데이터 처리 프로그래밍

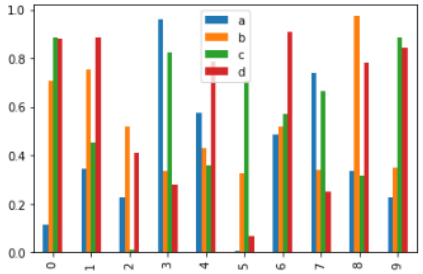
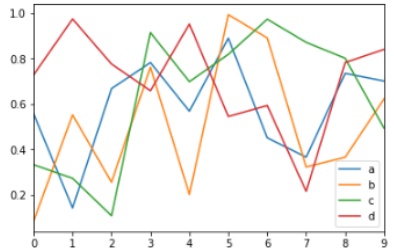
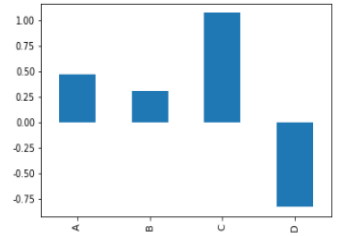
박소현

여섯 번째 주제:

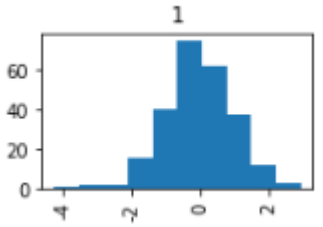
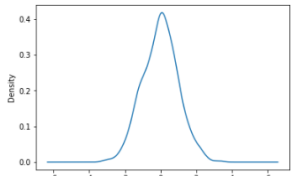
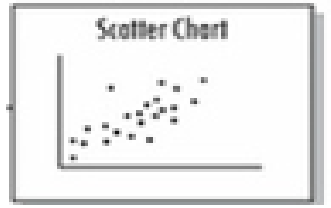
데이터 시각화

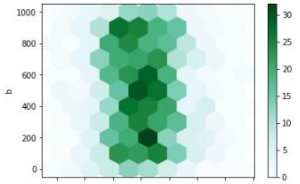
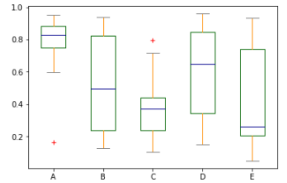
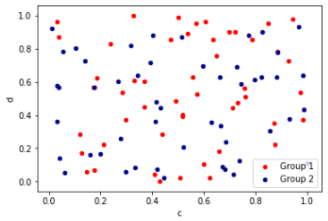
용도별 그래프 종류

비교(comparison) :
막대그래프, (꺾은)선그래프

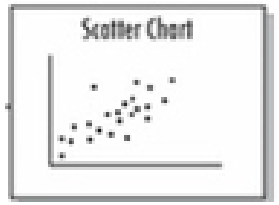




분포(distribution) :
히스토그램, 산포도(scatter),
열지도(hitmap), 수염상자

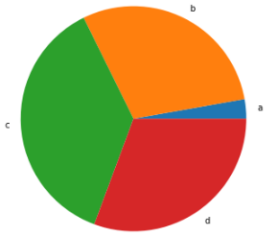
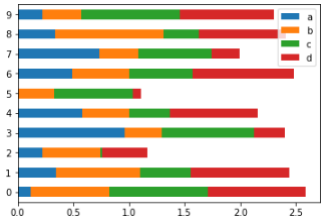
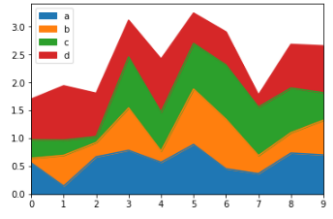




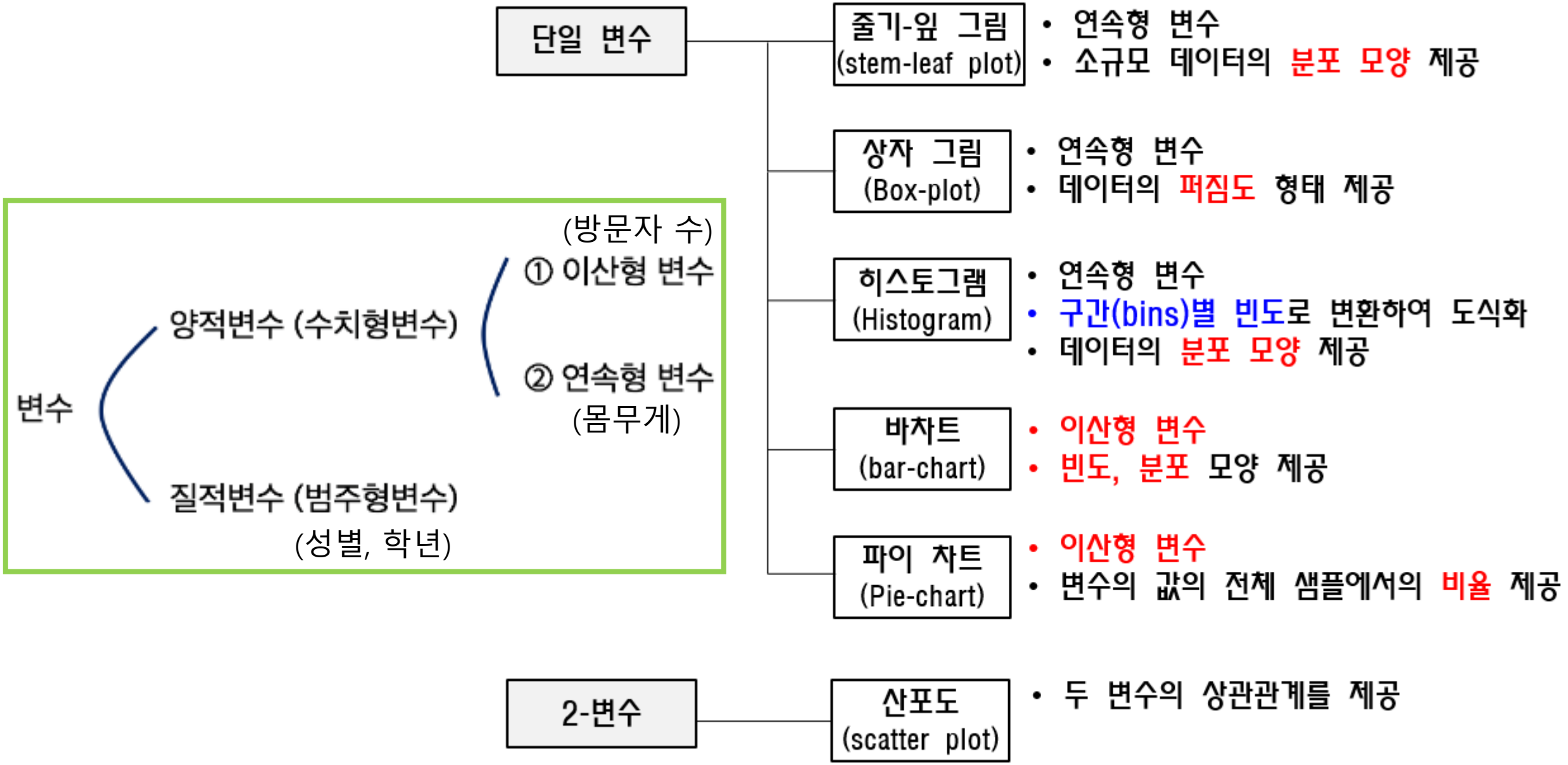




관계(relationship) : 산점도(scatter)



구성(composition) : 파이차트, 누적막대, 누적 area


변수
양적변수 (수치형변수)
① 이산형 변수
(방문자 수)
② 연속형 변수
(몸무게)
질적변수 (범주형변수)
(성별, 학년)
단일 변수
줄기-잎 그림
(stem-leaf plot)
• 연속형 변수
• 소규모 데이터의 분포 모양 제공
상자 그림
(Box-plot)
• 연속형 변수
• 데이터의 퍼짐도 형태 제공
히스토그램
(Histogram)
• 연속형 변수
• 구간(bins)별 빈도로 변환하여 도식화
• 데이터의 분포 모양 제공
바차트
(bar-chart)
• 이산형 변수
• 빈도, 분포 모양 제공
파이 차트
(Pie-chart)
• 이산형 변수
• 변수의 값의 전체 샘플에서의 비율 제공
2-변수
산포도
(scatter plot)
• 두 변수의 상관관계를 제공

그래프 종류 고를 때 Tip - seaborn 라이브러리의 코드

막대그래프
- countplot : x축 애들 각각의 개수
- barplot: y축의 높이가 주 관심사 일때, y축이 연속 데이터일땐 barplot이 더 나음, 편차도 표시 됨, 특이값 등의 판단에 도움(검정 선)

선그래프
- lineplot : x축에 순서의 개념(시간 흐름 등) 있으면 pointplot, 아니라면 barplot
- pointplot : y축의 평균과 편차

산점도
- lmplot(엘엠플롯) : 좌표데이터에 점을 찍는 개념, 추세선(회귀선)도 나옴

분포도
- distplot: 분포 모양 파악(정규분포모양, 왜도, 첨도..), 가우시안(Gaussian) 개수 파악 등이 가능

수염상자
- 데이터의 5개 정보 표현, 최소/ 1사분위/ 2사분위(중앙값)/ 3사분위/최대 값/특이값

그래프 그리기 전 세팅할 것들

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```
pandas처럼 시각화 때 기본으로 import

```
import seaborn as sns
```
seaborn 라이브러리로 그래프 그리고 싶을 때

```
%matplotlib inline
```
쥬피터 노트북에서 그래프 결과 안보일 때
(반드시 그래프 코드 등장하기 전 윗줄에 써야 함)

한글폰트 사용 시 mpl의 그래프에서 마이너스 폰트 깨질 경우

```
mpl.rcParams['axes.unicode_minus'] = False
```

그래프에서 한글이 깨질 경우

```
from matplotlib import font_manager, rc

변수명 = font_manager.FontProperties(fname='C:/Windows/Fonts/ 폰드명 .ttf').get_name()
rc('font', family= 변수명 )
```

```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

mpl.rcParams['axes.unicode_minus'] = False
```

```
df = pd.DataFrame(np.random.rand(6, 3),
                  columns=['a', 'b', 'c'])
df['d'] = pd.Series([1, 1, 1, 0.5, 0.5, 0.3])
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

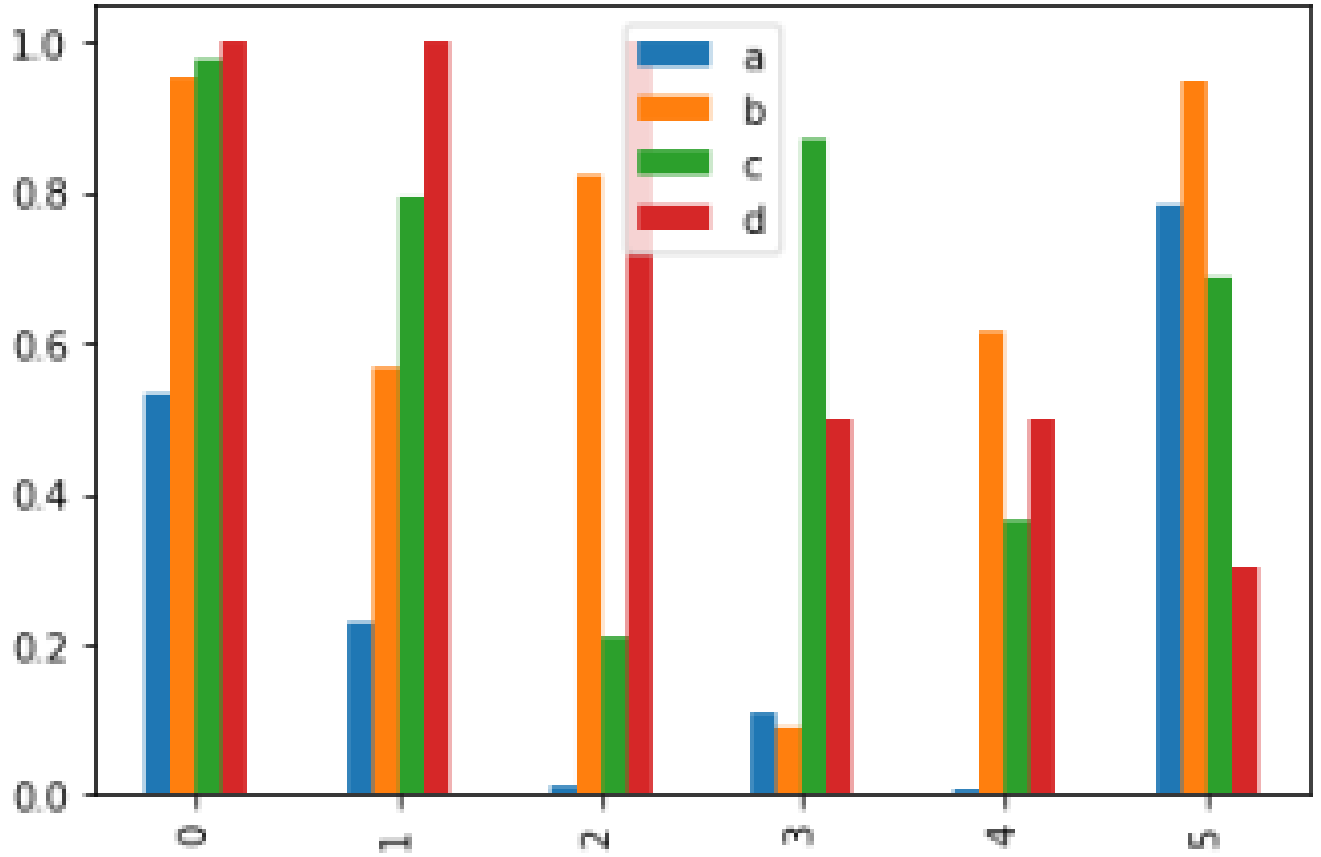
1. 막대그래프

```
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

```
df.plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24dE
```

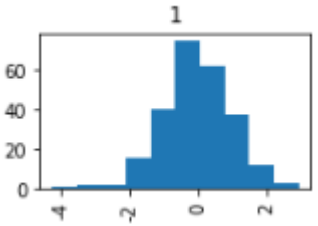


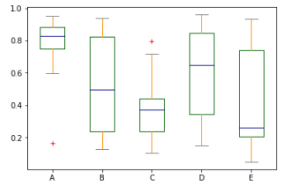
```
df.plot?
```

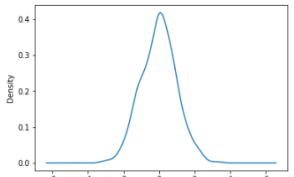
판다스에 내장되어 있는 그래프 기능 plot

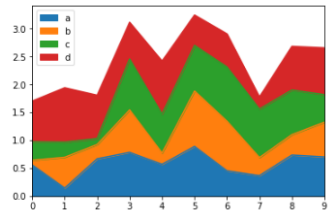
kind : str

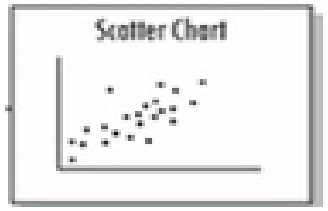
```
df.plot(kind = 'bar')
```

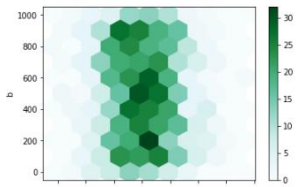












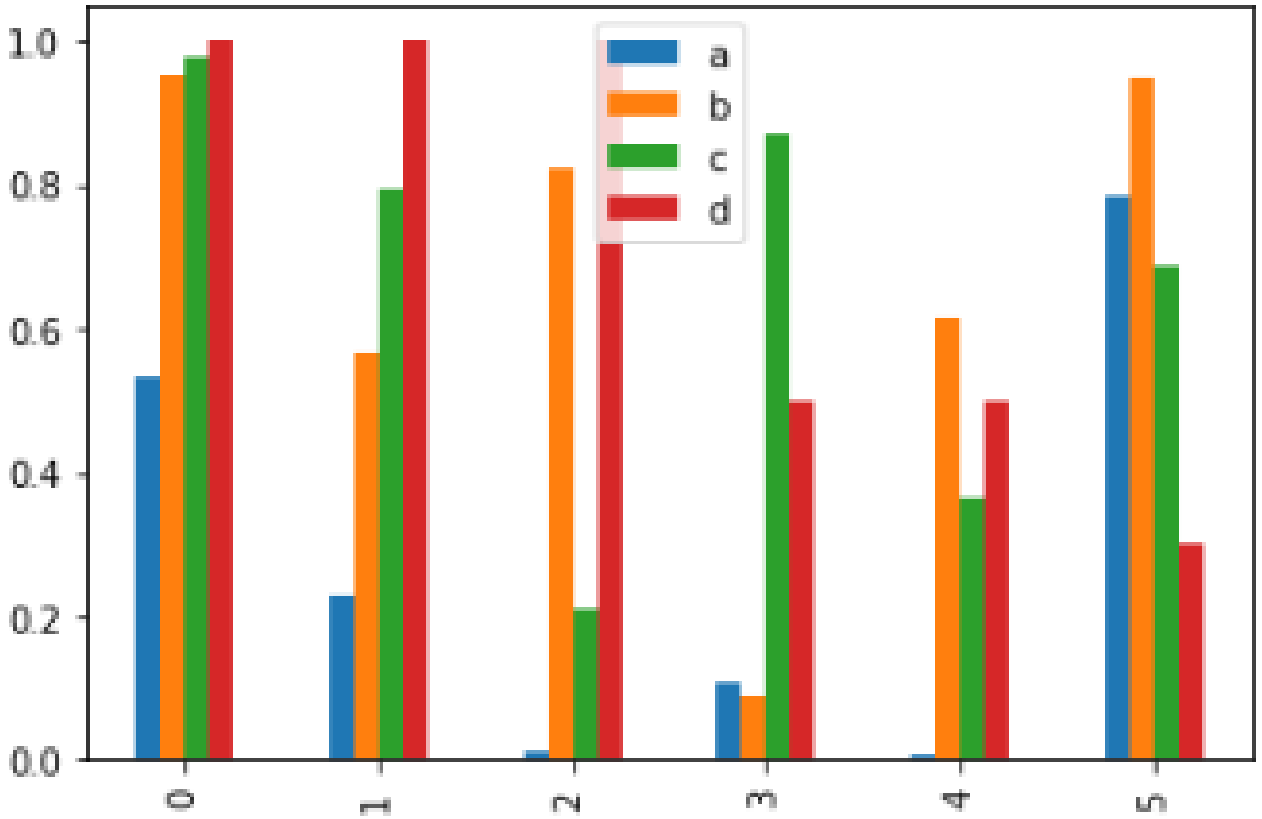
- 'line' : line plot (default)
- 'bar' : vertical bar plot
- 'barh' : horizontal bar plot : bar에서 x, y축 바꾸어 줌
- 'hist' : histogram
- 'box' : boxplot
- 'kde' : Kernel Density Estimation plot
- 'density' : same as 'kde' = kde
- 'area' : area plot
- 'pie' : pie plot
- 'scatter' : scatter plot
- 'hexbin' : hexbin plot

1. 막대그래프 – 동일 표기법

```
df.plot.bar()
```

```
df.plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24dE
```



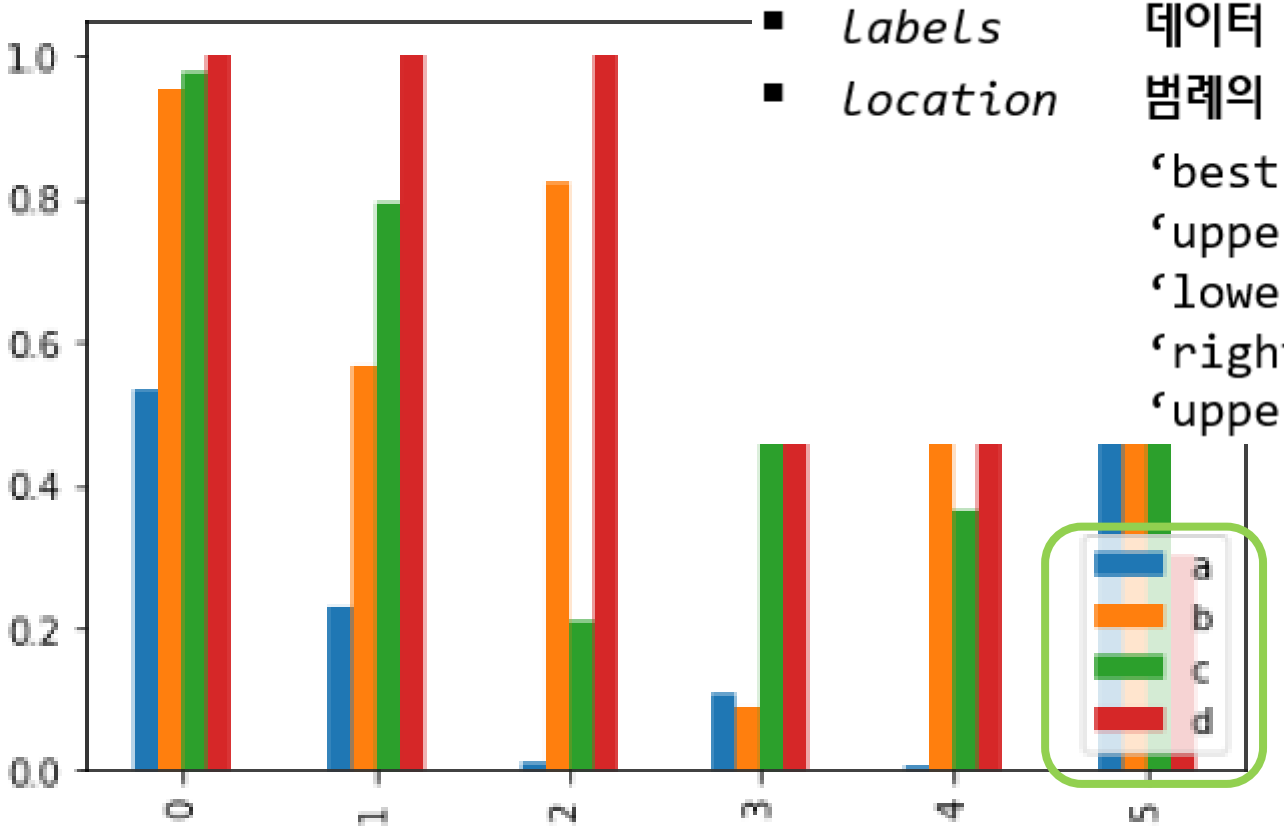
범례 위치가
거슬려 ㅠㅠ

```
df.plot.bar()
plt.legend(loc = 'lower right')
```

```
plt.legend(('토끼', '시라소니', '당근'), loc='upper right')
```

```
<matplotlib.legend.Legend at
```

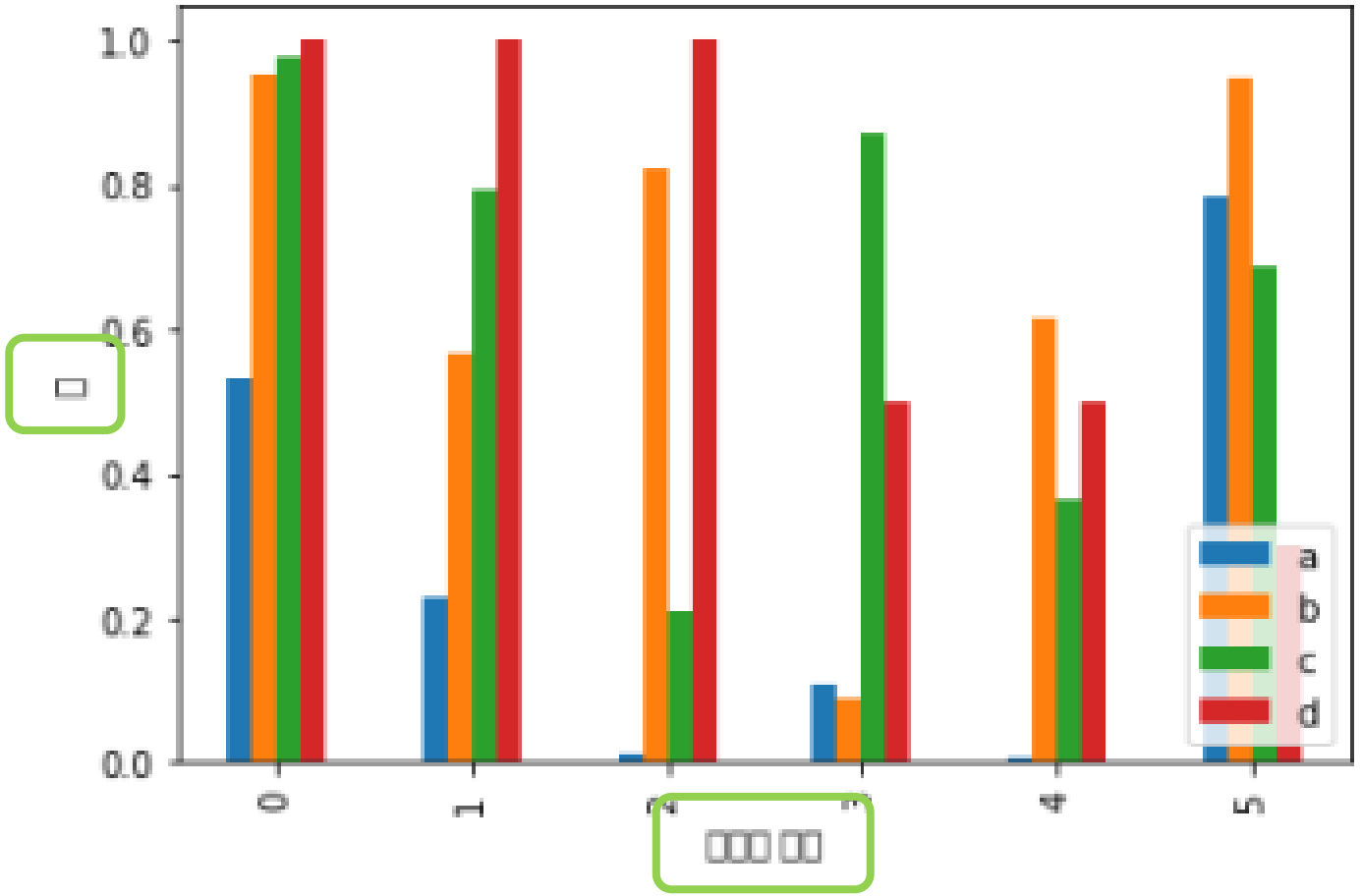
```
legend(labels, loc=location)
```



- *labels* 데이터 구분을 하기 위한 레이블 튜플(리스트)
- *location* 범례의 위치값

‘best’,
‘upper right’, ‘upper left’, ‘upper center’,
‘lower left’, ‘lower right’,
‘right’, ‘center right’, ‘center left’,
‘upper center’, ‘center’

```
df.plot.bar()
plt.legend(loc = 'lower right')
plt.xlabel('인덱스 번호')
plt.ylabel('값')
```



x축, y축 이름 지정

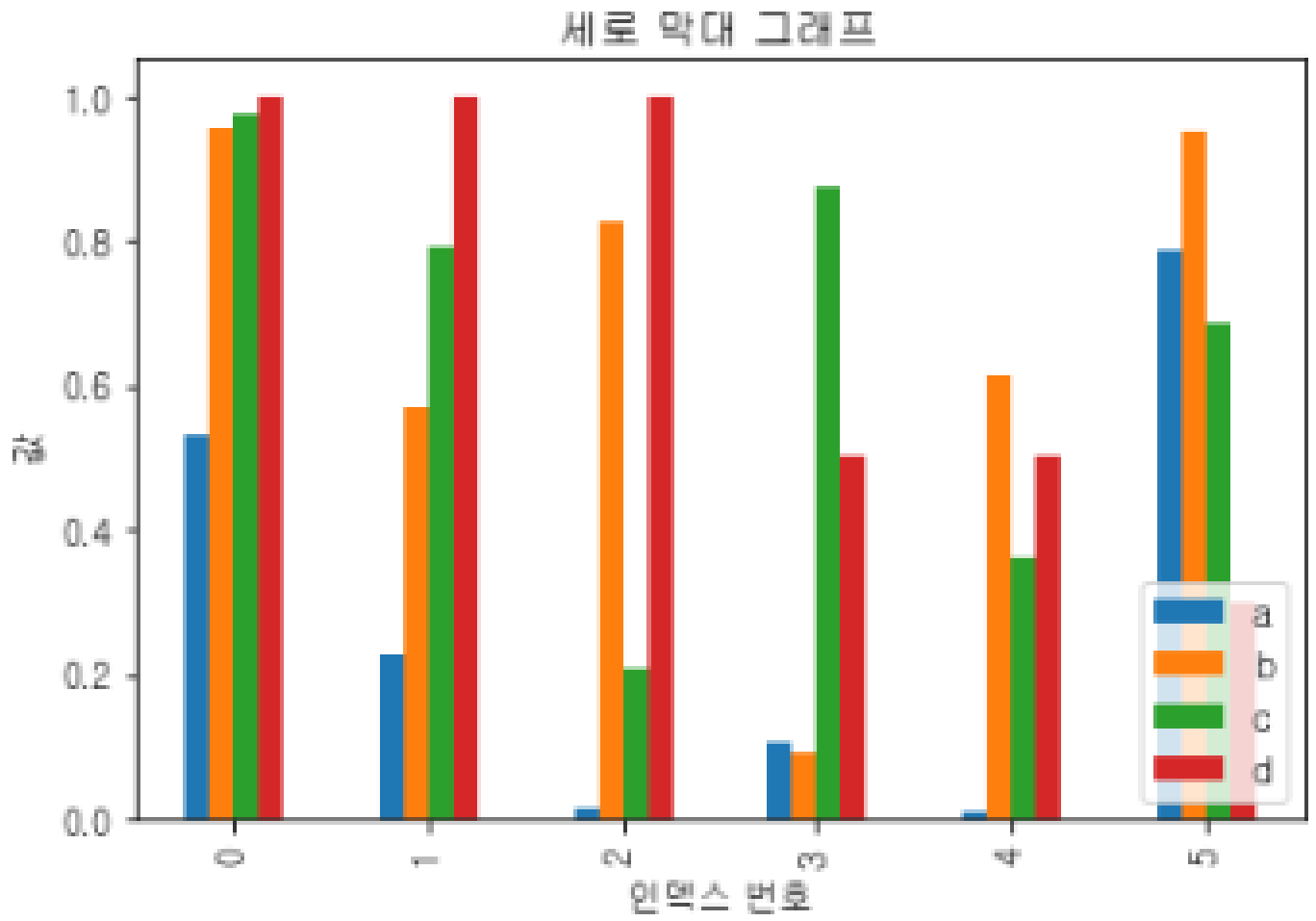
```
xlabel(label)
ylabel(label)
```

- *label* 축이름으로 사용할 문자열

```
plt.xlabel('키')
plt.ylabel('몸무게')
```

```
from matplotlib import font_manager, rc

f_name = font_manager.FontProperties(fname='C:/Windows/Fonts/NGULIM.ttf').get_name()
rc('font', family=f_name)
```

```
df.plot.bar()
plt.legend(loc = 'lower right')
plt.xlabel('인덱스 번호')
plt.ylabel('값')
plt.title('세로 막대 그래프')
```



그래프의 x축, y축 데이터의 범위 지정

```
xlim(xmin, xmax)
ylim(ymin, ymax)
```

```
plt.xlim(-2, 2)
plt.ylim(-10, -10)
```

- 그래프의 x축(y축) 하한값과 상한값을 *xmin*(*ymin*), *xmax*(*ymax*)로 지정

pandas 그래프 - 예

- https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html#visualization

그래프를 별도의 파일로 저장 하려면 plt.savefig('이름.pdf')

```
plt.savefig('samplefig.pdf')
```

그래프를 쥬피터에서 별도의 창에 그리려면 `%matplotlib qt5`

https://seaborn.pydata.org/examples/index.html

https://matplotlib.org/gallery/index.html

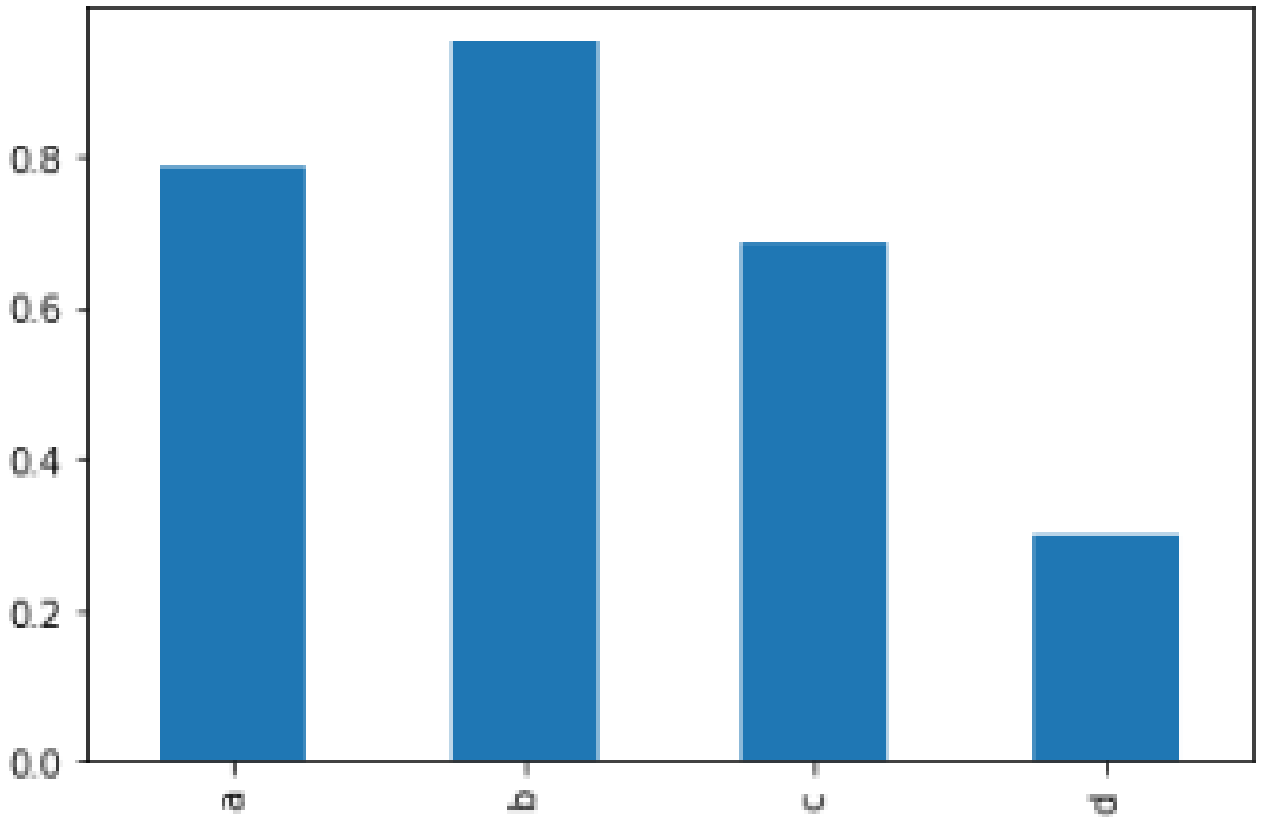
1. 막대그래프 – 행 하나만 그리기

```
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

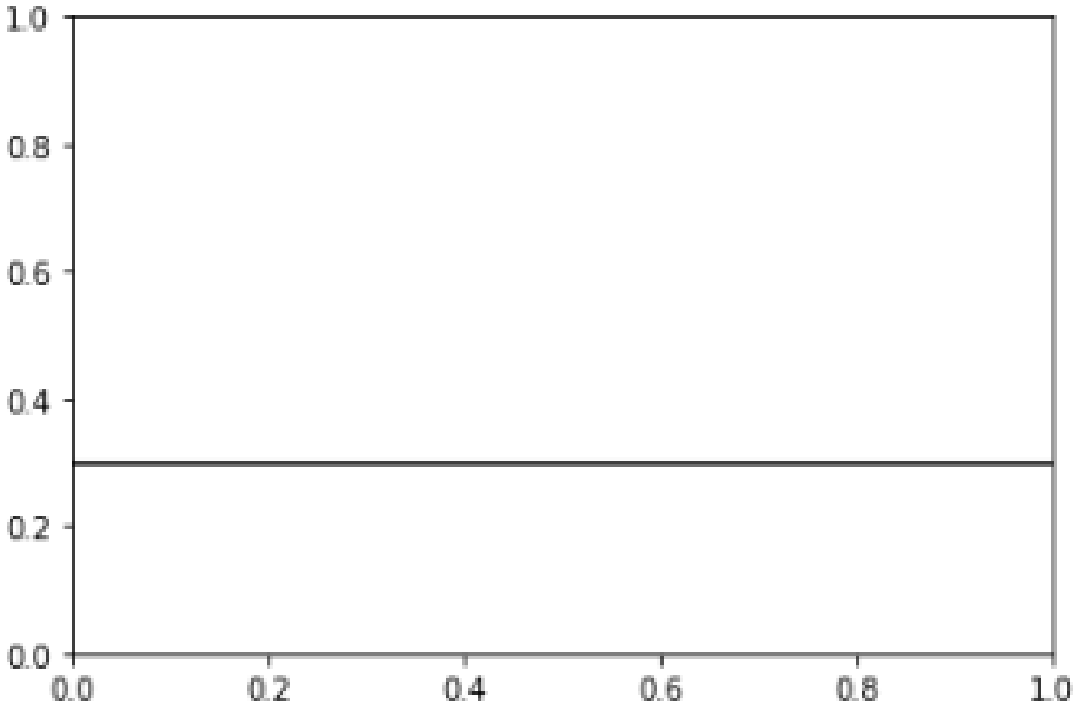
```
df.iloc[5].plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d
```



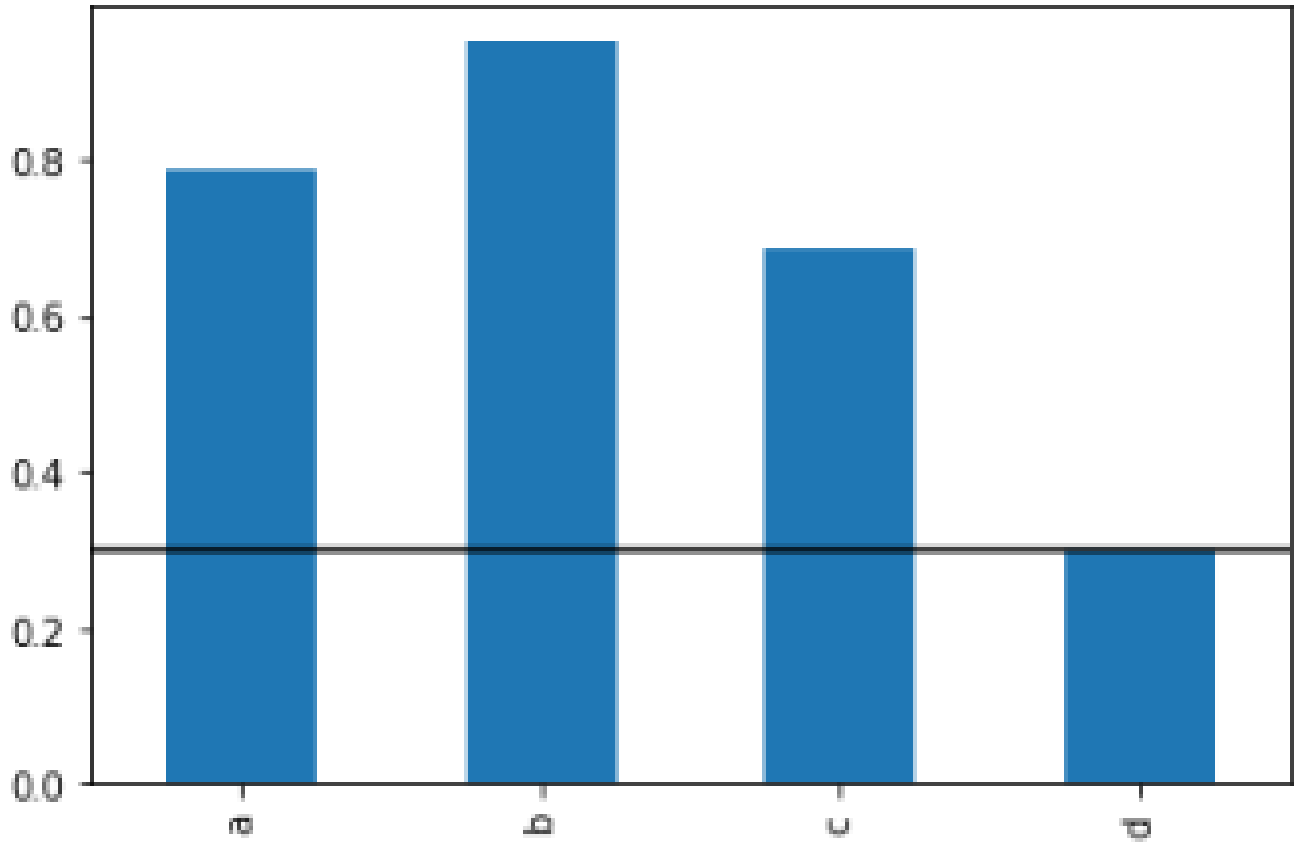

```
df.iloc[5].plot.bar()
```

```
plt.axhline(0.3, color = 'k')
```



cf) `plt.axvline(0.3, color = 'k')`

```
df.iloc[5].plot.bar()
plt.axhline(0.3, color = 'k')
```



color는 ‘r’, ‘g’, ‘b’, ‘m’, ‘c’, ‘y’, ‘k’(검정색), ‘w’(흰색) 등등

```
import seaborn as sns
```

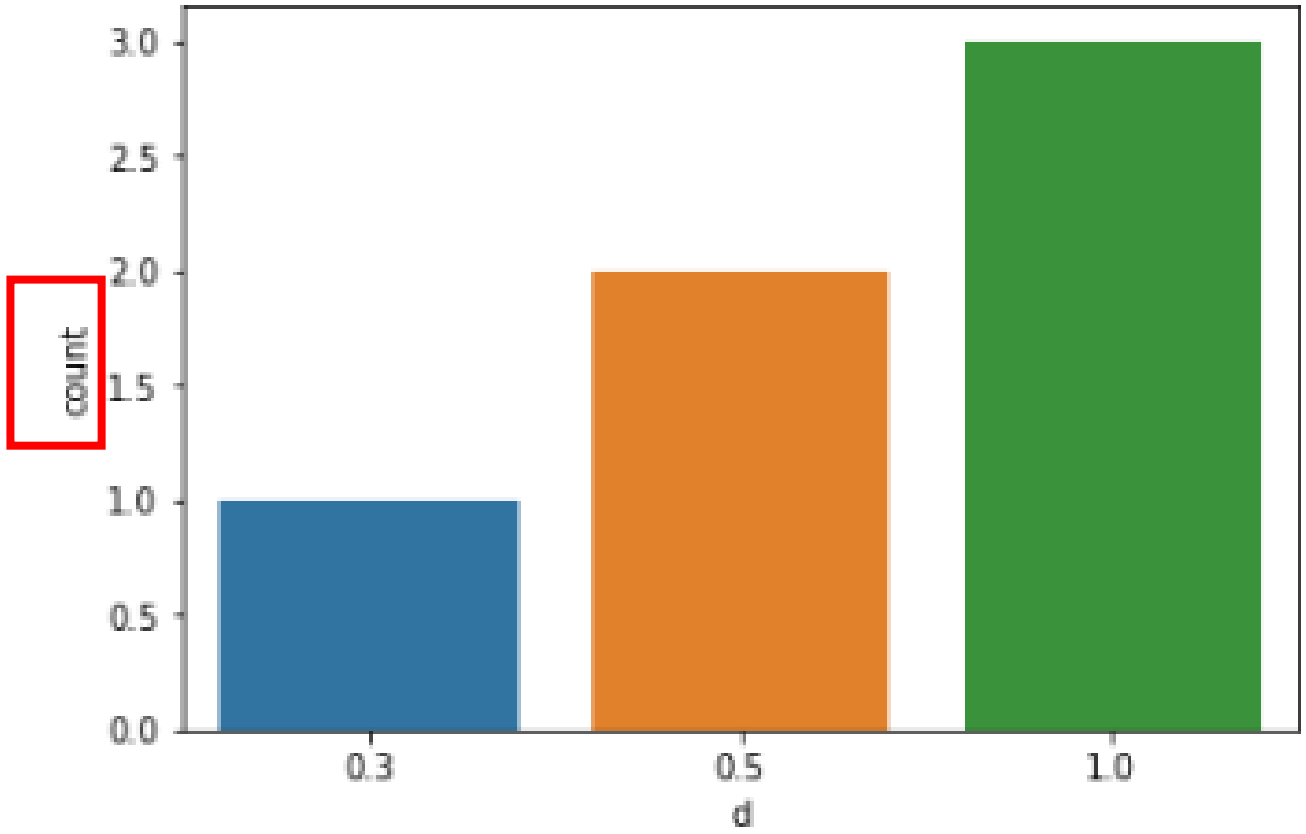
```
sns.countplot(data=df, x='d')
```

y축 : 개수

```
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

```
sns.countplot(data = , x = , hue = )
```



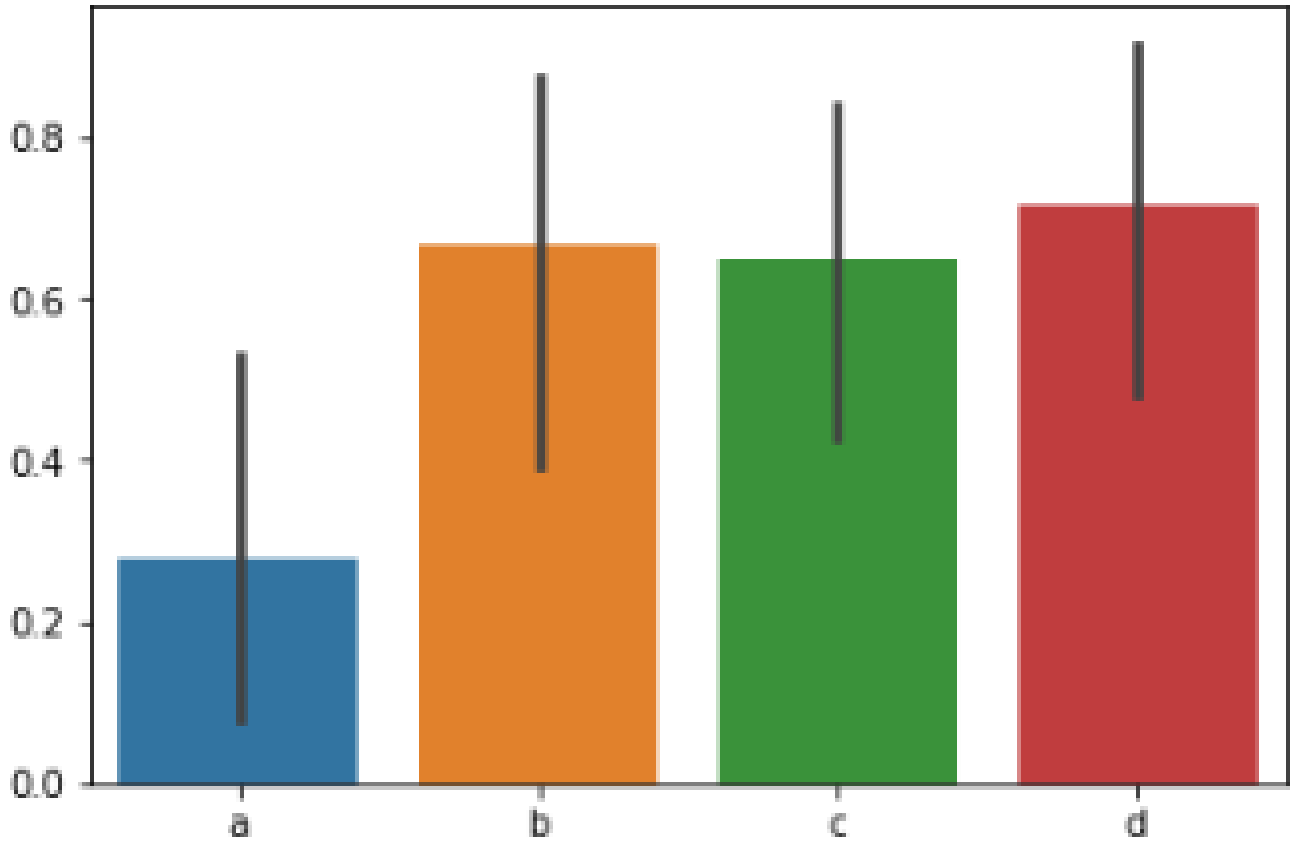
```
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

```
sns.barplot(data=df)
```

y축 : 평균값(과 편차)

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d7
```

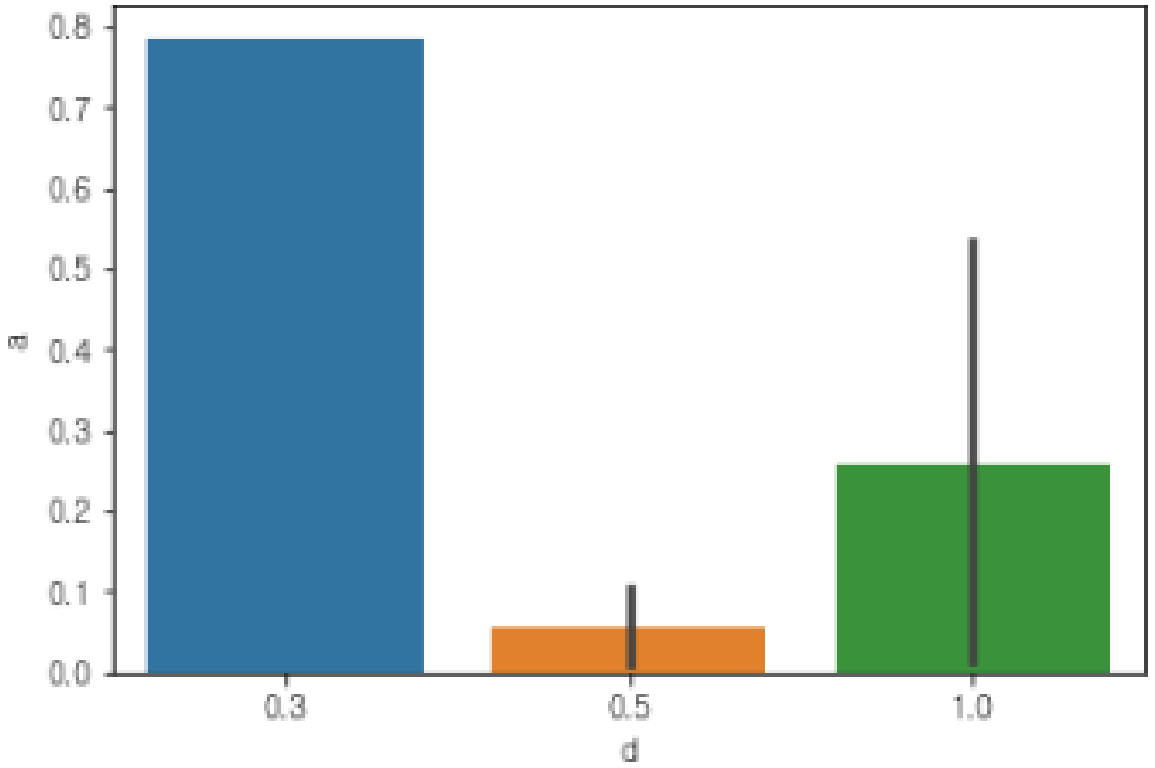


```
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

```
sns.barplot(data=df, x='d', y='a')
```

```
sns.barplot(data = , x = , y = , hue = )
```



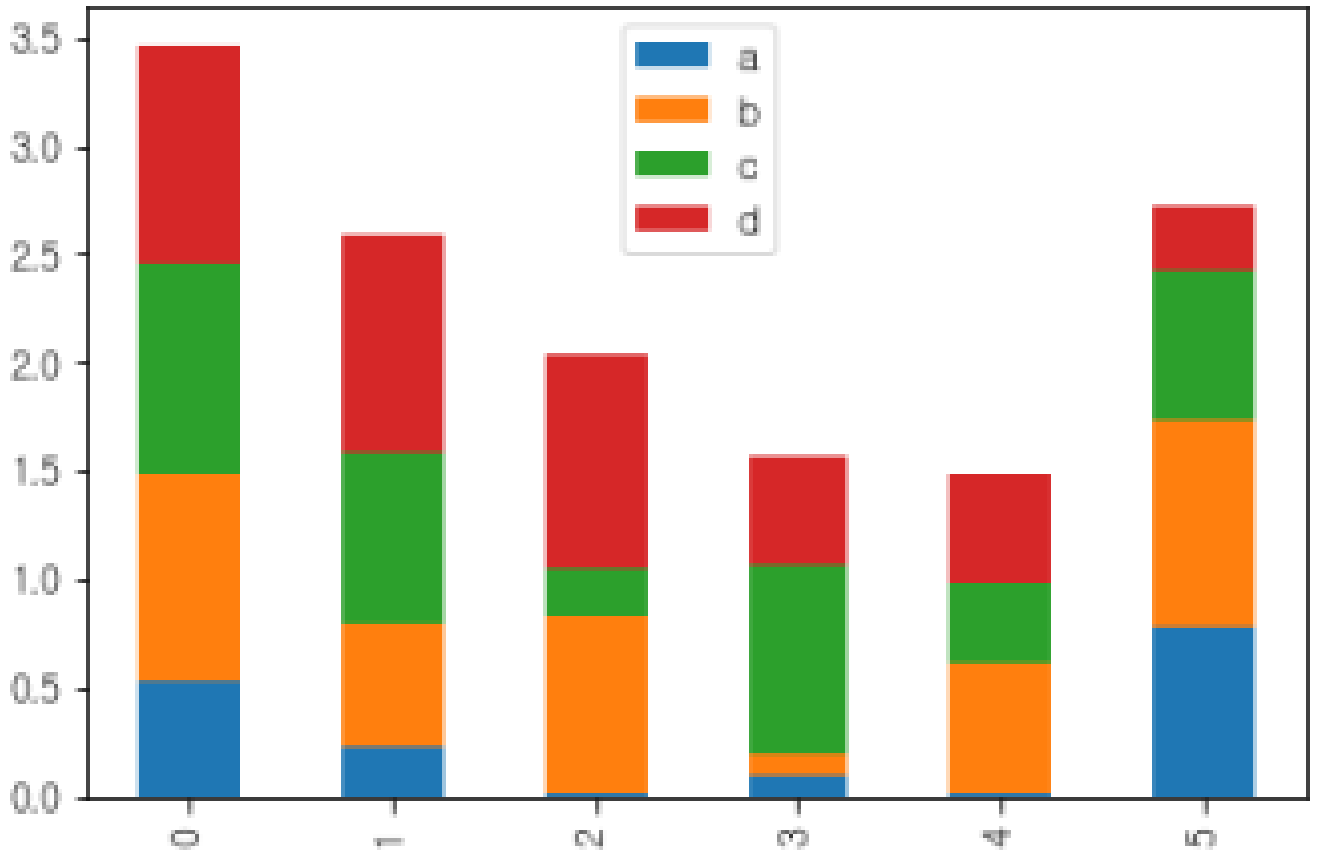
- countplot : x축 애들 각각의 개수
- barplot: y축의 높이가 주 관심사 일때, y축이 연속데이터일땐 barplot이 더 나음, 편차도 표시 됨, 특이값 등의 판단에 도움(검정 선)

```
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

```
df.plot.bar(stacked=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d
```

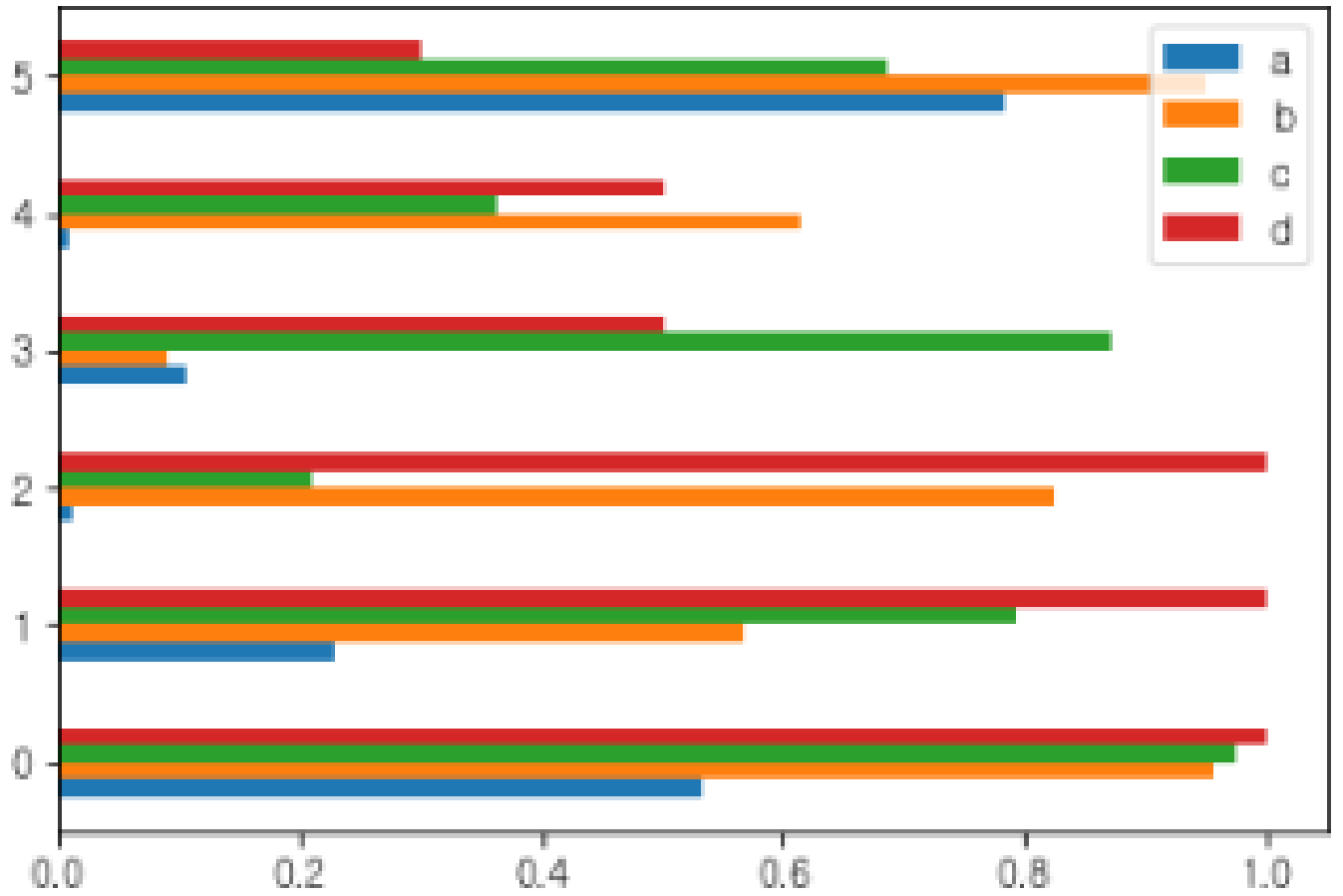


```
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

```
df.plot.barh()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24
```

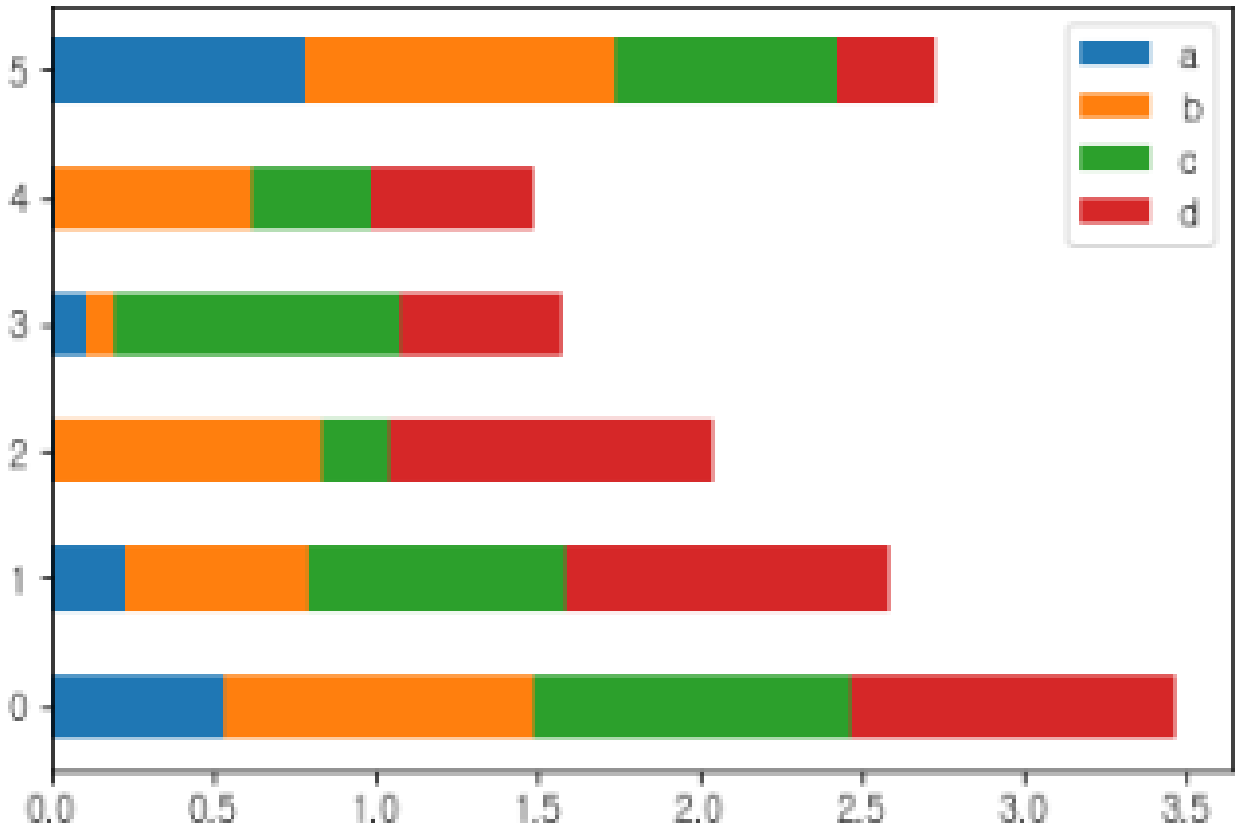


```
df
```

	a	b	c	d
0	0.532203	0.955185	0.975651	1.0
1	0.227575	0.567579	0.793520	1.0
2	0.012863	0.823781	0.208456	1.0
3	0.106074	0.091248	0.871766	0.5
4	0.008764	0.613667	0.363355	0.5
5	0.785297	0.949217	0.685388	0.3

```
df.plot.barh(stacked=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24
```



import matplotlib.pyplot as plt

plt.bar( )로 막대그래프 그릴 경우의 구성

plt.bar

```
bar(x, height, width=0.8, bottom=0, align='center')
```

- *x* 정성적 변수가 가지는 명목값 (x축 값)
- *height* 정성적 변수의 값 (y축 값)
- width 막대 그래프의 폭 (비율값)
- bottom y 좌표의 값 (실수, 실수 배열)
- align 막대들의 정렬 방식 지정, {'center', 'edge'}

plt.barh

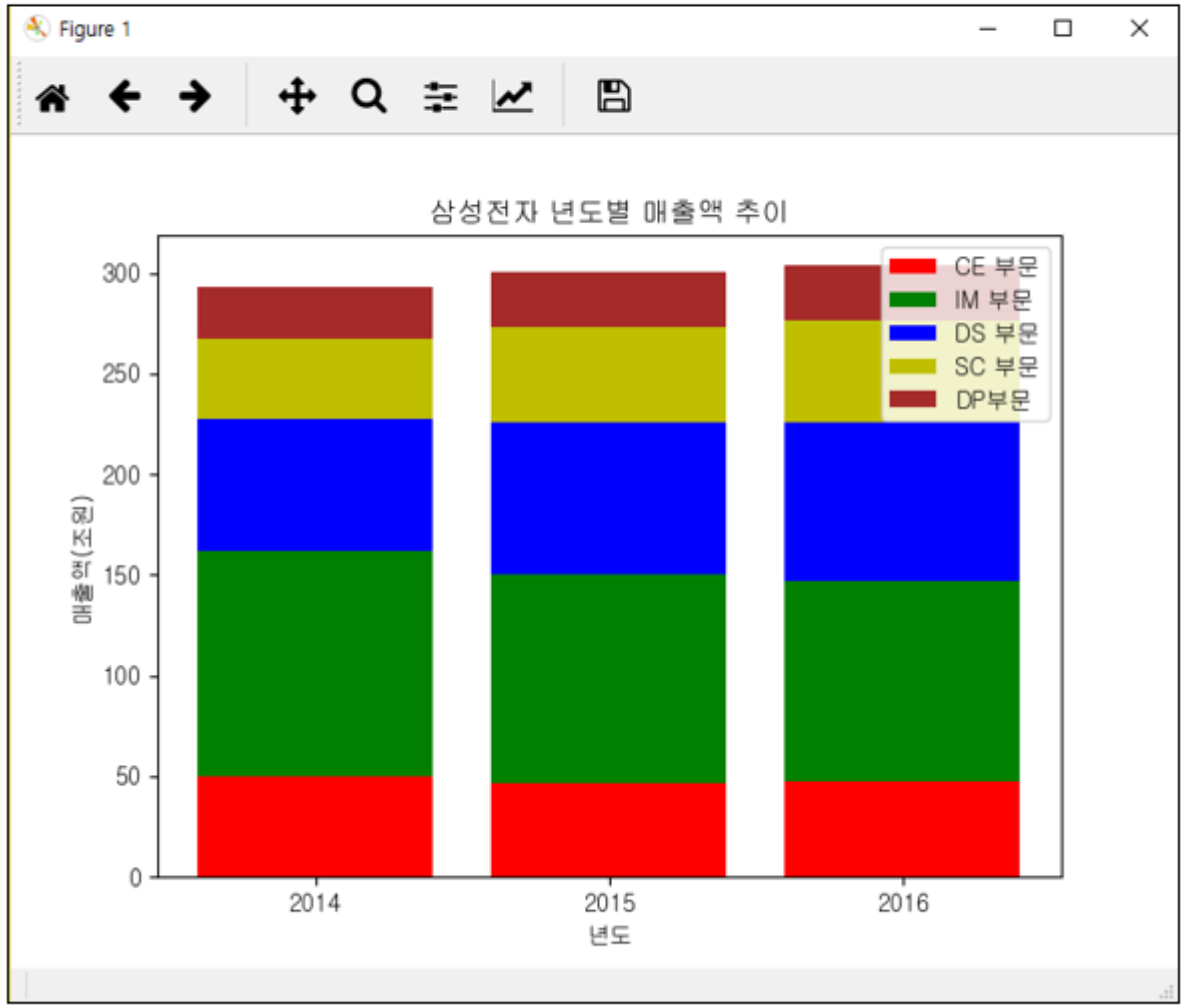
```
barh(x, width, height=0.8, left=0, align='center')
```

- *x* 정성적 변수가 가지는 명목값 (y축 값)
- *width* 정성적 변수의 값 (x축 값)
- height 막대 그래프의 폭 (비율값)
- left x 좌표의 값 (실수, 실수 배열)
- align 막대들의 정렬 방식 지정, {'center', 'edge'}

2014년 ~ 2016년 삼성전자 부문별 매출액 자료를 다음과 같은 막대그래프로 나타내어라.

data/se_sales.txt

```
2014 ~ 2016년 분기별 부분별 삼성전자 매출액
CE : Customer Electronics
IM : IT & Mobile Communications
DS : Device Solutions
SC : Semi-Conductor
DP : Displays & TV

년도 분기 CE    IM     DS     SC     DP
2014 1  11.32  32.44  15.56   9.39   6.1
2014 2  13.    28.45  16.23   9.78   6.33
2014 3  11.6   24.58  16.29   9.89   6.25
2014 4  14.27  26.29  17.71  10.66   7.05
2015 1  10.26  25.89  17.1   10.27   6.85
2015 2  11.2   26.06  17.87  11.29   6.62
2015 3  11.59  26.61  20.31  12.82   7.49
2015 4  13.85  25.    19.74  13.21   6.53
2016 1  10.62  27.6   17.18  11.15   6.04
2016 2  11.55  26.56  18.43  12.     6.42
2016 3  11.24  22.54  20.29  13.15   7.06
2016 4  13.64  23.61  22.26  14.86   7.42
```



```
# -*- coding:utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc

f_name = font_manager.FontProperties(fname='C:/Windows/Fonts/NGULIM.ttf').get_name()
rc('font', family=f_name)

sales_data = np.loadtxt('data/se_sales.txt', skiprows=8)
years = np.arange(2014, 2017)

s_2014 = sales_data[sales_data[:, 0]==2014][:, 2:].sum(axis=0)
s_2015 = sales_data[sales_data[:, 0]==2015][:, 2:].sum(axis=0)
s_2016 = sales_data[sales_data[:, 0]==2016][:, 2:].sum(axis=0)
sales = np.array([s_2014, s_2015, s_2016])

plt.figure(figsize=(7, 5))
plt.title('삼성전자 년도별 매출액 추이')
plt.xlabel('년도')
plt.ylabel('매출액(조원)')
plt.xticks(years)

plt.bar(years, sales[:,0], color='r')
plt.bar(years, sales[:,1], bottom=sales[:,0], color='g')
plt.bar(years, sales[:,2], bottom=sales[:,0]+sales[:,1], color='b')
plt.bar(years, sales[:,3], bottom=sales[:,0]+sales[:,1]+sales[:,2], color='y')
plt.bar(years, sales[:,4], bottom=sales[:,0]+sales[:,1]+sales[:,2]+sales[:,3],
color='brown')

plt.legend(('CE 부문', 'IM 부문', 'DS 부문', 'SC 부문', 'DP부문'))
plt.show()
```

```
sd =pd.read_csv('se_sales.txt', skiprows=7, sep='₩t', encoding='utf-8')
sd
```

```
sd =pd.read_csv('se_sales.txt', skiprows=7, index_col=['년도'], sep='₩t', encoding='utf-8')
sd
```

```
sd_g=sd.groupby('년도').sum()
sd_g
```

```
sd_g2 = sd_g.drop(columns='분기')
sd_g2
```

```
sd.dtypes
```

sd.dtypes 하면 연도 데이터를 int64로 인식하고 있음.

연도, 즉 날짜로 인식하게 하려면 파일 불러올 때 parse_dates=

pd.read_csv("se_sales.txt", parse_dates=["년도"])

```
sd_g2.plot.bar(stacked=True, color=['r', 'g', 'b', 'y', 'brown'])
plt.legend(('CE부문', 'IM부문', 'DS부문', 'SC부문','DP부문'),  loc = 'upper right')
plt.title('삼성전자 년도별 매출액 추이')
plt.ylabel('매출액(조원)')
```

	datetime
0	2011-01-01 00:00:00
1	2011-01-01 01:00:00
2	2011-01-01 02:00:00
3	2011-01-01 03:00:00
4	2011-01-01 04:00:00

	datetime	datetime-year	datetime-month	datetime-day	datetime-hour	datetime-minute	datetime-second	datetime-dayofweek
0	2011-01-01 00:00:00	2011	1	1	0	0	0	5
1	2011-01-01 01:00:00	2011	1	1	1	0	0	5
2	2011-01-01 02:00:00	2011	1	1	2	0	0	5

2. 선그래프

```
df.plot?
```

```
df.plot.line?
```

```
df.plot.line(x=None, y=None, **kwds)
```

- 선 스타일 '-' (실선), '--' (파선), ':' (점선), '-.' (파-점선)
- 선의 색 'r', 'g', 'b', 'm', 'c', 'y', 'k'(검정색), 'w'(흰색)
- 데이터 표식 'o'(원), 's' (사각형), 'x' (x), '^' (삼각형), 'd' (다이아몬드)

그래프 각종 유형, 옵션 참고 사이트

- http://matplotlib.org/gallery.html
- https://seaborn.pydata.org/generated/seaborn.lineplot.html

```
df
```

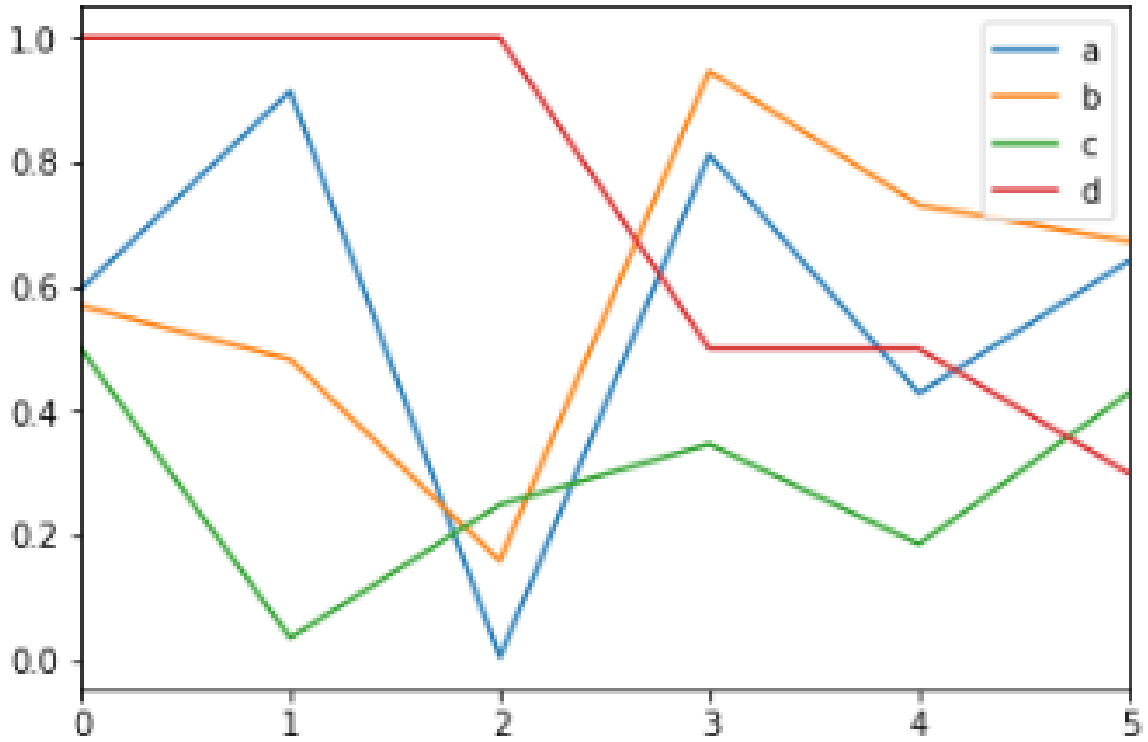
	a	b	c	d
0	0.595138	0.569050	0.500250	1.0
1	0.913631	0.482730	0.035095	1.0
2	0.004119	0.158990	0.249517	1.0
3	0.811525	0.946090	0.346448	0.5
4	0.428091	0.730734	0.185360	0.5
5	0.640128	0.673343	0.428033	0.3

```
df.plot()
```

```
df.plot.line()
```

```
plt.plot(df)
```

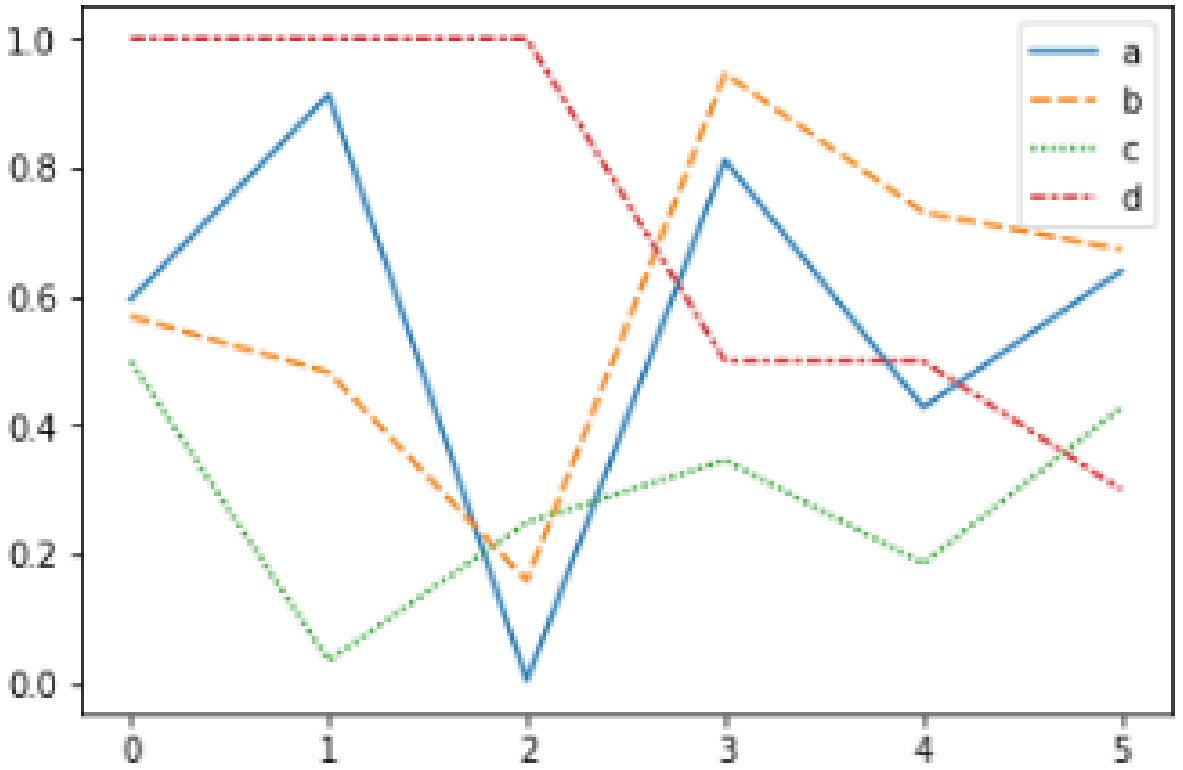
sns는 ?



```
df
```

	a	b	c	d
0	0.595138	0.569050	0.500250	1.0
1	0.913631	0.482730	0.035095	1.0
2	0.004119	0.158990	0.249517	1.0
3	0.811525	0.946090	0.346448	0.5
4	0.428091	0.730734	0.185360	0.5
5	0.640128	0.673343	0.428033	0.3

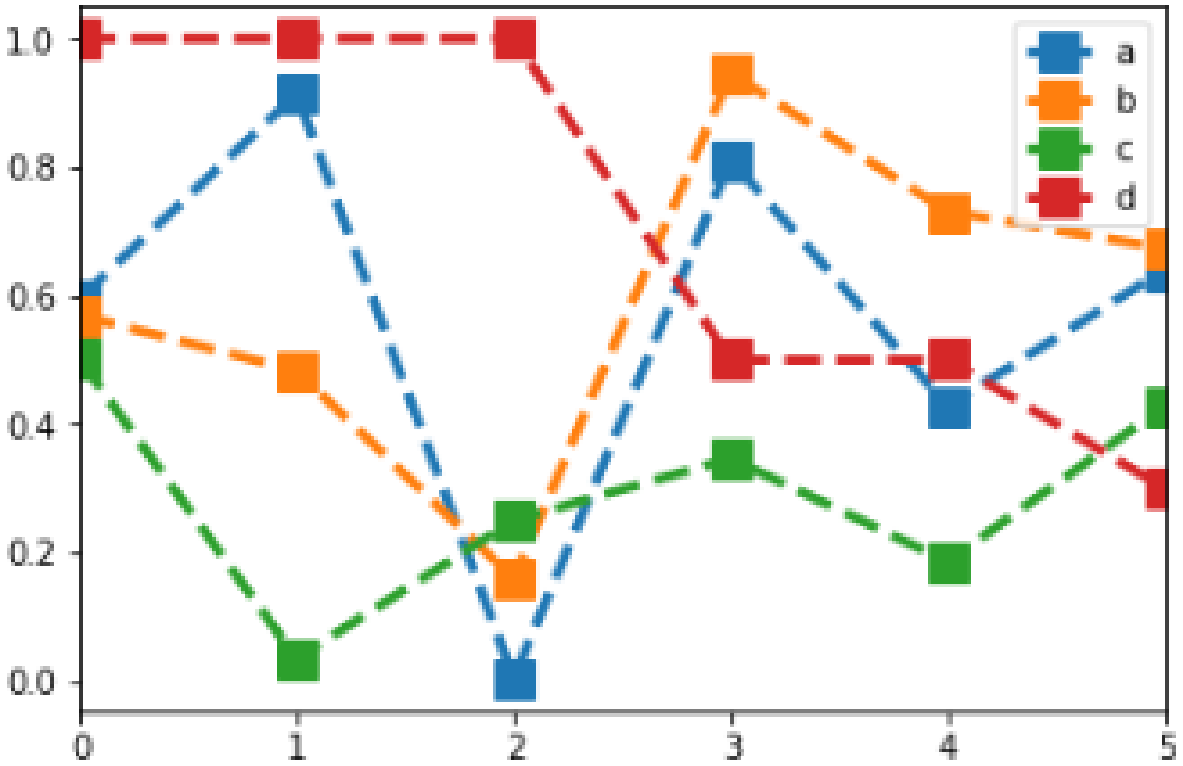
```
sns.lineplot(data=df)
```



```
df.plot(marker = 's', linestyle = 'dashed', linewidth = 3, markersize = 12)
```

```
plt.plot(df, marker='s', linestyle = 'dashed', linewidth = 3, markersize = 12)
```

```
plt.plot(df, 's--', linewidth = 3, markersize = 12)
```



linestyle
'solid', 'dashed',
'dashdot', 'dotted'

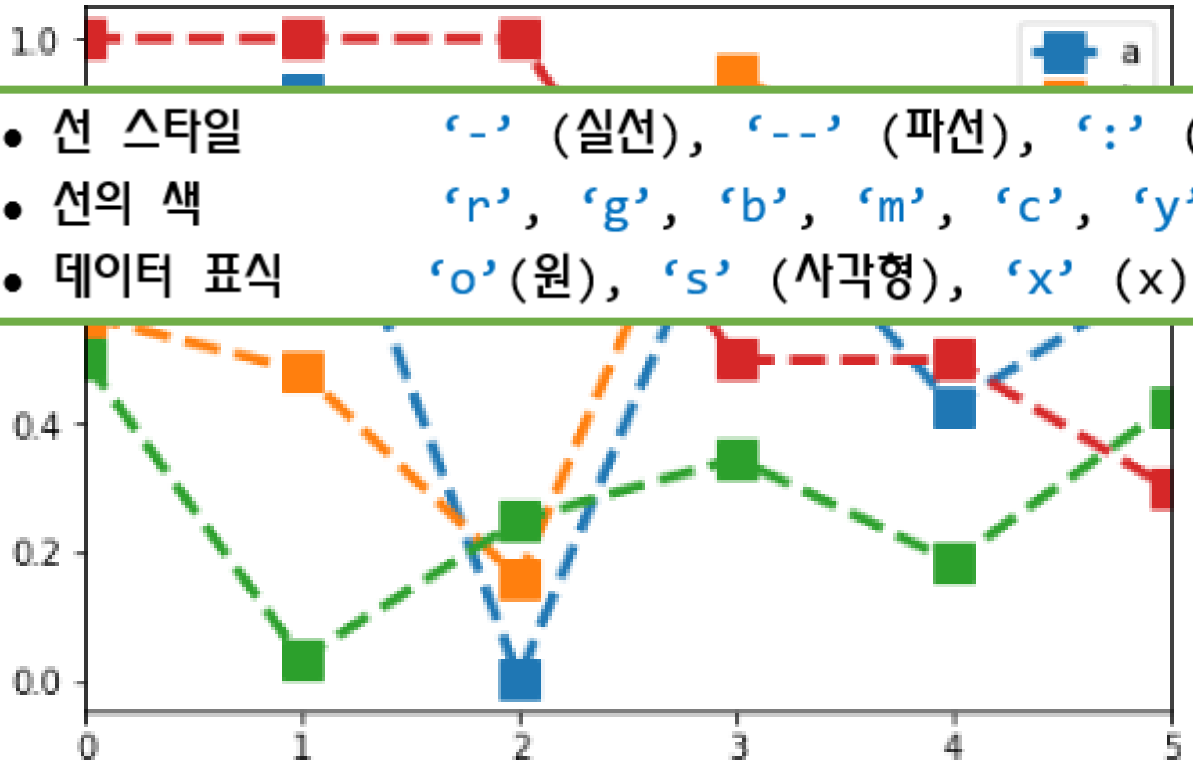
https://matplotlib.org/api/markers_api.html

https://matplotlib.org/gallery/lines_bars_and_markers/line_styles_reference.html

```
df.plot(marker = 's', linestyle = 'dashed', linewidth = 3, markersize = 12)
```

```
plt.plot(df, marker='s', linestyle = 'dashed', linewidth = 3, markersize = 12)
```

```
plt.plot(df, 's--', linewidth = 3, markersize = 12)
```



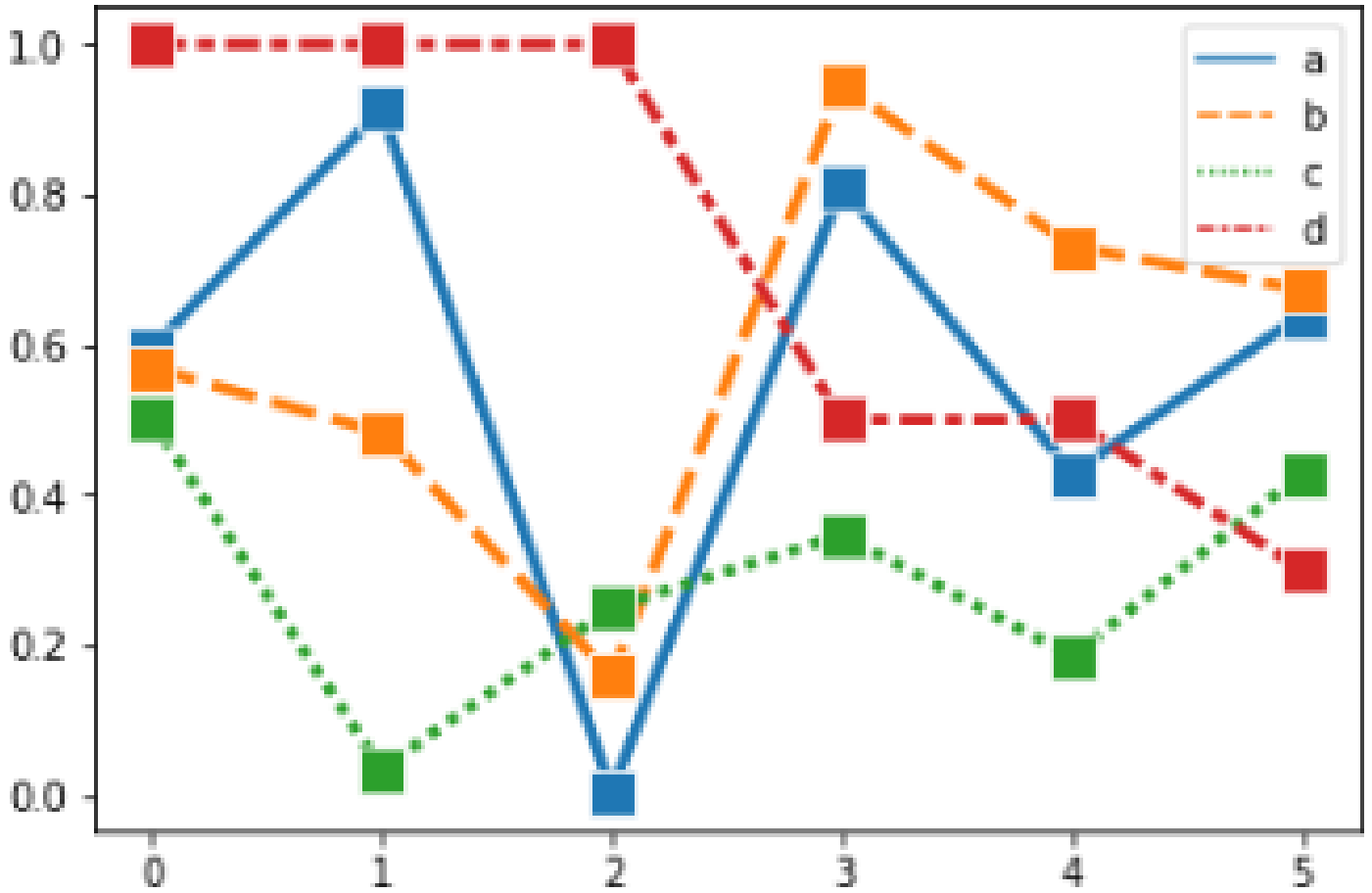
linestyle

- 선 스타일 '-' (실선), '--' (파선), ':' (점선), '-.' (파-점선)
- 선의 색 'r', 'g', 'b', 'm', 'c', 'y', 'k'(검정색), 'w'(흰색)
- 데이터 표식 'o'(원), 's' (사각형), 'x' (x), '^' (삼각형), 'd' (다이아몬드)

https://matplotlib.org/api/markers_api.html

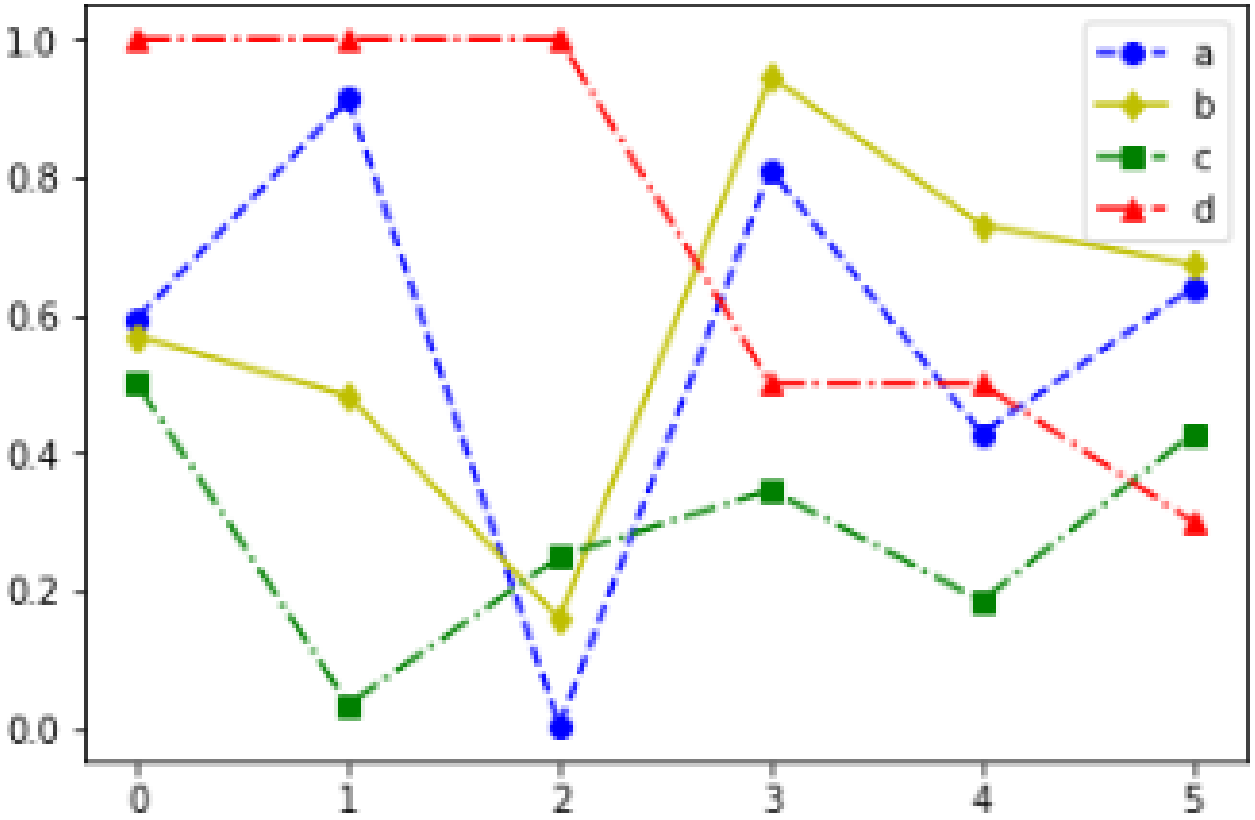
https://matplotlib.org/gallery/lines_bars_and_markers/line_styles_reference.html

```
sns.lineplot(data=df, marker='s', linewidth = 3, markersize = 12)
```



https://seaborn.pydata.org/generated/seaborn.lineplot.html

```
plt.plot(df['a'], 'bo--')
plt.plot(df['b'], 'yd-')
plt.plot(df['c'], 'gs-.')
plt.plot(df['d'], 'r^-.')
plt.legend(('a', 'b', 'c', 'd'), loc = 'best')
```



plt.기준

loc 인수	설명	loc 인수	설명
없는 경우	자동	0	가능한 최적의 위치
1	오른쪽 위	2	왼쪽 위
3	왼쪽 아래	4	오른쪽 아래
5	오른쪽	6	왼쪽 중앙
7	오른쪽 중앙	8	중앙 아래
9	중앙 위	10	중앙

글자	표시	글자	표시
-	실선	--	대사선
-.	대시-점선	:	점선
.	점 표시	,	픽셀 표시
o	원 표시		아래 쪽 삼각형 표시
^	위쪽 삼각형 표시	<	왼쪽 삼각형 표시
>	오른쪽 삼각형 표시	1	아래 쪽 삼각형 표시
2	위쪽 삼각형 표시	3	왼쪽 삼각형 표시
4	오른쪽 삼각형 표시	s	사각형 표시
p	오각형 표시	*	별 표시
h	육각형1 표시	H	육각형2 표시
+	덧셈 기호 표시	x	곱셈 기호 표시
D	다이아몬드 표시	d	가는 다이아몬드 표시
|	브이 라인 표시(Vline)	-	-

plt.plot()의 인자	설명
x	x 좌표(가로 축)의 값을 담고 있는 배열이나 리스트
y	y 좌표(세로 축)의 값을 담고 있는 배열이나 리스트
color	색상을 지정한다.(color='black')
linestyle	라인 스타일을 설정한다. (linestyle='dashed')
marker	실제 데이터를 돋보이게 하기 위한 마커를 설정한다.
label	범례를 위한 라벨을 설정한다.(label='Hohoho')
drawstyle	연결되어 있는 선들을 연결하기 위한 스타일을 지정한다. (drawstyle='steps-post')
ax	서브 플롯을 설정/지정하기 위한 매개 변수이다.

plt.기준

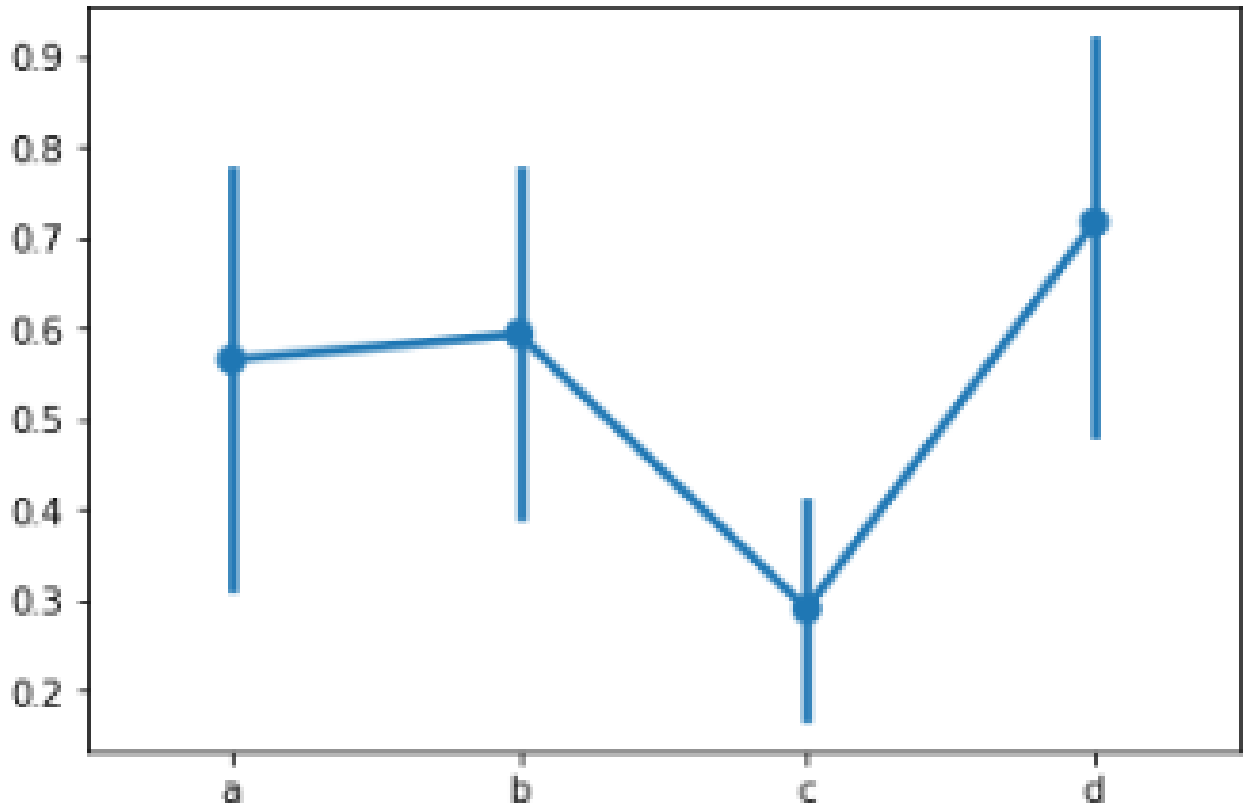
plt 관련 함수	설명
figure()	Figure란 matplotlib에서 그래프가 들어 가는 영역을 의미한다. Figure 객체를 생성한다. plt.figure(figsize=(8,8)) # 크기 8, 8의 그림 영역을 만들어 준다.
savefig()	그래프을 파일 형식으로 저장한다.
show()	그래프를 화면에 보여 준다.
grid(boolean)	그리드 표시 여부
xlabel()	x 축에 라벨을 출력한다.
ylabel()	y 축에 라벨을 출력한다.
title()	타이틀을 출력한다.
axis(string)	축 간격을 조정한다. 예시) plt.axis('tight')이면 축 간격을 좀 더 조밀하게 조정한다. 예시) plt.axis('off')
xlim()	x 축의 상한/하한 값을 설정한다. 예시) plt.xlim([0, 10]) : x축의 범위를 0부터 10까지로 설정한다.
ylim()	y 축의 상한/하한 값을 설정한다. 예시) plt.ylim(np.min(y) - 1 , np.max(y) + 1)
bar()	bar chart를 그린다.
pie()	Pie 차트를 그린다.
hist()	히스토그램을 그린다.
subplot()	서브 플롯을 지정한다. 예시) plt.subplot(211) # 2행 1열의 1번째 서브 플롯을 의미한다.
legend()	범례를 그려 준다.

```
df
```

	a	b	c	d
0	0.595138	0.569050	0.500250	1.0
1	0.913631	0.482730	0.035095	1.0
2	0.004119	0.158990	0.249517	1.0
3	0.811525	0.946090	0.346448	0.5
4	0.428091	0.730734	0.185360	0.5
5	0.640128	0.673343	0.428033	0.3
mean	0.565439	0.593490	0.290784	0.716667
std	0.323172	0.265026	0.169709	0.318852

```
sns.pointplot(data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x227
```

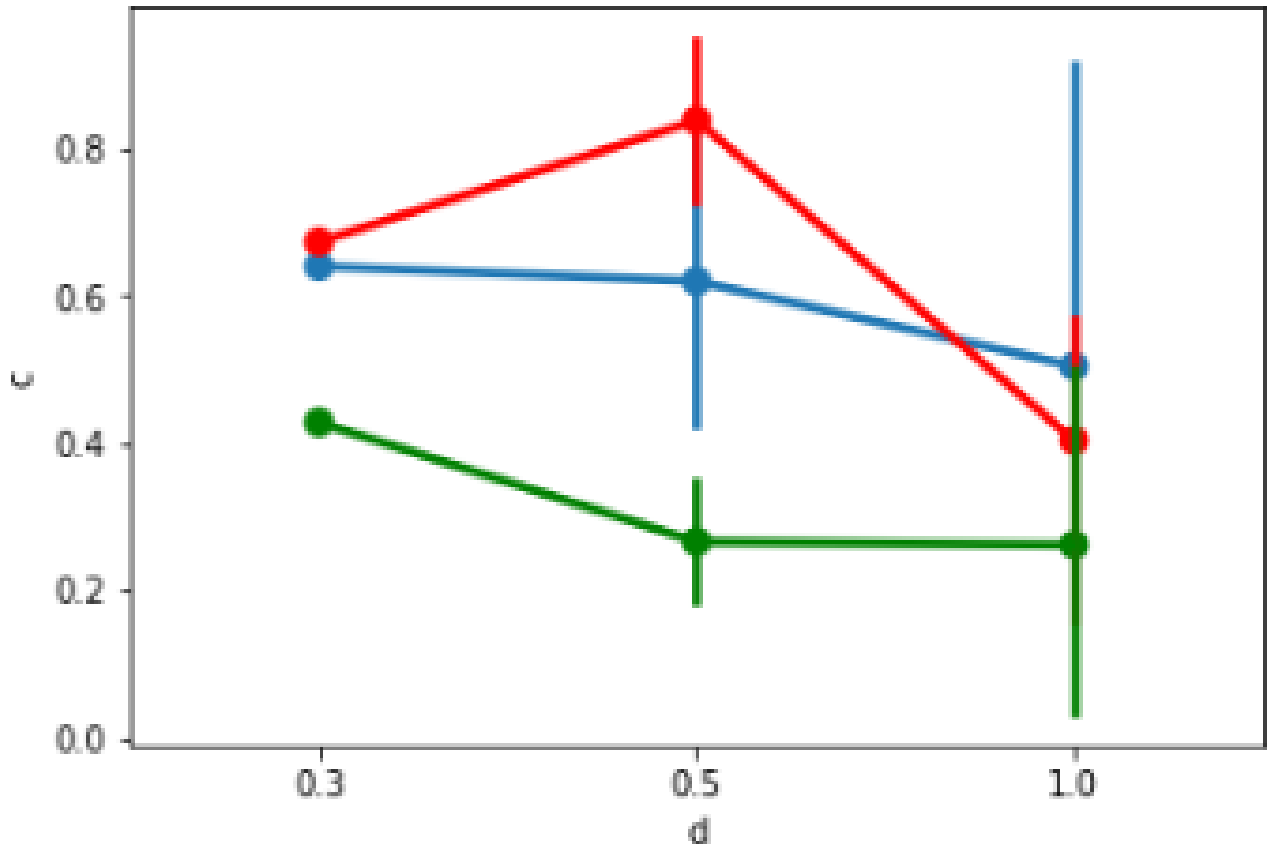


```
df
```

	a	b	c	d
0	0.595138	0.569050	0.500250	1.0
1	0.913631	0.482730	0.035095	1.0
2	0.004119	0.158990	0.249517	1.0
3	0.811525	0.946090	0.346448	0.5
4	0.428091	0.730734	0.185360	0.5
5	0.640128	0.673343	0.428033	0.3

```
sns.pointplot(data=df, x = 'd', y = 'a')
sns.pointplot(data=df, x = 'd', y = 'b', color = 'r')
sns.pointplot(data=df, x = 'd', y = 'c', color = 'g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x227f23b16a
```

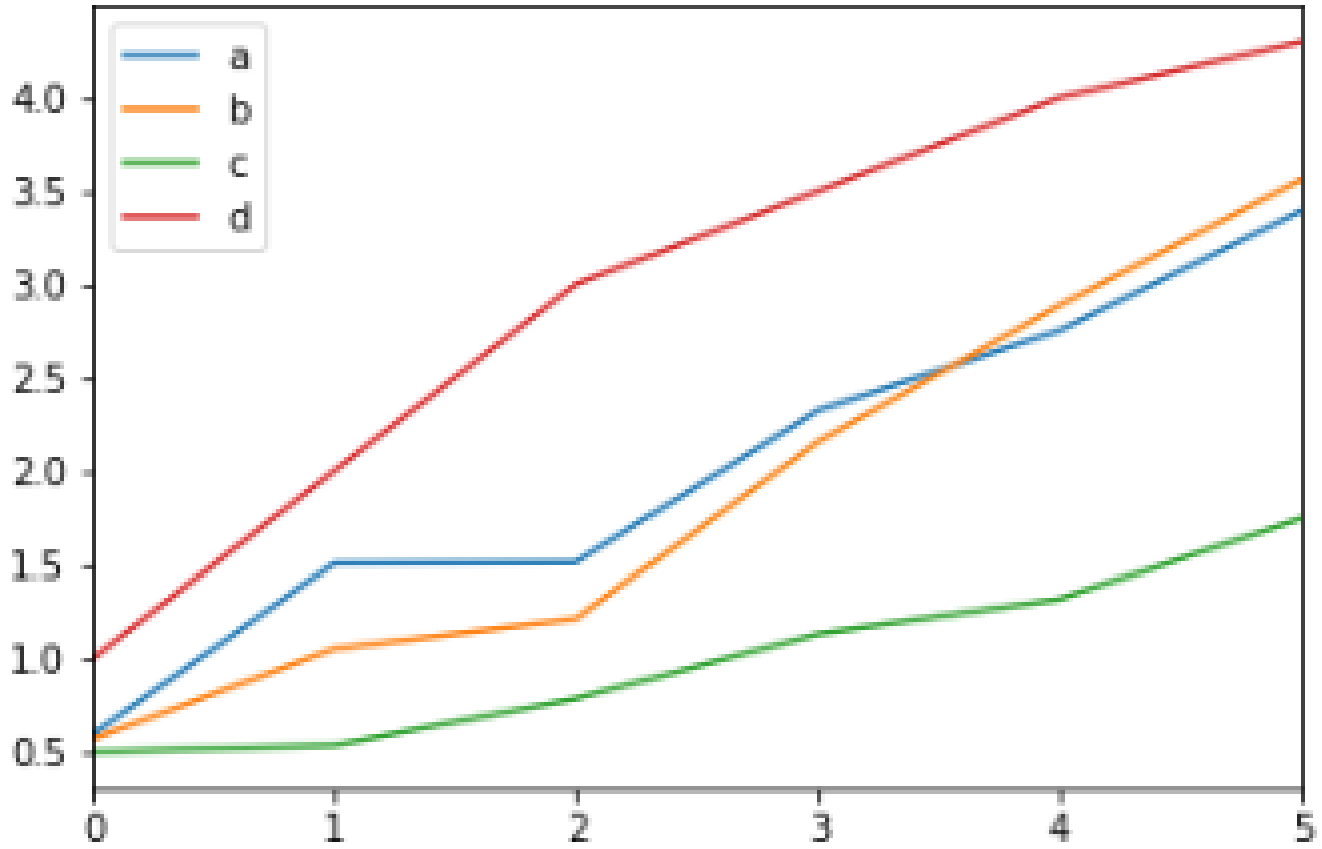


```
df
```

	a	b	c	d
0	0.595138	0.569050	0.500250	1.0
1	0.913631	0.482730	0.035095	1.0
2	0.004119	0.158990	0.249517	1.0
3	0.811525	0.946090	0.346448	0.5
4	0.428091	0.730734	0.185360	0.5
5	0.640128	0.673343	0.428033	0.3

```
df2 = df.cumsum()
df2.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x227f
```



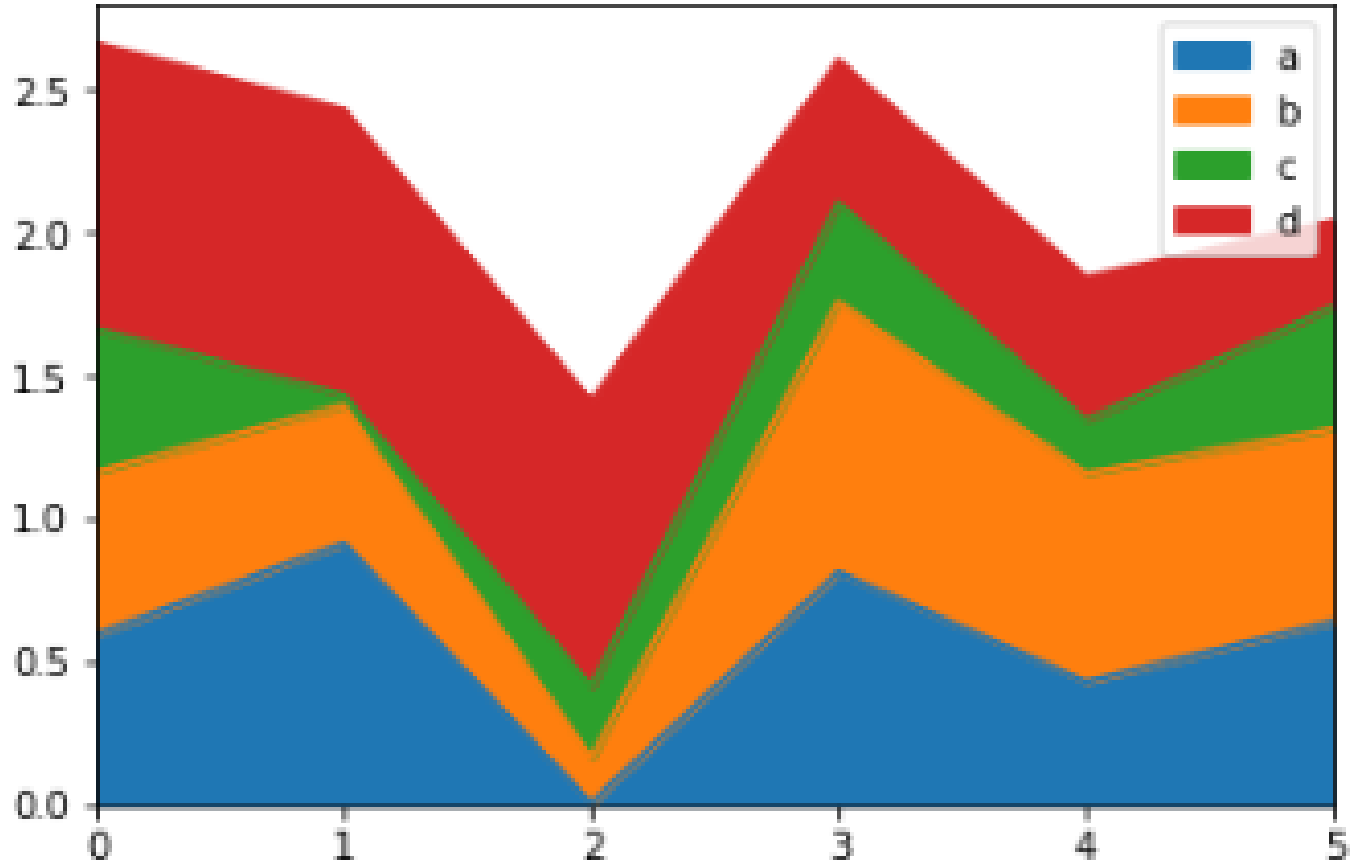
cf) area 그래프 – (행 별 데이터)누적

```
df
```

	a	b	c	d
0	0.595138	0.569050	0.500250	1.0
1	0.913631	0.482730	0.035095	1.0
2	0.004119	0.158990	0.249517	1.0
3	0.811525	0.946090	0.346448	0.5
4	0.428091	0.730734	0.185360	0.5
5	0.640128	0.673343	0.428033	0.3

```
df.plot.area()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x227f
```



https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html#area-plot

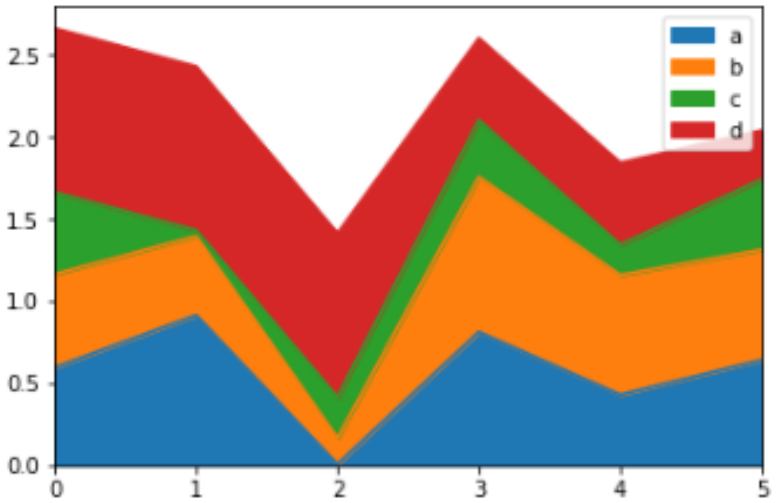
```
df
```

	a	b	c	d
0	0.595138	0.569050	0.500250	1.0
1	0.913631	0.482730	0.035095	1.0
2	0.004119	0.158990	0.249517	1.0
3	0.811525	0.946090	0.346448	0.5
4	0.428091	0.730734	0.185360	0.5
5	0.640128	0.673343	0.428033	0.3

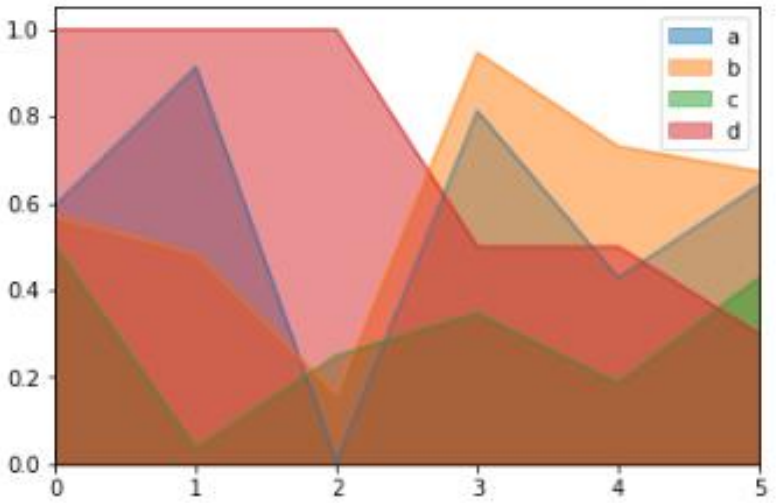
Area그래프는

- Nan값이 있으면 자동으로 0으로 채워서 그려 줌, 따로 처리 원하면 area plot 전에 손 보고 실행할 것
- 여러 열의 데이터를 쌓아 올려 그리는 것이 기본 설정 되어 있음 (stacked = True, cf) cumulative)

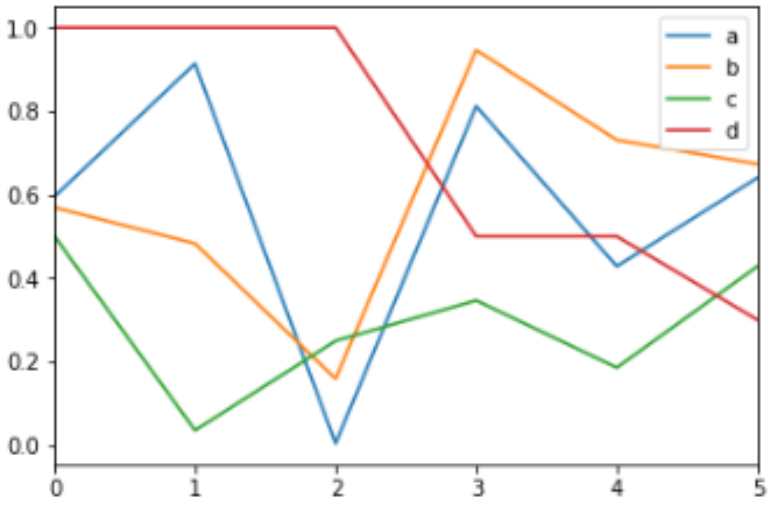
```
df.plot.area()
```




```
df.plot.area(stacked = False)
```




```
df.plot()
```



Plotting with missing data

Pandas tries to be pragmatic about plotting `DataFrames` or `Series` that contain missing data. Missing values are dropped, left out, or filled depending on the plot type.

Plot Type	NaN Handling
Line	Leave gaps at NaNs
Line (stacked)	Fill 0's
Bar	Fill 0's
Scatter	Drop NaNs
Histogram	Drop NaNs (column-wise)
Box	Drop NaNs (column-wise)
Area	Fill 0's
KDE	Drop NaNs (column-wise)
Hexbin	Drop NaNs
Pie	Fill 0's

If any of these defaults are not what you want, or if you want to be explicit about how missing values are handled, consider using **`fillna()`** or **`dropna()`** before plotting.

3. 히스토그램

히스토그램(histogram)

- 연속형 변수의 데이터 값을 구간(bins)별 빈도로 변환하여 도식화하는 그래프
- 데이터의 분포 모양을 표현하는 데 유용
  - 퍼짐도, 왜도(skewness), 모드(mode) 등
- 분포의 이상값(outlier) 또는 특이한 패턴 파악에 용이

히스토그램 이슈

- 데이터의 수가 작은 경우, 잘못된 결론이 나올 수 있음\
  - 데이터, 데이터 구간, 데이터 구간의 수 등의 작은 변동에 따라 그래프의 모양의 큰 변동이 발생 가능

- 데이터의 수가 큰 경우, 분포의 일반적인 특성을 보여주는 효과적인 도구

https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html#histograms

plt.

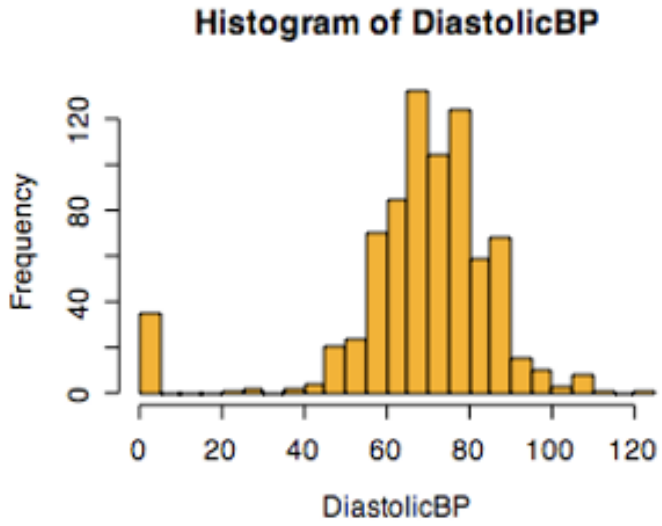
```
hist(x, bins=None, label=None, density=False, cumulative=False,
     histtype='bar', orientation='horizontal', facecolor=None,
     alpha=None)
```

- *x* 데이터, numpy 배열
- bins 구간 지정, 정수 또는 리스트(튜플)
  - 정수 : 구간의 수, (bins+1)개의 edge가 생성
  - 리스트(튜플) : 구간값 지정

    bins=[1, 2, 3, 4] ➔ [1, 2), [2, 3), [3, 4]
- label 데이터에 대한 레이블 설정, (문자열, 문자열 리스트(튜플))
- density True인 경우 y축이 빈도가 아닌 확률값으로 설정
- cumulative True인 경우, 누적 빈도수를 표현
- histtype 막대 모양 지정, {'bar', 'barstacked', 'step', 'stepfilled'}
  - bar : 막대 그래프
  - barstacked : 누적 막대 그래프
  - step : 직선 그래프 (채움색이 없음)
  - stepfilled : 직선 그래프 (채움색을 적용)
- orientation 히스토그램의 수평, 수직 그리기 설정, {'horizontal', 'vertical'}

plt.기준

인수	설명
x	리스트 객체 혹은 ndarray 객체
bins	빈도 구분 값(bin)의 수
range	빈도 구분 값(bin)의 위 아래 범위
normed	전체 값의 합이 1이 되도록 정규화 하는 지의 여부
weights	x에 대한 가중치
cumulative	각 빈도 구분 값이 하위의 빈도 구분 값을 누적하는 지의 여부
histtype	옵션(문자열) : bar, barstacked, step, stepfilled
align	옵션(문자열) : left, mid, right
orientation	옵션(문자열) : horizontal, vertical
rwidth	각 막대의 상대적인 폭
log	로그 스케일
color	각 자료의 색상
label	라벨로 사용되는 문자열 혹은 문자열의 목록
stacked	여러 개의 자료를 쌓아 올려서 표시하는 지의 여부

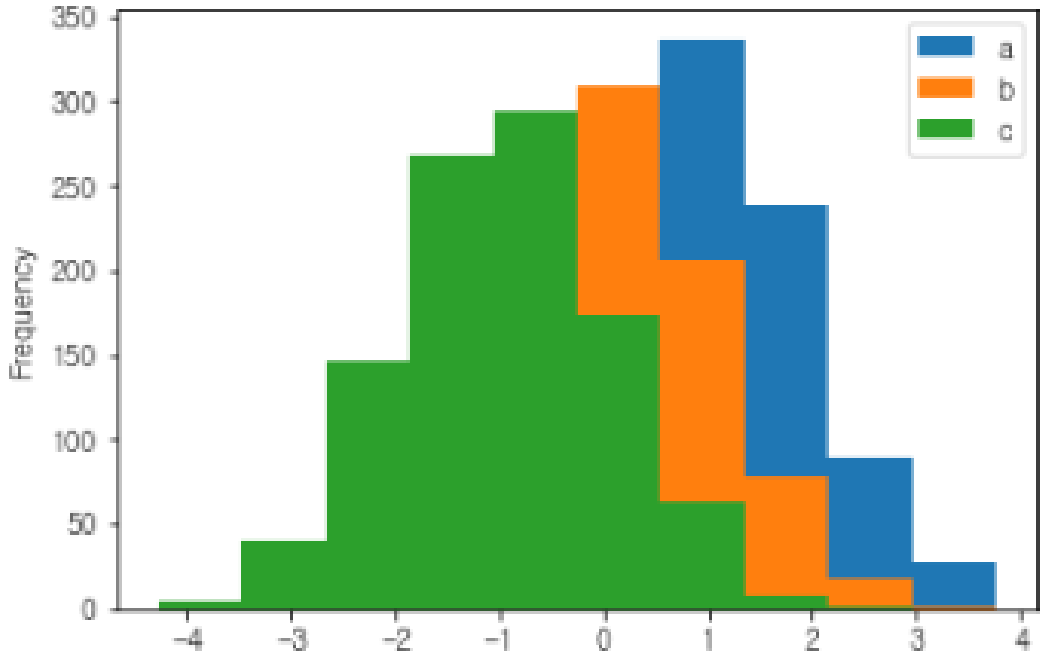


그래프의 세부 옵션 중,

(alpha = 숫자)는 투명도를 줌

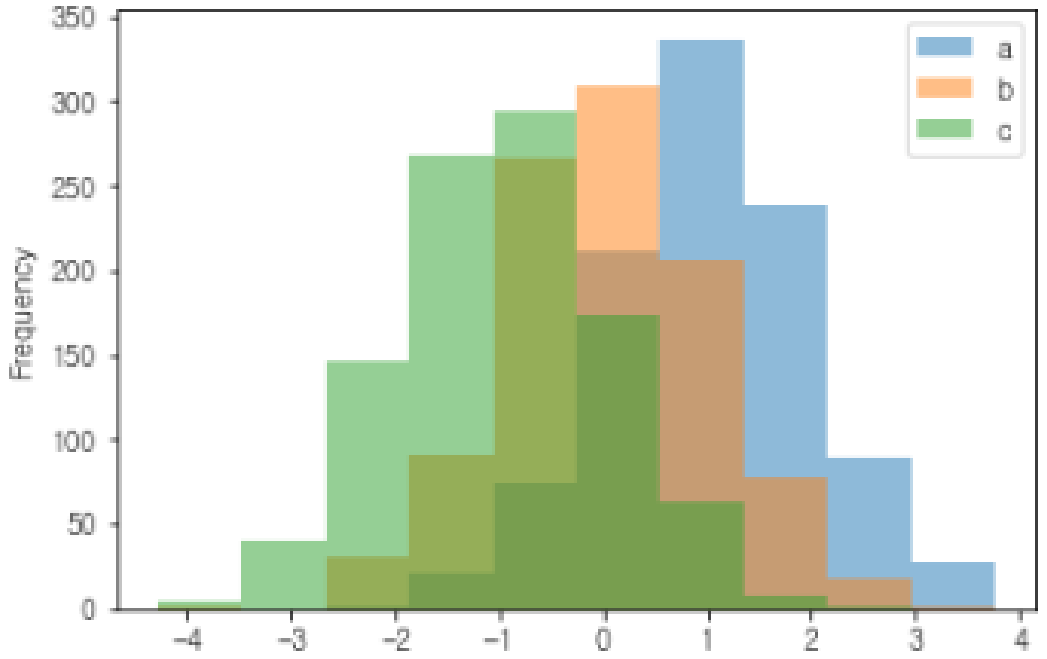
```
df2.plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbad2
```




```
df2.plot.hist(alpha=0.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb0f
```

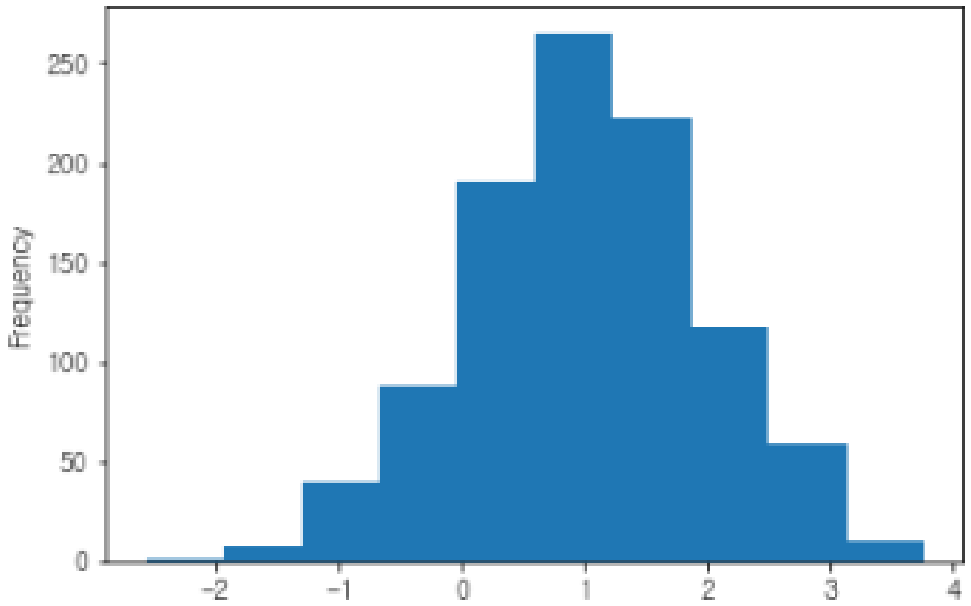


히스토그램의 세부 옵션 중,

(bins = 숫자)로 나눌 구간의 개수를 지정할 수 있음

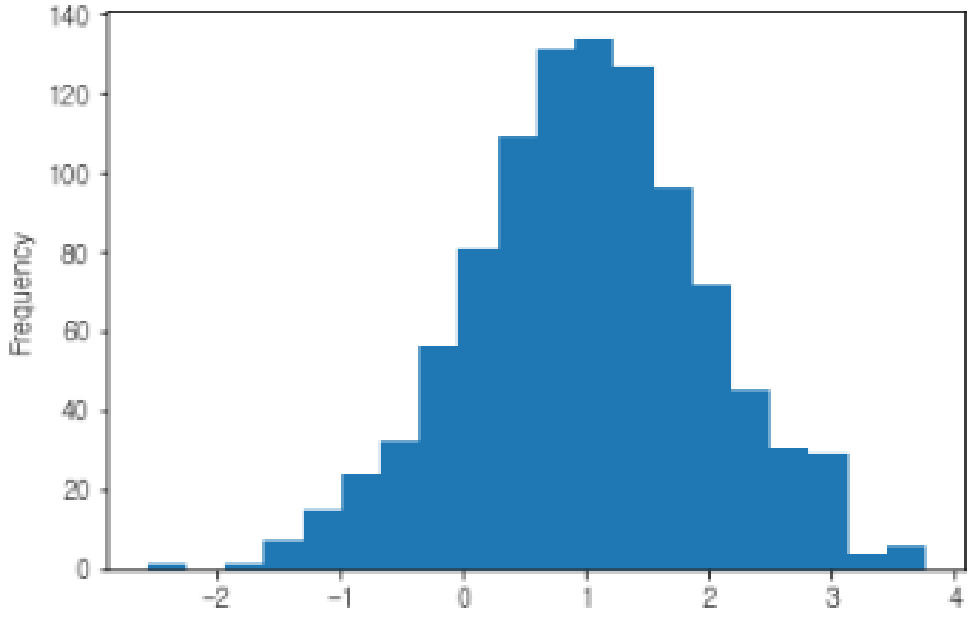
```
df2['a'].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb0e
```




```
df2['a'].plot.hist(bins = 20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb0f(
```



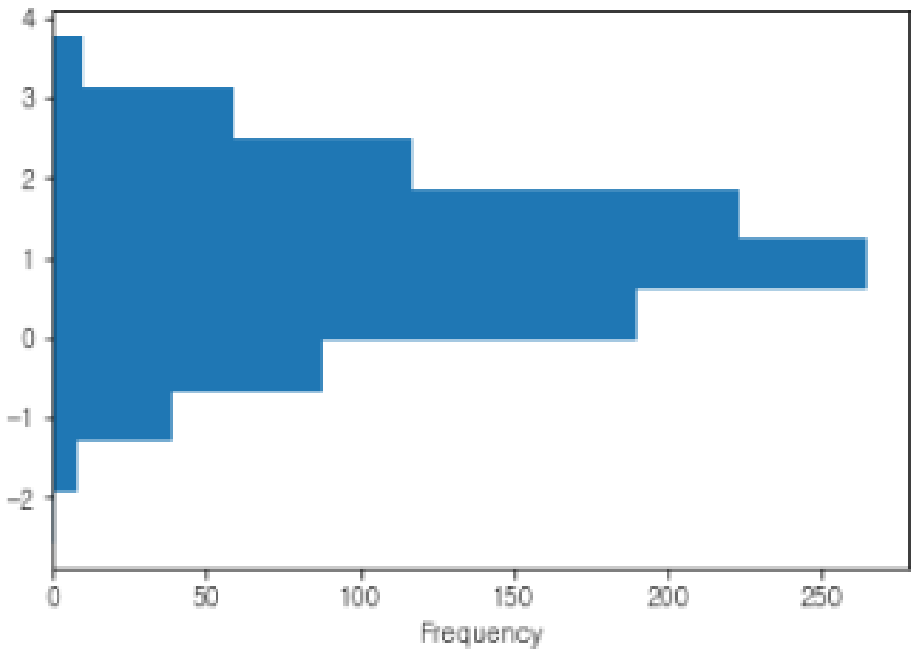
그래프의 세부 옵션 중,

(orientation = 'horizontal')은 수평 즉, 가로방향으로 그래프를 바꿀 수 있음

(cumulative = True)는 값을 누적해 그래프를 그림 (cf: stacked는 열의 누적)

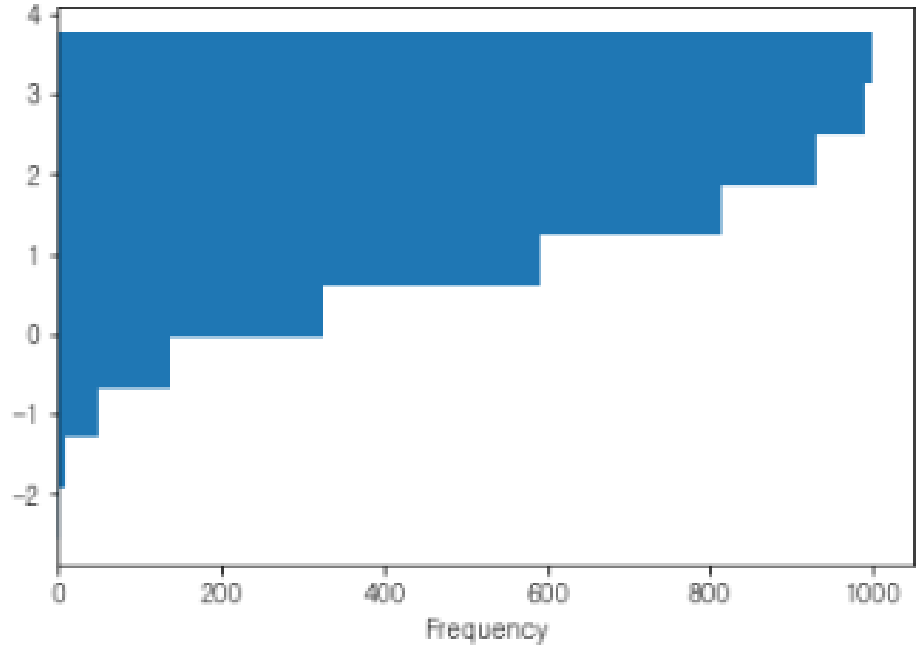
```
df2['a'].plot.hist(orientation='horizontal')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fb
```




```
df2['a'].plot.hist(cumulative=True ,orientation='horizontal')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb108ed68>
```



plt.기준

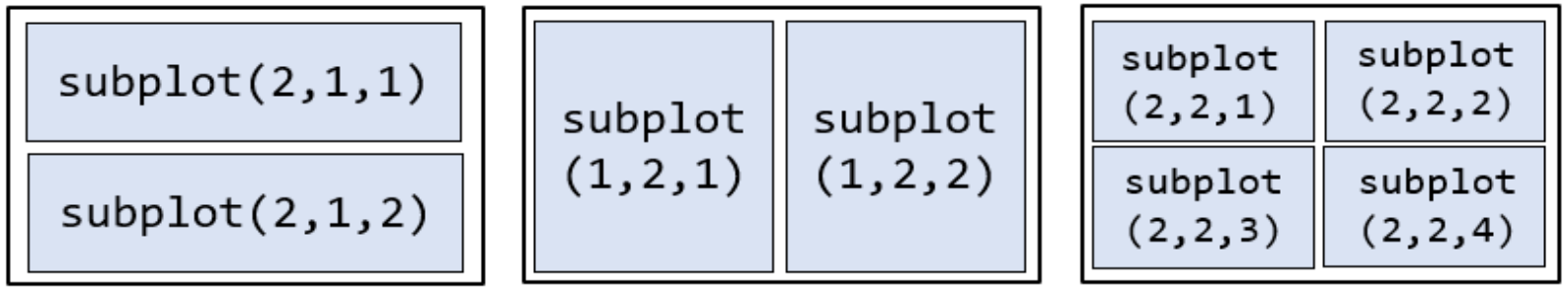
- 그래프가 그려지는 윈도우 속성 설정

```
figure(figsize=(폭, 높이), facecolor='배경색')
```

  - figsize 윈도우의 폭과 높이 지정(인치 단위)
  - facecolor 윈도우 배경색 지정

- 윈도우에 그려질 그래프의 수와 배치 설정

```
subplot(행의수, 열의수, 그래프위치)
```



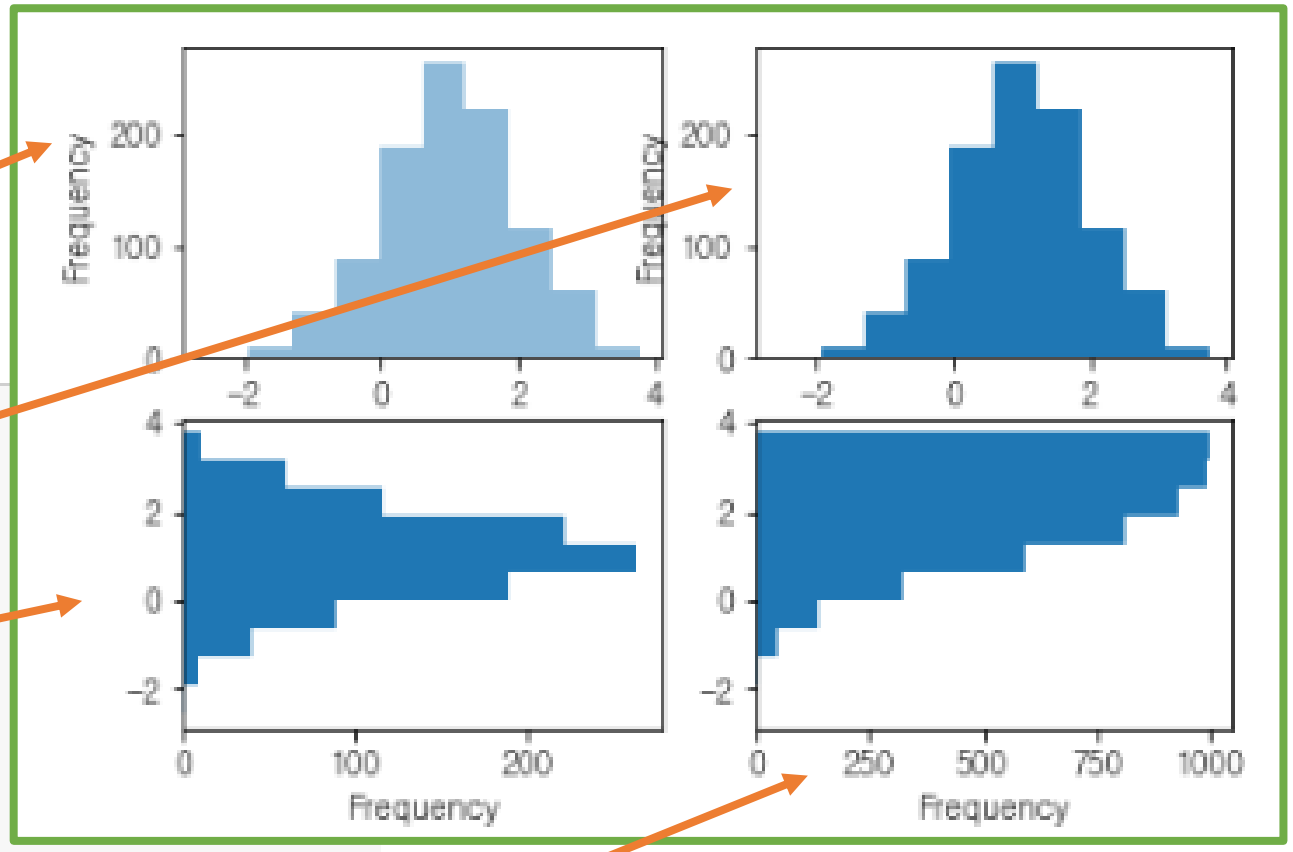
- Subplots간의 간격 설정

```
subplots_adjust(wspace=None, hspace=None)
```

  - wspace subplot간의 폭 간격의 비율 (기본값 0.2)
  - hspace subplot간의 높이 간격의 비율 (기본값 0.2)

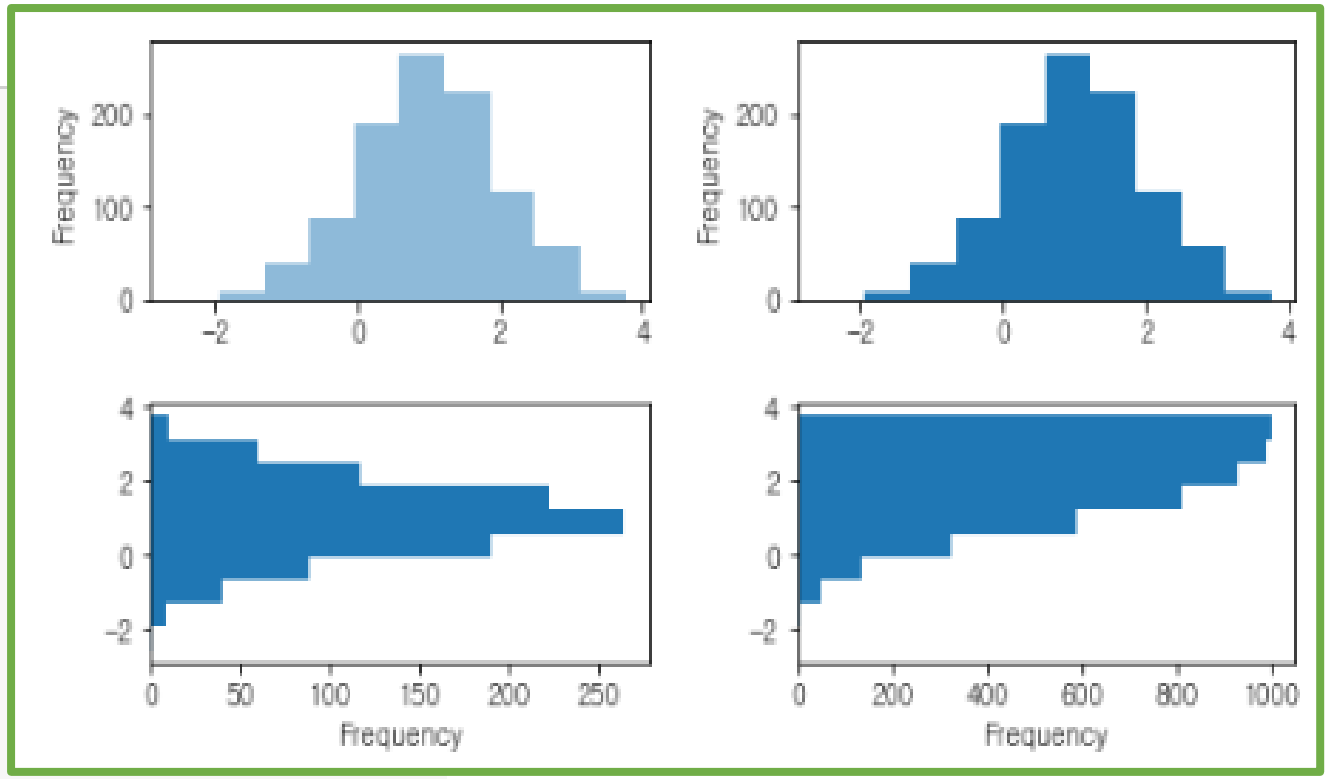
3. 히스토그램 – subplot 그리기

```
plt.subplot(2, 2, 1)
df2['a'].plot.hist(alpha=0.5)

plt.subplot(2, 2, 2)
df2['a'].plot.hist()

plt.subplot(2, 2, 3)
df2['a'].plot.hist(orientation='horizontal')

plt.subplot(2, 2, 4)
df2['a'].plot.hist(cumulative=True ,orientation='horizontal')
```



3. 히스토그램 – subplot 그리기

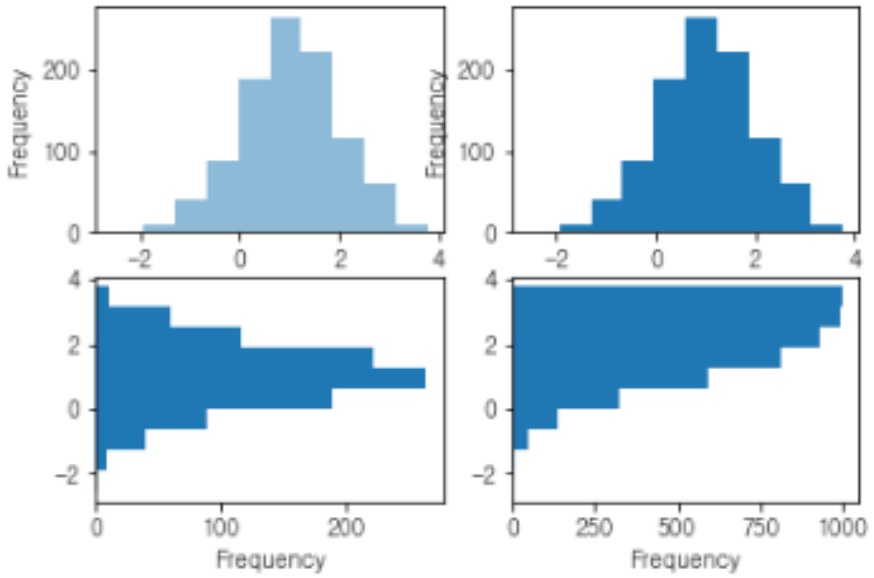
```
plt.figure(figsize=((8, 4)))
plt.subplots_adjust(wspace=0.3, hspace=0.4)


plt.subplot(2, 2, 1)
df2['a'].plot.hist(alpha=0.5)

plt.subplot(2, 2, 2)
df2['a'].plot.hist()

plt.subplot(2, 2, 3)
df2['a'].plot.hist(orientation='horizontal')

plt.subplot(2, 2, 4)
df2['a'].plot.hist(cumulative=True ,orientation='horizontal')
```



3. 히스토그램 – subplot 그리기

```
plt.subplot(2, 2, 1)
df2['a'].plot.hist(alpha=0.5)

plt.subplot(2, 2, 2)
df2['a'].plot.hist()

plt.subplot(2, 2, 3)
df2['a'].plot.hist(orientation='horizontal')

plt.subplot(2, 2, 4)
df2['a'].plot.hist(cumulative=True ,orientation='horizontal')
```

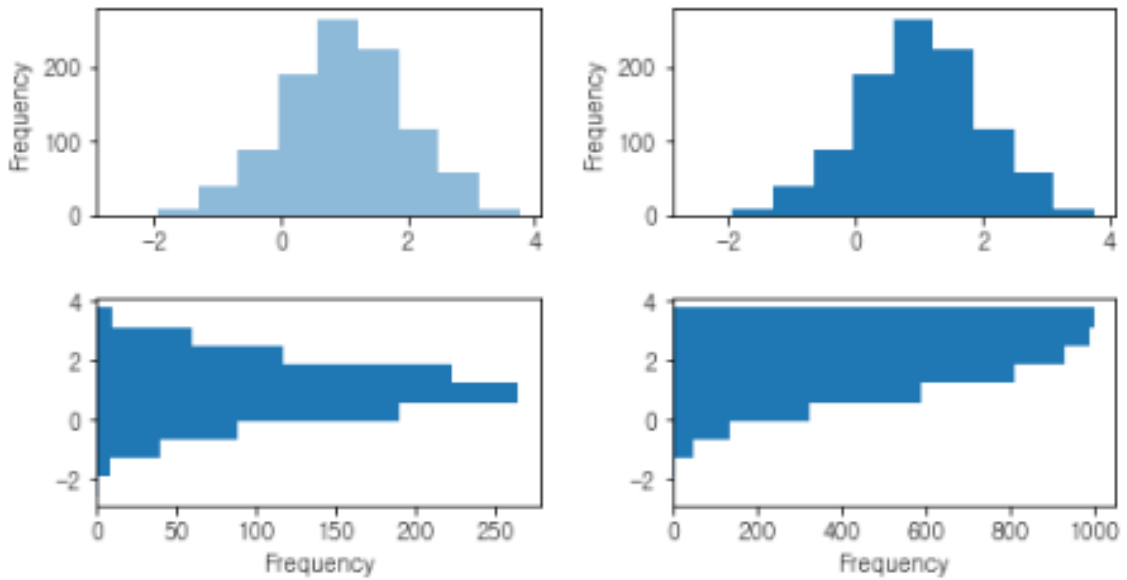
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb27a5b70>
```



그래프
크기, 간격
비교

```
plt.figure(figsize=((8, 4)))
plt.subplots_adjust(wspace=0.3, hspace=0.4)


plt.subplot(2, 2, 1)
df2['a'].plot.hist(alpha=0.5)

plt.subplot(2, 2, 2)
df2['a'].plot.hist()

plt.subplot(2, 2, 3)
df2['a'].plot.hist(orientation='horizontal')

plt.subplot(2, 2, 4)
df2['a'].plot.hist(cumulative=True ,orientation='horizontal')
```

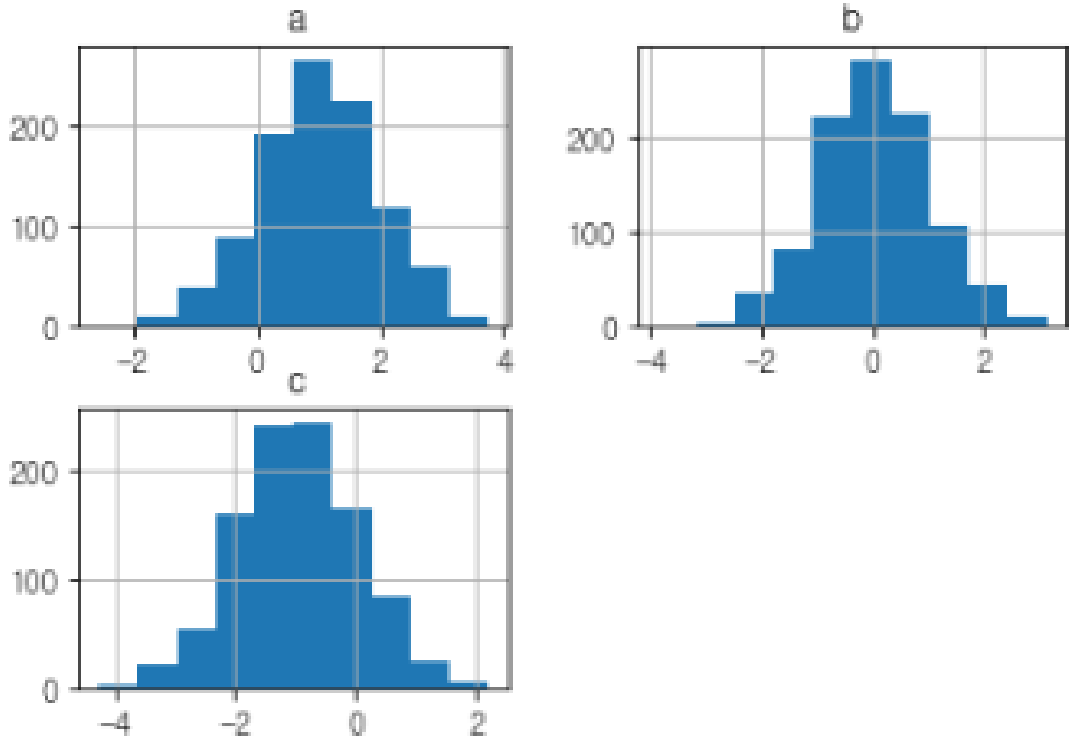
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb28efda0>
```



subplot 따로 지정 안 해도 같은 변수에 있는 개별 열 히스토그램은 여러 개 그리기 가능

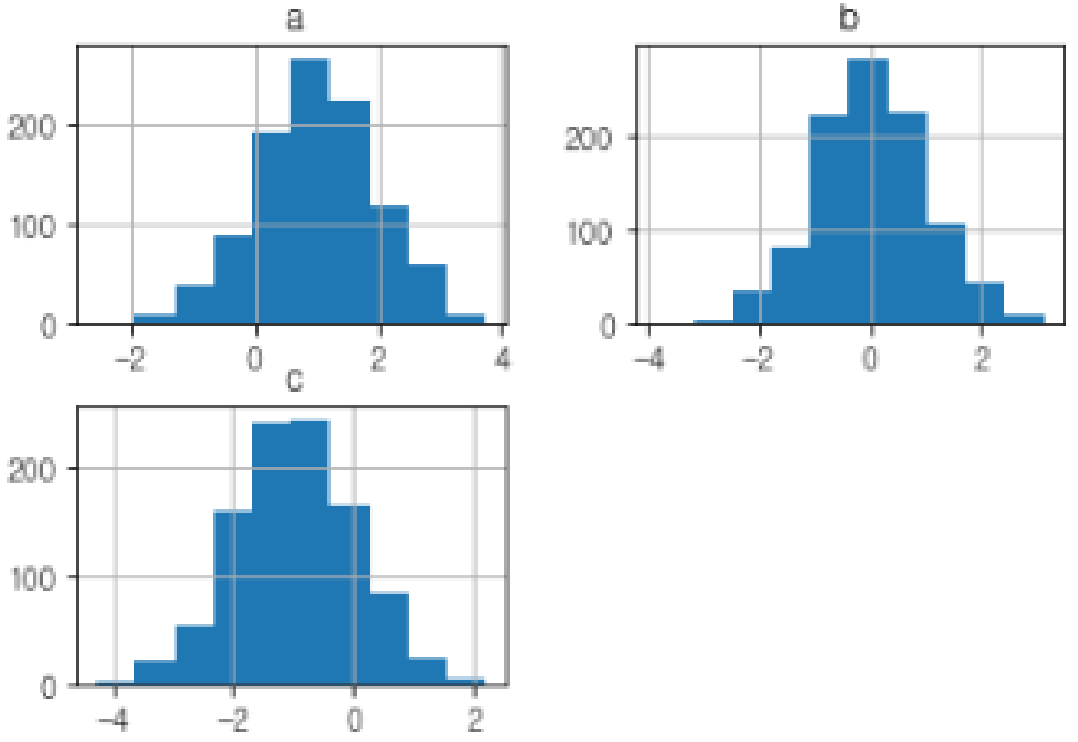
```
df2[ ['a', 'b', 'c'] ].hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot o
        <matplotlib.axes._subplots.AxesSubplot o
       [<matplotlib.axes._subplots.AxesSubplot o
        <matplotlib.axes._subplots.AxesSubplot o
      dtype=object)
```




```
df2.hist(figsize = (6, 4))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot c
        <matplotlib.axes._subplots.AxesSubplot c
       [<matplotlib.axes._subplots.AxesSubplot c
        <matplotlib.axes._subplots.AxesSubplot c
      dtype=object)
```



cf) kde 커널밀도함수 (Kernel Density Estimate plot)

- 히스토그램은 이산형으로 그래프가 나옴(끊기는 형태), Density plot은 부드러운 곡선형태로 그래프 그려 줌
- 곡선으로 만들때 kde방법으로 추정 함
- 커널 밀도 추정(KDE)은 임의 변수의 확률 밀도 함수(PDF)를 추정하는 비모수적 방법
- 가우스 커널을 사용, 자동 대역폭 결정 포함
- 커널 밀도 추정치는 적절한 커널을 사용, 매끄럽고 연속성과 같은 속성을 부여할 수 있음
- 커널 함수는 원점을 중심으로 대칭, 적분값이 1인 함수

정규분포

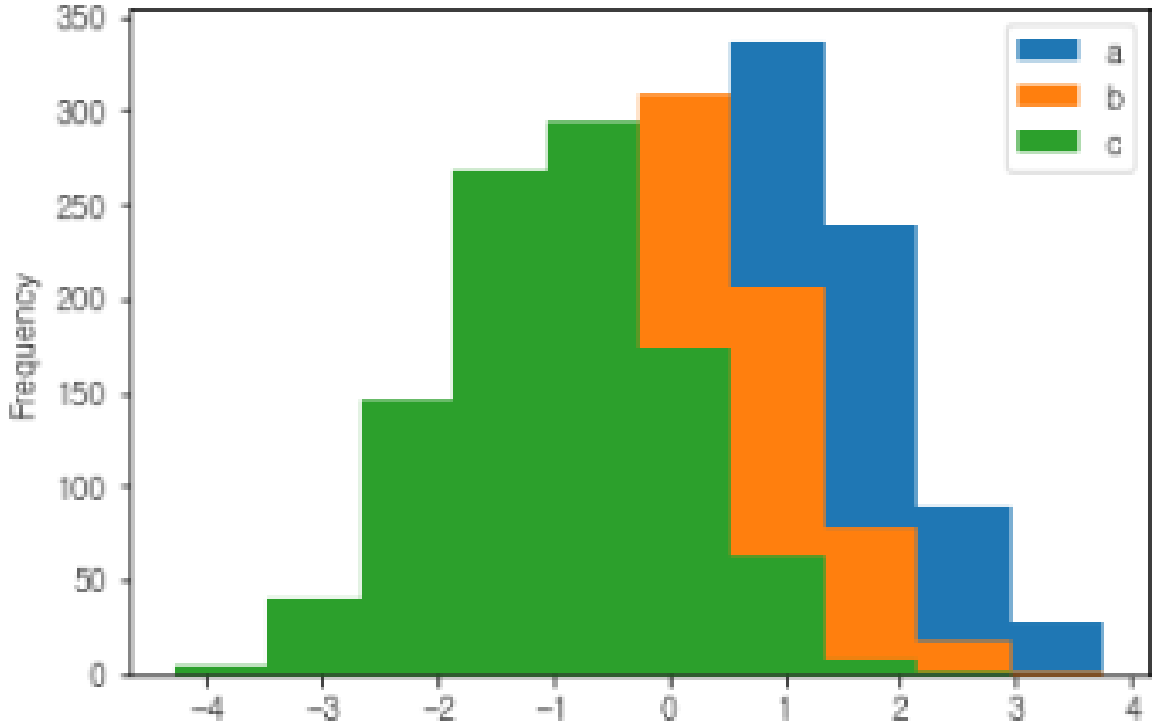
- 정규분포(normal distribution) 또는 가우시안 분포(Gaussian distribution)는 연속 확률 분포
- 정규분포는 중심극한정리에 의해 독립적인 확률변수들의 평균은 정규분포에 가까워지는 성질이 있음

https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html#density-plot
https://seaborn.pydata.org/examples/distplot_options.html

번외. 히스토그램 과 확률분포도

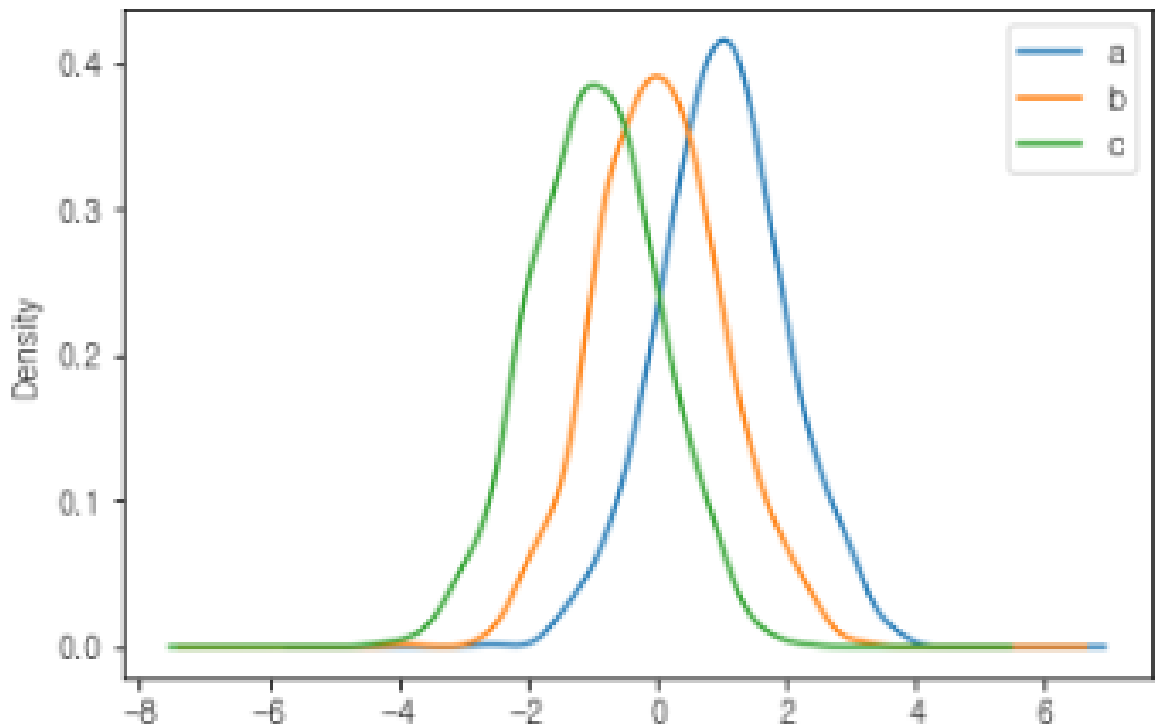
```
df2.plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb22
```




```
df2.plot.density()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb4c
```



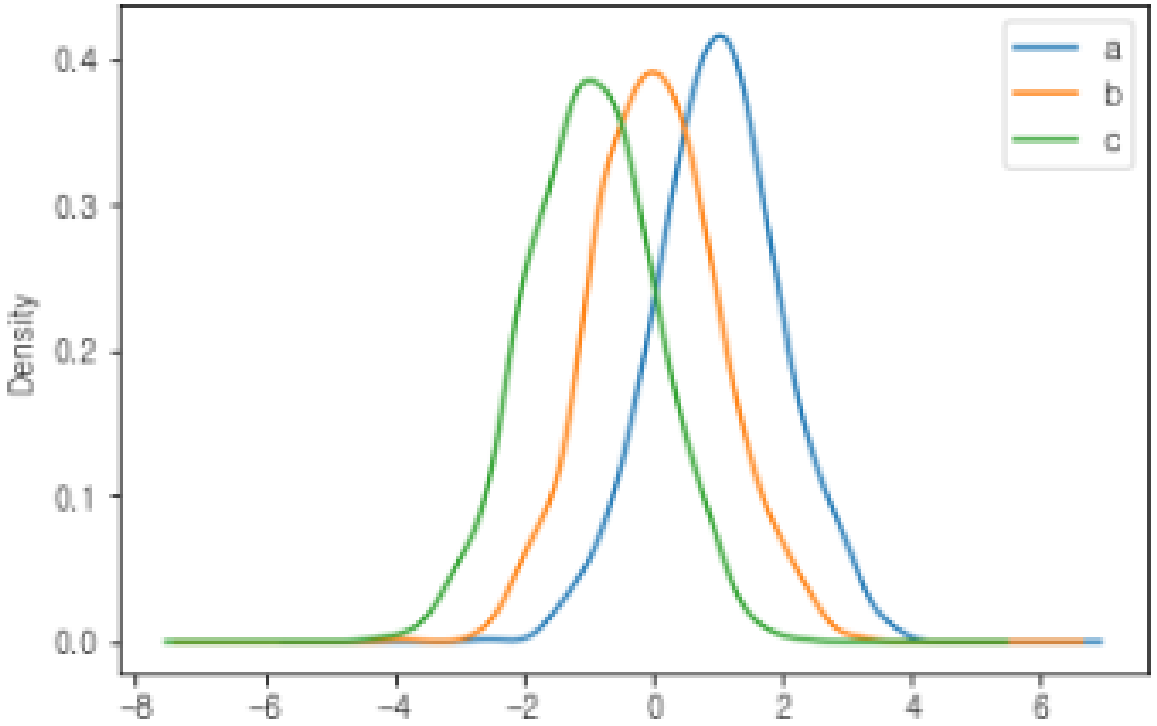
density 자체는 밀도

번외. 히스토그램 과 확률분포도

밀도추정때 KDE를 애용

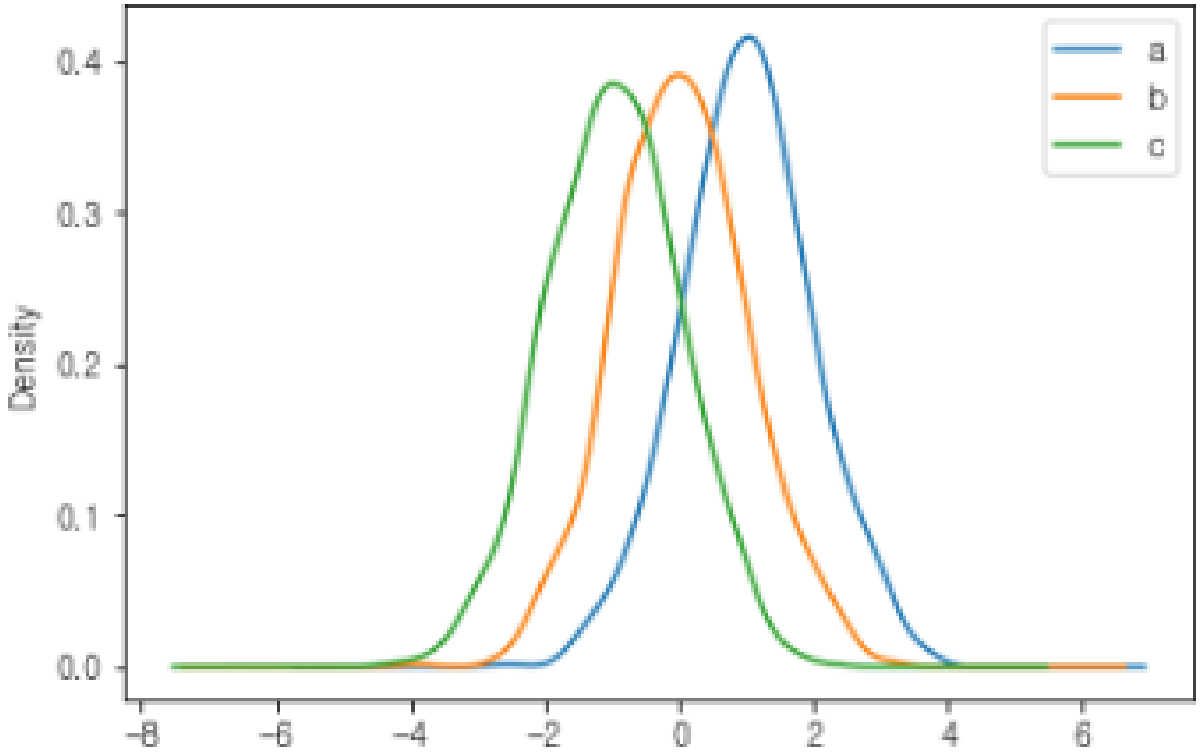
```
df2.plot.density()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb4cb
```



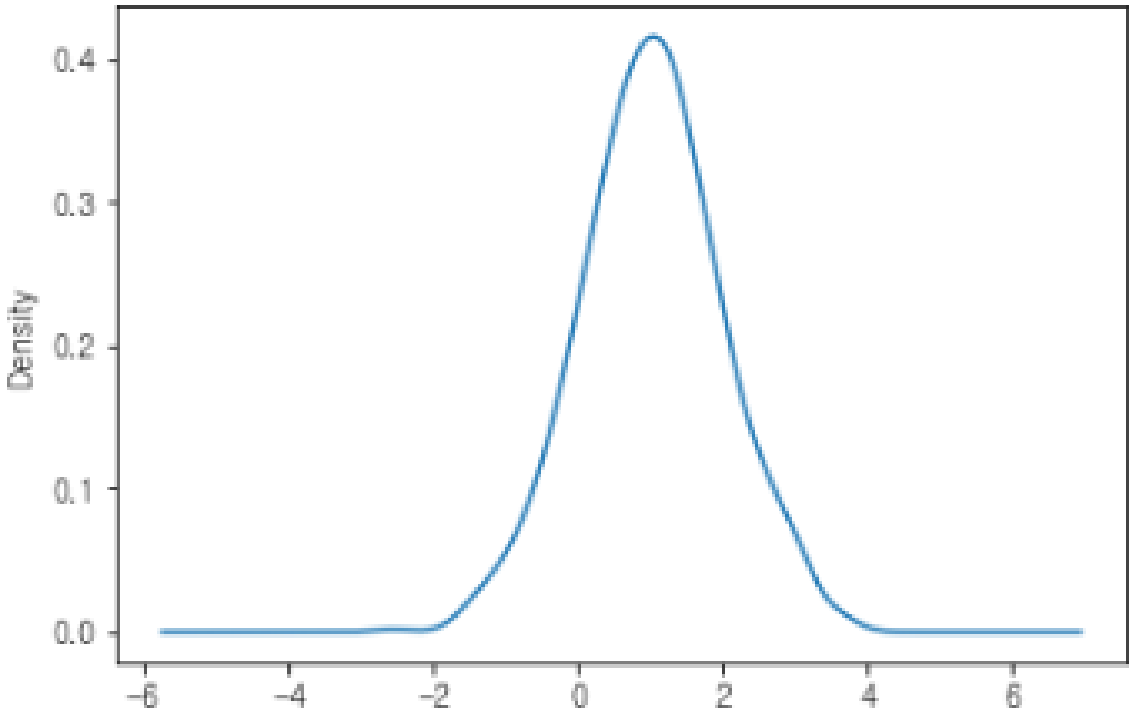

```
df2.plot.kde()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb45
```

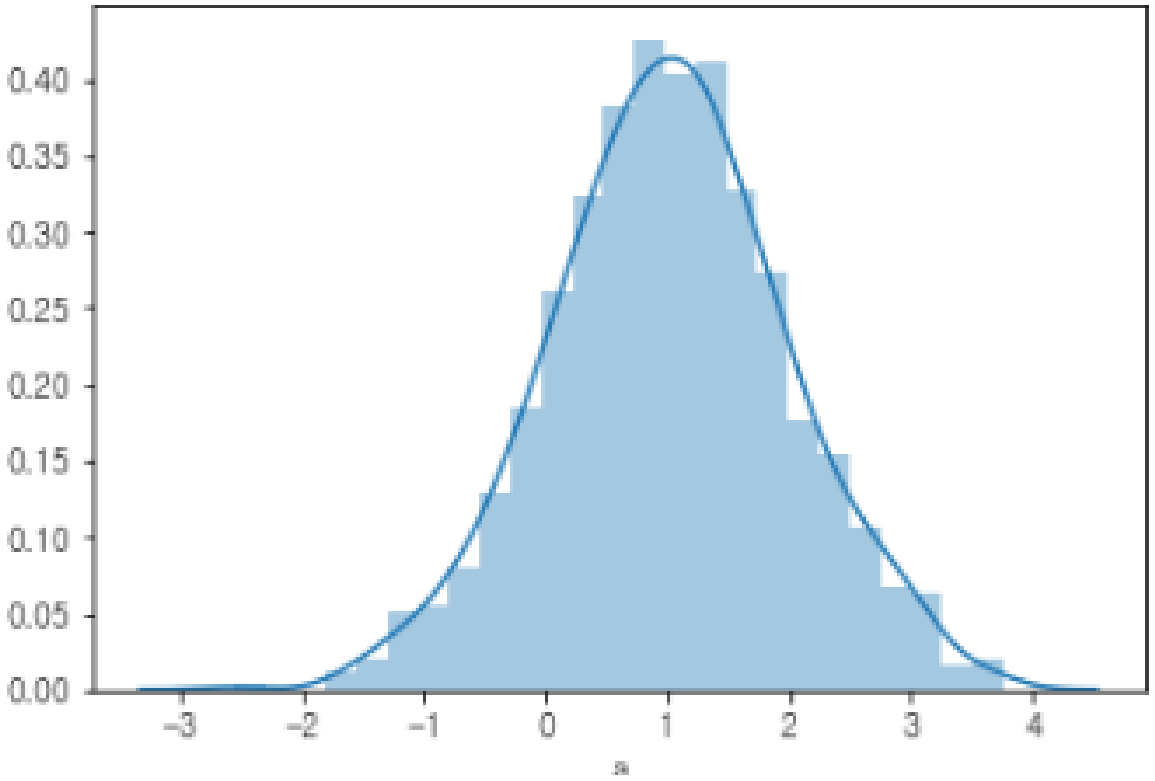


번외. 히스토그램 과 확률분포도

```
df2['a'].plot.density()
```



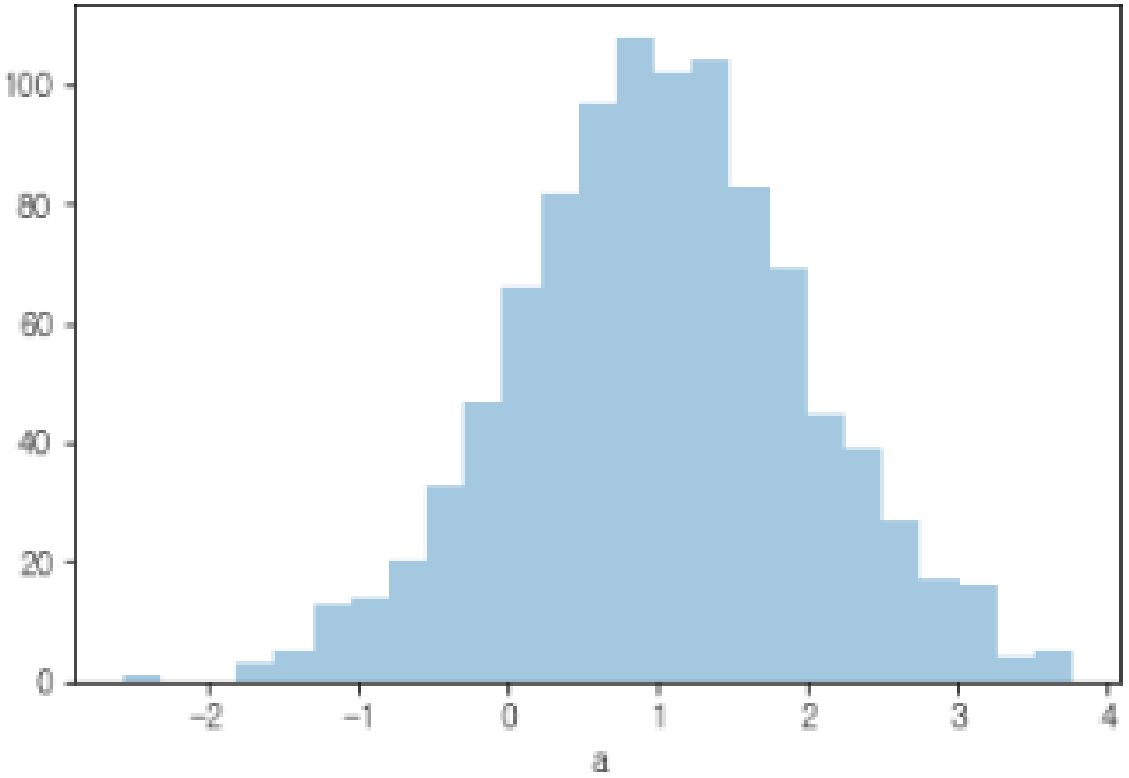

```
sns.distplot(df2['a'])
```



번외. 히스토그램 과 확률분포도

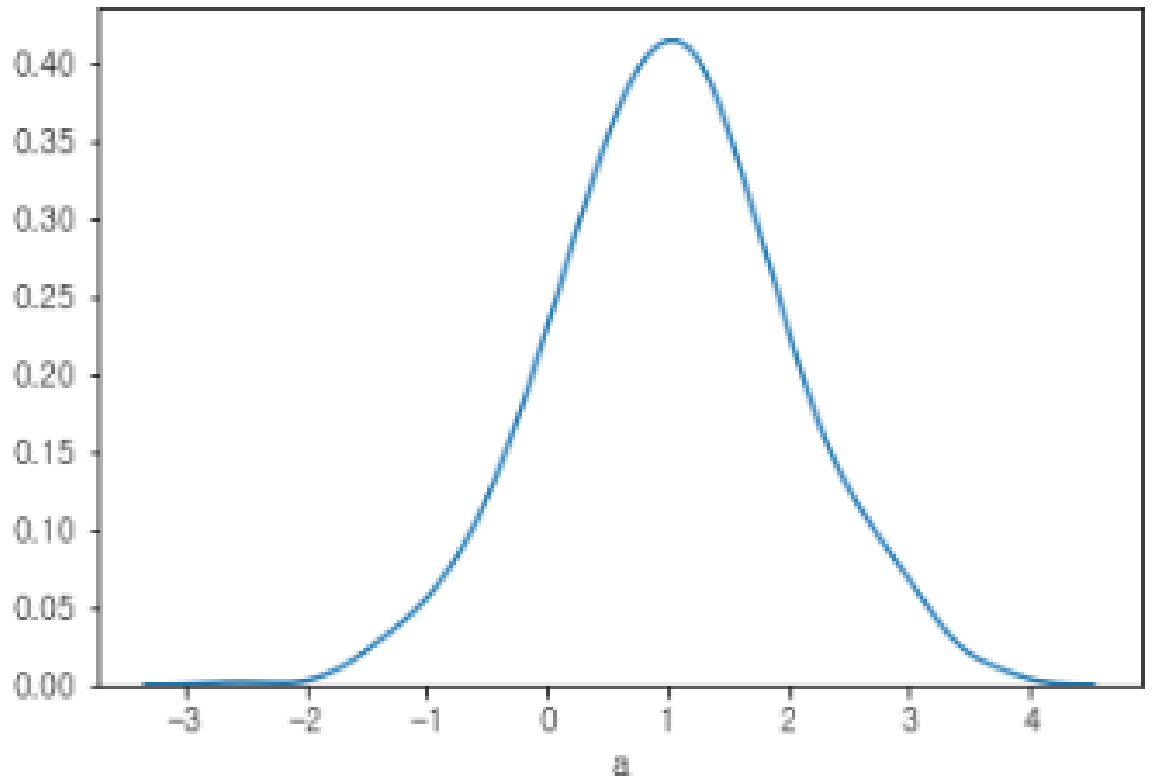
```
sns.distplot(df2['a'], kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb
```




```
sns.distplot(df2['a'], hist=False)
```

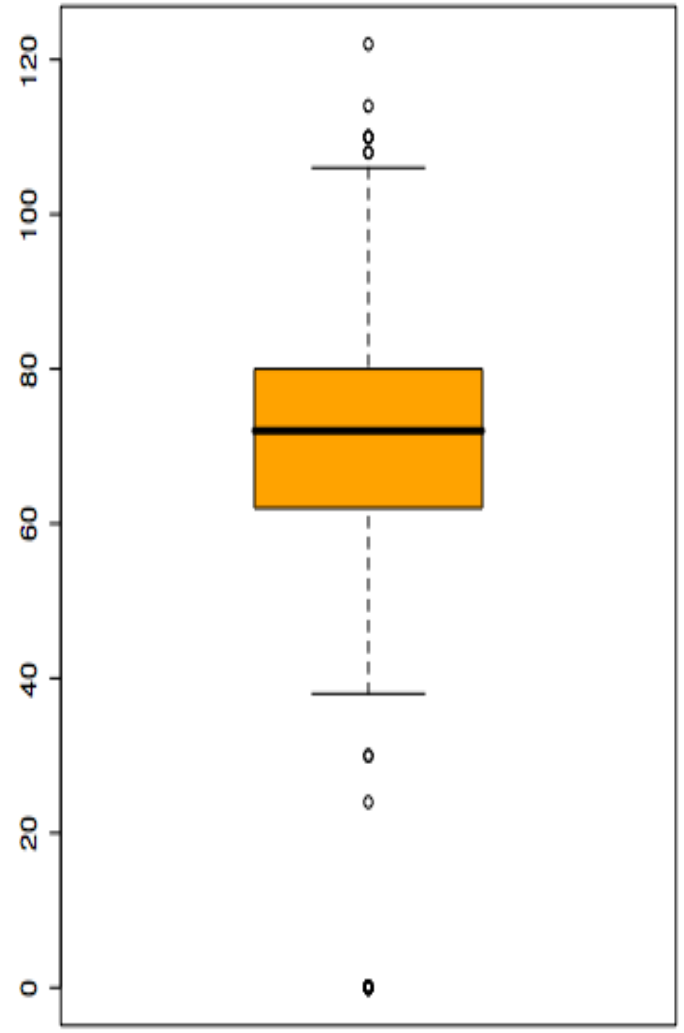
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fbb4
```



4. 상자수염 그래프

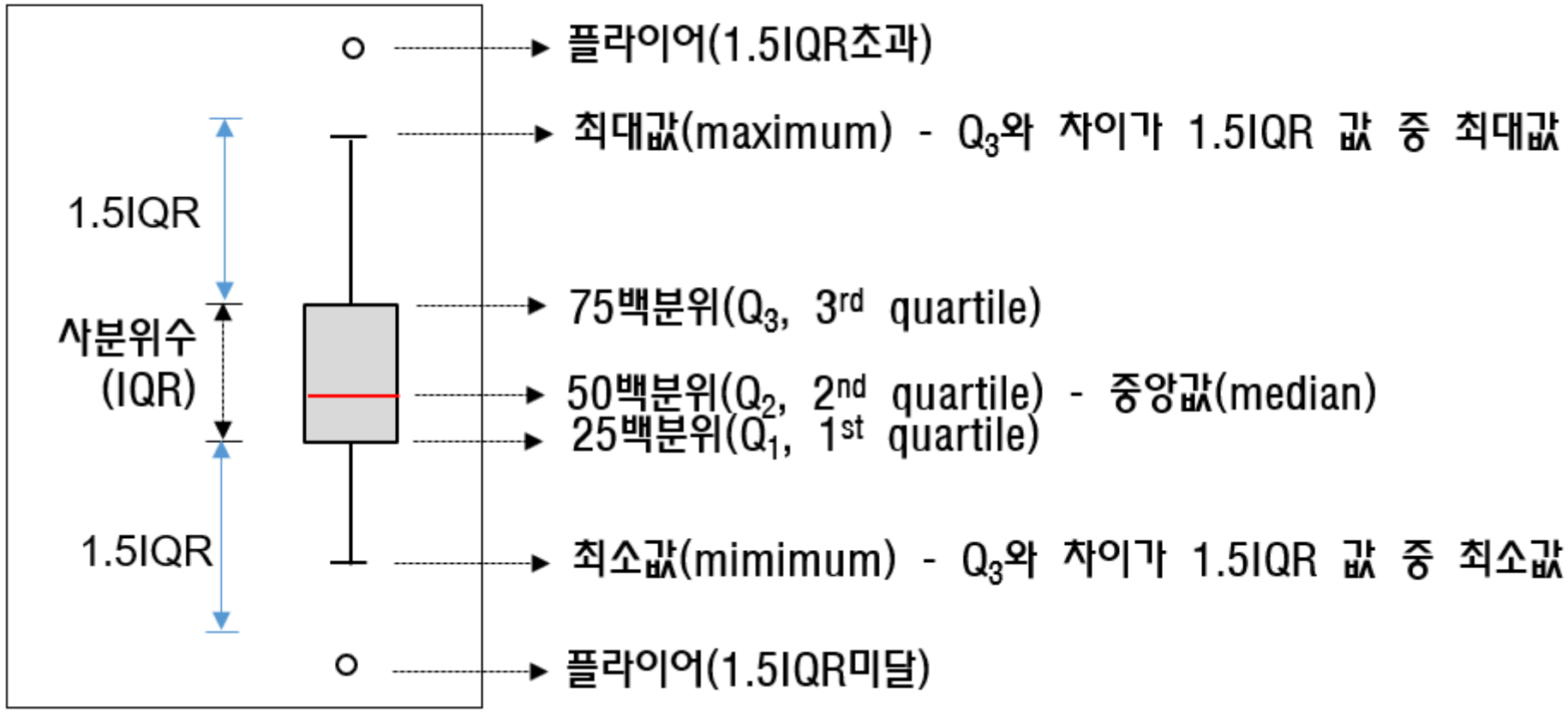
- 상자 그림(Box plot)
  - 단일 변수 측정값의 퍼짐도 분석을 하기 위한 그래프
  - 단일 변수에 대한 다음과 같은 정보를 얻을 수 있음
    - 중앙값(median)
    - 사분위수(IQR, Interquartile Range)
    - 이상값(outlier)
    - 범위(range)
    - 왜도(skewness)

- 상자 그림의 문제점
  - 여러 개의 동일한 데이터가 존재해도 겹쳐서 표현(overplotting)
  - 분포의 모양을 파악하는 데 제한



https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html#box-plots

4. 상자수염 그래프 – boxplot 그리기

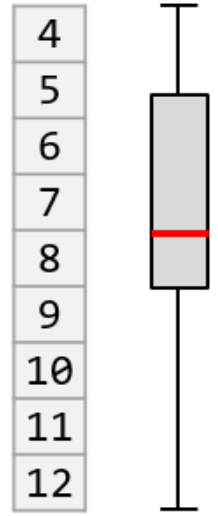


[12, 6, 4, 9, 8, 4, 9, 8, 5, 9, 8, 10]

sorting → [4, 4, 5, 6, 8, 8, 8, 9, 9, 9, 10, 12]

25백분위(Q_1) = 5.5　중앙값(Q_2) = 8　75백분위(Q_3) = 9

사분위수(IQR) = 9 – 5.5
= 3.5

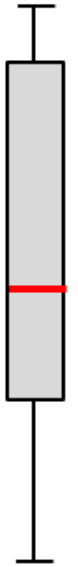


[6, 3, 9, 8, 4, 10, 8, 4, 12, 8, 10]

sorting → [3, 4, 4, 6, 8, 8, 8, 9, 10, 10, 12]

25백분위(Q_1) = 4　중앙값(Q_2) = 8　75백분위(Q_3) = 10

사분위수(IQR) = 10 – 4
= 6

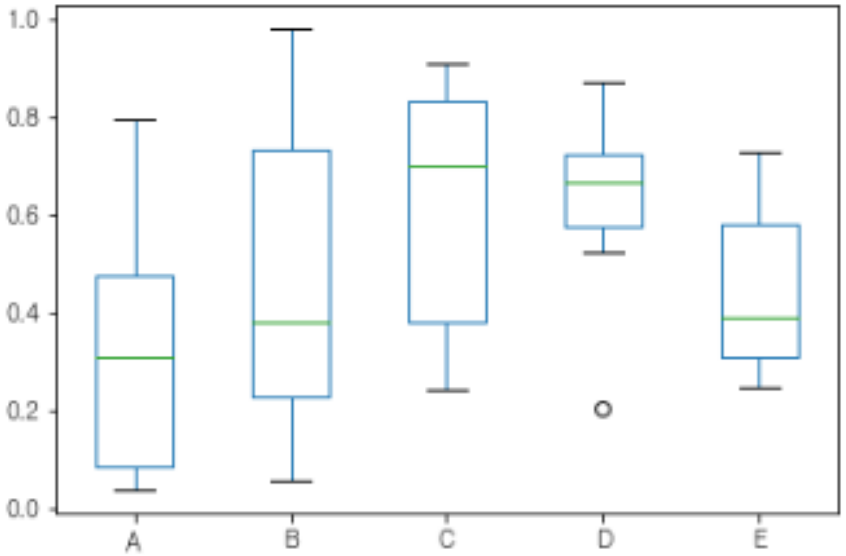


plt.기준

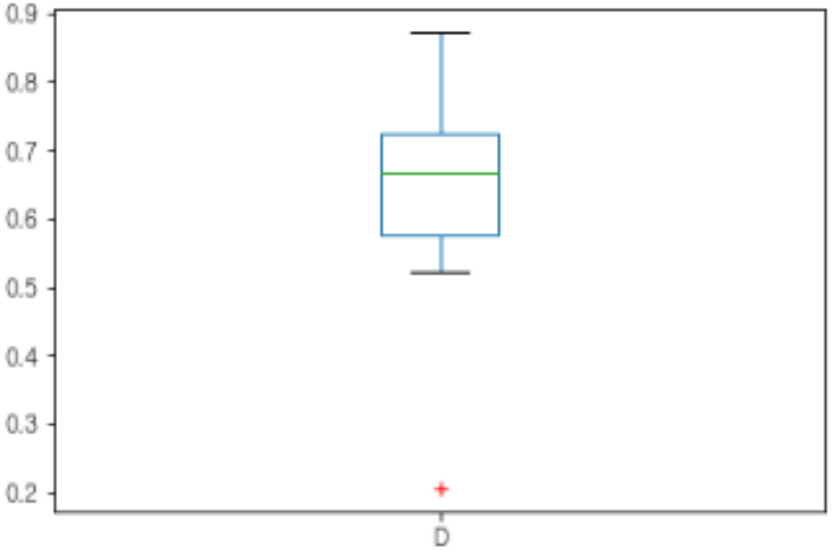
```
boxplot(x, sym=None, labels=None, meanline=False, vert=True)
```

- *x* 데이터, numpy 배열
- sym flier points들에 대한 기호
- labels 데이터에 대한 레이블(배열) 설정
- meanline 평균에 대한 라인 그리기 설정
- vert 수직, 수평 상자 그림 그리기

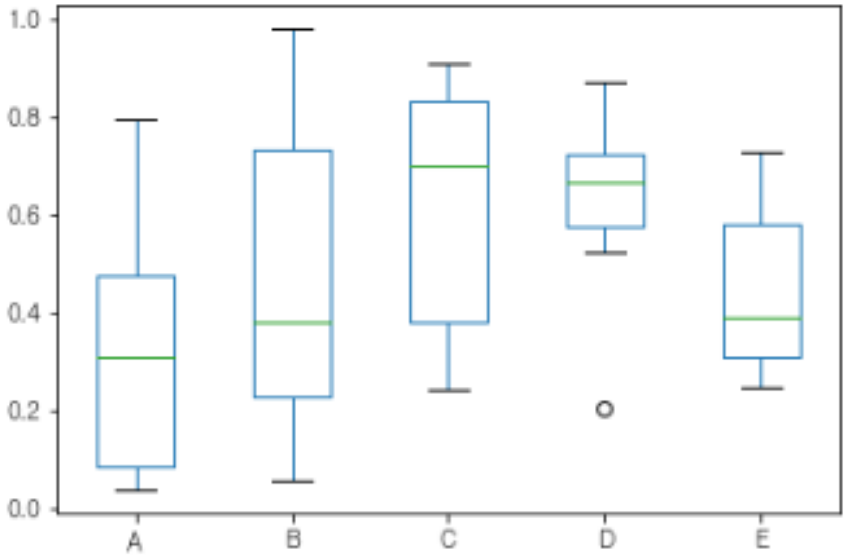
```
df.plot.box()
```




```
df['D'].plot.box(sym='r+')
```

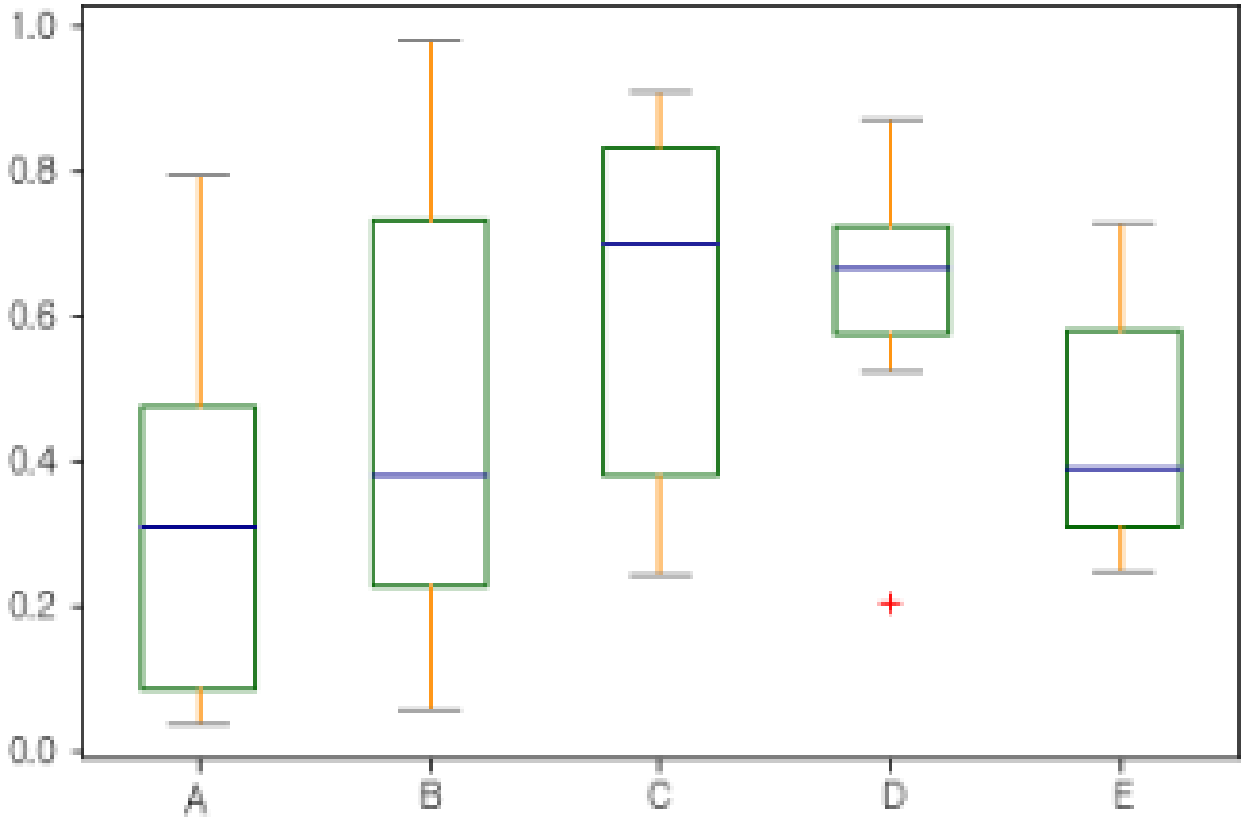


4. 상자수염 그래프 – boxplot 그리기

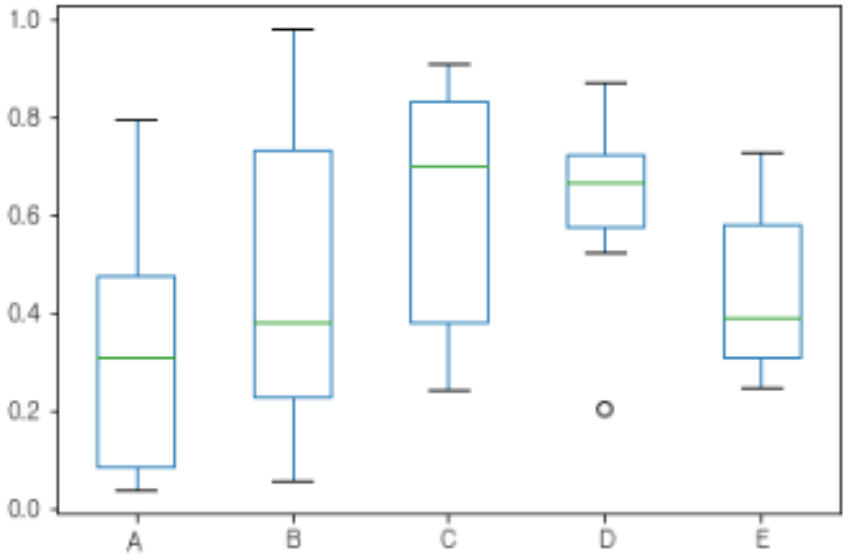



```
color = {'boxes': 'DarkGreen', 'whiskers': 'DarkOrange',
         'medians': 'DarkBlue', 'caps': 'Gray'}
```

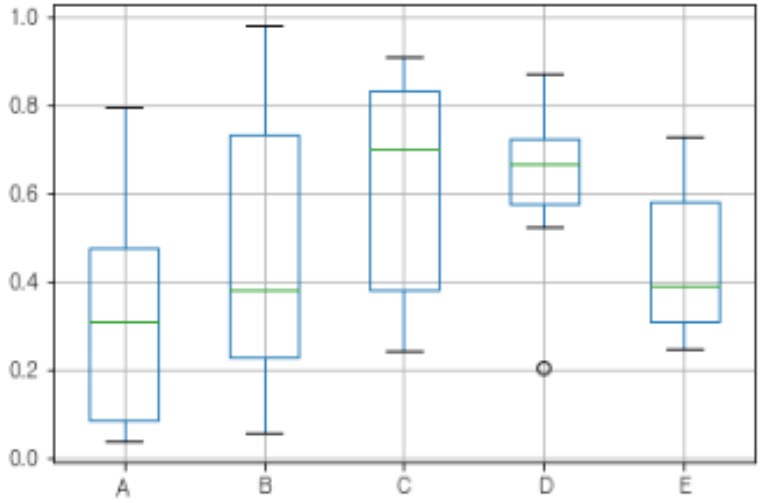
```
df.plot.box(color=color, sym='r+')
```



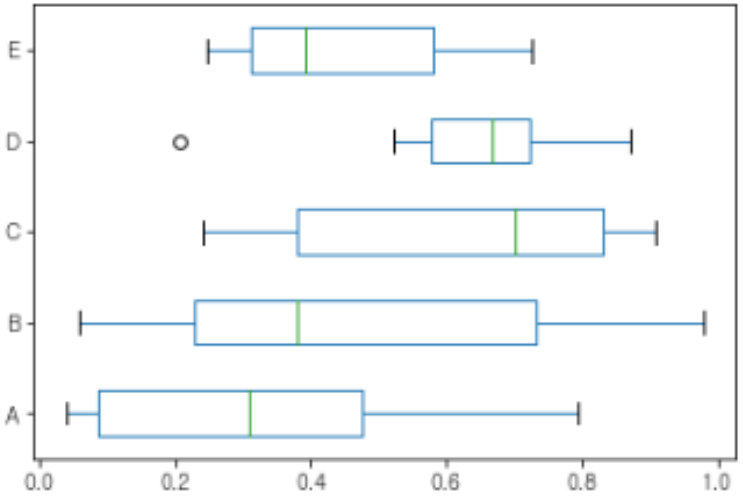
4. 상자수염 그래프 – boxplot 그리기

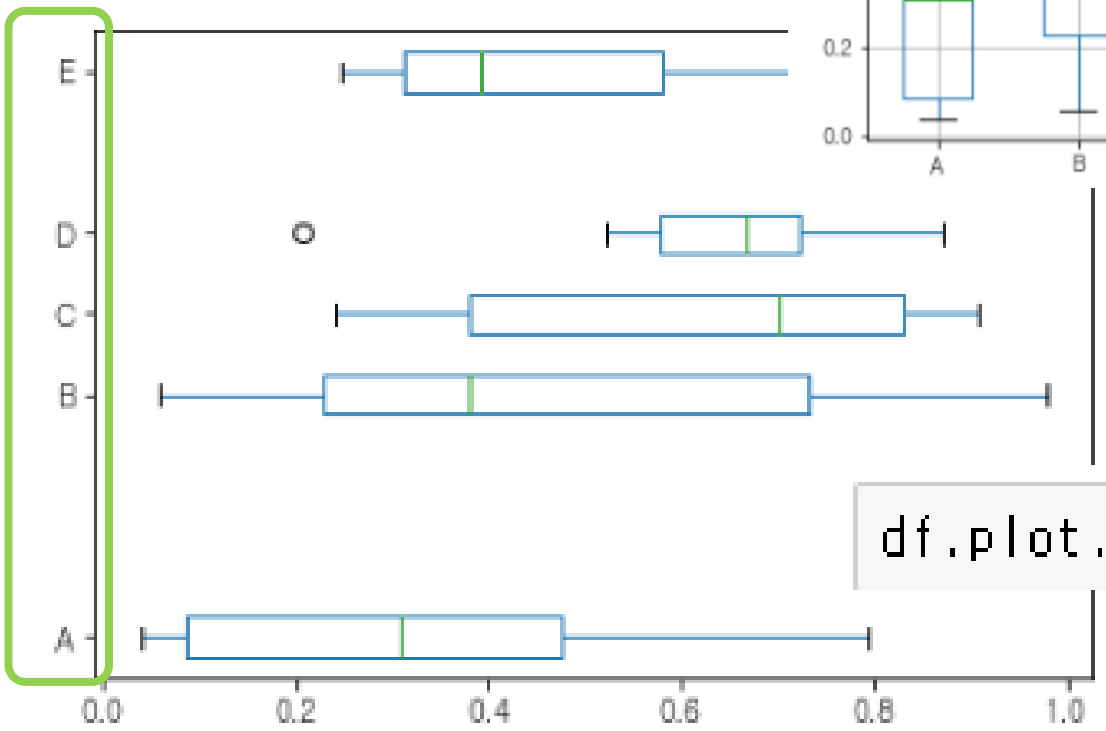



```
df.boxplot(grid = True)
```




```
df.plot.box(vert=False)
```



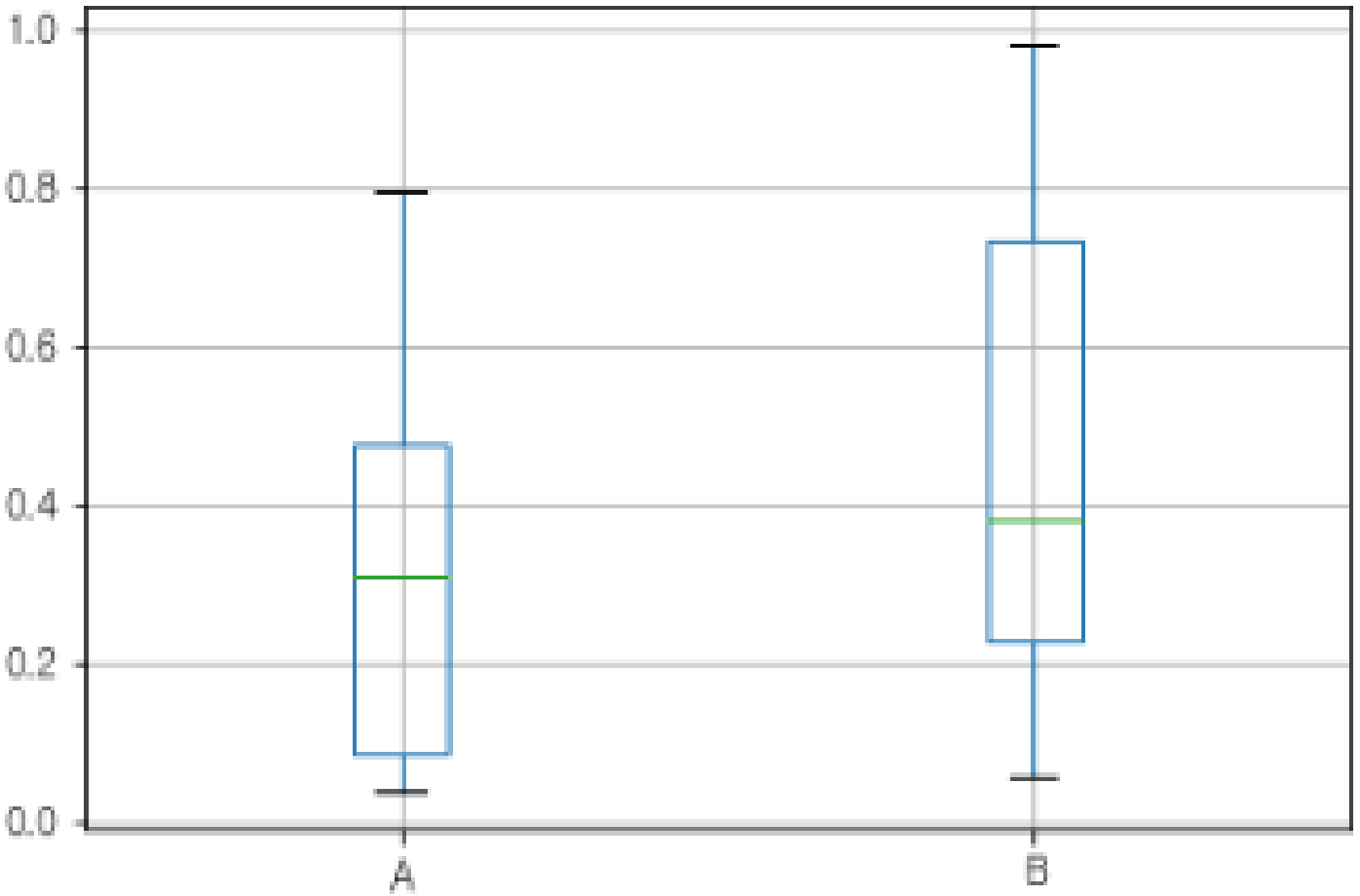



```
df.plot.box(vert=False, positions=[1, 4, 5, 6, 8])
```

4. 상자수염 그래프 – boxplot 그리기

	A	B	X
0	0.038952	0.856496	G
1	0.080483	0.231636	H
2	0.104182	0.425848	H
3	0.713791	0.058862	G
4	0.496344	0.127552	G
5	0.221893	0.228571	G
6	0.795700	0.979995	H
7	0.395632	0.774485	H
8	0.419510	0.604399	G
9	0.058230	0.335766	H

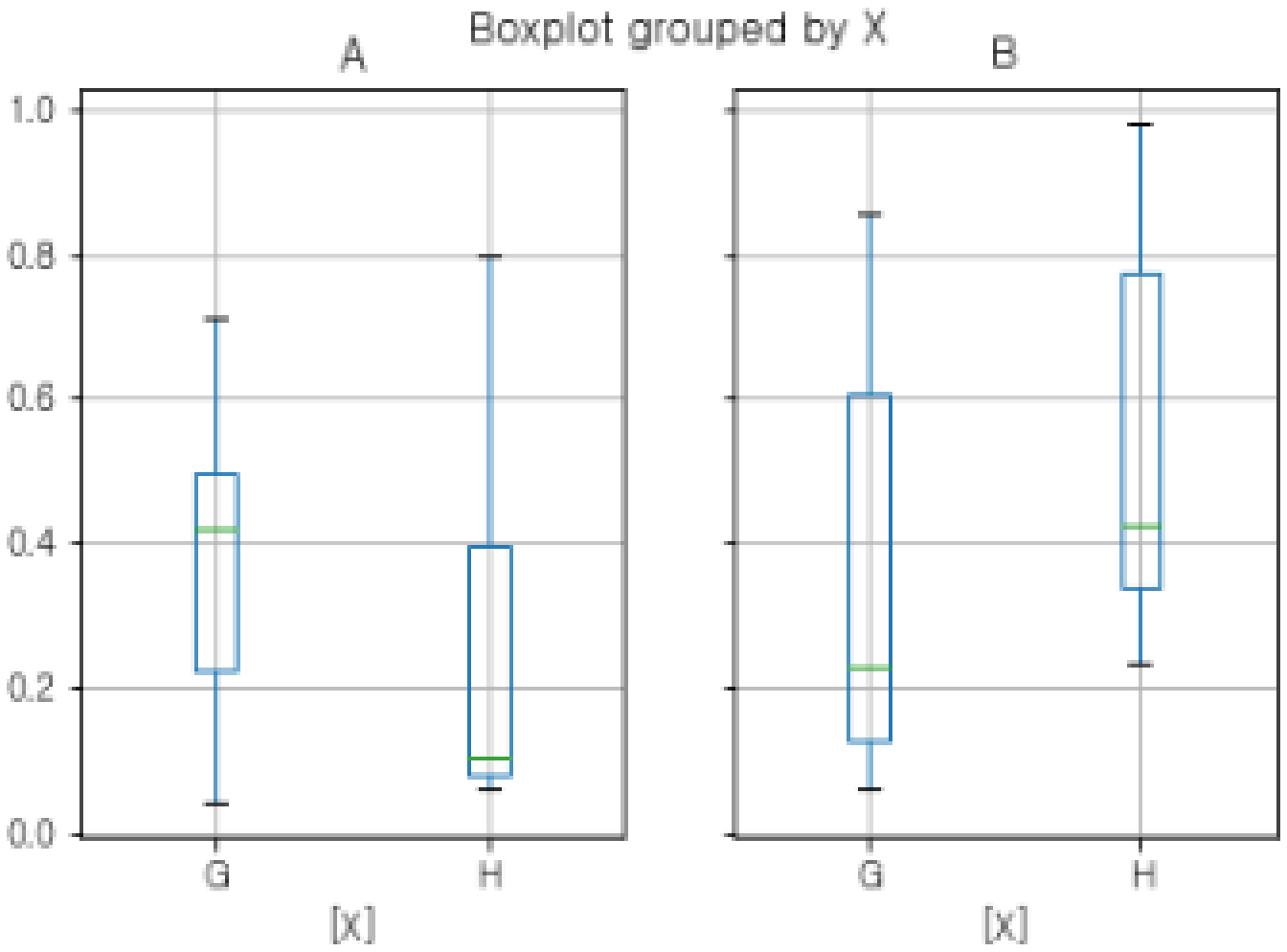
```
df2.boxplot()
```



4. 상자수염 그래프 – boxplot 그리기

	A	B	X
0	0.038952	0.856496	G
1	0.080483	0.231636	H
2	0.104182	0.425848	H
3	0.713791	0.058862	G
4	0.496344	0.127552	G
5	0.221893	0.228571	G
6	0.795700	0.979995	H
7	0.395632	0.774485	H
8	0.419510	0.604399	G
9	0.058230	0.335766	H

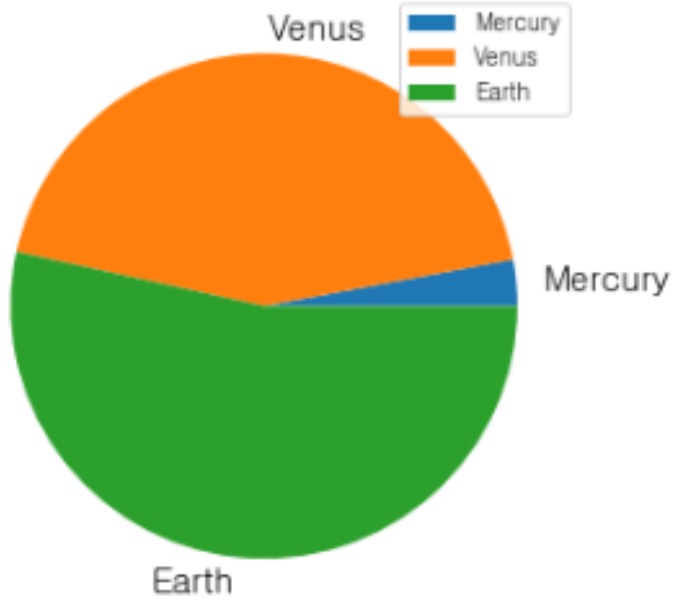
```
df2.boxplot(by = 'X')
```



5. 파이 그래프

파이 차트(pie chart)

- 전체에서 변수의 값이 차지하는 비율을 부채꼴로 나타낸 그래프
- 각 변수의 값이 전체에서 차지하는 비율을 한눈에 파악 가능



https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html#pie-plot

https://pandas.pydata.org
/pandas-docs/stable/reference/api/pandas.DataFrame.plot.pie.html#pandas.DataFrame.plot.pie

https://seaborn.pydata.org/
* seaborn에서는 파이 차트 지원 안함 https://github.com/mwaskom/seaborn/issues/766

파이 차트 그리기 plt.기준

```
pie(x, explode=None, labels=None, rotatelabels=False, colors=None,
        autopct=None, pctdistance=None, startangle=None, shadow=False)
```

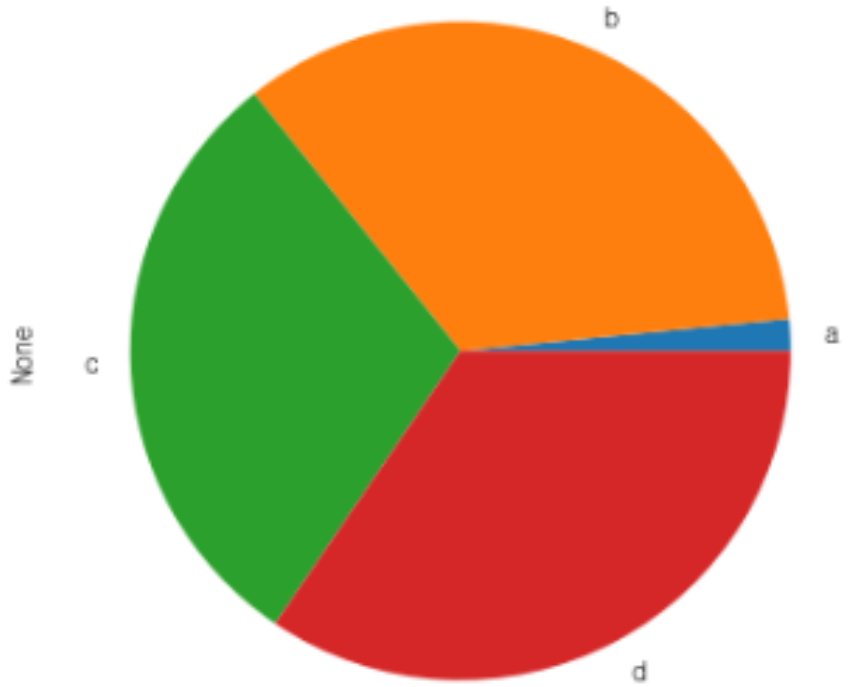
pctdistance autopct에 의해 만들어지는 슬라이스의 중심과 텍스트 시작간의 비율
rotatelabels 대응되는 조각에 맞추어 레이블을 회전
pctdistance autopct에 의해 만들어지는 슬라이스의 중심과 텍스트 시작간의 비율
shadow 파이 차트의 그림자 설정

인수	설명
explode	파이 조각이 돌출되는 크기, 0이면 돌출이 되지 않는다.
labels	파이 조각 외부에 보여 지는 라벨
autopct	파이 조각의 전체 대비 백분율을 표시해준다.
shadow	파이 차트의 그림자 효과 유무
startangle	파이 조각의 시작 위치를 의미한다. 90이면 12시 방향이다.(반시계 방향)

```
ss
```

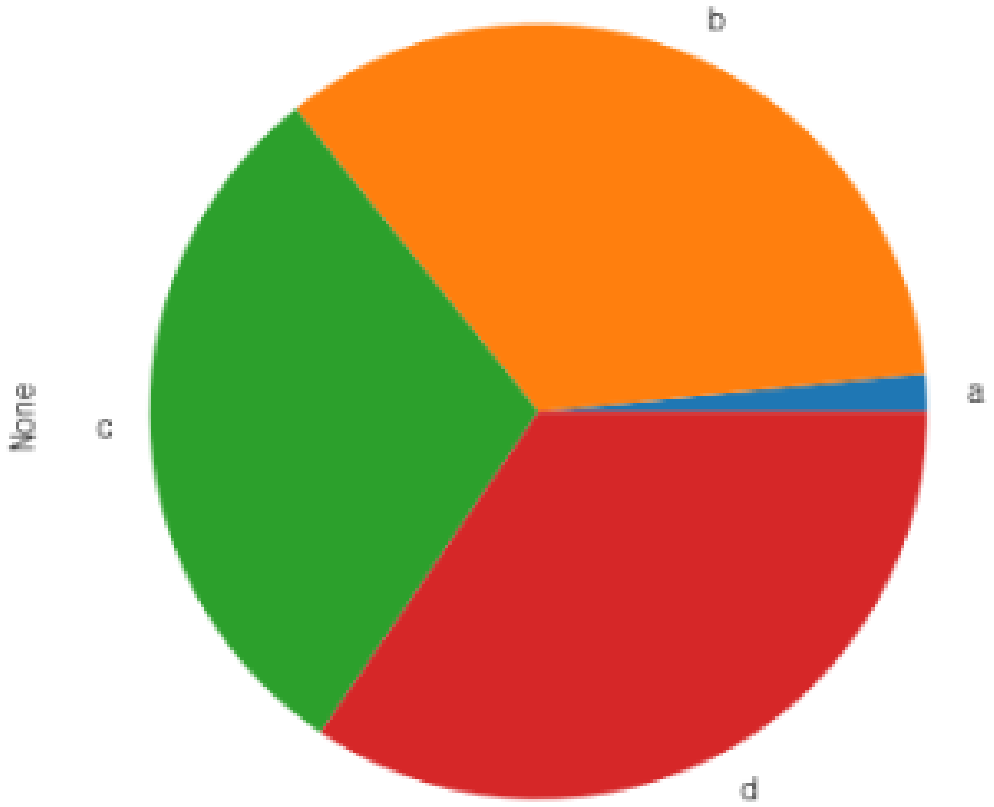
```
a    0.116855
b    2.569488
c    2.239501
d    2.591784
```

```
ss.plot.pie(figsize = (6, 6))
```



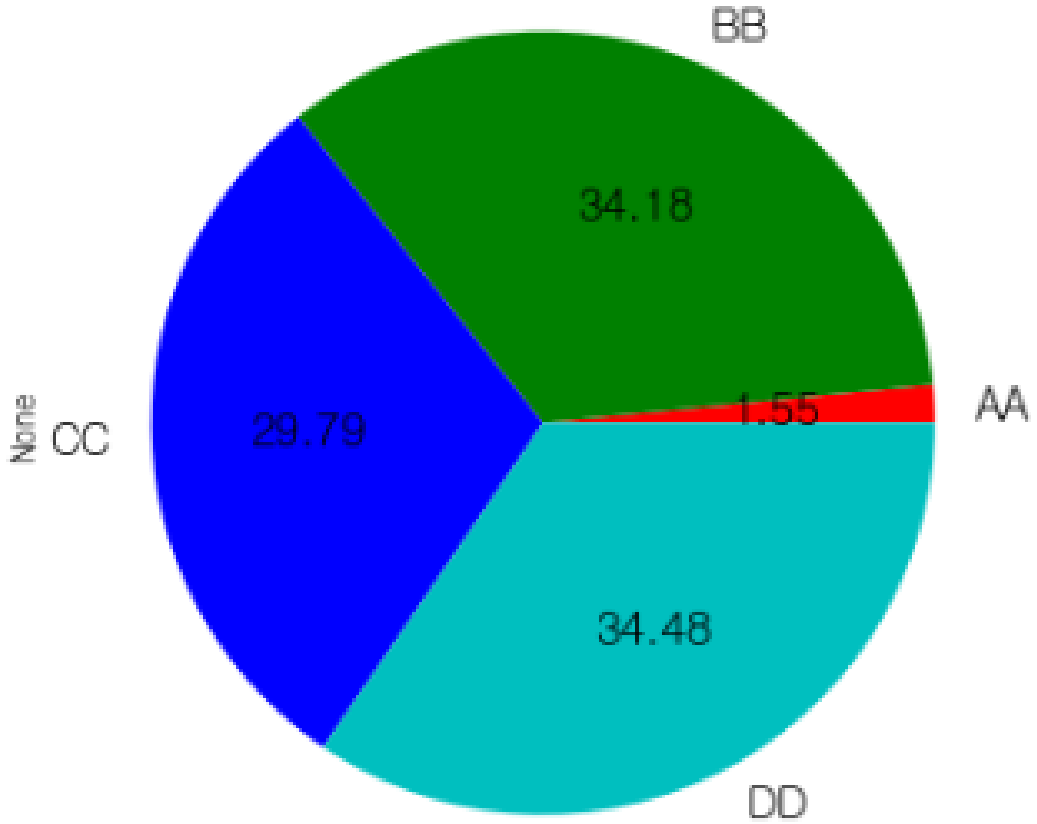
```
ss.plot.pie(figsize = (6, 6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x
```



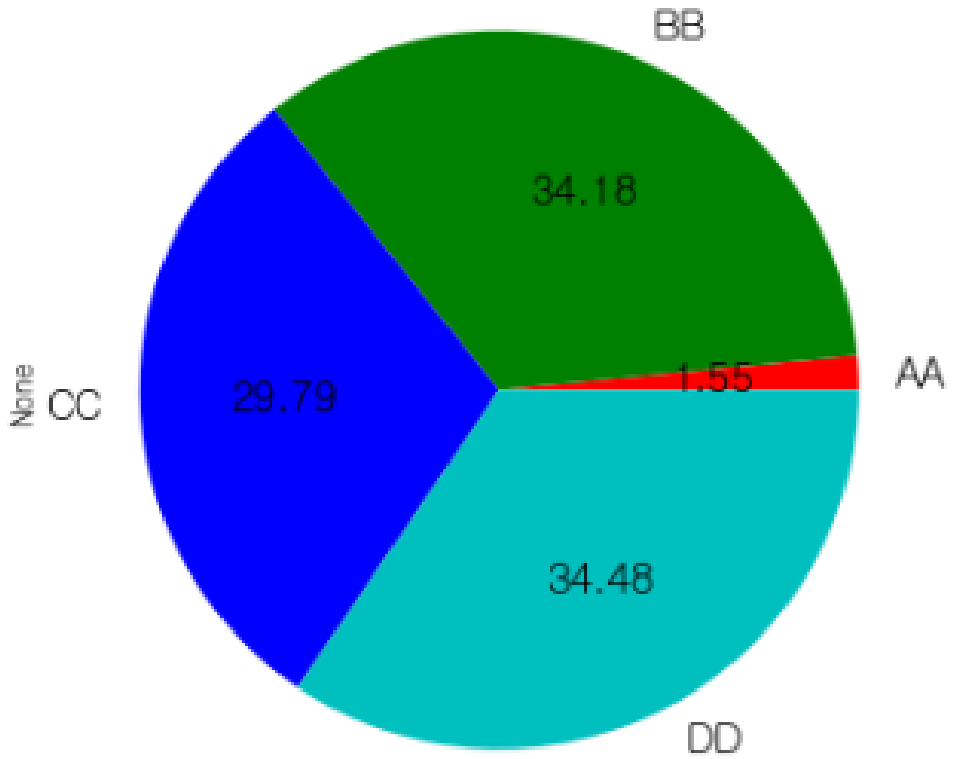

```
ss.plot.pie(labels=['AA', 'BB', 'CC', 'DD'],
            colors=['r', 'g', 'b', 'c'],
            autopct='%.2f', fontsize=15, figsize=(6, 6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x166bd5d7f60>
```



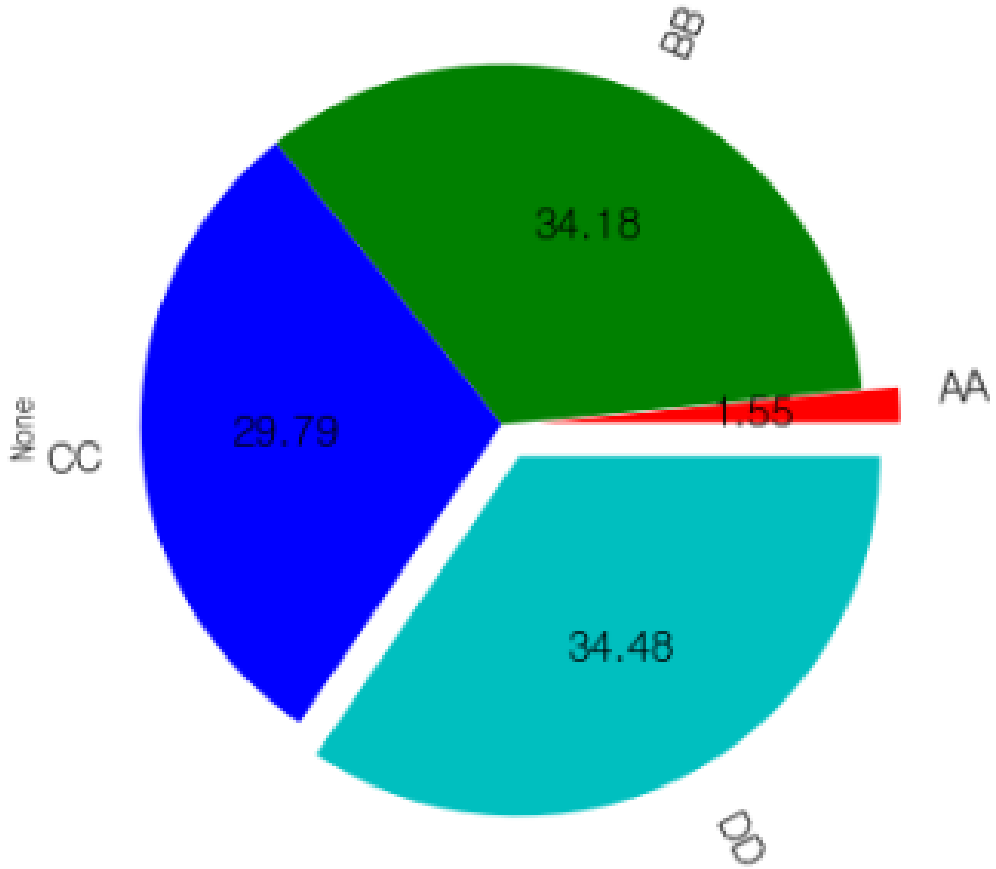
```
ss.plot.pie(labels=['AA', 'BB', 'CC', 'DD'],
            colors=['r', 'g', 'b', 'c'],
            autopct='%.2f', fontsize=15, figsize=(6, 6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x166bd5d7f60>
```




```
ss.plot.pie(labels=['AA', 'BB', 'CC', 'DD'],
            colors=['r', 'g', 'b', 'c'],
            autopct='%.2f', fontsize=15, figsize=(6, 6),
             explode = [0.1, 0, 0, 0.1],
             rotatelabels = 25)
```

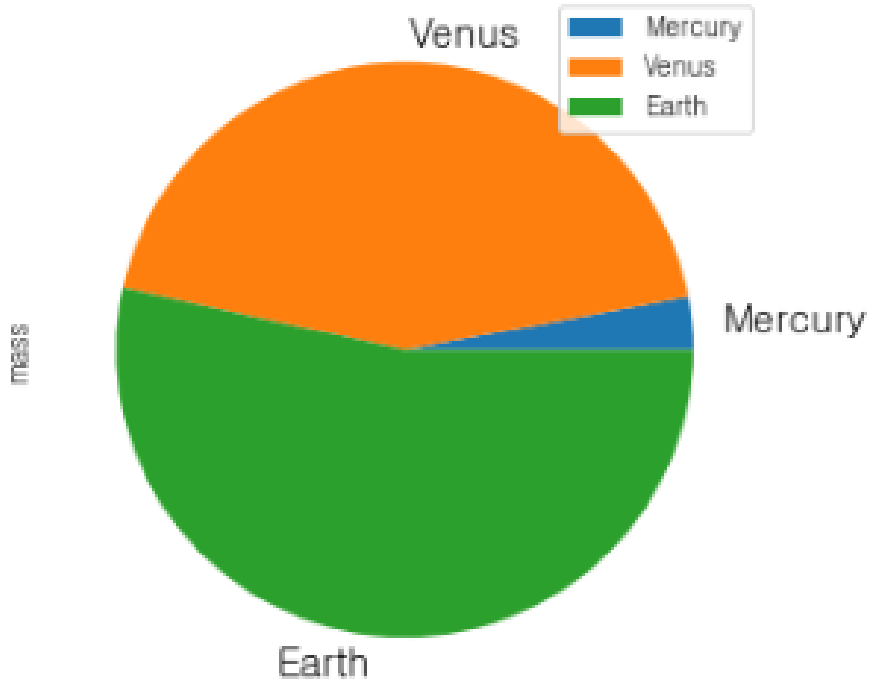
```
<matplotlib.axes._subplots.AxesSubplot at 0x166beb55208>
```



```
df
```

	mass	radius
Mercury	0.33	2439.7
Venus	4.87	6051.8
Earth	5.97	6378.1

```
df.plot.pie(y='mass', figsize=(5, 5), fontsize = 15)
```



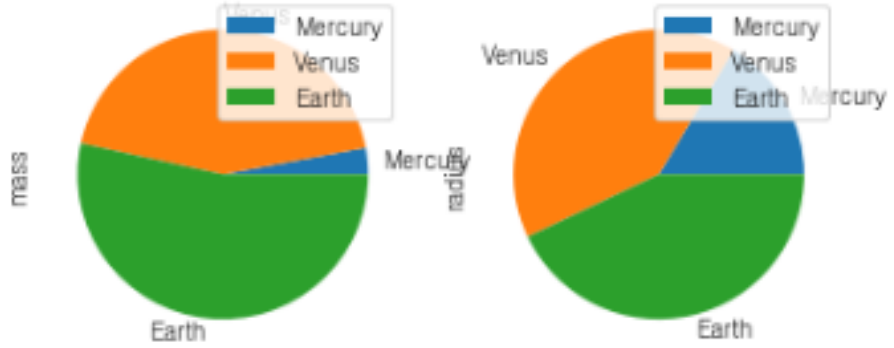
	mass	radius
Mercury	0.33	2439.7
Venus	4.87	6051.8
Earth	5.97	6378.1

```
df.plot.pie(subplots=True, figsize=(6, 3))
```

```
df.plot.pie(subplots=True, figsize=(10, 5))
```

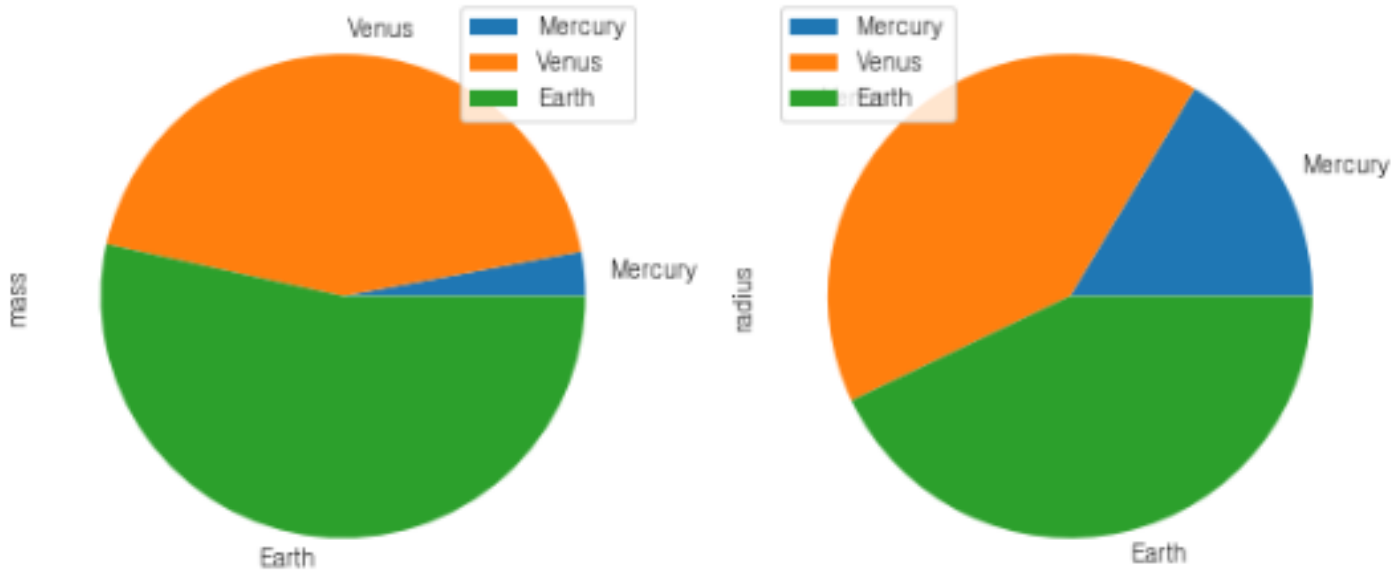

```
df.plot.pie(subplots=True, figsize=(6, 3))
```


```
array([<matplotlib.axes._subplots.AxesSubplot ob
       <matplotlib.axes._subplots.AxesSubplot ob
      dtype=object)
```



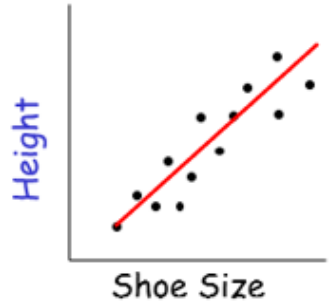

```
df.plot.pie(subplots=True, figsize=(10, 5))
```


```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x00000166BDA609B0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000166BDA239B0>]
      dtype=object)
```

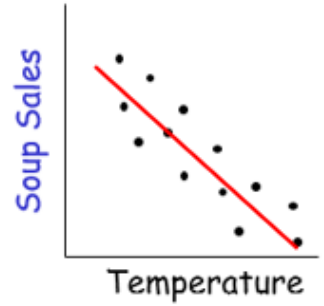


6. 산포도

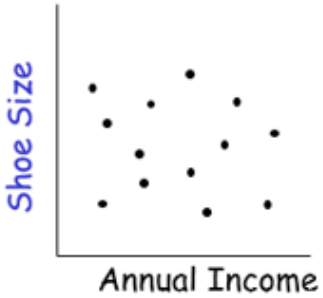
- 산포도(scatter diagram)
  - 두 변수의 상관관계를 판단하기 위해 사용하는 그래프



상관관계: 양(positive)

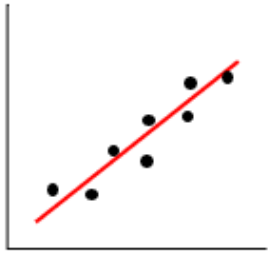


상관관계: 음(negative)



상관관계: 없음

- 최적선(line of best-fit)
  - 상관관계 파악을 위한 직선



강한 양의 상관관계

약한 양의 상관관계

plt.기준

```
scatter(x, y s=None, c='b', marker='o', alpha=None)
```

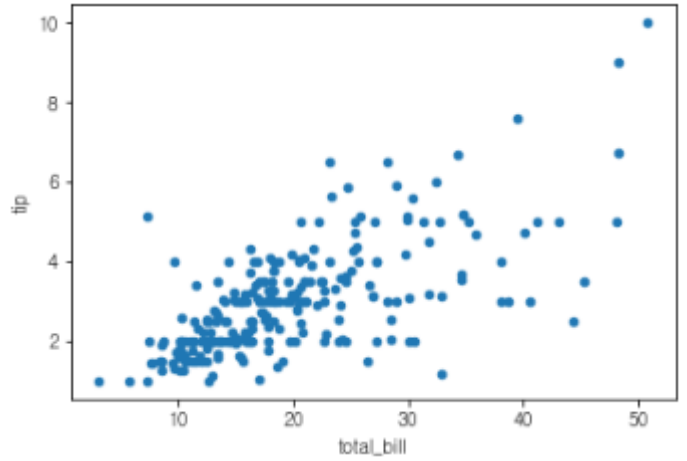
```
df = sns.load_dataset("tips")
df.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
df.plot.scatter( )
```

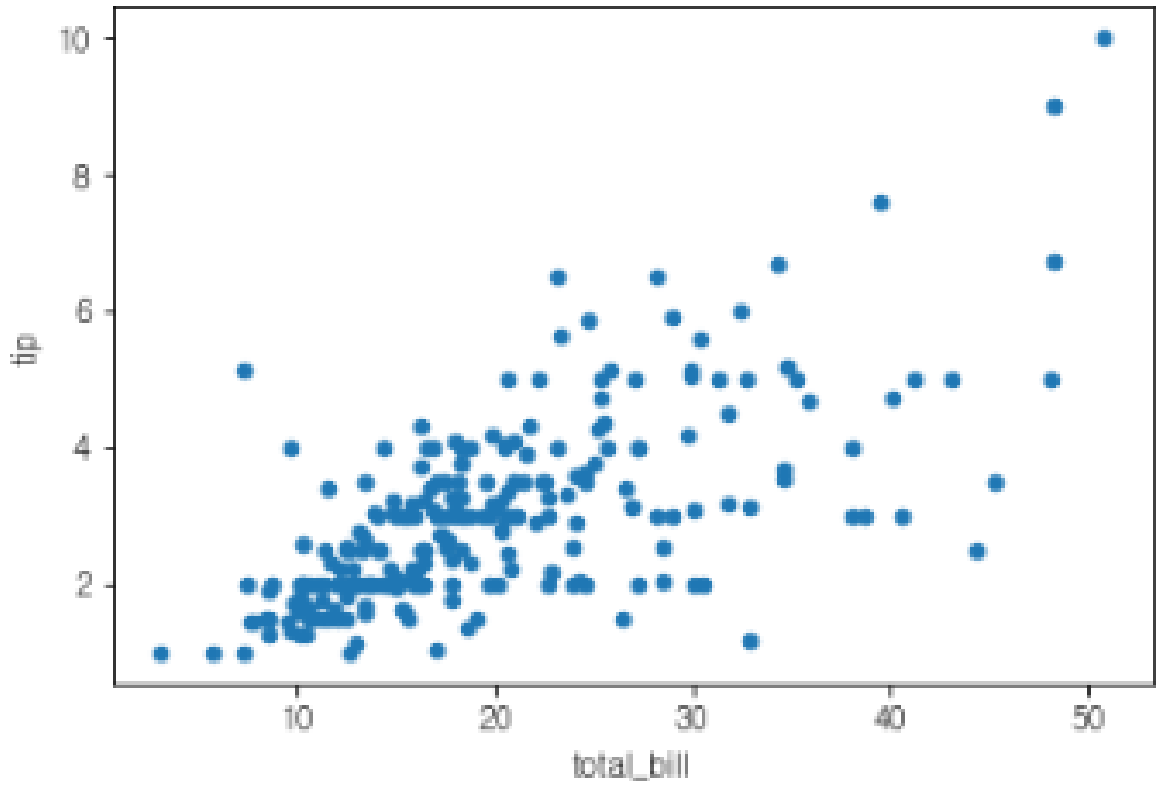
에러남!! 꼭 x, y 지정해야 한다는..

```
df.plot.scatter(x='total_bill', y='tip')
```



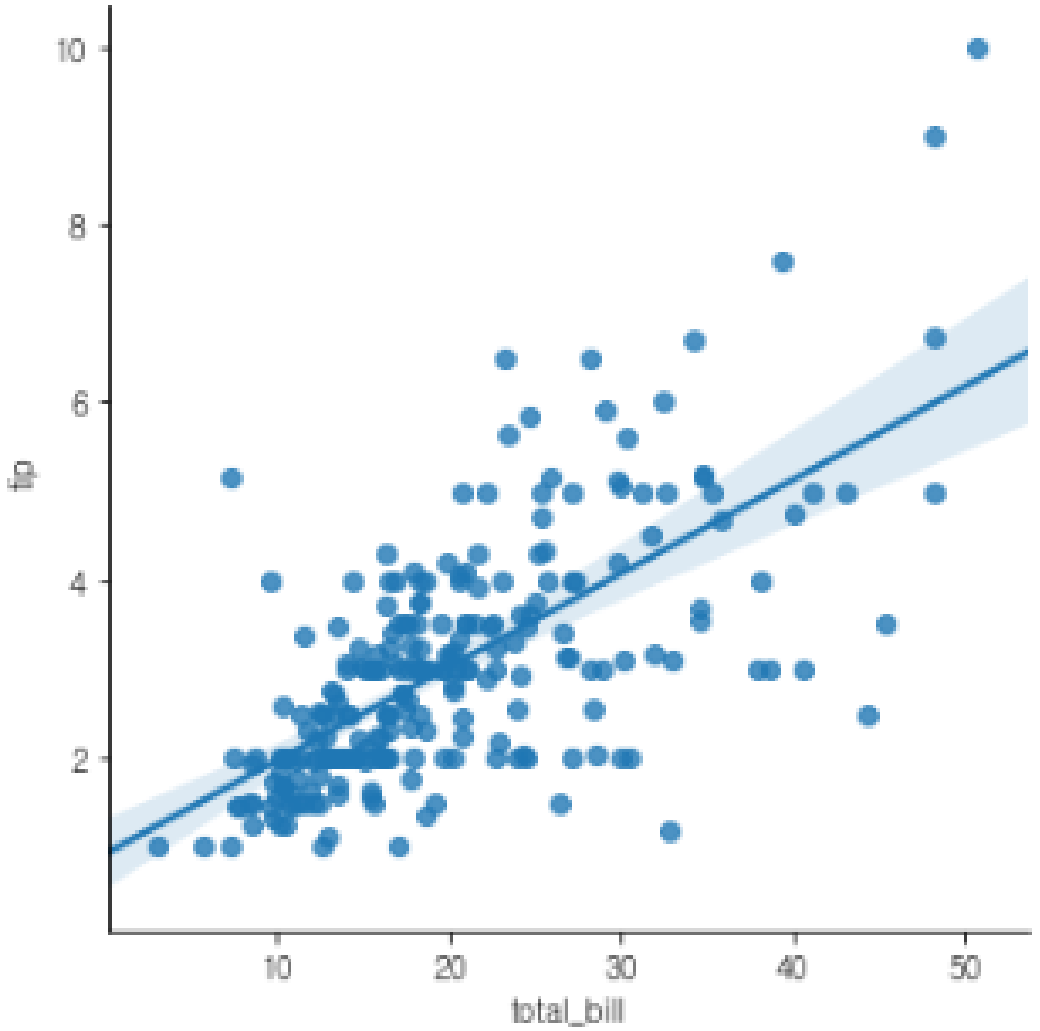
```
df.plot.scatter(x='total_bill', y='tip')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x166be
```




```
sns.lmplot(x="total_bill", y="tip", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x166bf231d30>
```

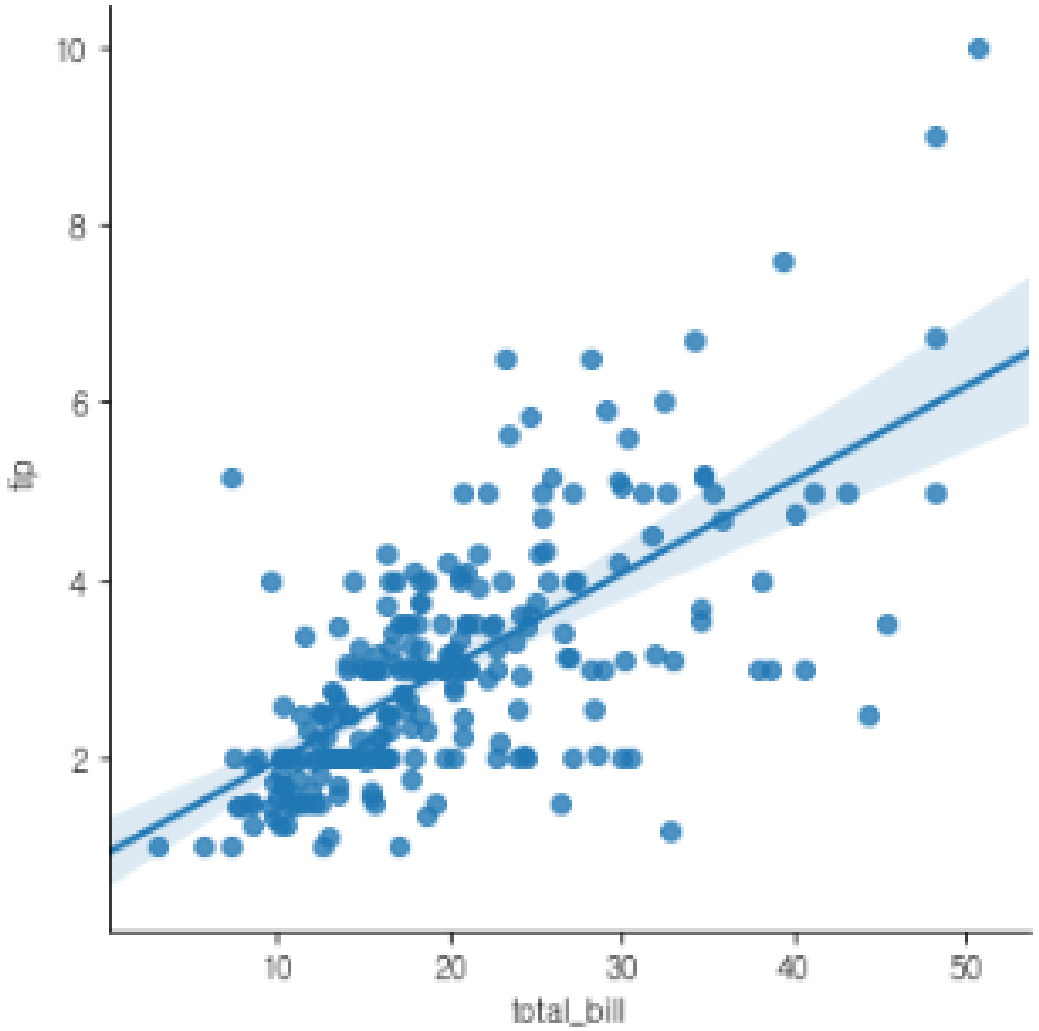


https://seaborn.pydata.org/generated/seaborn.lmplot.html

6. 산포도

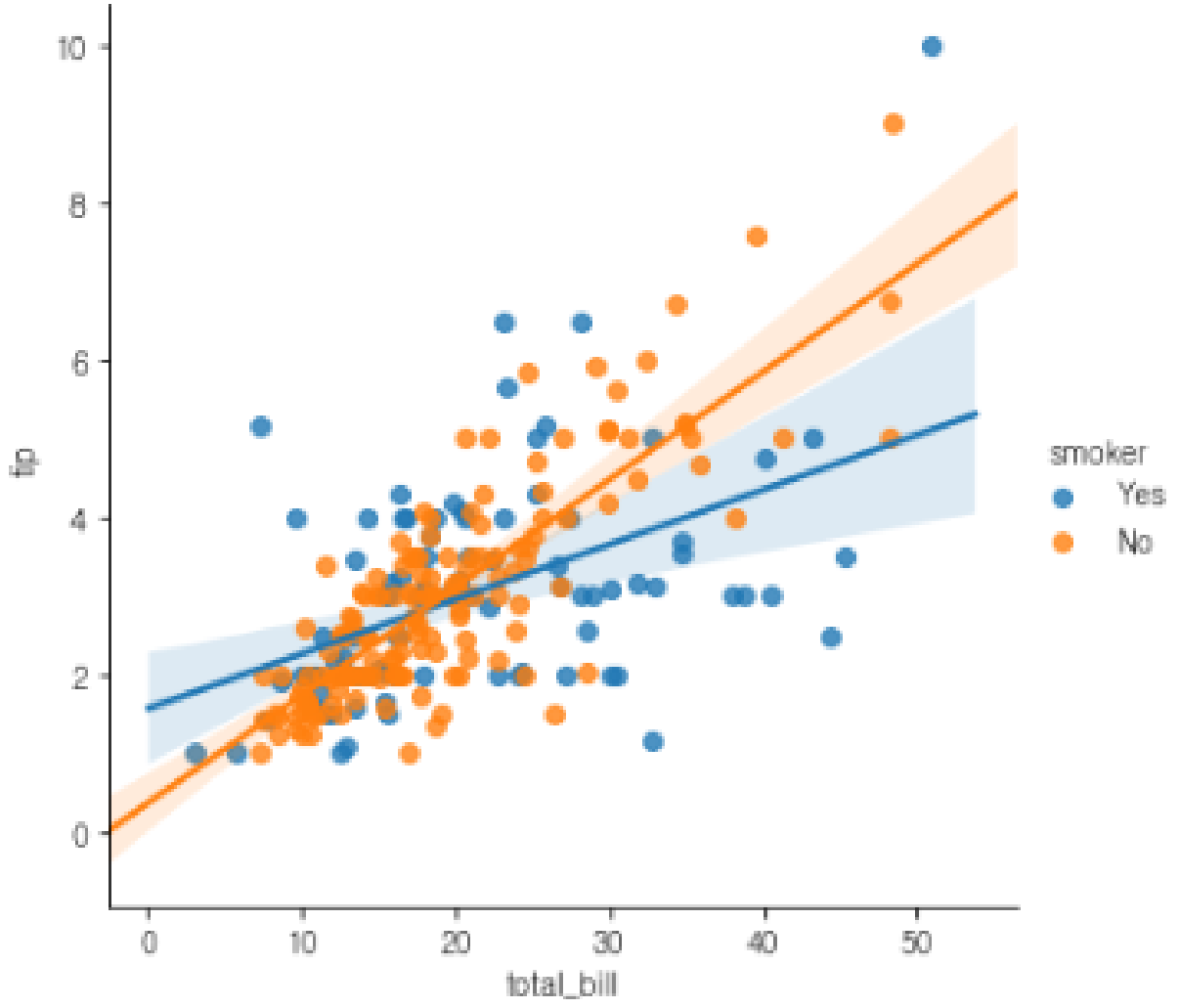
```
sns.lmplot(x="total_bill", y="tip", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x166bf231d30>
```



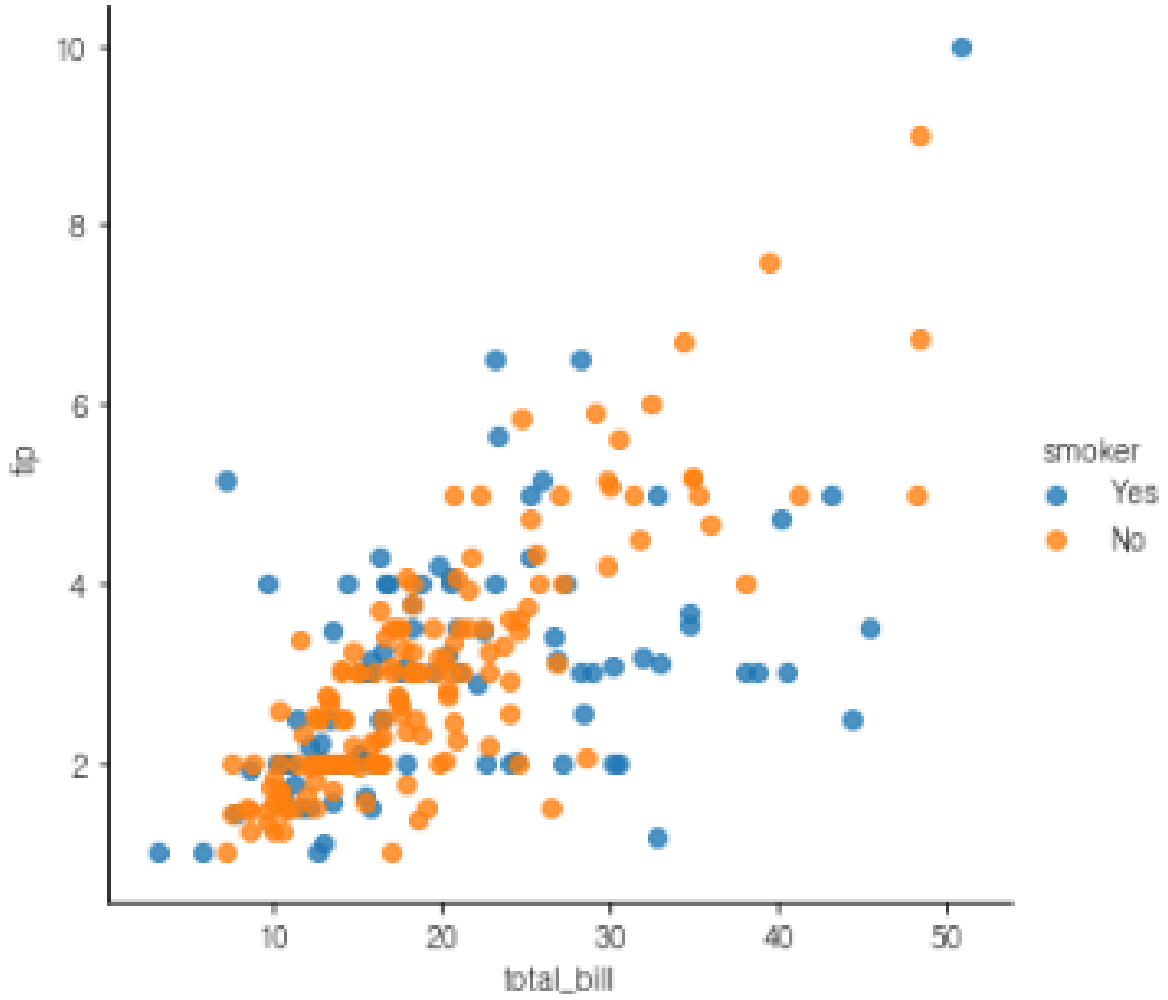

```
sns.lmplot(x="total_bill", y="tip", hue="smoker", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x166c05e0e80>
```



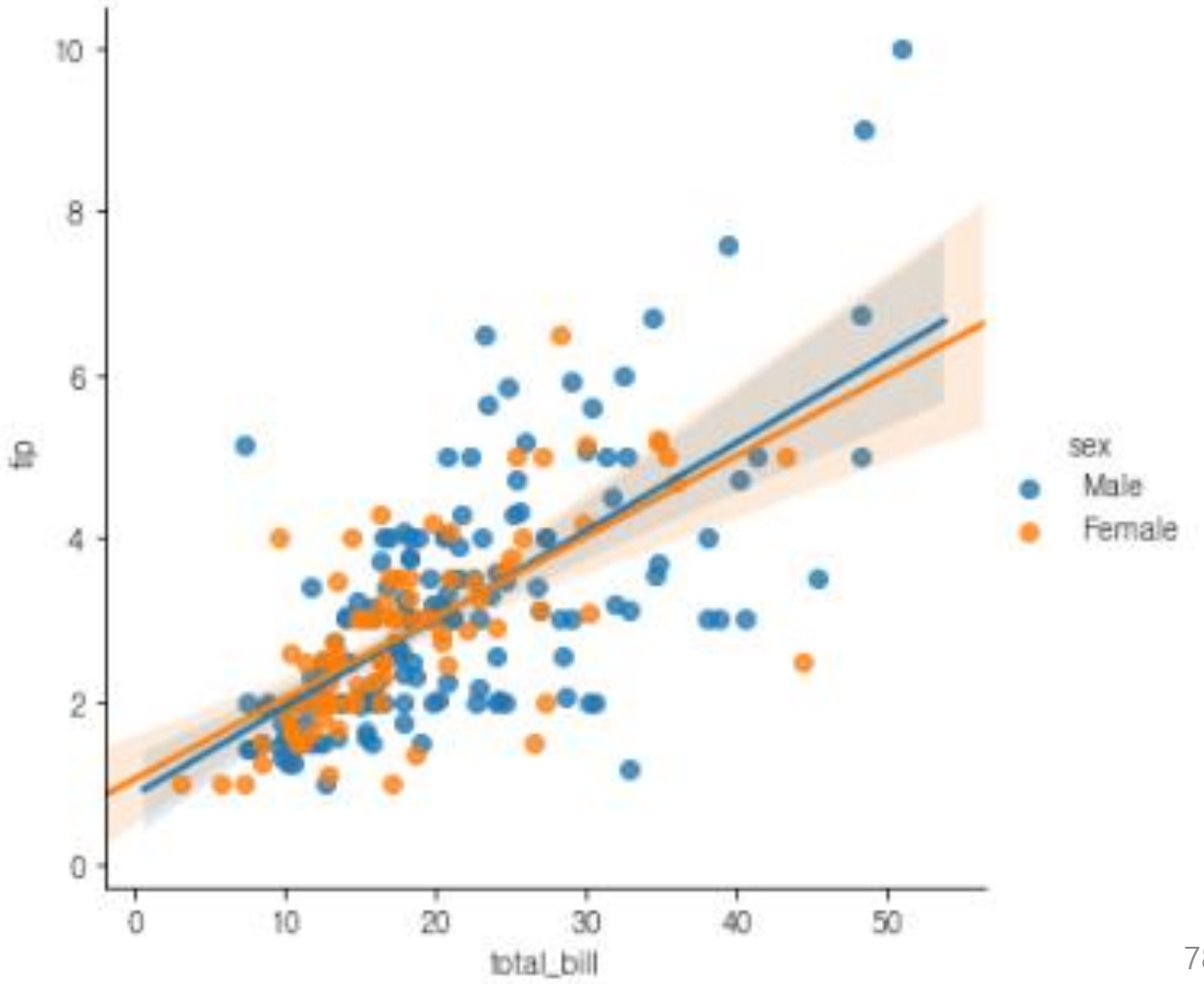
```
sns.lmplot(x="total_bill", y="tip", hue="smoker",
           data=df, fit_reg = False)
```

```
<seaborn.axisgrid.FacetGrid at 0x166c02ed1d0>
```




```
sns.lmplot(x="total_bill", y="tip", hue="sex", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x166c0575f28>
```



7. Hexagonal bin plot

Hexbin plot은 데이터 많을 때 어느 부분의 수가 큰지 작은지 구분해서 볼 수 없는 산점도(scatter plot)의 단점을 보완

scatter plot과 histogram의 보완용
데이터의 크고 작음을 비교할 수 있게 함

기본색 기준으로, 진할수록 다른 것에 비해 큰 값들이 몰려 있음

https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html#hexagonal-bin-plot
https://seaborn.pydata.org/examples/hexbin_marginals.html

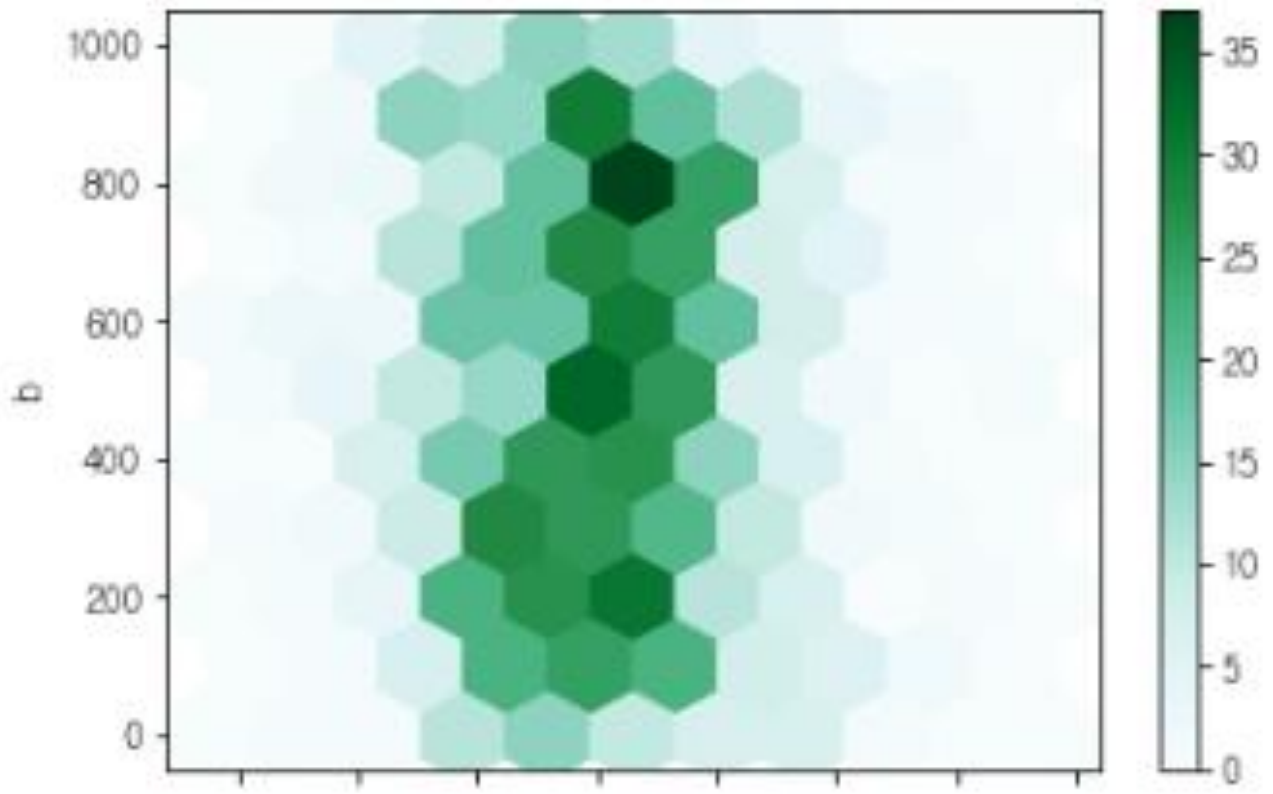
7. Hexbin

```
df.tail( )
```

	a	b
995	-0.239106	993.833254
996	0.339041	995.565155
997	0.767092	995.924927
998	-0.707685	996.715501
999	-0.515246	998.627703

```
df.plot.hexbin(x='a', y='b', gridsize=10)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x166
```



https://seaborn.pydata.org/generated/seaborn.lmplot.html

7. Hexbin

```
df.plot.hexbin(x='a', y='b', gridsize=35)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x166
```