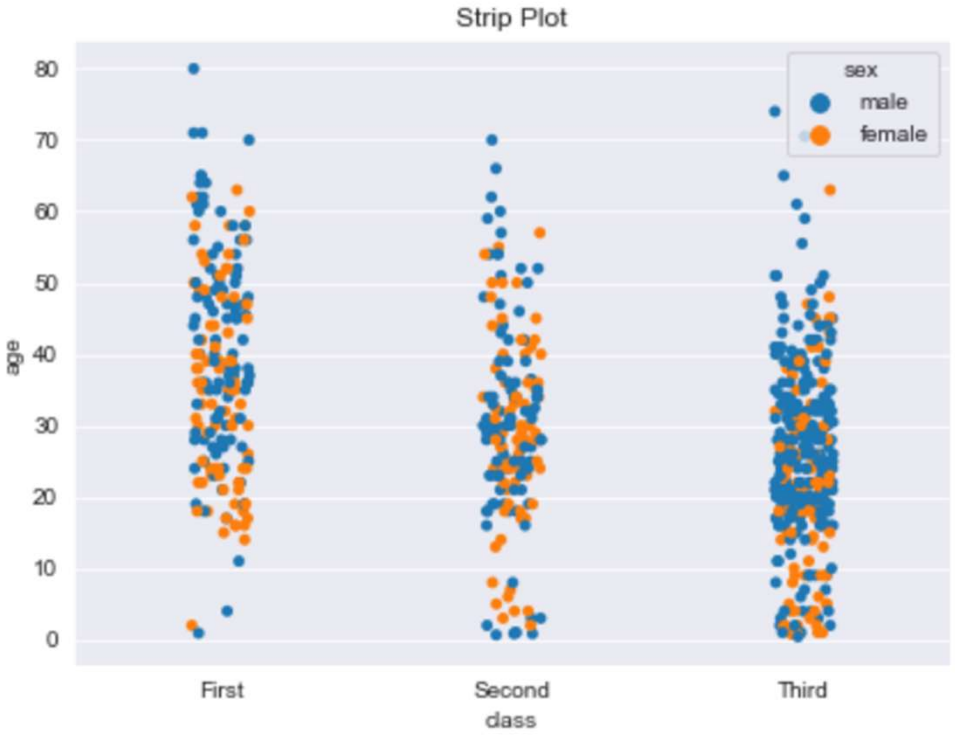
범주형 데이터의 산점도

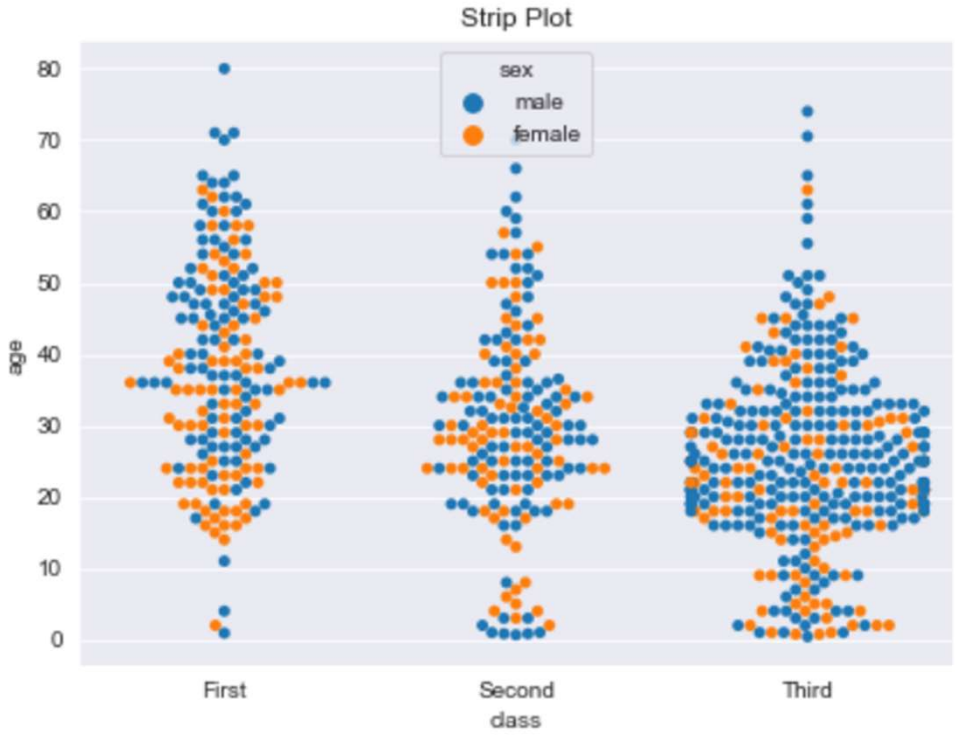

Strip Plot
age
80
70
60
50
40
30
20
10
0
First
Second
Third
class


Strip Plot
age
80
70
60
50
40
30
20
10
0
First
Second
Third
class

범주형 데이터의 산점도

앞의 예제에서 hue='sex' 옵션을 stripplot() 함수 또는 swarmplot() 함수에 추가하면, 'sex' 열의 데이터 값인 남녀 성별을 생삭으로 구분하여 출력한다.





막대 그래프

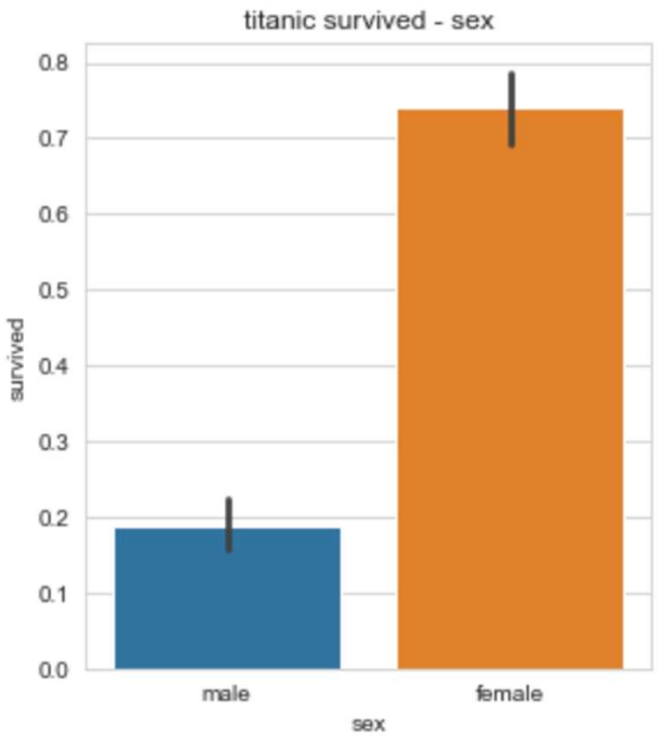
막대 그래프를 그리는 barplot() 함수를 살펴보자.
3개의 axe 객체(서브 플롯)을 만들고, 옵션에 변화를 주면서 차이를 살펴보자.
X축, y축에 변수 할당, x축, y축에 변수 할당하고 hue 옵션 추가, x축, y축에 변수 할당하고 hue 옵션을 추가하여 누적 출력 순으로 실행한다.

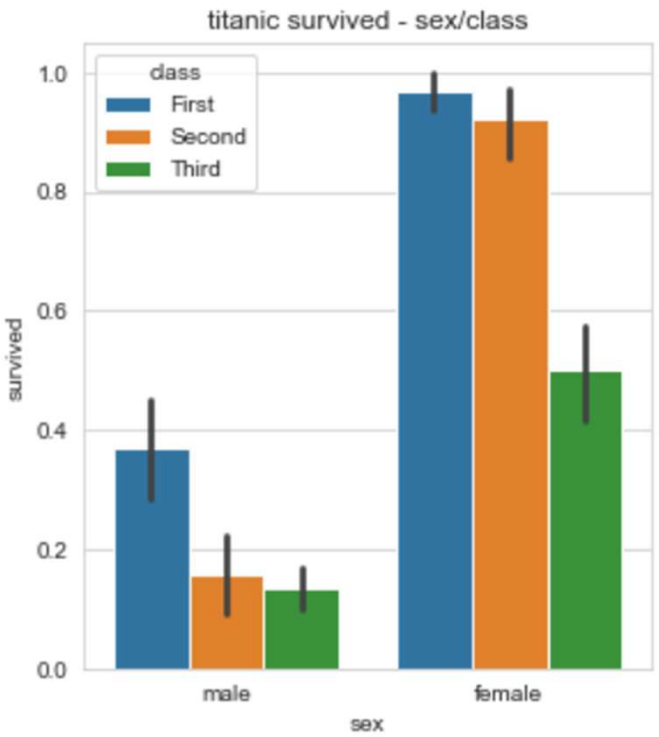
```
# 라이브러리 불러오기 생략
# Seaborn 제공 데이터셋 가져오기 생략
# 스타일 테마 설정 생략

# 그래프 객체 생성 (figure에 3개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 5))
ax1 = fig.add_subplot(1, 3, 1)
ax2 = fig.add_subplot(1, 3, 2)
ax3 = fig.add_subplot(1, 3, 3)

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

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

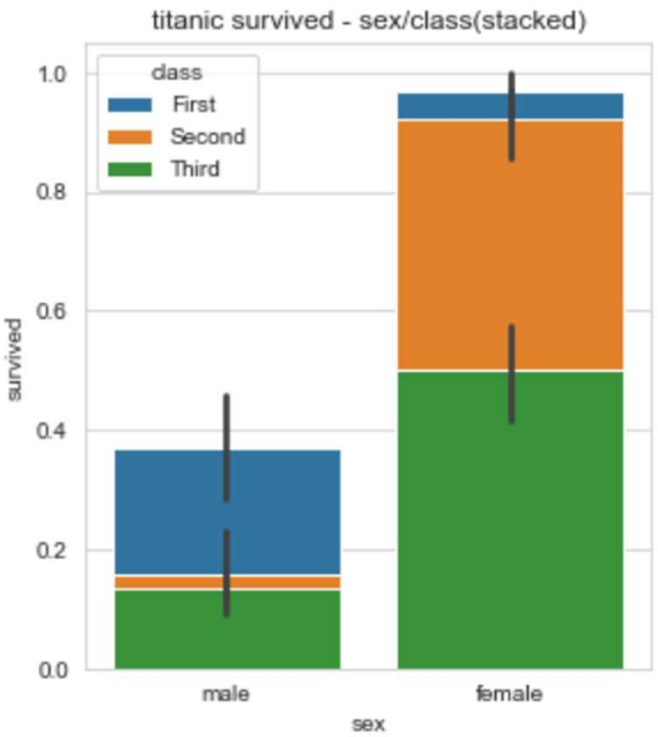
막대 그래프

```
# x축, y축에 변수 할당하고 hue 옵션을 추가하여 누적 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()
```

막대 그래프


titanic survived - sex
survived
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
male
female
sex


titanic survived - sex/class
class
First
Second
Third
survived
0.0
0.2
0.4
0.6
0.8
1.0
male
female
sex


titanic survived - sex/class(stacked)
class
First
Second
Third
survived
0.0
0.2
0.4
0.6
0.8
1.0
male
female
sex

빈도 그래프

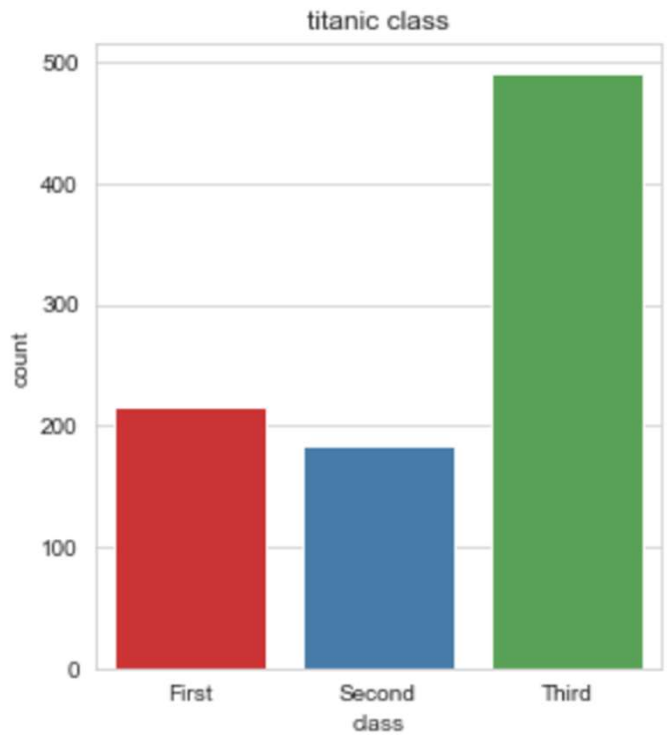
각 범주에 속하는 데이터와 개수를 막대 그래프로 나타내는 countplot() 함수를 살펴보자.
예제를 통해서 3개의 서브 플롯을 비교한다.
“기본설정, hue 옵션 추가, 축 방향으로 hue 변수를 분리하지 않고 위로 쌓아 올리는 누적 그래프로 출력“ 순으로 실행한다.
그래프 색 구성을 다르게 하려면 palette 옵션을 변경하여 적용한다.

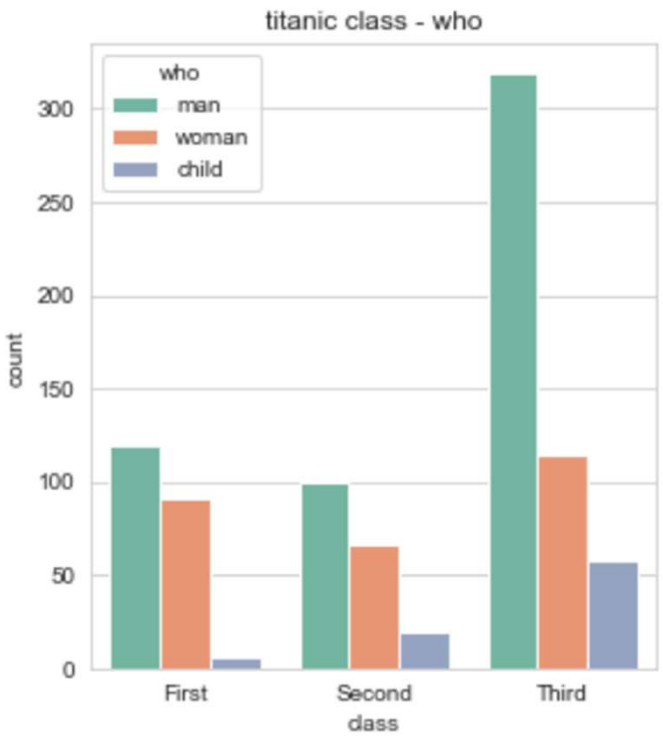
```
# 라이브러리 불러오기 생략
# Seaborn 제공 데이터셋 가져오기 생략
# 스타일 테마 설정 생략

# 그래프 객체 생성 (figure에 3개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 5))
ax1 = fig.add_subplot(1, 3, 1)
ax2 = fig.add_subplot(1, 3, 2)
ax3 = fig.add_subplot(1, 3, 3)

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

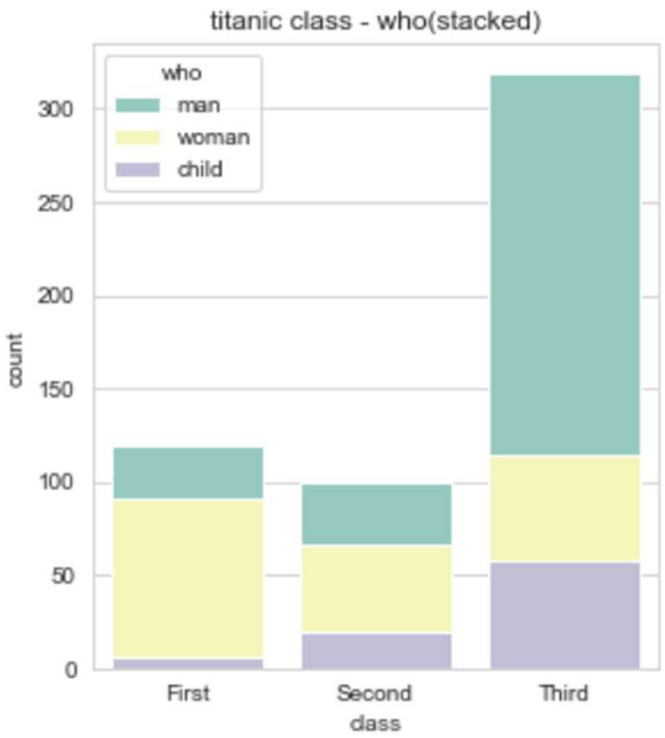
빈도 그래프

```
# hue 옵션에 'who' 추가
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 그래프


titanic class
count
500
400
300
200
100
0
First
Second
Third
class


titanic class - who
who
man
woman
child
count
300
250
200
150
100
50
0
First
Second
Third
class


titanic class - who(stacked)
who
man
woman
child
count
300
250
200
150
100
50
0
First
Second
Third
class

박스 플롯/바이올린 그래프

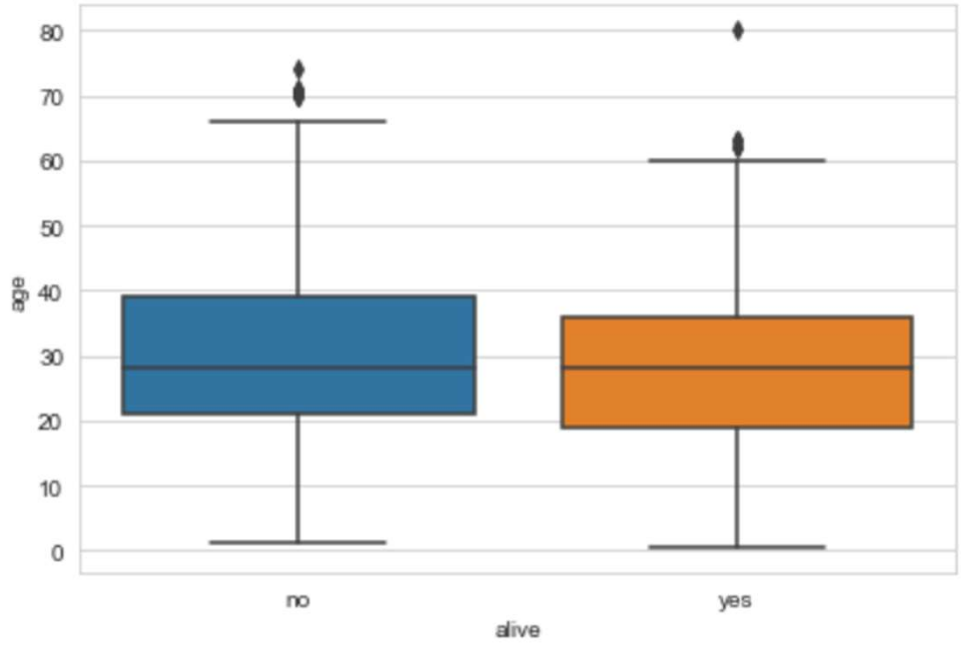
박스 플롯은 범주형 데이터 분포와 주요 통계 지표를 함께 제공한다.
다만 박스 플롯만으로는 데이터가 퍼져 있는 분산의 정도를 정확하게 알기는 어렵기 때문에 커널 밀도 함수 그래프를 y축 방향에 추가하여 바이올린 그래프를 그리는 경우도 있다.
박스 플롯은 boxplot() 함수로 그리고 바이올린 그래프는 violinplot() 함수로 그린다.
예제에서는 타이타닉 생존자의 분포를 파악한다.

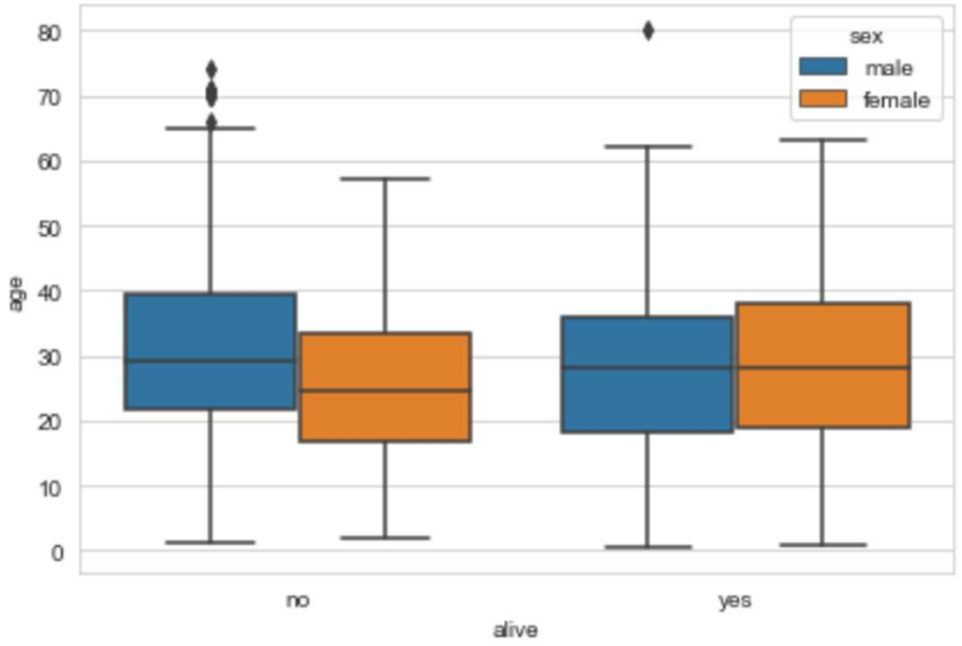
```
# 그래프 객체 생성 (figure에 4개의 서브 플롯을 생성)
fig = plt.figure(figsize=(15, 10))
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(2, 2, 4)

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

# 박스 그래프 - hue 변수 추가
sns.boxplot(x='alive', y='age', hue='sex', data=titanic, ax=ax2)
```

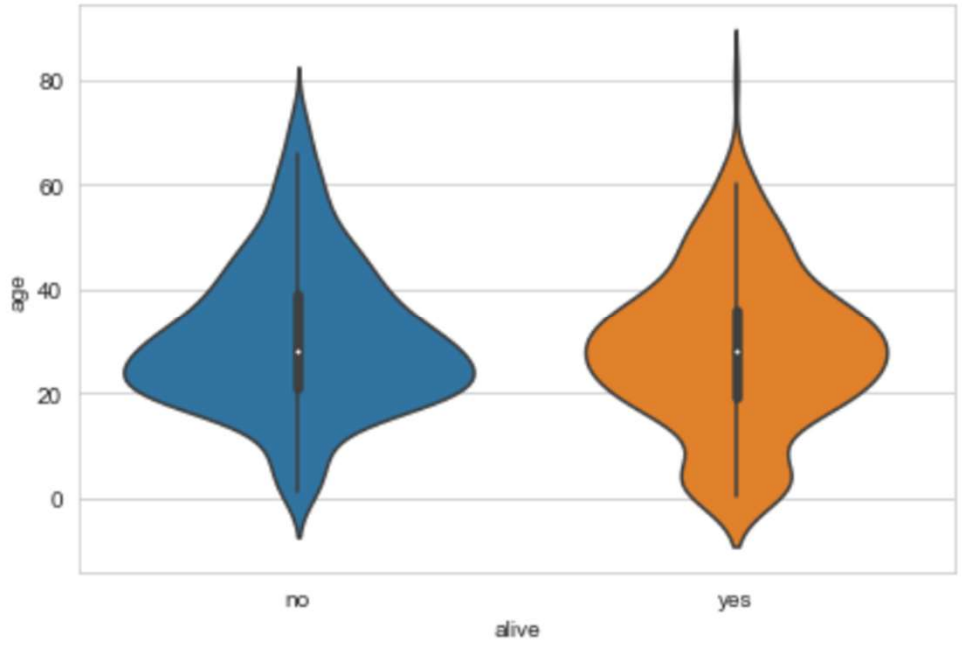
박스 플롯/바이올린 그래프

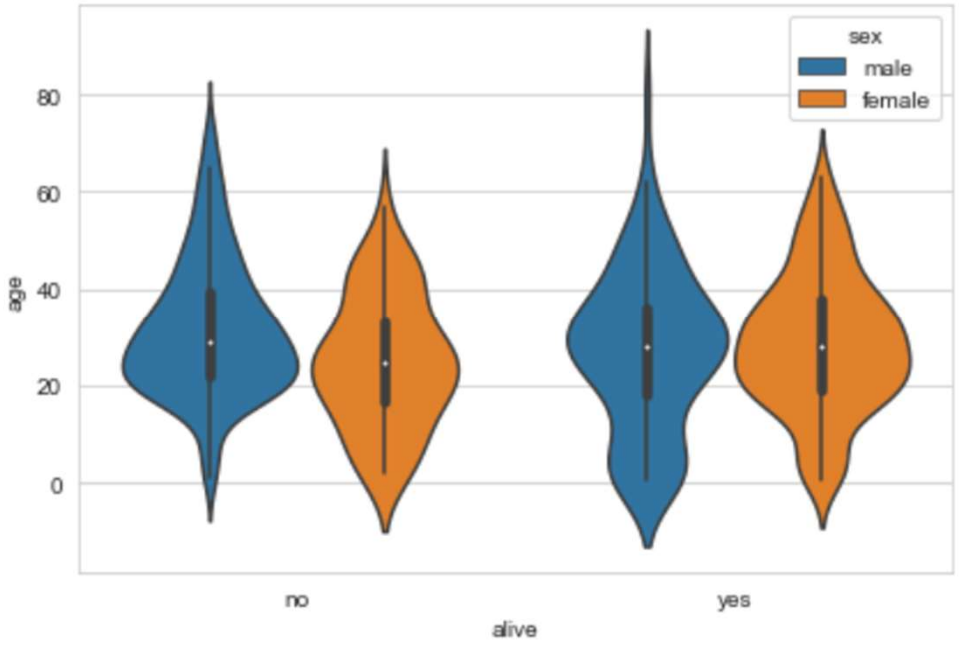
```
# 바이올린 그래프 - 기본값
sns.violinplot(x='alive', y='age', data=titanic, ax=ax3)

# 바이올린 그래프 - hue 변수 추가
sns.violinplot(x='alive', y='age', hue='sex', data=titanic, ax=ax4)

plt.show()
```

박스 플롯/바이올린 그래프







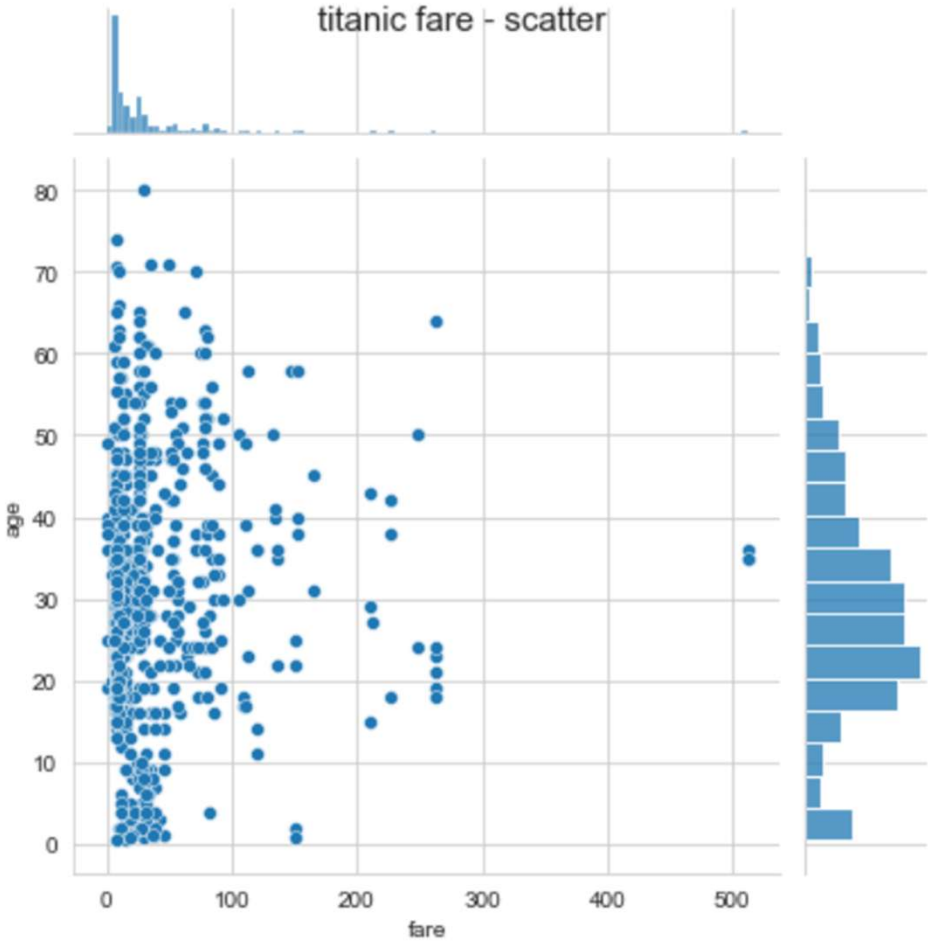


조인트 그래프

jointplot() 함수는 산점도를 기본으로 표시하고, x – y 축에 각 변수에 대한 히스토그램을 동시에 보여준다. 따라서 두 변수의 관계와 데이터가 분산되어 있는 정도를 한눈에 파악하기 좋다.
예제에서는 산점도(기본값), 휘귀선 추가(kind='reg'), 육각 산점도(kind='hex'), 커널 밀집 그래프(kind='kde') 순으로 조인트 그래프를 그리고 차이를 비교해 보자.

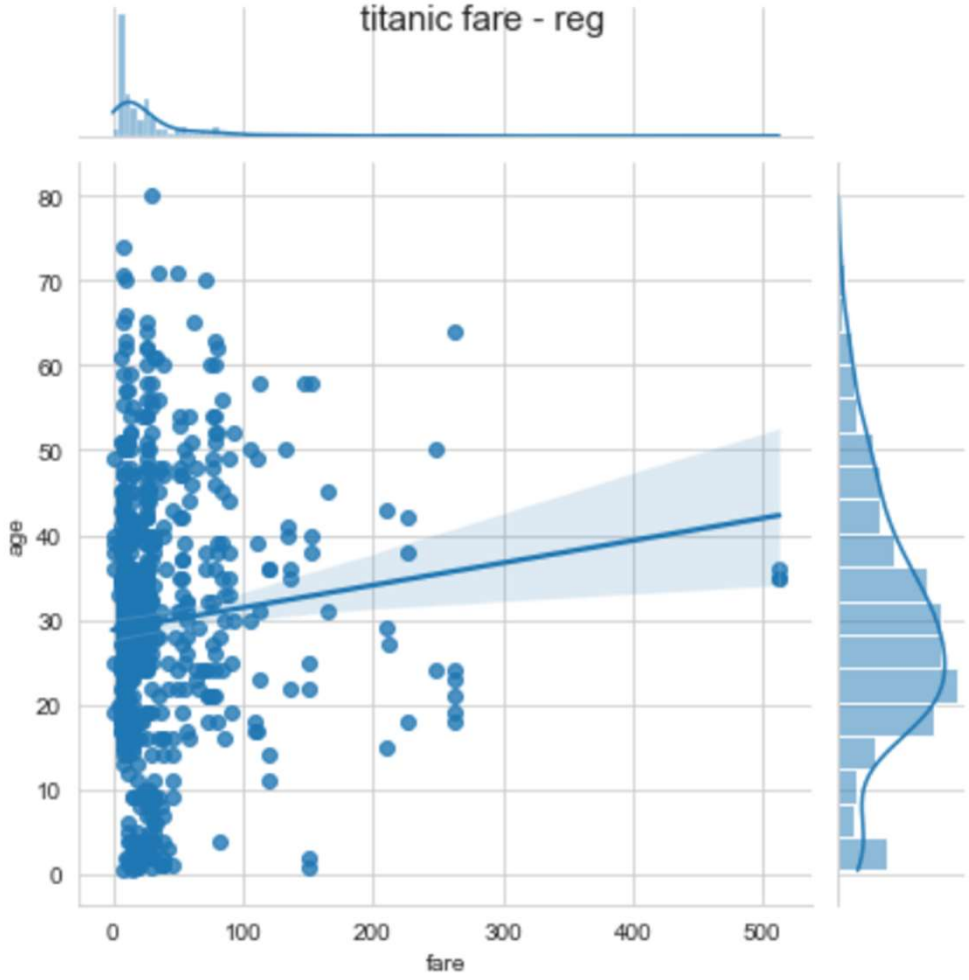
조인트 그래프

```
# 조인트 그래프 - 산점도(기본값)
j1 = sns.jointplot(x='fare', y='age', data=titanic)
j1.fig.suptitle('titanic fare - scatter', size=15)
plt.show()
```



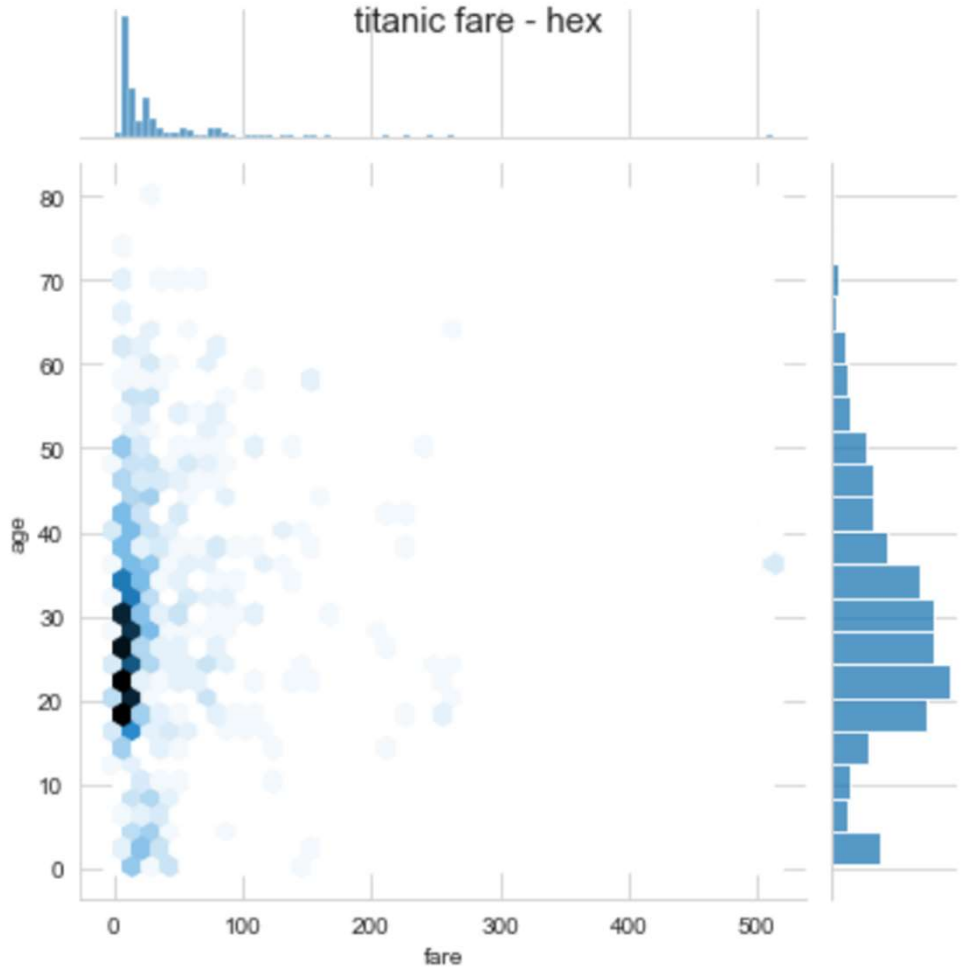
조인트 그래프

```
# 조인트 그래프 - 회귀선
j2 = sns.jointplot(x='fare', y='age', kind='reg', data=titanic)
j2.fig.suptitle('titanic fare - reg', size=15)
plt.show()
```



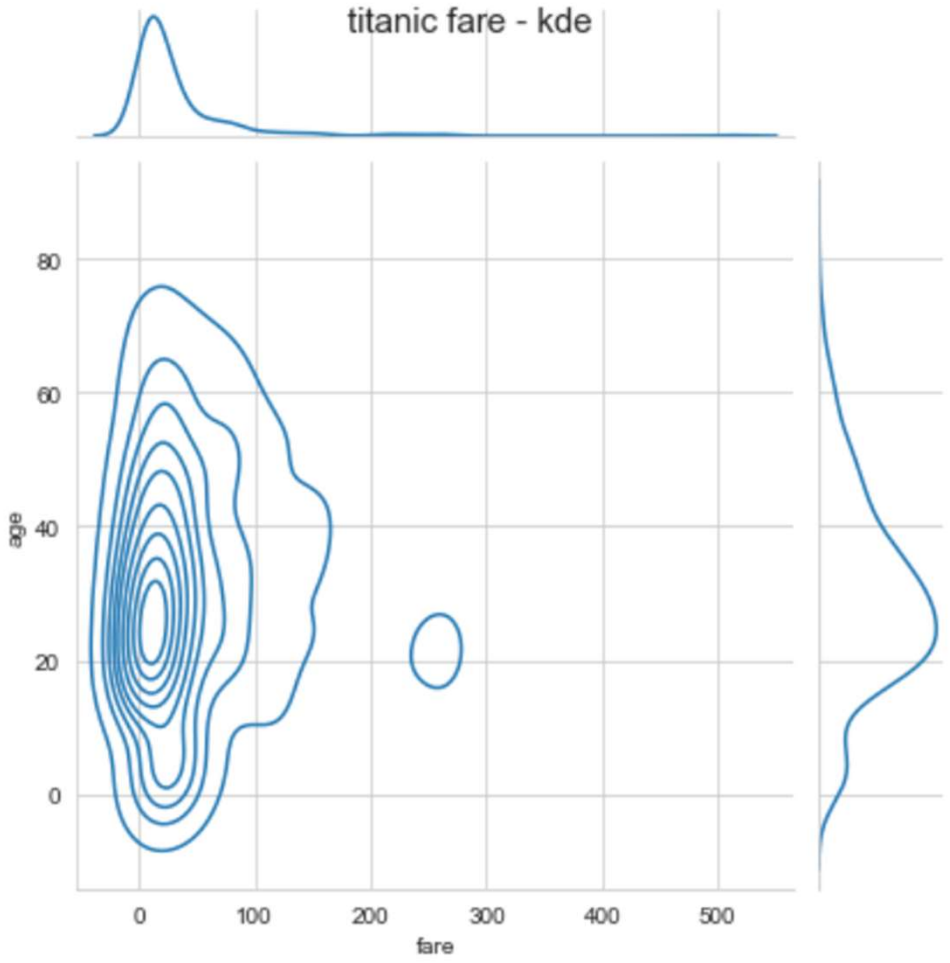
조인트 그래프

```
# 조인트 그래프 - 육각 그래프
j3 = sns.jointplot(x='fare', y='age', kind='hex', data=titanic)
j3.fig.suptitle('titanic fare - hex', size=15)
plt.show()
```



조인트 그래프

```
# 조인트 그래프 - 커럴 밀집 그래프
j4 = sns.jointplot(x='fare', y='age', kind='kde', data=titanic)
j4.fig.suptitle('titanic fare - kde', size=15)
plt.show()
```

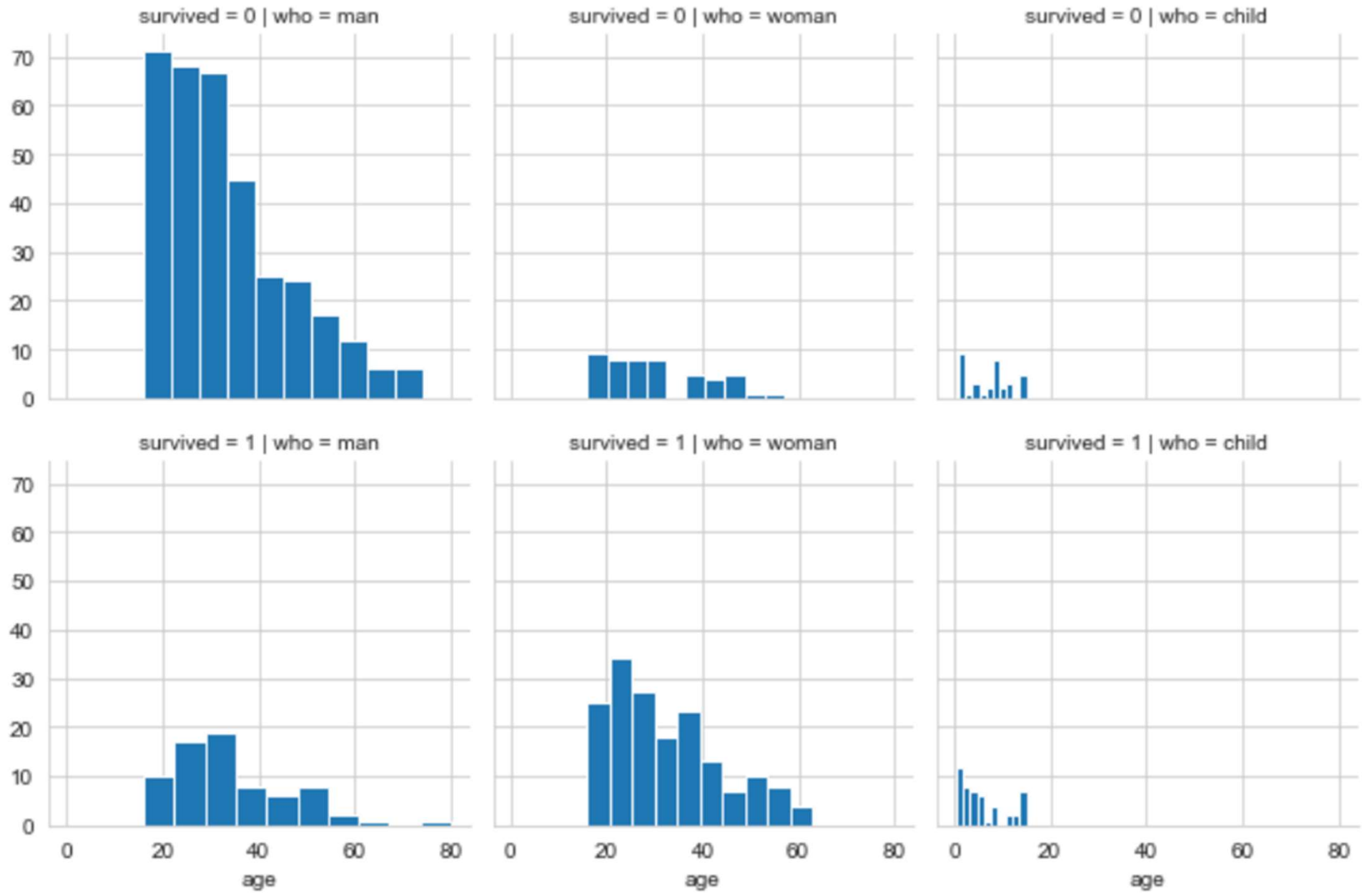


조건을 적용하여 화면을 그리드로 분할하기

FacatGrid() 함수는 행, 열 방향으로 서로 다른 조건을 적용하여 여러 개의 서브 플롯을 만든다.
다음의 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 x 3행 모양의 그리드를 만든다.
각 조건에 맞는 탑승객을 부분하여, ‘age’ 열의 나이를 기준으로 히스토그램을 그려보자.
남성에 비해 여성 생존자가 상대적으로 많은 편이고, 성인 중에서는 활동성이 좋은 20~40대의 생존자가 많은 것으로 나타난다.

```
# 조건에 따라 그리드 나누기
g = sns.FacetGrid(data=titanic, col='who', row='survived')

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

조건을 적용하여 화면을 그리드로 분할하기



이변수 데이터의 분포

pairplot() 함수는 인자로 전달되는 데이터프레임의 열(변수)을 두 개씩 짝을 지을 수 있는 모든 조합에 대해 표현한다. 그래프 그리기 위해 만들어진 짝의 개수만큼 화면을 그리드로 나눈다.
다음의 예제에서는 3개의 열을 사용하기 때문에 3행x3열 크기로 모두 9개의 그리드를 만든다.
각 그리드에 두 변수 간의 관계를 나타내는 그래프를 하나씩 그린다.
같은 변수끼리 KWR을 이루는 대각선 방향으로는 히스토그램을 그리고 서로 다른 변서 간에는 산점도를 그린다.

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

# 조건에 따라 그리드 나누기
g = sns.pairplot(titanic_pair)
```

이변수 데이터의 분포