

데이터 처리 및 분석 이론과 실습

2022.04

목 차

- 0. 개발 환경 구축
- 1. 과학 계산 패키지 (Numpy)
- 2. 데이터 분석 패키지 (Pandas)
- 3. 시각화 패키지 (Matplotlib)
- 4. 데이터 분석 사례

0. 개발 환경 구축

❖ 개인 PC에 설치

- Anaconda3 + VS Code 추천
- VS Code에서 Jupyter Notebook 사용

❖ Cloud 활용

- Google Colab 활용
- 크롬 브라우저와 Google ID만 있으면 어디에서든 접속 가능
- 인공지능, 데이터분석에 사용되는 대부분의 모듈(라이브러리)은 이미 설치되어 있음
- 한글 관련 모듈은 접속할 때 마다 설치해 주어야 함
- GPU, TPU 등을 무상으로 사용할 수 있음
- 한글 폰트 설치 및 브라우저 설정
 - D2 Coding 글꼴 설치 (<https://github.com/naver/d2codingfont>)
 - 크롬 브라우저 설정 > 글꼴 맞춤설정 > 고정폭 글꼴 > D2Coding
 - 크롬 브라우저 설정 > 고급 > 다운로드 전에 각 파일의 저장위치 확인 Check

0. 개발 환경 구축

- Colab 설정
 - 도구 > 설정 > 편집기

설정

사이트

편집기

Colab Pro

기타

편집기 키 바인딩
default ▼

글꼴 크기
20 ▼ px

들여쓰기 너비(공백 개수)
4 ▼

세로 눈금자 열
80

☒ 코드 완성 제안을 자동으로 표시

☒ 행 번호 표시

☐ 들여쓰기 가이드 표시

☐ 편집기에서 코드 폴딩을 사용 설정합니다.

☒ 코드 셀에서 자동으로 괄호 및 인용부호 닫기

☒ Enter 키를 눌러 제안 수락

취소

저장

1. Numpy

❖ 넘파이란?

- 파이썬 과학 처리 패키지
 - Numerical Python
 - 파이썬의 고성능 과학 계산을 위한 패키지
 - Matrix와 Vector와 같은 Array 연산의 사실상의 표준
- 특징
 - 일반 List에 비해 빠르고, 메모리를 효율적으로 사용
 - 반복문 없이 데이터 배열에 대한 처리를 지원함
 - 선형대수와 관련된 다양한 기능을 제공함
 - C, C++ 등의 언어와 통합 가능
- References
 - <https://docs.scipy.org/doc/numpy/user/quickstart.html>
 - 데이터 사이언스 스쿨 (데이터 과학을 위한 파이썬 기초)
<https://datascienceschool.net/view-notebook/39569f0132044097a15943bd8f440ca5>
 - Numpy 강좌 <https://www.youtube.com/playlist?list=PLBHVuYlKEkULZLnKLzRq1CnNBOBIBTkqp>

1. Numpy

❖ ndarray(Numpy Dimensional Array)

- import

```
import numpy as np    # 표준화되어 있음
```

- Array 생성

```
test_array = np.array([1, 4, 5, 8], float)
```

```
print(test_array)
```

```
type(test_array[3])
```

```
print(test_array.dtype)    # Array 전체의 데이터 타입을 반환함
```

```
print(test_array.shape)    # Array의 shape(차원 구성)을 반환함
```

- numpy는 np.array 함수를 활용하여 배열을 생성함 → ndarray
- numpy는 하나의 데이터 타입만 배열에 넣을 수 있음
- List와 가장 큰 차이점, Dynamic typing(예, [1, 2, "5", 4.2]) not supported
- C의 Array를 사용하여 배열을 생성함

1. Numpy

❖ Array shape

- Vector (1차원)

```
test_array = np.array([1, 4, 5, 8], float)
```

➔ shape은 (4,) : 1차원에 4개의 element가 있는 벡터

- Matrix (2차원)

```
matrix = [[1,2,5,8], [2,3,4,9], [4,5,6,7]]  
np.array(matrix, int).shape
```

➔ shape은 (3, 4) : 행이 3개, 열이 4개인 매트릭스

- Tensor (3차원)

```
tensor = [[[1,2,5,8], [2,3,4,9], [4,5,6,7]],  
          [[1,2,5,8], [2,3,4,9], [4,5,6,7]],  
          [[1,2,5,8], [2,3,4,9], [4,5,6,7]],  
          [[1,2,5,8], [2,3,4,9], [4,5,6,7]]]  
np.array(tensor, int).shape
```

➔ shape은 (4, 3, 4) : 평면이 4개, 행이 3개, 열이 4개인 텐서

1. Numpy

❖ Array shape

- ndim & size

```
np.array(tensor, int).ndim    # 3, number of dimension
np.array(tensor, int).size    # 48
```

- dtype

```
np.array([[1, 2, 3], [4.5, '5', '6']], dtype=np.float32)
array([[1. , 2. , 3. ],
       [4.5, 5. , 6. ]], dtype=float32)
```

- Single element가 가지는 데이터 타입
- C의 데이터 타입과 호환
- nbytes – ndarray object의 메모리 크기를 바이트 단위로 반환함

- reshape

- Array의 shape을 변경함 (element의 개수는 동일)

1. Numpy

❖ Array shape

▪ reshape

```
test_matrix = [[1,2,3,4], [5,6,7,8]]  
np.array(test_matrix).shape → (2, 4)  
np.array(test_matrix).reshape(8, ) → array([1,2,3,4,5,6,7,8])  
np.array(test_matrix).reshape(8, ).shape → (8, )
```

- Array의 shape을 변경함 (element의 개수는 동일)
- Array의 size만 같다면 다차원으로 자유로이 변형가능

```
np.array(test_matrix).reshape(2, 4).shape → (2, 4)  
np.array(test_matrix).reshape(-1, 2).shape → (4, 2)  
-1: size를 기반으로 row 개수 선정  
np.array(test_matrix).reshape(2, 2, 2).shape → (2, 2, 2)
```

▪ flatten

```
test_matrix = [[[1,2,3,4], [5,6,7,8]], [[2,3,4,5], [6,7,8,9]]]  
np.array(test_matrix).flatten()  
→ array([1,2,3,4,5,6,7,8,2,3,4,5,6,7,8,9])
```

- 다차원 array를 1차원 array로 변환

1. Numpy

❖ Indexing & slicing

▪ Indexing

```
a = np.array([[1,2,3], [4,5,6]], int)
print(a)
print(a[0,0])    # 2차원 배열 표기법 1
print(a[0][0])   # 2차원 배열 표기법 2
a[0, 0] = 1
```

- List와 달리 이차원 배열에서 [0, 0]과 같은 표기법을 제공함
- Matrix일 경우 앞은 행(row) 뒤는 열(column)을 의미함

▪ Slicing

```
a = np.array([[1,2,3,4,5], [6,7,8,9,10]], int)
a[:, 2:]      # 전체 row의 2열 이상
a[1, 1:3]     # row 1의 1~2열
a[1:3]        # 1 row ~ 2 row 전체, column은 무시
a[:, ::2]     # step 가능
```

- List와 달리 행과 열 부분을 나눠서 slicing이 가능함
- Matrix의 부분 집합을 추출할 때 유용함

1. Numpy

❖ Creation function

▪ arange

```
np.arange(10) # arange – List의 range와 같은 효과
➔ array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
np.arange(0, 5, 0.5) # floating point도 표시가능
➔ array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
np.arange(0, 5, 0.5).tolist() # List로 만들 수 있음
np.arange(30).reshape(5, 6) # size가 같으면 가능

np.arange(-2, 2.01, 0.02) # 201개의 원소
np.linspace(-2, 2, 201) # 201개의 원소
```

▪ ones, zeros and empty

```
np.zeros(shape=(10,), dtype=np.int8) # 원소가 10개인 벡터 생성
➔ array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int8)
np.ones((2, 5)) # 2 x 5 – 값이 1인 matrix 생성
➔ array([[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]])
np.empty((3, 5)) # 메모리가 초기화되어 있지 않음
```

- empty – shape만 주어지고 비어있는 ndarray 생성

1. Numpy

❖ Creation function

- identity (단위 행렬 생성)

```
np.identity(n=3, dtype=np.int8)
→ array([1, 0, 0],
        [0, 1, 0],
        [0, 0, 1]), dtype=int8)
```

n → number of rows

```
np.identity(5)
→ array([1., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0.],
        [0., 0., 1., 0., 0.],
        [0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 1.]])
```

- eye (대각선이 1인 행렬)

```
np.eye(N=3, M=5, dtype=np.int8)
→ array([[1, 0, 0, 0, 0],
        [0, 1, 0, 0, 0],
        [0, 0, 1, 0, 0]], dtype=int8)
```

1. Numpy

❖ Creation function

- Random sampling(데이터 분포에 따른 sampling으로 array를 생성)

```
np.random.seed(seed=1000) # 시드로 난수 생성 초기값 지정
```

```
np.random.uniform(0, 1, 10).reshape(2,5) # 균등 분포  
      최소 최대 개수
```

```
np.random.rand(10)
```

```
np.random.normal(0, 1, 10).reshape(2,5) # 정규 분포  
      평균 표준편차 개수
```

```
np.random.randn(10)
```

```
np.random.binomial(n, p, size) # 이항 분포
```

```
np.random.standard_t(df, size) # t-분포
```

1. Numpy

❖ Operation function

▪ Sum

```
test_array = np.arange(1,11)
test_array.sum(dtype=np.float) → 55.0
```

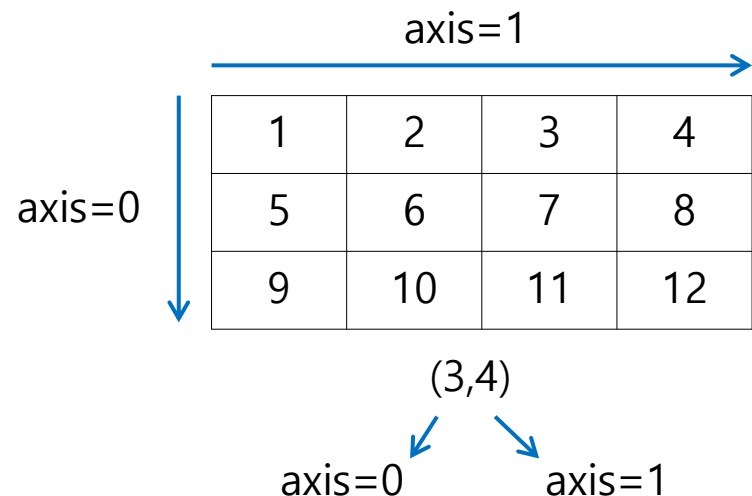
▪ Axis

- 모든 operation function을 실행할 때, 기준이 되는 dimension 축

```
test_array = np.arange(1,13).reshape(3,4)
→ array([1, 2, 3, 4],
        [5, 6, 7, 8],
        [9,10,11,12]))
```

```
test_array.sum(axis=1)
→ array([10, 26, 42])
```

```
test_array.sum(axis=0)
→ array([15, 18, 21, 24])
```



1. Numpy

❖ Operation function

▪ Sorting

```
test_array = np.array([[4,3,5,7],[1,12,11,9],[2,5,1,14]])  
np.sort(test_array)    # axis=1
```

```
np.sort(test_array, axis=0)
```

```
test_array.sort(axis=1)    # in-place method
```

```
a = np.array([42,38,12,25])  
indices = np.argsort(a)    # 순서만 알고 싶을 때  
a[indices]
```

```
a[indices][::-1]          # 내림차순
```

▪ Mathematical functions

지수 함수: exp, expm1, exp2, log, log10, loglp, log2, power, sqrt

삼각 함수: sin, cos, tan, arcsin, arccos, arctan

Hyperbolic: sinh, cosh, tanh, arcsinh, arccosh, arctanh

```
np.exp(test_array)  
np.sqrt(test_array)
```

1. Numpy

❖ Operation function

- Concatenate (Numpy array를 합치는 함수)

```
a = np.array([[1, 2], [3, 4]])
```

```
b = np.array([[5, 6]])
```

```
np.vstack((a,b))
```

```
→ array([[1, 2],  
         [3, 4],  
         [5, 6]])
```

```
np.concatenate((a,b), axis=0) # 위의 결과와 동일
```

```
a = np.array([[1], [2], [3]])
```

```
b = np.array([[2], [3], [4]])
```

```
np.hstack((a,b))
```

```
→ array([[1, 2],  
         [2, 3],  
         [3, 4]])
```

```
a = np.array([[1, 2], [3, 4]])
```

```
b = np.array([[5, 6]])
```

```
np.concatenate((a, b.T), axis=1)
```

```
→ array([[1, 2, 5],  
         [3, 4, 6]])
```

T - Transpose

1. Numpy

❖ Array operation

- Operations btw arrays (기본적인 사칙 연산 지원)

```
test_a = np.array([[1,2,3], [4,5,6]], float)
test_a + test_a
→ array([2., 4., 6.],
        [8., 10., 12.]])
```

- Dot product

```
test_a = np.arange(1,7).reshape(2,3)    # Matrix 곱셈
test_b = np.arange(7,13).reshape(3,2)    # (l,m) x (m,n) → (l,n)
test_a.dot(test_b)
→ array([ 58,  64],
        [139, 154])
```

- Transpose

```
test_a = np.arange(1,7).reshape(2,3)
test_a.transpose()
test_a.T
```

- Broadcasting (Shape이 다른 배열간 연산 지원)

```
test_a + 3
```

1. Numpy

❖ Comparison

▪ All & Any

```
a = np.arange(10)
```

```
→ array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.any(a>5) → True # any – 하나라도 조건에 만족하면 True
```

```
np.any(a<0) → False
```

```
np.all(a>5) → False # all – 모두가 조건을 만족해야 True
```

```
np.all(a<10) → True
```

```
a > 5
```

```
→ array([False, False, False, False, False, False, True, True, True, True], dtype=bool)
```

▪ Logical operation

```
a = np.array([1, 3, 0], float)
```

```
b = np.logical_and(a>0, a<3) # and 조건
```

```
→ array([True, False, False], dtype=bool)
```

```
c = np.logical_not(b)
```

```
→ array([False, True, True], dtype=bool)
```

```
np.logical_or(b,c)
```

```
→ array([True, True, False], dtype=bool)
```

1. Numpy

❖ Comparison

- `argmax` & `argmin` (array내 최대값 또는 최소값의 index를 리턴)

```
a = np.array([1,2,4,5,8,78,23,3])
```

```
np.argmax(a), np.argmin(a)
```

```
→ (5, 0)
```

```
a = np.array([[1,2,4,7],[9,88,6,45],[8,78,23,3]])
```

```
np.argmax(a, axis=1)
```

```
→ array([3, 1, 1])
```

```
np.argmax(a, axis=0)
```

```
→ array([1, 1, 2, 1])
```

```
np.argmin(a, axis=1)
```

```
→ array([0, 2, 3])
```

```
np.argmin(a, axis=0)
```

```
→ array([0, 0, 0, 2])
```

1	2	4	7
9	88	6	45
8	78	23	3

1. Numpy

❖ Boolean & fancy index

▪ Boolean index

```
test_array = np.array([1,4,0,2,3,8,9,7], float)
test_array > 3
→ array([False, True, False, False, False, True, True, True],
        dtype=bool)

test_array[test_array > 3]    # 조건이 True인 index의 element만 추출
→ array([4., 8., 9., 7.]
```

▪ Fancy index

```
a = np.array([2, 4, 6, 8], float)
b = np.array([0, 0, 1, 3, 2, 1], int)    # 반드시 integer로 선언

a[b]                                     # b 배열의 값을 인덱스로 하여 a의 값들을 추출함
→ array([2., 2., 4., 8., 6., 4.])        # bracket index

a.take(b)    # take 함수: bracket index와 같은 효과
```

1. Numpy

❖ 기술 통계

- 표본 평균 – `np.mean()`
- 표본 분산
 - `np.var(x)` # 모분산, 분모가 N
 - `np.var(x, ddof=1)` # 표본분산, 분모가 N-1
- 표본 표준편차 – `np.std()`
- 최대값, 최소값 – `np.max()`, `np.min(x)`
- 중앙값 – `np.median()`
- 사분위수(quartile)
 - `np.percentile(x, 0)` # 최소값
 - `np.percentile(x, 25)` # 1사분위 수
 - `np.percentile(x, 50)` # 2사분위 수
 - `np.percentile(x, 75)` # 3사분위 수
 - `np.percentile(x, 100)` # 최대값

1. Numpy - 연습 문제

1. 넘파이를 사용하여 다음과 같은 행렬을 만드시오.

```
10 20 30 40
50 60 70 80
```

2. 다음 행렬과 같은 행렬이 있다.

```
m = np.array([[ 0, 1, 2, 3, 4],
               [ 5, 6, 7, 8, 9],
               [10, 11, 12, 13, 14]])
```

- 1) 이 행렬에서 값 7 을 인덱싱한다.
- 2) 이 행렬에서 값 14 을 인덱싱한다.
- 3) 이 행렬에서 배열 [6, 7] 을 슬라이싱한다.
- 4) 이 행렬에서 배열 [7, 12] 을 슬라이싱한다.
- 5) 이 행렬에서 배열 [[3, 4], [8, 9]] 을 슬라이싱한다.

3. 2번의 행렬 m을 1차원 벡터 f 로 변환한 후 다음의 문제를 푸시오.

- 1) 이 배열에서 3의 배수를 찾아라.
- 2) 이 배열에서 4로 나누면 10이 남는 수를 찾아라.
- 3) 이 배열에서 3으로 나누면 나누어지고 4로 나누면 10이 남는 수를 찾아라.

4. 값을 직접 입력하지 말고
우측의 행렬을 만드시오.

2, 1, 0, 0, 0
3, 2, 1, 0, 0
0, 3, 2, 1, 0
0, 0, 3, 2, 1
0, 0, 0, 3, 2

1. Numpy - 연습 문제

5. 0에서 10까지 랜덤 실수값으로 이루어진 5 x 6 형태의 데이터 행렬을 만들고 이 데이터에 대해 다음과 같은 값을 구하시오.
- 1) 전체의 최댓값
 - 2) 각 행의 합
 - 3) 각 행의 최댓값
 - 4) 각 열의 평균
 - 5) 각 열의 최솟값
6. 다음 배열은 첫번째 행(row)에 학번, 두번째 행에 영어 성적, 세번째 행에 수학 성적을 적은 배열이다. 영어 성적을 기준으로 각 열(column)을 재정렬하시오.
- ```
array([[1, 2, 3, 4],
 [46, 99, 100, 71],
 [81, 59, 90, 100]])
```
7. 주사위를 100번 던지는 가상 실험을 파이썬으로 작성하고, 던져서 나오는 숫자의 평균을 구하시오.
8. 가격이 10,000원인 주식이 있다. 이 주식의 일간 수익률(%)은 기댓값이 0%이고 표준편차가 1%인 표준 정규 분포를 따른다고 하자. 250일 동안의 주가를 무작위로 생성하시오.

---

## 목 차

1. 과학 계산 패키지 (Numpy)
2. **데이터 분석 패키지 (Pandas)**
3. 시각화 패키지 (Matplotlib)
4. 데이터 분석 사례



## 2. Pandas

### ❖ Pandas란?

- 데이터 분석에 가장 많이 사용되는 파이썬 패키지
- 열과 행으로 구성된 테이블 형태의 데이터를 다루는 데 주로 사용
- Series 클래스와 DataFrame 클래스로 구성
- 사용하는 방법이 매우 다양하고 어려워 데이터 분석을 하는 동안 계속 배워나가는 패키지
- References
  - API 문서 - <https://pandas.pydata.org/docs/reference/index.html>
  - Cheat Sheet - [https://pandas.pydata.org/Pandas\\_Cheat\\_Sheet.pdf](https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf)
  - 데이터 사이언스 스쿨 - <https://datascienceschool.net/intro.html>

## 2. Pandas

### ❖ Series Class

- Numpy 1차원 배열 + 데이터의 의미를 표시하는 인덱스
- DataFrame의 하나의 열

- Series 생성

```
s = pd.Series([9904312, 3448737, 2890451, 2466052],
 index=["서울", "부산", "인천", "대구"])
```

- Series 인덱싱

```
s[1], s["부산"] # Single value
s[[0, 3, 1]] # Series
```

- Series 연산

```
s / 1000000
```

## 2. Pandas

### ❖ Series Class

- 시리즈와 딕셔너리 자료형

```
for key, value in s.items():
 print(f"{key}: {value}명")
```

- 인덱스 기반 연산

```
s2 = pd.Series({"서울":9631482,"부산":3393191,"인천":2632035,"대전":1490158},
 index=["부산", "서울", "인천", "대전"])
ds = s - s2
```

- 속성과 메소드

```
s.values, s.index
```

```
s.count()
```

# 데이터 갯수 세기(NaN은 제외)

```
s.value_counts()
```

# 카테고리 값 세기

```
s.unique(), s.nunique()
```

# 카테고리 이름, 갯수

```
s.sum(), s.mean()
```

# 합계, 평균

```
s.sort_values(), s.sort_index()
```

## 2. Pandas

### ❖ DataFrame Class 생성

- 2차원 데이터 + 행 인덱스 + 열 인덱스
- 기본이 되는 생성 방법
  - 우선 하나의 열이 되는 데이터를 리스트나 일차원 배열로 준비한다.
  - 이 각각의 열에 대한 이름(라벨)을 키로 가지는 딕셔너리를 만든다.
  - 이 데이터를 DataFrame 클래스 생성자에 넣는다.
  - 동시에 열방향 인덱스는 columns 인수로, 행방향 인덱스는 index 인수로 지정한다.

```
data = {
 "국어": [80, 90, 70, 30],
 "영어": [90, 70, 60, 40],
 "수학": [90, 60, 80, 70]
}
columns = ["국어", "영어", "수학"] # 생략 가능
index = ["춘향", "몽룡", "향단", "방자"]

df = pd.DataFrame(data, index=index, columns=columns)
```

## 2. Pandas

### ❖ DataFrame 파일 입출력

- 지원하는 파일 형식

CSV, Excel, JSON, SAS, SQL 등

- 한글 인코딩에 주의

- EUC-KR(CP949): 국가 표준, MS-Windows, 2바이트 완성형
- UTF-8: Web, 대부분의 프로그램에서 사용하는 코드, 초성+중성+종성의 조합형

- CSV 파일 입력

```
%%writefile sample1.csv
```

```
c1,c2,c3
```

```
1,1.11,one
```

```
2,2.22,two
```

```
3,3.33,three
```

```
pd.read_csv('sample1.csv')
```

- CSV 파일 출력

```
df.to_csv('sample2.tsv', sep='\t', index=False)
```

## 2. Pandas

### ❖ DataFrame 인덱싱

- 리스트나 배열 인덱싱과는 다르게 열을 먼저 선택하고 행을 선택함

```
df['c1'][0] # 1
```

```
df.c1[0] # 열의 명칭을 바로 쓸 수 있음. (JS의 Object 스타일)
```

```
df['c1'] # 시리즈
```

```
df[['c1', 'c3']] # 데이터프레임
```

- loc 인덱서 - [행인덱스, 열인덱스]

```
df.loc[1, 'c2'] # 2.22
```

```
df.loc[1] # 시리즈
```

- iloc 인덱서 - [숫자 행인덱스, 숫자 열인덱스]

```
df.iloc[1, 1] # 2.22
```

```
df.iloc[:2, 1:] # 슬라이싱 가능
```

## 2. Pandas

### ❖ DataFrame 데이터 조작

- 탐색하기 위한 메소드

|                             |                |
|-----------------------------|----------------|
| <code>head(), tail()</code> | # 앞/뒤의 5줄 보여주기 |
| <code>info()</code>         | # 열의 정보        |
| <code>describe()</code>     | # 요약 통계량       |

- sum 메소드

```
np.random.seed(1)
df2 = pd.DataFrame(np.random.randint(10, size=(4, 8)))
df2.sum(axis=1) # 행 합계
df2.sum() 또는 df2.sum(axis=0) # 열 합계
```

- mean 메소드 - 결과는 실수로

|                                             |        |
|---------------------------------------------|--------|
| <code>df2.mean(axis=1)</code>               | # 행 평균 |
| <code>df2.mean() 또는 df2.mean(axis=0)</code> | # 열 평균 |

## 2. Pandas

### ❖ DataFrame 데이터 조작

- 결측치 처리

```
import seaborn as sns
dt = sns.load_dataset('titanic')
```

```
dt.info() # Null 데이터가 있는지 확인
dt.isnull().sum() # 0이 아니면 Null(Na) 데이터가 있는 것임
```

```
dt.age.fillna(dt.age.mean(), inplace=True)
```

age열의 결측치를 age  
열의 평균으로 대체

데이터프레임 dt의  
데이터를 변경함

```
dt.dropna(subset=['embarked'], inplace=True) # 행 삭제
dt.dropna(axis=1, inplace=True) # 넓값이 있는 모든 열 삭제
dt.drop(columns=['deck'], inplace=True) # 특정한 열 삭제
```

- astype 메소드 - 데이터 타입을 변경



## 2. Pandas

### ❖ DataFrame 데이터 조작

- apply 메소드 - 행이나 열 단위로 더 복잡한 처리를 할 경우

```
df3 = pd.DataFrame({
 'A': [1, 3, 4, 3, 4],
 'B': [2, 3, 1, 2, 3],
 'C': [1, 5, 2, 4, 4]
})
```

# 각 열의 최대값과 최소값의 차이를 구하고 싶을 때

```
df3.apply(lambda x: x.max() - x.min())
```

```
df3.apply(lambda x: x.max() - x.min(), axis=1) # 행에 대해 적용
```

# 타이타닉호의 승객 중 나이 20살을 기준으로 성인(adult)과 미성년자(child)를

# 구별하는 라벨 열을 만들 때

```
titanic["adult/child"] =
 titanic.apply(lambda r: "adult" if r.age >= 20 else "child", axis=1)
```

## 2. Pandas

### ❖ DataFrame 데이터 조작 - 문자열 관련 메소드 (.str 이 앞에 붙음)

- 실습 데이터 준비(행정표준관리시스템: <https://www.code.go.kr/index.do>)

```
from google.colab import files
uploaded = files.upload()
filename = list(uploaded.keys())[0]
```

```
df = pd.read_csv(filename, sep='\t', encoding='euc-kr')
df = df[df.폐지여부 == '존재']
df.head()
```

- 인덱싱 : .str[]

```
df['법정동명'].str[:5].tail() # 앞 5글자만 추출
```

- 찾기 : .str.find()

```
df['법정동명'].str.find('동').head()
```

- 분할 : .str.split()

```
df['법정동명'].str.split().head() # 공백으로 분할
```

## 2. Pandas

### ❖ DataFrame 데이터 조작 - 문자열 관련 메소드 (.str 이 앞에 붙음)

- 포함글자 인식 : .str.contains()

```
df[df['법정동명'].str.contains('강서구')].head() # 강서구 포함 법정동 추출
```

- 문자 대체 : .str.replace()

```
스트링 메소드와는 달리 정규 표현식을 사용할 수 있음
```

```
df['법정동명'].str.replace(' ', '_').head() # 공백을 _로 대체
```

- 공백제거 : .str.strip()

```
df2 = pd.DataFrame({'c1': [' Abc ', ' xYz '], 'c2': [' abC ', ' xyZ ']}).
```

```
df2.c1.str.strip()
```

- 소문자 변경: .str.lower()

```
df2.c2.str.lower()
```

```
df2.c2.apply(lambda x: x.lower())
```

## 2. Pandas

### ❖ DataFrame 인덱스 조작

- `set_index` : 기존의 행 인덱스를 제거하고 데이터 열 중 하나를 인덱스로 설정

```
np.random.seed(2021)
data = np.vstack([list('ABCDE'), np.round(np.random.rand(3, 5), 2)])

df = pd.DataFrame(data.T, columns=['C1', 'C2', 'C3', 'C4'])
df2 = df.set_index("C1") # df는 변하지 않고, df2만 변함

df.set_index("C1", inplace=True) # df 자체를 변하게 할 경우
df.index.name = 'Index'
```

- `reset_index` : 기존의 행 인덱스를 제거하고 인덱스를 데이터 열로 추가

```
df3 = df.reset_index() # df는 변하지 않고, df3만 변함

df.reset_index(inplace=True) # df 자체를 변하게 할 경우
df.rename(columns={'Index': 'C1'}, inplace=True)
```

## 2. Pandas

### ❖ DataFrame 합성

- merge 함수 : 두 데이터 프레임의 공통 열 혹은 인덱스를 기준으로 두 개의 테이블을 합침

```
df1 = pd.DataFrame({
 '고객번호': [1001, 1002, 1003, 1004, 1005, 1006, 1007],
 '이름': ['둘리', '도우너', '또치', '길동', '희동', '마이콜', '영희']
}, columns=['고객번호', '이름'])
```

```
df2 = pd.DataFrame({
 '고객번호': [1001, 1001, 1005, 1006, 1008, 1001],
 '금액': [10000, 20000, 15000, 5000, 100000, 30000]
}, columns=['고객번호', '금액'])
```

```
df3 = pd.merge(df1, df2) # Inner join
```

```
df4 = pd.merge(df1, df2, how='outer') # Outer join
```

- join 메소드

```
df3 = df1.set_index('고객번호')
df4 = df2.set_index('고객번호')
df3.join(df4, how='inner')
```

## 2. Pandas

### ❖ DataFrame 그룹 분석

- groupby 메소드
  - 분석하고자 하는 시리즈나 데이터프레임에 groupby 메서드를 호출하여 그룹화를 한다.
  - 그룹 객체에 대해 그룹연산을 수행한다.
- 그룹연산 메소드
  - size, count: 그룹 데이터의 갯수
  - mean, median, min, max: 그룹 데이터의 평균, 중앙값, 최소, 최대
  - sum, prod, std, var, quantile : 그룹 데이터의 합계, 곱, 표준편차, 분산, 사분위수
  - first, last: 그룹 데이터 중 가장 첫번째 데이터와 가장 나중 데이터
- 아이리스 각 품종별 피쳐의 평균은?

```
import seaborn as sns
iris = sns.load_dataset('iris')
```

```
iris.groupby('species').mean()
iris[['sepal_length', 'species']].groupby('species').std()
iris.groupby('species')['sepal_length'].std()
iris.groupby('species')['sepal_length'].agg(['mean', 'std', 'max', 'min'])
```

## 2. Pandas

### ❖ DataFrame 그룹 분석

- tips 데이터

```
import seaborn as sns
tips = sns.load_dataset('tips')
tips['tip_pct'] = np.round(tips.tip / tips.total_bill * 100, 2)
```

```
tips.groupby('sex')[['tip_pct']].mean()
tips.groupby('smoker')[['tip_pct']].mean()
```

```
tips.groupby(['sex', 'smoker'])[['tip_pct']].mean()
tips.groupby(['sex', 'smoker', 'time'])[['tip_pct']].mean()
```

## 2. Pandas - 연습문제

1) Iris - sns.load\_dataset('iris')

- a. 붓꽃 종(species)별로 꽃잎길이(sepal\_length), 꽃잎폭(sepal\_width), 꽃받침길이(petal\_length), 꽃받침폭(petal\_width)의 평균, 표준편차 등 기초통계량(describe())을 구하시오.
- b. 3분위수(Q3)와 1분위수(Q1)의 차이보다 1.5배가 크거나 작은 데이터는 이상치이다.  
즉,  
     $Q1 - 1.5 * (Q3 - Q1)$  보다 작은 데이터  
     $Q3 + 1.5 * (Q3 - Q1)$  보다 큰 데이터

이 이상치를 제거하고 위의 4가지 항목에 대해서 평균, 표준편차를 구하시오.



## 2. Pandas - 연습문제

### 2) Titanic - sns.load\_dataset('titanic')

- a. 타이타닉호의 승객에 대해 나이와 성별에 의한 카테고리 열인 category1 열을 만드시오.  
category1 카테고리는 다음과 같이 정의됨
  - 1) 20살이 넘으면 성별을 그대로 사용한다.
  - 2) 20살 미만이면 성별에 관계없이 “child”라고 한다.
- b. 타이타닉호의 승객 중 나이를 명시하지 않은 고객은 나이를 명시한 고객의 평균 나이 값이 되도록 titanic 데이터프레임을 고치시오.
- c. 성별, 선실(class)별, 출발지(embark\_town)별 생존율을 구하시오.
- d. 타이타닉호 승객을 ‘미성년자’, ‘청년’, ‘중년’, ‘장년’, ‘노년’ 나이 그룹으로 나누고, 각 그룹별 생존율을 구하시오.  
  

```
bins = [1, 20, 30, 50, 70, 100]
labels = ["미성년자", "청년", "중년", "장년", "노년"]
```
- e. qcut 명령으로 세 개의 나이 그룹을 만들고, 나이 그룹별 남녀 성비와 생존율을 구하시오.

## 2. Pandas - 연습문제

### 3) Mile Per Gallon - `sns.load_dataset('mpg')`

- a. 배기량(displacement) 대비 마력(horsepower) 열(`hp_per_cc`)을 추가하시오.
- b. `name`으로부터 `manufacturer`(제조사)와 모델을 추출하여 새로운 열 `manufacturer`와 `model`을 추가하고, `name` 열은 삭제하시오.
- c. 엔진의 실린더(`cylinders`) 갯수별 연비(`mpg`)의 평균을 구하시오.
- d. 생산지(`origin`)별 배기량 대비 마력(`hp_per_cc`)의 평균을 구하시오.
- e. 모델이 5개 이상인 제조사에 대하여 연비(`mpg`)의 평균이 가장 좋은 제조사 Top 5를 구하시오.

---

## 목 차

1. 과학 계산 패키지 (Numpy)
2. 데이터 분석 패키지 (Pandas)
3. 시각화 패키지 (**Matplotlib**)
4. 데이터 분석 사례

### 3. Matplotlib

#### ❖ Matplotlib 란?

- 개요
  - 파이썬에서 자료를 시각화(Visualization)해주는 패키지
  - Pandas 시리즈와 데이터프레임의 plot 메소드 활용
- 그릴 수 있는 Chart / Plot
  - 라인 플롯(line plot)
  - 산점도(scatter plot)
  - 막대 그래프(bar chart), 히스토그램(histogram)
  - 박스 플롯(box plot)
  - 파이 차트(pie chart)
  - 컨투어 플롯(contour plot), 서피스 플롯(surface plot) 등
- 참고 사이트
  - 갤러리(<http://matplotlib.org/gallery.html>)

### 3. Matplotlib

#### ❖ Matplotlib 란?

- 그래프 그려보기
  - 모듈 импорт

```
import matplotlib.pyplot as plt
%matplotlib inline
```

- 라인 플롯

```
plt.title("Plot")
plt.plot([1, 4, 9, 16]) # y 값 리스트
plt.show()
```

```
plt.title("x ticks")
plt.plot([10, 20, 30, 40], [1, 4, 9, 16]) # x 값 리스트, y 값 리스트
plt.grid()
plt.show()
```

### 3. Matplotlib

#### ❖ 한글 설정

- Local 개발 환경

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
```

```
mpl.rcParams['axes.unicode_minus'] = False # minus 표시
```

```
[(f.name, f.fname) for f in fm.fontManager.ttflist if 'Malgun' in f.name]
[('Malgun Gothic', 'C:\\WINDOWS\\Fonts\\malgun.ttf'),
 ('Malgun Gothic', 'C:\\Windows\\Fonts\\malgun.ttf'),
 ('Malgun Gothic', 'C:\\Windows\\Fonts\\malgunsl.ttf'),
 ('Malgun Gothic', 'C:\\Windows\\Fonts\\malgunbd.ttf'),
 ('Malgun Gothic', 'C:\\WINDOWS\\Fonts\\malgunsl.ttf'),
 ('Malgun Gothic', 'C:\\WINDOWS\\Fonts\\malgunbd.ttf')]
```

```
plt.rcParams["font.family"] = 'Malgun Gothic'
```

```
plt.title("X ticks를 사용한 플롯")
plt.plot([10, 20, 30, 40], [1, 4, 9, 16])
plt.grid()
plt.show()
```

### 3. Matplotlib

#### ❖ 한글 설정

- Colab 환경

```
!apt-get install -y fonts-nanum > /dev/null
!fc-cache -fv > /dev/null
!rm -rf ~/.cache/matplotlib > /dev/null
```

런타임 > 런타임 다시 시작

```
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rcParams['axes.unicode_minus'] = False
plt.rc('font', family='NanumBarunGothic')

plt.title("X ticks를 사용한 플롯")
plt.plot([10, 20, 30, 40], [1, 4, 9, 16])
plt.grid()
plt.show()
```

### 3. Matplotlib

#### ❖ Line Plot

- 데이터가 시간, 순서 등에 따라 어떻게 변화하는지 보여주기 위해 사용
- Series 타입에서 value가 y축, index가 x축

```
x = [0, 1, 2, 3]
y = [0, 1, 4, 9]
plt.plot(x, y)
plt.show()
```

```
s = pd.Series(np.random.randn(10).cumsum(), index=np.arange(10))
plt.plot(s) # s.plot()
```

```
plt.plot(s, color='g', marker='o', linestyle='-')
plt.plot(s, 'bs--')
```

```
plt.plot(s, 'r*:', lw=3, ms=10) # linewidth, markersize
```

color: blue, green, red, cyan, magenta, yellow, black(k), white  
marker: circle(o), 삼각형(^), 역삼각형(v), 사각형(s), 별(\*), 플러스(+)  
linestyle: 직선(-), dashed(--), dash-dot(-.), dotted(:)



### 3. Matplotlib

#### ❖ 산점도(Scatter Plot)

- Anscombe's quartet: 기술 통계치가 거의 같은 4가지 데이터 셋
- x와 y의 관계를 눈으로 확인하고자 할 때

```
x = np.random.randn(100)
y = np.random.randn(100)
plt.scatter(x, y)
plt.show()
```

```
x = np.random.rand(100)
y = np.random.rand(100)
colors = np.random.rand(100)
area = np.pi * (15 * np.random.rand(100))**2
plt.scatter(x, y, label='Samples', s=area, c=colors, alpha=0.5)
```

s: 도형의 크기(size)

c: 도형의 색상(color)

alpha: 투명도 (0-완전 투명, 1-완전 불투명)

```
x1 = np.random.normal(1, 1, size=(100, 1))
x2 = np.random.normal(-2, 4, size=(100, 1))
plt.scatter(x1, x2, color='r', marker='s')
```

### 3. Matplotlib

#### ❖ 막대 그래프(Bar chart), 히스토그램(histogram)

- 특정 그룹의 데이터의 추세와 정량적인 분포

```
import pandas as pd
s2 = pd.Series(np.random.rand(10), index=list('abcdefghij'))
plt.bar(s2.index, s2.values) # s2.plot(kind='bar')
plt.show()
```

```
df2 = pd.DataFrame(np.random.rand(6, 4),
 index = ['one', 'two', 'three', 'four', 'five', 'six'],
 columns = pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df2.plot(kind='barh', figsize=(12,8))
df2.plot(kind='barh', stacked=True)
```

```
s3 = pd.Series(np.random.randn(200))
plt.hist(s3) # s3.hist()
```

```
plt.hist(s3, bins=50) # default는 10
```

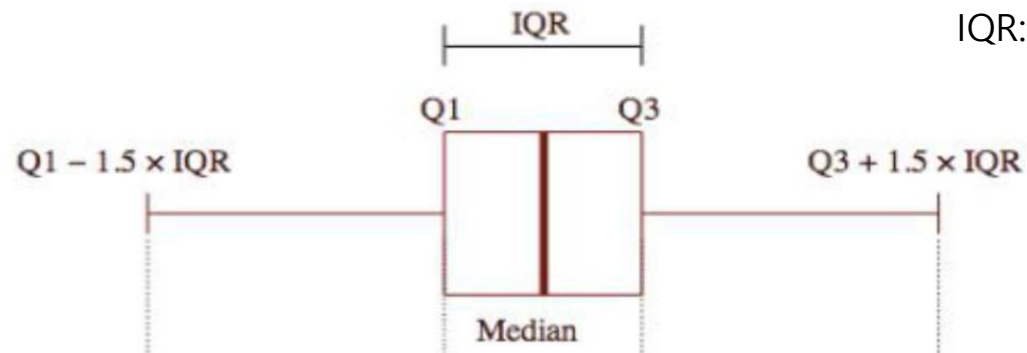
```
plt.hist(s3, bins=100, normed=True)
```

### 3. Matplotlib

#### ❖ 박스 플롯(Box plot)

- 기초 통계량 확인하고자 할 때

```
s4 = np.random.randn(1000)
s5 = np.random.normal(loc=10, scale=2, size=1000)
plt.boxplot([s4, s5])
```



IQR: Inter Quartile Range

```
Setosa 품종
setosa = iris[iris.species == 'setosa']
plt.figure(figsize=(8,10))
plt.boxplot([setosa.sepal_length, setosa.sepal_width,
 setosa.petal_length, setosa.petal_width],
 labels=['sl', 'sw', 'pl', 'pw'])
plt.title('Setosa 품종의 각 피쳐', fontsize=15)
plt.show()
```

### 3. Matplotlib

#### ❖ 파이 차트(Pie Chart)

- 범주별 구성 비율을 원형으로 표현

```
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']

plt.pie(ratio, labels=labels, autopct='%1f%%')
plt.show()
```

#### ❖ 이미지 보기(Imshow)

```
from PIL import Image
image = Image.open(filename)

plt.imshow(image)
plt.axis('off')
plt.show()
```

### 3. Matplotlib

#### ❖ 제목, 레이블, 범례 작성

```
import numpy as np
xs = np.linspace(0, 2*np.pi, 800) # 0 ~ 2*pi 구간을 800 등분
ysin = np.sin(xs)
ycos = np.cos(xs)

plt.plot(xs, ysin, label='Sine curve')
plt.plot(xs, ycos, label='Cosine curve')
plt.xlim([-0.1, 2*np.pi+0.1])
plt.ylim([-1.2, 1.2])
plt.xlabel('X-Value'); plt.ylabel('Y-Value')
plt.title('삼각함수')
plt.legend() # label을 범례로 표시

for y_val in [-1, 0, 1]:
 plt.axhline(y=y_val, color='k', linewidth=0.5) # 수평선
for x_val in np.arange(0, 2*np.pi+0.01, np.pi/2):
 plt.axvline(x=x_val, c='k', lw=0.5) # 수직선

plt.show()
plt.savefig('image.png') # image.png 로 저장
```

### 3. Matplotlib

#### ❖ 여러 개의 그래프를 그리는 방법(Subplot)

```
def f(t):
 return np.exp(-t) * np.cos(2*np.pi*t)

def g(t):
 return np.sin(np.pi*t)

t1 = np.arange(0.0, 5.0, 0.01)
t2 = np.arange(0.0, 5.0, 0.01)

plt.subplot(221) # 2 x 2, 그래프중 첫번째
plt.plot(t1, f(t1))
plt.subplot(2, 2, 2)
plt.plot(t2, g(t2))

plt.subplot(223)
plt.plot(t1, f(t1), 'r-')
plt.subplot(224)
plt.plot(t2, g(t2), 'r-')

plt.show()
```

### 3. Matplotlib

#### ❖ 여러 개의 그래프를 그리는 방법

- Anscombe's Quartet

```
import seaborn as sns
ans = sns.load_dataset('anscombe')

fig, axes = plt.subplots(2, 2, figsize=(10,8),
 sharex=True, sharey=True)
for index, roman in enumerate(['I', 'II', 'III', 'IV']):
 ds = ans[ans.dataset == roman]
 ax = axes[index//2, index%2]
 ax.plot(ds.x, ds.y, 'o', markersize=10)
 ax.plot([3, 20], [4.5, 13], 'r-', lw=2) # $y = 0.5 * x + 3$
 ax.set_title(f'Dataset {roman}')

fig.suptitle("Anscombe's Quartet", fontsize=16)
plt.tight_layout()
plt.show()
```

### 3. Matplotlib

#### ❖ Pandas 시각화

- plot 메소드

```
np.random.seed(0)
df1 = pd.DataFrame(np.random.randn(100, 3),
 index=pd.date_range('1/1/2021', periods=100),
 columns=['A', 'B', 'C']).cumsum()

df1.plot()
plt.title("Pandas의 Plot메소드 사용 예")
plt.xlabel("시간")
plt.ylabel("Data")
plt.show()
```

- 다양한 plot

- kind 인수값 설정
- bar, pie, hist, kde, box, scatter, area 등
- plot.bar 와 같이 직접 메소드로 이용도 가능



### 3. Matplotlib

#### ❖ Pandas 시각화

- 다양한 플롯 예

```
iris = sns.load_dataset("iris") # 붓꽃 데이터
titanic = sns.load_dataset("titanic") # 타이타닉호 데이터
```

```
df2 = iris.groupby(iris.species).mean()
df2.columns.name = "feature"
df2.plot(kind='bar', rot=0)
plt.title("각 종의 Feature별 평균")
plt.xlabel("평균"); plt.ylabel("종"); plt.ylim(0, 8)
plt.show()
```

```
df3 = titanic.pclass.value_counts()
df3.plot.pie(autopct='%.2f%')
plt.title("선실별 승객 수 비율")
plt.axis('equal')
plt.show()
```

### 3. Matplotlib

#### ❖ 다른 시각화 패키지 - Seaborn

- 세련된 그래프
- <https://seaborn.pydata.org/examples/index.html>

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

```
titanic = sns.load_dataset("titanic")
sns.countplot(x="class", data=titanic)
plt.title("타이타닉호의 각 클래스별, 승객 수")
plt.show()
```

```
iris = sns.load_dataset("iris")
sns.pairplot(iris)
plt.title("Iris Data의 Pair Plot")
plt.show()
```

```
tips = sns.load_dataset('tips')
sns.boxplot(x="day", y="total_bill", data=tips)
plt.title("요일 별 전체 팁의 Box Plot")
plt.show()
```

# hue='sex'

### 3. Matplotlib

#### ❖ 다른 시각화 패키지 - Folium

- 지도위에 위치정보를 시각화, 웹에서 사용 가능

```
import folium
```

```
map = folium.Map(location=[37.559868, 126.967109], zoom_start=14)
```

```
folium.Marker(
```

```
 location=[37.559868, 126.967109], popup='한국경제신문사'
```

```
).add_to(map)
```

```
folium.Marker(
```

```
 location=[37.566317, 126.977829],
```

```
 tooltip='서울특별시청',
```

```
 icon=folium.Icon(color="red", icon="info-sign")
```

```
).add_to(map)
```

```
folium.Circle(
```

```
 location=[37.56412, 126.998009],
```

```
 radius=100, tooltip='중구청',
```

```
 color='crimson', fill=False
```

```
).add_to(map)
```

```
title_html = '<h3 align="center" style="font-size:20px">3개 지역 표시</h3>'
```

```
map.get_root().html.add_child(folium.Element(title_html))
```

```
map
```

---

## 목 차

1. 과학 계산 패키지 (Numpy)
2. 데이터 분석 패키지 (Pandas)
3. 시각화 패키지 (Matplotlib)
4. **데이터 분석 사례**

## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 필요 데이터

- 서울 열린데이터 광장(<https://data.seoul.go.kr/index.do>)
  - 자치구 년도별 CCTV 설치 현황
  - 서울통계서비스 > 주민등록인구(구별) 통계

### ❖ CCTV 데이터 정리하기

- 데이터 파일 읽기

```
cctv = pd.read_csv(filename, skiprows=1, encoding='euc-kr')
```

```
구분 총계 2012년 이전 2012년 2013년 2014년 2015년 2016년 2017년 2018년 2019년 2020년 2021년
```

- Null data 있는지 확인하기

```
cctv.isnull().sum().sum()
```

- 1000 단위 구분기호 없애고 정수로 변환하기

```
for column in cctv.columns[1:]:
```

```
 cctv[column] = cctv[column].apply(lambda x: int(x.replace(',',''))))
```

## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ CCTV 데이터 정리하기

- 2021년을 제외한 최근 3개년간의 CCTV 증가율 구하기

```
cctv['2017년_이전'] = cctv.총계 - cctv['2018년'] - cctv['2019년'] \
 - cctv['2020년'] - cctv['2021년']
```

```
cctv['최근증가율'] = ((cctv['2018년']+cctv['2019년']+cctv['2020년'])) / \
 cctv['2017년_이전'] * 100).round(2)
```

- 구분, 총계, 최근증가율 컬럼만 추출하고 컬럼명을 구별, CCTV댓수로 변경하기

```
cctv = cctv[['구분', '총계', '최근증가율']]
```

```
cctv.rename(columns={'구분':'구별', '총계':'CCTV댓수'}, inplace=True)
```

- 구 이름에서 공백지우기

```
cctv['구별'] = cctv.구별.apply(lambda x: x.replace(' ',''))
```

- 첫번째 행 지우기

```
cctv.drop([0], inplace=True)
```

## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 인구 데이터 정리하기

- 데이터 파일 읽기

```
pop = pd.read_csv(filename, sep='\t', skiprows=2)
```

기간 자치구 세대 계 남자 여자 계.1 남자.1 여자.1 계.2 남자.2 여자.2 세대당인구 65세이상고령자

- Null data 있는지 확인하기

```
pop.isnull().sum().sum()
```

- '자치구','계','계.1','계.2','65세이상고령자' 컬럼만 추출하고 이름을 변경

```
pop = pop[['자치구','계','계.1','계.2','65세이상고령자']]
```

```
pop.columns = ['구별','인구수','내국인','외국인','고령자']
```

- 1000 단위 구분기호 없애고 정수로 변환하기

```
for column in pop.columns[1:]:
```

```
 pop[column] = pop[column].apply(lambda x: int(x.replace(',','')))
```

- 첫번째 행 지우기

```
pop.drop([0], inplace=True)
```

## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 인구 데이터 정리하기

- '외국인비율'과 '고령자비율' 컬럼 만들기

```
pop['외국인비율'] = (pop.외국인/pop.인구수*100).round(2)
```

```
pop['고령자비율'] = np.round(pop.고령자/pop.인구수*100, 2)
```

```
구별 인구수 내국인 외국인 고령자 외국인비율 고령자비율
```

### ❖ CCTV 데이터와 인구 데이터 합치기

- 데이터 합치기

```
df = pd.merge(cctv, pop)
```

- '구별' 컬럼을 인덱스로 만들기

```
df.set_index('구별', inplace=True)
```

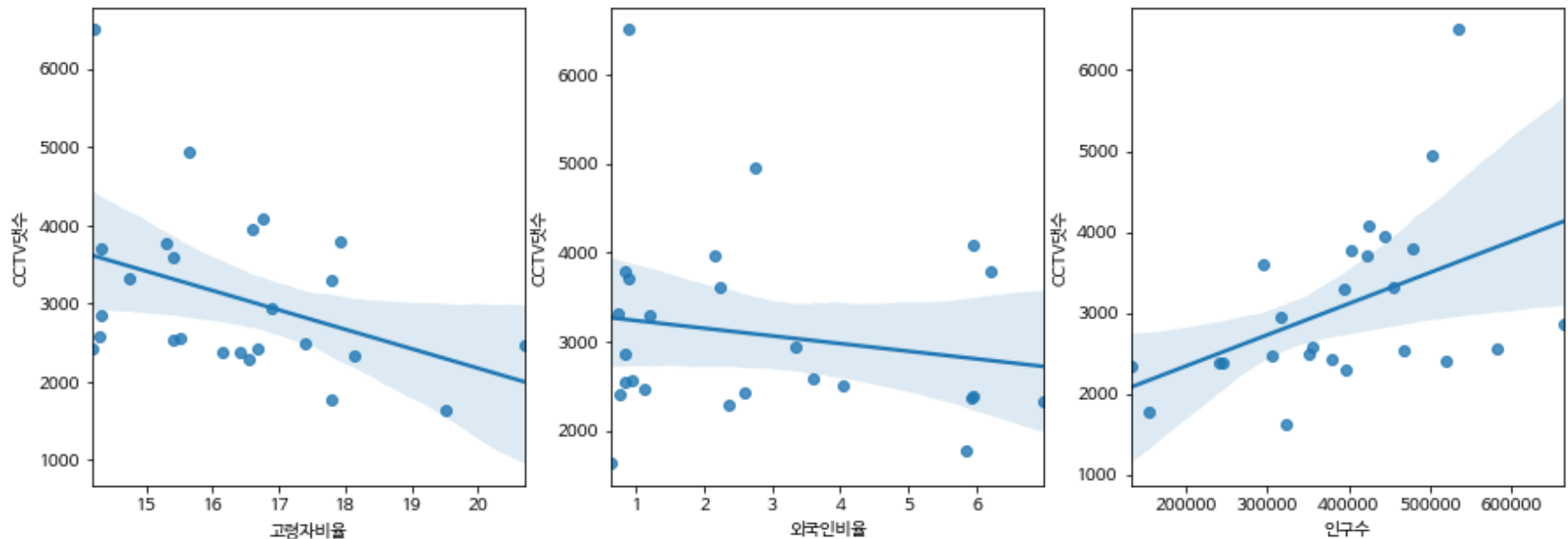


## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 상관관계 분석하기

- '고령자비율', '외국인비율', '인구수' vs 'CCTV댓수'

```
fig, axs = plt.subplots(figsize=(15,5), ncols=3, nrows=1)
features = ['고령자비율', '외국인비율', '인구수']
for i, feature in enumerate(features):
 sns.regplot(x=feature, y='CCTV댓수', data=df, ax=axs[i])
```



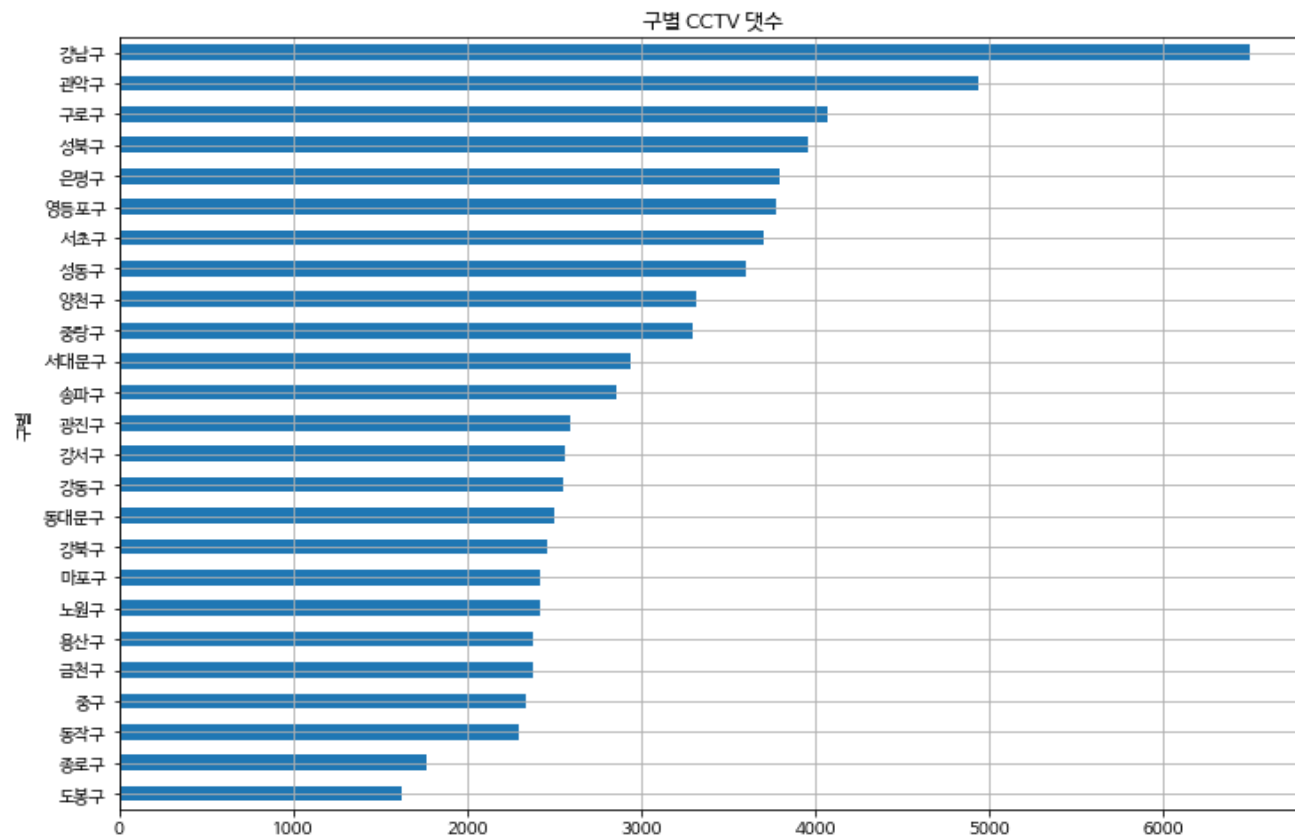
## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 그래프 분석

#### ▪ 구별 CCTV 댓수

```
df.CCTV댓수.sort_values().plot(kind='barh', grid=True, figsize=(12,8),
 title='구별 CCTV 댓수')

plt.show()
```

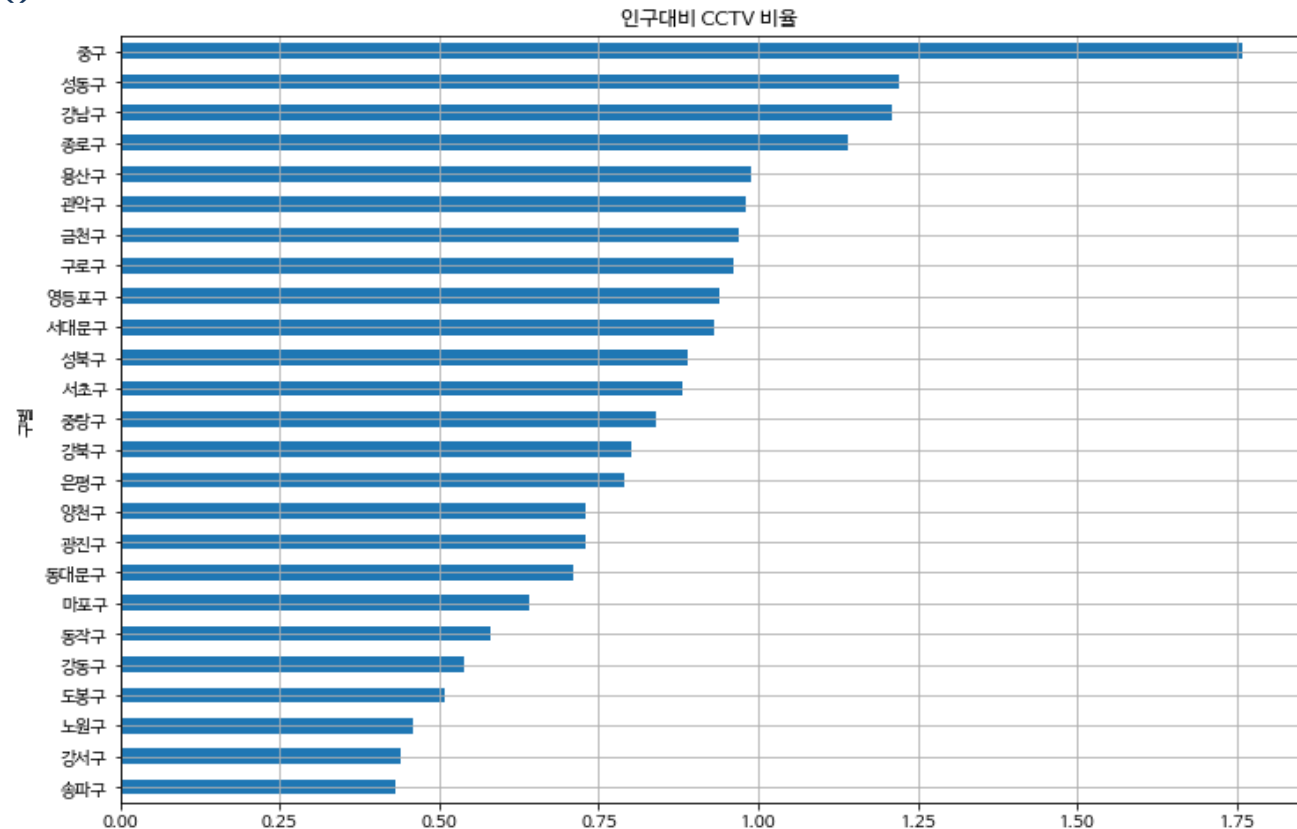


## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 그래프 분석

#### ▪ 인구대비 CCTV 비율

```
df['cctv비율'] = (df.CCTV갯수 / df.인구수 * 100).round(2)
df.cctv비율.sort_values().plot(kind='barh', grid=True, figsize=(12,8))
plt.title('인구대비 CCTV 비율')
plt.show()
```



## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 인구수와 CCTV 댓수 분석

- 산점도

```
plt.scatter(df.인구수, df.CCTV댓수, s=50)
plt.grid(True)
plt.title('인구수와 CCTV 댓수의 산점도')
plt.xlabel('인구수')
plt.ylabel('CCTV')
plt.show()
```

- 산점도 위에 상관관계를 나타내는 선

```
fp1 = np.polyfit(df.인구수, df.CCTV댓수, 1) # 1은 함수의 차수
fx = np.array([100000, 700000])
f1 = np.poly1d(fp1) # 1차원의 회귀식
fy = f1(fx)
```

- 오차를 계산하고, 오차의 내림차순으로 소팅된 데이터프레임 만들기

```
df['오차'] = np.abs(df.CCTV댓수 - f1(df.인구수)).round(2)
df_sort = df.sort_values('오차', ascending=False)
```

## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 인구수와 CCTV 댓수 분석

- 최종

```
plt.figure(figsize=(14,10))
plt.scatter(df.인구수, df.CCTV댓수, c=df.오차, s=50)
plt.plot(fx, fy, ls='dashed', lw=3, color='g')

for i in range(10):
 plt.text(df_sort.인구수[i]+9000, df_sort.CCTV댓수[i]-50,
 df_sort.index[i], fontsize=15)

plt.grid(True)
plt.title('인구수와 CCTV 댓수의 관계', fontsize=20)
plt.xlabel('인구수')
plt.ylabel('CCTV')
plt.colorbar()
plt.show()
```

## 4. 데이터 분석 사례 - 서울시 구별 CCTV 현황 분석

### ❖ 인구수와 CCTV 댓수 분석

- 최종

