데이터 분석(Python / R)

3일차 : 데이터 시각화

삼성전자공과대학교 3학년 3학기

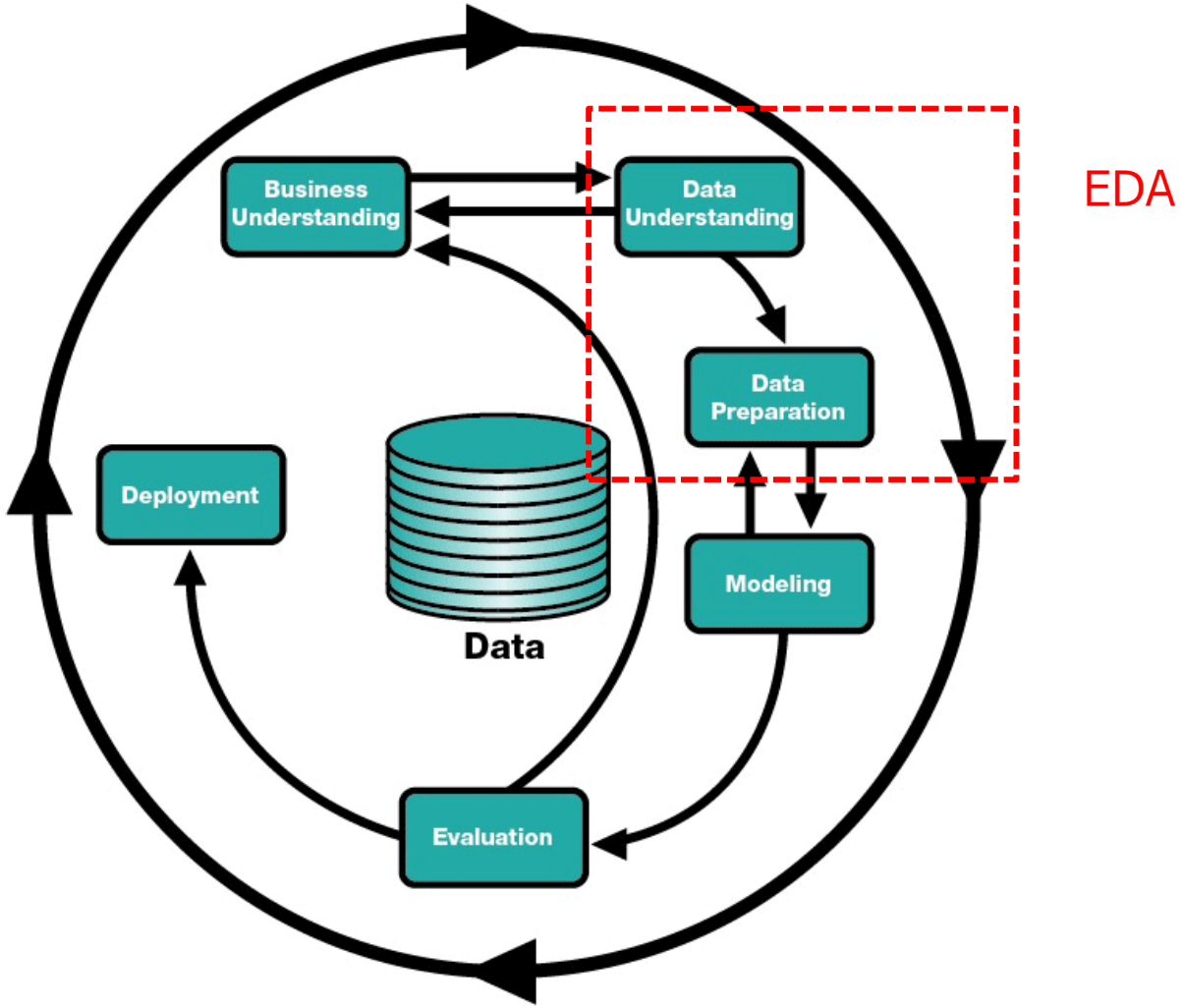
탐색적 데이터 분석(EDA)

탐색적 데이터 분석(EDA)

- John W. Tukey, “탐색적 자료분석(EDA)”, 1977
  - ✓ 자료가 무엇을 말하려 하는가를 보기 위한 것, 단순한 계산과 그리기 쉬운 그림에 집중
  - ✓ 추론통계학이 아닌 기술통계학(descriptive statistics)의 중요성을 강조

목적	설명
Maximize	데이터에 대한 인사이트 최대화
Uncover	데이터의 기본 구조 파악
Extract	중요 변수 추출
Detect	데이터의 이상치 및 특이치 검출
Test	기본 가정 테스트
Develop	정교한 모델 개발
Determine	알고리즘의 최적 인자 선택

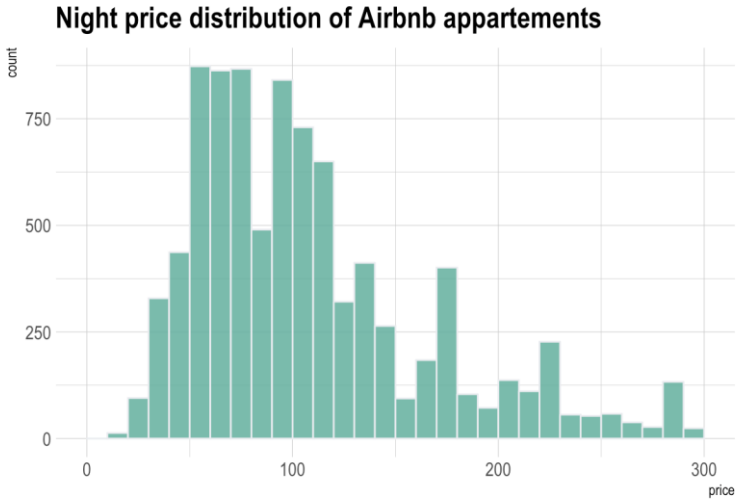
EDA 수행 단계

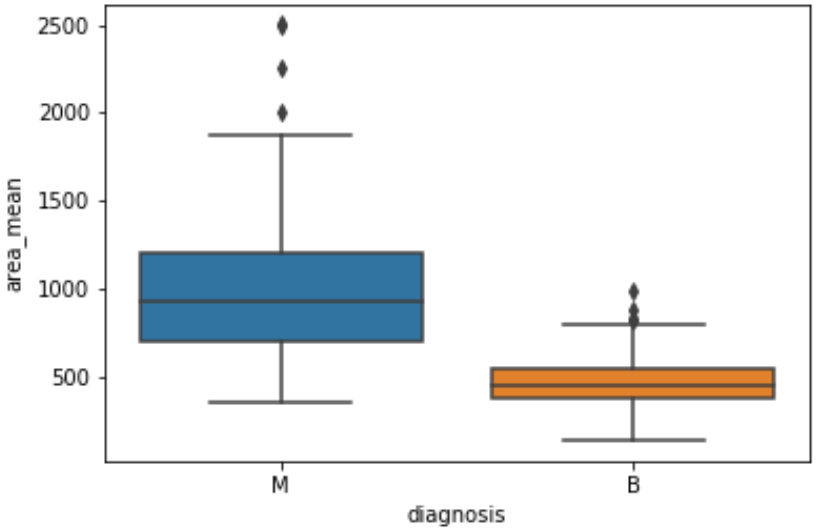

Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data
EDA

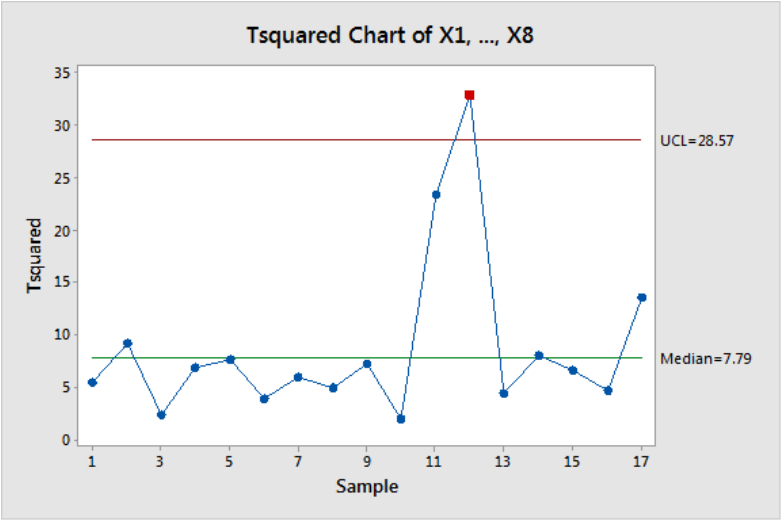
EDA vs CDA

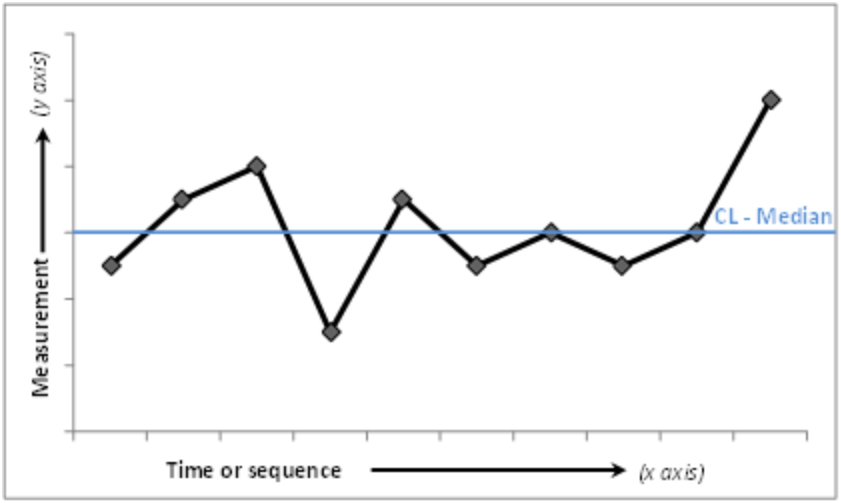
- EDA(Exploratory Data Analysis)
  - ✓ 데이터의 구조와 특징을 파악하며 여기서 얻은 정보를 바탕으로 분석 모델을 만드는 단계
- CDA(Confirmatory Data Analysis)
  - ✓ 관측된 형태나 효과의 재현성 평가, 유의성 검정, 신뢰구간 추정 등 통계적 추론을 하는 단계
  - ✓ 관련된 다른 자료분석에서 얻어진 정보를 적절히 배려하는 일
  - ✓ 새로 수집된 자료가 앞서의 분석결과에 의한 예측과 얼마나 일치하는가를 평가하는 일

도구 – 시각화


Night price distribution of Airbnb appartements
count
750
500
250
0
0
100
200
300
price


2500
2000
1500
1000
500
area_mean
M
B
diagnosis


Tsquared Chart of X1, ..., X8
35
30
25
20
15
10
5
0
Tsquared
UCL=28.57
Median=7.79
1
3
5
7
9
11
13
15
17
Sample


(y axis)
Measurement
CL - Median
Time or sequence
(x axis)

도구 – Pandas-profiling

- **Essentials**: type, unique values, missing values
- **Quantile statistics** like minimum value, Q1, median, Q3, maximum, range, interquartile range
- **Descriptive statistics** like mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, skewness
- **Most frequent values**
- **Histogram**
- **Correlations** highlighting of highly correlated variables, Spearman, Pearson and Kendall matrices
- **Missing values** matrix, count, heatmap and dendrogram of missing values

도구 – Pandas-profiling

Variables

created_at
Date

Distinct count	84	**Minimum**	2018-10-10 12:59:59
Unique (%)	0.0%	**Maximum**	2018-10-10 13:06:43
Missing (%)	0.0%		
Missing (n)	0		
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

favorite_count
Numeric

Distinct count	7	**Mean**	0.98
Unique (%)	0.0%	**Minimum**	0
Missing (%)	0.0%	**Maximum**	44
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

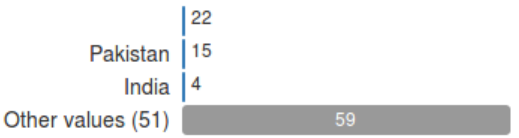
~~geo~~
Constant

This variable is constant and should be ignored for analysis

Constant value

location
Categorical

Distinct count	54		
Unique (%)	0.0%		
Missing (%)	0.0%		
Missing (n)	0		



Toggle details

데이터 시각화

- 데이터에 대한 이해, 처리, 가치 추출, 시각화, 설명 등의 능력은 향후 수십년간 아주 중요한 기술이 될 것이다. 왜냐하면, 현재의 우리는 언제 어디서나 자유롭게 데이터를 얻을 수 있기 때문이다.

  [Hal Varian*, Google's Chief Economist, 2008]

- 컴퓨터 기반의 대화형 데이터 시각화는 데이터의 인지하는데 크게 도움이 된다. [Card et al., 1999]

- 데이터의 표현하고 드러내는 시각화는 데이터를 이해하는데 도움이 된다. [Kirk, 2016]

시각화의 효율성(1/2)

X	Y
0.9	0.5
2.7	1.1
6.7	28.6
10.9	32.8
6.0	15.7
6.3	19.7
7.0	32.6
8.7	32.5
4.8	7.3
12.5	33.1
13.4	32.9
2.0	0.75
3.6	3.6
14.6	33

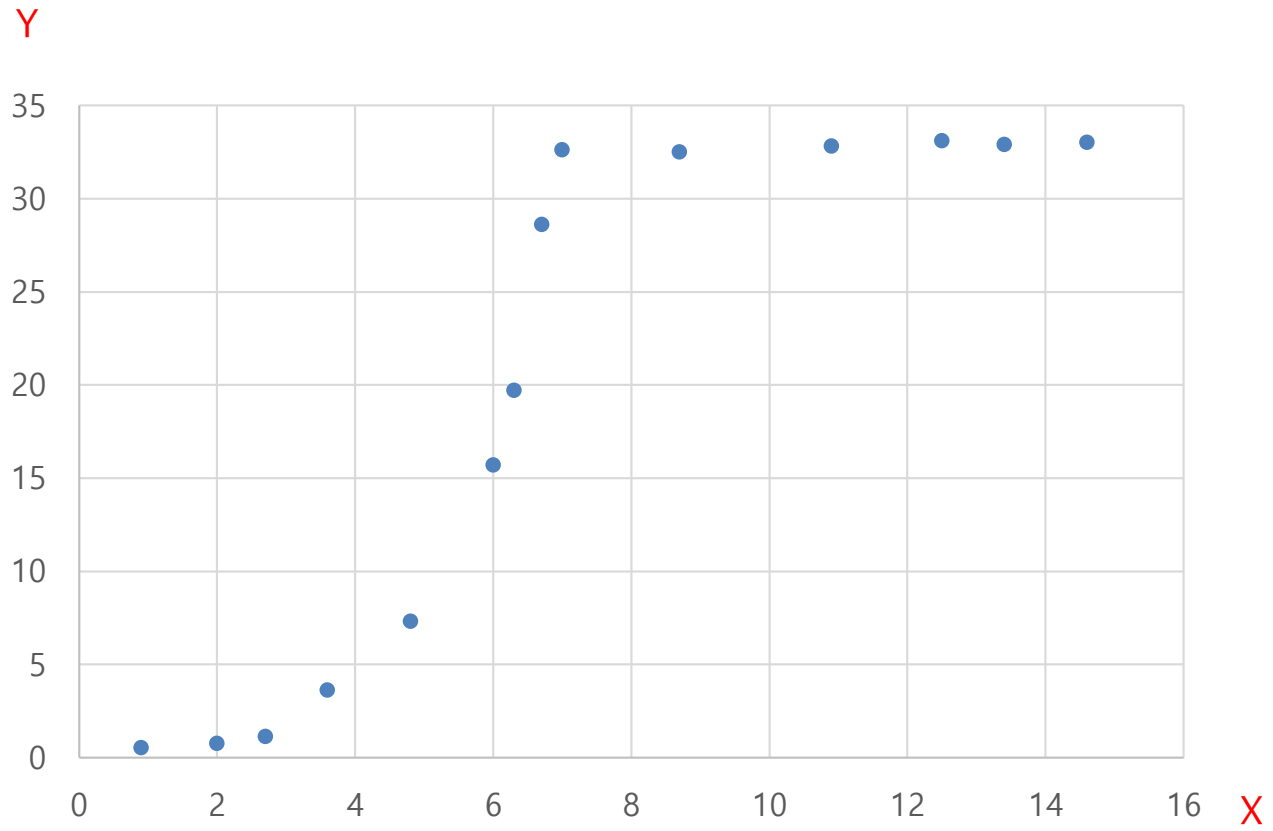
- 기술 통계

	X	Y
중앙값	6.5	23.8
평균	7.2	19.5
표준편차	4.2	13.6

✓ Correlation = 0.88

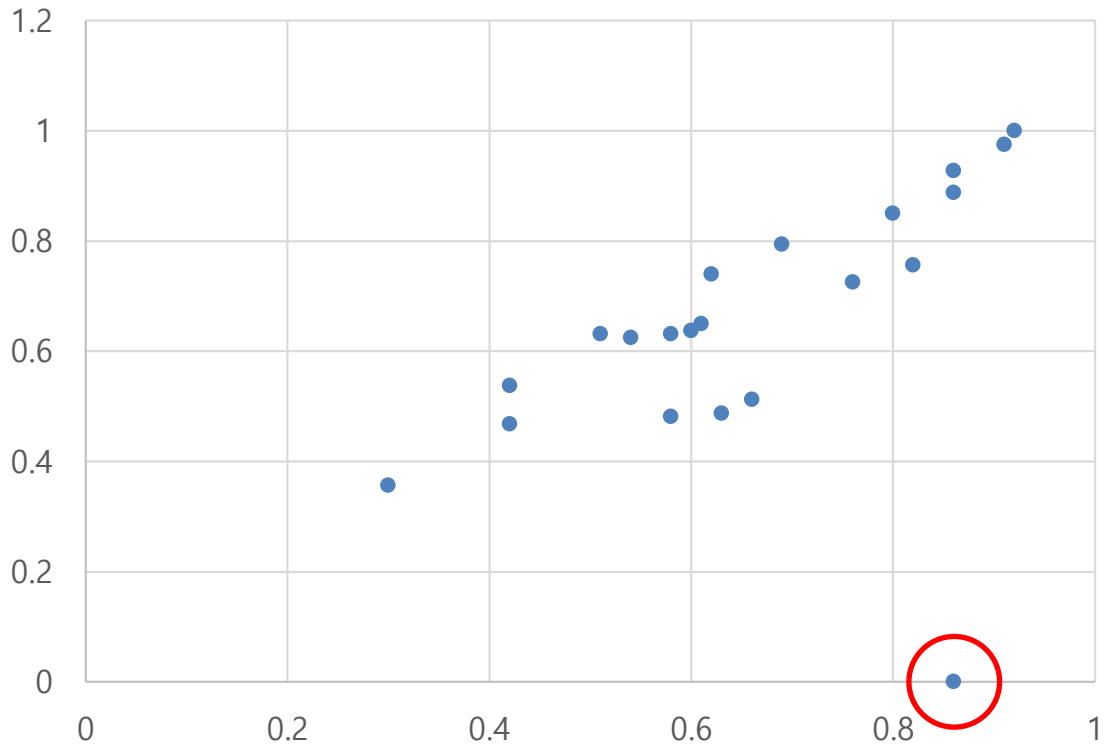
시각화의 효율성 (1/2)

X	Y
0.9	0.5
2.7	1.1
6.7	28.6
10.9	32.8
6.0	15.7
6.3	19.7
7.0	32.6
8.7	32.5
4.8	7.3
12.5	33.1
13.4	32.9
2.0	0.75
3.6	3.6
14.6	33



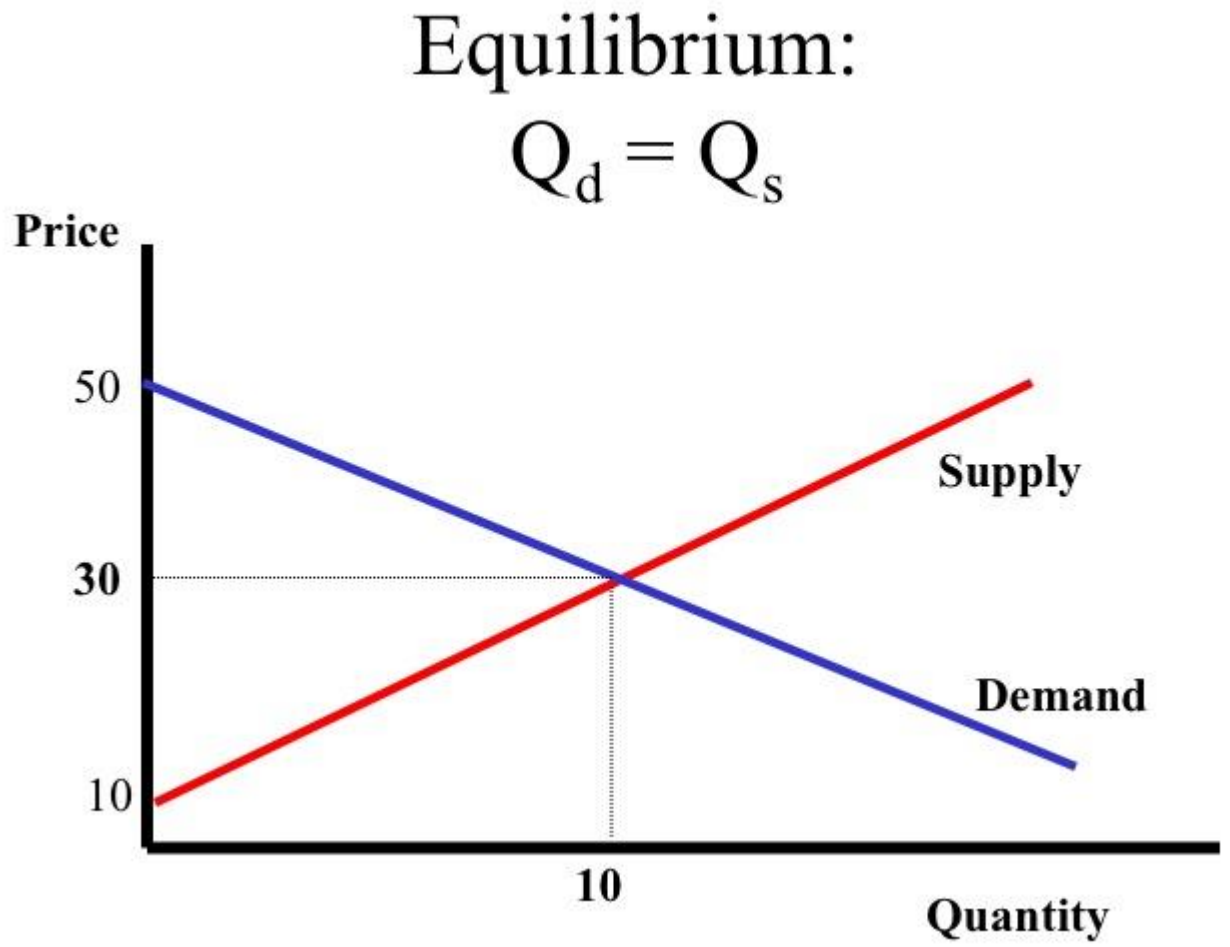
시각화의 효율성(2/2)

X	Y
0.42	0.46750
0.54	0.62500
0.42	0.53750
0.86	0.92750
0.6	0.63750
0.51	0.63125
0.3	0.35625
0.61	0.65000
0.58	0.63125
0.76	0.72500
0.58	0.48125
0.66	0.51250
0.63	0.48750
0.92	1.00000
0.86	0.88750
0.91	0.97500
0.82	0.75625
0.86	0.00000
0.8	0.85000
0.69	0.79375
0.62	0.74000

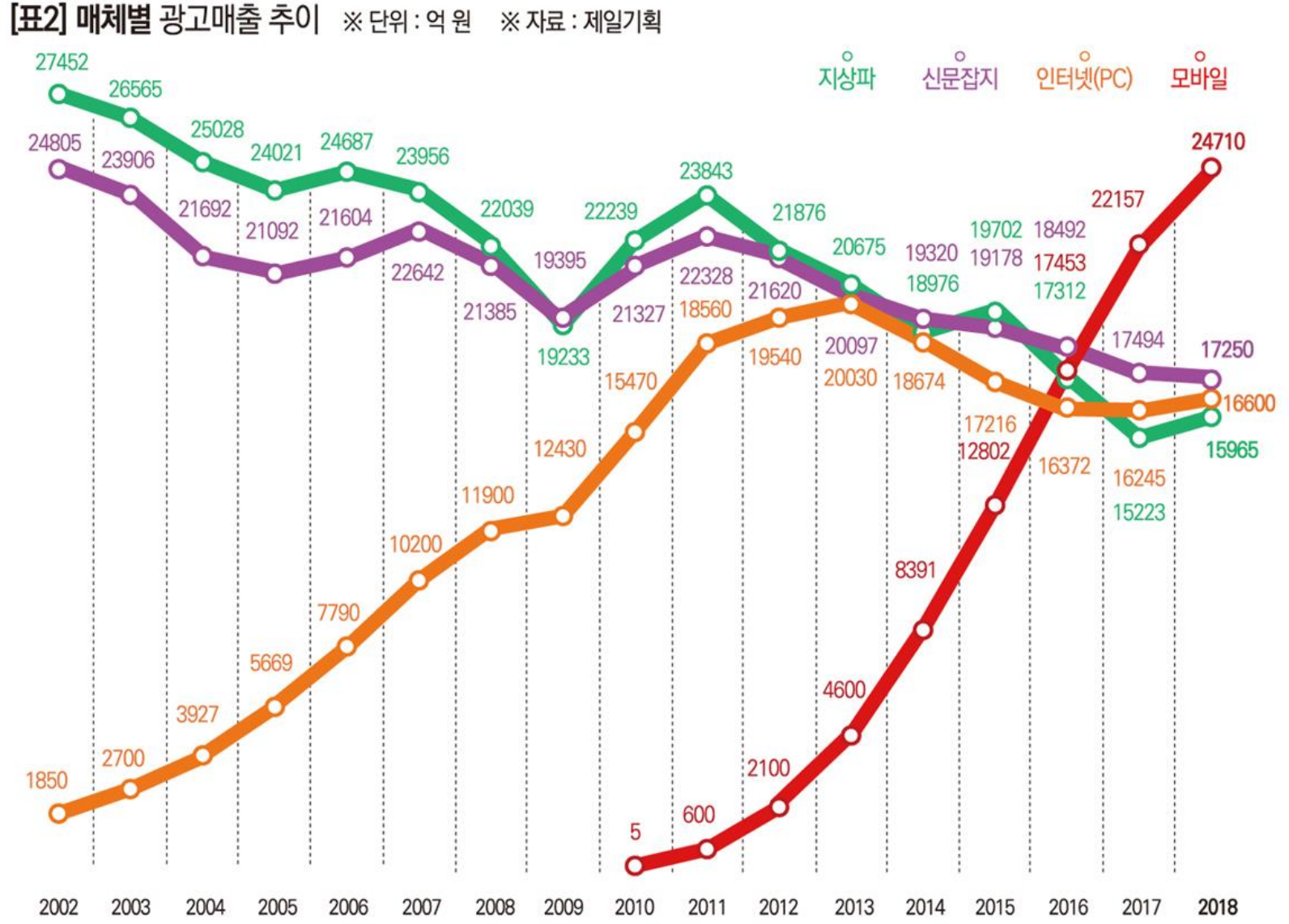


시각화의 관점

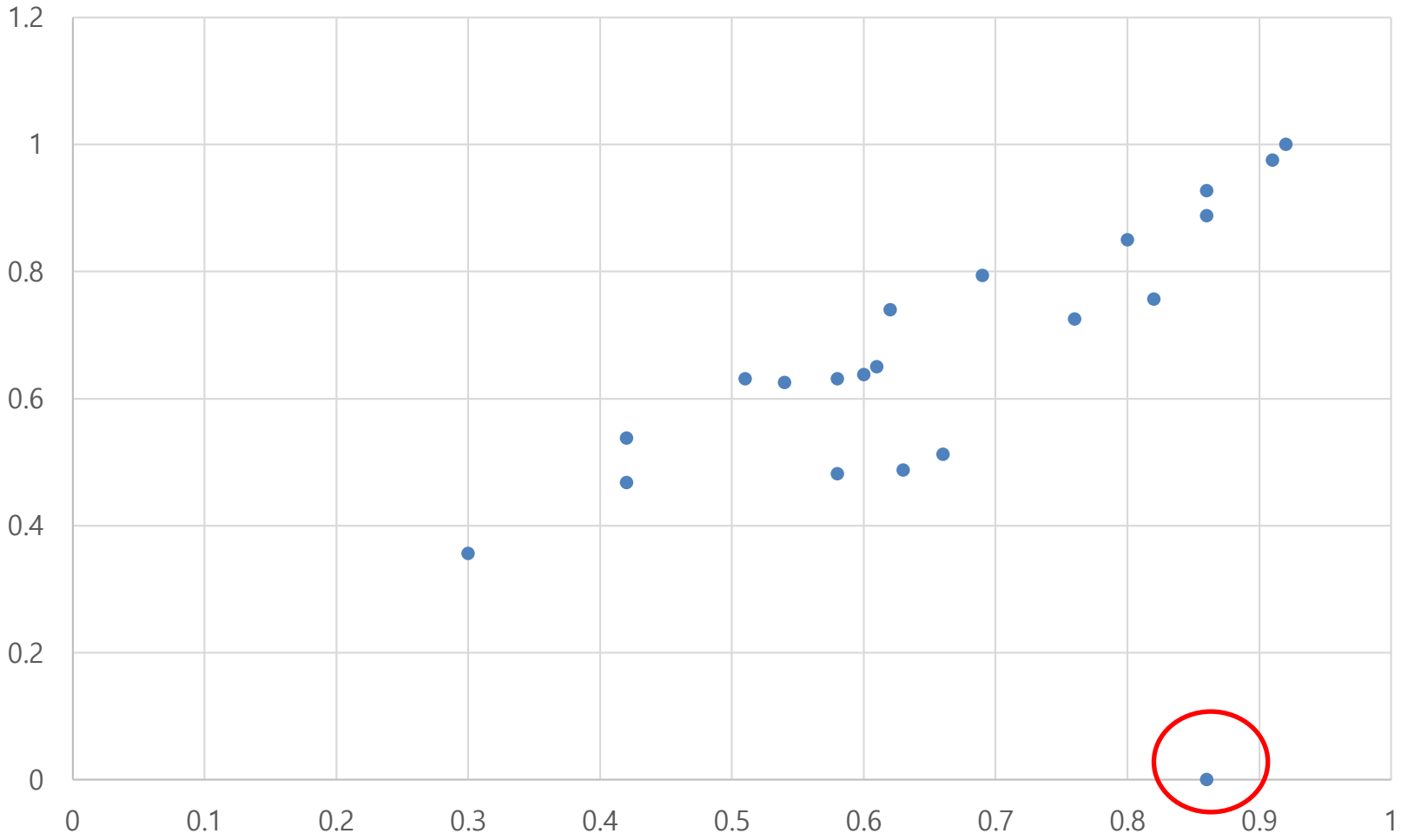
Conceptual : Declarative 시각화



Data-Driven : Declarative 시각화



Data-Driven : Explorative 시각화



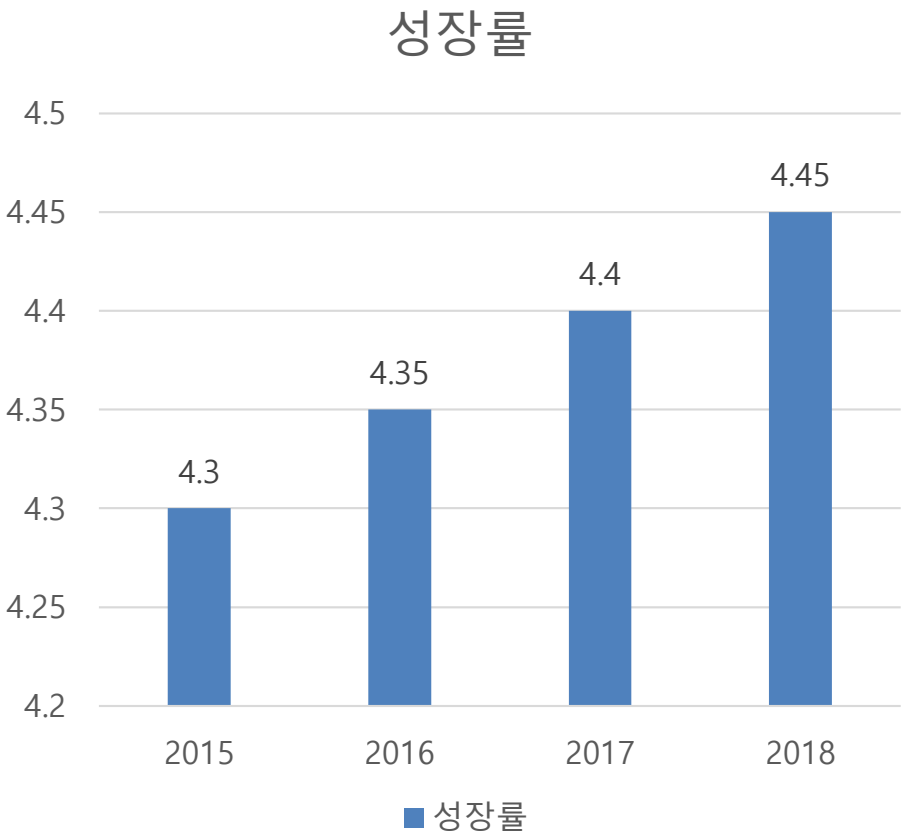
좋은 데이터 시각화

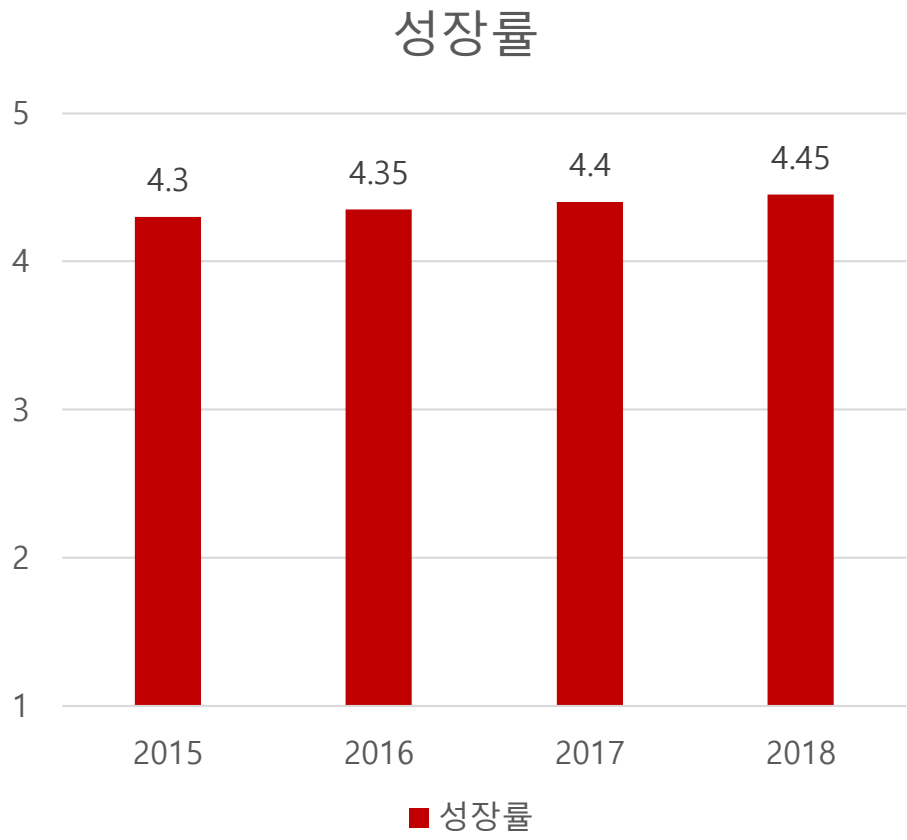
- Trustworthy
- Accessible
- Elegant

Kirk, A. Data Visualisa8on: A handbook for Data Driven Design. SAGE publica8ons, 2016.

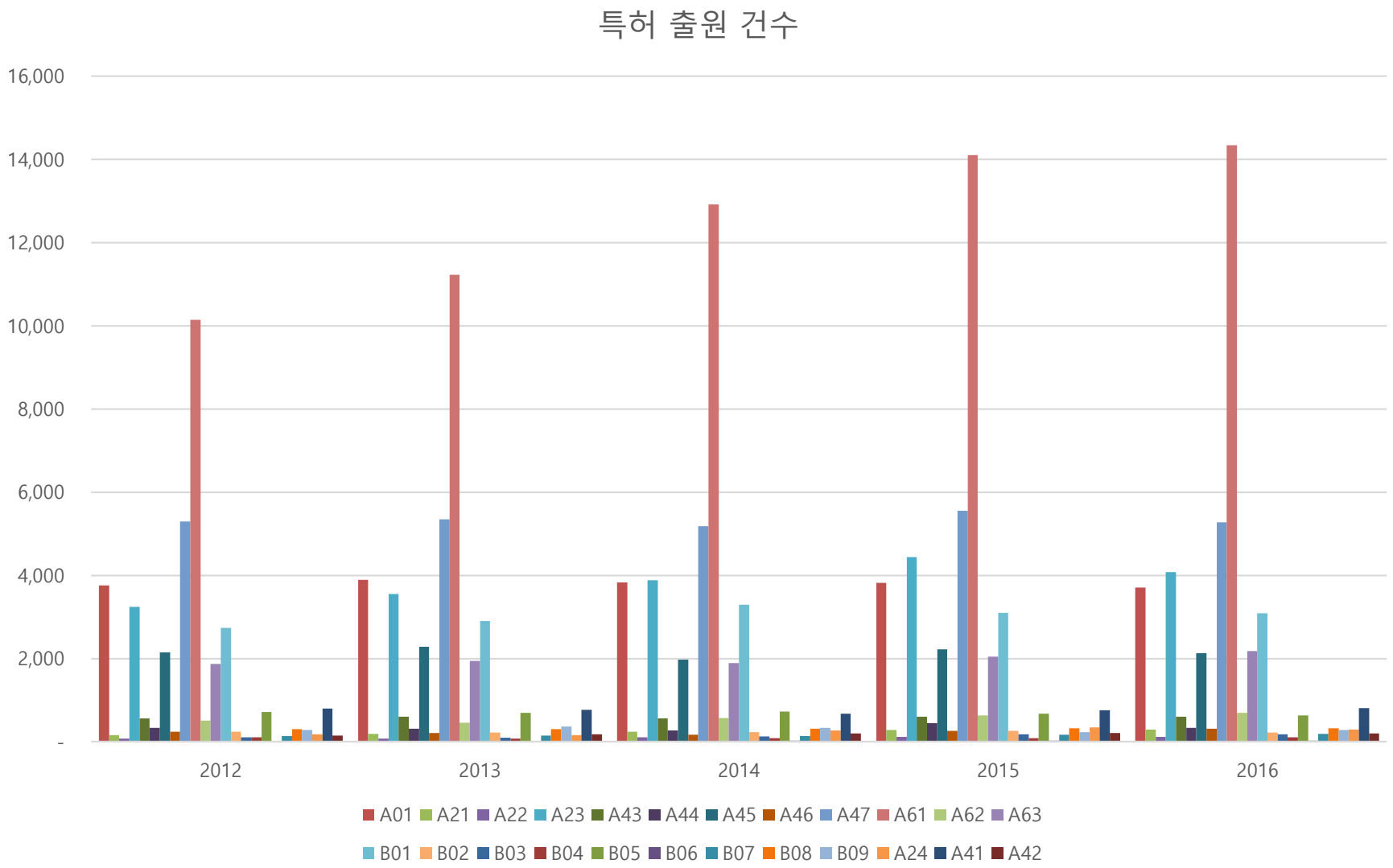
Trustworthy

Trustworthy

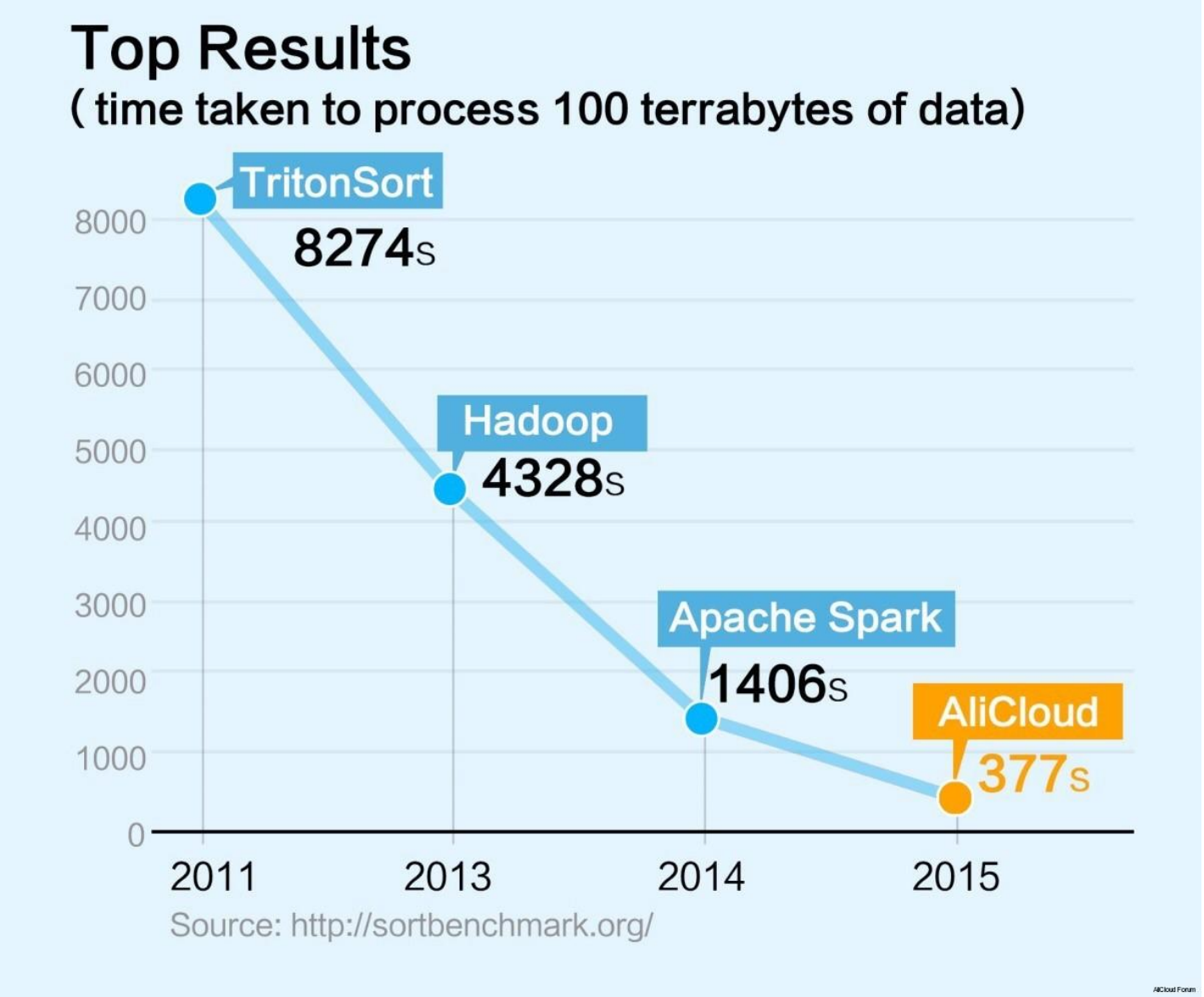




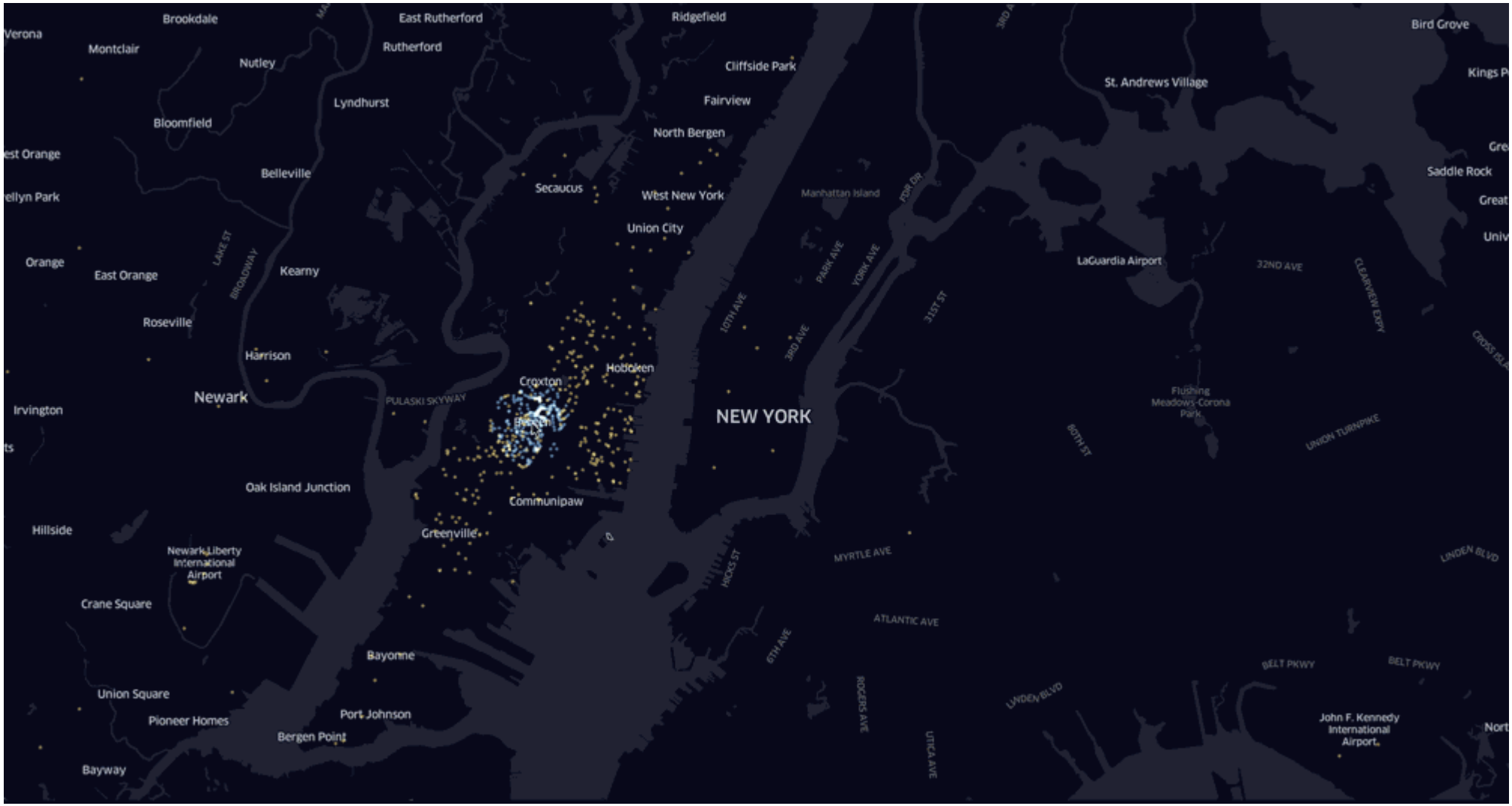
Accessible


특허 출원 건수
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
-
2012
2013
2014
2015
2016
A01 A21 A22 A23 A43 A44 A45 A46 A47 A61 A62 A63
B01 B02 B03 B04 B05 B06 B07 B08 B09 A24 A41 A42

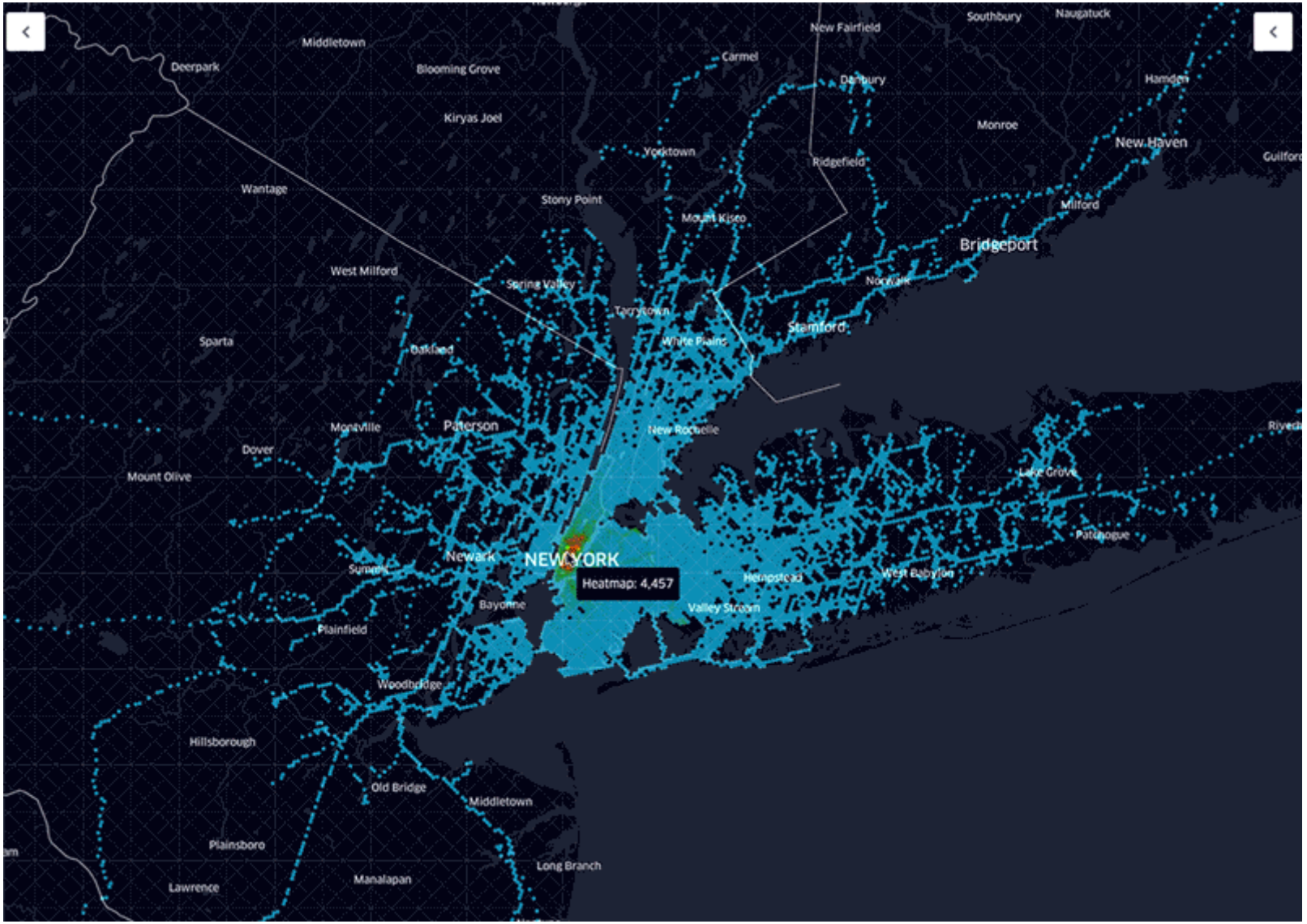
Accessible


Top Results
(time taken to process 100 terrabytes of data)
TritonSort
8274s
Hadoop
4328s
Apache Spark
1406s
AliCloud
377s
8000
7000
6000
5000
4000
3000
2000
1000
0
2011
2013
2014
2015
Source: http://sortbenchmark.org/
AliCloud Forum

Elegant


Brookdale
East Rutherford
Ridgefield
Verona
Montclair
Rutherford
Nutley
Cliffside Park
Bird Grove
St. Andrews Village
Lyndhurst
Fairview
Bloomfield
North Bergen
Belleville
Secaucus
West New York
Manhattan Island
Saddle Rock
Union City
Orange
East Orange
Kearny
LaGuardia Airport
32ND AVE
Roseville
Harrison
Hoboken
Newark
Croxton
PULASKI SKYWAY
NEW YORK
Irvington
Flushing Meadows-Corona Park
Oak Island Junction
Communipaw
Hillside
Greenville
Newark Liberty International Airport
MYRTLE AVE
Crane Square
ATLANTIC AVE
Bayonne
Union Square
Pioneer Homes
Port Johnson
Bergen Point
BELT PKWY
John F. Kennedy International Airport
Bayway
LINDEN BLVD
LAKE ST
BROADWAY
PARK AVE
YORK AVE
FDR DR
10TH AVE
3RD AVE
31ST ST
CLEARVIEW EXPY
UNION TURNPIKE
80TH ST
HICKS ST
6TH AVE
ROGERS AVE
UTICA AVE

Elegant


Middletown
Deerpark
Blooming Grove
Kiryas Joel
Wantage
Stony Point
West Milford
Sparta
Oakland
Monsville
Paterson
Dover
Mount Olive
Newark
NEW YORK
Heatmap: 4,457
Bayonne
Plainfield
Woodbridge
Hillsborough
Old Bridge
Middletown
Plainsboro
Lawrence
Manalapan
Long Branch
Carmel
Yorktown
Ridgefield
New Fairfield
Danbury
Southbury
Naugatuck
Hamden
Monroe
New Haven
Guilford
Milford
Bridgeport
Norwalk
Stamford
White Plains
Tarrytown
Spring Valley
Mount Kisco
New Rochelle
Lake Grove
Patchogue
Hempstead
West Babylon
Valley Stream

Matplotlib

- matplotlib, pyplot, pylab
- figure 구성도
- figure, axes 개념
- plot 함수
- subplot
- subplots
- 점 그래프(scatter)
- 막대 그래프(bar)
- 원 그래프(pie)
- 히스토그램(hist)
- 박스 그래프(boxplot)

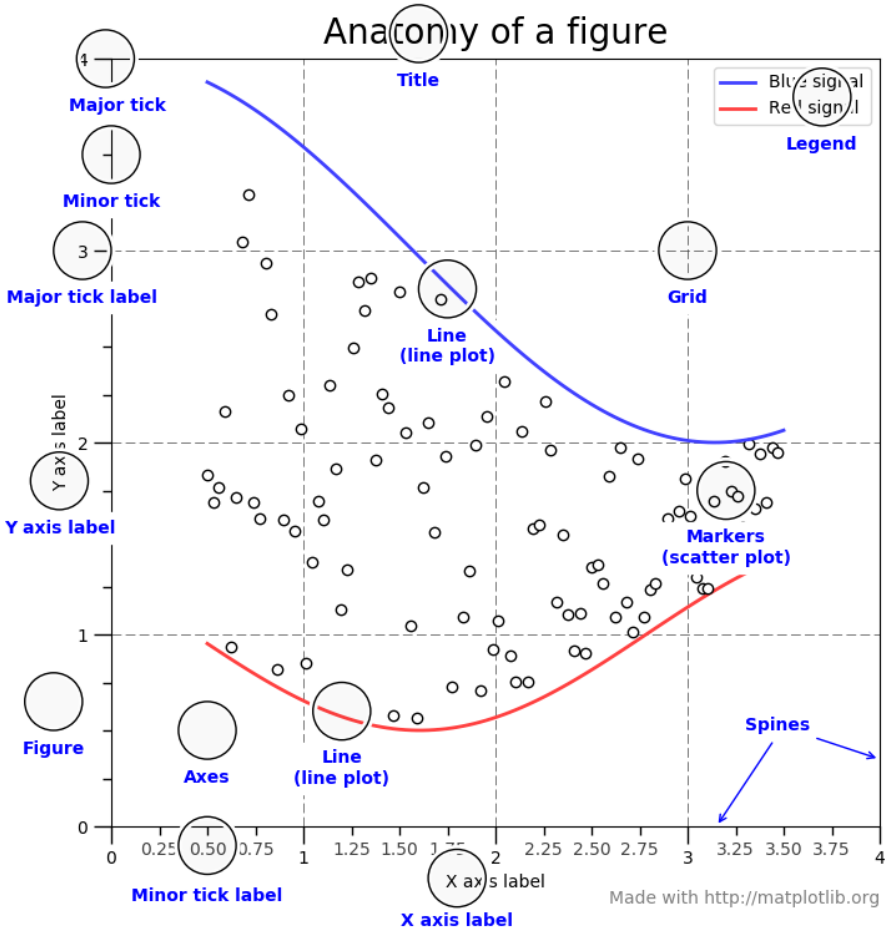
matplotlib / pyplot / pylab

- Matplotlib 란
  - python 에서 차트나 플롯으로 시각화 하는 패키지

- pyplot
  - matplotlib 패키지의 모듈
  - matplotlib의 기본 모듈
  - 일반적으로 from matplotlib import pyplot as plt 로 임포트 사용 권장

- pylab
  - matplotlib 패키지의 모듈
  - pyplot과 numpy의 기능을 합쳐 matlab과 유사하게 사용할 수 있도록 만든 모듈

Figure 구성



- figure
  - Plot이 그려지는 공간
  - 하나 이상의 Axes를 갖음
- axes
  - 데이터가 있는 이미지 영역
  - 하나의 Figure에만 존재(공유불가)
  - 2D의 경우 두개의 Axis로 구성
- axis
  - 그래프 리미트 및 tick, tick label 설정
- artist
  - Figure에 표시되는 모든 구성원

Figure

❑ figure만 생성 후 pyplot 실행시키게 되면 빈 Figure 실행 됨

- Axes가 0개인 사이즈가 432 * 288인 Figure 생성

```
import matplotlib.pyplot as plt
import numpy as np
```

```
fig = plt.figure()
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

실행코드

실행화면

Axes

- axes
  - 데이터가 표현되는 영역으로 axis, ticks, line, text 등의 artist를 포함하고 있음
  - figure 생성 후 add_subplot 메소드를 통해 생성하거나, subplot함수를 통해 생성 가능
  - axes를 생성할 때 projection 옵션을 통해 3d 그래프로 변환 가능
  - 자주 사용되는 내장 메소드 : set_xlim(), set_ylim(), set_title(), set_xlabel(), set_ylabel()
  - axes.spines 를 통해 axes 좌표 축 조정 가능
    - Ex) ax.spines['right'].set_visible(False) =〉 우측 axes 경계 선 삭제 됨

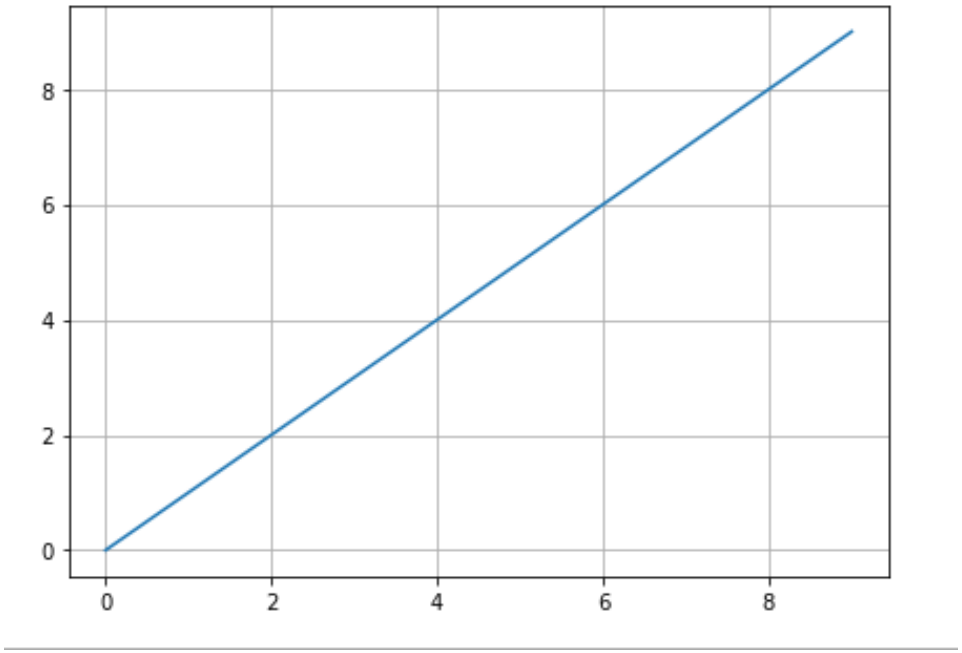
Pyplot

함수	설명	함수	설명
arrow()	Add an arrow to the axes.	plot()	Plot y versus x as lines and/or markers.
axhline()	Add a horizontal line across the axis.	plot_date()	Plot data that contains dates.
axhspan()	Add a horizontal span (rectangle) across the axis.	polar()	Make a polar plot.
axvline()	Add a vertical line across the axes.	savefig()	Save the current figure.
axvspan()	Add a vertical span (rectangle) across the axes.	scatter()	A scatter plot of *y* vs *x* with varying marker size and/or color.
bar()	Make a bar plot.	setp()	Set a property on an artist object.
box()	Turn the axes box on or off.	stem()	Create a stem plot.
colors()		step()	Make a step plot.
Figledend	Place a legend in the figure.	subplot()	Return a subplot axes at the given grid position.
figtext()	Add text to figure.	suptitle()	Add a centered title to the figure.
fill()	Plot filled polygons.	table()	Add a table to the current axes.
grid()	Turn the axes grids on or off.	text()	Add text to the axes.
hist()	Plot a histogram.	tick_params()	Change the appearance of ticks, tick labels, and gridlines.
hist2d()	Make a 2D histogram plot.	title()	Set a title of the current axes.
imsave()	Save an array as in image file.	xlabel() / ylabel()	Set the x-axis label of the current axes.
legend()	Places a legend on the axes.	xlim() / ylim()	Get or set the x limits of the current axes.
locator_params()	Control behavior of tick locators.	xscale() / yscale()	Set the scaling of the x-axis.
pie()	Plot a pie chart.	xticks() / yticks()	Get or set the current tick locations and labels of the x-axis.

Plot 함수

- plot 함수
  - 선 그래프를 그리는 함수
  - 실행하면 Line2D 객체 반환
- figure 함수의 figsize
  - (width, height) tuple in inches
- grid 함수
  - 바탕면에 grid 추가
- pyplot.show()
  - pyplot의 figure를 보이게 하는 함수

```
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t)
plt.grid(True)
plt.show()
```



plot 데이터 유형

- 입력 값 유형
  - np.array, np.ma.masked_array
  - pandas data object, np.matrix 형식은 오류 발생 가능성 존재

- pandas.DataFrame 의 사용

```
a = pandas.DataFrame(np.random.rand(4,5), columns=list('abcde'))
a_asndarray = a.values
```

- np.matrix 의 사용

```
b = np.matrix([[1,2],[3,4]])
b_asarray = np.asarray(b)
```

Style

❑ plot 함수 스타일 설정

- color, marker, line style
- 실행하면 Line2D 객체 반환

```
plt.clf()
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t,'g.:',)
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

✓ color

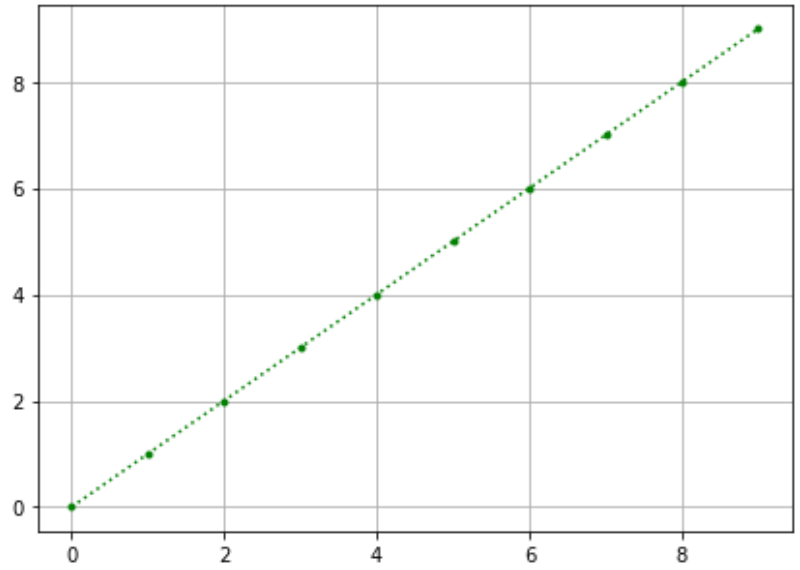
의미	약자
blue	b
green	g
red	r
cyan	c
magenta	m
yellow	y
black	k
white	w

✓ marker

기호	의미	기호	의미
.	point	5	tri_right
,	pixel	s	square
o	circle	p	pentagon
v	triangle_up	*	star
^	triangle_down	h	hexagon1
<	triangle_left	H	hexagon2
>	triangle_right	+	plus
1	tri_down	x	x
2	tri_up	D	diamond
4	tri_left	d	thin_diamond

✓ Line style

기호	의미
-	solid
--	dashed
-.	dash-dot
:	dotted



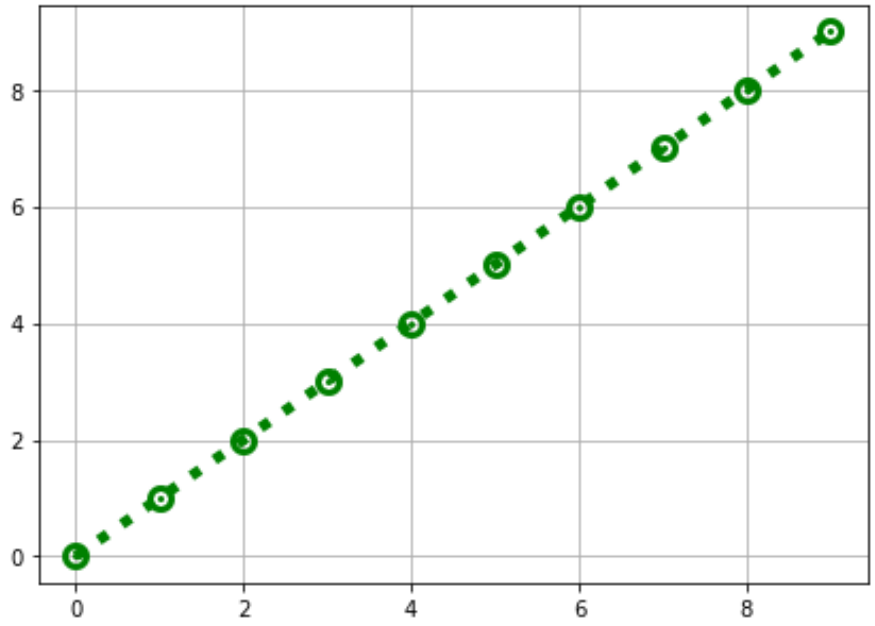
plot 함수 스타일

❑ plot 함수 추가 스타일 설정

기호	의미	의미
ls	linestyle	선 스타일
lw	linewidth	선 굵기
	marker	마커 종류
ms	markersize	마커크기
mec	markeredgecolor	마커 선 색깔
mew	markeredgewidth	마커 선 굵기
mfc	markerfacecolor	마커 내부 색깔

```
plt.clf()
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t,'g.:', lw=5, mew=10)
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Axes 범위 지정

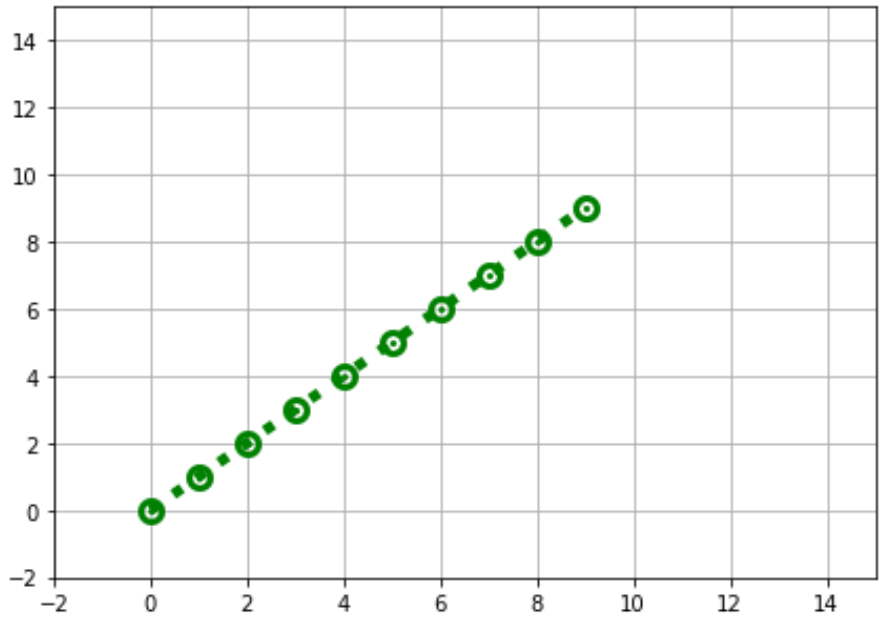
- X, Y 축의 범위 지정
  - X, Y 각 범위가 같을 필요 없음
  - 지정하지 않을 시 plot에 맞춰 자동 지정

- X-축은 pyplot.xlim() 함수
  - xlim((xmin, xmax))
  - xlim(xmin,xmax)

- Y-축은 pyplot.ylim() 함수
  - ylim((ymin, ymax))
  - ylim(ymin, ymax)

```
plt.clf()
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t,'g.:', lw=5, mew=10)
plt.xlim((-2,15))
plt.ylim((-2,15))
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



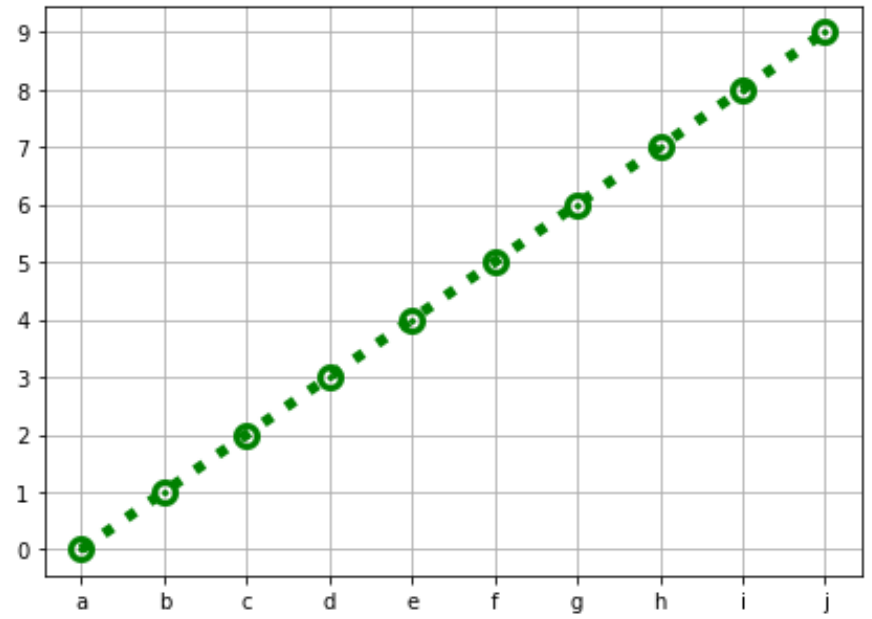
Tick 지정

- X, Y 축의 눈금 간격 지정
  - X, Y 각 범위가 같을 필요 없음
  - 지정하지 않을 시 plot에 맞춰 자동 지정

- pyplot.xticks(), pyplot.yticks() 함수
  - xticks(locs=숫자배열[, labels=문자배열]))
  - Locs는 필수 이며, labels는 옵션
  - Ticks를 사용하지 않으려면 Locs= [] (빈 배열)

```
plt.clf()
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t,'g.:', lw=5, mew=10)
plt.xticks(t, 'abcdefghijklmnopqrstuvwxyz')
plt.yticks(t)
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



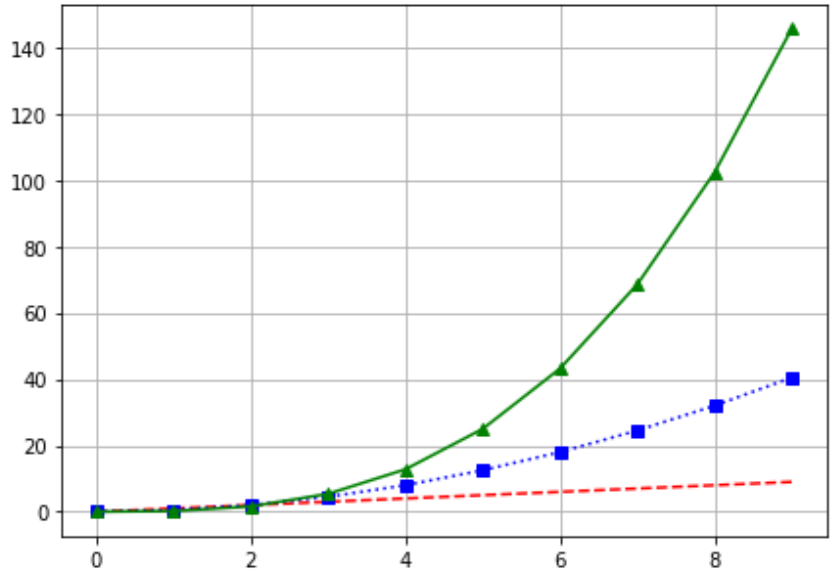
Plot 중첩

- plot([x], y, [fmt], data=None, **kwargs)
  - 한 개의 선을 표현 할 때

- plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
  - 여러 도표를 표현할 때
  - X축 값 생략 가능하고 fmt(format) 생략 가능
  - plot(y, y2 [, x2], y3, …, **kwargs) 가능

```
plt.clf()
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t, t, 'r--', t, 0.5 * t**2, 'bs:', t, 0.2 * t**3, 'g^-')
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Legend

- pyplot.legend() 함수
  - plot 함수 호출 시 label value setting
  - plot 함수의 label 키워드는 legend 만을 위함
  - loc=10은 legend 위치 지정

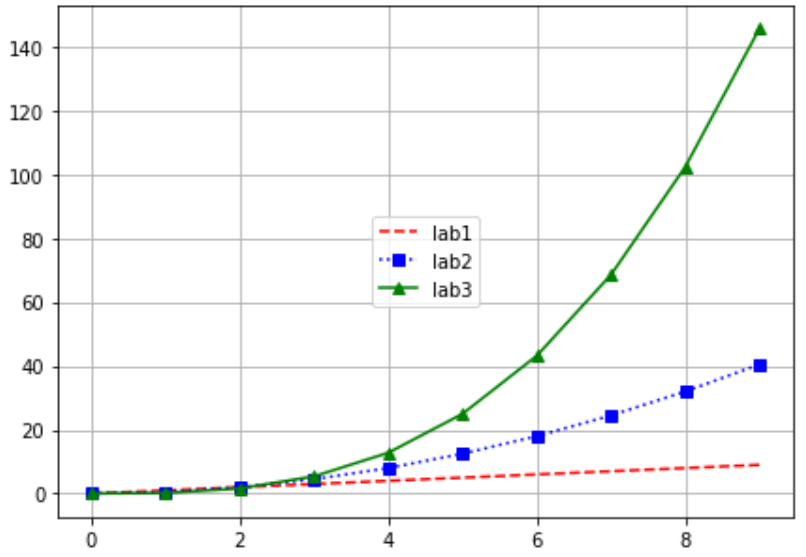
Location String	code	Location String	code	Location String	code
'best'	0	'lower right'	4	'lower center'	8
'upper right'	1	'right'	5	'upper center'	9
'upper left'	2	'center left'	6	'center'	10
'lower left'	3	'center right'	7		

- plot 함수 하나에서 여러 그래프 호출 시
  - plot 함수를 통해 각 그래프에 label value setting 불가
  - legend 함수를 통해 label values setting

```
plt.plot(t, t, 'r--', t, 0.5 * t**2, 'bs:', t, 0.2 * t**3, 'g^-')
plt.legend(labels=['lab1','lab2','lab3'])
```

```
plt.clf()
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t, t, 'r--', label='lab1')
plt.plot(t, 0.5 * t**2, 'bs:', label='lab2')
plt.plot(t, 0.2 * t**3, 'g^-', label='lab3')
plt.legend(loc=10)
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



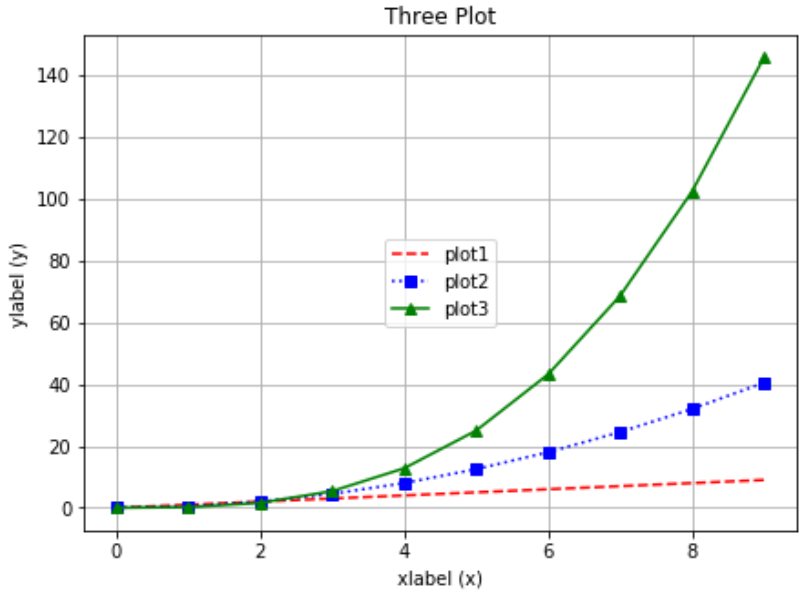
Title / Label

- pyplot.title()
  - label : title text
  - fontdict : font style
  - loc : center, left, right

- pyplot.xlabel(), pyplot.ylabel()
  - label : text
  - fontdict : font style
  - labelpad : Spacing in points between the label and the axis

```
plt.clf()
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t, t, 'r--', t, 0.5 * t**2, 'bs:', t, 0.2 * t**3, 'g^-')
plt.legend(labels=['plot1','plot2','plot3'],loc='center')
plt.xlabel("xlabel (x)")
plt.ylabel("ylabel (y)")
plt.title("Three Plot")
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

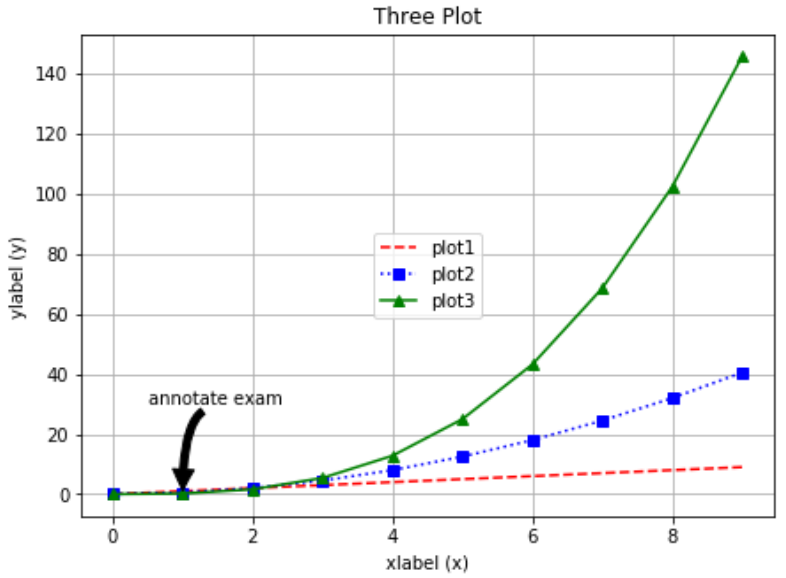


Annotation

- pyplot.annotate()
  - s : text of annotate
  - xy : (x,y) 형태, point 위치
  - xytext : (x,y) 형태, text 시작 위치
  - arrowprops : 화살표 설정, dictionary 형태
    - ✓ facecolor : 색
    - ✓ shrink : point 지점과 화살표와의 간격
    - ✓ connectionstyle : 화살표 연결선 스타일
    - ✓ arrowstyle : 화살표 스타일

```
plt.clf()
t = np.arange(10)
plt.figure(figsize=(7,5))
plt.plot(t, t, 'r--', t, 0.5 * t**2, 'bs:', t, 0.2 * t**3, 'g^-')
plt.legend(labels=['plot1','plot2','plot3'],loc='center')
plt.xlabel("xlabel (x)")
plt.ylabel("ylabel (y)")
plt.title("Three Plot")
plt.annotate('annotate exam', xy=(1, 1), xytext=(0.5, 30),
             arrowprops=dict(facecolor='black', shrink=0.05,
             connectionstyle='angle3,angleA=0,angleB=90'),
             )
plt.grid(True)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



subplot

- pyplot.subplot()
  - 한 figure에 여러 axes 사용 시 사용
  - pyplot.subplot(nrows, ncols, index)
    - ✓ nrows : 나눌 행 입력
    - ✓ ncols : 나눌 열 입력
    - ✓ Index : axes index(몇 번째 axes)
  - 예시
    - ✓ pyplot.subplot(2,2,2)
    - ✓ 2 by 2로 네 개의 axes 영역 생성

11 pyplot.subplot(2,2,1)	12 pyplot.subplot(2,2,2)
21 pyplot.subplot(2,2,3)	22 pyplot.subplot(2,2,4)

Subplot

- pyplot.subplot()
  - pyplot.subplot(2, 1, 1)
    - ✓ 2 by 1 로 2개의 axes 영역 생성

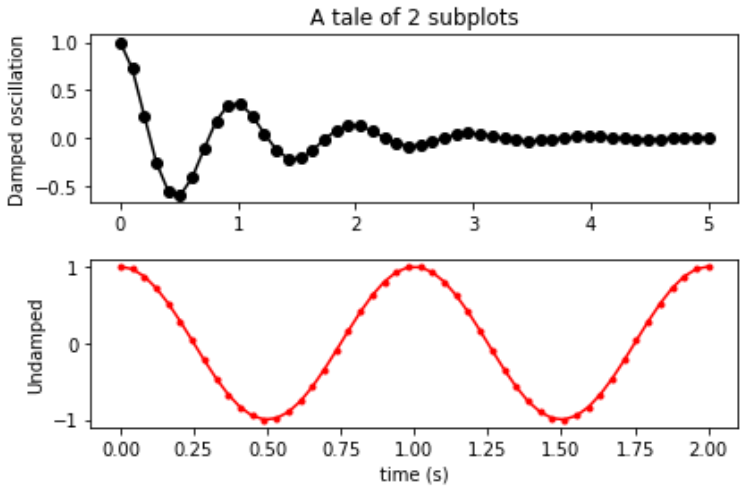
11 pyplot.subplot(2,1,1)
21 pyplot.subplot(2,1,2)

    - ✓ nrow, ncol이 동일할 때만 정상적으로 생성

11 pyplot.subplot(2,1,1)	
	21 pyplot.subplot(2,1,2)

```
plt.clf()
x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

ax1 = plt.subplot(2, 1, 1) # plt.subplot(211) 동일
plt.plot(x1, y1, 'ko-')
plt.title('A tale of 2 subplots')
plt.ylabel('Damped oscillation')
print(ax1)

ax2 = plt.subplot(2, 1, 2) # plt.subplot(212) 동일
plt.plot(x2, y2, 'r.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')
print(ax2)

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

```
AxesSubplot(0.125,0.536818;0.775x0.343182)
AxesSubplot(0.125,0.125;0.775x0.343182)
```



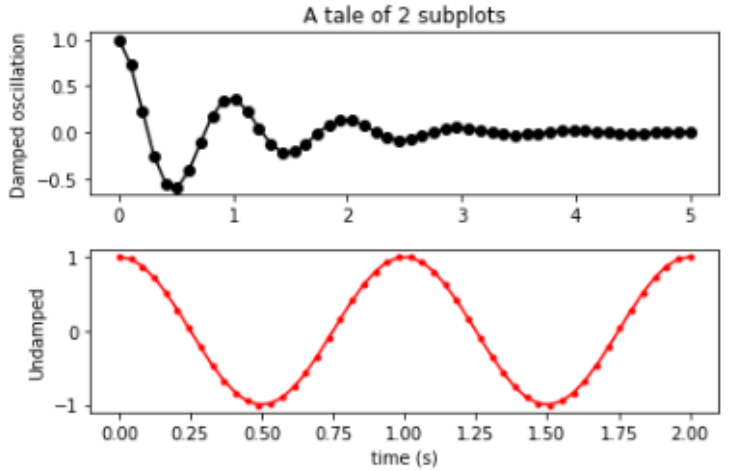
subplots

- pyplot.subplots(nrows=2, ncols=1)
  - 2 by 1 로 2개의 axes 영역 생성
  - figure, axes 반환
  - Axes 반환 시 axes가 여러 개 일 경우 ndarray 형태로 반환

- axes 객체 메소드 활용
  - Axes.plot()
  - Axes.set_title(), Axes.set_xlabel(), axes.set_ylabel()
  - Pyplot.plot(), pyplot.title(), pyplot.xlabel() 등과 동일한 기능

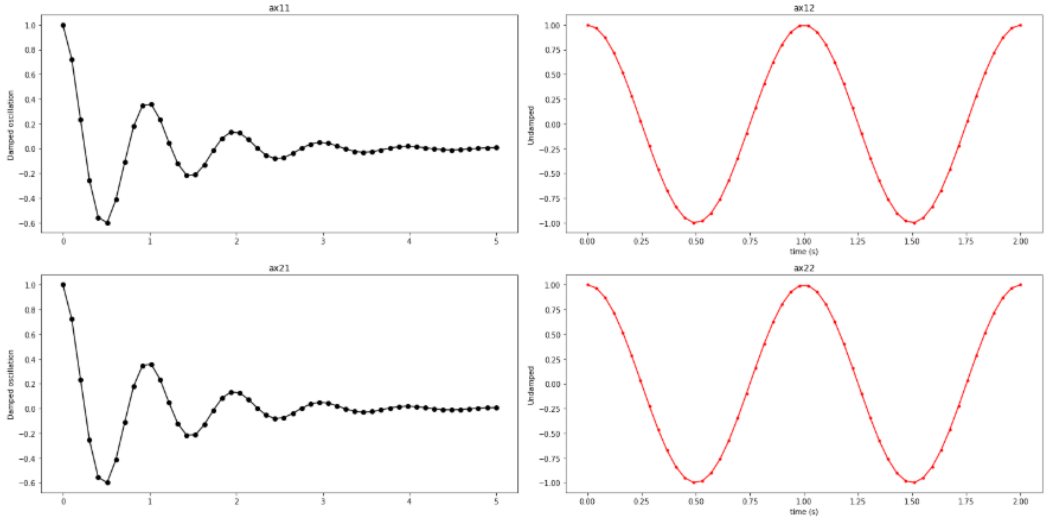
```
plt.clf()
x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

fig, (ax1,ax2) = plt.subplots(nrows=2, ncols=1) #
ax1.plot(x1, y1, 'ko-')
ax1.set_title('A tale of 2 subplots')
ax1.set_ylabel('Damped oscillation')
print(ax1)

ax2.plot(x2, y2, 'r.-')
ax2.set_xlabel('time (s)')
ax2.set_ylabel('Undamped')
print(ax2)

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

```
AxesSubplot(0.125,0.536818;0.775x0.343182)
AxesSubplot(0.125,0.125;0.775x0.343182)

<Figure size 432x288 with 0 Axes>
```



subplots

- pyplot.subplots(nrows=2, ncols=2)
  - 2 by 2 axes 생성
  - ((ax11,ax12), (ax21,ax22)) 로 axes 생성



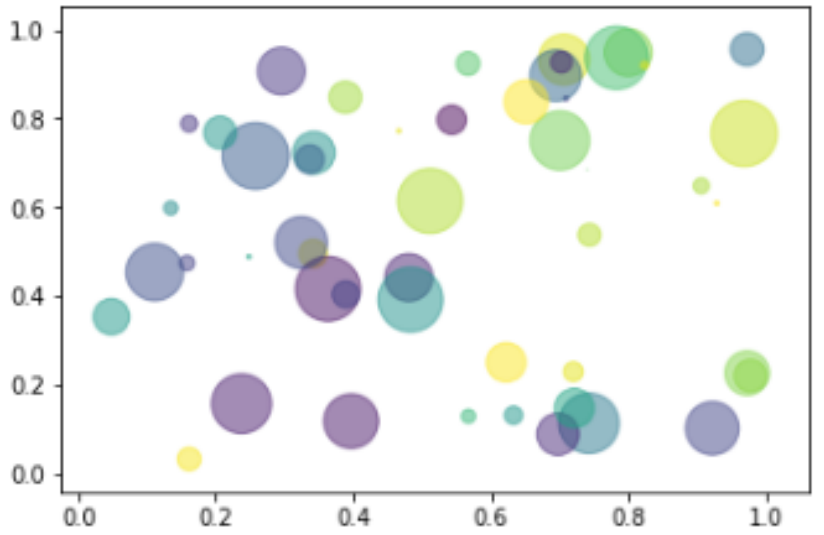

```
plt.clf()
x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

fig, ((ax11,ax12),(ax21,ax22)) = plt.subplots(nrows=2, ncols=2, figsize=(20,10))
ax11.plot(x1, y1, 'ko-')
ax11.set_ylabel('Damped oscillation')
ax11.set_title('ax11')
print(ax11)

ax12.plot(x2, y2, 'r.-')
ax12.set_xlabel('time (s)')
ax12.set_ylabel('Undamped')
ax12.set_title('ax12')
print(ax12)

ax21.plot(x1, y1, 'ko-')
ax21.set_ylabel('Damped oscillation')
ax21.set_title('ax21')
print(ax21)

ax22.plot(x2, y2, 'r.-')
ax22.set_xlabel('time (s)')
ax22.set_ylabel('Undamped')
ax22.set_title('ax22')
print(ax22)

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

```
AxesSubplot(0.125,0.536818;0.352273x0.343182)
AxesSubplot(0.547727,0.536818;0.352273x0.343182)
AxesSubplot(0.125,0.125;0.352273x0.343182)
AxesSubplot(0.547727,0.125;0.352273x0.343182)
```

Scatter Plot

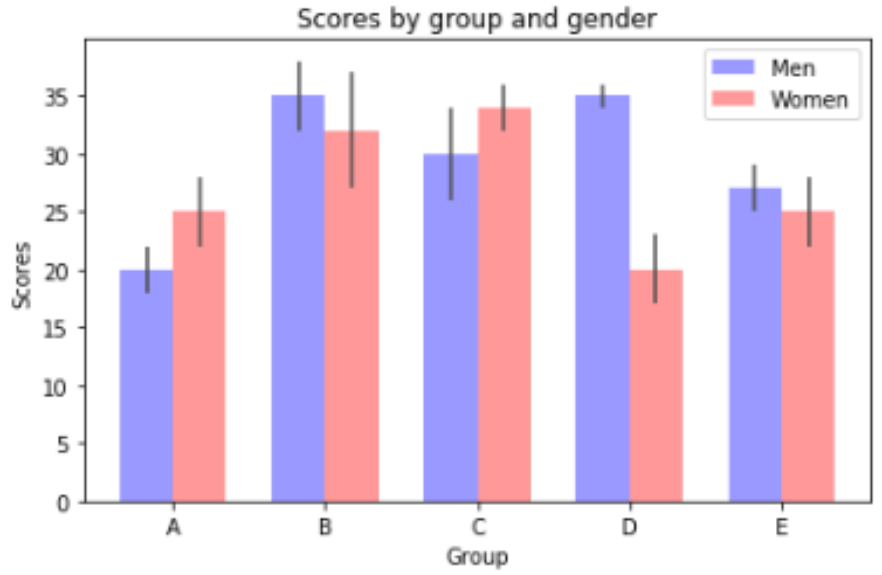
- pyplot.scatter()
  - 데이터가 2차원 이상인 경우 마커의 크기 혹은 색깔을 이용하여 표현
  - s : marker 크기
  - c : marker 색깔

```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)


N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2  # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



Bar plot

- pyplot.bar()
  - 데이터가 카테고리인 경우 주로 사용
  - bar()는 세로 방향의 막대 그래프

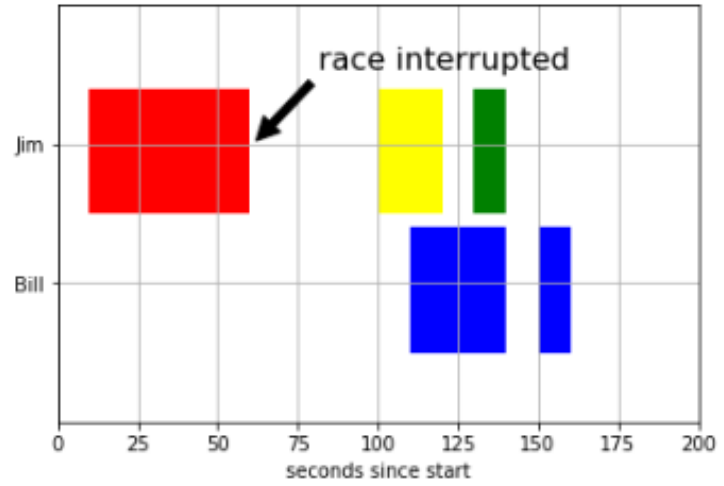



```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
from collections import namedtuple

n_groups = 5

means_men = (20, 35, 30, 35, 27)
std_men = (2, 3, 4, 1, 2)

means_women = (25, 32, 34, 20, 25)
std_women = (3, 5, 2, 3, 3)

fig, ax = plt.subplots()

index = np.arange(n_groups)
bar_width = 0.35

opacity = 0.4
error_config = {'ecolor': '0.3'}

rects1 = ax.bar(index, means_men, bar_width,
                alpha=opacity, color='b',
                yerr=std_men, error_kw=error_config,
                label='Men')

rects2 = ax.bar(index + bar_width, means_women, bar_width,
                alpha=opacity, color='r',
                yerr=std_women, error_kw=error_config,
                label='Women')

ax.set_xlabel('Group')
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(('A', 'B', 'C', 'D', 'E'))
ax.legend()

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

Barh plot

- pyplot.barh()
  - 데이터가 카테고리인 경우 주로 사용
  - barh()는 가로 방향의 막대 그래프

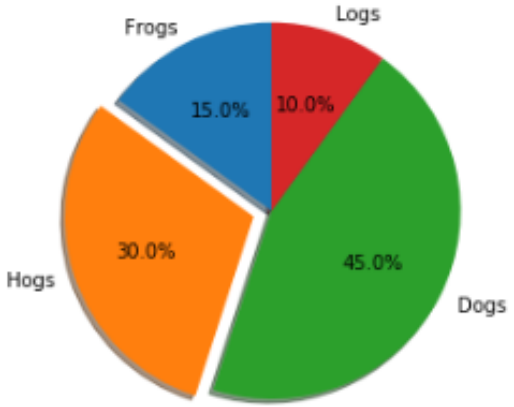
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.broken_barh([(110, 30), (150, 10)], (10, 9), facecolors='blue')
ax.broken_barh([(10, 50), (100, 20), (130, 10)], (20, 9),
               facecolors=('red', 'yellow', 'green'))
ax.set_ylim(5, 35)
ax.set_xlim(0, 200)
ax.set_xlabel('seconds since start')
ax.set_yticks([15, 25])
ax.set_yticklabels(['Bill', 'Jim'])
ax.grid(True)
ax.annotate('race interrupted', (61, 25),
            xytext=(0.8, 0.9), textcoords='axes fraction',
            arrowprops=dict(facecolor='black', shrink=0.05),
            fontsize=16,
            horizontalalignment='right', verticalalignment='top')

plt.show()
```



Pie plot

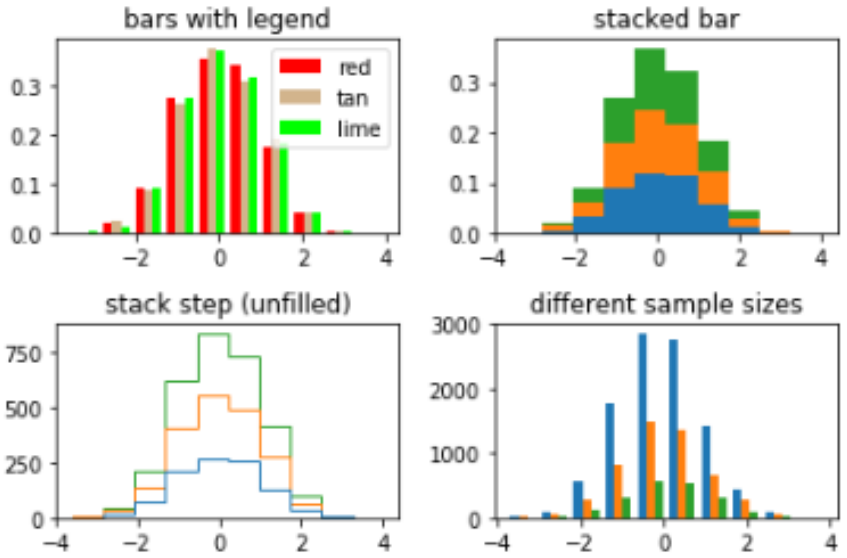
- pyplot.pie()
  - 카테고리 별 상대적 비교를 할 때 주로 사용
  - plt.axis('equal') : 원의 형태를 유지

```
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```



Histogram

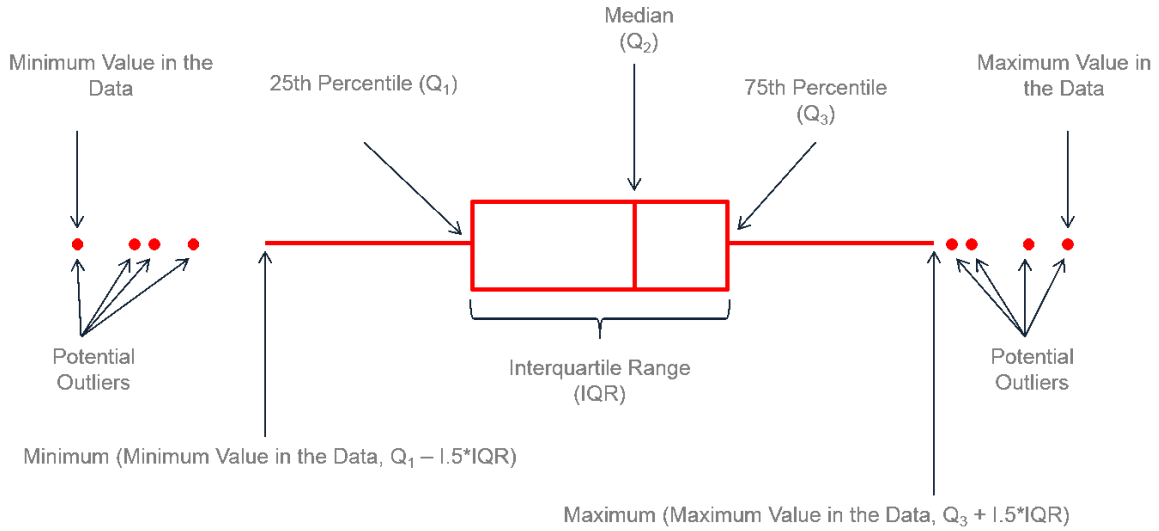
- pyplot.hist()
  - 구간별 분포도를 표현할 때 주로 사용




```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(19680801)

n_bins = 10
x = np.random.randn(1000, 3)

fig, axes = plt.subplots(nrows=2, ncols=2)
ax0, ax1, ax2, ax3 = axes.flatten()

colors = ['red', 'tan', 'lime']
ax0.hist(x, n_bins, density=True, histtype='bar', color=colors, label=colors)
ax0.legend(prop={'size': 10})
ax0.set_title('bars with legend')

ax1.hist(x, n_bins, density=True, histtype='bar', stacked=True)
ax1.set_title('stacked bar')

ax2.hist(x, n_bins, histtype='step', stacked=True, fill=False)
ax2.set_title('stack step (unfilled)')

# Make a multiple-histogram of data-sets with different length.
x_multi = [np.random.randn(n) for n in [10000, 5000, 2000]]
ax3.hist(x_multi, n_bins, histtype='bar')
ax3.set_title('different sample sizes')

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

boxplot

- pyplot.boxplot()
  - 많은 데이터를 눈으로 확인하기 어려울 때 그림을 이용해 데이터 집합의 범위와 중앙값을 빠르게 확인할 수 있는 목적

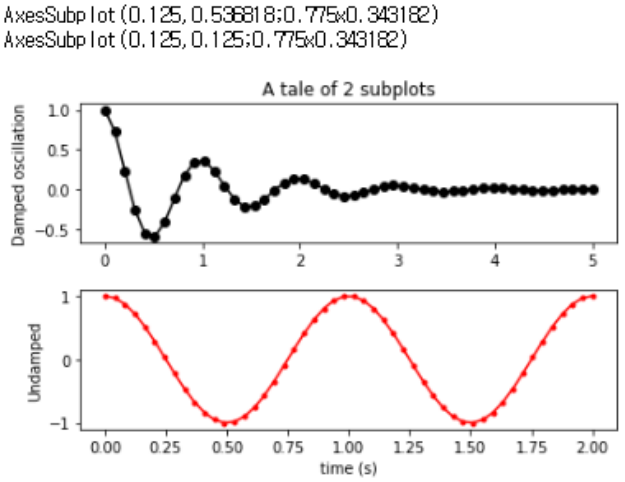


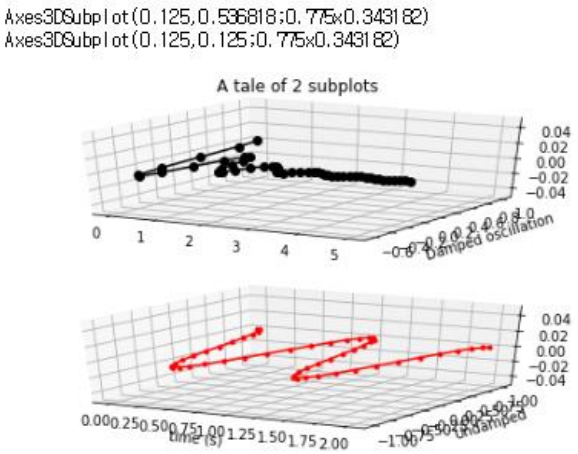

```python
import matplotlib.pyplot as plt
import numpy as np

# Random test data
np.random.seed(19680801)
all_data = [np.random.normal(0, std, size=100) for std in range(1, 4)]
labels = ['x1', 'x2', 'x3']

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(9, 4))

# rectangular box plot
bplot1 = axes[0].boxplot(all_data,
                         vert=True,  # vertical box alignment
                         patch_artist=True,  # fill with color
                         labels=labels)  # will be used to label x-ticks
axes[0].set_title('Rectangular box plot')

# notch shape box plot
bplot2 = axes[1].boxplot(all_data,
                         notch=True,  # notch shape
                         vert=True,  # vertical box alignment
                         patch_artist=True,  # fill with color
                         labels=labels)  # will be used to label x-ticks
axes[1].set_title('Notched box plot')

# fill with colors
colors = ['pink', 'lightblue', 'lightgreen']
for bplot in (bplot1, bplot2):
    for patch, color in zip(bplot['boxes'], colors):
        patch.set_facecolor(color)

# adding horizontal grid lines
for ax in axes:
    ax.yaxis.grid(True)
    ax.set_xlabel('Three separate samples')
    ax.set_ylabel('Observed values')

plt.show()
```

3D plot

❑ 3d plot

- axes 생성 시 projection='3d'로 셋팅






```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)


ax1 = plt.subplot(2, 1, 1, projection='3d')
plt.plot(x1, y1, 'ko-')
plt.title('A tale of 2 subplots')
plt.ylabel('Damped oscillation')
print(ax1)

ax2 = plt.subplot(2, 1, 2, projection='3d')
plt.plot(x2, y2, 'r.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')
print(ax2)

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