

# 기술 통계

# 공통 코드

## ❖ 공통 코드

```
import numpy as np
import pandas as pd

#시각화 패키지
import matplotlib.pyplot as plt

#시각화에서 한글을 사용하기 위한 설정
import platform
from matplotlib import font_manager, rc

if platform.system() == 'Darwin':
    rc('font', family='AppleGothic')
#윈도우의 경우
elif platform.system() == 'Windows':
    font_name =
    font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
    rc('font', family=font_name)

#시각화에서 음수를 표현하기 위한 설정
import matplotlib
matplotlib.rcParams['axes.unicode_minus'] = False
```

# 공통 코드

## ❖ 공통 코드

```
# Jupyter Notebook의 출력을 소수점 이하 3자리로 제한  
%precision 3  
# precision은 소수점은 과학적 표기법으로 변환할 자릿수를 설정  
# 아래와 같이 하면 소수점 셋째 자리 밑으로는 과학적 표기법으로 표시  
pd.options.display.precision = 3
```

# 통계학

## ❖ 통계학

- ✓ 통계학이란 논리적 사고와 객관적인 사실에 따르며 일반적이고 확률적 결정론에 따라서 인과 관계를 규명
- ✓ 연구 목적에 의해 설정된 가설들에 대하여 분석 결과가 어떤 결과를 뒷받침하고 있는지를 통계적 방법으로 검정
- ✓ 통계학은 수집된 자료의 특성을 쉽게 파악하기 위해서 자료를 표, 그래프, 대표 값 등으로 정리 요약해서 다루는 기술 통계학과 모집단에서 추출한 정보를 이용하여 모집단의 다양한 특성(회귀분석, T-검정, 분산분석 등)을 과학적으로 추론하는 추론 통계학으로 분류

# 데이터

## ❖ 데이터 읽어오기 – sports\_test.csv

```
# 학생번호를 인덱스로 csv 파일을 읽어들여, 변수 df에 저장  
df = pd.read_csv('./data/sport_test.csv', index_col='학생번호')  
# 변수 df를 출력  
df
```

학년	학생번호	악력	윗몸일으키기	점수	순위
1	1	40.2	34	15	4
2	1	34.2	14	7	10
3	1	28.8	27	11	7
4	2	39.0	27	14	5
5	2	50.9	32	17	2
6	2	36.5	20	9	9
7	3	36.6	31	13	6
8	3	49.2	37	18	1
9	3	26.0	28	10	8
10	3	47.4	32	16	3

# 데이터

## ❖ 특정 열 추출

```
#Series 로 리턴  
df['악력']
```

학생번호

1	40.2
2	34.2
3	28.8
4	39.0
5	50.9
6	36.5
7	36.6
8	49.2
9	26.0
10	47.4

Name: 악력, dtype: float64

# 데이터

## ❖ 특정 열 추출

#DataFrame 으로 리턴  
df[['악력']]

학생번호	악력
1	40.2
2	34.2
3	28.8
4	39.0
5	50.9
6	36.5
7	36.6
8	49.2
9	26.0
10	47.4

# 데이터

## ❖ 특정 열 추출

```
print(type(df['악력']))  
print(type(df[['악력']]))
```

```
<class 'pandas.core.series.Series'>  
<class 'pandas.core.frame.DataFrame'>
```

# 데이터

- ❖ 데이터의 크기 확인

`df.shape`

(10, 5)

# 데이터

## ❖ 기술 통계

- ✓ 자료를 요약하는 기초적인 통계량
- ✓ 데이터 분석 전에 전체적인 데이터 분포의 이해와 통계적 수치를 제공
- ✓ 모집단의 특성을 유추하는데 이용
- ✓ 설문 조사를 시행한 논문에서는 응답자의 일반적인 특성을 반드시 제시
- ✓ 빈도 분석: 설문조사 결과에 대한 가장 기초적인 정보를 제공해주는 분석 방법으로 광범위하게 이용하는데 성별이나 직급을 수치화하는 명목 척도나 서열 척도 같은 범주형 데이터를 대상으로 비율을 측정하는데 주로 이용
- ✓ 기술 통계 분석: 빈도 분석과 유사하지만 등간 척도나 비율 척도와 같은 연속적인 데이터를 분석할 때 주로 이용하는데 유형으로는 대푯값(분포의 중심 위치 측정), 산포도(자료가 대푯값으로 부터 얼마나 흩어져 분포하는 가는 보여주는 값), 비대칭도(분포가 기울어진 방향과 정도를 나타내는 왜도 와 분포도가 얼마나 중심에 집중되어 있는 가를 나타내는 첨도 등)

# 데이터

## ❖ pandas의 기술 통계 함수

- ✓ count, min, max, sum, mean, median, var, std
- ✓ argmin(최소값 위치), argmax, idxmin(최소값 색인), idxmax, quantile(사분위수)
- ✓ describe(요약) – Series로 리턴
- ✓ cumsum(누적합), cumin, cummax, comprod
- ✓ diff(산술 차)
- ✓ pct\_change: 이전 데이터와의 차이로 -1을 대입하면 반대로
- ✓ unique(): 동일한 값을 제외한 배열 리턴 – Series에만 사용
- ✓ value\_counts(): 도수를 리턴, 기본적으로 내림차순 정렬을 수행하며 sort 속성에 False를 대입하면 정렬하지 않습니다. – Series 에만 사용

# 데이터

## ❖ 변수의 종류

### ✓ 질적 변수 와 양적 변수

- 변수는 크게 질적 변수와 양적 변수 두 가지로 분류할 수 있음
- 질적 변수
  - ◆ 설문조사에서 만족도 등을 묻는 질문에 대해 1.매우 좋음 2.좋음 3.보통 4.나쁨 5.매우 나쁨 과 같이 선택이 필요한 변수
  - ◆ 혈액형을 조사할 때 A형 B형 O형 AB형 같이 종류를 구별하기 위한 변수
  - ◆ 질적 변수 중에서도 남성과 여성, 흡연 여부 등 값이 2개뿐인 질적 변수를 2진 변수라 부르기도 함
- 양적 변수: 시험 점수나 신장과 같이 양을 표현하는 변수
- 남성이나 여성은 질적 변수이지만 데이터상에서 쉽게 처리하기 위해 남성은 0, 여성은 1과 같이 숫자 데이터로 표시하는 경우가 있기 때문에 수치 변수라 해서 전부 양적 변수인 것은 아님

# 데이터

## ❖ 변수의 종류

### ✓ 척도(Scale) 수준

- 명목(nominal, 명의) 척도, 순서(ordinal, 서열) 척도, 등간(interval, 간격) 척도, 비율(ratio, 비례) 척도 네 가지가 척도 수준
- 질적 변수: 명목 척도 와 순서 척도
- 양적 변수: 등간 척도 와 비율 척도

# 데이터

## ❖ 변수의 종류

### ✓ descriptive.csv

- 부모의 학력 수준에 따른 자녀의 대학진학 합격 여부를 조사한 데이터 셋
- 300개의 행과 8개의 컬럼으로 구성
- 컬럼
  - ◆ resident: 거주지역(1,2,3 – 특별시, 광역시, 시군)
  - ◆ gender: 성별(1,2 – 남, 여)
  - ◆ age: 나이
  - ◆ level: 학력 수준(1,2,3 – 고졸, 대졸, 대학원졸 이상)
  - ◆ cost: 생활비
  - ◆ type: 학교 유형(1,2)
  - ◆ survey: 만족도(1-5)
  - ◆ pass: 합격여부(1,2)

# 데이터

## ❖ 변수의 종류

### ✓ 데이터 읽어오기

-----데이터 읽기 -----

	resident	gender	age	level	cost	type	survey	pass
0	1	1	50	1	5.1	1	1.0	2
1	2	1	54	2	4.2	1	2.0	2
2		1	62	2	4.7	1	1.0	1
3	4	2	50		3.5	1	4.0	1
4	5	1	51	1	5	1	3.0	1

-----데이터 특성 확인 -----

(300, 8)

-----데이터 특성 확인 -----

	gender	age	survey
count	300.000000	300.000000	187.000000
mean	1.420000	53.880000	2.609626
std	0.545826	6.813247	0.974135
min	0.000000	40.000000	1.000000
25%	1.000000	48.000000	2.000000
50%	1.000000	53.000000	3.000000
75%	2.000000	60.000000	3.000000
max	5.000000	69.000000	5.000000

# 데이터

## ❖ 변수의 종류

### ✓ 데이터 읽어오기

```
university = pd.read_csv("./data/descriptive.csv", encoding='ms949')
print('-----데이터 읽기 -----')
print(university.head())
print('-----데이터 특성 확인 -----')
print(university.shape)
print('-----데이터 특성 확인 -----')
print(university.describe())
```

# 데이터

- ❖ 변수의 종류
  - ✓ 명목 척도
    - 구별 만을 위해서 의미 없는 수치로 구성한 데이터
    - 거주 지역이나 성별과 같은 명목 척도를 대상으로 요약 통계량을 구하는 것은 의미가 없지만 성별의 구성 비율은 표본의 통계량으로 의미가 있음

# 데이터

- ❖ 변수의 종류
  - ✓ 명목 척도

```
print('--의미없는 요약 통계량--')  
print(university['gender'].describe())
```

```
print('--구성 비율--')  
print(university['gender'].value_counts())
```

# 데이터

- ❖ 변수의 종류
  - ✓ 명목 척도

--의미없는 요약 통계량--

count 300.000000

mean 1.420000

std 0.545826

min 0.000000

25% 1.000000

50% 1.000000

75% 2.000000

max 5.000000

Name: gender, dtype: float64

--구성 비율--

1 173

2 124

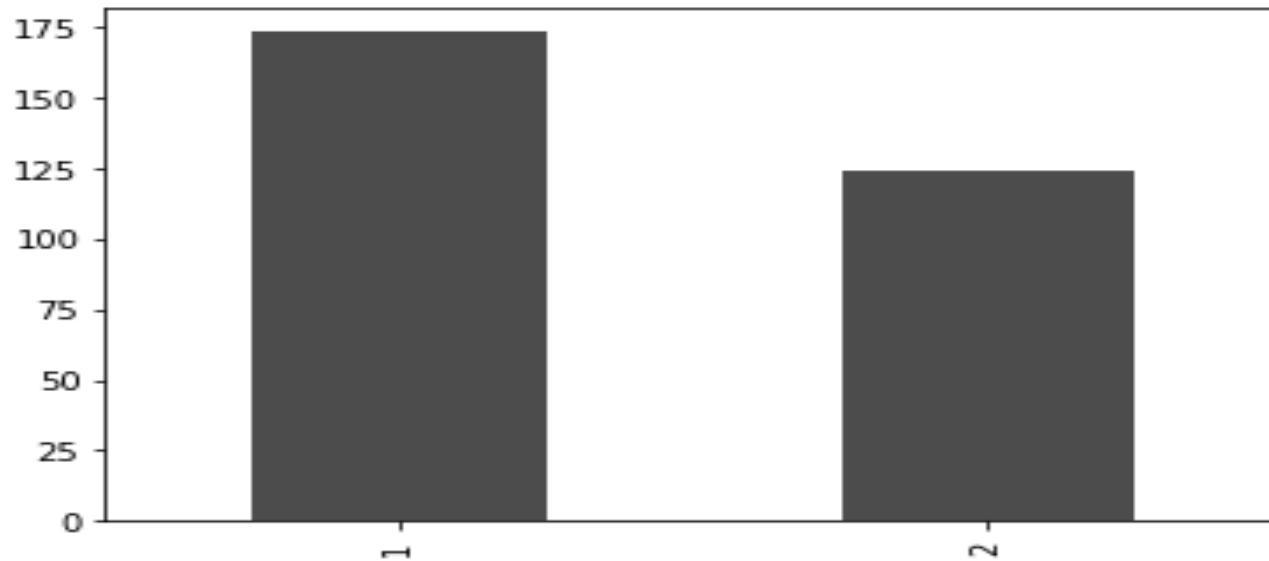
0 2

5 1

Name: gender, dtype: int64

# 데이터

- ❖ 변수의 종류
    - ✓ 명목 척도
      - 1 173
      - 2 124
- Name: gender, dtype: int64



# 데이터

- ❖ 변수의 종류
  - ✓ 명목 척도

```
#데이터 정제  
university_gender = university[(university['gender'] == 1) | (university['gender']  
==2)]  
print(university_gender['gender'].value_counts())  
#범주형 데이터 시각화  
university_gender['gender'].value_counts().plot.bar(color='k', alpha=0.7)
```

# 데이터

## ❖ 변수의 종류

### ✓ 순서 척도

- 순서를 정하기 위해서 만들어진 수치 데이터
- 계급 순위를 수치로 표현한 직급이나 학력 수준 등과 같은 순서 척도 변수를 대상으로 기초 통계량을 구하는 경우 최대값이나 최소값 및 평균 등은 큰 의미는 없지만 빈도 수는 의미가 있음

#순서 척도 시각화

```
print(university_gender['level'].value_counts())
```

```
plt.figure()
```

```
university_gender['level'].value_counts().plot.bar(color='k', alpha=0.7)
```

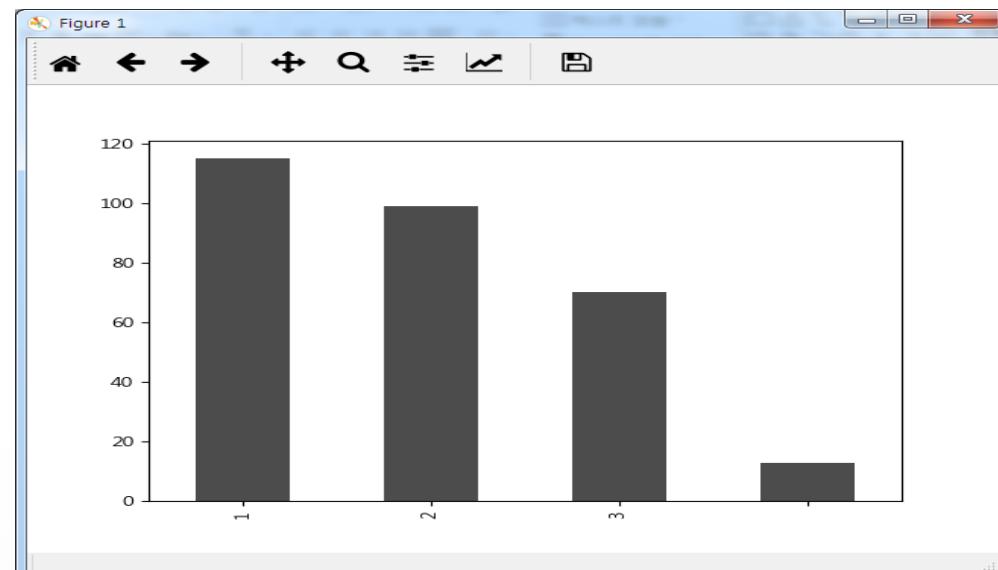
```
plt.show()
```

# 데이터

- ❖ 변수의 종류
  - ✓ 순서 척도

1	115
2	99
3	70
	13

Name: level, dtype: int64



# 데이터

## ❖ 변수 종류

### ✓ 등간 척도

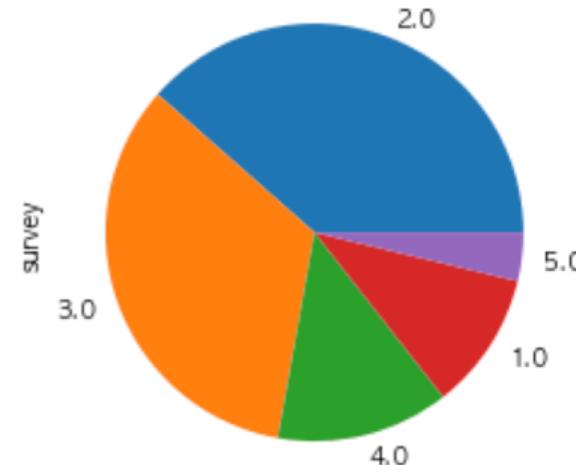
- 속성의 간격이 일정한 값을 갖는 척도
- 예를 들면 귀하는 교육 내용에 만족하십니까? 라는 질문에 1-매우 만족, 2-만족, 3-보통 의 범주에서 응답을 얻은 다음 가중치를 적용하여 가감산하여 응답자의 생각을 측정
- 요약 통계량이 의미가 있음

```
plt.figure()
print(university_gender['survey'].describe())
#등간 척도 데이터 시작화
university_gender['survey'].value_counts().plot.pie()
plt.show()
```

# 데이터

- ❖ 변수 종류
  - ✓ 등간 척도

```
count    187.000000  
mean     2.609626  
std      0.974135  
min      1.000000  
25%     2.000000  
50%     3.000000  
75%     3.000000  
max      5.000000  
Name: survey, dtype: float64
```



# 데이터

## ❖ 변수 종류

### ✓ 비율 척도

- 등간 척도의 특성에 절대 원점이 존재하는 척도를 비율 척도 변수라고 하는데 응답자가 직접 수치로 입력한 변수로 속성이 0을 기준으로 한 수치로 되어 있기 때문에 사칙 연산이 가능한 척도로 비율 척도라고 함
- 빈도분석과 기술 통계량 등 가장 많은 표본의 통계량을 사용
- 대표적인 예는 성적, 키, 나이, 무게 등
- cost는 비율 척도

# 데이터

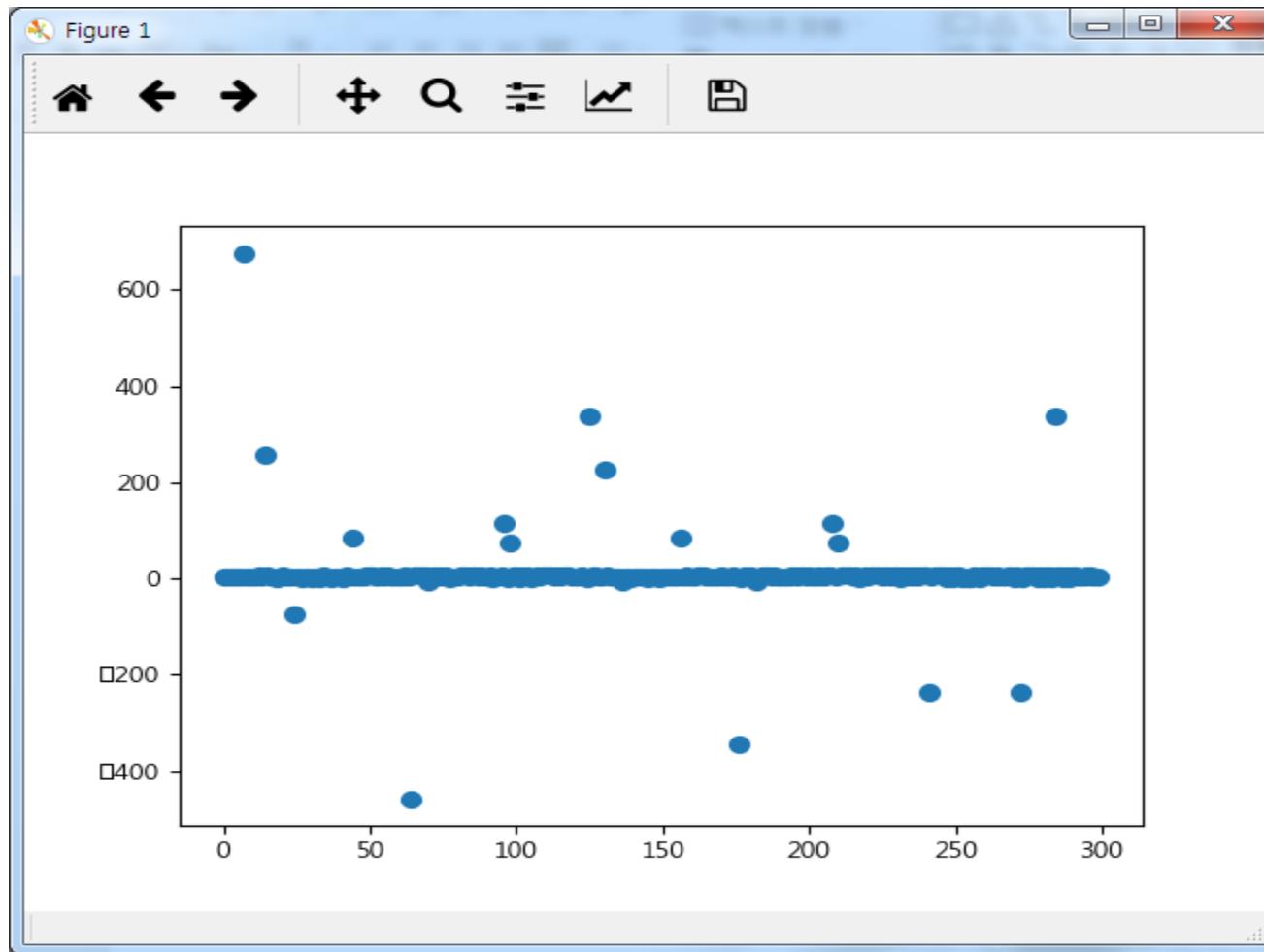
- ❖ 변수 종류
  - ✓ 비율 척도

```
plt.figure()  
print(university_gender['cost'].value_counts())  
plt.scatter(university_gender.index, university_gender['cost'], s=50)  
plt.show()
```

5.0	18
6.3	16
4.0	15
6.0	14
6.2	13
6.4	11
5.1	10
5.2	9
4.1	9
6.1	8
6.7	8
5.5	8

# 데이터

- ❖ 변수 종류
- ✓ 비율 척도



# 데이터

## ❖ 변수 종류

### ✓ 비율 척도

- 비율 척도를 대상으로 직접 빈도 분석을 수행한 결과는 큰 의미가 없을 수 있음
- Cost 값은 응답자가 직접 입력하는 수치이기 때문에 동일한 수치의 빈도수를 나타내는 것은 의미가 없고 일정한 간격으로 범주화하는 것이 필요

#범주화

```
cost = university_gender['cost']
print(cost[(cost>=2)&(cost<=10)].describe())
print(cost[(cost>=2)&(cost<=10)].value_counts())
```

# 데이터

- ❖ 변수 종류
  - ✓ 비율 척도

```
count    248.000000
mean     5.354032
std      1.138783
min      2.100000
25%     4.600000
50%     5.400000
75%     6.200000
max     7.900000
Name: cost, dtype: float64
```

# 데이터

- ❖ 변수 종류
  - ✓ 비율 척도

5.0	18
6.3	16
4.0	15
6.0	14
6.2	13
6.4	11
5.1	10

# 데이터

## ❖ 변수 종류

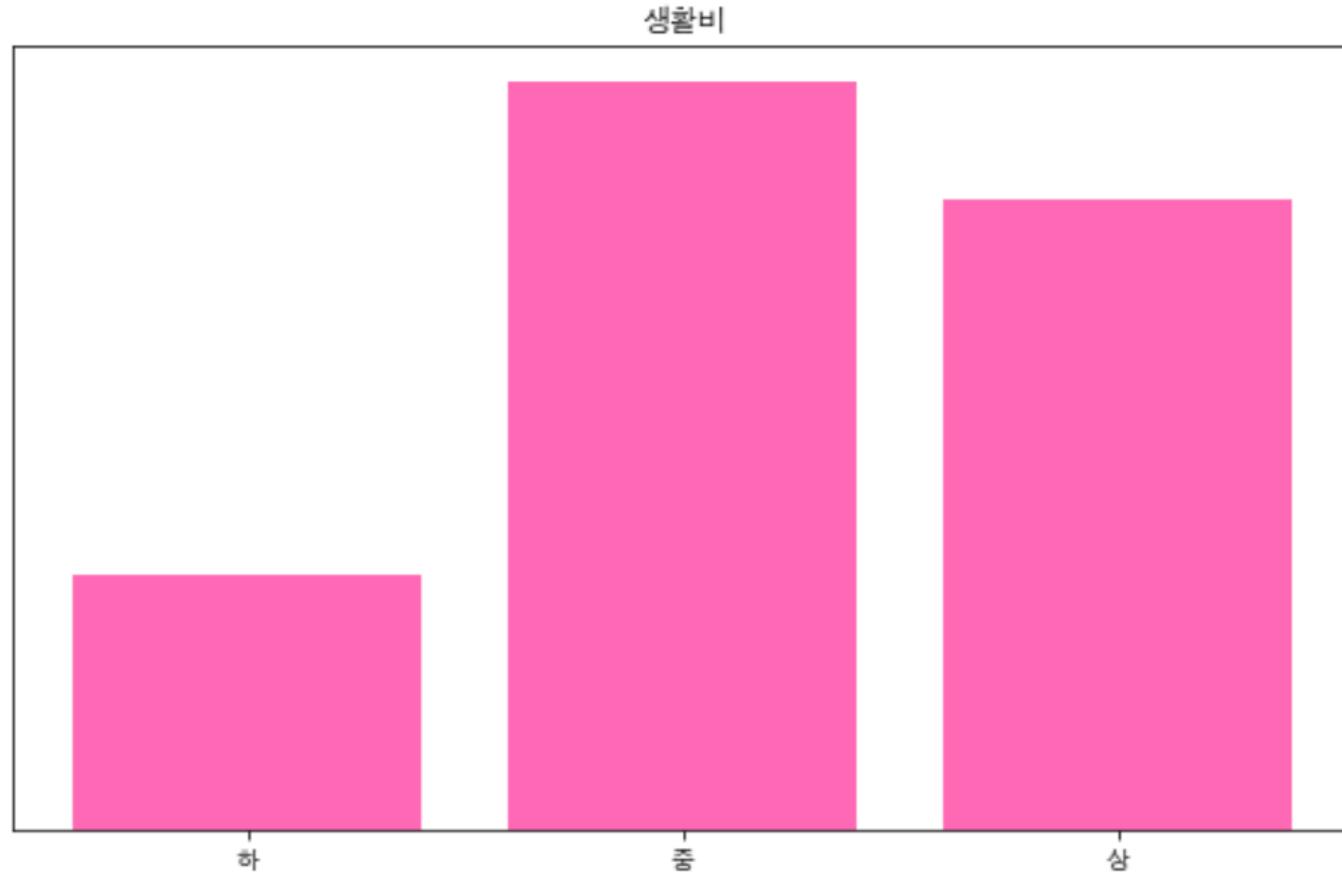
### ✓ 비율 척도

```
plt.figure(figsize=(10, 6))
## histogram의 경우 내가 값 리스트를 넣고, 입력한 bin 개수에 따라 알아서 분류해
## 줌
## ys: y값,
## xs: x 값
ys, xs, patches = plt.hist(cost[(cost>=2)&(cost<=10)],
bins=3, ## 몇 개의 바구니로 구분할 것인가.
density=True, ## ytick을 퍼센트비율로 표현해줌
cumulative=False, ## 누적으로 표현하고 싶을 때는 True
histtype='bar', ## 타입. or step으로 하면 모양이 바뀜.
orientation='vertical', ## or horizontal
rwidth=0.8, ## 1.0일 경우, 꽉 채움 작아질수록 간격이 생김
color='hotpink', ## bar 색깔
)
```

```
plt.yticks([])## text로 표시하고 있기 때문에 이 부분은 삭제해줌
## xticks를 변경해줌
plt.xticks([(xs[i]+xs[i+1])/2 for i in range(0, len(xs)-1)],
["하", "중", "상"])
plt.title('생활비')
plt.show()
```

# 데이터

- ❖ 변수 종류
- ✓ 비율 척도



# 데이터

## ❖ 변수 종류

### ✓ 비율 척도

```
plt.figure()
```

```
cost = cost[(cost>=1)&(cost<=10)]
```

```
cost[(cost>=1)&(cost<=3)] = 1
```

```
cost[(cost>3)&(cost<=6)] = 2
```

```
cost[(cost>6)] = 3
```

```
cost = cost.astype(int)
```

```
label = ["하", "중", "상"]
```

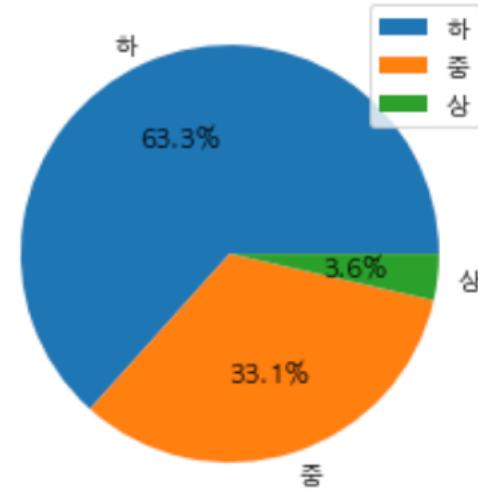
```
plt.pie(cost.value_counts(), labels=label, autopct='%.1f%%')
```

```
plt.title('생활비 분포')
```

```
plt.legend()
```

```
plt.show()
```

생활비 분포



# 데이터

## ❖ 변수 종류

### ✓ 이산형 변수와 연속형 변수

- 이산형 변수는 0, 1, 2, ... 와 같이 하나 하나의 값을 취하는 변수로 서로 인접한 숫자 사이에 값이 존재하지 않는 변수로 예를 들면 주사위의 눈은 1, 2, 3, 4, 5, 6 이라는 값을 취하고 1.3이라는 값을 취하지 않으므로 이산형 변수로 분류
- 연속형 변수는 연속적인 값을 취할 수 있는 변수로 어떤 두 숫자 사이에도 반드시 숫자가 존재하는데 길이나 무게, 시간 등은 대표적인 연속형 변수
- 연속형 변수라도 측정 정밀도에 한계가 있어 띄엄띄엄 떨어진 값을 취할 수밖에 없는데 예를 들어 키를 소수점 첫 번째 자리까지의 정밀도로 측정하면 170.3cm와 170.4cm 사이에는 숫자가 존재하지 않아 엄밀하게 말하면 이산형 변수로 분류되지만 이와 같이 측정 정밀도의 문제로 이산형이 되는 변수는 연속형 변수로 취급

# 데이터의 특성

## ❖ 데이터의 특성 파악

- ✓ 통계 분석의 시작은 데이터를 정리하여 데이터의 특징을 대략적으로 파악하는 것
- ✓ 데이터의 개요를 파악하면 수많은 통계 분석 기법 중에서 적절한 기법을 선택해 다음 걸음을 내딛을 수 있음
- ✓ 데이터의 특징을 파악하는 방법
  - 평균이나 분산 등의 수치 지표에 따라 데이터를 요약
  - 시각적으로 데이터를 파악

# 데이터의 특성

## ❖ 대푯값(Typical Value)

- ✓ 데이터를 중심으로 나타내는 지표로 데이터를 하나의 값으로 요약한 지표
- ✓ 평균(mean): 모든 값의 총합을 개수로 나눈 값으로 보통 산술 평균(Average)을 의미
  - 산술 평균: 모든 값의 총합을 개수로 나눈 값
  - 기하 평균: 평균 비율을 구할 때 사용하는 평균으로 각 데이터의 비율을 곱한 후 제곱근을 구하는 것

매출액이 100 인 회사에서 다음 해의 매출이 110을 기록했고 그 다음해의 매출이 107.8을 기록했을 때 연 평균 성장률은?

첫 해 10프로 인상됐고 두 번째 해에 2프로 감소 했으므로  $(10-2)/2 = 4$ 프로 ?

1.1 \* 0.98의 제곱근

- 조화 평균: 2개의 수의 곱에 2를 곱한 후 2개의 수를 더한 값으로 나누는 것으로 속도의 평균을 구할 때 사용

동일한 거리를 시속 100km로 한 번 가고 60km로 갔을 때 평균 속도는?

$$2*100*60/100+60$$

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 평균(mean)

- 매출액이 10 인 상태에서 다음에는 11 그리고 그 다음에는 10.78 인 경우의 평균 성장률 계산

평균 성장률?: 0.040000000000000036

평균 성장률(기하평균): 1.0382677881933928

- 동일한 거리를 한번은 시속 100으로 한번은 시속 60으로 갔을 때의 평균 속도

평균 속도?: 80.0

평균 속도(조화평균): 75.0

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 평균(mean)

#기하평균 과 조화평균

```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import math
s = Series([10, 11, 10.78])
print("평균 성장률?:", s.pct_change().mean())
print("평균 성장률(기하평균):", math.sqrt((11/10)*(10.78/11)))

print("평균 속도?:", (100+60) / 2)
print("평균 속도(조화평균):", 2*100*60 / (100+60))
```

평균 성장률?: 0.04000000000000036

평균 성장률(기하평균): 1.0382677881933928

평균 속도?: 80.0

평균 속도(조화평균): 75.0

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 평균(mean)

- 가중 평균(Weighted Mean): 가중치를 곱한 값의 총합을 가중치의 총합으로 나눈 값
  - ◆ 데이터를 수집할 때 모든 사용자 그룹에 대해서 정확히 같은 비율을 반영하는 데이터를 수집하는 것은 어렵기 때문에 데이터가 부족한 소수 그룹에 대해 더 높은 가중치를 부여해서 적용
- 특잇값(outlier, 이상치): 대부분의 값과 매우 다른 데이터 값(극단값)
- 로버스트(robust): 극단값들에 민감하지 않는 것을 의미하는데 저항성(resistant)이 있다고도 함
- 절사 평균(trimmed mean): 정해진 개수의 극단 값을 제외한 나머지 값들의 평균
  - ◆ 5명의 심판이 매긴 점수 중에서 가장 큰 값과 가장 작은 값을 제외한 3개의 심판이 매긴 점수의 평균 – 극단치의 영향을 적게 받기 위한 평균

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 중앙값(median)

- 중앙값은 데이터를 크기 순서대로 나열할 때 정확하게 중앙에 위치한 값
- 중앙값은 평균값에 비해 이상 값에 강하다는 특성이 있는데 이상값에 영향을 덜 받는다는 의미
- [1, 2, 3, 4, 5, 6, 1000] 이라는 큰 이상값을 포함한 데이터가 있다고 가정해보면 이 데이터의 대푯값을 구하려고 할 때 평균값은 1000이라는 값에 크게 영향을 받아 150에 가까운 값이 되어버리는데 150이라는 수치는 이 데이터를 적절하게 표현한다고 할 수 없는데 데이터의 개수가 7이므로 4번째인 4가 중앙값이 되는데 이와 같이 데이터에 큰 이상값이 있는 경우 대푯값으로 평균값보다 중앙값이 적절
- 중앙값은 데이터를 순서대로 나열할 때 정확하게 중앙에 위치하는 값이지만 데이터의 개수가 짝수일 때는 중앙에 위치하는 값이 2개가 되는데 [1, 2, 3, 4, 5, 6] 이라는 데이터를 생각해보면 이 데이터의 중앙에 위치하는 값은 3과 4가 되고 이 경우 중앙값은 두 값의 평균값으로 정의되므로 이 데이터의 중앙값은 3.5

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 평균과 중앙값

```
from scipy import stats
tdata = pd.read_csv('./data/tdata.csv', encoding='cp949')
print('평균:', tdata['성적'].mean())
print('중앙값:', tdata['성적'].median())
#상위와 하위 10% 잘라내고 평균 구하기
print('절사평균:', stats.trim_mean(tdata['성적'], 0.1))
```

평균: 77.1

중앙값: 77.5

절사평균: 77.0

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 평균과 중앙값

```
state = pd.read_csv("./data/state.csv")
print("평균:", state['Population'].mean())
print("절사 평균:", stats.trim_mean(state['Population'], 0.1))
print("중앙값:", state['Population'].median())
```

평균: 6162876.3

절사 평균: 4783697.125

중앙값: 4436369.5

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 평균과 중앙값

#가중 평균이나 가중 중앙값 구하기

#wquantiles 패키지 설치

import wquantiles

#가중 평균을 구하기 위해서는 numpy 의 average 함수를 이용

```
print(state['Murder.Rate'].mean())
```

```
print(np.average(state['Murder.Rate'], weights=state['Population']))
```

#가중 중앙값은 wquantiles 패키지의 median을 이용 - 설치 해야 함

```
print(wquantiles.median(state['Murder.Rate'], weights=state['Population']))
```

4.066

4.445833981123393

4.4

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 최빈값(mode)

- 데이터에서 가장 많이 나타나는 값
- 최빈값은 DataFrame이나 Series의 mode 메서드를 사용하여 구할 수 있음
- 최빈값은 질적 데이터의 대푯값을 구할 때 사용하며 시험 점수와 같은 양적 데이터에서는 최빈값을 구하려고 해도 완전히 동일한 점수가 여러 번 나오는 경우가 거의 없어 유일한 값이 결정되지 않을 때가 많기 때문에 사용하기 어려움

# 데이터의 특성

## ❖ 대푯값(Typical Value)

### ✓ 최빈값(mode)

#최빈값

```
print(pd.Series([1, 2, 3, 4, 5]).mode())
print(state['Murder.Rate'].mode())
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
0    1.6
1    2.0
2    5.7
dtype: float64
```

# 데이터의 특성

## ❖ 변이 추정

- ✓ 변이 추정은 데이터 값이 얼마나 밀집해 있는지 혹은 퍼져 있는지를 나타내는 것으로 산포도 (dispersion)라고도 함
- ✓ 변이를 측정해서 실제 변이와 랜덤을 구분하고 실제 변이의 다양한 요인들을 알아보고 변이가 있는 상황에서 결정을 내리는 등 의 작업을 위해서 추정

# 데이터의 특성

## ❖ 변이 추정

### ✓ 지표

- 편차(Deviation): 관측값과 위치 추정값 사이의 차이로 오차 또는 잔차라고도 함
- 분산(Variance): 평균과의 편차를 제곱한 값들의 합을  $n-1$ 로 나눈 값으로 평균 제곱 오차라고도 함
- 표준 편차(Standard Deviation): 분산의 제곱근으로 L2 norm 또는 유클리드 norm
- 평균 절대 편차(Mean Absolute Deviation): 평균과의 편차의 절대값의 평균으로 L1 norm 또는 맨하탄 norm이라고도 함
- 중간 값의 중위 절대 편차(Median Absolute Deviation from the Median): 중간 값과의 편차의 절댓값의 중간 값
- 범위(range): 데이터의 최댓값과 최솟값과의 차이
- 순서 통계량(Order Statistics): 최소에서 최대까지 정렬된 데이터 값에 따른 계량형으로 순서라고도 함
- 백분위 수(Percentile): 어떤 값들의 P퍼센트가 이 값 혹은 더 작은 값을 갖고 (100-P) 퍼센트가 이 값 혹은 더 큰 값을 갖도록 하는 값으로 분위수라고도 함
- 사분위 범위(Interquartile Range): 75번째 백분위 수와 25번째 백분위수 사이의 차이로 IQR이라고도 함

# 데이터의 특성

## ❖ 변이 추정

### ✓ 표준 편차와 관련 추정 값들

- 변위 추정은 관측 데이터와 위치 추정 값 사이의 차이 즉 편차가 기본
- [1, 4, 4]라는 데이터가 있는 경우 평균은 3이고 중값은 4이며 편차는 -2, 1, 1
- 편차는 데이터가 중앙값을 주변으로 얼마나 퍼져있는지를 표현
- 편차 자체의 구해서 사용하는 것은 바람직하지 않은데 그 이유는 음의 편차는 양의 편차를 상쇄하기 때문에 이를 보완하기 가장 간단한 방법은 편차의 절대 값의 평균을 구해서 사용하는 방법이 있는데 이를 평균 절대편차라고 함 – 1.33
- 분산은 제곱 편차의 평균이고 표준 편자는 분산의 제곱근
- 표준 편자는 원래 데이터와 같은 Scale에 있기 때문에 분산보다 해석하기가 쉬우며 수학적으로 제곱한 값이 절댓값보다 통계 모델을 다루는데 더 편리
- 자유도(degrees of freedom)
  - ◆ 분산을 구할 때 데이터 개수  $n$ 이 아닌  $n-1$ 로 나누는데  $n$ 이 충분히 크다면 별 의미가 없지만 그렇지 않은 경우에는 차이가 발생하게 되는데  $n$ 을 분모로 해서 나누면 모집단의 분산과 표준 편자의 참 값을 과소 평가하게 되는데 이를 편향(biased) 추정이라고 하며  $n-1$ 로 나눈 것을 비편향(unbiased) 추정이라고 함
  - ◆ 표준 편자는 표본의 평균에 따른다는 제약조건을 가지므로  $n-1$  개 데이터의 값이 주어지면 다른 하나의 값은 추정이 가능하므로 자유도를  $n-1$ 로 설정

# 데이터의 특성

## ❖ 변이 추정

### ✓ 표준 편차 와 관련 추정 값 들

- 분산, 표준 편차, 평균 절대 편차 등은 특잇값과 극단값에 로버스트 하지 않음
- 분산과 표준 편차는 제곱 편차를 사용하기 때문에 특잇값에 민감한데 이러한 경우에는 중간값으로부터의 중위 절대 편차(MAD)를 사용
  - ◆ 관측값에서 중앙값을 뺀 값들의 중앙값을 구함
  - ◆ 회귀 분석에 쓰는 최소 절대 편차(least absolute deviation)가 유사함

$$MAD = \text{median}(|X_i - \text{median}(X)|)$$

# 데이터의 특성

## ❖ 변이 추정

### ✓ 백분위 수에 기초한 추정

- 범위를 추정하는 다른 접근 방법은 정렬된 데이터가 얼마나 퍼져 있는지 보는 것
- 정렬 데이터를 나타내는 통계량을 순서 통계량이라고 함
- 가장 기본이 되는 측도는 가장 큰 값과 작은 값의 차이를 나타내는 범위인데 최대 최솟값 자체가 특잇값을 분석하는데 큰 도움을 주는데 이 범위는 특잇값에 매우 민감하며 데이터의 범위를 측정하는데 그렇게 유용하지 않음
- 백분위수
  - ◆ 특잇값에 민감한 것을 피하기 위해 범위의 양 끝에서 값들을 지운 후 범위를 다시 확인해 볼 수 있으며 백분위수 사이의 차이를 가지고 추정을 하는 방법이 있음
  - ◆ 데이터에서 P번째 백분위 수는 P퍼센트의 값이 그 값 혹은 그보다 작은 값을 갖고 (100-P) 퍼센트의 값이 그 값 혹은 그보다 큰 값을 갖는 어떤 값을 의미
  - ◆ 데이터가 정렬된 경우 큰 값에서 전체 값의 80%가 되는 곳까지 찾아가서 만나는 값이 0.8 분위수
- 사분위 범위(IQR)
  - ◆ 25번째 백분위 수 와 75번째 백분위 수 차이

# 데이터의 특성

- ❖ 변이 추정

- ✓ 편차

```
print(state.head(8))
#표준 편차
print(state['Population'].std())
#IQR - 사분위 범위
print(state['Population'].quantile(0.75) - state['Population'].quantile(0.25))
#중위 절대 편차(MAD)
from statsmodels import robust
print(robust.scale.mad(state['Population']))
print(abs(state['Population'] - state['Population'].median()).median() /
0.6744897501960817)
```

# 데이터의 특성

## ❖ 변이 추정

### ✓ 편차

	State	Population	Murder.Rate	Abbreviation
0	Alabama	4779736	5.7	AL
1	Alaska	710231	5.6	AK
2	Arizona	6392017	4.7	AZ
3	Arkansas	2915918	5.6	AR
4	California	37253956	4.4	CA
5	Colorado	5029196	2.8	CO
6	Connecticut	3574097	2.4	CT
7	Delaware	897934	5.8	DE
		6848235.347401142		
		4847308.0		
		3849876.1459979336		
		3849876.1459979336		

# 데이터의 특성

## ❖ 데이터의 분포 탐색

- ✓ 분포 탐색을 위한 시각화
  - 상자 그림(Box Plot): 상자 수염도라고도 하는데 데이터의 분포를 시각화 하기 위한 방법
  - 도수 분포표(Frequency Table): 어떤 구간(interval, bin)에 해당하는 수치 데이터 값들의 빈도를 나타내는 기록
  - 히스토그램(Histogram): x축은 구간들을 y 축은 빈도수를 나타내는 도수 테이블의 그림
  - 밀도 그림(Density Plot): 히스토그램을 부드러운 곡선으로 나타낸 그림으로 커널 밀도 추정(Kernel Density Estimation)을 주로 사용
- ✓ 전체 분포를 알아보는 경우에는 주로 사분위수(quantile - 25, 50, 75번째 백분위 수)나 십분위수(decile – 10, 20...90 번째 백분위수)를 사용하며 백분위수는 분포의 꼬리(Tail) 부분을 묘사하는데 제격

# 데이터의 특성

- ❖ 데이터의 분포 탐색

- ✓ 백분위 수 확인

```
#백분위 수 출력
```

```
print(state['Murder.Rate'].quantile([0.05, 0.25, 0.5, 0.75, 0.95]))
```

```
0.05    1.600  
0.25    2.425  
0.50    4.000  
0.75    5.550  
0.95    6.510
```

```
Name: Murder.Rate, dtype: float64
```

# 데이터의 특성

- ❖ 데이터의 분포 탐색
  - ✓ 백분위 수 확인

```
percentages = [0.05, 0.25, 0.5, 0.75, 0.95]
df = pd.DataFrame(state['Murder.Rate'].quantile(percentages))
df.index = [f'{p * 100}%' for p in percentages]
print(df.transpose())
```

	5.0%	25.0%	50.0%	75.0%	95.0%
Murder.Rate	1.6	2.425	4.0	5.55	6.51

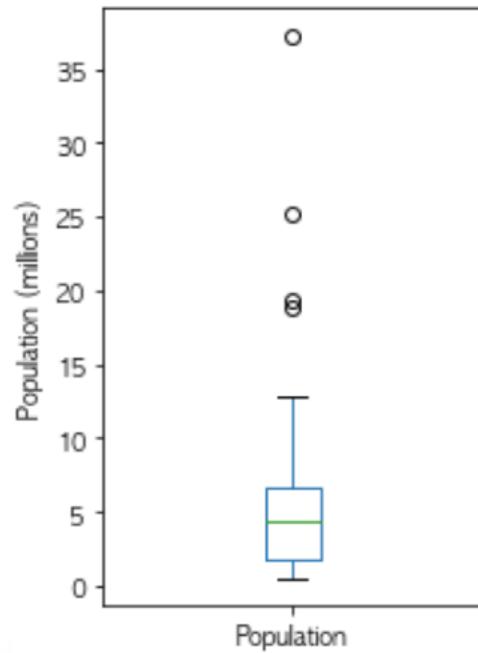
# 데이터의 특성

- ❖ 데이터의 분포 탐색

- ✓ 상자 그림

#인구에 대한 상자 그림

```
ax = (state['Population']/1_000_000).plot.box(figsize=(3, 4))  
ax.set_ylabel('Population (millions)')  
plt.tight_layout()  
plt.show()
```



# 데이터의 특성

## ❖ 데이터의 분포 탐색

### ✓ 분포 탐색을 위한 시각화

- 도수 분포표(Frequency Table)

- ◆ 계급값은 각 계급을 대표하는 값으로 계급의 중앙값이 이용되는데 60~70의 계급이면 계급값은 65점
- ◆ 상대 도수는 전체 데이터에 대해서 해당 계급의 데이터가 어느 정도의 비율을 차지하고 있는지를 나타냄
- ◆ 누적 상대 도수는 해당 계급까지의 상대 도수의 합을 나타내는 것으로 누적 합을 계산할 때 np.cumsum 함수를 사용하면 편리

# 데이터의 특성

## ❖ 데이터의 분포 탐색

### ✓ 도수 분포표

#10개의 구간으로 분할

```
binnedPopulation = pd.cut(state['Population'], 10)  
print(binnedPopulation.value_counts())
```

(526935.67, 4232659.0]	24
(4232659.0, 7901692.0]	14
(7901692.0, 11570725.0]	6
(11570725.0, 15239758.0]	2
(15239758.0, 18908791.0]	1
(18908791.0, 22577824.0]	1
(22577824.0, 26246857.0]	1
(33584923.0, 37253956.0]	1
(26246857.0, 29915890.0]	0
(29915890.0, 33584923.0]	0

Name: Population, dtype: int64

# 데이터의 특성

## ❖ 데이터의 분포 탐색

### ✓ 도수 분포표

```
binnedPopulation.name = 'binnedPopulation'  
df = pd.concat([state, binnedPopulation], axis=1)  
df = df.sort_values(by='Population')  
  
groups = []  
for group, subset in df.groupby(by='binnedPopulation'):   
    groups.append({  
        'BinRange': group,  
        'Count': len(subset),  
        'States': ','.join(subset.Abbreviation)  
    })  
print(pd.DataFrame(groups))
```

# 데이터의 특성

- ❖ 데이터의 분포 탐색

- ✓ 도수 분포표

BinRange	Count
0 (526935.67, 4232659.0]	24
1 (4232659.0, 7901692.0]	14
2 (7901692.0, 11570725.0]	6
3 (11570725.0, 15239758.0]	2
4 (15239758.0, 18908791.0]	1
5 (18908791.0, 22577824.0]	1
6 (22577824.0, 26246857.0]	1
7 (26246857.0, 29915890.0]	0
8 (29915890.0, 33584923.0]	0
9 (33584923.0, 37253956.0]	1

# 데이터의 특성

- ❖ 데이터의 분포 탐색
  - ✓ 도수 분포표

## States

0	WY, VT, ND, AK, SD, DE, MT, RI, NH, ME, HI, ID, NE, WV, NM, N...
1	KY, LA, SC, AL, CO, MN, WI, MD, MO, TN, AZ, IN, MA, WA
2	VA, NJ, NC, GA, MI, OH
3	PA, IL
4	FL
5	NY
6	TX
7	
8	
9	CA

# 데이터의 특성

## ❖ 데이터의 분포 탐색

### ✓ 도수 분포표

#데이터 가져오기

```
scores = pd.read_csv('./data/scores_em.csv',  
                     index_col='student number')
```

# df의 처음 5행을 표시

```
print(scores.head())
```

# 50명의 영어 점수 array

```
english_scores = np.array(scores['english'])
```

```
freq, _ = np.histogram(english_scores, bins=10, range=(0, 100))
```

```
print(freq)
```

# 0~10, 10~20, ... 이라는 문자열의 리스트를 작성

```
freq_class = [f'{i}~{i+10}' for i in range(0, 100, 10)]
```

# freq\_class를 인덱스로 DataFrame을 작성

```
freq_dist_df = pd.DataFrame({'frequency':freq},  
                           index=pd.Index(freq_class,  
                                         name='class'))
```

```
print(freq_dist_df)
```

# 데이터의 특성

## ❖ 데이터의 분포 탐색

### ✓ 도수 분포표

	english	mathematics
student number		
1	42	65
2	69	80
3	56	63
4	41	63
5	57	76
[ 0 0 0 2 8 16 18 6 0 0 ]		
frequency		
class		
0~10	0	
10~20	0	
20~30	0	
30~40	2	
40~50	8	
50~60	16	
60~70	18	
70~80	6	
80~90	0	
90~100	0	

# 데이터의 특성

## ❖ 데이터의 분포 탐색

### ✓ 도수 분포표

```
class_value = [(i+(i+10))/2 for i in range(0, 100, 10)]  
print("계급값:", class_value)
```

```
rel_freq = freq / freq.sum()  
print("상대도수:", rel_freq)
```

```
cum_rel_freq = np.cumsum(rel_freq)  
print("누적 상대 도수:", cum_rel_freq)
```

```
#계급값과 상대 도수 와 누적 상대 도수를 도수분포표에 추가  
freq_dist_df['class value'] = class_value  
freq_dist_df['relative frequency'] = rel_freq  
freq_dist_df['cumulative relative frequency'] = cum_rel_freq  
freq_dist_df = freq_dist_df[['class value', 'frequency',  
                           'relative frequency', 'cumulative relative frequency']]  
  
print(freq_dist_df)  
  
print("빈도가 가장 많은 계급:", freq_dist_df.loc[freq_dist_df['frequency'].idxmax(),  
                                                'class value'])
```

# 데이터의 특성

## ❖ 데이터의 분포 탐색

### ✓ 도수 분포표

계급값: [5, 15, 25, 35, 45, 55, 65, 75, 85, 95]

상대도수: [0. 0. 0. 0.04 0.16 0.32 0.36 0.12 0. 0. ]

누적 상대 도수: [0. 0. 0. 0.04 0.2 0.52 0.88 1. 1. 1. ]

class value frequency relative frequency └─┐

class

0~10	5	0	0.00
10~20	15	0	0.00
20~30	25	0	0.00
30~40	35	2	0.04
40~50	45	8	0.16
50~60	55	16	0.32
60~70	65	18	0.36
70~80	75	6	0.12
80~90	85	0	0.00
90~100	95	0	0.00

# 데이터의 특성

- ❖ 데이터의 분포 탐색
- ✓ 도수 분포표

cumulative relative frequency

class	cumulative relative frequency
0~10	0.00
10~20	0.00
20~30	0.00
30~40	0.04
40~50	0.20
50~60	0.52
60~70	0.88
70~80	1.00
80~90	1.00
90~100	1.00

빈도가 가장 많은 계급: 65

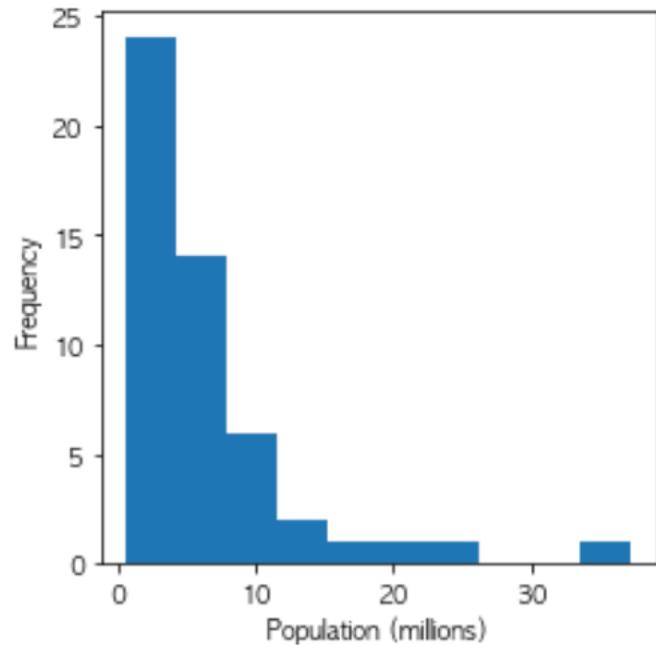
# 데이터의 특성

- ❖ 데이터의 분포 탐색

- ✓ 히스토그램

```
ax = (state['Population'] / 1_000_000).plot.hist(figsize=(4, 4))  
ax.set_xlabel('Population (millions)')
```

```
plt.tight_layout()  
plt.show()
```



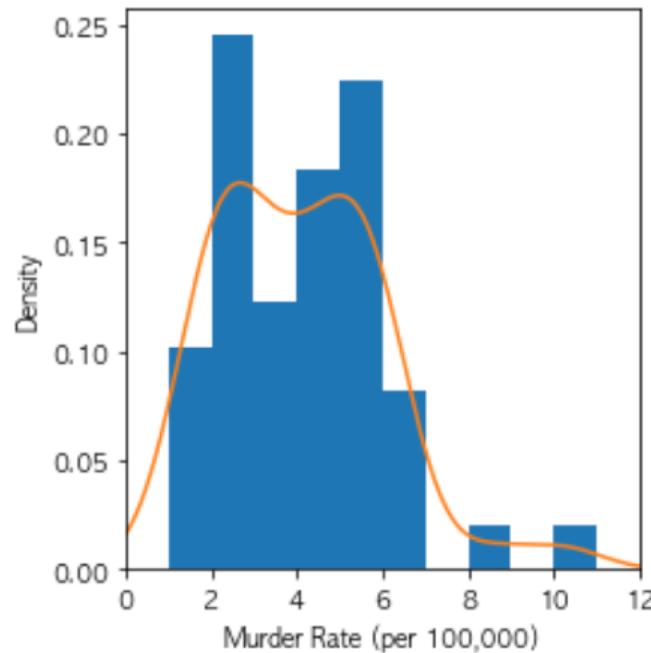
# 데이터의 특성

## ❖ 데이터의 분포 탐색

- ✓ 밀도 추정: 밀도 그림은 데이터의 분포를 연속된 선으로 출력해주는 부드러운 히스토그램

```
ax = state['Murder.Rate'].plot.hist(density=True, xlim=[0, 12],  
                                     bins=range(1,12), figsize=(4, 4))  
state['Murder.Rate'].plot.density(ax=ax)  
ax.set_xlabel('Murder Rate (per 100,000)')
```

```
plt.tight_layout()  
plt.show()
```



# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석
  - ✓ Chipotle 데이터셋의 의미
    - order\_id: 주문번호
    - quantity: 아이템의 주문 수량 – 비율 척도(숫자)
    - item\_name: 주문한 아이템의 이름
    - choice\_description: 주문한 아이템의 상세 선택 옵션
    - item\_price: 주문 아이템의 가격 정보

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 데이터셋의 기본 정보

```
# read_csv 함수로 데이터를 Dataframe 형태로 불러옵니다.  
file_path = './data/chipotle.tsv'  
chipo = pd.read_csv(file_path, sep = '\t')  
print(chipo.shape)  
print("-----")  
chipo.info()
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 데이터셋의 기본 정보

(4622, 5)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   order_id        4622 non-null    int64  
 1   quantity        4622 non-null    int64  
 2   item_name       4622 non-null    object  
 3   choice_description 3376 non-null  object  
 4   item_price      4622 non-null    object  
dtypes: int64(2), object(3)
memory usage: 180.7+ KB
```

결과에서 choice\_description에는 NULL이 포함되어 있음

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ Chipotle 데이터셋의 행과 열, 데이터

# chipo라는 Dataframe에서 순서대로 10개의 row 데이터를 보여줍니다.

```
print(chipo.head(10))
print("-----")
print(chipo.columns)
print("-----")
print(chipo.index)
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석
  - ✓ Chipotle 데이터셋의 행과 열, 데이터

	order_id	quantity	item_name
0	1	1	Chips and Fresh Tomato Salsa
1	1	1	Izze
2	1	1	Nantucket Nectar
3	1	1	Chips and Tomatillo-Green Chili Salsa
4	2	2	Chicken Bowl
5	3	1	Chicken Bowl
6	3	1	Side of Chips
7	4	1	Steak Burrito
8	4	1	Steak Soft Tacos
9	5	1	Steak Burrito

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ Chipotle 데이터셋의 행과 열, 데이터

	choice_description	item_price
0	NaN	\$2.39
1	[Clementine]	\$3.39
2	[Apple]	\$3.39
3	NaN	\$2.39
4	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	\$16.98
5	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	\$10.98
6	NaN	\$1.69
7	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	\$11.75
8	[Tomatillo Green Chili Salsa, [Pinto Beans, Ch...	\$9.25
9	[Fresh Tomato Salsa, [Rice, Black Beans, Pinto...	\$9.25

```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
       dtype='object')
```

```
RangelIndex(start=0, stop=4622, step=1)
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석
  - ✓ Chipotle 데이터셋의 수치적 특징 파악

```
# order_id는 숫자의 의미를 가지지 않기 때문에 str으로 변환  
chipo['order_id'] = chipo['order_id'].astype(str)
```

```
# chipo dataframe에서 수치형 피처들의 요약 통계량을 확인  
print(chipo.describe())
```

```
          quantity
count  4622.000000
mean    1.075725
std     0.410186
min    1.000000
25%   1.000000
50%   1.000000
75%   1.000000
max   15.000000
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ Chipotle 데이터셋의 행과 열의 개수

```
# order_id의 개수를 출력  
print(len(chipotle['order_id'].unique()))  
# item_name의 개수를 출력  
print(len(chipotle['item_name'].unique()))
```

1834

50

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 가장 많이 주문한 아이템 TOP 10

- # 가장 많이 주문한 item : top 10을 출력

```
item_count = chipo['item_name'].value_counts()[:10]
for idx, (val, cnt) in enumerate(item_count.iteritems(), 1):
    print("Top", idx, ":", val, cnt)
```

Top 1 : Chicken Bowl 726

Top 2 : Chicken Burrito 553

Top 3 : Chips and Guacamole 479

Top 4 : Steak Burrito 368

Top 5 : Canned Soft Drink 301

Top 6 : Steak Bowl 211

Top 7 : Chips 211

Top 8 : Bottled Water 162

Top 9 : Chicken Soft Tacos 115

Top 10 : Chips and Fresh Tomato Salsa 110

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 아이템 별 주문 개수와 총량

```
# item당 주문 개수를 출력
```

```
order_count = chipo.groupby('item_name')['order_id'].count()
```

```
order_count[:10] # item당 주문 개수를 출력
```

item\_name

6 Pack Soft Drink	54
Barbacoa Bowl	66
Barbacoa Burrito	91
Barbacoa Crispy Tacos	11
Barbacoa Salad Bowl	10
Barbacoa Soft Tacos	25
Bottled Water	162
Bowl	2
Burrito	6
Canned Soda	104
Name: order_id, dtype: int64	

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 아이템 별 주문 개수와 총량

# item당 주문 총량을 출력

```
item_quantity = chipo.groupby('item_name')['quantity'].sum()
```

```
item_quantity[:10] # item당 주문 총량을 출력
```

item\_name

6 Pack Soft Drink	55
Barbacoa Bowl	66
Barbacoa Burrito	91
Barbacoa Crispy Tacos	12
Barbacoa Salad Bowl	10
Barbacoa Soft Tacos	25
Bottled Water	211
Bowl	4
Burrito	6
Canned Soda	126

Name: quantity, dtype: int64

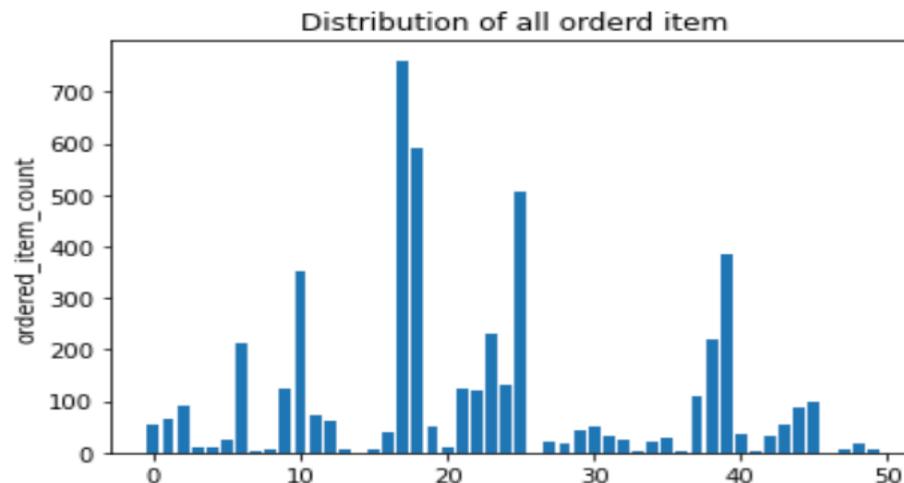
# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석
  - ✓ 아이템 별 주문 개수와 총량 시각화

```
item_name_list = item_quantity.index.tolist()  
x_pos = np.arange(len(item_name_list))  
order_cnt = item_quantity.values.tolist()
```

```
plt.bar(x_pos, order_cnt, align='center')  
plt.ylabel('ordered_item_count')  
plt.title('Distribution of all ordered item')
```

```
plt.show()
```



# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 전처리 – item\_price를 숫자 타입으로 변경

```
print(chipo.info())
print('-----')
chipo['item_price'].head()
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석
  - ✓ 전처리 – item\_price를 숫자 타입으로 변경

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   order_id         4622 non-null    object  
 1   quantity         4622 non-null    int64  
 2   item_name        4622 non-null    object  
 3   choice_description 3376 non-null    object  
 4   item_price       4622 non-null    object  
dtypes: int64(1), object(4)
memory usage: 180.7+ KB
None
-----
0   $2.39
1   $3.39
2   $3.39
3   $2.39
4   $16.98
Name: item_price, dtype: object
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 전처리 – item\_price를 숫자 타입으로 변경

```
# column 단위 데이터에 apply 함수로 전처리를 적용
```

```
chipo['item_price'] = chipo['item_price'].apply(lambda x: float(x[1:]))  
chipo.describe()
```

quantity	item_price	
count	4622.000000	4622.000000
mean	1.075725	7.464336
std	0.410186	4.245557
min	1.000000	1.090000
25%	1.000000	3.390000
50%	1.000000	8.750000
75%	1.000000	9.250000
max	15.000000	44.250000

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 전처리 – item\_price를 숫자 타입으로 변경

```
chipo['item_price'].head()
```

```
0    2.39  
1    3.39  
2    3.39  
3    2.39  
4   16.98
```

```
Name: item_price, dtype: float64
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

- # 주문당 평균 계산 금액을 출력

```
chipo.groupby('order_id')['item_price'].sum().mean()
```

18.811428571428717

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

```
chipo.groupby('order_id')['item_price'].sum().describe()[:10]
```

```
count    1834.000000
mean     18.811429
std      11.652512
min      10.080000
25%     12.572500
50%     16.200000
75%     21.960000
max     205.250000
Name: item_price, dtype: float64
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

# 한 주문에 10달러 이상 사용한 id를 출력

```
chipo_orderid_group = chipo.groupby('order_id').sum()  
results = chipo_orderid_group[chipo_orderid_group.item_price >= 10]
```

```
print(results[:10])
```

```
print(results.index.values)
```

order_id	quantity	item_price
1	4	11.56
10	2	13.20
100	2	10.08
1000	2	20.50
1001	2	10.08
1002	2	10.68
1003	2	13.00
1004	2	21.96
1005	3	12.15
1006	8	71.40
['1' '10' '100' ... '997' '998' '999']		

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

- # 각 아이템의 가격을 계산

```
chipo_one_item = chipo[chipo.quantity == 1]
```

```
price_per_item = chipo_one_item.groupby('item_name').min()
```

```
price_per_item.sort_values(by = "item_price", ascending = False)[:10]
```

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

item_name	order_id	quantity	choice_description	item_price
Steak Salad Bowl	1032	1	[Fresh Tomato Salsa, Lettuce]	9.39
Barbacoa Salad Bowl	1283	1	[Fresh Tomato Salsa, Guacamole]	9.39
Carnitas Salad Bowl	1035	1	[Fresh Tomato Salsa, [Rice, Black Beans, Chees...]	9.39
Carnitas Soft Tacos	1011	1	[Fresh Tomato Salsa (Mild), [Black Beans, Rice...]	8.99
Carnitas Crispy Tacos	1774	1	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...]	8.99
Steak Soft Tacos	1054	1	[Fresh Tomato Salsa (Mild), [Cheese, Sour Cream]]	8.99
Carnitas Salad	1500	1	[[Fresh Tomato Salsa (Mild), Roasted Chili Cor...]	8.99
Carnitas Bowl	1007	1	[Fresh Tomato (Mild), [Guacamole, Lettuce, Ric...]	8.99
Barbacoa Soft Tacos	1103	1	[Fresh Tomato Salsa, [Black Beans, Cheese, Let...]	8.99
Barbacoa Crispy Tacos	110	1	[Fresh Tomato Salsa, Guacamole]	8.99

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

- # 아이템 가격 분포 그래프를 출력

```
item_name_list = price_per_item.index.tolist()
```

```
x_pos = np.arange(len(item_name_list))
```

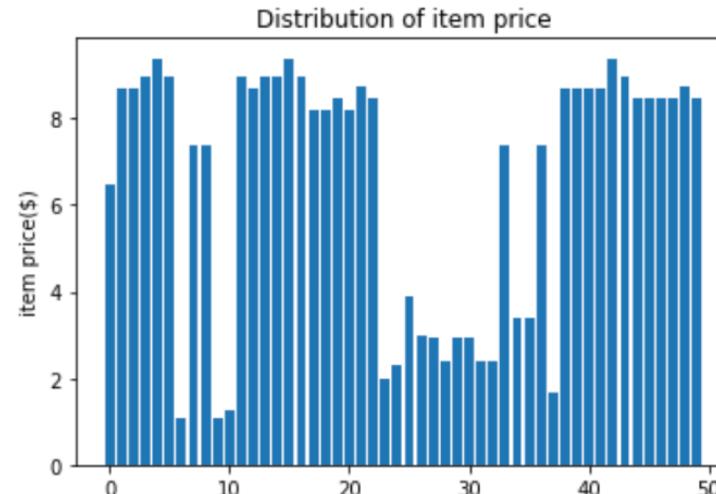
```
item_price = price_per_item['item_price'].tolist()
```

```
plt.bar(x_pos, item_price, align='center')
```

```
plt.ylabel('item price($)')
```

```
plt.title('Distribution of item price')
```

```
plt.show()
```



# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

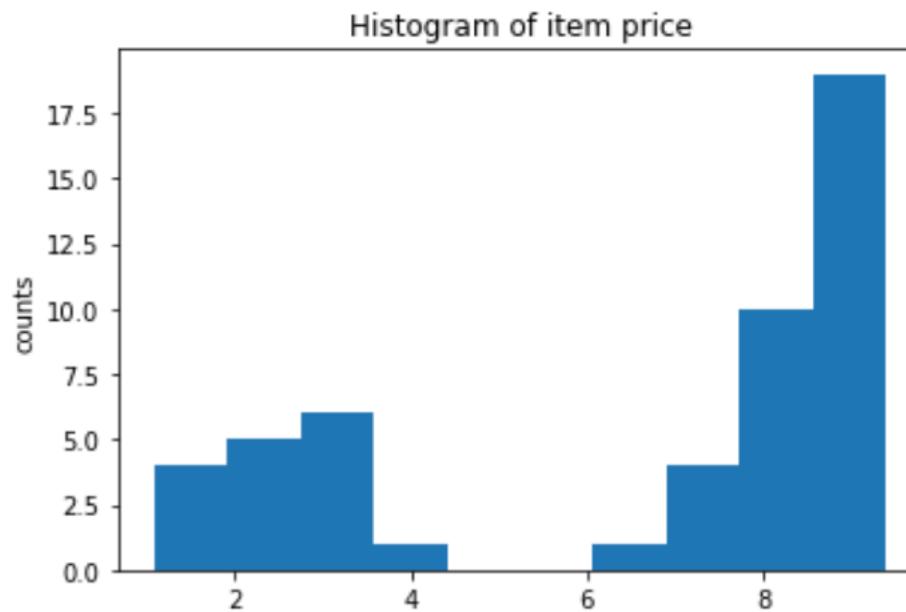
- # 아이템 가격 히스토그램을 출력

```
plt.hist(item_price)
```

```
plt.ylabel('counts')
```

```
plt.title('Histogram of item price')
```

```
plt.show()
```



# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

- # 가장 비싼 주문에서 item이 총 몇개 팔렸는지를 계산

```
chipo.groupby('order_id').sum().sort_values(by='item_price', ascending=False)[:5]
```

order_id	quantity	item_price
926	23	205.25
1443	35	160.74
1483	14	139.00
691	11	118.25
1786	20	114.30

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

# "Veggie Salad Bowl"이 몇 번 주문되었는지를 계산

```
chipo_salad = chipo[chipo['item_name'] == "Veggie Salad Bowl"]
```

```
chipo_salad = chipo_salad.drop_duplicates(['item_name', 'order_id']) # 한 주문 내에서 중복 집계된 item_name을 제거
```

```
print(len(chipo_salad))  
chipo_salad.head(5)
```

18

order_id	quantity	item_name	choice_description	item_price
186	83	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita
Vegetables, Rice,...		11.25		
295	128	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita
Vegetables, Lettu...		11.25		
455	195	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita
Vegetables, Rice,...		11.25		
496	207	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Rice,
Lettuce, Guacamole...		11.25		
960	394	1	Veggie Salad Bowl	[Fresh Tomato Salsa, [Fajita
Vegetables, Lettu...		8.75		

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

# "Chicken Bowl"을 2개 이상 주문한 주문 횟수를 구함

```
chipo_chicken = chipo[chipo['item_name'] == "Chicken Bowl"]  
chipo_chicken_result = chipo_chicken[chipo_chicken['quantity'] >= 2]  
print(chipo_chicken_result.shape[0])
```

33

# 데이터 탐색적 분석

- ❖ 멕시코 풍 프랜차이즈 chipotle 주문 데이터의 탐색적 분석

- ✓ 개념적 탐색 분석

```
# "Chicken Bowl"을 2개 이상 주문한 고객들의 "Chicken Bowl" 메뉴의 총 주문 수량  
을 구함
```

```
chipo_chicken = chipo[chipo['item_name'] == "Chicken Bowl"]  
chipo_chicken_ordersum = chipo_chicken.groupby('order_id').sum()['quantity']  
chipo_chicken_result = chipo_chicken_ordersum[chipo_chicken_ordersum >= 2]
```

```
print(len(chipo_chicken_result))  
chipo_chicken_result.head(5)
```

114

order\_id

1004 2

1023 2

1072 2

1078 2

1091 2

Name: quantity, dtype: int64

# 다면량 탐색

## ❖ 다변량 탐색

- ✓ 평균과 분산과 같은 추정값들은 한 번에 하나의 변수를 다루는데 이를 일변량 분석(univariate analysis)이라고 함
- ✓ 두개 이상의 변수의 관계를 파악하는 것이 다변량 분석(multivariate analysis)인데 이 중 특별히 2개의 변수 간의 관계를 파악하는 것이 이변량 분석(bivariate analysis)
- ✓ 다변량 탐색에서 많이 사용하는 시각화
  - 분할표(Contingency Table): 두 가지 이상의 범주형 변수의 빈도수를 기록한 표
  - 육각형 구간(Hexagonal Binning): 두 변수를 육각형 모양의 구간으로 나눈 그림
  - 등고 도표(Contour Plot): 지도상에 같은 높이의 지점을 등고선으로 나타내는 것처럼 두 변수의 밀도를 등고선으로 표시한 도표
  - 바이올린 도표(Violin Plot): 상자 그림과 비슷하지만 밀도 추정을 함께 보여주는 도표

# 다변량 탐색

## ❖ 교차 분석(Cross Table Analyze)

- ✓ 교차 분석은 범주형 자료를 대상으로 두 개 이상의 변수들에 대한 관련성을 알아보기 위해서 결합 분포를 나타내는 교차 분할표를 작성하고 이를 통해서 변수 상호 간의 관련성 여부를 분석하는 방법
- ✓ 교차 분석에 사용되는 변수는 값이 10가지 미만인 범주형 변수가 적합

# 다변량 탐색

## ❖ 교차 분석(Cross Table Analyze)

### ✓ 데이터 전처리

```
#university = pd.read_csv("data/descriptive.csv", encoding='ms949')
print(university.columns)

university.drop("cost", axis=1, inplace=True)
university['성별'] = '남자'
idx = 0
for val in university['gender']:
    if val == 2:
        university['성별'][idx] = '여자'
    idx = idx + 1
university.drop("gender", axis=1, inplace=True)

print(university.head())
```

# 다변량 탐색

## ❖ 교차 분석(Cross Table Analyze)

### ✓ 데이터 전처리

```
Index(['resident', 'gender', 'age', 'level', 'cost', 'type', 'survey', 'pass'], dtype='object')
```

	resident	age	level	type	survey	pass	성별
0	1.0	50	1.0	1.0	1.0	2.0	남자
1	2.0	54	2.0	1.0	2.0	2.0	남자
2	NaN	62	2.0	1.0	1.0	1.0	남자
3	4.0	50	NaN	1.0	4.0	1.0	여자
4	5.0	51	1.0	1.0	3.0	1.0	남자

# 다변량 탐색

## ❖ 교차 분석(Cross Table Analyze)

### ✓ 데이터 전처리

```
university['학력'] = '응답없음'  
idx = 0  
for val in university['level']:  
    if val == 1.0:  
        university['학력'][idx] = '고졸'  
    if val == 2.0:  
        university['학력'][idx] = '대졸'  
    if val == 3.0:  
        university['학력'][idx] = '대학원졸'  
    idx = idx + 1  
university.drop("level", axis=1, inplace=True)
```

# 다변량 탐색

## ❖ 교차 분석(Cross Table Analyze)

### ✓ 데이터 전처리

```
university['합격여부'] = '응답없음'  
idx = 0  
for val in university['pass']:  
    if val == 1.0:  
        university['합격여부'][idx] = '합격'  
    if val == 2.:  
        university['합격여부'][idx] = '불합격'  
    idx = idx + 1  
university.drop("pass", axis=1, inplace=True)
```

# 다변량 탐색

## ❖ 교차 분석(Cross Table Analyze)

### ✓ 데이터 전처리

```
university['거주지'] = '응답없음'  
idx = 0  
for val in university['resident']:  
    if val == 1.0:  
        university['거주지'][idx] = '특별시'  
    if val == 2.0:  
        university['거주지'][idx] = '광역시'  
    if val == 3.0:  
        university['거주지'][idx] = '시군'  
    idx = idx + 1  
university.drop("resident", axis=1, inplace=True)  
  
print(university.head())
```

# 다변량 탐색

## ❖ 교차 분석(Cross Table Analyze)

### ✓ 데이터 전처리

	age	type	survey	성별	학력	합격여부	거주지
0	50	1.0	1.0	남자	고졸	불합격	특별시
1	54	1.0	2.0	남자	대졸	불합격	광역시
2	62	1.0	1.0	남자	대졸	합격	응답없음
3	50	1.0	4.0	여자	응답없음	합격	응답없음
4	51	1.0	3.0	남자	고졸	합격	응답없음

# 다변량 탐색

## ❖ 교차 분석(Cross Table Analyze)

### ✓ 교차 분할표

```
university = university[(university["학력"]=='고졸') | (university["학력"]=='대졸') |  
(university["학력"]=='대학원졸') ]
```

```
university = university[(university["합격여부"]=='합격') | (university["합격여부"]=='  
불합격')]
```

```
print(pd.crosstab(university['학력'], university['합격여부']))
```

합격여부 불합격 합격

학력

고졸 49 60

대졸 35 58

대학원졸 31 36

# 다변량 탐색

## ❖ 요인 분석

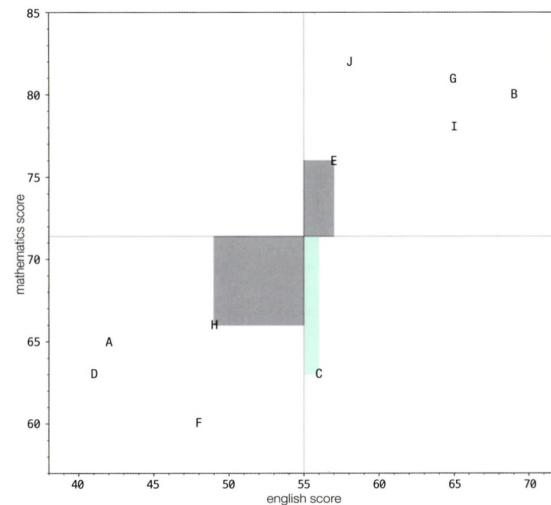
- ✓ 다수의 변수를 대상으로 변수간의 관계를 분석하여 공통 차원으로 축약하는 통계 기법
- ✓ 분석을 할 때 사전에 어떤 변수끼리 묶어야 한다는 전제를 두지 않고 분석하는 탐색적 요인 분석과 사전에 묶일 것으로 기대되는 항목끼리 묶어 지는지를 조사하는 확인적 요인 분석이 있음

# 다면량 탐색

❖ 두 데이터 사이의 관계를 나타내는 지표

✓ 공분산(covariance)

- 2 종류의 데이터를 가지고 결합 분포의 평균을 중심으로 각 자료들이 어떻게 분포되어 있는지를 보여줌
- 공분산이 분산과 다른 점은 가로축과 세로축의 데이터가 다르기 때문에 편차들로 만든 도형이 직사각형이 될 뿐만 아니라 음의 면적도 얻을 수 있다는 것
- 분산은 편차가 음의 값이 되어도 면적은 제곱값이므로 항상 양의 값이 되지만 공분산에서는 가로와 세로의 데이터가 다르므로 한쪽은 편차가 양의 값이고 다른 한쪽은 편차가 음의 값인 경우에 면적이 음의 값이 됨



# 다변량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표
  - ✓ 공분산(covariance)
    - 결합 분포의 평균을 중심으로 각 자료들이 어떻게 분포되어 있는지를 보여줌

샘플 공분산(sample covariance)은 다음과 같이 정의된다. 여기에서  $x_i$ 와  $y_i$ 는 각각  $i$ 번째의  $x$  자료와  $y$ 자료의 값을 가리키고,  $m_x$ 와  $m_y$ 는  $x$  자료와  $y$ 자료의 샘플 평균을 가리킨다.

$$s_{xy}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - m_x)(y_i - m_y)$$

- 키와 체중의 공분산을 구한다고 하면  
(키[0]-키의평균)(몸무게[0]-몸무게평균) + (키[1]-키의평균)(몸무게[1]-몸무게평균).. 의 형태로 모든 데이터의 합계를 구한 후 데이터 개수로 나눈 값

# 다변량 탐색

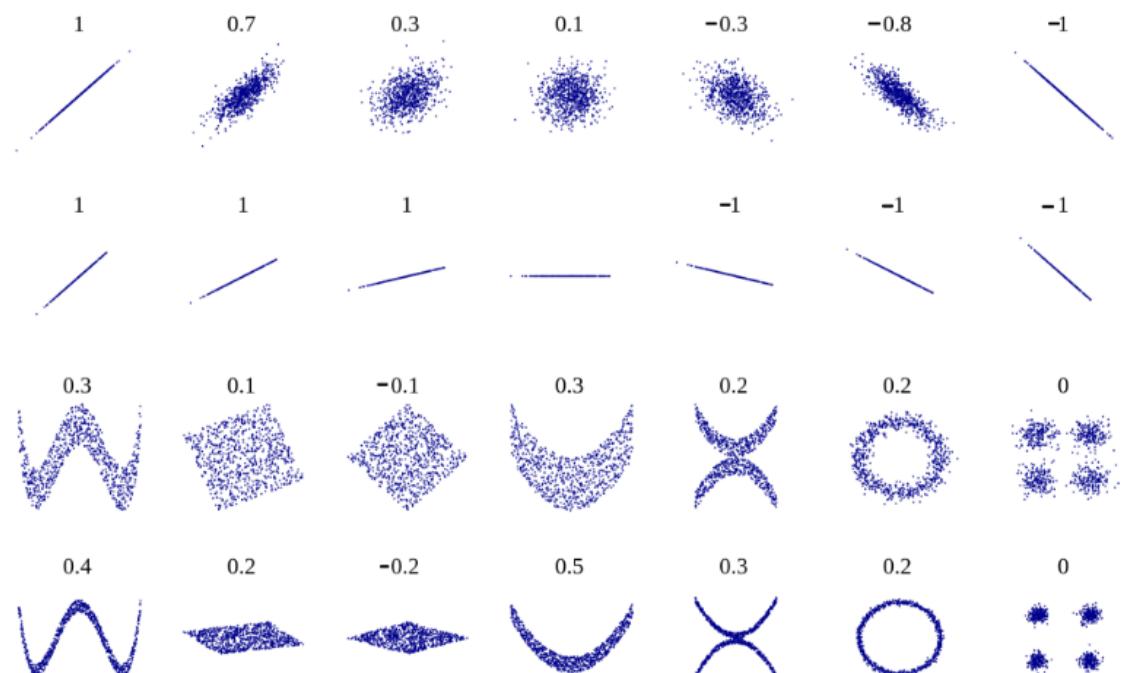
## ❖ 두 데이터 사이의 관계를 나타내는 지표

### ✓ 상관 계수(correlation coefficient)

- 대부분의 경우 자료 분포의 방향성만 분리하여 보는 것이 유용한데 이를 위해서 만든 지표가 상관 계수(correlation coefficient)
- 공분산은 단위나 자료의 범위 등에 따라 값의 차이가 크기 때문에 이를 가지고 연산을 한 번 더 해서 -1.0~ 1.0 사이의 값으로 변환한 것이 상관 계수  
$$\text{상관 계수} = \text{공분산} / \text{키의 표준편차} * \text{몸무게의 표준편차}$$
- 값의 범위는 -1에서 1사이
- 두 변수 사이에는 강한 양의 상관 관계가 존재하면 상관 계수의 값은 1에 가까워지고 반대로 두 변수 사이에 특별한 상관 관계가 존재하지 않으면 상관 계수의 값은 0에 가까울 것이며, 두 변수 사이에 음의 상관 관계가 존재한다면 상관 계수의 값은 -1에 가까워짐
- 보통 절대값 0.9 이상이면 매우 높은 상관 관계이고 0.7 이상이면 높은 상관 관계이고 0.4이상이면 다소 높은 상관 관계이며 0.2이상이면 낮은 상관 관계 그 이외는 상관 관계 없음

# 다변량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표
  - ✓ 상관 계수(correlation coefficient)



# 다변량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표
  - ✓ 상관 계수(correlation coefficient)
    - 상관 분석이나 회귀 분석을 하기 전에 산점도 그리기
      - ◆ matplotlib.pyplot의 scatter 함수를 이용해서 그릴 수 있음
      - ◆ pandas에서 제공하는 DataFrame의 plot 함수를 호출하고 kind 옵션에 scatter를 지정해서 그릴 수 있음
      - ◆ seaborn 패키지의 fairplot(데이터프레임)를 이용해서 데이터프레임의 모든 컬럼들의 산점도를 그릴 수 있음
      - ◆ seaborn 패키지의 regplot(x=컬럼이름, y=컬럼이름, data=데이터프레임, fit\_reg=회귀선표시여부)를 이용해서 특정 컬럼끼리의 산점도를 그릴 수 있음
      - ◆ seaborn 패키지의 jointplot(x=컬럼이름, y=컬럼이름, data=데이터프레임, kind='reg')를 이용하면 두 개의 컬럼의 히스토그램이 x, y 축에 별도로 표시

# 다면량 탐색

❖ 두 데이터 사이의 관계를 나타내는 지표

✓ 상관 계수(correlation coefficient)

- 상관 계수의 종류

- ◆ 피어슨 상관 계수
- ◆ 스피어만 상관 계수
- ◆ 켄달 상관 계수

- 피어슨 상관 계수

- ◆ 0보다 큰 상관 계수 값은 한 변수가 커지면 다른 변수도 큰 값을 갖게 됨을 뜻하고 음의 상관 계수는 한 변수가 커지면 다른 변수가 작은 값을 갖게 됨을 의미
- ◆ 특잇값에 영향을 많이 받음
- ◆ 피어슨 상관 계수는 선형 관계를 판단
- ◆ 비선형 관계(예를 들어  $Y = aX^2 + b$  형태)에 대해서는 제대로 판별하지 못할 수 있음
- ◆ pandas의 DataFrame의 corr 함수를 이용해서 구할 수 있음
- ◆ scipy의 stats 서브패키지는 피어슨 상관 계수를 계산하는 pearsonr() 함수를 제공하며 pearsonr() 함수는 상관 계수와 유의 확률을 반환

# 다변량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

```
# 데이터 읽어오기
```

```
mpg = pd.read_csv('./data/auto-mpg.csv', header=None)
```

```
# 열 이름을 설정
```

```
mpg.columns = ['mpg','cylinders','displacement','horsepower','weight',
               'acceleration','model year','origin','name']
```

```
print(mpg.head(5))
```

# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	₩
0	18.0	8	307.0	130.0	3504.0	12.0	70	
1	15.0	8	350.0	165.0	3693.0	11.5	70	
2	18.0	8	318.0	150.0	3436.0	11.0	70	
3	16.0	8	304.0	150.0	3433.0	12.0	70	
4	17.0	8	302.0	140.0	3449.0	10.5	70	

	origin	name	
0	1	chevrolet	chevelle malibu
1	1	buick	skylark 320
2	1	plymouth	satellite
3	1	amc	rebel sst
4	1	ford	torino

# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	₩
0	18.0	8	307.0	130.0	3504.0	12.0	70	
1	15.0	8	350.0	165.0	3693.0	11.5	70	
2	18.0	8	318.0	150.0	3436.0	11.0	70	
3	16.0	8	304.0	150.0	3433.0	12.0	70	
4	17.0	8	302.0	140.0	3449.0	10.5	70	

	origin	name	
0	1	chevrolet	chevelle malibu
1	1	buick	skylark 320
2	1	plymouth	satellite
3	1	amc	rebel sst
4	1	ford	torino

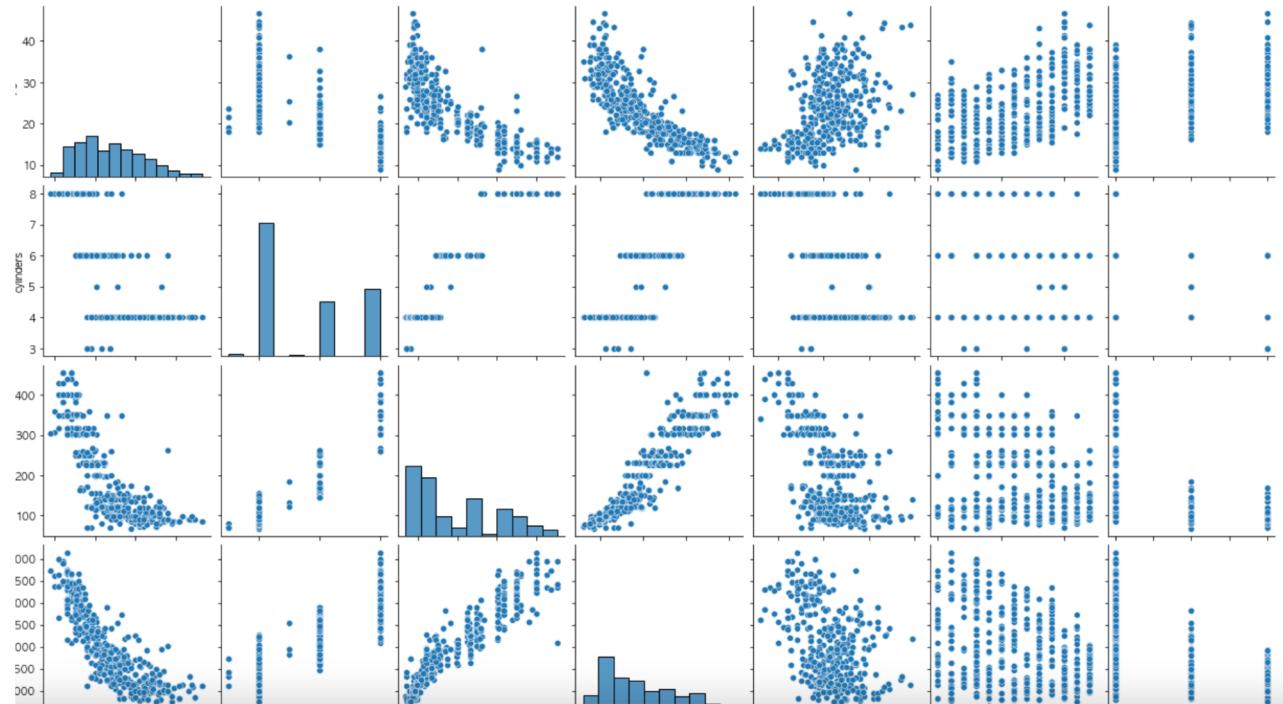
# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

#산점도 그리기

```
import seaborn as sns
```

```
sns.pairplot(mpg)
```



# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

#전체 열들의 상관계수 확인

```
print(mpg.corr())
```

```
mpg cylinders displacement weight acceleration \
mpg      1.000000 -0.775396 -0.804203 -0.831741   0.420289
cylinders -0.775396  1.000000  0.950721  0.896017  -0.505419
displacement -0.804203  0.950721   1.000000  0.932824  -0.543684
weight     -0.831741  0.896017   0.932824  1.000000  -0.417457
acceleration  0.420289  -0.505419  -0.543684 -0.417457   1.000000
model year   0.579267  -0.348746  -0.370164 -0.306564   0.288137
origin      0.563450  -0.562543  -0.609409 -0.581024   0.205873
```

```
model year origin
mpg      0.579267  0.563450
cylinders -0.348746 -0.562543
displacement -0.370164 -0.609409
weight     -0.306564 -0.581024
acceleration  0.288137  0.205873
model year   1.000000  0.180662
origin      0.180662  1.000000
```

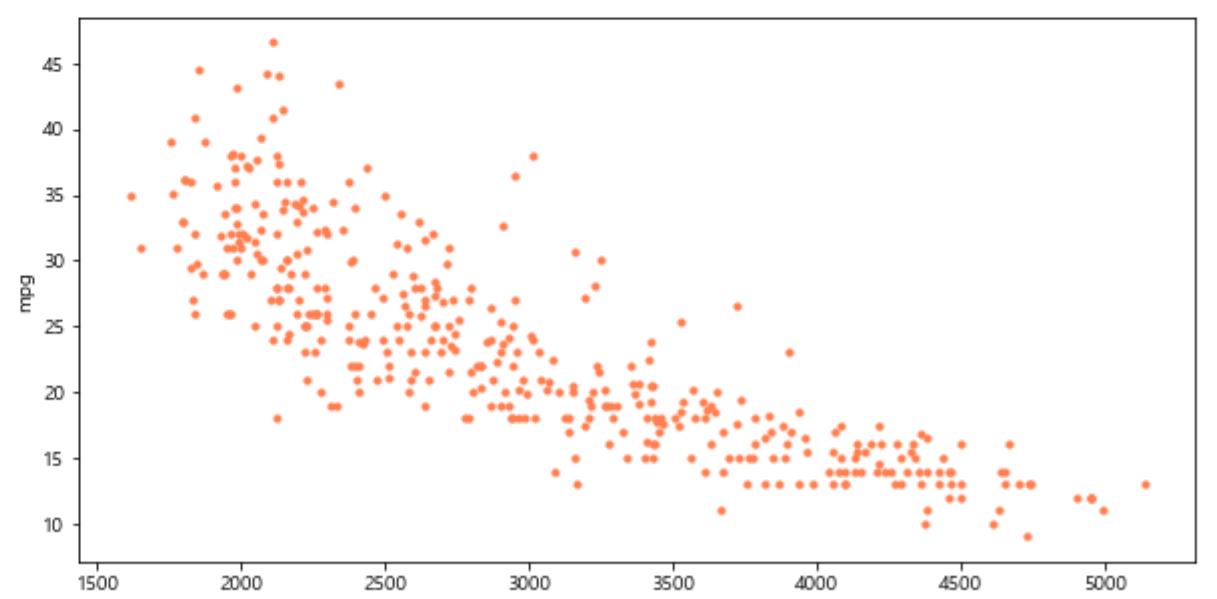
# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

#mpg 와 weight 그리고 horsepower 와의 상관계수

#weight 와 mpg 컬럼의 산점도 그리기

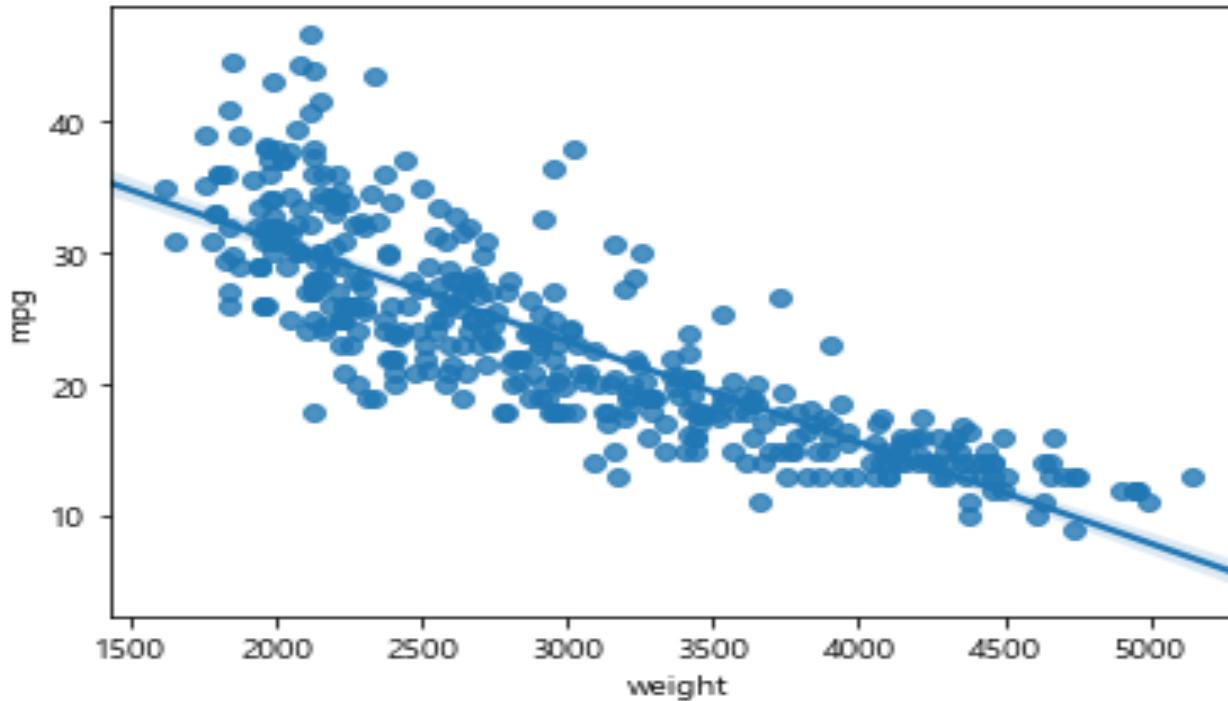
```
mpg.plot(kind='scatter', x='weight',y='mpg', c='coral', s=10, figsize=(10,5))  
plt.show()
```



# 다변량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

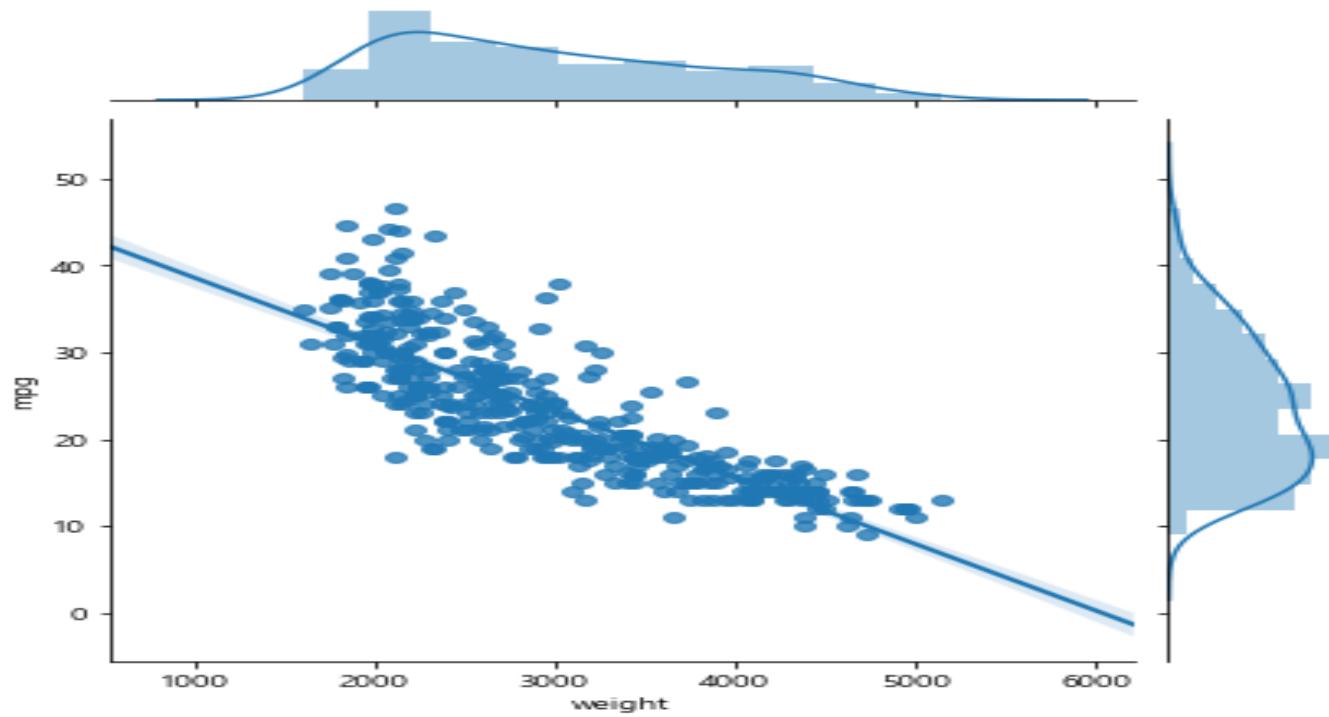
```
#seaborn 패키지의 regplot()을 이용한 산점도 그리기  
#fit_reg=False 옵션을 이용해서 회귀선을 제거할 수 있음  
sns.regplot(x='weight', y='mpg', data=mpg)
```



# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

```
sns.jointplot(x='weight', y='mpg', kind='reg', data=mpg)
```



# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

#mpg 와 weight 그리고 horsepower 와의 상관계수

```
print('mpg와 weight의 상관계수:', mpg['mpg'].corr(mpg['weight']))
```

```
#print(mpg['mpg'].corr(mpg['horsepower'])) #horsepower는 숫자 컬럼이 아니라서 안됨  
mpg['horsepower'].replace('?', np.nan, inplace=True) # '?'을 np.nan으로 변경  
mpg.dropna(subset=['horsepower'], axis=0, inplace=True) # 누락데이터 행을 삭제  
mpg['horsepower'] = mpg['horsepower'].astype('float') # 문자열을 실수형으로 변환
```

```
print('mpg와 horsepower의 상관계수:', mpg['mpg'].corr(mpg['horsepower']))
```

mpg와 weight의 상관계수: -0.8322442148315754

mpg와 horsepower의 상관계수: -0.7784267838977761

# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

```
import scipy as sp  
result = sp.stats.pearsonr(mpg['mpg'].values, mpg['horsepower'].values)  
print(result)
```

(-0.7784267838977761, 7.031989029404151e-81)

# 다변량 탐색

- ❖ 주식 데이터를 이용한 상관 관계 확인

#데이터 가져오기

```
sp500_sym = pd.read_csv("./data/sp500_sectors.csv")
sp500_px = pd.read_csv("./data/sp500_data.csv.gz", index_col=0)
```

#통신사 데이터만 가져오기

```
telecomSymbols = sp500_sym[sp500_sym['sector'] ==
    'telecommunications_services']['symbol']
```

#특정 기간의 데이터만 추출

```
telecom = sp500_px.loc[sp500_px.index >= '2012-07-01', telecomSymbols]
telecom.corr()
print(telecom)
```

# 다면량 탐색

- ❖ 주식 데이터를 이용한 상관 관계 확인

	T	CTL	FTR	VZ	LVLT
2012-07-02	0.422496	0.140847	0.070879	0.554180	-0.519998
2012-07-03	-0.177448	0.066280	0.070879	-0.025976	-0.049999
2012-07-05	-0.160548	-0.132563	0.055128	-0.051956	-0.180000
2012-07-06	0.342205	0.132563	0.007875	0.140106	-0.359999
2012-07-09	0.136883	0.124279	-0.023626	0.253943	0.180000
...	...	...	...	...	...
2015-06-25	0.049342	-1.600000	-0.040000	-0.187790	-0.330002
2015-06-26	-0.256586	0.039999	-0.070000	0.029650	-0.739998
2015-06-29	-0.098685	-0.559999	-0.060000	-0.504063	-1.360000
2015-06-30	-0.503298	-0.420000	-0.070000	-0.523829	0.199997
2015-07-01	-0.019737	0.080000	-0.050000	0.355811	0.139999

[754 rows x 5 columns]

# 다면량 탐색

- ❖ 주식 데이터를 이용한 상관 관계 확인

```
#(sector == 'etf').  
etfs = sp500_px.loc[sp500_px.index > '2012-07-01',  
                    sp500_sym[sp500_sym['sector'] == 'etf']['symbol']]  
print(etfs.head())
```

	XLI	QQQ	SPY	DIA	GLD	...	XTL	₩								
2012-07-02	-0.376098	0.096313	0.028223	-0.242796	0.419998	...	0.019076									
2012-07-03	0.376099	0.481576	0.874936	0.728405	0.490006	...	0.000000									
2012-07-05	0.150440	0.096313	-0.103487	0.149420	0.239991	...	0.019072									
2012-07-06	-0.141040	-0.491201	0.018819	-0.205449	-0.519989	...	-0.429213									
2012-07-09	0.244465	-0.048160	-0.056445	-0.168094	0.429992	...	0.000000									

	XLV	XLP	XLF	XLK												
2012-07-02	-0.009529	0.313499	0.018999	0.075668												
2012-07-03	0.000000	0.129087	0.104492	0.236462												
2012-07-05	-0.142955	-0.073766	-0.142490	0.066211												
2012-07-06	-0.095304	0.119865	0.066495	-0.227003												
2012-07-09	0.352630	-0.064548	0.018999	0.009457												

[5 rows x 17 columns]

# 다면량 탐색

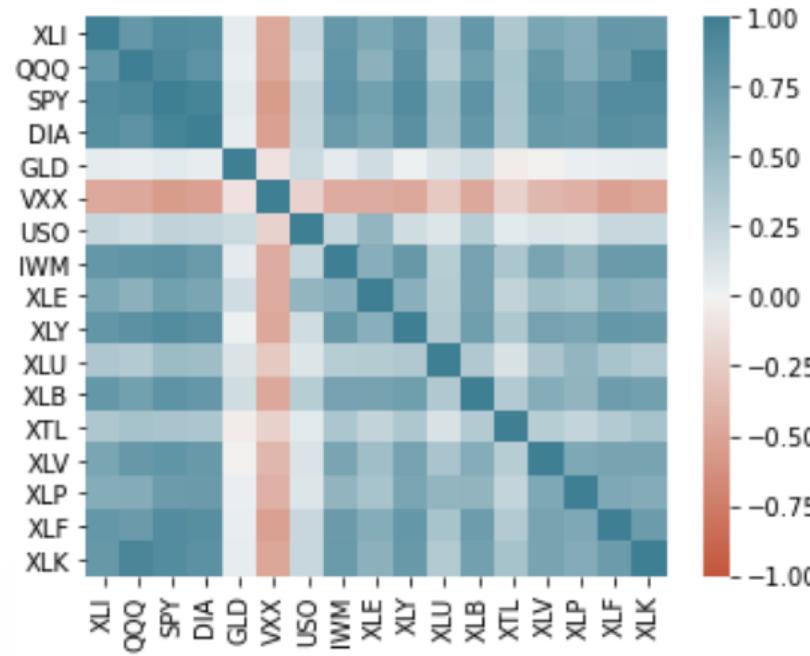
- ❖ 주식 데이터를 이용한 상관 관계 확인

#heatmap을 이용한 수익 간의 상관관계 파악

```
fig, ax = plt.subplots(figsize=(5, 4))
ax = sns.heatmap(etfs.corr(), vmin=-1, vmax=1,
                 cmap=sns.diverging_palette(20, 220, as_cmap=True),
                 ax=ax)
```

```
plt.tight_layout()
```

```
plt.show()
```



# 다면량 탐색

- ❖ 주식 데이터를 이용한 상관 관계 확인

```
from matplotlib.collections import EllipseCollection
from matplotlib.colors import Normalize

def plot_corr_ellipses(data, figsize=None, **kwargs):
    """ https://stackoverflow.com/a/34558488 """
    M = np.array(data)
    if not M.ndim == 2:
        raise ValueError('data must be a 2D array')
    fig, ax = plt.subplots(1, 1, figsize=figsize, subplot_kw={'aspect':'equal'})
    ax.set_xlim(-0.5, M.shape[1] - 0.5)
    ax.set_ylim(-0.5, M.shape[0] - 0.5)
    ax.invert_yaxis()

    # xy locations of each ellipse center
    xy = np.indices(M.shape)[::-1].reshape(2, -1).T

    # set the relative sizes of the major/minor axes according to the strength of
    # the positive/negative correlation
    w = np.ones_like(M).ravel() + 0.01
    h = 1 - np.abs(M).ravel() - 0.01
    a = 45 * np.sign(M).ravel()
```

# 다면량 탐색

- ❖ 주식 데이터를 이용한 상관 관계 확인

```
ec = EllipseCollection(widths=w, heights=h, angles=a, units='x', offsets=xy,
                      norm=Normalize(vmin=-1, vmax=1),
                      transOffset=ax.transData, array=M.ravel(), **kwargs)
ax.add_collection(ec)

# if data is a DataFrame, use the row/column names as tick labels
if isinstance(data, pd.DataFrame):
    ax.set_xticks(np.arange(M.shape[1]))
    ax.set_xticklabels(data.columns, rotation=90)
    ax.set_yticks(np.arange(M.shape[0]))
    ax.set_yticklabels(data.index)

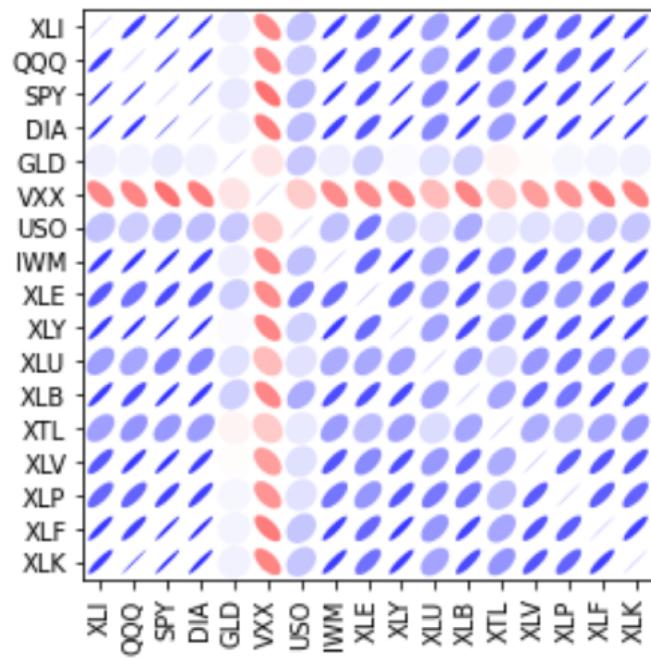
return ec

m = plot_corr_ellipses(etfs.corr(), figsize=(5, 4), cmap='bwr_r')
cb = fig.colorbar(m)
cb.set_label('Correlation coefficient')

plt.tight_layout()
plt.show()
```

# 다면량 탐색

- ❖ 주식 데이터를 이용한 상관 관계 확인



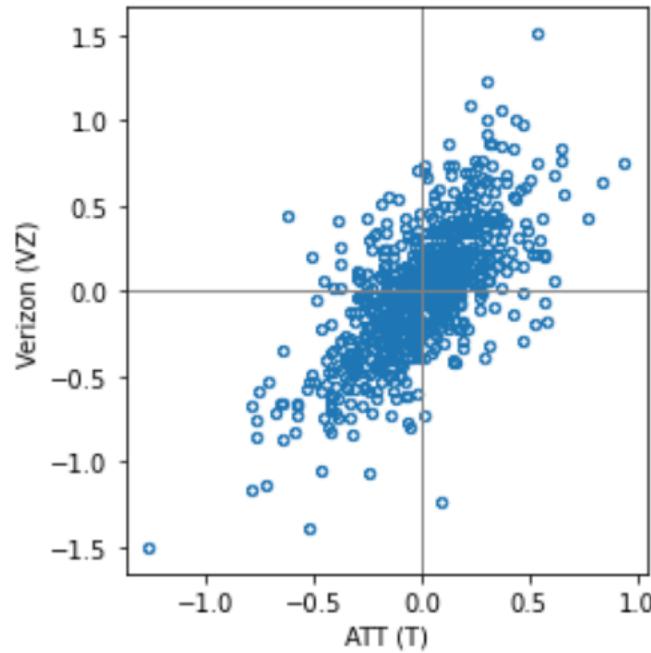
# 다면량 탐색

- ❖ 주식 데이터를 이용한 상관 관계 확인

#ATT와 Verizon 과의 상관관계

```
ax = telecom.plot.scatter(x='T', y='VZ', figsize=(4, 4), marker='u25EF$')
ax.set_xlabel('ATT (T)')
ax.set_ylabel('Verizon (VZ)')
ax.axhline(0, color='grey', lw=1)
ax.axvline(0, color='grey', lw=1)
```

```
plt.tight_layout()
plt.show()
```



# 다변량 탐색

## ❖ 상관 관계 확인

### ✓ 앤스콤 데이터

- 상관 계수로 분포의 형상을 추측할 때 개별 자료가 상관 계수에 미치는 영향력이 각각 다르다는 점에 유의해야 함
- 프랭크 앤스콤(Frank Anscombe)의 논문에 예시된 데이터는 서로 다른 4종류의 2차원 데이터 셋을 포함하는데 4종류 데이터셋 모두 상관 계수가 약 0.816로 동일
- 첫번째 데이터셋은 평범한 데이터셋이지만 두번째 데이터셋은 비선형으로 완벽한 상관 관계를 가지기 때문에  $x$ 값을 알면  $y$ 값을 완벽하게 알 수 있지만 상관 계수는 약 0.816으로 1인 아닌 값을 가지게 되는데 상관 계수는 비선형 상관 관계를 표현하지 못함
- 세번째 데이터셋과 네번째 데이터셋에서 볼 수 있듯이 나머지 데이터의 상관 계수가 1 또는 0인 경우에도 단 하나의 특이값(outlier) 자료에 의해 상관계수가 크게 달라짐

# 다면량 탐색

## ❖ 상관 관계 확인

### ✓ 앤스콤 데이터

```
import statsmodels.api as sm  
data = sm.datasets.get_rdataset("anscombe")  
df = data.data  
print(df[["x1", "y1", "x2", "y2", "x3", "y3", "x4", "y4"]])
```

	x1	y1	x2	y2	x3	y3	x4	y4
0	10	8.04	10	9.14	10	7.46	8	6.58
1	8	6.95	8	8.14	8	6.77	8	5.76
2	13	7.58	13	8.74	13	12.74	8	7.71
3	9	8.81	9	8.77	9	7.11	8	8.84
4	11	8.33	11	9.26	11	7.81	8	8.47
5	14	9.96	14	8.10	14	8.84	8	7.04
6	6	7.24	6	6.13	6	6.08	8	5.25
7	4	4.26	4	3.10	4	5.39	19	12.50
8	12	10.84	12	9.13	12	8.15	8	5.56
9	7	4.82	7	7.26	7	6.42	8	7.91
10	5	5.68	5	4.74	5	5.73	8	6.89

# 다면량 탐색

## ❖ 상관 관계 확인

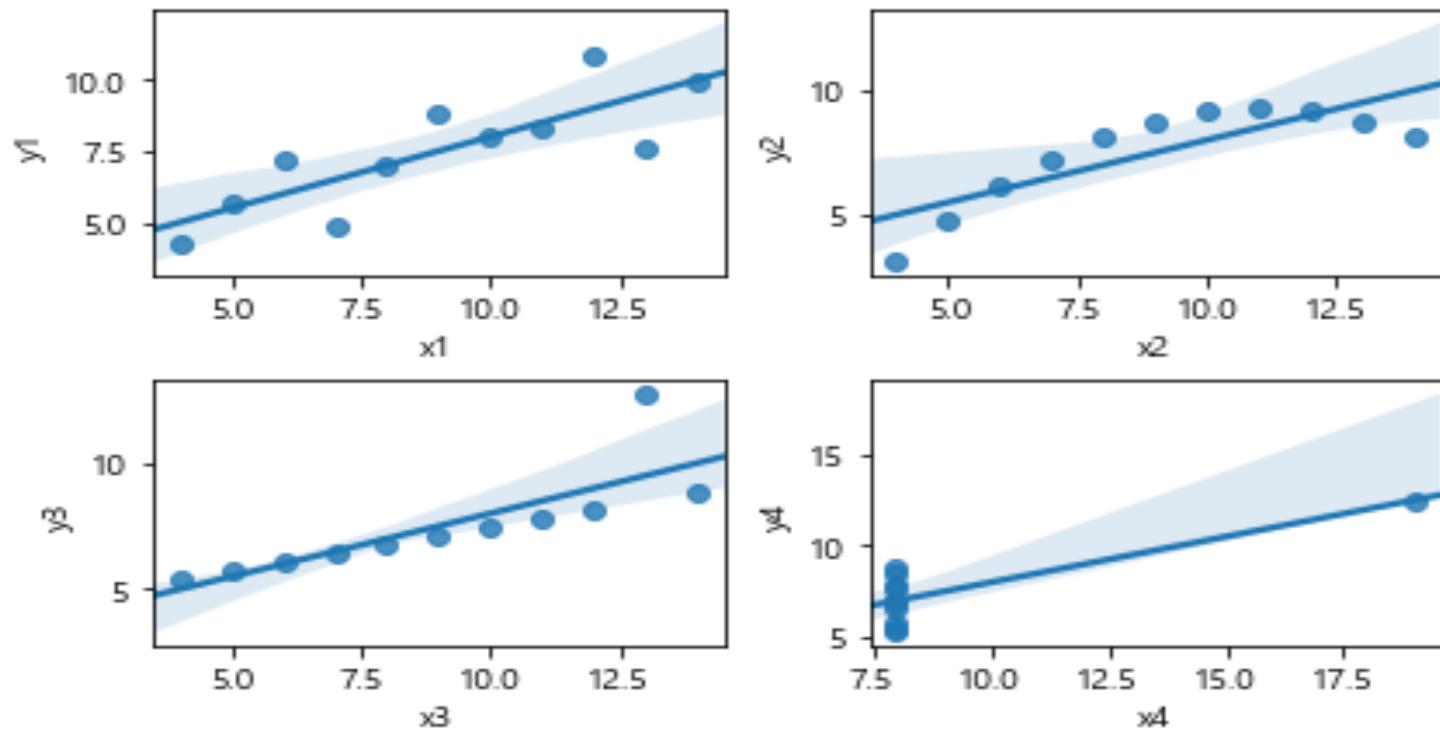
### ✓ 앤스콤 데이터

```
plt.subplot(221)
sns.regplot(x="x1", y="y1", data=df)
plt.subplot(222)
sns.regplot(x="x2", y="y2", data=df)
plt.subplot(223)
sns.regplot(x="x3", y="y3", data=df)
plt.subplot(224)
sns.regplot(x="x4", y="y4", data=df)
plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.suptitle("앤스콤의 데이터")
plt.show()
```

# 다면량 탐색

- ❖ 상관 관계 확인
  - ✓ 앤스콤 데이터

앤스콤의 데이터



# 다변량 탐색

❖ 두 데이터 사이의 관계를 나타내는 지표

✓ 상관 계수(correlation coefficient)

- 스피어만 상관 계수(Spearman's Rank Correlation Coefficient)

- ◆ 스피어만 상관계수는 상관 계수를 계산할 두 데이터의 실제 값 대신 두 값의 순위를 사용해 상관 계수를 구하는 방식
- ◆ 계산 방법이 피어슨 상관 계수와 유사해 이해가 쉽고 피어슨 상관 계수와 달리 비선형 관계의 연관성을 파악할 수 있다는 장점이 있음
- ◆ 순위만 매길 수 있다면 적용이 가능하므로 연속형(Continuous) 데이터에 적합한 피어슨 상관 계수와는 달리 이산형(Discrete) 데이터, 순서형(Ordinal) 데이터에 적용이 가능
- ◆ 국어 점수와 영어 점수간의 상관 계수는 피어슨 상관 계수로 계산할 수 있고 국어 성적 석차와 영어 성적 석차의 상관 계수는 스피어만 상관 계수로 계산 가능
- ◆ pandas의 corr 함수에 method 라는 옵션에 spearman 을 설정
- ◆ scipy.stats.spearmanr(a, b=None, axis=0) 라는 함수를 이용해서도 가능

# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표
  - ✓ 상관 계수(correlation coefficient)
    - 스피어만 상관 계수(Spearman's Rank Correlation Coefficient)

#스피어만 상관계수

```
s1 = pd.Series([1,2,3,4,5])
```

```
s2 = pd.Series([1,4,9,16,25])
```

```
print("피어슨 상관계수:", s1.corr(s2))
```

```
print("스피어만 상관계수:", s1.corr(s2, method='spearman'))
```

```
print("스피어만 상관계수:", sp.stats.spearmanr(s1, s2))
```

피어슨 상관계수: 0.981104910251593

스피어만 상관계수: 0.9999999999999999

스피어만 상관계수: SpearmanResult(correlation=0.9999999999999999,  
pvalue=1.4042654220543672e-24)

# 다면량 탐색

- ❖ 두 데이터 사이의 관계를 나타내는 지표

- ✓ 상관 계수(correlation coefficient)

- 켄달 상관 계수(Kendal's Rank Correlation Coefficient)

- ◆ 켄달 상관 계수는(X,Y)형태의 순서 쌍으로 데이터가 있을 때  $x_i < x_j$ ,  $y_i < y_j$ 가 성립하면 concordant,  $x_i < x_j$  이지만  $y_i > y_j$  이면 discordant라고 정의
      - ◆ x 가 클 때 y 도 크면 concordant, x가 크지만 y는 작다면 discordant로 보는 것
      - ◆ `scipy.stats.kendalltau` 함수를 이용해서 구할 수 있음

```
## 켄달 상관계수  
#스피어만 상관계수  
s1 = pd.Series([1,2,3,4,5])  
s2 = pd.Series([1,4,9,16,25])  
print("켄달 상관계수:", s1.corr(s2, method='kendall'))  
print("켄달 상관계수:", sp.stats.kendalltau(s1, s2))
```

켄달 상관계수: 0.9999999999999999

켄달 상관계수: KendalltauResult(correlation=0.9999999999999999,  
pvalue=0.01666666666666666)

# 다변량 탐색

## ❖ 육각형 그래프 와 등고선

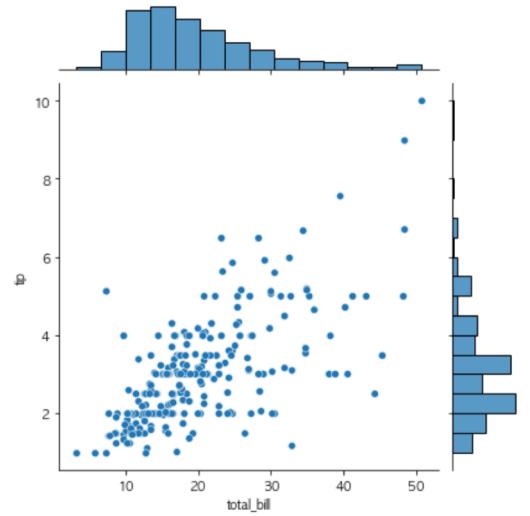
- ✓ 산점도는 데이터의 개수가 상대적으로 적을 때는 무난하지만 수십, 수백만의 레코드를 나타내기에는 산점도의 점들이 너무 밀집되어 알아보기 힘듬
- ✓ 육각형 구간 차트
  - 육각형 구간 차트는 점으로 표시하는 대신 기록값들을 육각형 모양의 구간들로 나누고 각 구간에 포함된 기록 값의 개수에 따라 색깔을 표시
  - 색깔의 모양이 구름의 형태로 표현됨
- ✓ 등고선 차트
  - 두 변수로 이루어진 지형에서의 등고선

# 다면량 탐색

- ❖ 육각형 그래프 와 등고선

- ✓ 산점도

```
tips = sns.load_dataset("tips")    # 팁 데이터  
sns.jointplot(x="total_bill", y="tip", data=tips, kind='scatter')
```



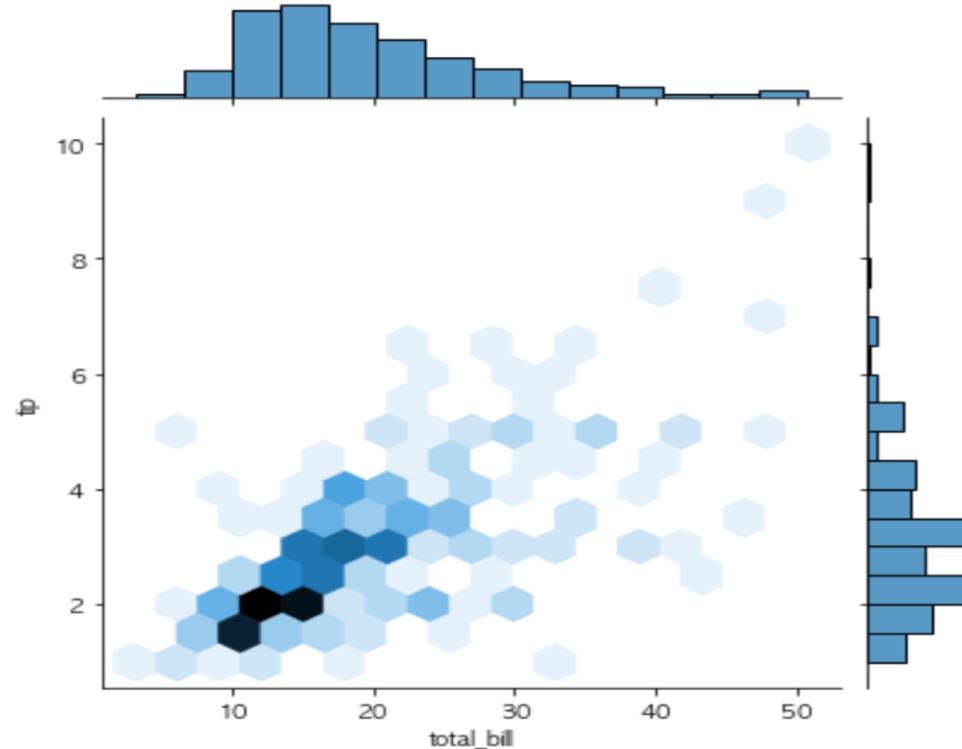
# 다면량 탐색

- ❖ 육각형 그래프 와 등고선

- ✓ 육각형 그래프

- #육각형 그래프

```
sns.jointplot(x="total_bill", y="tip", data=tips, kind='hex')
```



# 다면량 탐색

- ❖ 육각형 그래프 와 등고선
  - ✓ 육각형 그래프 – 집의 크기와 과세 평가액을 육각형 그래프로 그리기

```
kc_tax = pd.read_csv("./data/kc_tax.csv.gz")
kc_tax0 = kc_tax.loc[(kc_tax.TaxAssessedValue < 750000) &
                     (kc_tax.SqFtTotLiving > 100) &
                     (kc_tax.SqFtTotLiving < 3500), :]
print(kc_tax0.shape)
```

(432693, 3)

# 다면량 탐색

- ❖ 육각형 그래프 와 등고선

- ✓ 육각형 그래프

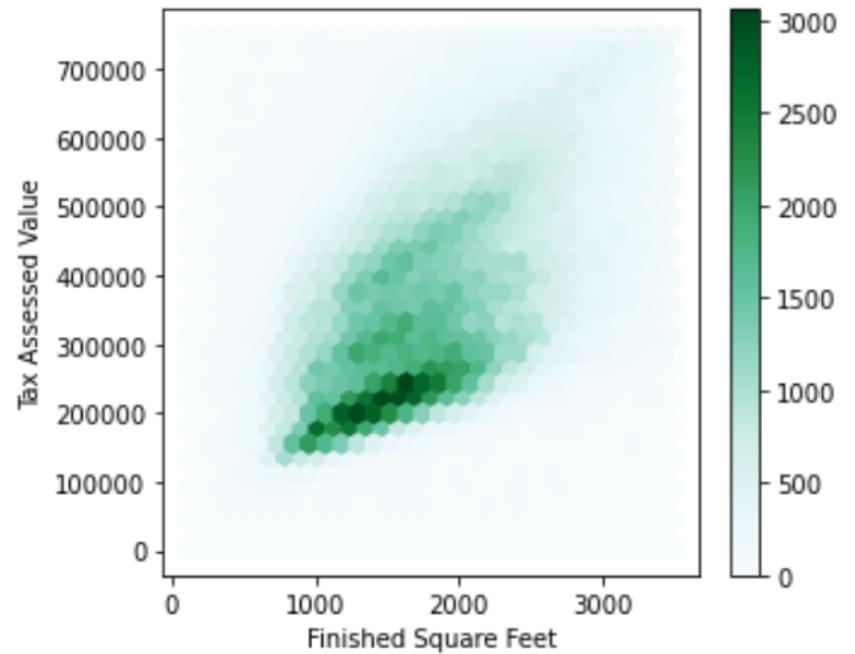
```
ax = kc_tax0.plot.hexbin(x='SqFtTotLiving', y='TaxAssessedValue',  
                        gridsize=30, sharex=False, figsize=(5, 4))
```

```
ax.set_xlabel('Finished Square Feet')
```

```
ax.set_ylabel('Tax Assessed Value')
```

```
plt.tight_layout()
```

```
plt.show()
```



# 다면량 탐색

- ❖ 육각형 그래프 와 등고선

- ✓ 등고선 그래프

```
x=np.arange(-1,1,0.1)
```

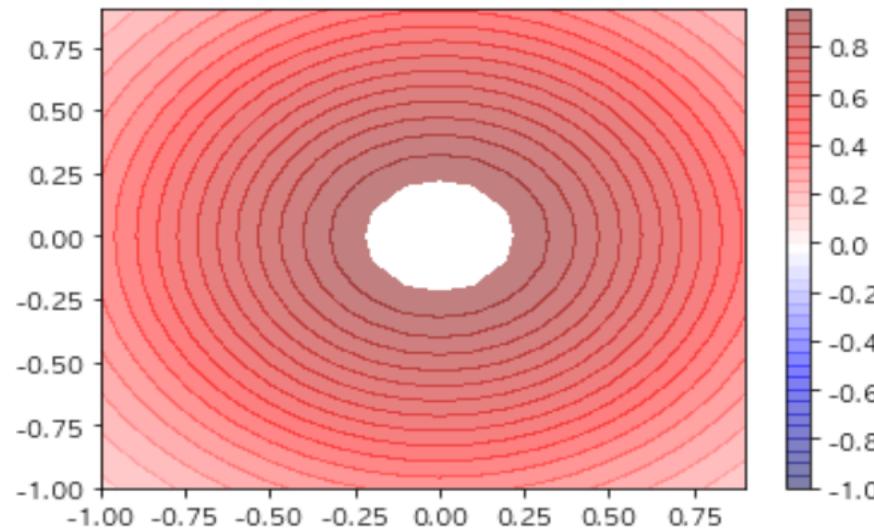
```
y=np.arange(-1,1,0.1)
```

```
X,Y=np.meshgrid(x,y)
```

```
Z=np.exp(-(X**2+Y**2))
```

```
CS=plt.contourf(X,Y,Z,levels=np.arange(-1,1,0.05),alpha=0.5,cmap='seismic')
```

```
plt.colorbar()
```



# 다면량 탐색

- ❖ 육각형 그래프 와 등고선

- ✓ 등고선 그래프

```
# Vector Field
```

```
Y, X = np.mgrid[-2:2:20j, -2:2:20j]
```

```
U =(1 - 2*(X**2))*np.exp(-((X**2)+(Y**2)))
```

```
# Countour Plot
```

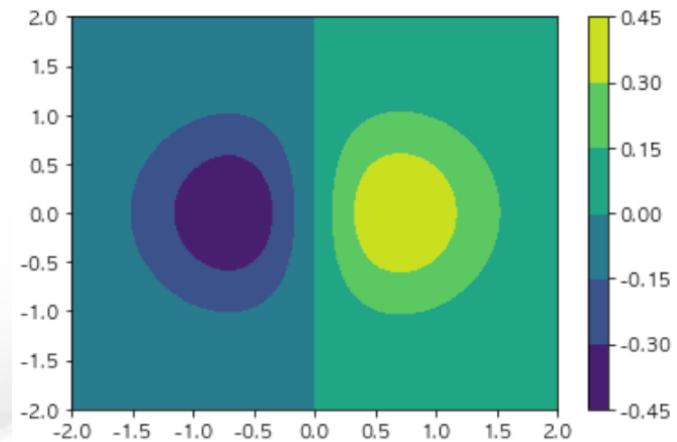
```
X, Y = np.mgrid[-2:2:100j, -2:2:100j]
```

```
Z = X*np.exp(-(X**2 + Y**2))
```

```
cp = plt.contourf(X, Y, Z)
```

```
plt.colorbar(cp)
```

```
plt.show()
```



# 다면량 탐색

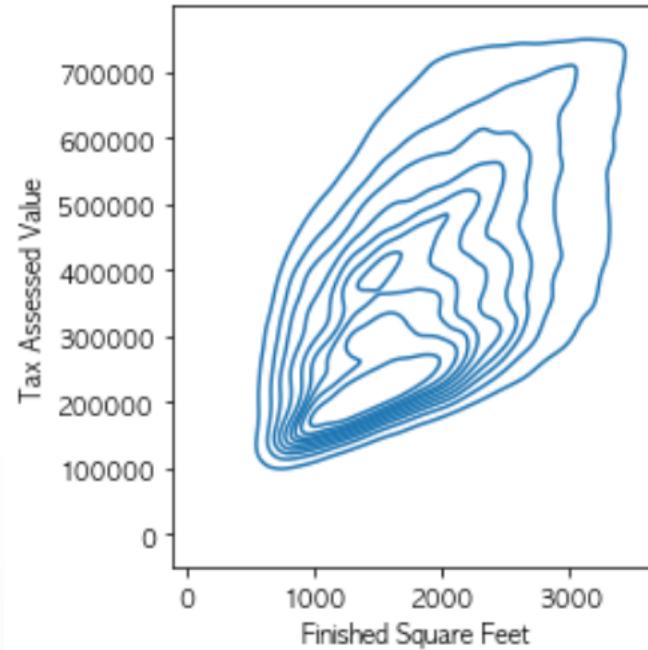
- ❖ 육각형 그래프 와 등고선

- ✓ 등고선 그래프

#등고선 차트 - 시간이 오래 걸림

```
fig, ax = plt.subplots(figsize=(4, 4))
sns.kdeplot(data=kc_tax0, x='SqFtTotLiving', y='TaxAssessedValue', ax=ax)
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax Assessed Value')
```

```
plt.tight_layout()
plt.show()
```



# 다면량 탐색

- ❖ 범주형 과 범주형 데이터

- ✓ 분할표 이용

```
lc_loans = pd.read_csv('./data/lc_loans.csv')
crosstab = lc_loans.pivot_table(index='grade', columns='status',
                                 aggfunc=lambda x: len(x), margins=True)
print(crosstab)
```

grade	status	Charged Off	Current	Fully Paid	Late	All
A	1562	50051	20408	469	72490	
B	5302	93852	31160	2056	132370	
C	6023	88928	23147	2777	120875	
D	5007	53281	13681	2308	74277	
E	2842	24639	5949	1374	34804	
F	1526	8444	2328	606	12904	
G	409	1990	643	199	3241	
All	22671	321185	97316	9789	450961	

# 다면량 탐색

## ❖ 범주형 과 범주형 데이터

### ✓ 분할표 이용

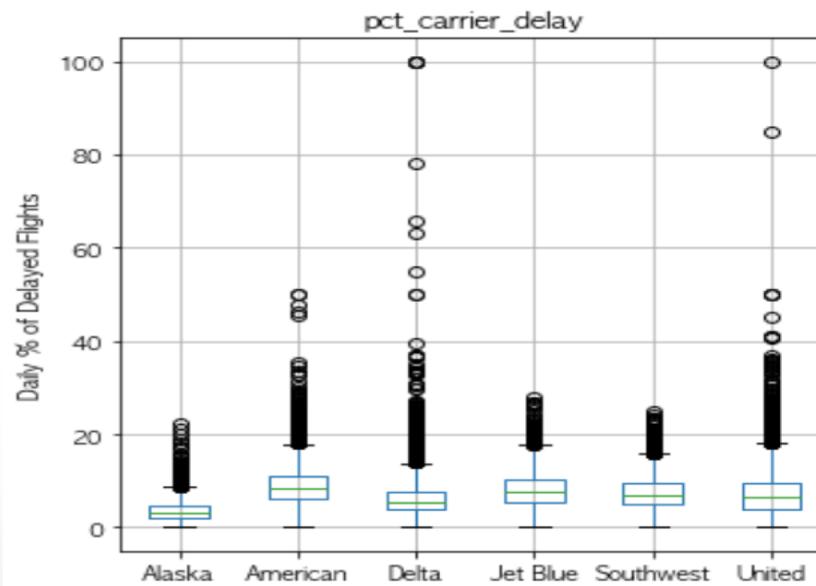
```
df = crosstab.copy().loc['A':'G',:]  
df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged Off':'Late'].div(df['All'], axis=0)  
df['All'] = df['All'] / sum(df['All'])  
perc_crosstab = df  
print(perc_crosstab)
```

status	Charged Off	Current	Fully Paid	Late	All
grade					
A	0.021548	0.690454	0.281528	0.006470	0.160746
B	0.040054	0.709013	0.235401	0.015532	0.293529
C	0.049828	0.735702	0.191495	0.022974	0.268039
D	0.067410	0.717328	0.184189	0.031073	0.164708
E	0.081657	0.707936	0.170929	0.039478	0.077177
F	0.118258	0.654371	0.180409	0.046962	0.028614
G	0.126196	0.614008	0.198396	0.061401	0.007187

# 다면량 탐색

- ❖ 범주형 과 수치형 변수
  - ✓ 상자 그림 이용

```
airline_stats = pd.read_csv("./data/airline_stats.csv")
airline_stats.head()
ax = airline_stats.boxplot(by='airline', column='pct_carrier_delay',
                           figsize=(5, 5))
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
plt.suptitle('')
plt.tight_layout()
plt.show()
```

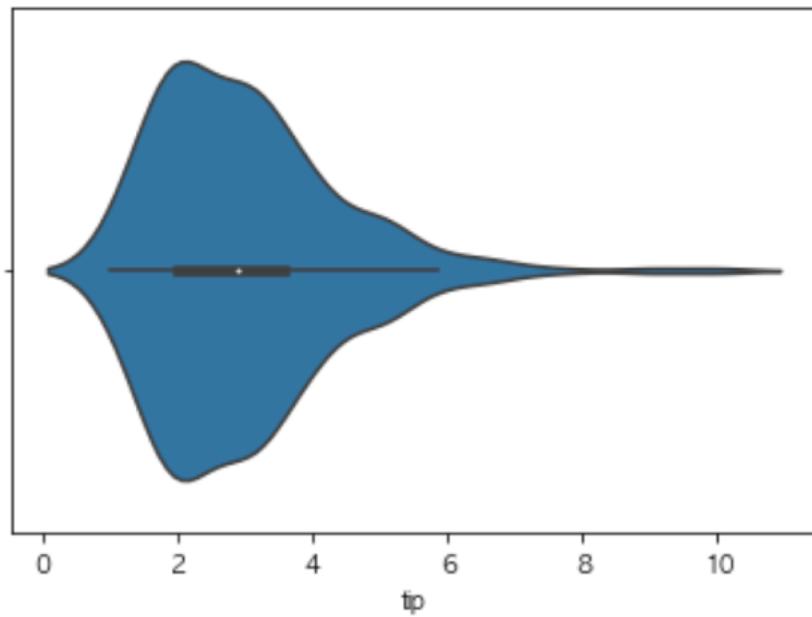


# 다면량 탐색

## ❖ 바이올린 차트

- ✓ 상자 그림을 보완한 형태로 y축을 따라 밀도 추정 결과를 동시에 시각화
- ✓ 밀도 분포 모양을 좌우 대칭으로 서로 겹쳐지도록 해놓고 보면 바이올린 과 유사한 모양

```
sns.violinplot(tips['tip'])
```

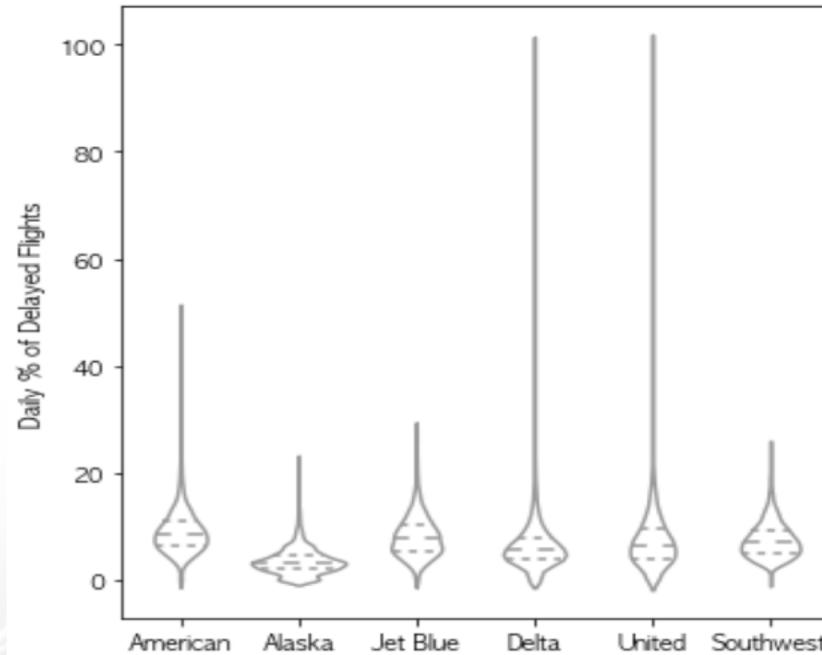


# 다면량 탐색

## ❖ 바이올린 차트

```
fig, ax = plt.subplots(figsize=(5, 5))
sns.violinplot(data=airline_stats, x='airline', y='pct_carrier_delay',
                ax=ax, inner='quartile', color='white')
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
```

```
plt.tight_layout()
plt.show()
```



# 다면량 탐색

## ❖ 조건화

- ✓ 조건화(conditioning)라는 개념을 통해 두 변수 비교용 도표(산점도, 육각형 구간, 상자 그림)를 여러 변수를 비교하는 용도로 확장하여 활용
- ✓ 주택 크기와 과세 평가액 간의 관계를 볼 때 우편번호 별로 데이터를 묶어서 도식화

```
zip_codes = [98188, 98105, 98108, 98126]
kc_tax_zip = kc_tax0.loc[kc_tax0.ZipCode.isin(zip_codes),:]
kc_tax_zip

def hexbin(x, y, color, **kwargs):
    cmap = sns.light_palette(color, as_cmap=True)
    plt.hexbin(x, y, gridsize=25, cmap=cmap, **kwargs)

g = sns.FacetGrid(kc_tax_zip, col='ZipCode', col_wrap=2)
g.map(hexbin, 'SqFtTotLiving', 'TaxAssessedValue',
      extent=[0, 3500, 0, 700000])
g.set_axis_labels('Finished Square Feet', 'Tax Assessed Value')
g.set_titles('Zip code {col_name:.0f}')

plt.tight_layout()
plt.show()
```

# 다면량 탐색

## ❖ 조건화

