



SESSION # 02

데이터 시각화 & 10장 데이터 다루기'

정재근, 빈다은, 고병욱

Date _ 2018.04.07

CONTENTS

3. 데이터 시각화

3.1 상황에 맞는 시각화

3.2 파이썬 시각화 기본

3.3 Tableau

10. 데이터 다루기

10.1 1차원 데이터 분석

10.2 2차원 데이터 분석

10.3 척도 조절

10.4 차원 축소

3.0 Intro

데이터 시각화?

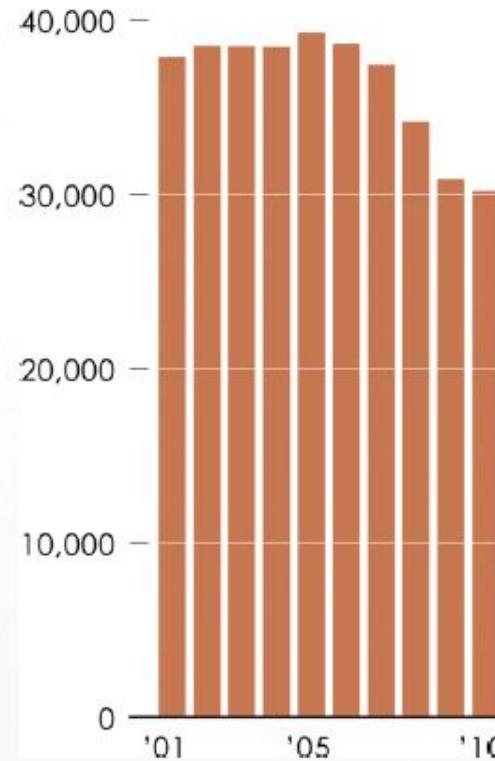
해마다 교통사고가 발생하지만
2006년부터는 감소한다.

3.0 Intro

데이터 시각화?

해마다 교통사고가 발생하지만
2006년부터는 감소한다.

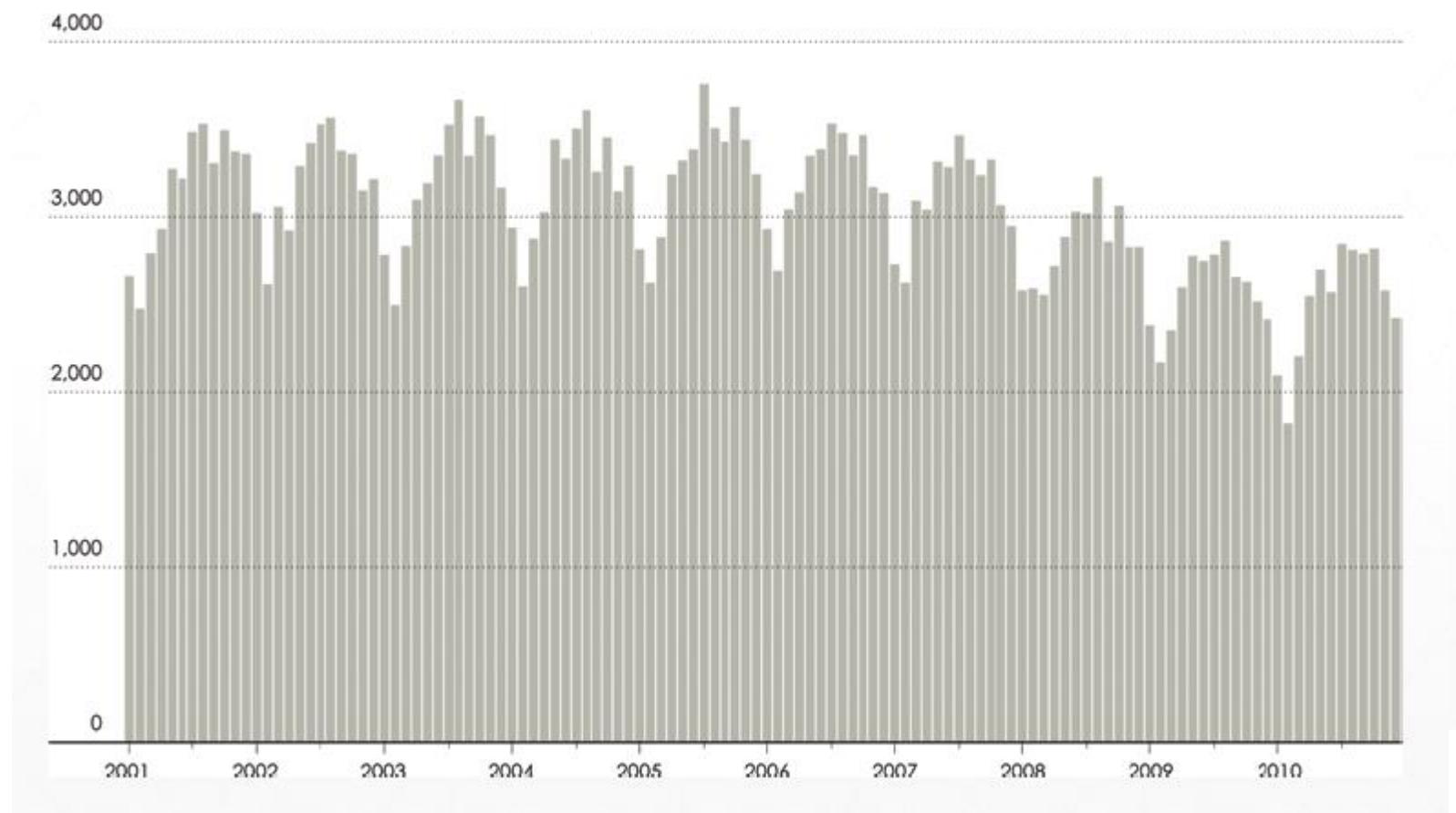
Annual fatal crashes



3.0 Intro

데이터 시각화?

Monthly fatal crashes

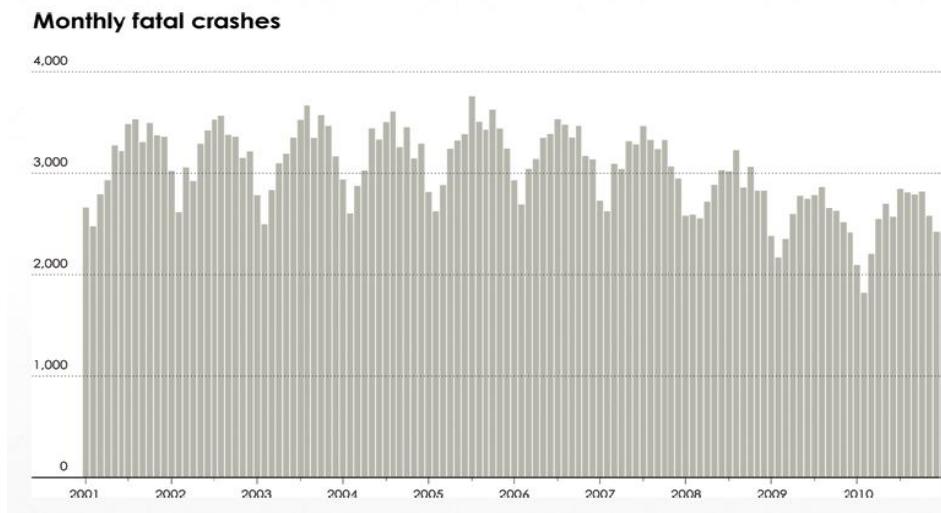


3.0 Intro

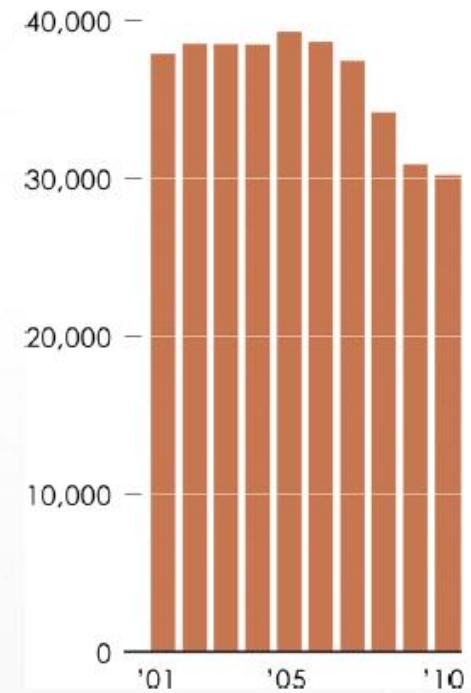
데이터 시각화?

해마다 교통사고가 발생하지만
2006년부터는 감소한다.

교통사고는 7,8월에 가장 많고
2월에 가장 적다.

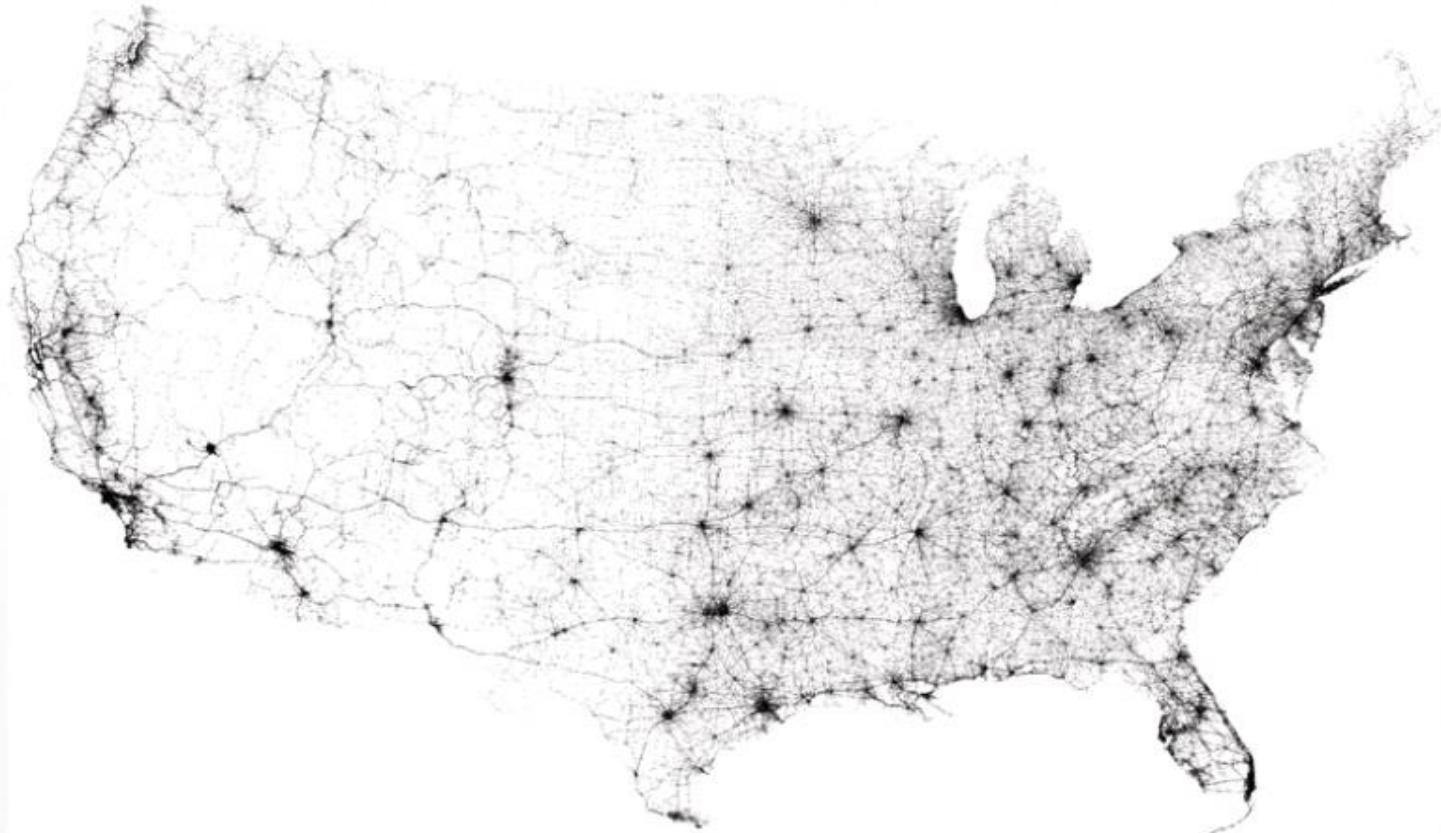


Annual fatal crashes



3.0 Intro

데이터 시각화?



지도로 표현한 미국 교통사고

3.0 Intro 목적

■ 데이터 시각화

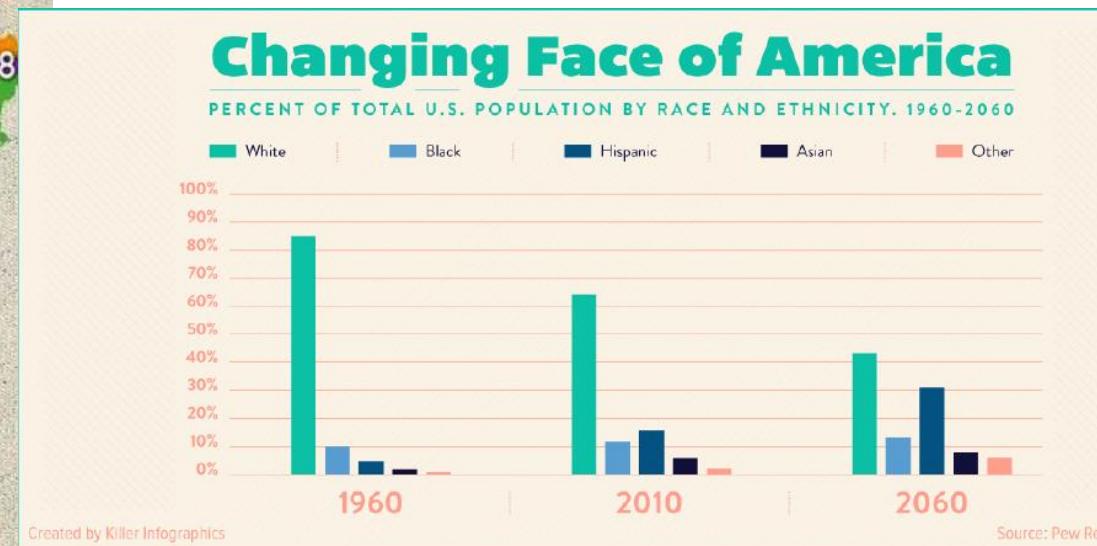
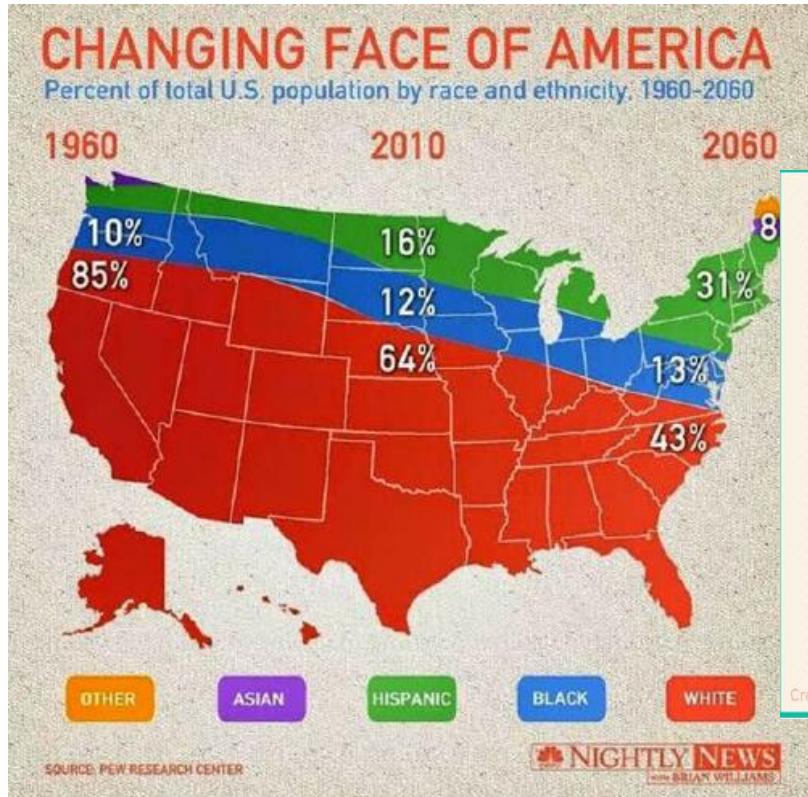
설득과 사실확인을 목적으로 한다.

맥락과 상황에 맞는 사실확인을 하자.

맥락과 상황에 맞는 데이터를 제공하되 과도한 정보를 제공해서는 안된다.

3.0 Intro 목적

데이터 시각화



3.1 상황에 맞는 시각화

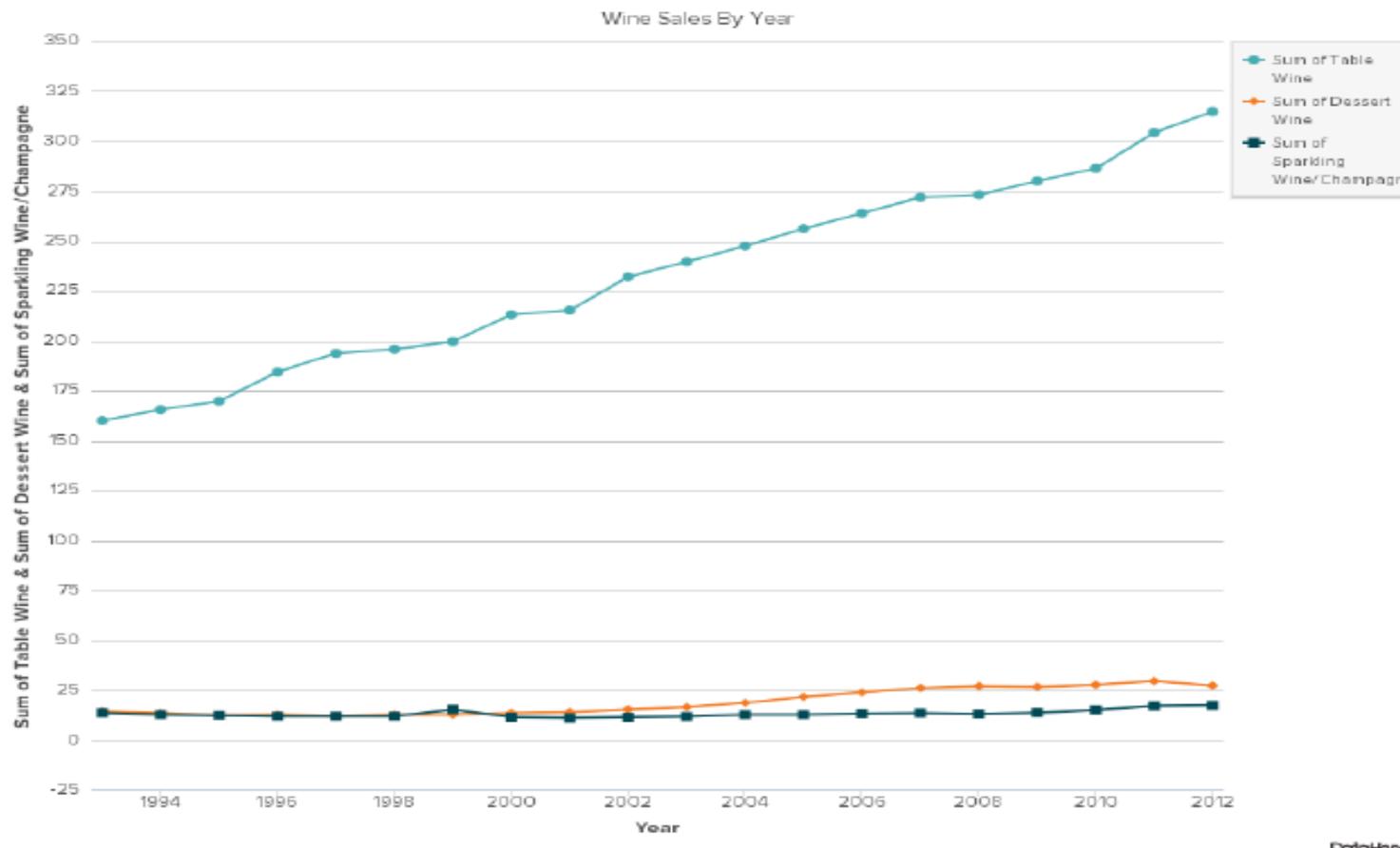
■ 와인 종류에 따른 연별 대한 판매량

선? 막대? 히스토그램? 산점도????

직접 해보는 것, 경험이 중요!

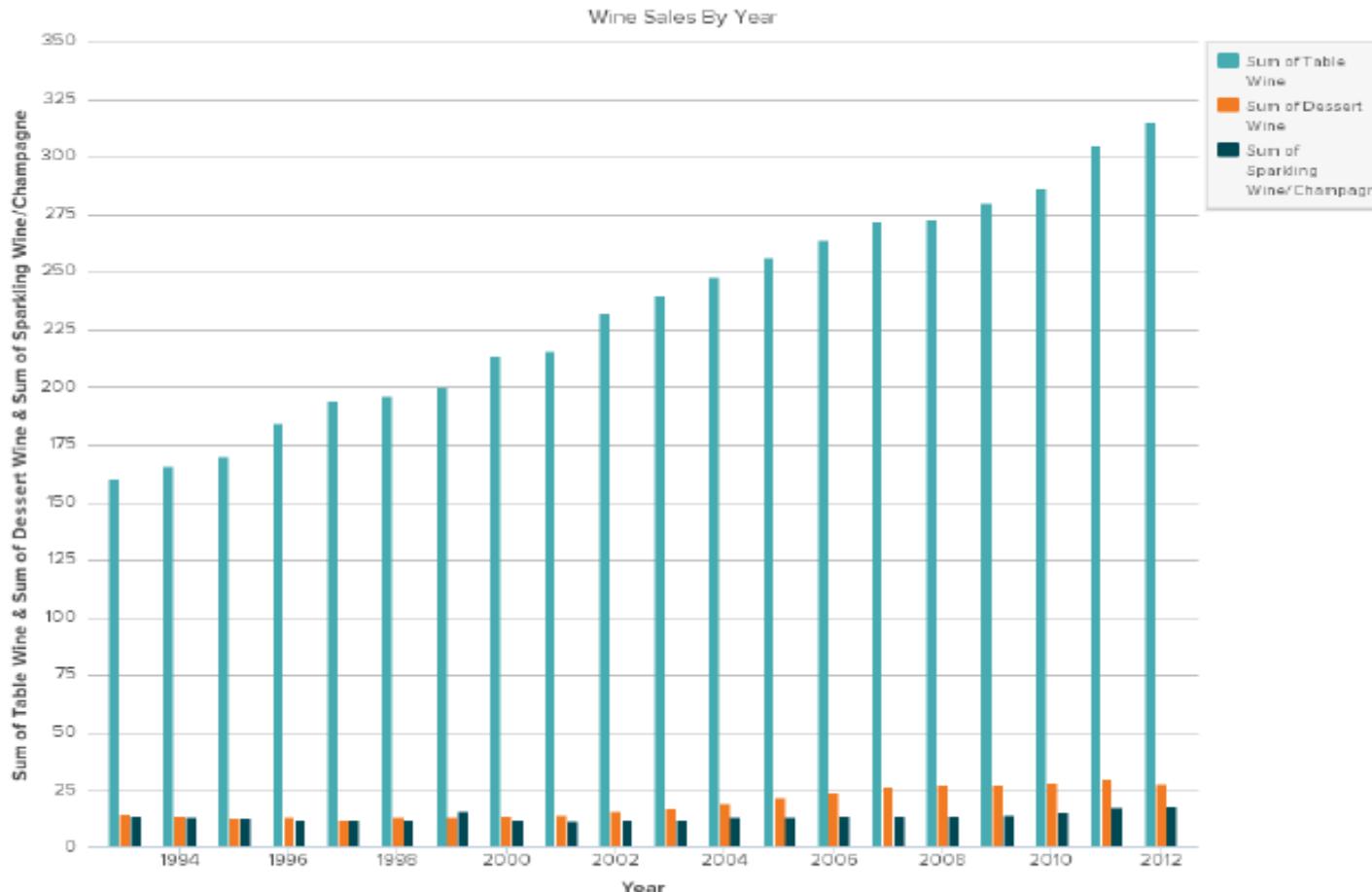
3.1 상황에 맞는 시각화

■ 와인 종류에 따른 연별 대한 판매량



3.1 상황에 맞는 시각화

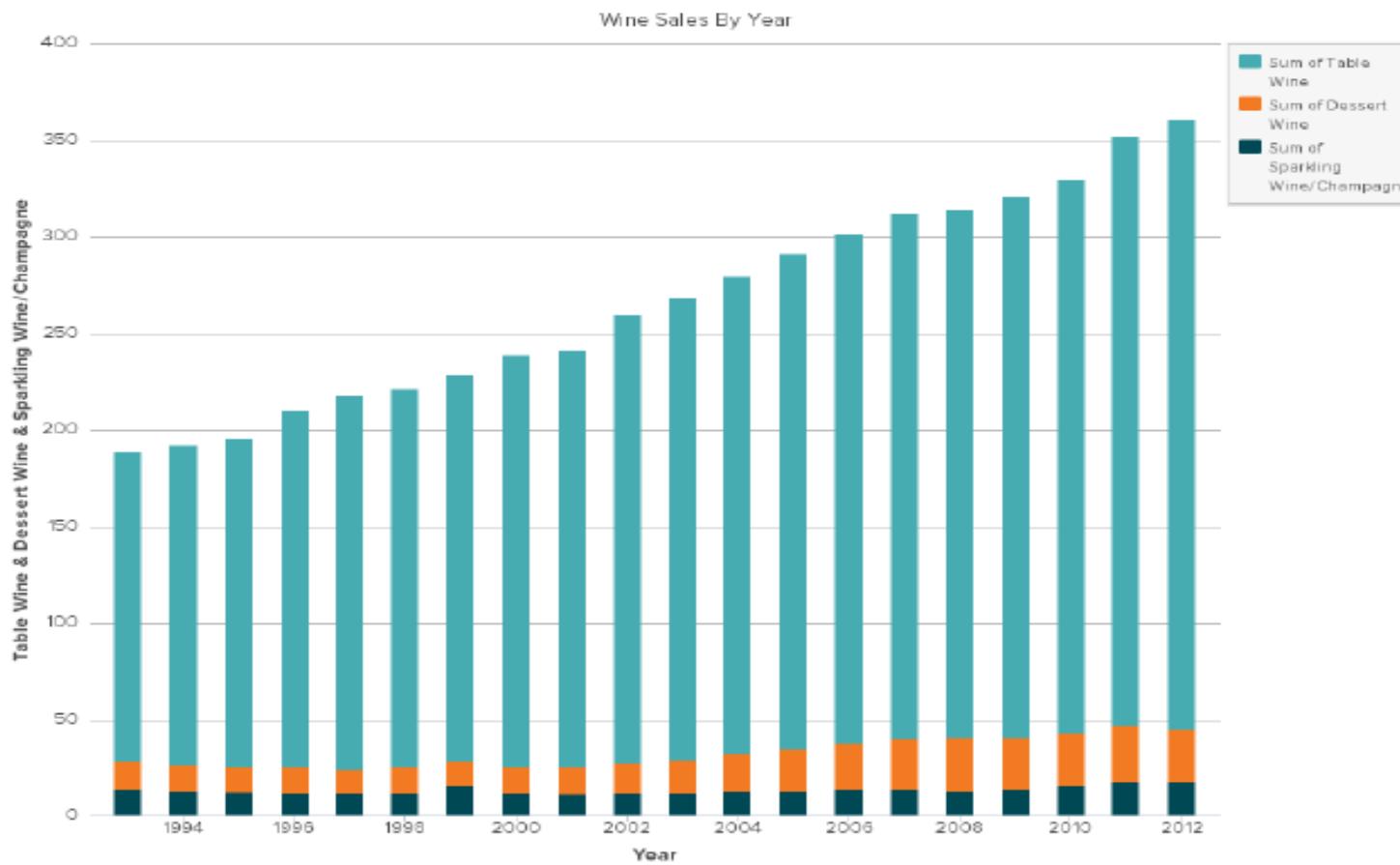
■ 와인 종류에 따른 연별 대한 판매량(grouped)



DataHero

3.1 상황에 맞는 시각화

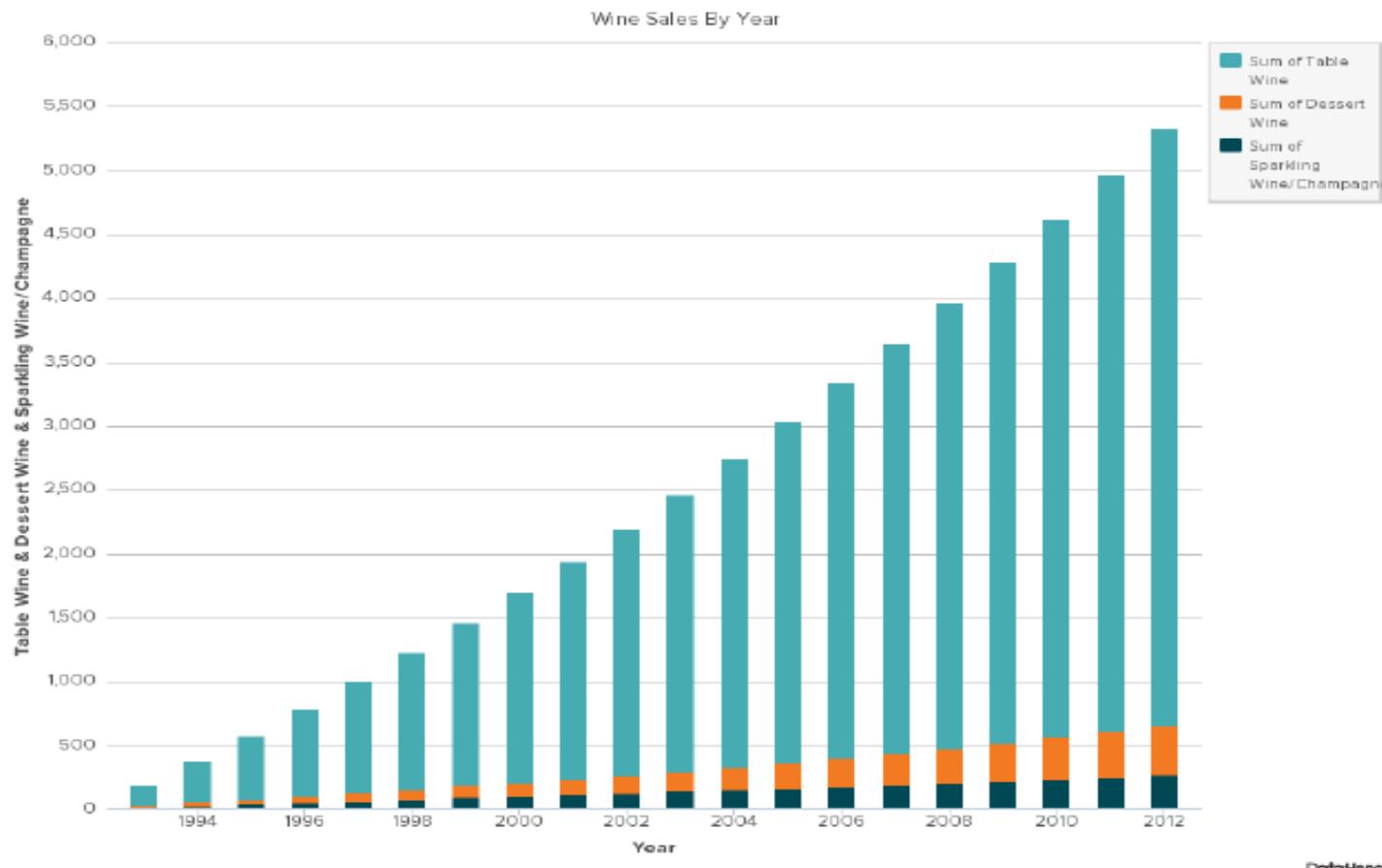
■ 와인 종류에 따른 연별 대한 판매량(stacked)



DataHero

3.1 상황에 맞는 시각화

■ 와인 종류에 따른 연별 대한 판매량(cumulative)



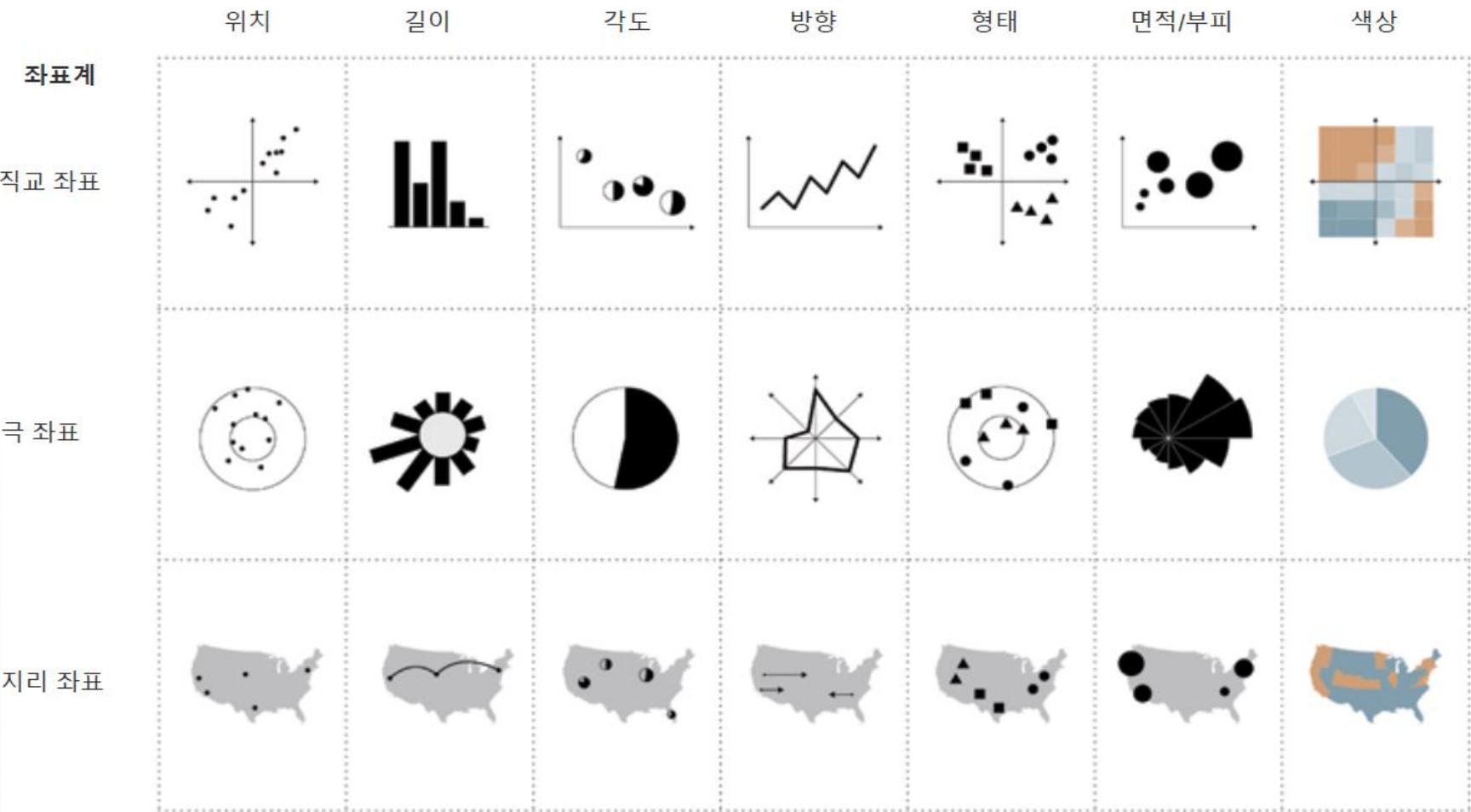
3.1 상황에 맞는 시각화

■ 시각화의 종류

정보 시각화 방법				
시간 시각화	분포 시각화	관계 시각화	비교 시각화	공간 시각화
막대그래프 (Bar graph) 누적 막대그래프 (Stacked Bar graph) 점그래프 (Point graph)	파이차트 (Pie chart) 도넛 차트 (Donut chart) 트리맵 (Tree map) 누적 연속 그래프 (Cumulative continuous graph)	스캐터 플롯 (Scatter plot) 버블 차트 (Bubble chart) 히스토그램 (Histogram)	히트맵 (heat map) 체르노프 페이스 (Chernoff face) 별그래프 (Star graph) 평행 좌표계 (Parallel coordinate system) 다차원 척도법 (Multi-dimensional scaling)	지도 매핑 (Dataviz on map)

3.1 상황에 맞는 시각화

시각적 단서들

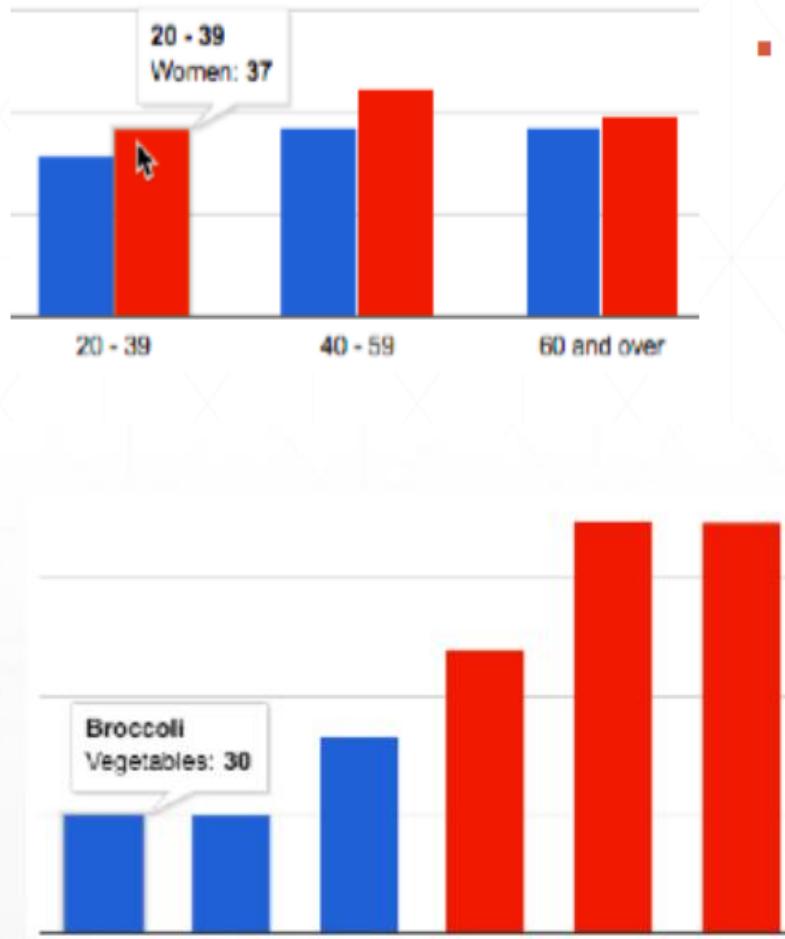


3.1 상황에 맞는 시각화

경험? 공식?

1. 다양한 카테고리가 시간이 변화함에 따라 변하는 1차원 데이터를 갖고 있을 때
2. 특정 카테고리에서 시간별로 변동량을 확인하고 싶을 때
3. 세 카테고리를 합친 전체 판매량의 변화를 보고 싶을 때
4. 전체에서 카테고리별로 비중의 변화를 확인하고 싶을 때
5. 예전부터 현재까지 세 카테고리를 합친 누적 판매량이 얼마나 되는지 확인하고 싶을 때

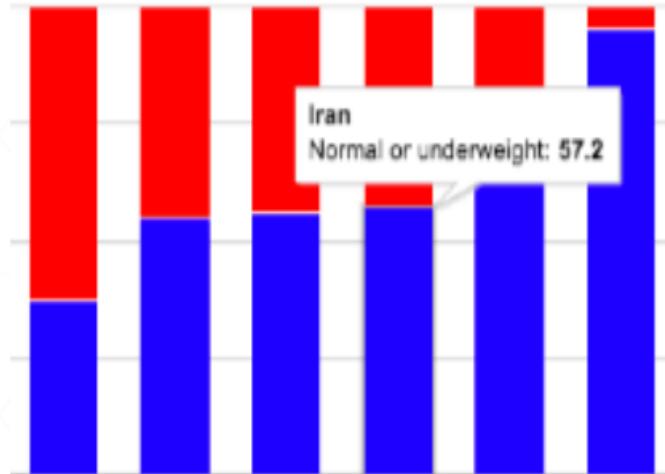
3.1 상황에 맞는 시각화



- Grouped column or bar
 - Best to compare categories side-by-side. Vertical columns, or horizontal bars for long labels.

- Separated column or bar
 - Best to compare categories in separate clusters.

3.1 상황에 맞는 시각화

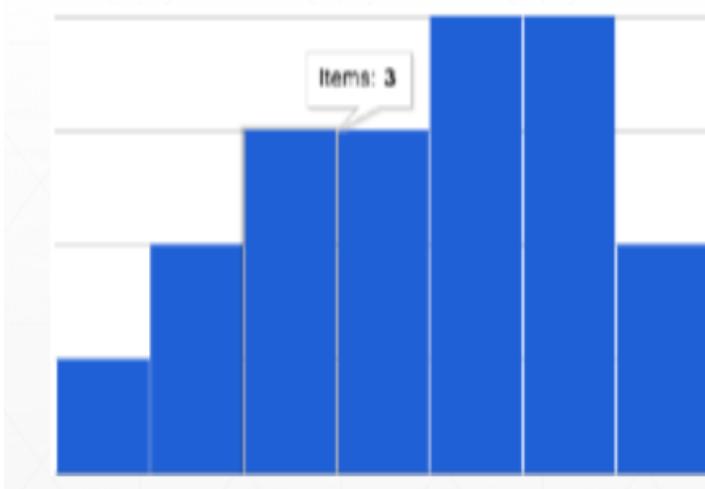


- Stacked column or bar

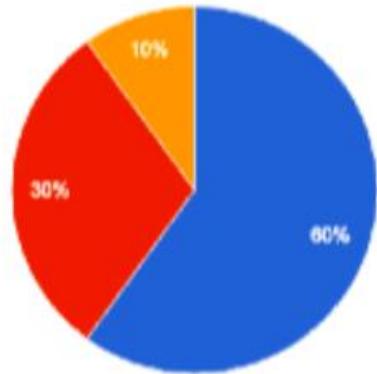
- Best to compare sub-categories, or parts of a whole. Vertical columns, or horizontal bars for long labels.

- Histogram

- Best to show distribution of raw data, with number of values in each bucket.

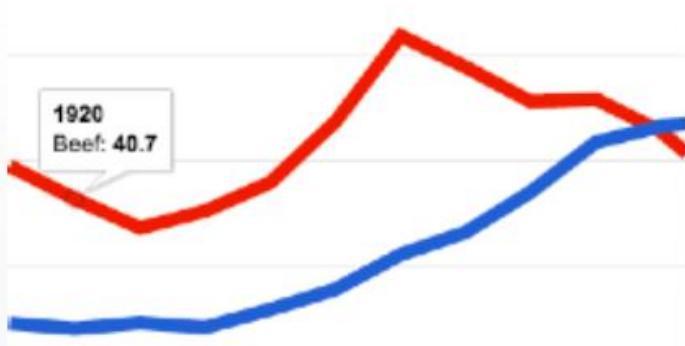


3.1 상황에 맞는 시각화



- Pie chart

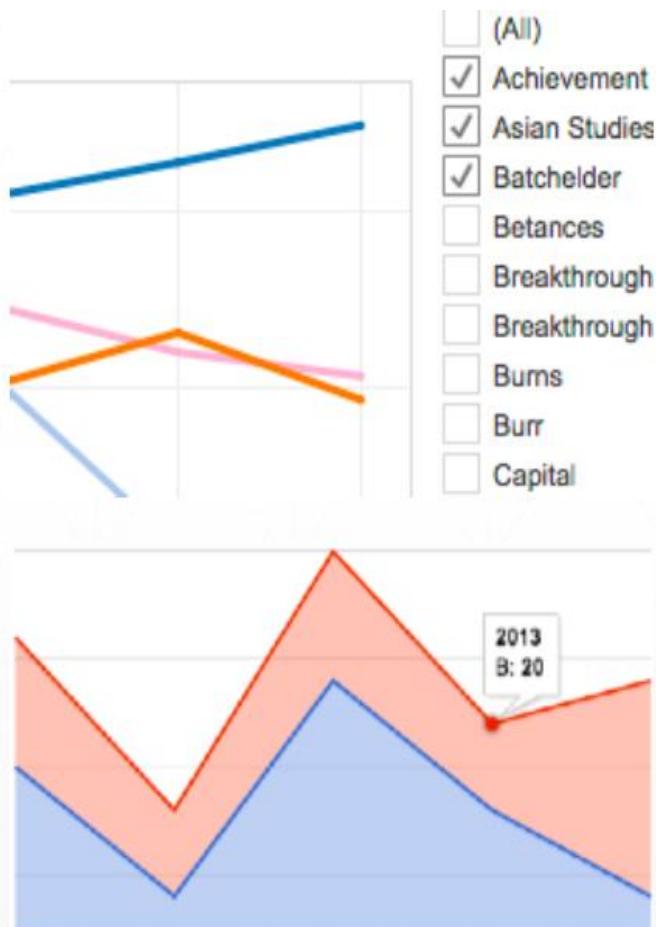
- Best to show parts of a whole, but hard to estimate size of slices.



- Line chart

- Best to show continuous data, such as change over time.

3.1 상황에 맞는 시각화



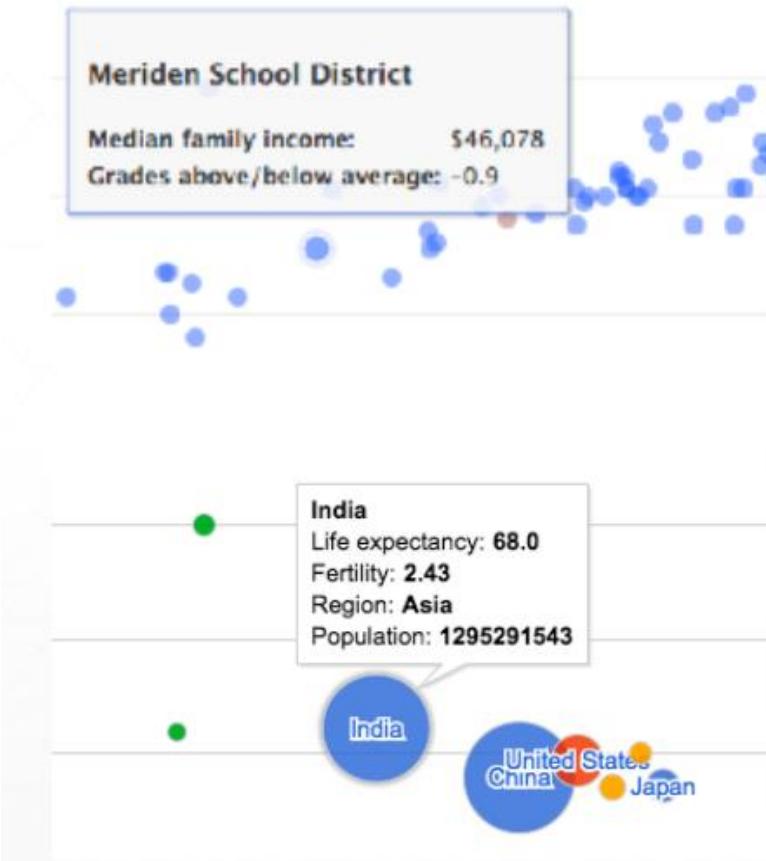
- Filtered line chart

- Best to show multiple lines of continuous data, with on-off toggle buttons.

- Stacked area chart

- Best to show parts of a whole, with change over time.

3.1 상황에 맞는 시각화



▪ Scatter chart

- Best to show relationship between two sets of data. Also called an XY chart.

▪ Bubble chart

- Best to show relationship between three or four sets of data, using bubble size and color.

3.1 상황에 맞는 시각화

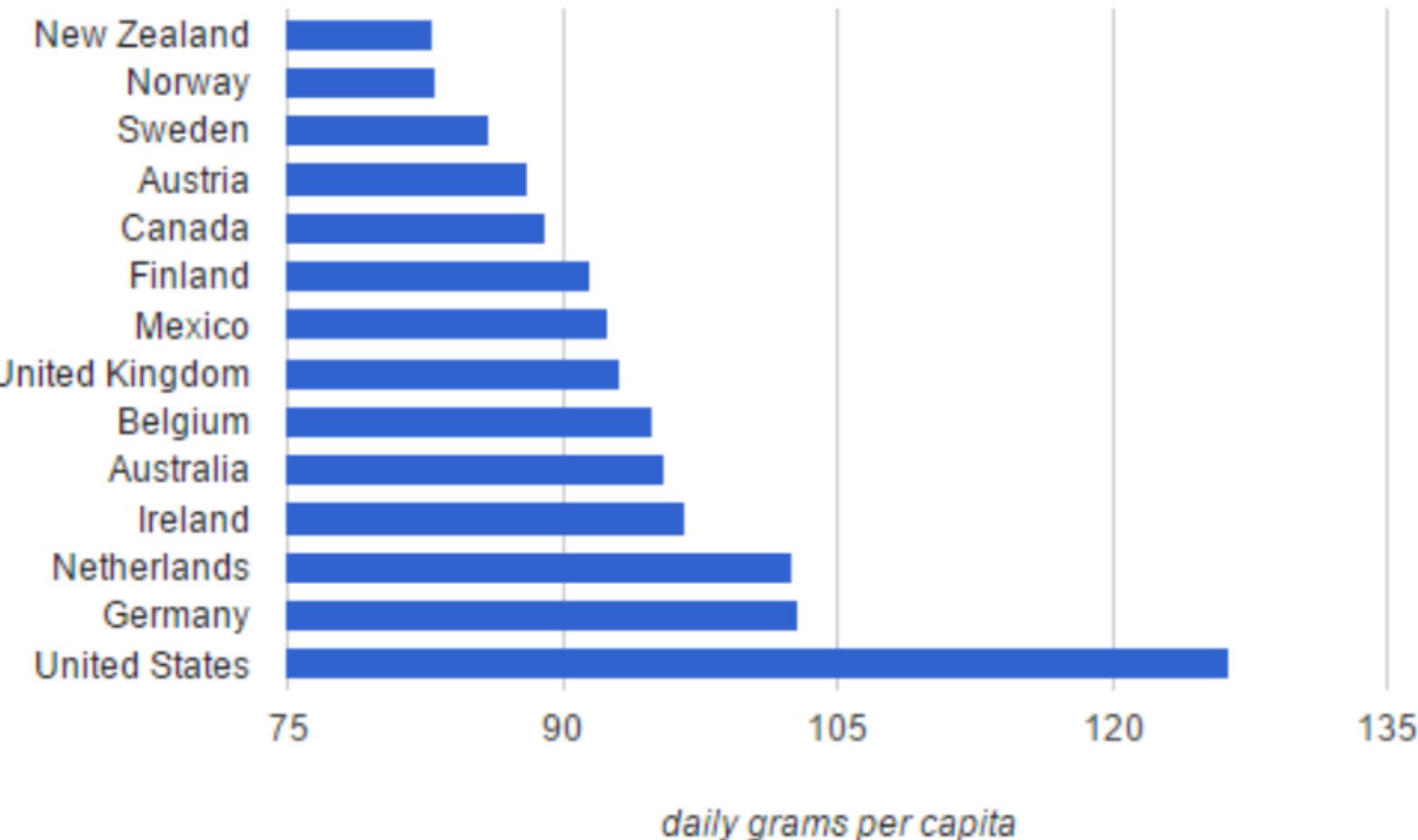
Quiz

	A	B
1	nation	sugar daily grams per capita in 2015
2	New Zealand	82.9
3	Norway	83.1
4	Sweden	86.1
5	Austria	88.1
6	Canada	89.1
7	Finland	91.5
8	Mexico	92.5
9	United Kingdom	93.2
10	Belgium	95
11	Australia	95.6
12	Ireland	96.7
13	Netherlands	102.5
14	Germany	102.9
15	United States	126.4

- 이 데이터를 표현하기에 가장 적합한 차트는?
 - Column chart
 - Pie chart
 - Line chart
 - Stacked area chart
 - Bar chart
 - Scatter chart
 - Bubble chart

3.1 상황에 맞는 시각화

Sugar consumption for selected nations in 2015



3.1 상황에 맞는 시각화

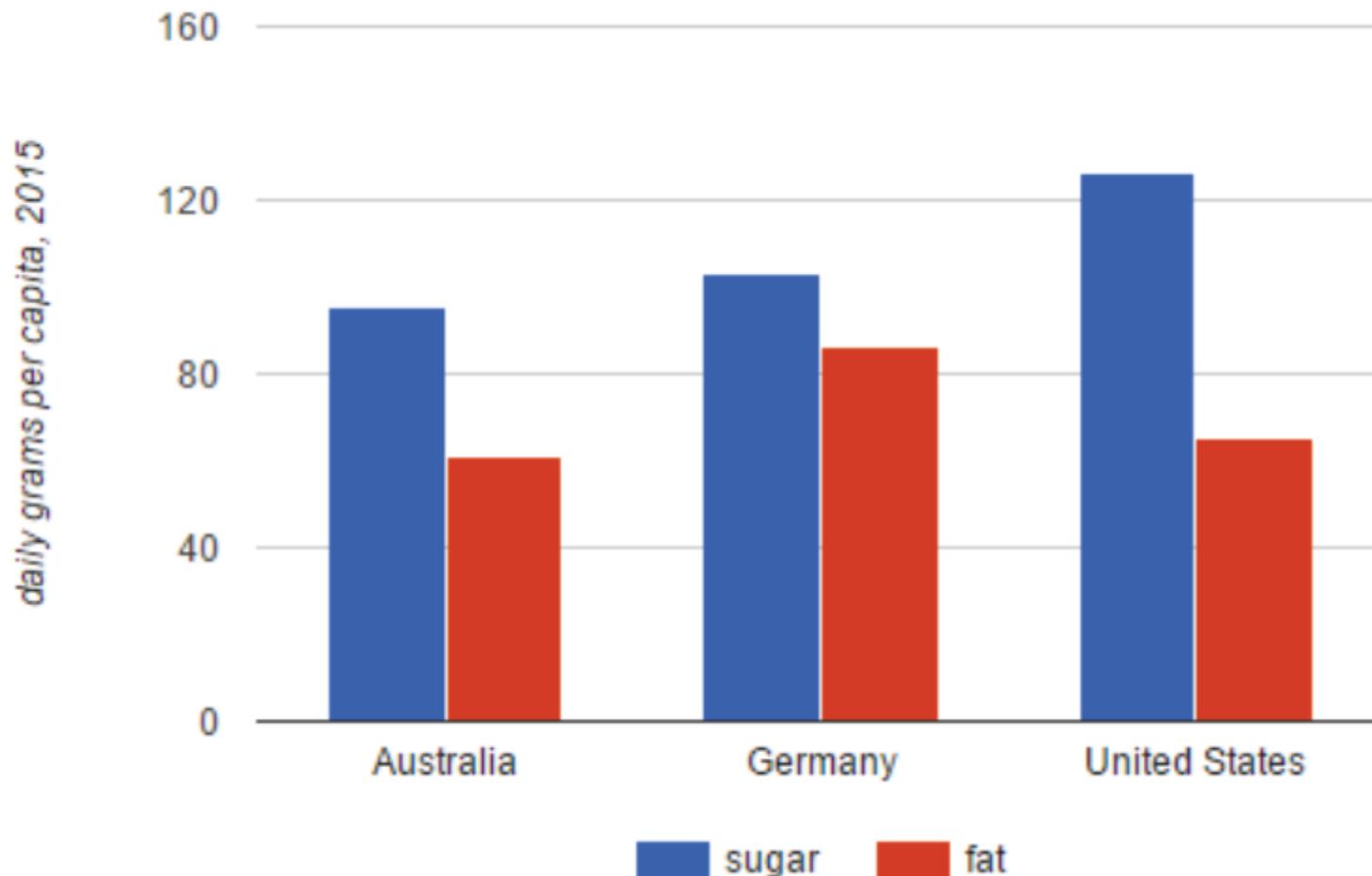
Quiz

	A	B	C
1	nation	sugar daily grams per capita in 2015	fat daily grams per capita in 2015
2	Australia	95.6	60.9
3	Germany	102.9	86.5
4	United States	126.4	65.5

- 하루 평균 섭취하는 설탕과 지방을 비교하는 차트로 적당한 것은?
 - Column or bar chart
 - Grouped column chart
 - Pie chart
 - Line chart
 - Scatter chart

3.1 상황에 맞는 시각화

Sugar vs. fat consumption in three nations, 2015



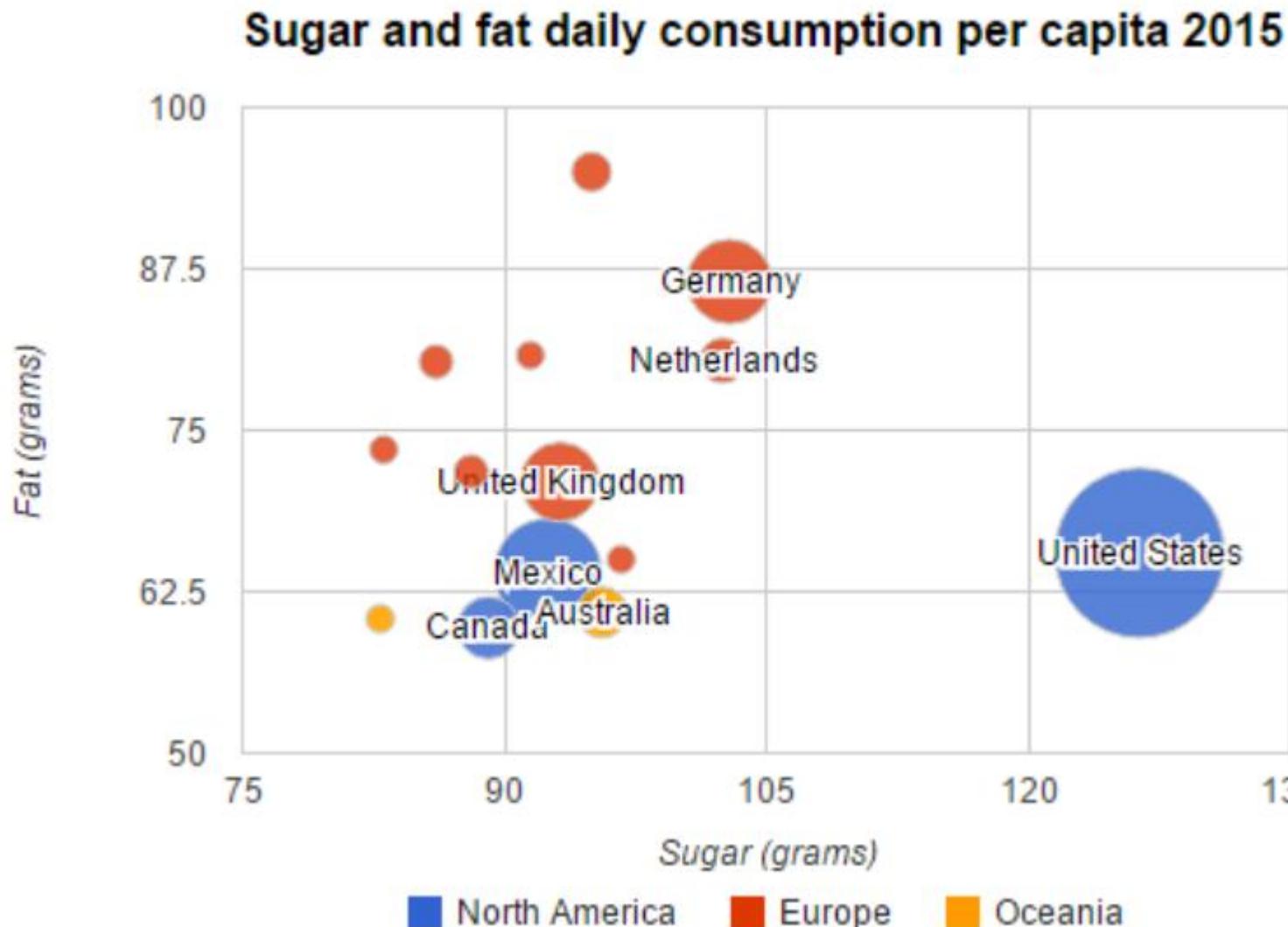
- Scatter chart

3.1 상황에 맞는 시각화

Quiz

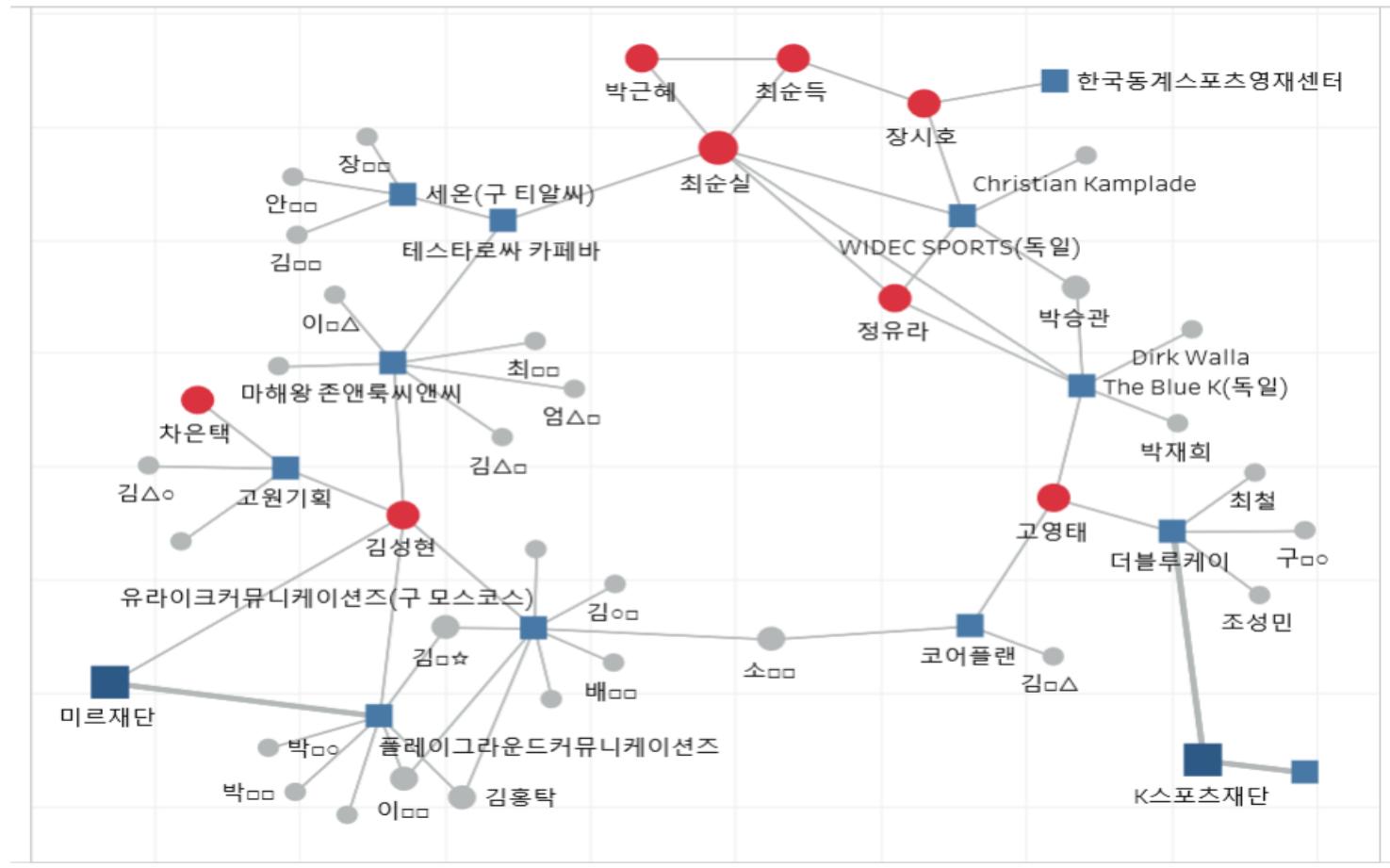
	A	B	C	D	E
1	nation	sugar (daily grams per capita, 2015)	fat (daily grams per capita, 2015)	continent	population (2015)
2	United States	126.4	65.5	North America	321,419,000
3	Germany	102.9	86.5	Europe	81,413,000
4	Netherlands	102.5	80.4	Europe	16,937,000
5	Ireland	96.7	65	Europe	4,641,000
6	Australia	95.6	60.9	Oceania	23,781,000
7	Belgium	95	95	Europe	11,286,000
8	United Kingdom	93.2	71	Europe	65,138,000
9	Mexico	92.5	64	North America	127,017,000
10	Finland	91.5	80.8	Europe	5,482,000
11	Canada	89.1	59.7	North America	35,852,000
12	Austria	88.1	71.8	Europe	8,611,000
13	Sweden	86.1	80.3	Europe	9,799,000
14	Norway	83.1	73.5	Europe	5,196,000
15	New Zealand	82.9	60.4	Oceania	4,596,000

3.1 상황에 맞는 시각화



3.1상황에 맞는 시각화

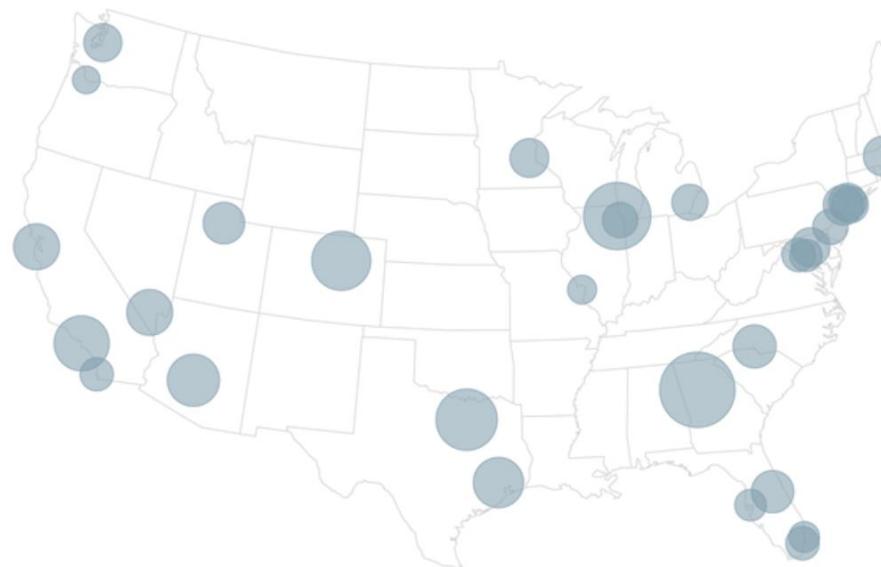
그렇다면?



<http://newstapa.org/35607>

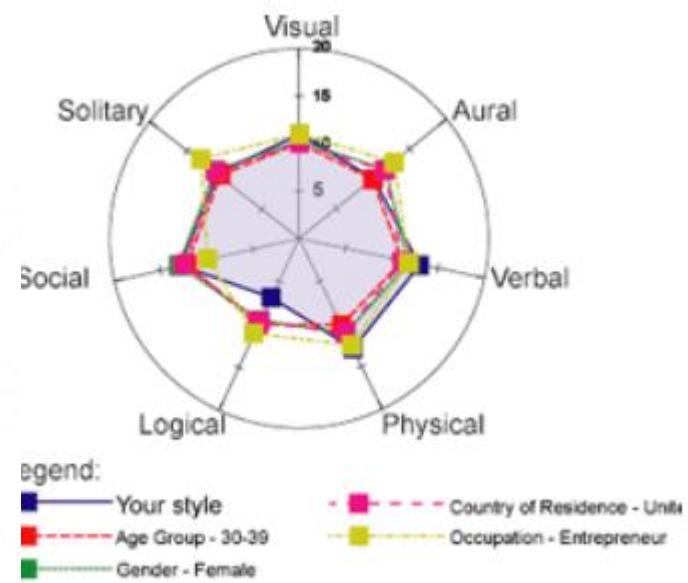
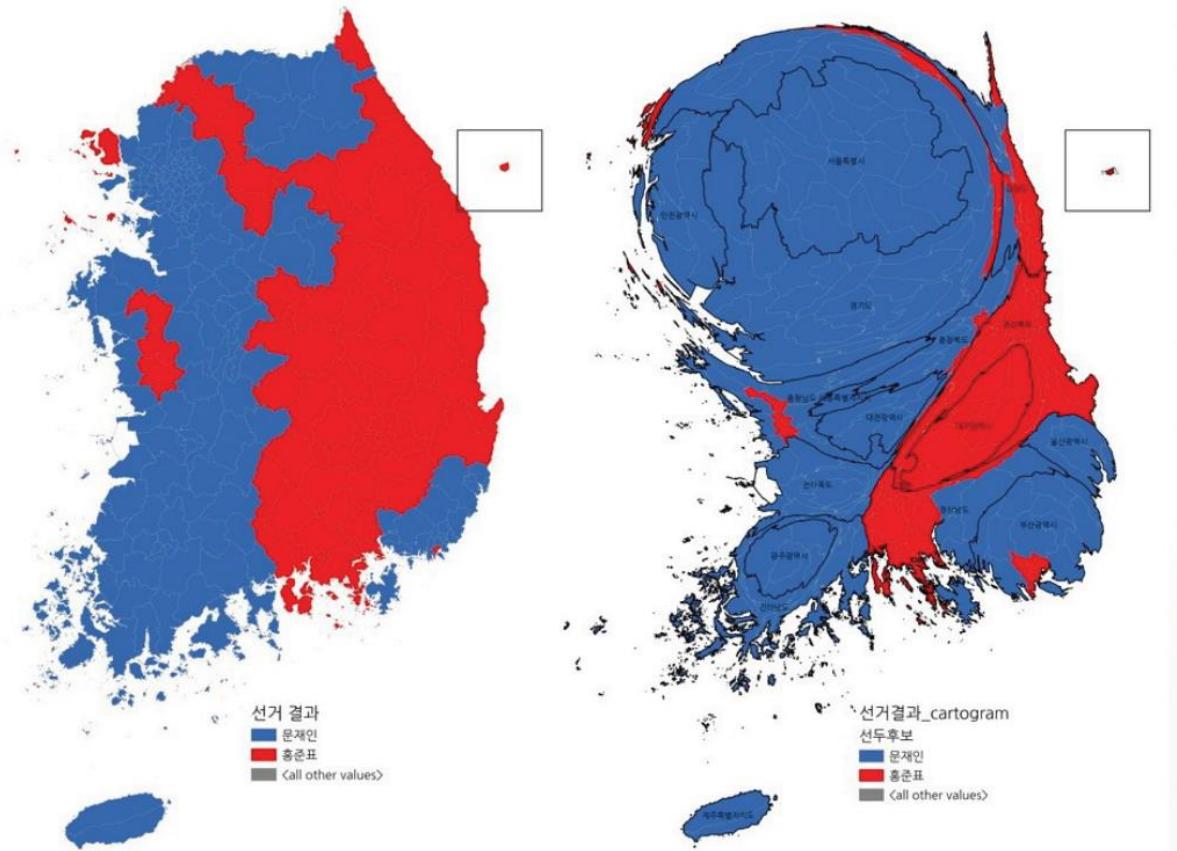
3.1 상황에 맞는 시각화

지도 기반의 시각화



3.1 상황에 맞는 시각화

지도 기반 주의사항



3.2 파이썬 - 시각화 기본

■ Matplotlib?

파이썬에서 자료를 차트나 플롯으로 시각화 하는 패키지

웹을 위한 복잡하고 인터랙티브한 시각화에 적절하진 않음

저수준의 api를 사용한 다양한 시각화 기능도 제공

3.2 파이썬 - 시각화 기본

■ Matplotlib

`from matplotlib import pyplot as plt`
(항상 상단에 입력!)

`plt.plot`
`plt.bar`

관련 링크

: http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot

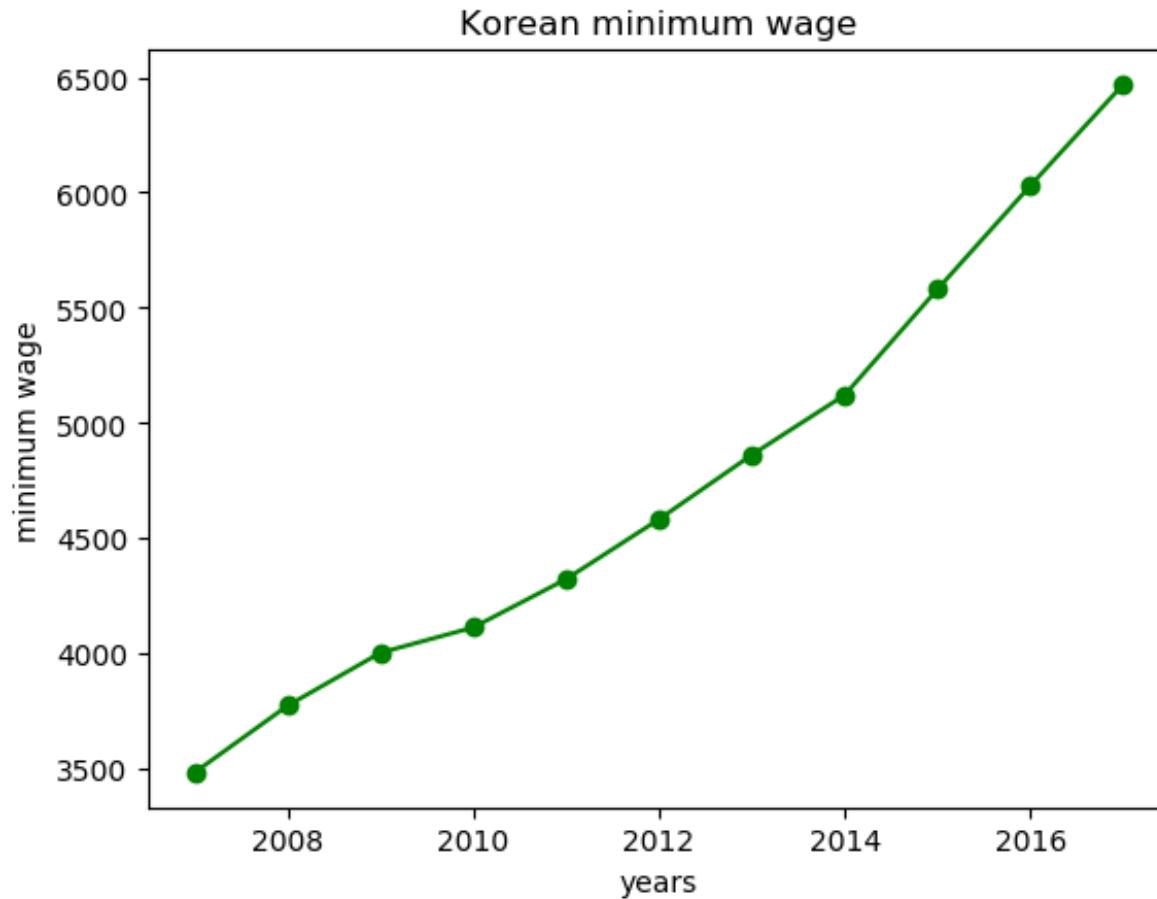
3.2 파이썬 - 시각화 기본

1. Line charts

```
from matplotlib import pyplot as plt  
  
years = [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]  
wage = [3480, 3770, 4000, 4110, 4320, 4580, 4860, 5120, 5580, 6030, 6470]  
  
#x축에 연도 y축에 최저임금이 되는 그래프를 만들자.  
plt.plot(years, wage, color='green', marker='o', linestyle='solid')  
  
#제목과 라벨을 달아보자.  
plt.title("Korean minimum wage")  
plt.xlabel("years")  
plt.ylabel("minimum wage")  
  
plt.show()
```

3.2 파이썬 - 시각화 기본

```
from matplotlib import  
years = [2007, 2008, 2  
wage = [3480, 3770, 40  
  
#x축에 연도 y축에 최  
plt.plot(years, wage  
  
#제목과 라벨을 달아보  
plt.title("Korean mi  
plt.xlabel("years")  
plt.ylabel("minimum  
  
plt.show()
```



3.2 파이썬 - 시각화 기본

1. Line charts++

```
from matplotlib import pyplot as plt  
  
years = [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]  
wage = [3480, 3770, 4000, 4110, 4320, 4550, 4850, 5150, 5550, 6050, 6500]  
  
#x축에 연도 y축에 최저임금이 되는 :  
plt.plot(years, wage, color='green',  
         label='wage')  
  
#제목과 라벨을 달아보자.  
plt.title("Korean minimum wage")  
plt.xlabel("years")  
plt.ylabel("minimum wage")  
  
#눈금을 표시해주는 함수  
plt.grid(True)  
  
#범례를 표시해주는 함수  
plt.legend()  
  
plt.show()
```



3.2 파이썬 - 시각화 기본

1. Line charts++

Commander	Function	입력 표현
plt.show()	그래프 나타내기	>>> plt.show()
plt.savefig()	저장하기	>>> plt.savefig()
plt.xlabel("x축 label") plt.ylabel("y축 label")	축 label	>>> plt.xlabel("# of students")
plt.xticks(인덱스, 변량명) plt.yticks(인덱스, 변량명)	막대별 label	>>> plt.xticks(x, object)
plt.axis()	축 조절	>>> plt.axis([xmin, xmax, ymin, ymax]) >>> plt.axis('off') >>> plt.axis('equal')
plt.grid(T/F)	격자눈금	>>> plt.grid(true)
plt.legend(위치)	범주	>>> plt.legend(loc='upper left') >>> plt.legend(loc='top center')
plt.annotate()	포인트 레이블	

3.2 파이썬 - 시각화 기본

1. Line charts ++

```
from matplotlib import pyplot as plt

variance = [1,2,4,8,16,32,64,128,256]
bias_squared = [256,128,64,32,16,8,4,2,1]
total_error = [x+y for x,y in zip(variance, bias_squared)]
xs = [i for i, _ in enumerate(variance)]

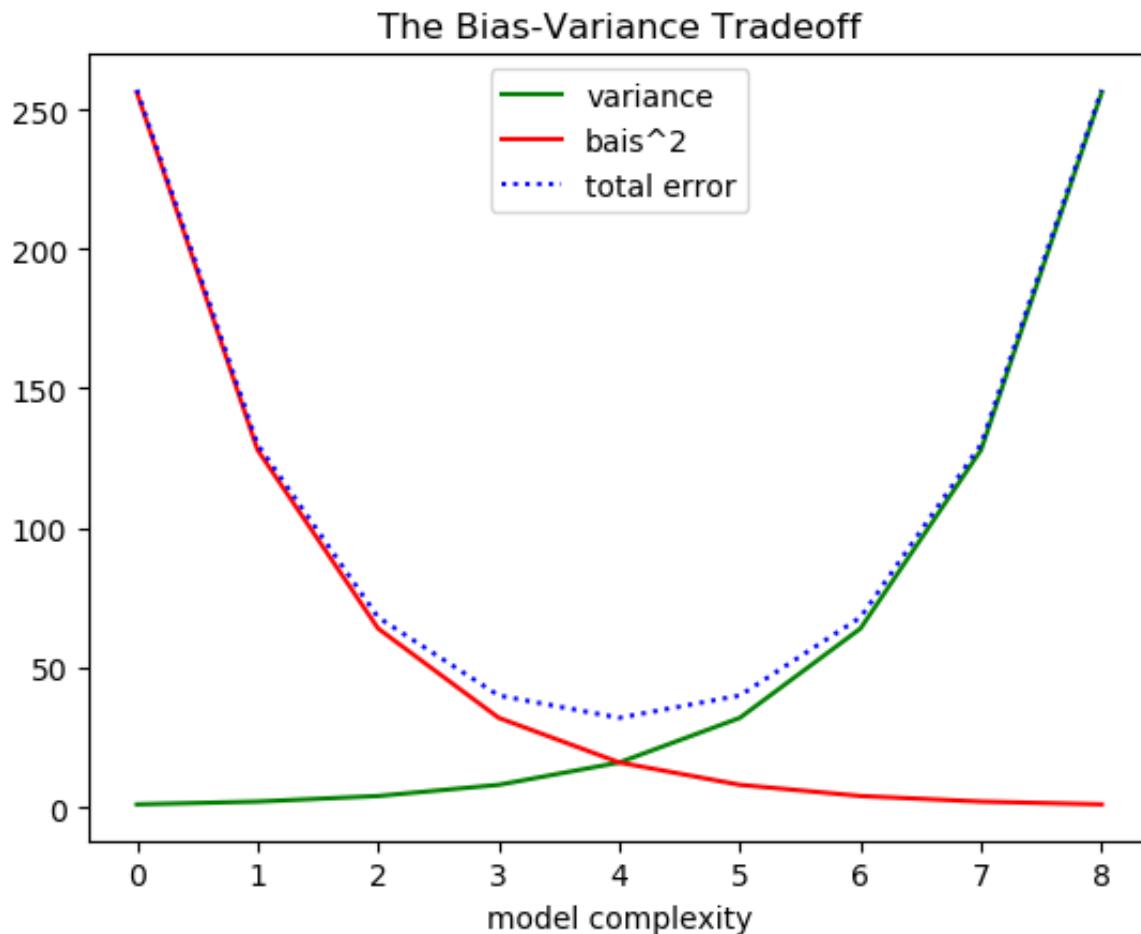
plt.plot(xs, variance, 'g-', label='variance')
plt.plot(xs, bias_squared, 'r-', label='bias^2')
plt.plot(xs, total_error, 'b:', label='total error')

plt.legend(loc=9)
plt.xlabel("model complexity")
plt.title("The Bias-Variance Tradeoff")
plt.show()
```

3.2 파이썬 - 시각화 기본

1. Line charts ++

```
from matplotlib import pyplot as plt  
  
variance = [1, 2, 4]  
bias_squared = [250, 100, 20]  
total_error = [250, 130, 40]  
xs = [i for i in range(9)]  
  
plt.plot(xs, variance, color='green', label='variance')  
plt.plot(xs, bias_squared, color='red', label='bias^2')  
plt.plot(xs, total_error, color='blue', label='total error')  
  
plt.legend(loc=9)  
plt.xlabel("model complexity")  
plt.title("The Bias-Variance Tradeoff")  
plt.show()
```



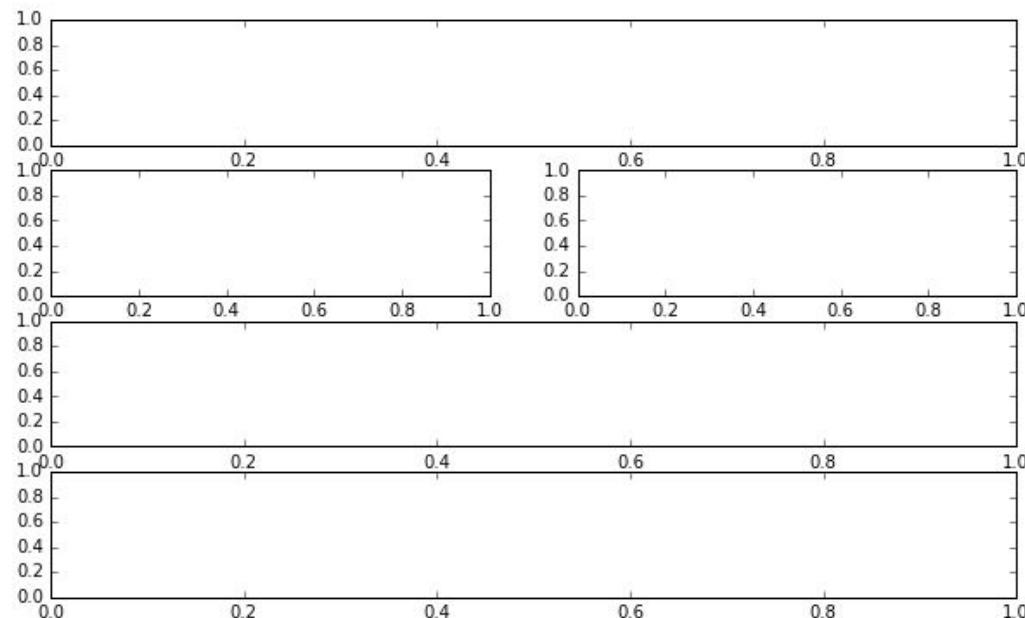
3.2 파이썬 - 시각화 기본

1. Line charts ++

```
In [7]: plt.figure(figsize=(10,6))

plt.subplot(411)
plt.subplot(423)
plt.subplot(424)
plt.subplot(413)
plt.subplot(414)

plt.show()
```



3.2 파이썬 - 시각화 기본

1. Line charts ++

```

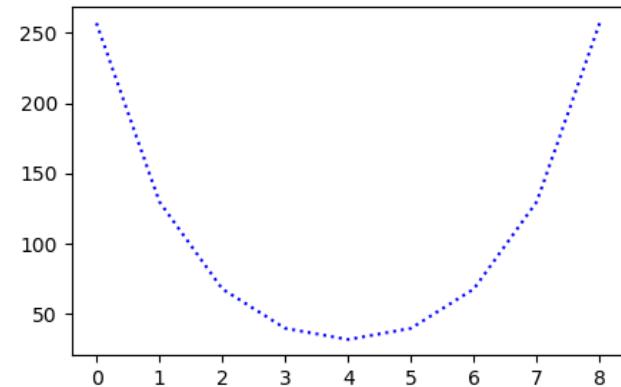
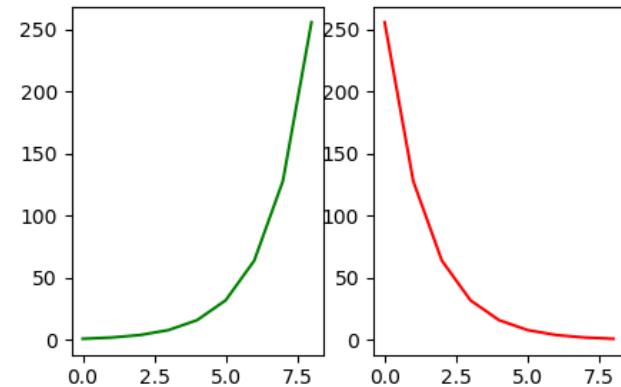
from matplotlib import pyplot as plt

variance = [1,2,4,8,16,32,64,128,256]
bias_squared = [256,128,64,32,16,8,4,2,1]
total_error = [x+y for x,y in zip(variance, bias_squared)]
xs = [i for i, _ in enumerate(variance)]

plt.figure(figsize=(5,7))
plt.subplot(221)
plt.plot(xs, variance, 'g-', label='variance')
plt.subplot(222)
plt.plot(xs, bias_squared, 'r-', label='bias^2')
plt.subplot(212)
plt.plot(xs, total_error, 'b:', label='total error')

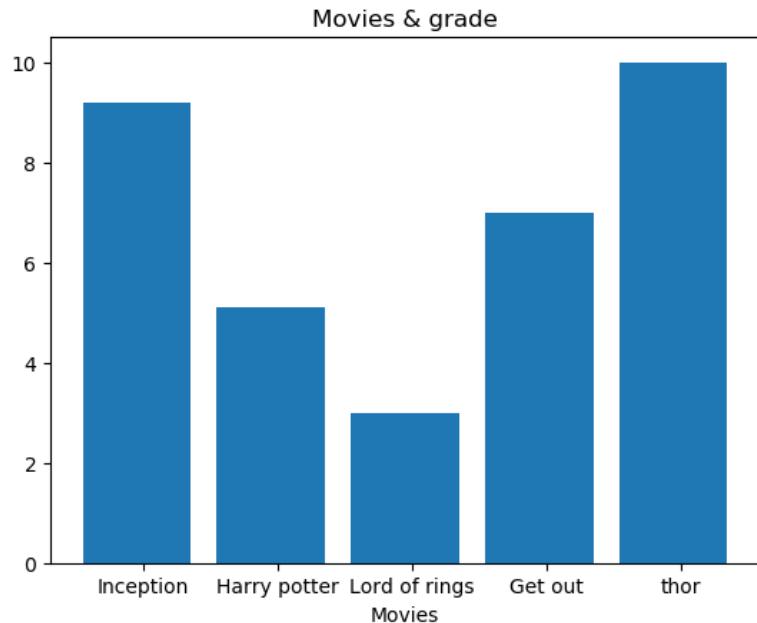
plt.show()

```

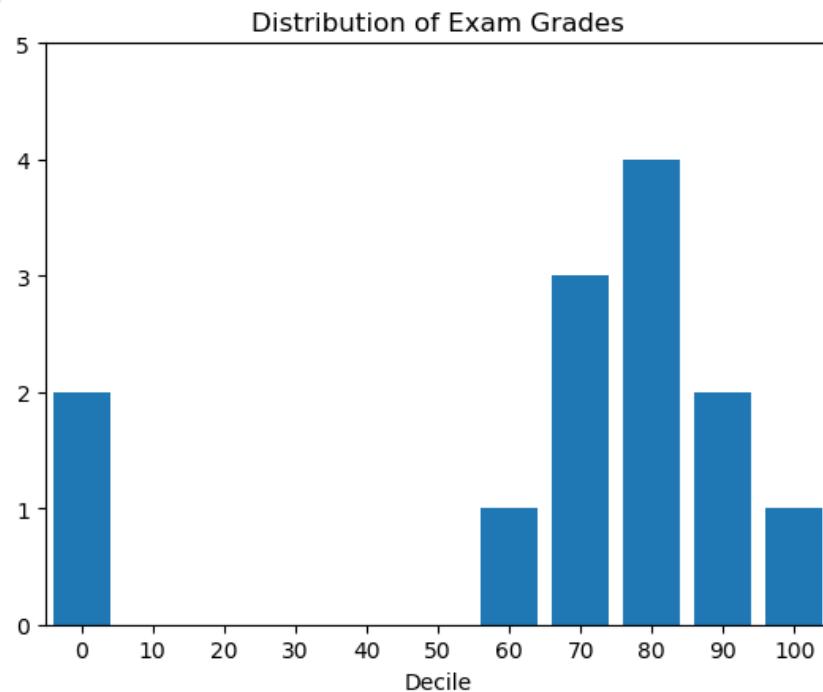


3.2 파이썬 - 시각화 기본

Bar charts VS histogram



이산과 연속



도수와 변량

3.2 파이썬 - 시각화 기본

2. Bar charts

```
from matplotlib import pyplot as plt

movies = ["Inception", "Harry potter", "Lord of rings", "Get out", "thor"]
grade = [9.2, 5.1, 3.7, 0.1, 10]

xs = [i + 0.1 for i, _ in enumerate(movies)]
#xs = (0.1, 1.1, 2.1, 2.1, 3.1, 4.1)

plt.bar(xs, grade)
plt.title("Movies & grade")
plt.xlabel("Movies")

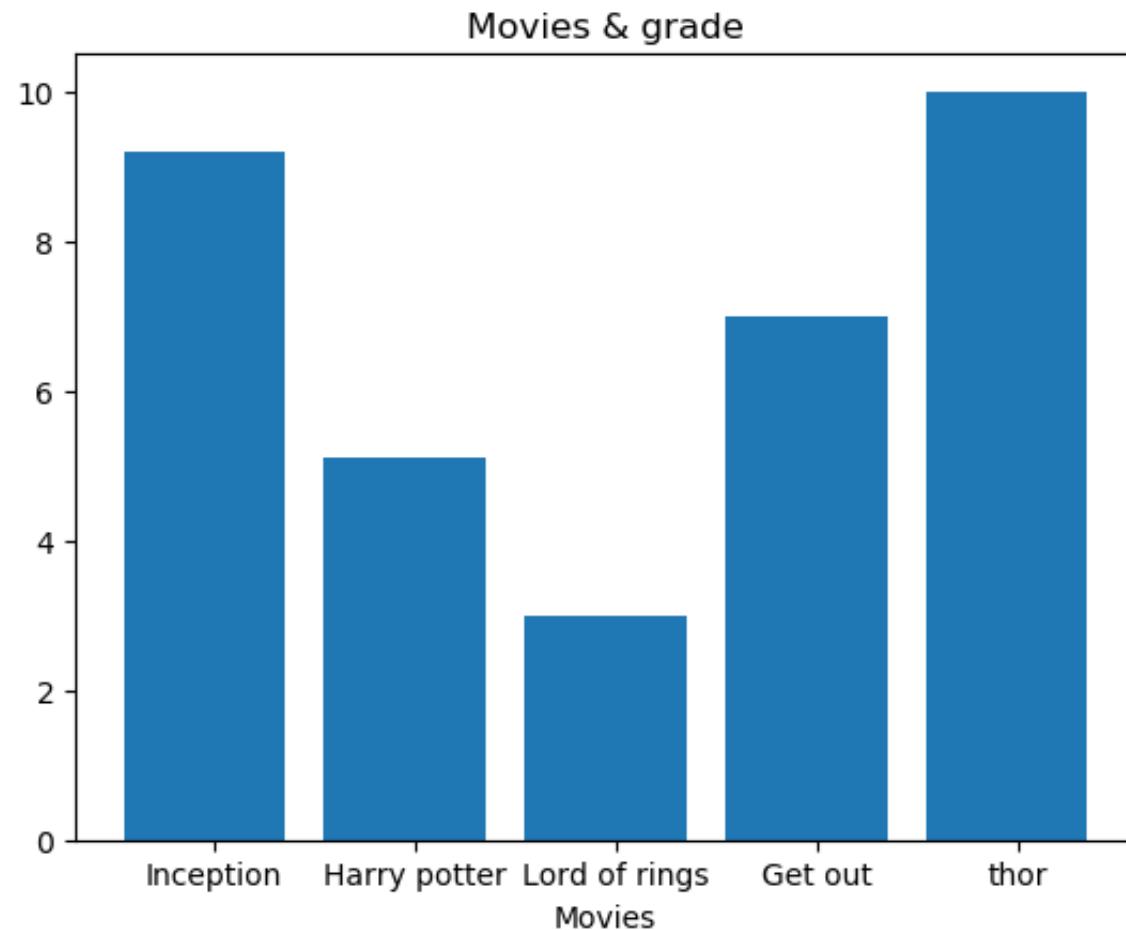
plt.xticks([i + 0.15 for i, _ in enumerate(movies)], movies)

plt.show()
```

3.2 파이썬 - 시각화 기본

2. Bar charts

```
from matplotlib import  
  
movies = ["Inception",  
grade = [9.2,5.1,3.7,6.2,  
xs = [i + 0.1 for i, _  
plt.bar(xs,grade)  
plt.title("Movies & grade")  
plt.xlabel("Movies")  
plt.xticks([i + 0.15 for i, _  
plt.show()
```



3.2 파이썬 - 시각화 기본

3. Histogram

```
from collections import Counter
from matplotlib import pyplot as plt

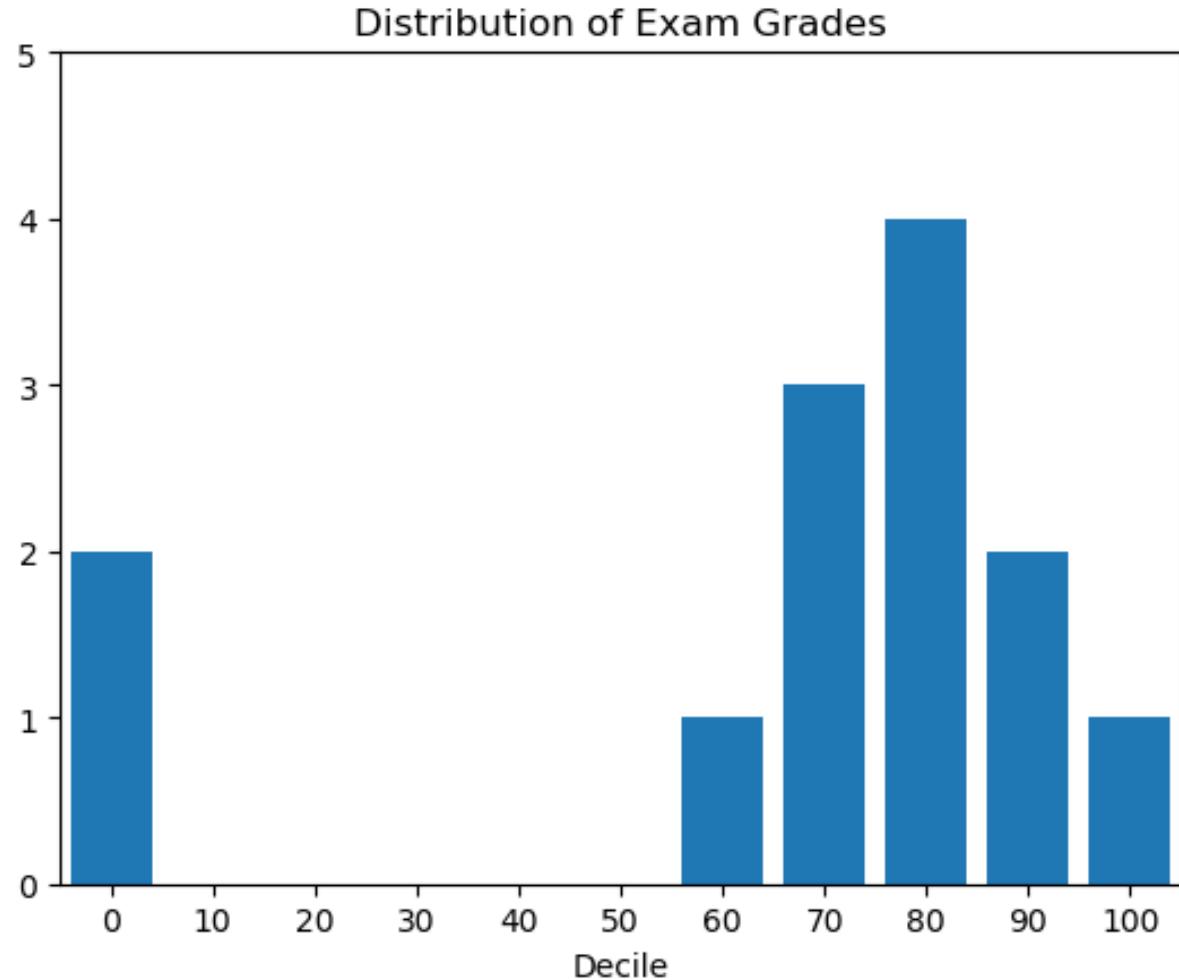
grades = [83,95,91,87,70,0,85,82,100,67,73,77,0]
decile = lambda grade : grade// 10*10
his = Counter(decile(grade) for grade in grades)
plt.bar([x for x in his.keys()], his.values(),8)

plt.axis([-5,105,0,5])
plt.xticks([10*i for i in range(11)])
plt.xlabel("Decile")
plt.title("Distribution of Exam Grades")
plt.show()
```

3.2 파이썬 - 시각화 기본

3. Histogram

```
from collections import  
from matplotlib import  
  
grades = [83,95,91,87,  
decile = lambda grade:  
his = Counter(decile(  
plt.bar([x for x in h  
  
plt.axis([-5,105,0,5])  
plt.xticks([10*i for  
plt.xlabel("Decile")  
plt.title("Distribution of Exam Grades")  
plt.show()
```



3.2 파이썬 - 시각화 기본

4.Scatterplots

```
from matplotlib import pyplot as plt

math = [23,34,45,45,56,60,66,75,78,88]
history = [45,24,64,54,65,78,80,80,88,100]
labels = ['a','b','c','d','e','f','g','i','j','k']

plt.scatter(math,history)
plt.title("math & history grades")
plt.xlabel("math")
plt.ylabel("history")

#여기서 멈추지 않고 레이블을 달면 좋음

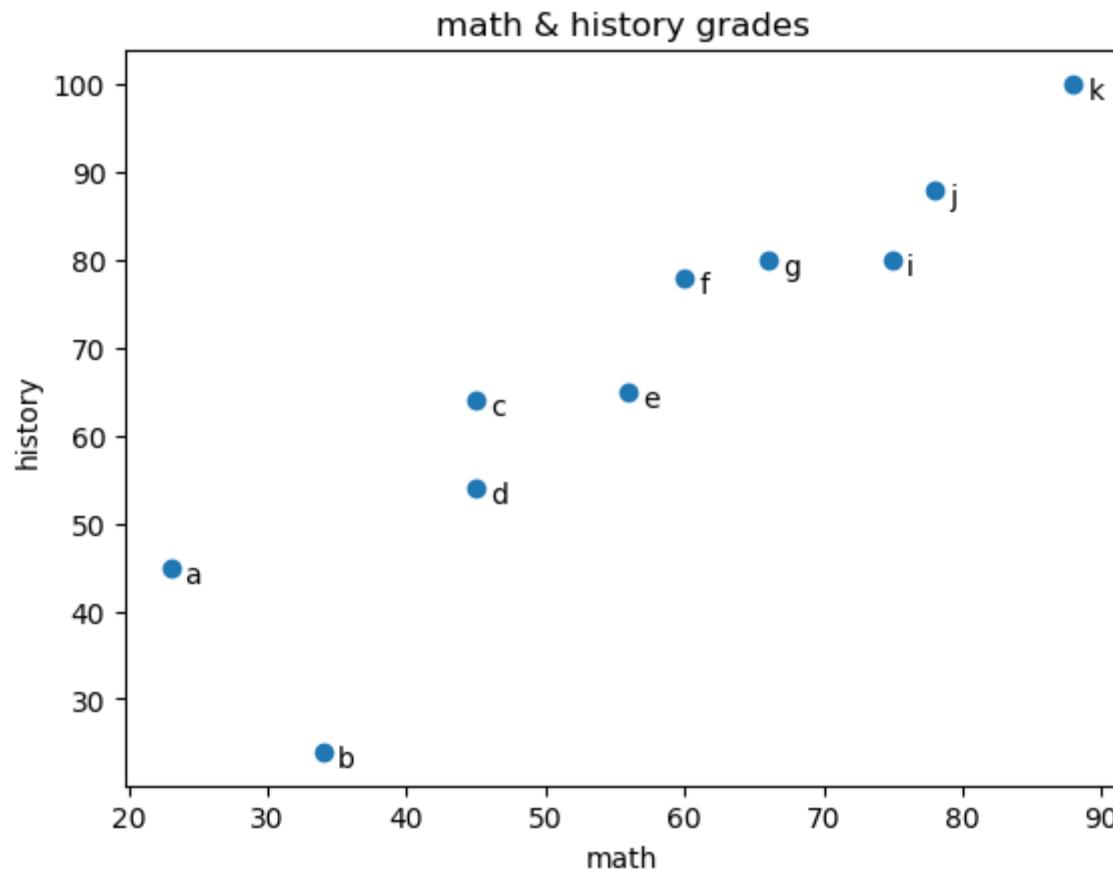
for label, math_count, his_count in zip(labels,math,history) :
    plt.annotate(label,
                 xy=(math_count,his_count),
                 xytext=(5,-5),
                 textcoords='offset points')

plt.show()
```

3.2 파이썬 - 시각화 기본

4.Sc

```
from matplotlib import pyplot as plt  
  
math = [23, 45, 60, 75, 88]  
history = [45, 55, 65, 80, 90]  
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'i', 'j', 'k']  
  
plt.scatter(math, history)  
plt.title("math & history grades")  
plt.xlabel("math")  
plt.ylabel("history")  
  
#여기서 멈추지  
  
for label, mat, hist in zip(labels, math, history):  
    plt.annotate(label, (mat, hist))  
  
plt.show()
```



3.2 파이썬 - 시각화 기본

4.Scatterplots

```
from matplotlib import pyplot as plt

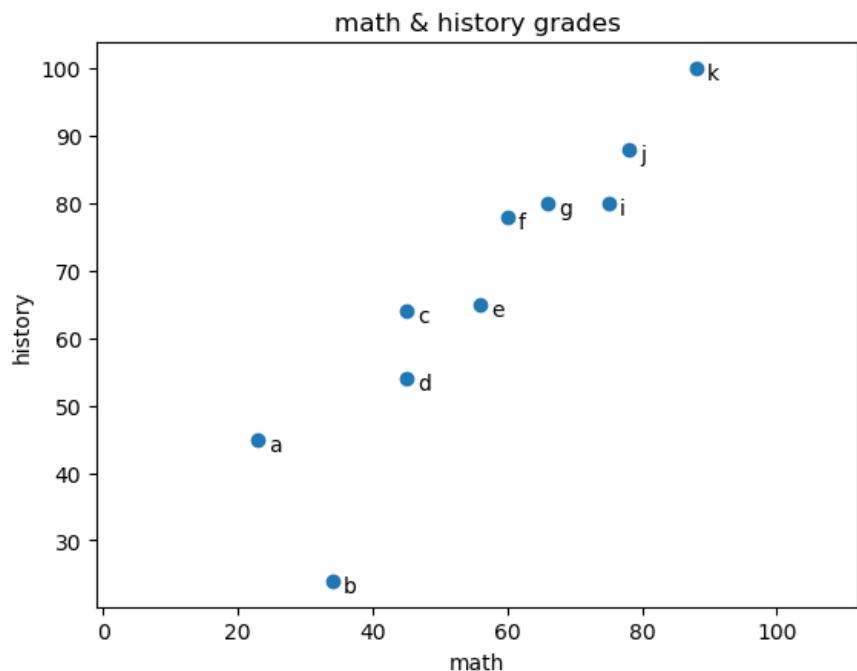
math = [23,34,45,45,56,60,66,75,78,88]
history = [45,24,64,54,65,78,80,80,88,100]
labels = ['a','b','c','d','e','f','g','i','j','k']

plt.scatter(math,history)
plt.title("math & history grades")
plt.xlabel("math")
plt.ylabel("history")

#여기서 멈추지 않고 레이블을 달면 좋음
for label, math_count, his_count in zip(labels,math,history) :
    plt.annotate(label,
                 xy=(math_count,his_count),
                 xytext=(5,-5),
                 textcoords='offset points')

#공평하게
plt.axis("equal")

plt.show()
```



3.3 Tableau

Tableau?

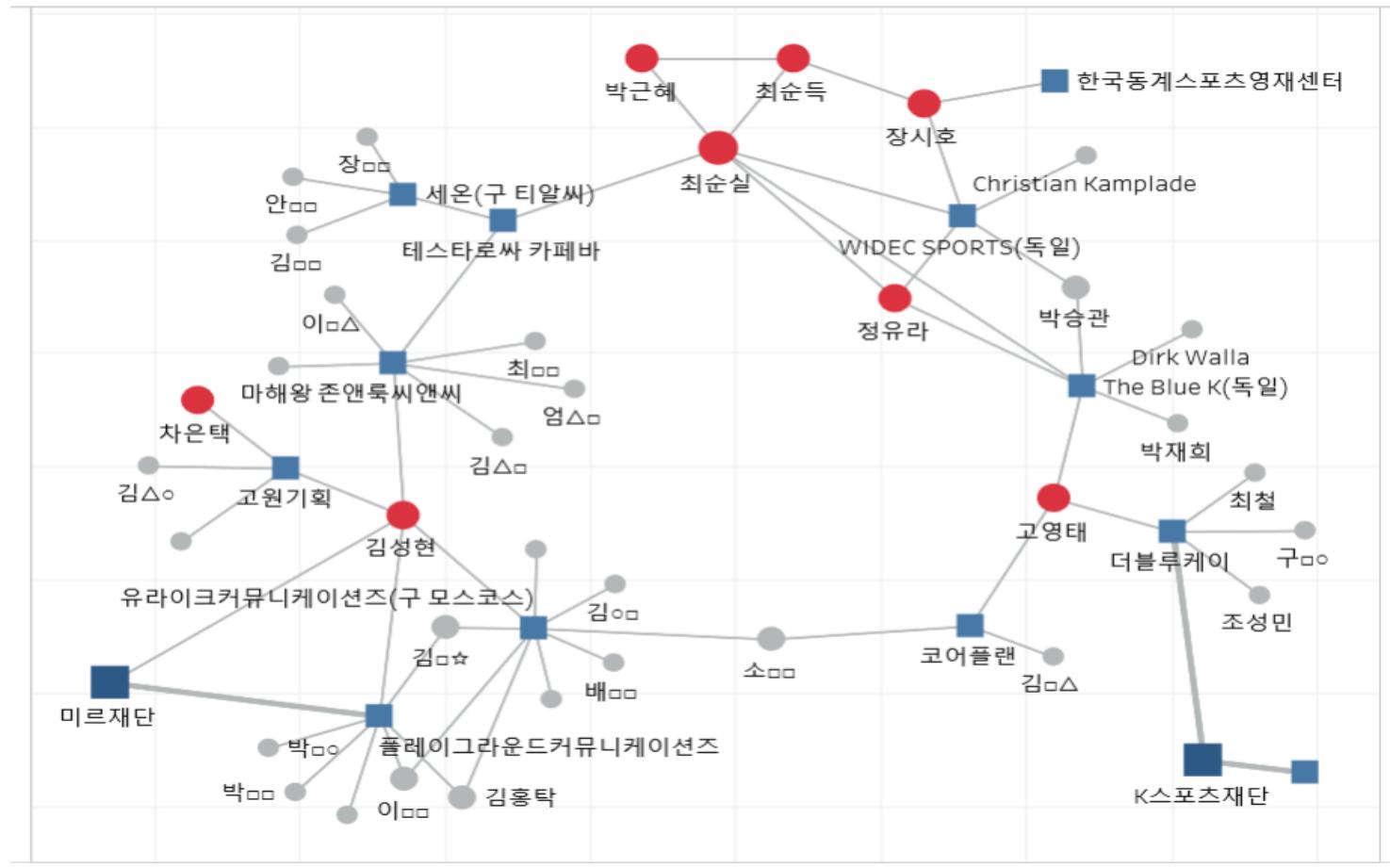
웹을 위한 복잡하고 인터랙티브한 시각화에 적절함
>디지털 미디어 시대에 적합.

세련된 디자인

너무너무 쉽다는 게 가장 큰 장점

3.3 Tableau

Interactive?



<http://newstapa.org/35607>

3.3 Tableau

Interactive?

독자의 행동에 의해 텍스트 혹은 시각화된 자료가 변화함.

미래에 변화하게 될 자료들

편하게 감상하시고 관심있는 분들은

공부해 보세요.

CONTENTS

3. 데이터 시각화

3.1 상황에 맞는 시각화

3.2 파이썬 시각화 기본

3.3 Tableau

10. 데이터 다루기

10.1 1차원 데이터 분석

10.2 2차원 데이터 분석

10.3 척도 조절

10.4 차원 축소

O. 데이터 정제

DATA CLEANSING

현재 데이터

```
2     f = open("bodydims.csv", 'r')
3
4     lines = f.readlines()
5
6     print(lines)
7
8
9
practice2
C:\Users\Administrator\AppData\Local\Programs\Python\Python36-32\python.exe C:/Users/Administrator/PycharmProjects/codeit/prac
['bii.di,che.de,e1b.di,kne.di,age,wgt,hgt,sex\n', '26,17.7,13.1,18.8,21,65.6,174,1\n', '28.5,16.9,14,20.6,23,71.8,175.3,1\n',
 Process finished with exit code 0
```

정제 후 데이터

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python36-32\python.exe C:/Users/Administrator/PycharmProjects/codeit/why
bii : [26.0, 28.5, 28.2, 29.9, 29.9, 27.0, 30.0, 29.8, 26.5, 28.0, 29.0, 29.0, 29.6, 27.5, 26.8, 27.0, 30.0, 26.5, 28.6, 29.1,
wgt : [65.6, 71.8, 80.7, 72.6, 78.8, 74.8, 86.4, 78.4, 62.0, 81.6, 76.6, 83.6, 90.0, 74.6, 71.0, 79.6, 93.8, 70.0, 72.4, 85.3,
hgt : [174.0, 175.3, 193.5, 186.5, 187.2, 181.5, 184.0, 184.5, 175.0, 184.0, 180.0, 177.8, 192.0, 176.0, 174.0, 184.0, 192.7]
```

O. 데이터 정제

DATA CLEANSING

1) Header 제거

2) .strip() 메소드

3) .split() 메소드

4) List 생성과 for 문으로
.append()

```
17 f = open("bodydims.csv", 'r')
18 lines = f.readlines()
19
20 del lines[0]
21
22
23
24 wgt = []
25 hgt = []
26 bii = []
27
28
29
30 for line in lines:
31     data = line.strip().split(',')
32     bii.append(data[0])
33     wgt.append(data[5])
34     hgt.append(data[6])
35
36 f.close()
37
38 print(bii)
39 print(wgt)
40 print(hgt)
41
```

O. 데이터 정제

DATA CLEANSING

1) Header 제거

2) `.strip()` 메소드

3) `.split()` 메소드

4) List 생성과 for 문으로
`.append()`

1	bii,di,che,de,elb,di,kne,di,age,wgt,hgt,sex
2	26,17,7,13,1,18,8,21,65,6,174,1
3	28,5,16,9,14,20,6,23,71,8,175,3,1
4	28,2,20,9,13,9,19,7,28,80,7,193,5,1
5	29,9,18,4,13,9,20,9,23,72,6,186,5,1

```

17
18     f = open("bodydims.csv", 'r')
19     lines = f.readlines()
20
21     del lines[0]
22
23
24     wgt = []
25     hgt = []
26     bii = []
27
28
29
30     for line in lines:
31         data = line.strip().split(',')
32         bii.append(data[0])
33         wgt.append(data[5])
34         hgt.append(data[6])
35
36     f.close()
37
38     print(bii)
39     print(wgt)
40     print(hgt)
41

```

O. 데이터 정제

DATA CLEANSING

1) Header 제거

2) .strip() 메소드

앞 뒤의
공백 제거

3) .split() 메소드

특정 키워드로
데이터 분리한
후 리스트 리턴

4) List 생성과 for 문으로
.append()

```

17 f = open("bodydims.csv", 'r')
18 lines = f.readlines()
19
20 del lines[0]
21
22
23
24 wgt = []
25 hgt = []
26 bii = []
27
28
29
30 for line in lines:
31     data = line.strip().split(',')
32     bii.append(data[0])
33     wgt.append(data[5])
34     hgt.append(data[6])
35
36 f.close()
37
38 print(bii)
39 print(wgt)
40 print(hgt)
41

```

O. 데이터 정제

DATA CLEANSING

1) Header 제거

2) .strip() 메소드

3) .split() 메소드

4) List 생성과 for 문으로
.append()

해당 리스트
끝에 원소 추가

```

17 f = open("bodydims.csv", 'r')
18 lines = f.readlines()
19
20 del lines[0]
21
22
23
24 wgt = []
25 hgt = []
26 bii = []
27
28
29
30 for line in lines:
31     data = line.strip().split(',')
32     bii.append(data[0])
33     wgt.append(data[5])
34     hgt.append(data[6])
35
36 f.close()
37
38 print(bii)
39 print(wgt)
40 print(hgt)
41

```

O. 데이터 정제

DATA CLEANSING

1) Header 제거

```

17
18     f = open("bodydims.csv", 'r')
19     lines = f.readlines()
20
21     del lines[0]

```

```
hy_datacleanse1(bodydims)
```

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/Ken/PycharmPr
['26', '28.5', '28.2', '29.9', '29.9', '27', '30', '29.8', '26.5', '28', '29', '29',
['65.6', '71.8', '80.7', '72.6', '78.8', '74.8', '86.4', '78.4', '62', '81.6', '76.6
['174', '175.3', '193.5', '186.5', '187.2', '181.5', '184', '184.5', '175', '184', '1
```

Process finished with exit code 0

여기서 끝나면 안 됨

4) List 생성과 for 문으로 .append()

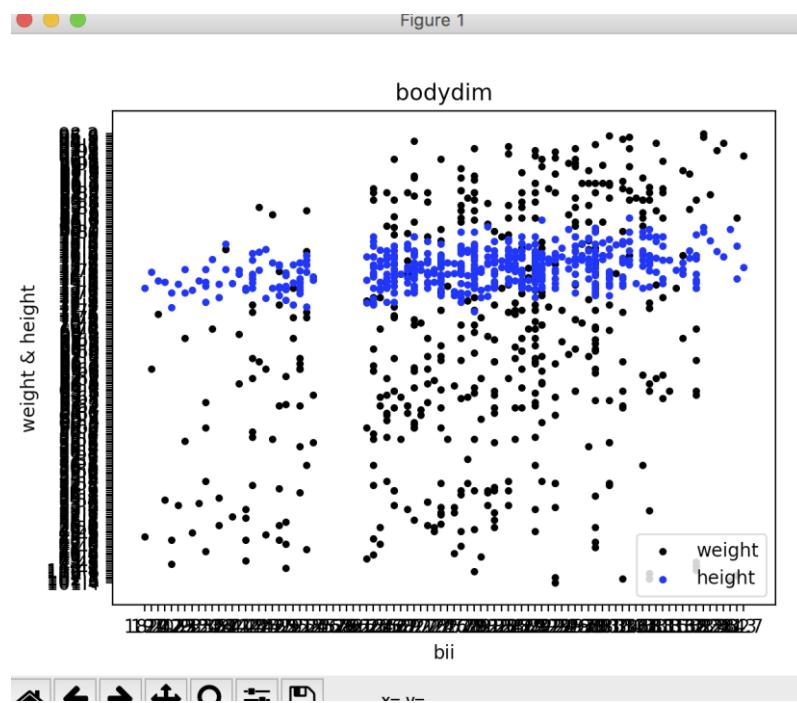
```

33             wgt.append(data[5])
34             hgt.append(data[6])
35
36             f.close()
37
38             print(bii)
39             print(wgt)
40             print(hgt)
41

```

O. 데이터 정제

DATA CLEANSING



```

17
18
19
20
21
f = open("bodydims.csv", 'r')
lines = f.readlines()
del lines[0]

ions/3.6/bin/python3.6 /Users/Ken/PycharmPr
7', '30', '29.8', '26.5', '28', '29', '29',
'74.8', '86.4', '78.4', '62', '81.6', '76.6
', '181.5', '184', '184.5', '175', '184', '1

```

산점도 실행시 오류
(why_datacleanse1(bodydims).py 파일 실행)

```

33
34
35
36
37
38
39
40
30
31
32
33
34
35
36
37
38
39
40
41
wgt.append(data[5])
hgt.append(data[6])

f.close()

print(bii)
print(wgt)
print(hgt)

```

O. 데이터 정제

DATA CLEANSING

1) Header 제거

2) .strip() 메소드

3) .split() 메소드

4) List 생성과 for 문으로
.append()

5) Int, float으로 type
변환하기

```
f = open("bodydims.csv", 'r')
lines = f.readlines()

del lines[0]

wgt = []
hgt = []
bii = []

for line in lines:
    data = line.strip().split(',')
    bii.append(float(data[0]))
    wgt.append(float(data[5]))
    hgt.append(float(data[6]))

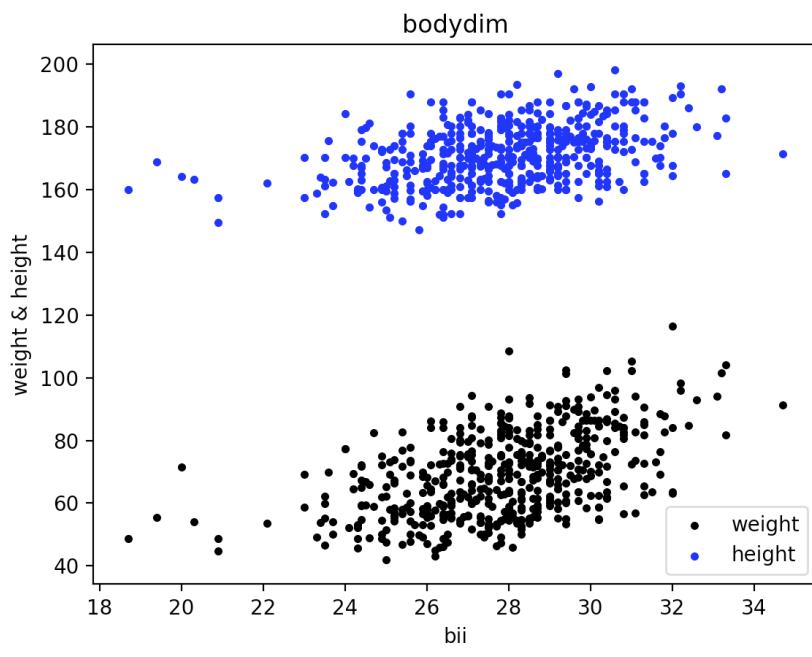
f.close()

print("bii : ", bii)
print("wgt : ", wgt)
print("hgt : ", hgt)
```

O. 데이터 정제

DATA CLEANSING

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python36-32\python.exe C:/Users/Administrator/PycharmProjects/codeit/why  
bill : [26.0, 28.5, 28.2, 29.9, 29.9, 27.0, 30.0, 29.8, 26.5, 28.0, 29.0, 29.0, 29.6, 27.5, 26.8, 27.0, 30.0, 26.5, 28.6, 29.0,  
wgt : [65.6, 71.8, 80.7, 72.6, 78.8, 74.8, 86.4, 78.4, 62.0, 81.6, 76.6, 83.6, 90.0, 74.6, 71.0, 79.6, 93.8, 70.0, 72.4, 85.0,  
hgt : [174.0, 175.3, 193.5, 186.5, 187.2, 181.5, 184.0, 184.5, 175.0, 184.0, 180.0, 177.8, 192.0, 176.0, 174.0, 184.0, 192.7]
```



```
wgt = []
hgt = []
bii = []

산점도 실행시 정상.
(why_datacleanse2(body)
파일 실행)

for line in lines:
    data = line.strip().split(',')
    bii.append(float(data[0]))
    wgt.append(float(data[1]))
    hgt.append(float(data[6]))

f.close()

print("bii : ", bii)
print("wgt : ", wgt)
print("hgt : ", hgt)
```

O. 데이터 정제

DATA CLEANSING

Int, float 타입 변환하기

→ 방법 두 가지 존재

1) List() & map()

```

for line in lines:
    data = line.strip().split(',')
    sat.append(data[0])
    avg_mhr.append(data[3])

f.close()

t_sat = list(map(float, sat))
int_avg_mhr = list(map(int, avg_mhr))

```

The code shows two lines highlighted with a red oval: `t_sat = list(map(float, sat))` and `int_avg_mhr = list(map(int, avg_mhr))`. These lines convert the lists `sat` and `avg_mhr` into lists of floats and integers respectively.

2) For문 내부에서 float(), int() 함수 돌리기

```

for line in lines:
    data = line.strip().split(',')
    sat.append(float(data[0]))
    avg_mhr.append(int(data[3]))

```

The code shows the same logic as the first method, but instead of using `list(map(...))`, it uses `float(data[0])` and `int(data[3])` directly within the `for` loop to convert the elements.

O. 데이터 정제

DATA CLEANSING - 실습

“HR_comma_sep.csv” 파일에서
만족도(“satisfaction_level”) 와
월평균 근로시간(“average_monthly_hours”)
데이터 추출 및 정제

	A	B	C	D	E	F	G	H	I	J
	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	left	promotion_last_5years	sales	salary
0	0.38	0.53	2	157	3	0	1	0 sales	low	
1	0.8	0.86	5	262	6	0	1	0 sales	medium	
2	0.11	0.88	7	272	4	0	1	0 sales	medium	
3	0.72	0.87	5	223	5	0	1	0 sales	low	
4	0.37	0.52	2	159	3	0	1	0 sales	low	
5	0.41	0.5	2	153	3	0	1	0 sales	low	
6	0.1	0.77	6	247	4	0	1	0 sales	low	
7	0.92	0.85	5	259	5	0	1	0 sales	low	
8	0.89	1	5	224	5	0	1	0 sales	low	
9	0.42	0.53	2	142	3	0	1	0 sales	low	
10	0.45	0.54	2	135	3	0	1	0 sales	low	
11	0.11	0.81	6	305	4	0	1	0 sales	low	
12	0.84	0.92	4	234	5	0	1	0 sales	low	
13	0.41	0.55	2	148	3	0	1	0 sales	low	
14	0.36	0.56	2	137	3	0	1	0 sales	low	
15	0.38	0.54	2	143	3	0	1	0 sales	low	
16	0.45	0.47	2	160	3	0	1	0 sales	low	
17	0.78	0.99	4	255	6	0	1	0 sales	low	
18	0.45	0.51	2	160	3	1	1	1 sales	low	
19	0.76	0.89	5	262	5	0	1	0 sales	low	
20	0.11	0.83	6	282	4	0	1	0 sales	low	
21	0.38	0.55	2	147	3	0	1	0 sales	low	
22	0.09	0.95	6	304	4	0	1	0 sales	low	
23	0.46	0.57	2	139	3	0	1	0 sales	low	
24	0.4	0.53	2	158	3	0	1	0 sales	low	

O. 데이터 정제

DATA CLEANSING - answer

```
1  f = open('HR_comma_sep.csv', 'r')
2
3  lines = f.readlines()
4
5  del lines[0]
6
7  sat = []
8  avg_mhr = []
9
10 for line in lines:
11     data = line.strip().split(',')
12     sat.append(float(data[0]))
13     avg_mhr.append(int(data[3]))
14
15
16 f.close()
17
18 print(sat)
19 print(avg_mhr)
```

```
Run: HR_datacleanse why_datacleanse2(bodydims) HR_hist ames_practice
> /Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/Ken/PycharmProjects/codeit/HR_datacle
[0.38, 0.8, 0.11, 0.72, 0.37, 0.41, 0.1, 0.92, 0.89, 0.42, 0.45, 0.11, 0.84, 0.41, 0.36, 0.38, 0.45, 0.78, 0
[157, 262, 272, 223, 159, 153, 247, 259, 224, 142, 135, 305, 234, 148, 137, 143, 160, 255, 160, 262, 282, 14
```

10.1 1차원 데이터 분석

■ 분석 방법

1) 요약 통계치

2) Histogram 사용

10.1 1차원 데이터 분석

■ 요약통계치

- 1) 구성요소 : 데이터 개수, 최솟값, 최댓값, 평균, 표준편차 등
- 2) How? : **pandas** 라이브러리

pandas 패키지를 사용하면 엑셀에서 하는 기본
데이터 요약 통계치 기능을 쉽게 처리할 수 있다

pandas is a **Python** package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in **Python**

import pandas as pd

<https://pandas.pydata.org/pandas-docs/stable/>



10.1 1차원 데이터 분석

■ 요약통계치

Pandas의 자료구조 : 변환 필요!

Python의 데이터 타입을 pandas로 바꿔줘야..

1) Series : 한 개의 열 (1차원 데이터)

2) Dataframe : 여러 개의 열 (2차원 데이터)

```
30
31     sat_sf = pd.Series(sat)
32     avg_mhr_sf = pd.Series(avg_mhr)
33     print(sat_sf)
34
35
```

	/Library/Frameworks/
0	0.38
1	0.80
2	0.11
3	0.72
4	0.37
5	0.41
6	0.10
7	0.92
8	0.89
9	0.42
10	0.45
11	0.11
12	0.84
13	0.41
14	0.36
15	0.38
16	0.45

10.1 1차원 데이터 분석

■ 요약통계치

Seriesform으로 바꿨으면 이제
.describe() 함수로 요약통계치를 표현해보자

```

23
24     sat_sf = pd.Series(sat)
25     avg_mhr_sf = pd.Series(avg_mhr)
26     print(sat_sf.describe())
27     print(' ')
28     print(avg_mhr_sf.describe())
29

```

개수
평균
표준편차
최솟값
제1사분위 값
중앙값
제3사분위 값
최댓값

```

/Library/Frameworks/Python.
count    14999.000000
mean      0.612834
std       0.248631
min       0.090000
25%      0.440000
50%      0.640000
75%      0.820000
max      1.000000
dtype: float64

```

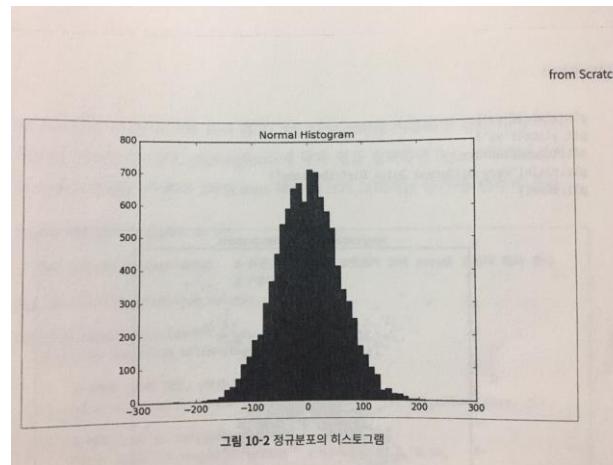
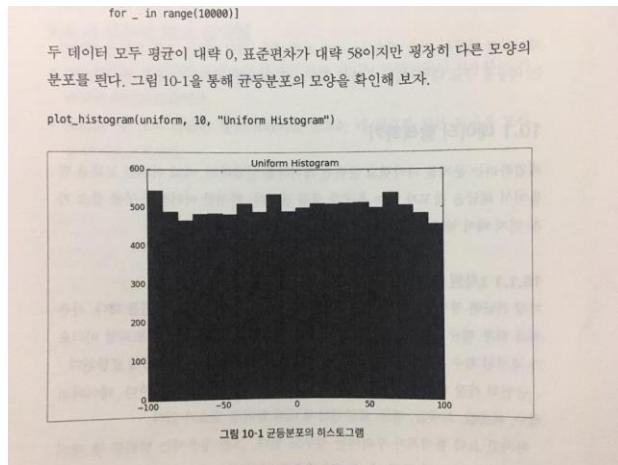
	/Library/Frameworks/
0	0.38
1	0.80
2	0.11
3	0.72
4	0.37
5	0.41
6	0.10
7	0.92
8	0.89
9	0.42
10	0.45
11	0.11
12	0.84
13	0.41
14	0.36
15	0.38
16	0.45

10.1 1차원 데이터 분석

■ 요약통계치 - 한계

But 요약통계치로 충분히 표현하지 못하는 자료 존재..

자료 개형 더 직관적으로 알기 위해 쓸 수 있는
1차원 데이터 분석법 → **histogram**



(밑바닥 책 p.125-127)

10.1 1차원 데이터 분석

Histogram

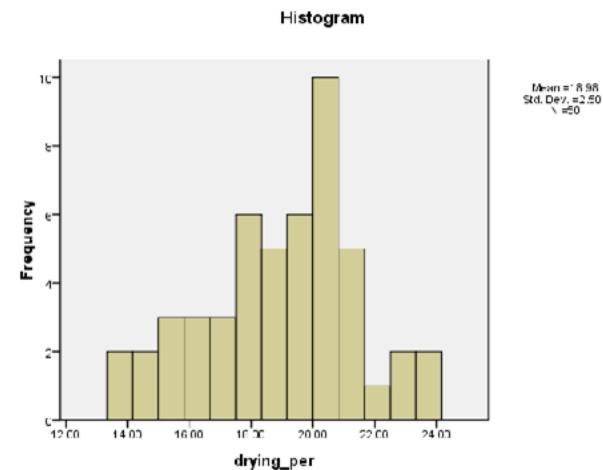
1) What :

도수 분포의 상태를 기둥 모양의 그래프로 나타낸 것

2) How :

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

→ `plt.hist()` 함수 사용



10.1 1차원 데이터 분석

Histogram

3) Example & Keywords :

```
sat_sf = pd.Series(sat)
avg_mhr_sf = pd.Series(avg_mhr)

plt.style.use('ggplot')

plt.hist(sat_sf, bins=20, alpha=0.5, color='c')
plt.axvline(x=sat_sf.mean(), color='b', linestyle='dashed', linewidth=2)
plt.xlabel("Satisfaction Level")
plt.show()

plt.hist(avg_mhr_sf, bins=20, alpha=0.5, color='r')
plt.axvline(x=avg_mhr_sf.mean(), color='b', linestyle='dashed', linewidth=2)
plt.xlabel("Average Monthly Hours")
plt.show()
```

10.1 1차원 데이터 분석

(1) argument

```
>>> plt.plot(x, y, color='', marker='', linestyle='')
```

(plt.plot은 선그래프 입력 함수, 각 그래프 별로 argument 조금씩 다름)

(2) commander

```
>>> plt.plot(x, y, color='', marker='', linestyle='')
```

```
>>> plt.xticks(x, objects)
```

```
>>> plt.grid(false)
```

```
>>> plt.show()
```

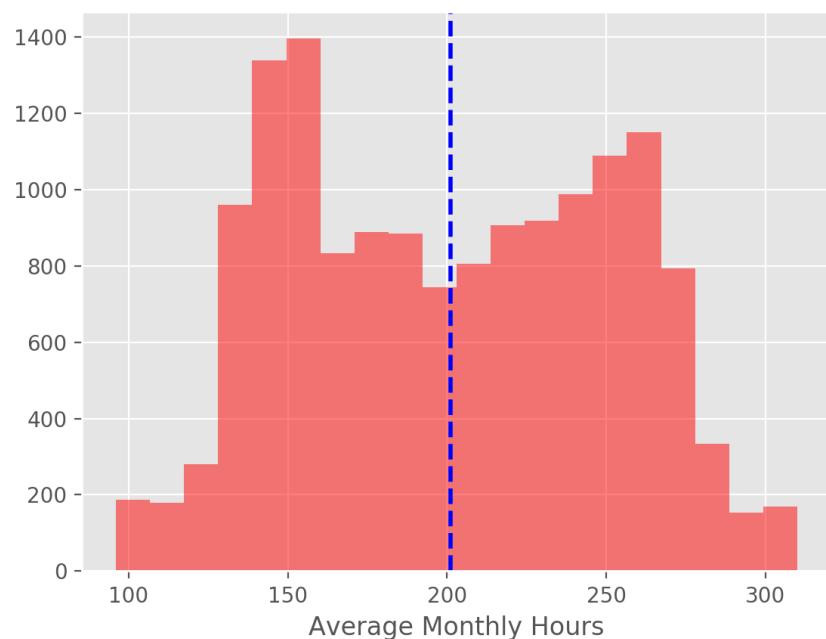
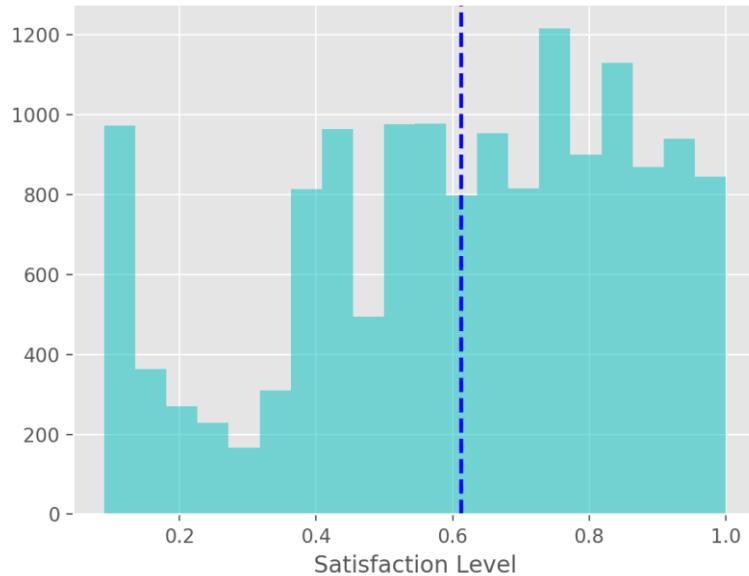
Commander	Function	입력 표현
plt.show()	그래프 나타내기	>>> plt.show()
plt.savefig()	저장하기	>>> plt.savefig()
plt.xlabel("x축 label") plt.ylabel("y축 label")	축 label	>>> plt.xlabel("# of students")
plt.xticks(인덱스, 변량명) plt.yticks(인덱스, 변량명)	막대별 label	>>> plt.xticks(x, object)
plt.axis()	축 조절	>>> plt.axis([xmin, xmax, ymin, ymax]) >>> plt.axis('off') >>> plt.axis('equal')
plt.grid(T/F)	격자눈금	>>> plt.grid(true)
plt.legend(위치)	범주	>>> plt.legend(loc='upper left') >>> plt.legend(loc='top center')
plt.annotate()	포인트 레이블	

10.1 1차원 데이터 분석

Histogram

3) Example & Keywords :

```
sat_sf = pd.Series(sat)
```



10.1 1차원 데이터 분석

■ 요약통계치 & 히스토그램 - 실습 : ames.csv

4.5 자료를 이용한 예제 (ames.csv)

주어진 자료는 Iowa의 도시 Ames의 2006년부터 2010년 사이의 부동산 거래내역 자료이다. 5년 동안 이 지역에서 발생한 총 2930건의 부동산 거래내역이 모두 기록되어 있다. 자료에 대한 자세한 설명은 다음을 참조한다.

(<http://www.openintro.org/stat/data/?data=ames>)

예제 1. 현재 주어진 자료는 일정 기간동안 지역 내의 모든 부동산 거래를 기록한 자료이므로 일종의 모집단이라고 생각할 수 있다. SalePrice 변수에 대해 히스토그램을 그려보고 수치적 요약값을 구해보자. 모집단의 분포는 어떠한가?

10.1 1차원 데이터 분석

■ 요약통계치 & 히스토그램 - 실습 : ames.csv (answer)

```

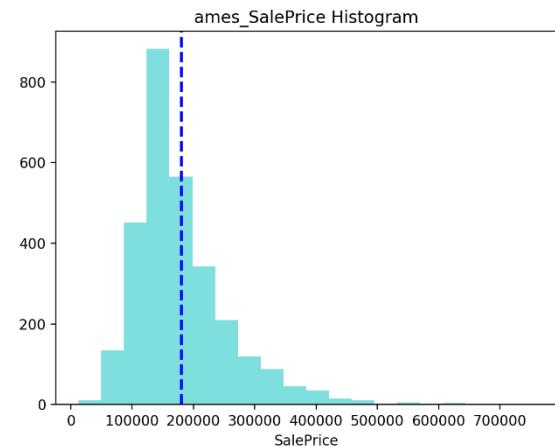
1  from matplotlib import pyplot as plt
2  import pandas as pd
3
4  f = open("ames.csv", 'r')
5
6  lines = f.readlines()
7
8  header = lines[0].strip().split(',')
9  price_index = header.index('SalePrice')
10
11 del lines[0]
12
13 price = []
14
15 for line in lines:
16     data = line.strip().split(',')
17     price.append(int(data[price_index]))
18
19 f.close()
20
21 price_sf = pd.Series(price)
22 print(price_sf.describe())
23
24 plt.hist(price, bins=20, alpha=0.5, color='c')
25 plt.axvline(x=price_sf.mean(), color='b', linestyle='dashed', linewidth=2)
26 plt.xlabel("SalePrice")
27 plt.title("ames_SalePrice Histogram")
28 plt.show()
29

```

```

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/pandas/core/generic.py:46: FutureWarning: pass index= to DataFrame constructor or use .loc[] accessor
  warnings.warn(msg, FutureWarning)
count      2930.000000
mean      180796.060068
std       79886.692357
min       12789.000000
25%      129500.000000
50%      160000.000000
75%      213500.000000
max      755000.000000
dtype: float64

```



10.2 2차원 데이터 분석

분석방법 : 산점도

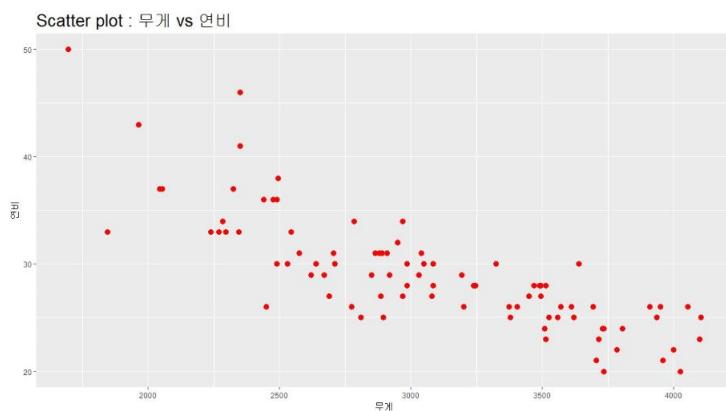
1) What :

두 개 이상 변수의 동시분포에서
각 개체를 점으로 표시한 그림

2) How :

```
from matplotlib import pyplot as plt
```

→ `plt.scatter()` 함수 사용



10.2 2차원 데이터 분석

■ 분석방법 : 산점도

3) Example : bodydims.csv

예) (bodydims.csv) 주어진 자료는 247명의 남성과 260명의 여성에 관한 신체 측정 자료이다. 원 자료는 신체의 각 부위를 측정한 총 25개의 변수가 있으며 본 예제에서는 그 중 8개 변수만 간추린 자료를 이용하기로 한다. 아래는 8개의 변수에 대한 설명이다. (원자료에 관한 자세한 설명은 다음을 참조하도록 한다. :

<http://www.openintro.org/stat/data/bdims.php>)

- bii.di : 숫자형 변수, 응답자의 골반의 넓이 (cm)
- che.de : 숫자형 변수, 응답자의 가슴 깊이 (cm)
- elb.di : 숫자형 변수, 응답자의 양쪽 팔꿈치 지름의 합. (cm)
- kne.di : 숫자형 변수, 응답자의 양쪽 무릎의 지름의 합. (cm)
- age : 숫자형 변수, 응답자의 나이 (years)
- wgt : 숫자형 변수, 응답자의 몸무게 (kg)
- hgt : 숫자형 변수, 응답자의 신장 (cm)
- sex : 범주형 변수, 응답자의 성별 (1=남성, 0=여성)

10.2 2차원 데이터 분석

분석방법 : 산점도

#강의자료: **bodydims.csv**
[Bodydims_splot1.py](#)

3) Example 1 : 골반둘레(bii.di) vs. 체중(wgt)

```
from matplotlib import pyplot as plt

f = open("bodydims.csv", 'r')
lines = f.readlines()

del lines[0]

wgt = []
bii = []

for line in lines:
    bii.append(float(line.strip().split(',') [0]))
    wgt.append(float(line.strip().split(',') [5]))

plt.scatter(bii, wgt, marker='.', color='black', label='weight')

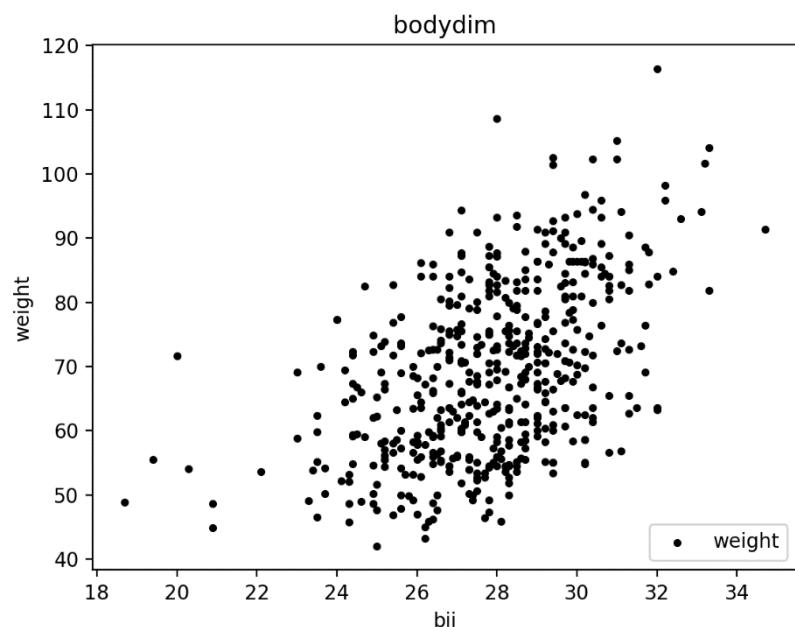
plt.xlabel("bii")
plt.ylabel('weight')
plt.legend(loc=4)
plt.title("bodydim")
plt.show()
f.close()
```

marker : 점 모양

color : 점 색상

label : 범례명

plt.legend(loc = '4') :
 범례표시의 위치 조정,
 'best'라고 놓으면 데이
 터 분포에 따라 가장
 잘 보이는 곳에 배치



10.2 2차원 데이터 분석

분석방법 : 산점도

#강의자료: **bodydims.csv**
[Bodydims_splot1.py](#)

3) Example 2 : 골반둘레(bii.di) vs. { 체중(wgt) & 신장(hgt) }

```
from matplotlib import pyplot as plt

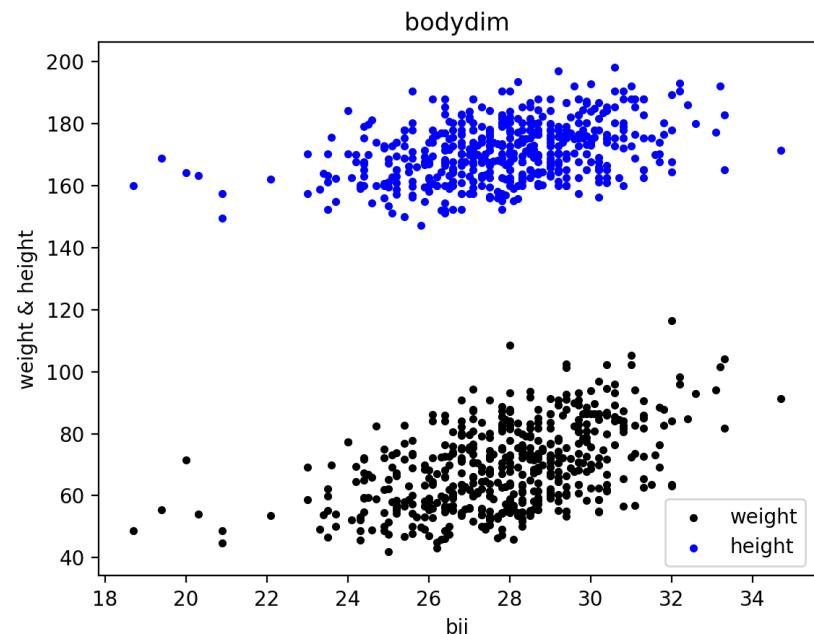
f = open("bodydims.csv", 'r')
lines = f.readlines();

del lines[0]

wgt = []
hgt = []
bii = []

for line in lines:
    bii.append(float(line.strip().split(',') [0]))
    wgt.append(float(line.strip().split(',') [5]))
    hgt.append(float(line.strip().split(',') [6]))

plt.scatter(bii, wgt, marker='.', color='black', label='weight')
plt.scatter(bii, hgt, marker='.', color='blue', label='height')
plt.xlabel("bii")
plt.ylabel('weight & height')
plt.legend(loc=4)
plt.title("bodydim")
plt.show()
f.close()
```



10.2 2차원 데이터 분석

분석방법 : 산점도

#강의자료: **bodydims.csv**
[Bodydims_splot2.py](#)

3) Example 3 : 남성 골반둘레(bii.di) vs. 체중(wgt) vs. 여성 골반둘레(bii.di) vs. 체중(wgt)

```
f = open("bodydims.csv", 'r')
lines = f.readlines()

del lines[0]

wgt_m = []
wgt_f = []

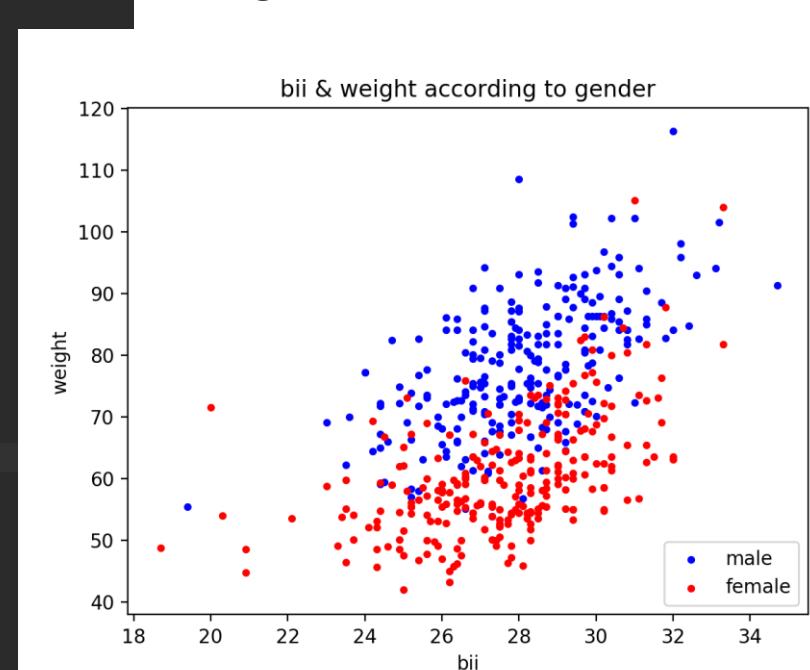
bii_m = []
bii_f = []

for line in lines:
    data = line.strip().split(',')
    if int(data[7]) == 1:
        bii_m.append(float(data[0]))
        wgt_m.append(float(data[5]))

    else:
        bii_f.append(float(data[0]))
        wgt_f.append(float(data[5]))

plt.scatter(bii_m, wgt_m, marker='.', color='b', label='male')
plt.scatter(bii_f, wgt_f, marker='.', color='r', label='female')

plt.legend(loc=.4)
plt.title("bii & weight according to gender")
plt.xlabel('bii')
plt.ylabel('weight')
plt.show()
```



10.2 2차원 데이터 분석

산점도 – 실습 : hospital.txt

예제3. (hospital.txt) 다음은 미국 내 113개의 병원들을 대상으로 입원 기간 동안 환자들이 받는 감염 위험과 관련된 사항들을 조사하였다. 다음은 주요 변수에 대한 설명이다.

변수명	설명
InfctRsk	종속변수. 감염 위험 정도
Stay	설명변수1. 환자들의 평균 입원 기간
Age	설명변수2. 환자들의 평균 나이
Xray	설명변수3. 해당 병원의 X-ray 검진 횟수

- 1) Stay와 InfctRsk 간의 산점도
- 2) Age와 InfctRsk 간의 산점도

10.2 2차원 데이터 분석

산점도 – 실습 : hospital.txt (answer 1)

```
f = open("hospital.txt", 'r')
lines = f.readlines()

inf = []
stay = []
age = []
xray = []

del lines[0]

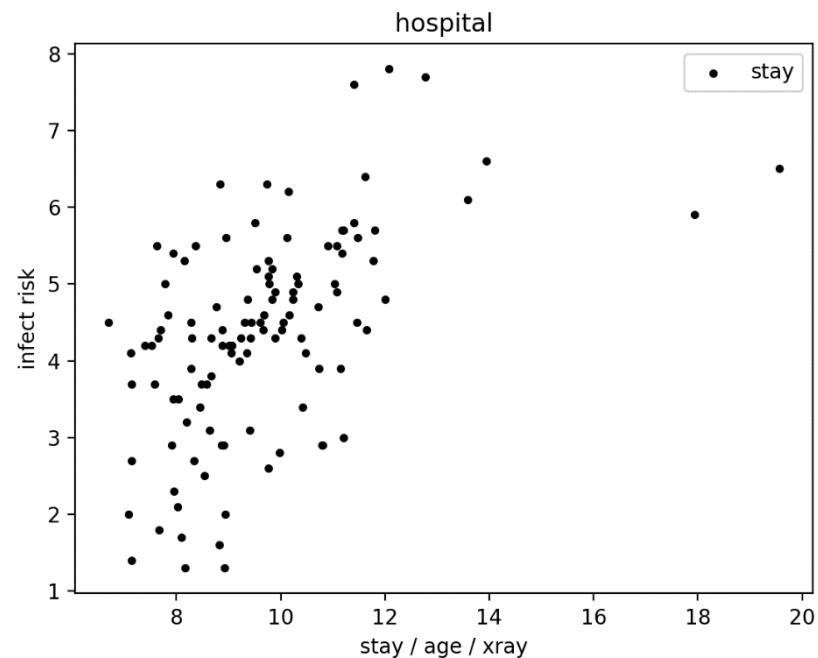
for line in lines:
    temp = line.strip().split('\t')
    inf.append(temp[3])
    stay.append(temp[1])
    age.append(temp[2])
    xray.append(temp[5])

fl_inf = list(map(float, inf))
fl_stay = list(map(float, stay))
fl_age = list(map(float, age))
fl_xray = list(map(float, xray))

print(fl_inf)
print(fl_age)
print(fl_stay)

plt.scatter(fl_stay, fl_inf, marker='.', color='black', label='stay')
plt.scatter(fl_age, fl_inf, marker='.', color='blue', label='age')
plt.scatter(fl_xray, fl_inf, marker='.', color='red', label='xray')

plt.ylabel("infect risk")
plt.xlabel('stay / age / xray')
plt.legend(loc=1)
plt.title("hospital")
plt.show()
```



10.2 2차원 데이터 분석

산점도 – 실습 : hospital.txt (answer 2)

```
f = open("hospital.txt", 'r')
lines = f.readlines()

inf = []
stay = []
age = []
xray = []

del lines[0]

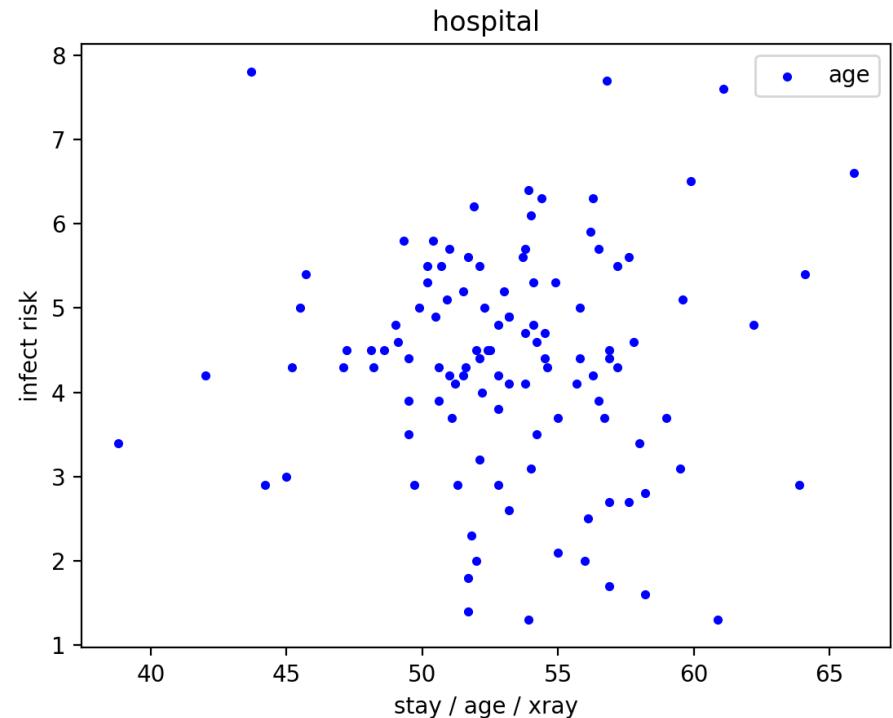
for line in lines:
    temp = line.strip().split('\t')
    inf.append(temp[3])
    stay.append(temp[1])
    age.append(temp[2])
    xray.append(temp[5])

fl_inf = list(map(float, inf))
fl_stay = list(map(float, stay))
fl_age = list(map(float, age))
fl_xray = list(map(float, xray))

print(fl_inf)
print(fl_age)
print(fl_stay)

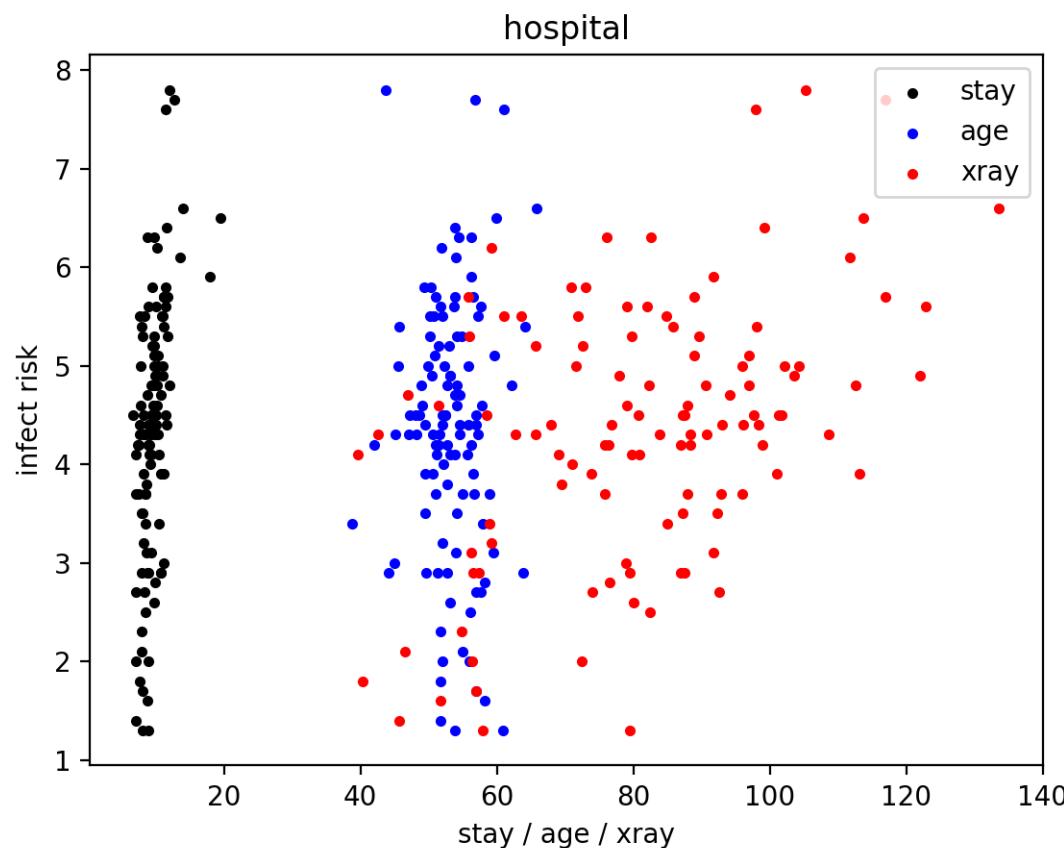
plt.scatter(fl_stay, fl_inf, marker='.', color='black', label='stay')
plt.scatter(fl_age, fl_inf, marker='.', color='blue', label='age')
plt.scatter(fl_xray, fl_inf, marker='.', color='red', label='xray')

plt.ylabel("infect risk")
plt.xlabel('stay / age / xray')
plt.legend(loc=1)
plt.title("hospital")
plt.show()
```



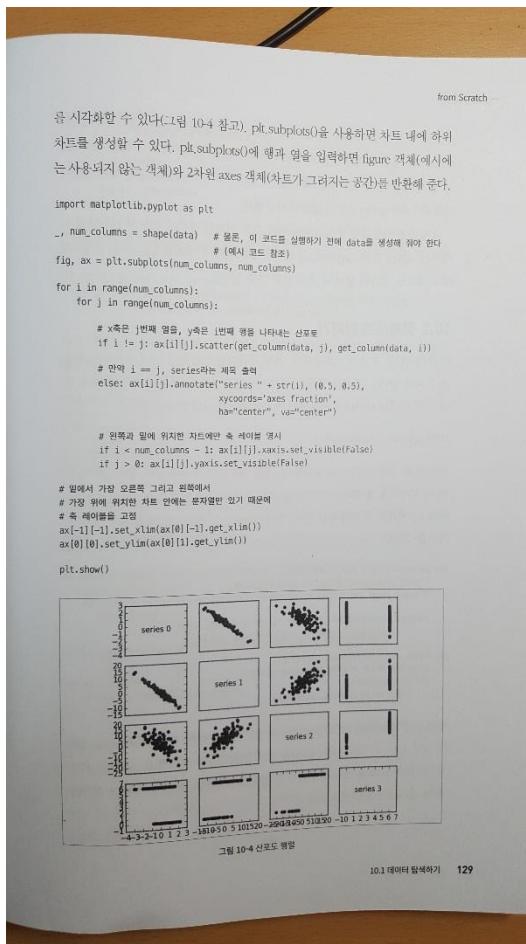
10.2 2차원 데이터 분석

산점도 – 실습 : hospital.txt (*additional)



10.2 다차원 데이터 분석

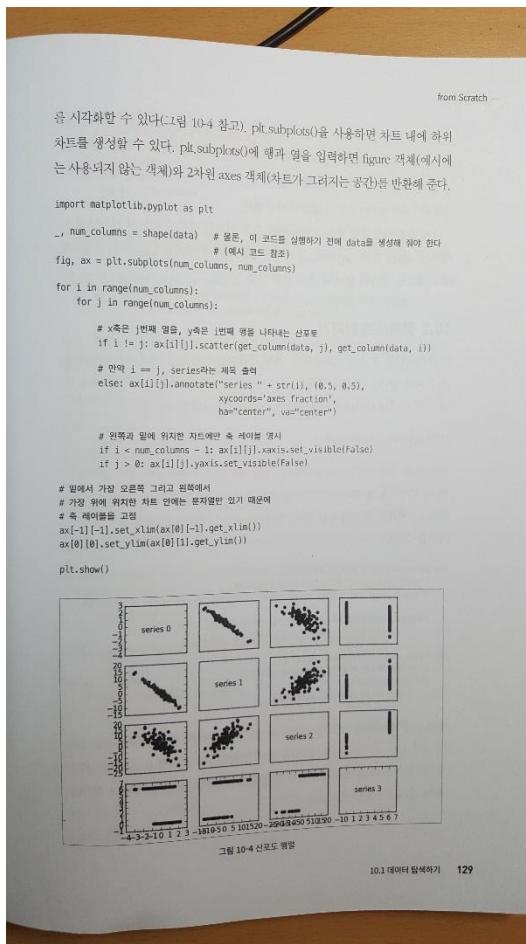
분석방법 1) matplotlib



```
from __future__ import division
from collections import Counter, defaultdict
from functools import partial
from linear_algebra import shape, get_row, get_column, make_matrix, \
    vector_mean, vector_sum, dot, magnitude, vector_subtract, scalar_multiply
from statistics import correlation, standard_deviation, mean
from probability import inverse_normal_cdf
from gradient_descent import maximize_batch
import math, random, csv
import matplotlib.pyplot as plt
import dateutil.parser
```

10.2 다차원 데이터 분석

분석방법 1) matplotlib



```
# first, generate some random data
from scratch

# 시작할 수 있다(그림 10-4 참고). plt.subplots()을 사용하면 차트 내에 하위
# 차트를 생성할 수 있다. plt.subplots()에 행과 열을 입력하면 figure 각체(예시에
#는 사용되지 않는 객체)와 2차원 axes 객체(차트가 그리지는 공간)를 반환해 준다.

import matplotlib.pyplot as plt

_, num_columns = shape(data) # 물론, 이 코드를 실행하기 전에 data를 생성해 줘야 한다
# (예시 코드 참조)
fig, ax = plt.subplots(num_columns, num_columns)

for i in range(num_columns):
    for j in range(num_columns):

        # x축은 j번째 열을, y축은 번째 행을 나타내는 산포도
        if i != j: ax[i][j].scatter(get_column(data, j), get_column(data, i))

        # 만약 i == j, series라는 제목을
        else: ax[i][j].annotate("series " + str(i), (0.5, 0.5),
                               xycoords='axes fraction',
                               ha="center", va="center")

# 원쪽과 밑에 위치한 차트에만 속 레이블 생기
if i < num_columns - 1: ax[i][j].xaxis.set_visible(False)
if j > 0: ax[i][j].yaxis.set_visible(False)

# 일에서 가장 오래된 그리고 편학에서
# 가장 위에 위치한 차트 인덱스 문자열인 있기 때문에
# 즉 예외를 고침
ax[-1][-1].set_xlim(ax[0][-1].get_xlim())
ax[0][0].set_ylim(ax[0][1].get_ylim())

plt.show()
```

random.seed(0)

data = [random_row()

for _ in range(num_points)]

10.2 다차원 데이터 분석

분석방법 1) matplotlib

then plot it

from Scratch

을 시작화할 수 있다(그림 10-4 참고). plt.subplots()을 사용하면 차트 내에 하위 차트를 생성할 수 있다. plt.subplots()에 행과 열을 입력하면 figure 객체(예시에 사용되지 않는 객체)와 2차원 axes 객체(차트가 그려지는 공간)를 반환해 준다.

```
import matplotlib.pyplot as plt
_, num_columns = shape(data) # 물론, 이 코드를 실행하기 전에 data를 생성해 줘야 한다
fig, ax = plt.subplots(num_columns, num_columns)
for i in range(num_columns):
    for j in range(num_columns):
        # x에 i번째 열을, y에 j번쨰 행을 나타내는 산포도
        if i != j: ax[i][j].scatter(get_column(data, j), get_column(data, i))
        # 만약 i == j, series라는 제목을 줘라
        else: ax[i][j].annotate("series " + str(i), (0.5, 0.5),
                               xycoords='axes fraction',
                               ha="center", va="center")
# 원쪽과 밑에 위치한 차트에만 레이블을 표시
if i < num_columns - 1: ax[i][j].xaxis.set_visible(False)
if j > 0: ax[i][j].yaxis.set_visible(False)
# 밑에서 가장 오른쪽 그리고 원쪽에서
# 가장 위에 위치한 차트 인덱스는 문자열인 것 때문에
# 즉 예외를 고침
ax[-1][-1].set_xlim(ax[0][-1].get_xlim())
ax[0][0].set_ylim(ax[0][1].get_ylim())
plt.show()
```

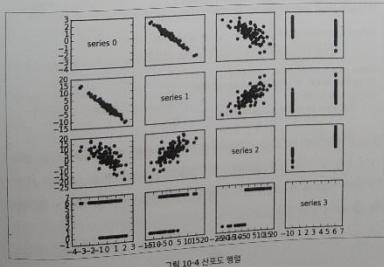


그림 10-4 산포도 행렬

10.1 데이터 활용하기 129

```
, num_columns = shape(data)
fig, ax = plt.subplots(num_columns, num_columns)

for i in range(num_columns):
    for j in range(num_columns):

        # scatter column_j on the x-axis vs column_i on the y-axis
        if i != j: ax[i][j].scatter(get_column(data, j), get_column(data, i))

        # unless i == j, in which case show the series name
        else: ax[i][j].annotate("series " + str(i), (0.5, 0.5),
                               xycoords='axes fraction',
                               ha="center", va="center")

# then hide axis labels except left and bottom charts
if i < num_columns - 1: ax[i][j].xaxis.set_visible(False)
if j > 0: ax[i][j].yaxis.set_visible(False)

# fix the bottom right and top left axis labels, which are wrong because
# their charts only have text in them
ax[-1][-1].set_xlim(ax[0][-1].get_xlim())
ax[0][0].set_ylim(ax[0][1].get_ylim())

plt.show()
```

10.2 다차원 데이터 분석

■ 분석방법 2) seaborn



주로 sns라는 이름으로 import합니다.

import seaborn as sns

`sns.set()` #스타일을 변경한다.
`sns.set_color_codes()`

seaborn에 대해 기초적인 내용 소개하고 있는
한국 웹사이트

<https://datascienceschool.net/view-notebook/4c2d5ff1caab4b21a708cc662137bc65/>

Seaborn 갤러리 (시각화 한번씩 보길 추천!)
<http://seaborn.pydata.org/examples/index.html>

10.2 다차원 데이터 분석

■ 분석방법 2) seaborn

산포도행렬을 그리기

```
import seaborn as sns; sns.set()
import pandas as pd
import matplotlib.pyplot as plt
import os

os.path.abspath(os.path.curdir)
os.chdir(r'Users/kangmina')
csv_file=pd.read_csv("HR_comma_sep.csv", dtype=
{'left':bool,'promotion_last':bool})

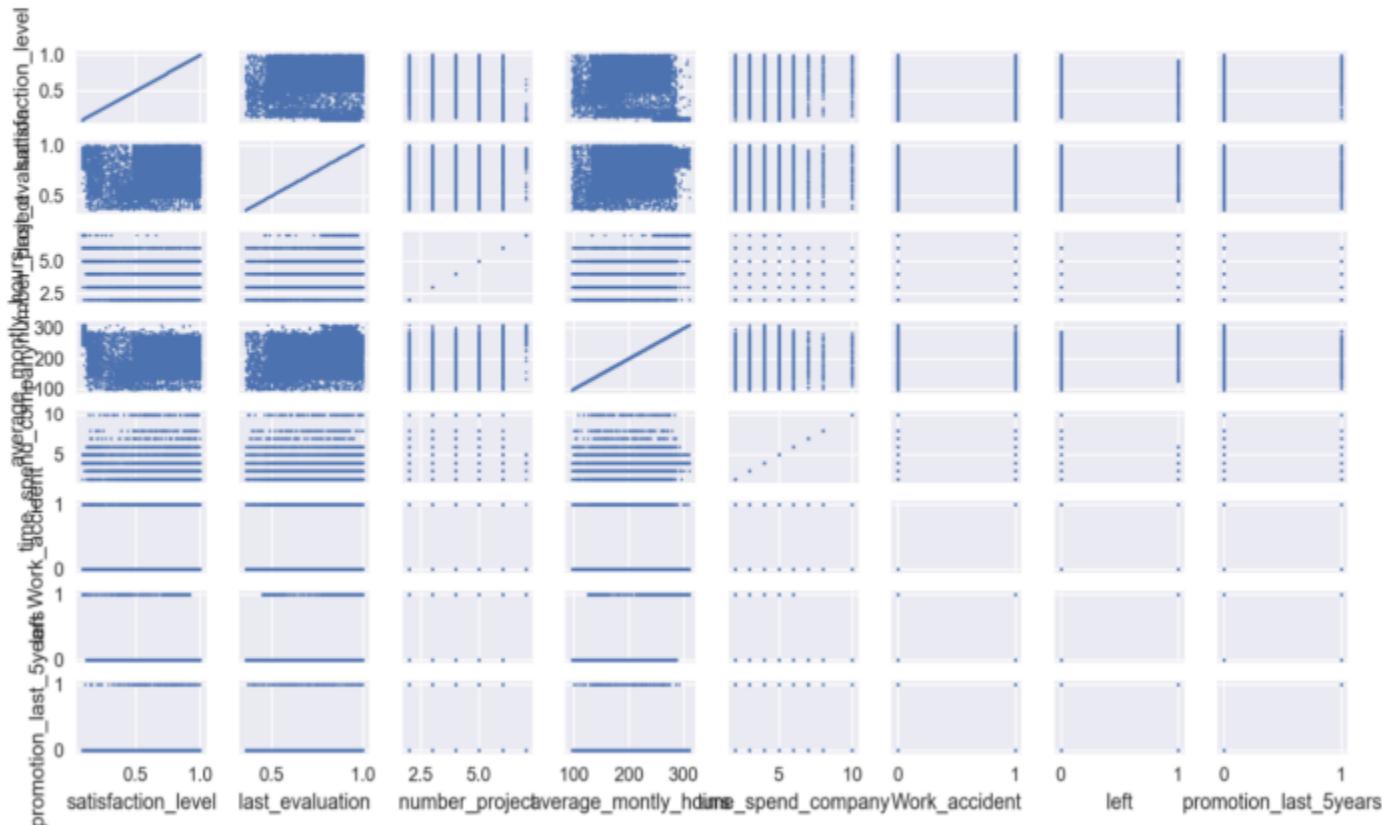
making_pair=sns.PairGrid(csv_file, size=2)
corr_matrix=making_pair.map(plt.scatter, s=1)
plt.show(corr_matrix)
```



10.2 다차원 데이터 분석

어떤 데이터 관계가 보이시나요?

<http://seaborn.pydata.org/generated/seaborn.PairGrid.html>



10.2 다차원 데이터 분석

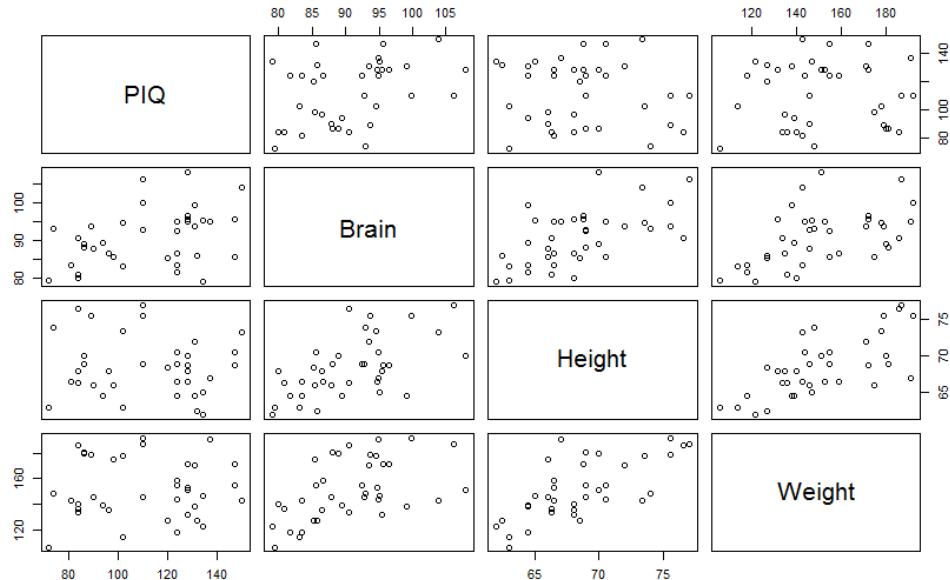
분석방법 3) R studio

<https://www.r-project.org/>

```

3 iqsize <- read.table("iqsize.txt", header = TRUE)
4 plot(iqsize)
5
6
7

```



CONTENTS

3. 데이터 시각화

3.1 상황에 맞는 시각화

3.2 파이썬 시각화 기본

3.3 Tableau

10. 데이터 다루기

10.1 1차원 데이터 분석

10.2 2차원 데이터 분석

10.3 척도 조절

10.4 차원 축소

10.3 척도 조절

척도 조절이란?

“각 차원의 평균을 0, 표준편차를 1로 변환하기”

단위를 제거할 수 있으며,

각 차원은 “평균으로부터 몇 표준편차만큼 떨어져 있는지 ”로 변환된다.

사람	키(cm)	몸무게(kg)	BMI 지수
A	160	65	30
B	170	60	25
C	180	55	20

사람	키	몸무게	BMI
A	-1.0	1.0	1.0
B	0.0	0.0	0.0
C	1.0	-1.0	-1.0

10.3 척도 조절

척도 조절 코딩하기(0)

```
def shape(A):
    num_rows = len(A)
    num_cols = len(A[0]) if A else 0
    return num_rows, num_cols

def make_matrix(num_rows, num_cols, entry_fn):
    return [[entry_fn(i, j) for j in range(num_cols)]
            for i in range(num_rows)]

def get_column(A, j):
    return [A_i[j] for A_i in A]
```

shape(행렬) :
행렬의 행 개수, 열개수 반환

**make_matrix(행개수, 열개수,
특정성분) :**
행과 열의 특정 성분 받아서
새로운 행렬 생성

get_column(행렬, 특정 열) :
특정 열 반환

10.3 척도 조절

척도 조절 코딩하기(1)

```
data = [[160, 65, 30], [170, 60, 25], [180, 55, 20]]
```

1. 각 열의 평균과 표준편차 계산하기

```
def scale(data_matrix):
    num_rows, num_cols = shape(data_matrix)
    means = [mean(get_column(data_matrix, j)) for j in range(num_cols)]
    stdevs = [standard_deviation(get_column(data_matrix, j)) for j in range(num_cols)]
    return means, stdevs
```

2. 각 열의 평균을 0, 표준편차를 1로 변환하기

```
def rescale(data_matrix):
    means, stdevs = scale(data_matrix)

    def rescaled(i, j):
        if stdevs[j] > 0:
            return (data_matrix[i][j] - means[j]) / stdevs[j]
        else:
            return data_matrix[i][j]

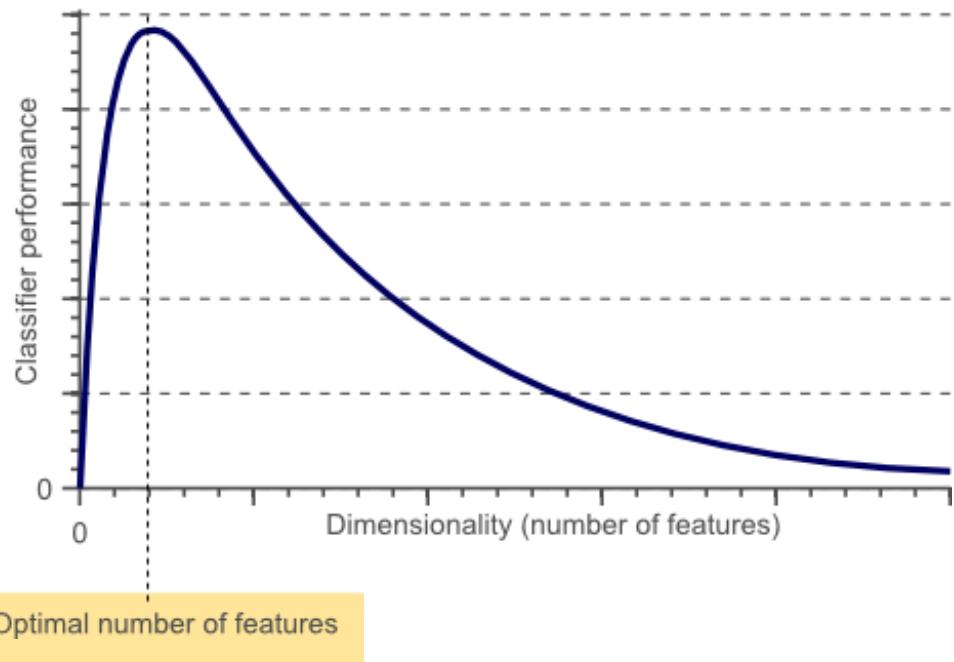
    num_rows, num_cols = shape(data_matrix)
    return make_matrix(num_rows, num_cols, rescaled)
```

10.4 차원 축소

차원의 저주 (Curse of Dimensionality)

- Rechard E Bellman

“차원이 증가하면 필요한 데이터의 양이 증가하고, 그로 인해 신뢰도가 낮아지고, 러닝에 걸리는 시간이 오래 걸리고, 정확도가 크게 감소하고 …”



차원 축소 :

“데이터의 의미를 제대로 표현하는 특징을 추려내는 것”

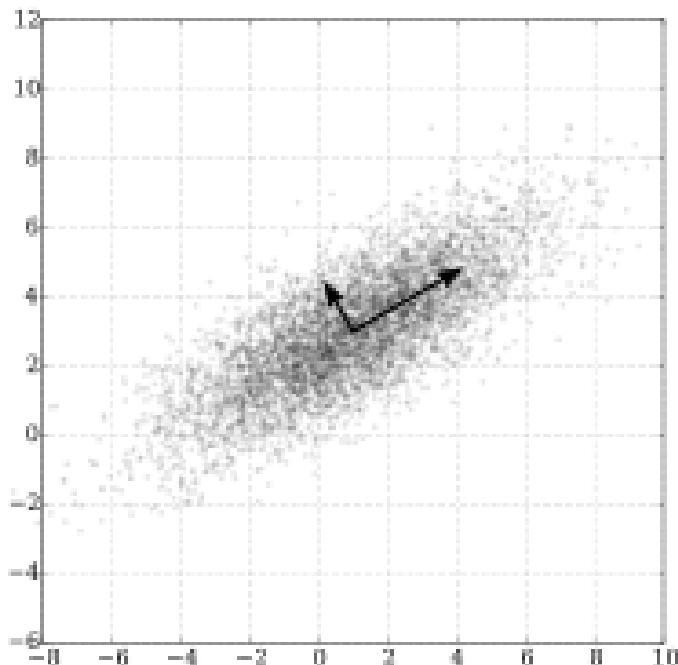
특징(feature)이 너무 많기 때문에 학습이 어렵고 더 좋은 특징만 가지고 사용하고자 함.

10.4 차원 축소

주성분분석(PCA—Principal Component Analysis)

“주성분분석(PCA)은 데이터의 분포를 가장 잘 표현하는 성분 찾기”

1. PCA의 기하학적 이해



주성분: 데이터들의 분산이 가장 큰 방향 벡터
분산: 데이터가 평균에서 떨어진 정도

옆의 그림에서 x, y 축보다 **화살표방향의 두 축**이
이 데이터를 더 잘 표현하는 것을 알 수 있음.

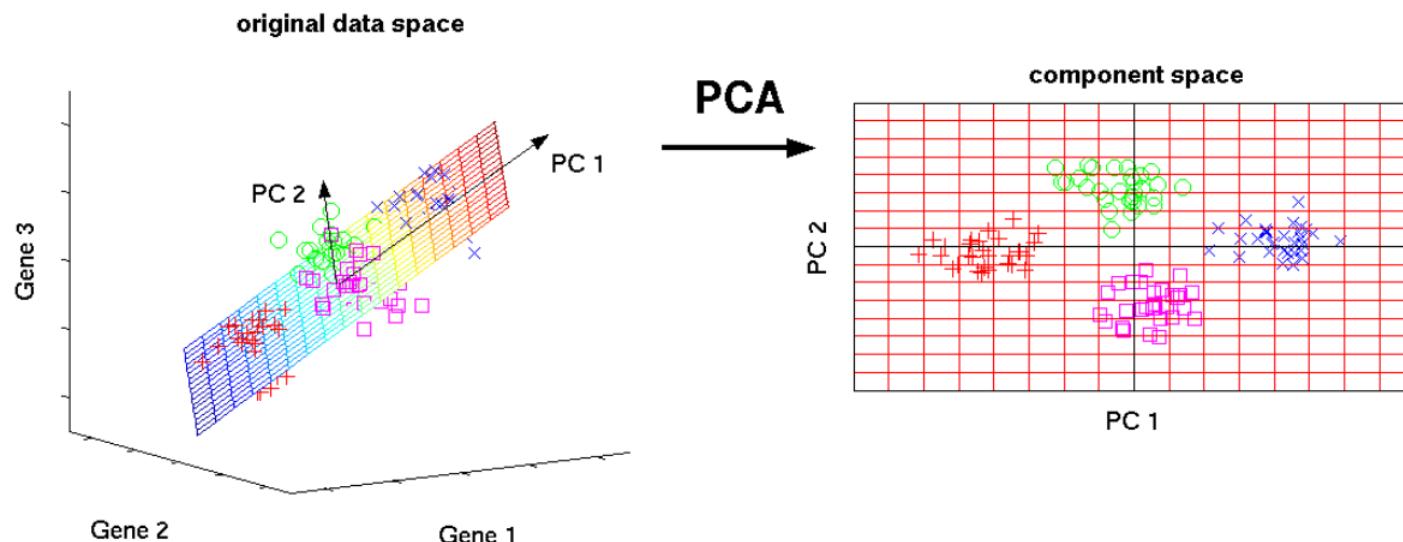
→ PCA를 사용하면 x, y축이 아닌 위의 두 축(제 1 주성분, 제 2 주성분)으로 데이터를 표현하게 됨

10.4 차원 축소

주성분분석(PCA—Principal Component Analysis)

“주성분분석(PCA)은 데이터의 분포를 가장 잘 표현하는 성분 찾기”

1. PCA의 기하학적 이해



여러 방향 중 데이터가 가장 넓게 퍼져있는 방향으로 축을 잡아 표현하는 것
 → 데이터의 **variance**(분산)가 가장 큰 방향으로 **projection**(정사영)시키는 것

Projection은 마치 그림자를 내리는 것과 같다고 생각하면 됨.

10.4 차원 축소

주성분분석(PCA—Principal Component Analysis)

“주성분분석(PCA)은 데이터의 분포를 가장 잘 표현하는 성분 찾기”

2. PCA의 수학적 이해

선형 대수에서 다루는 다양한 개념을 걸壑기만 할 예정입니다.
그 이유는 PCA 뿐만 아니라, SVD(특이값분해), Pseudo-Inverse 등 이후 자주 다루게 될 다양한 데이터 처리 과정 응용 모델의 기초가 되기 때문입니다.

여러 방향 중 데이터가 가장 넓게 퍼져있는 방향으로 축을 잡아 표현하는 것
 → 원래의 데이터 X 에 어떠한 회전행렬 P 를 곱해서 PX 가 최대의 분산들을 갖도록 만들기

최대의 분산 = P 와 X 간의 서로 간의 연관성을 최소로 만들기
 = P 와 X 간의 공분산행렬을 대각행렬로 만들기

선형대수학에서 회전행렬이란 유clidean 공간에서 원점을 중심으로 회전변환을 하게 하는 행렬이다. 2차원 유clidean 공간인 경우에 원점을 중심으로 시계 반대 방향으로 θ 만큼 회전한 회전행렬은 다음과 같다.

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

10.4 차원 축소

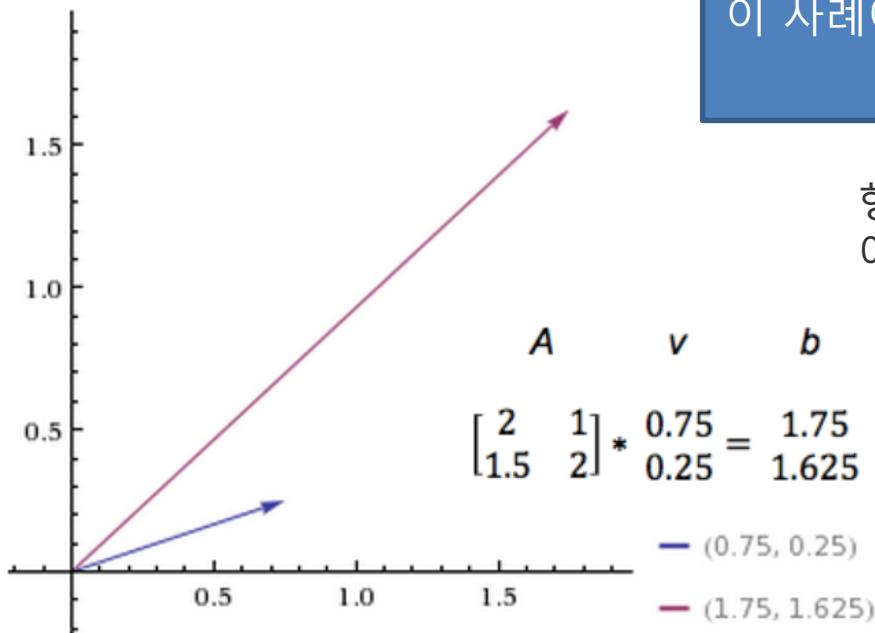
주성분분석(PCA—Principal Component Analysis)

2. PCA의 수학적 이해

<https://deeplearning4j.org/kr/eigenvector>

“어떤 벡터 v 에 행렬 A 를 곱한 결과가 벡터 b ”

Vector plot:



선형변환, 매핑(maps)

이 사례에서는 행렬 A 를 이용하여 벡터 v 를 다른 벡터 b 로 선형변환, 매핑했다고 함

행렬 A 가 짧은 파란색 벡터 v 를 긴 빨간색 벡터 b 에 매핑했는지 옆 그래프에 잘 드러남.

투영, 프로젝션(projection)

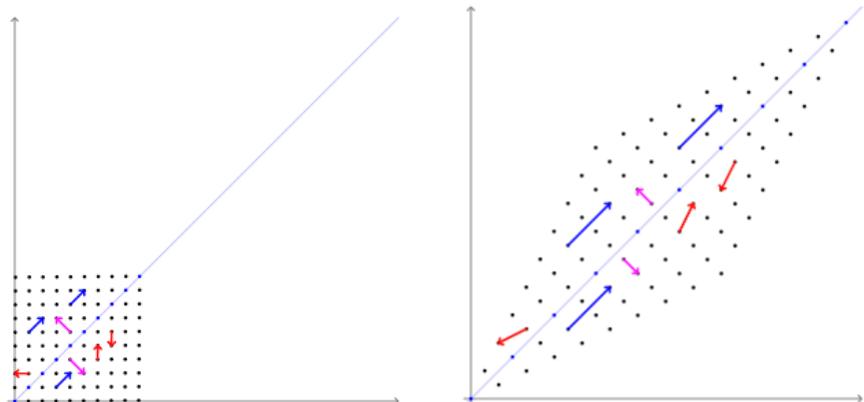
벡터를 행렬 A 에 곱해주면 더 높고 멀리 뻗은 새로운 벡터로 매핑이 될 텐데 이를 하나의 공간이 다른 공간으로 투영하는 것이라 함.

10.4 차원 축소

주성분분석(PCA—Principal Component Analysis)

2. PCA의 수학적 이해

“어떤 벡터 v 에 행렬 A 를 곱한 결과가 벡터 b ”



이 파란색 벡터의 방향이 변하지 않는 이유
 → 행렬 A 가 벡터를 투영하는 방향과 그 방향이 같기 때문.

고유 벡터, 아이겐 벡터(eigen-vector)
 행렬 A 를 선형변환으로 봤을 때, 선형변환 A 에 의한 변환 결과가 자기 자신의 상수배가 되는 0이 아닌 벡터.

아이겐 벡터는 행렬 A 를 곱하더라도 방향이 변하지 않고 그 크기만 변함.

$$Ax = \lambda x$$

행렬 A 를 좌변에 곱하였지만 그 결과가 마치 어떤 숫자를 벡터에 곱한 값과 같이 나오는 식에서의 x 가 **아이겐 벡터**가 됨.

10.4 차원 축소

주성분분석(PCA—Principal Component Analysis)

2. PCA의 수학적 이해

어떤 행렬로 선형 변환을 하는 경우, 행렬의 아이겐 벡터는 그 행렬의 힘의 방향

만일 이 행렬이 공분산 행렬일 경우?

이 공분산 행렬의 힘의 방향 = 데이터가 어떤 방향으로 분산되어 있는지

→ 공분산 값이 큰 순서대로 아이겐벡터를 정렬하면,
중요한 순서대로 주성분을 정렬하는 것과 같음.

고유 값, 아이겐 벨류(eigen-value)

$$Ax = \lambda x$$

행렬 A를 선형변환으로 봤을 때, 선형변환 A에 의한 변환
결과가 자기 자신이 되는 상수배(λ)

우리가 다루는 행렬이 공분산
행렬일 경우 아이겐 벨류는 각
축에 대한 공분산 값

10.4 차원 축소

■ 주성분분석(PCA—Principal Component Analysis)

0. 자료 불러오기 및 정제하기 (벡터화)



사진이 45*40 픽셀이라고 할 때 각 얼굴은 1800차원 공간에서 한 점에 대응함.
→ 1800차원에 존재하는 20개의 점 데이터

1. 공분산 행렬 만들고, eigen-vector(주성분 벡터) 구하기



여기에서의 eigen-vector 나열한 것을 보면 얼굴 형과 같은 전반적인 공통된 요소가
큰 분산을 가지고 있어서 앞에 존재, 뒤로 갈 수록 세부적인 차이 정보

10.4 차원 축소

주성분분석(PCA—Principal Component Analysis)

3. 주성분 개수 골라 기존의 데이터를 새로운 space로 projection 하기



전반부 k 개의 주성분 벡터들만을 가지고 원래 데이터를 표현하면 노이즈가 제거된 데이터 가질 수 있다.

→ k 의 개수가 작아질 수록 (작은 개수의 주성분을 고를 수록) 고유의 얼굴 특성만 남고 고유 얼굴 특성은 사라짐을 알 수 있음

10.4 차원 축소

■ PCA 코딩하기 (0)

0. 자료 불러오기 및 정제하기

```
import pandas as pd

df = pd.read_csv(
    filepath_or_buffer='https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',
    header=None,
    sep=',')

df.columns=['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid', 'class']

# split data table into data X and class labels y

X = df.ix[:,0:4].values
y = df.ix[:,4].values

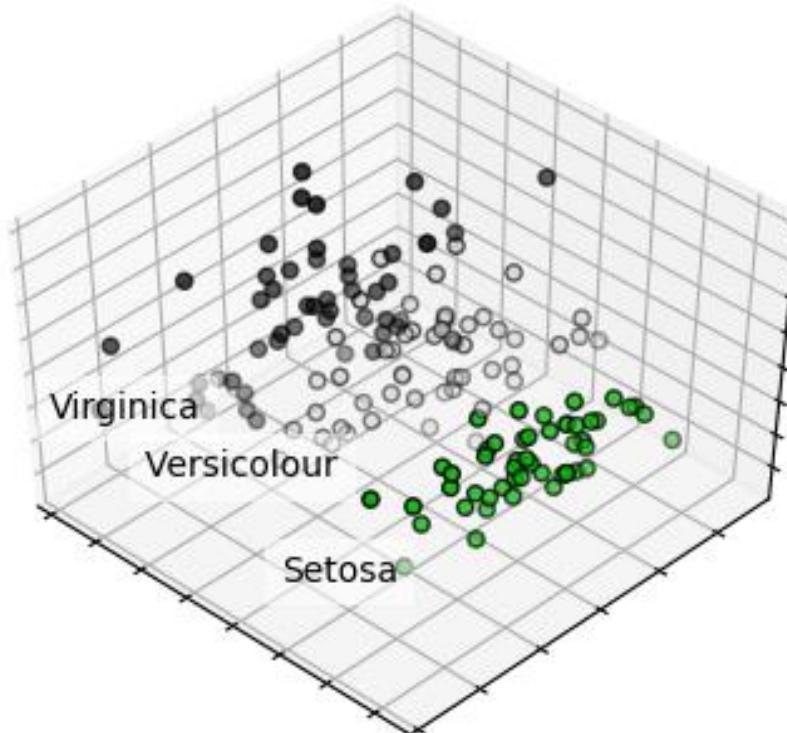
#standardizing
from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X)
```

Scikit-learn의 StandardScaler :
Transforming data such that its distribution will have a mean value 0 and standard deviation of 1.

10.4 차원 축소

■ PCA 코딩하기 (0)

0. 자료 불러오기 및 정제하기



```
from mpl_toolkits.mplot3d import Axes3D
```

10.4 차원 축소

■ PCA 코딩하기 (1)

1. 공분산 행렬 구하기

```
import numpy as np
mean_vec = np.mean(X_std, axis=0)
cov_mat = (X_std - mean_vec).T.dot((X_std - mean_vec)) / (X_std.shape[0]-1)
print('Covariance matrix \n%s' %cov_mat)
```

2. eigen-vector, eigen-value 구하기

```
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)

Covariance matrix
[[ 1.00671141 -0.11010327  0.87760486  0.82344326]
 [-0.11010327  1.00671141 -0.42333835 -0.358937]
 [ 0.87760486 -0.42333835  1.00671141  0.96921855]
 [ 0.82344326 -0.358937   0.96921855  1.00671141]]

Eigenvectors
[[ 0.52237162 -0.37231836 -0.72101681  0.26199559]
 [-0.26335492 -0.92555649  0.24203288 -0.12413481]
 [ 0.58125401 -0.02109478  0.14089226 -0.80115427]
 [ 0.56561105 -0.06541577  0.6338014   0.52354627]]

Eigenvalues
[ 2.93035378  0.92740362  0.14834223  0.02074601]
```

10.4 차원 축소

■ PCA 코딩하기 (2)

3. 주성분 개수 고르기 (위 사례에서는 2가지로 하기로 함)

```
# Make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eig_pairs.sort()
eig_pairs.reverse()

# Visually confirm that the list is correctly sorted by decreasing eigenvalues
print('Eigenvalues in descending order:')
for i in eig_pairs:
    print(i[0])

#construct our d×kd×k-dimensional eigenvector matrix Ψ
matrix_w = np.hstack((eig_pairs[0][1].reshape(4,1),
                      eig_pairs[1][1].reshape(4,1)))

print('Matrix Ψ:\n', matrix_w)
```

Matrix Ψ:

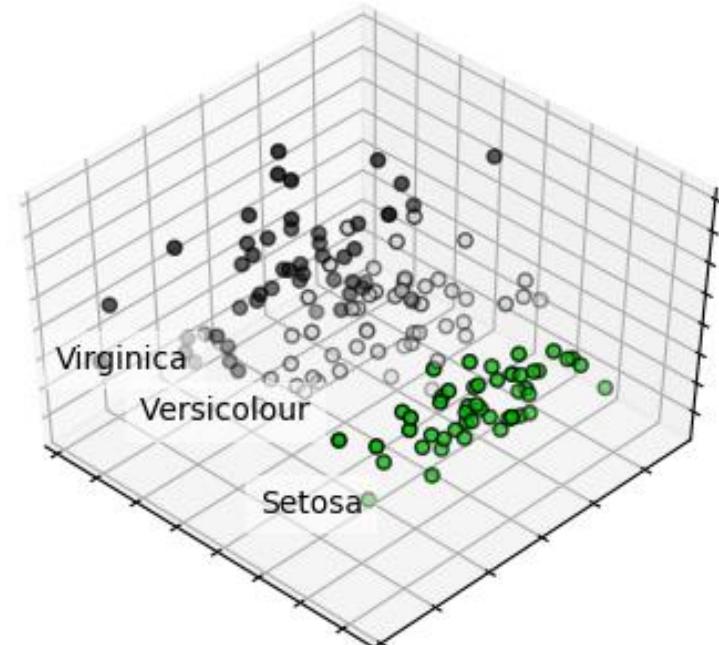
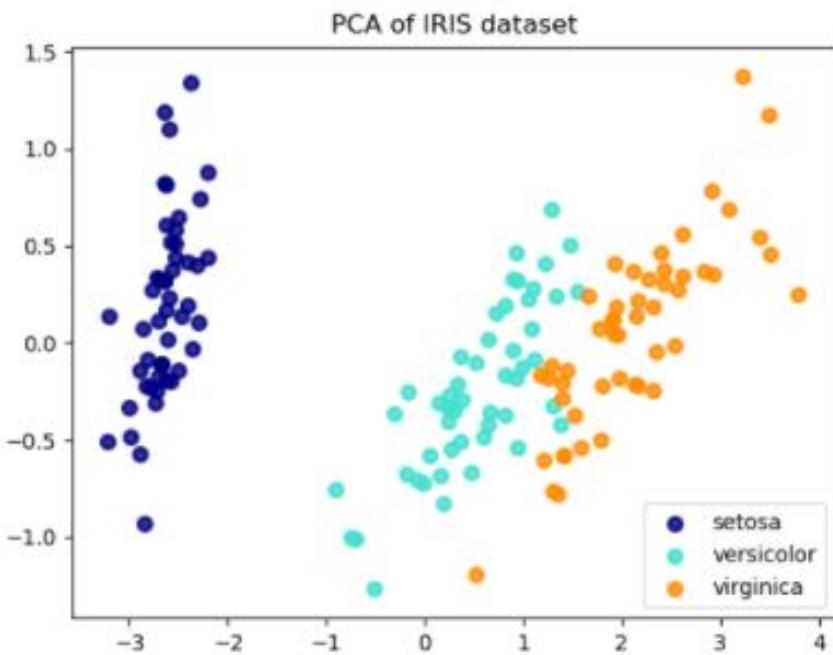
$$\begin{bmatrix} 0.52237162 & -0.37231836 \\ -0.26335492 & -0.92555649 \\ 0.58125401 & -0.02109478 \\ 0.56561105 & -0.06541577 \end{bmatrix}$$

10.4 차원 축소

■ PCA 코딩하기 (3)

4. 새로운 space에다가 data projection 하기

```
Y = X_std.dot(matrix_w)
```



10.4 차원 축소

PCA 코딩하기 (4)

사실은 scikit-learn 5줄이면 된다고 한다.

```
import pandas as pd

df = pd.read_csv(
    filepath_or_buffer=#
        'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',
    header=None,
    sep=',')

df.columns=['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid', 'class']

# split data table into data X and class labels y

X = df.ix[:,0:4].values
y = df.ix[:,4].values

#standardizing

from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X)

# PCA
from sklearn.decomposition import PCA as sklearnPCA
sklearn_pca = sklearnPCA(n_components=2)
Y_sklearn = sklearn_pca.fit_transform(X_std)

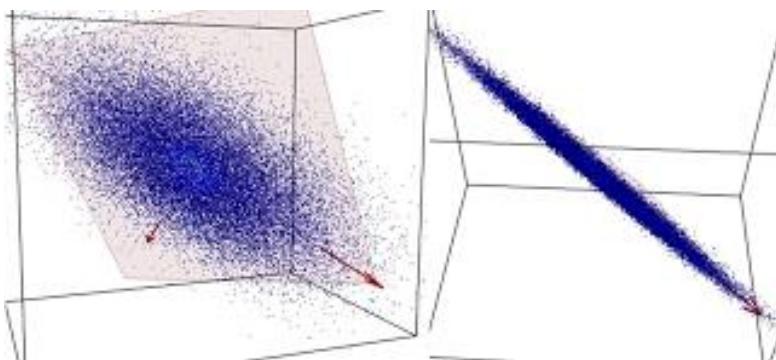
print(Y_sklearn)
```

10.4 차원 축소

PCA, 항상 좋은가?

PCA의 기본 가정을 토대로 PCA 의 한계 살펴보기

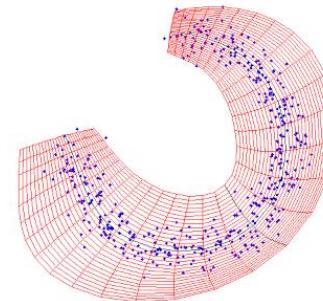
- 데이터는 **직선의 basis**를 가지고 있음.



PCA 를 적용하기 쉬운 직선의 basis 가진 데이터.

기저, Basis

벡터 공간 V 의 벡터들이 선형 독립이면서 벡터 공간 V 전체를 생성할 수 있는 벡터들의 집합



다시 말해서 2차원이라도 달팽이 모양 등 직선이 아닌 basis는 취급하지 않음.

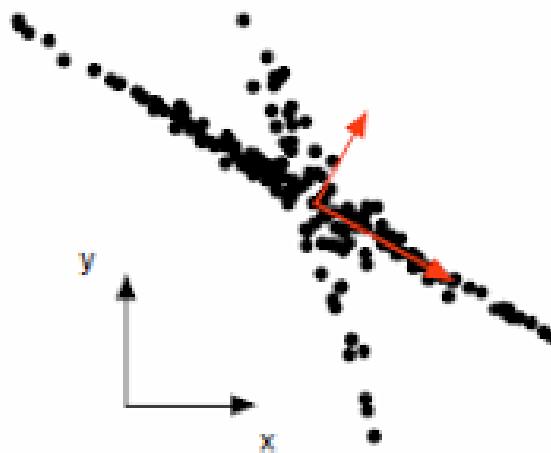
10.4 차원 축소

PCA, 항상 좋은가?

PCA의 기본 가정을 토대로 PCA 의 한계 살펴보기

2. 큰 분산을 갖는 방향이 중요한 정보를 담고 있다고 가정함.

3. 찾은 주축(principal component)들은 서로 직교한다고 가정함..



기계학습을 할 때 다양한 데이터를 마구 넣고 PCA, 인공신경망 등으로 처리를 기대할 것이 아니라, 데이터를 수집할 때부터 무엇을 수집해야 원하는 목표를 달성할 수 있는지 잘 생각해 보는 것이 선행되어야 함.

Reference

<http://darkpgmr.tistory.com/110>

<https://medium.com/@mldevhong/mathpresso-%EB%A8%B8%EC%8B%A0-%EB%9F%AC%EB%8B%9D-%EC%8A%A4%ED%84%B0%EB%94%94-15-%EC%B0%A8%EC%9B%90-%EC%B6%95%EC%86%8C-dimensionality-reduction-76b13460506f>

<http://t-robotics.blogspot.kr/2014/10/pca-principal-component-analysis.html#.WgEUuVu0Mr9>

<http://destrudo.tistory.com/15>

<https://plot.ly/ipython-notebooks/principal-component-analysis/>

<http://t-robotics.blogspot.kr/2015/12/pca.html#.Wg1C0Fu0Mr8>

3&10 Quest



데이터를 가지고 온 뒤 화면을 분할하여(4개 이상)
선, 막대, 히스토그램, 산점도 4가지를 최대한 활용하여
데이터 시각화를 해주세요.

(데이터 시각화를 연습하는 것이기 때문에 문맥은 중요 X)

THANK YOU !