8 시각화

Matplotlib

Matplotlib : 차트(chart)나 플롯(plot)으로 시각화하는 패키지

- 설치 : pip install matplotlib
- %matplotlib inline : jupyter notebook 내부에 그림을 표시하는 명령어
- Matplotlib 예제사이트 : http://matplotlib.org/gallery.html

Matplotlib import 방법

```
1 import matplotlib
2 import matplotlib.pyplot as plt
3
4 %matplotlib inline
```

Jupyter Matplotlib 한글 사용법

- http://hangeul.naver.com/2017/nanum 설치

```
1 import os
2
3 if os.name=='posix':
4     plt.rc('font',family='AppleGothic')
5 else:
6     plt.rc('font',family='Malgun Gothic')
```

Matplotlib

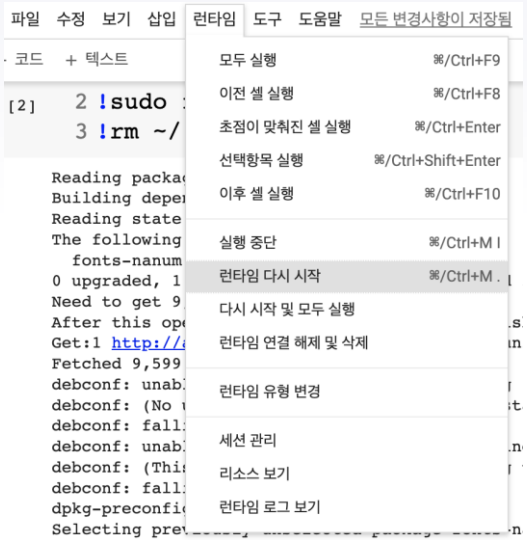
colab에서 matplotlib 한글 사용방법

1. 한글 폰트 설치

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

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  fonts-nanum
0 upgraded, 1 newly installed, 0 to remove and 23 not upgraded.
Need to get 9,599 kB of archives.
After this operation, 29.6 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 fonts-nanum all 20180306-3 [9,599
Fetched 9,599 kB in 2s (5,191 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be u
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package fonts-nanum.
(Reading database ... 128285 files and directories currently installed.)
Preparing to unpack .../fonts-nanum_20180306-3_all.deb ...
Unpacking fonts-nanum (20180306-3) ...
Setting up fonts-nanum (20180306-3) ...
Processing triggers for fontconfig (2.13.1-2ubuntu3) ...
/usr/share/fonts: caching, new cache contents: 0 fonts, 1 dirs
/usr/share/fonts/truetype: caching, new cache contents: 0 fonts, 3 dirs
/usr/share/fonts/truetype/humor-sans: caching, new cache contents: 1 fonts, 0 dirs
/usr/share/fonts/truetype/liberation: caching, new cache contents: 16 fonts, 0 dirs
/usr/share/fonts/truetype/nanum: caching, new cache contents: 10 fonts, 0 dirs
/usr/local/share/fonts: caching, new cache contents: 0 fonts, 0 dirs
/root/.local/share/fonts: skipping, no such directory
/root/.fonts: skipping, no such directory
/usr/share/fonts/truetype: skipping, looped directory detected
/usr/share/fonts/truetype/humor-sans: skipping, looped directory detected
/usr/share/fonts/truetype/liberation: skipping, looped directory detected
/usr/share/fonts/truetype/nanum: skipping, looped directory detected
```

2. 런타임 재시작



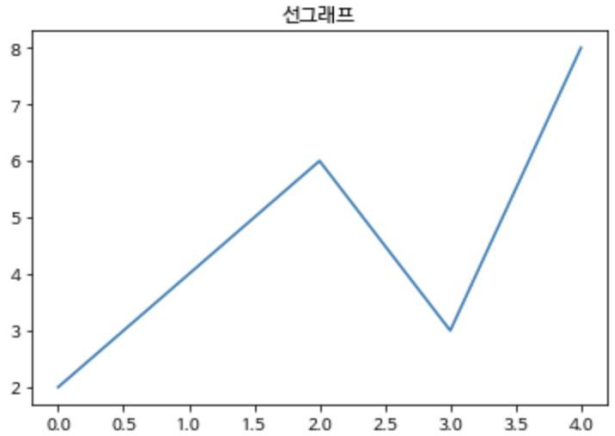
3. 폰트 등록

```
plt.rc('font', family='NanumBarunGothic')
```

Matplotlib

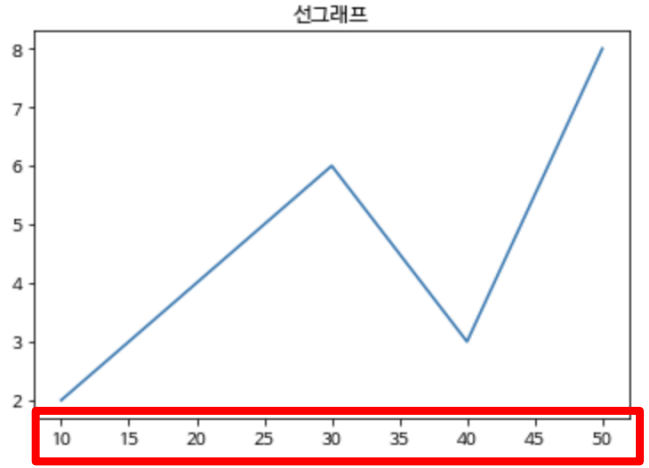
line plot : 선그래프

```
1 plt.title('선그래프')
2 plt.plot([2,4,6,3,8])
3 plt.show()
```



xtick 설정

```
1 plt.title('선그래프')
2 plt.plot([10,20,30,40,50],[2,4,6,3,8])
3 plt.show()
```

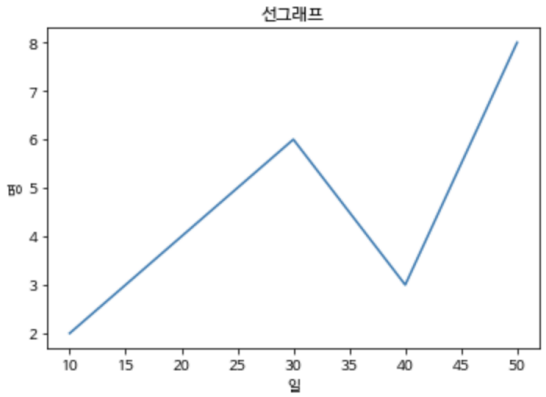


xtick

Matplotlib

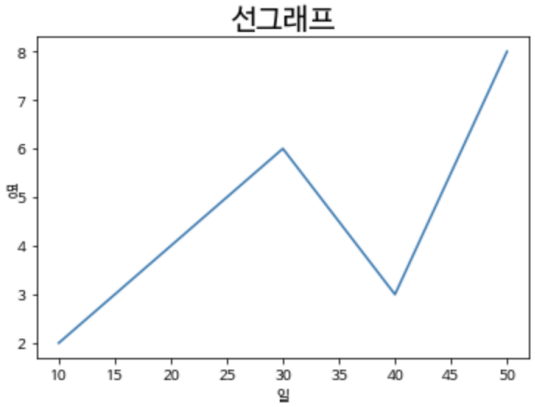
xlabel : xtick 이름　ylabel : ytick이름

```
1 plt.title('선그래프')
2 plt.plot([10,20,30,40,50],[2,4,6,3,8])
3 plt.xlabel('일')
4 plt.ylabel('명')
5 plt.show()
```



size : 타이틀 사이즈 , rotation : label 회전

```
1 plt.title('선그래프' , size=20)
2 plt.plot([10,20,30,40,50],[2,4,6,3,8])
3 plt.xlabel('일')
4 plt.ylabel('명',rotation=0)
5 plt.show()
```



Style

색지정

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

마커

. : point marker
, : pixel marker
o : circle marker
v : riangle_down marker
^ : riangle_up marker
< : triangle_left marker
> : triangle_right marker
1 : tri_down marker
2 : tri_up marker
3 : tri_left marker
4 : tri_right marker
s : square marker
p : pentagon marker
* : star marker
h : hexagon1 marker
H : hexagon2 marker
+ : plus marker
x : x marker
D : diamond marker
d : thin_diamond marker

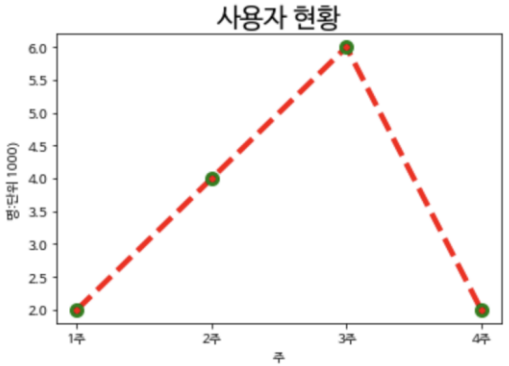
선스타일

- : solid line style
-- : dashed line style
-. : dash-dot line style
: : dotted line style

스타일 type

color(선 색깔) : c
linewidth(선 굵기) : lw
linestyle(선 스타일) : ls
marker(마커 종류)
markersize(마커 크기) : ms
markeredgecolor(마커 선 색) : mec
markeredgewidth(마커 선 굵기) : mew
markerfacecolor(마커 내부 색깔) : mfc

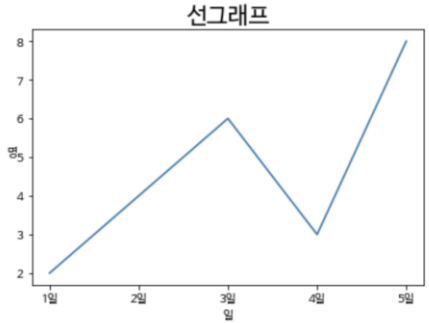
```
plt.title('사용자 현황' , size = 20)
plt.plot([1,2,3,4] , [2,4,6,2] , c = 'r' ,
         lw = 4 , ls = '--' , marker = 'o',
         ms = 8 , mec = 'g' , mew = 3, )

plt.xticks([1,2,3,4],['1주','2주','3주','4주'])
plt.xlabel('주')
plt.ylabel('명:단위 1000)' , rotation = 90)
plt.show()
```



Style

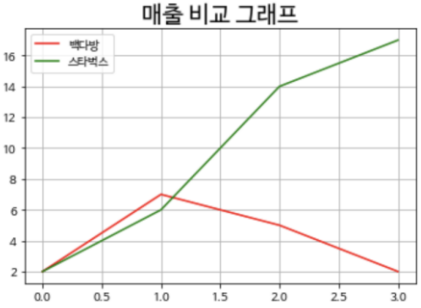
xticks 이름변경

```
plt.title('선그래프' , size=20)
plt.plot([10,20,30,40,50],[2,4,6,3,8])
plt.xticks([10,20,30,40,50],['1일','2일','3일','4일','5일'])
plt.xlabel('일')
plt.ylabel('명',rotation=0)
plt.show()
```



grid : 격자 , legend : 범례 (선이 무슨의미인지 표시)

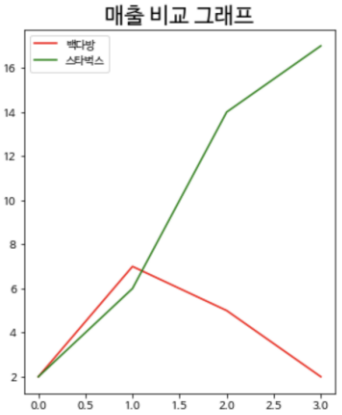
```
plt.title('매출 비교 그래프' , size = 20)
plt.plot([2,7,5,2] , c = 'r' , label='백다방')
plt.plot([2,6,14,17] , c = 'g' , label='스타벅스')
plt.grid(True)
plt.legend()
plt.show()
```



Figure

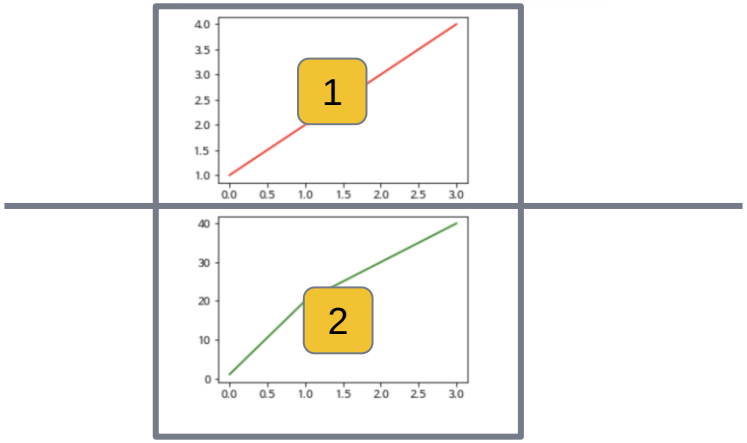
figure : 그래프가 그려지는 영역 (canvas)
plt.figure(fizsize = (가로사이즈,세로사이즈))

```
plt.figure(figsize=(5,6))
plt.title('매출 비교 그래프' , size = 20)
plt.plot([2,7,5,2] , c = 'r' , label='백다방')
plt.plot([2,6,14,17] , c = 'g' , label='스타벅스')

plt.legend()
plt.show()
```



subplot : figure안에 여러개의 그래프를 그릴때 사용
plt.subplot(행, 열, 순서)

```
plt.figure(figsize=(4,6))
plt.subplot(2,1,1)
plt.plot([1,2,3,4],c='r')

plt.subplot(2,1,2)
plt.plot([1,20,30,40],c='g')

plt.show()
```

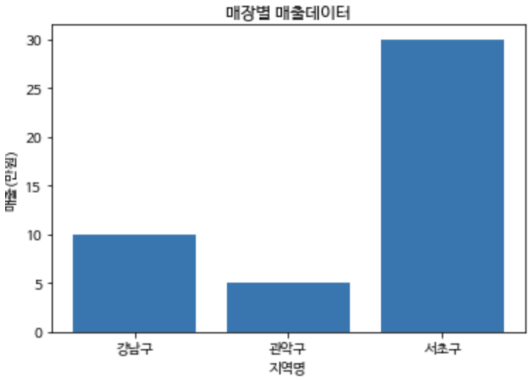


bar chart

더 많은 막대그래프 및 다른 종류의 그래프를 그리는 방법은 matplotlib 문서를 참고

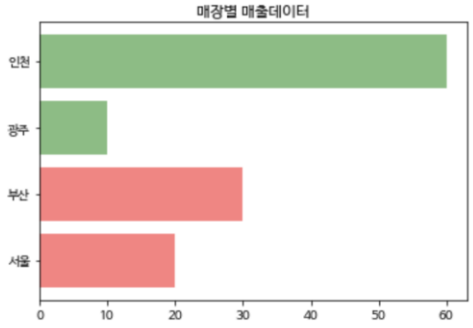
bar : 세로차트

```
1 plt.title('매장별 매출데이터')
2 plt.bar([0,1,2],[10,5,30])
3 plt.xticks([0,1,2],['강남구','관악구','서초구'])
4 plt.xlabel('지역명')
5 plt.ylabel('매출(만원)')
6 plt.show()
```



barh : 가로차트 **alpha : 투명도**

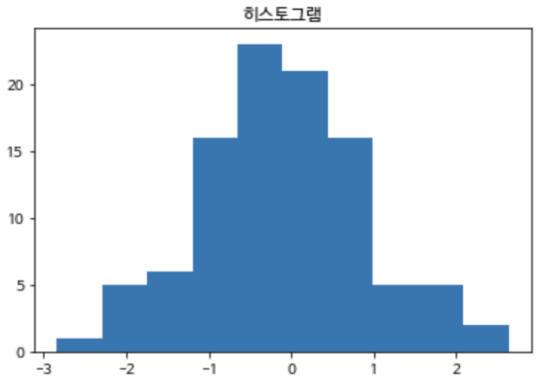
```
1 plt.title('매장별 매출데이터')
2 city = ['서울','부산','광주','인천']
3 y_pos = [0,1,2,3]
4 data = [20,30,10,60]
5 plt.barh(y_pos,data , alpha = 0.5 , color=['r','r','g','g'])
6 plt.yticks(y_pos,city)
7 plt.show()
```



다양한 그래프

hist : 히스토그램(bins : 집계구간)

```
1 plt.rcParams['axes.unicode_minus'] = False
2 x = np.random.randn(100)
3 plt.title('히스토그램')
4 plt.hist(x,bins=10)
5 plt.show()
```

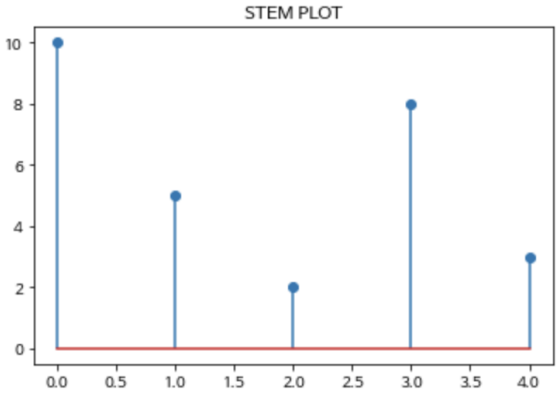


- 히스토그램은 도수분포표를 시각화한 그래프입니다.
- 데이터를일정한 간격으로 나누어 각 구간에 속하는 데이터의 개수를 막대 형태로 나타냅니다.
- 히스토그램을 통해 데이터의 분포를 쉽게 파악할 수 있습니다.

다양한 그래프

stem : stem plot (bar차트에 넓이가 없는 차트)

```
1 plt.title('STEM PLOT')
2 plt.stem([0,1,2,3,4],[10,5,2,8,3])
3 plt.show()
```

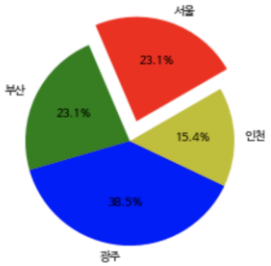


- stem plot은 수치 데이터를 좌표축에 점으로 찍어 표현하는 것이 아니라, 라인 플롯과 유사한 형태로 수직선을 그리고 그 위에 마크를 찍어 데이터를 시각화하는 방법입니다.
- 주로 시계열 데이터나 분포 데이터를 시각화할 때 사용합니다.

다양한 그래프

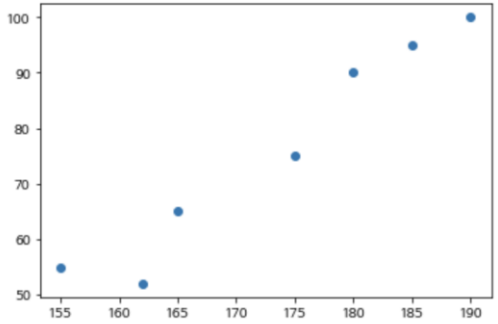
파이차트 : autopct (퍼센티지 자동계산) , shadow(그림자)

```
labels = ['서울','부산','광주','인천']
data = [30,30,50,20]
colors = ['r','g','b','y']
explode = [0.2, 0,0,0]
plt.pie(data,explode=explode,labels=labels,
        colors=colors,autopct='%1.1f%%',startangle=30)
plt.show()
```



scatter : 두데이터간의 상관관계 확인

```
data1 = [180,175,165,162,155,190,185]
data2 = [90,75,65,52,55,100,95]

plt.scatter(data1,data2)
plt.show()
```



Seaborn

seaborn

- Matplotlib을 기반으로 다양한 테마와 통계용 차트 등의 기능을 추가한 시각화 패키지
- 붓꽃 , 타이타닉, 팁, 여객운송 데이터를 기본으로 제공

데이터 불러오기

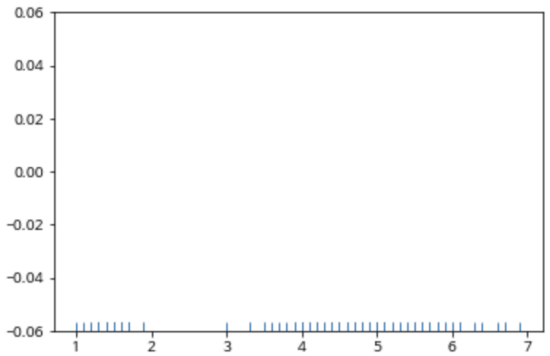
```
1 import seaborn as sns
2 iris = sns.load_dataset('iris')
3 titanic = sns.load_dataset('titanic')
4 tips = sns.load_dataset('tips')
5 flights = sns.load_dataset('flights')
```

```
1 iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

rugplot : 데이터 위치를 x축에 표현

```
1 x = iris.petal_length.values
2 sns.rugplot(x)
3 plt.show()
```



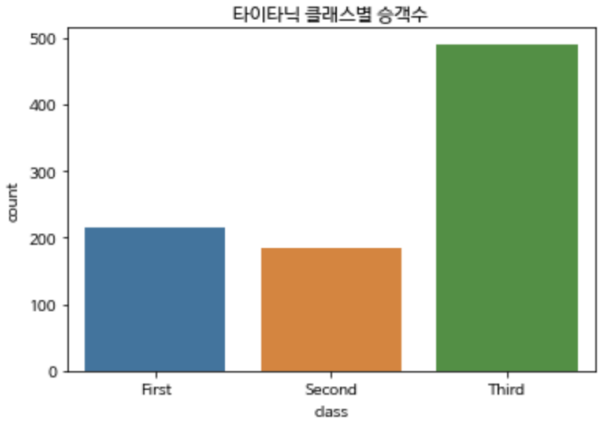
Seaborn

countplot : 카테고리별 데이터 갯수

Countplot은 범주형 변수의 각 카테고리별로 데이터가 몇 개씩 있는지를 시각화하는 데 사용되는 그래프입니다.

각 카테고리의 값들을 막대로 나타내어 각 값의 빈도수를 쉽게 비교할 수 있습니다.

```
1 sns.countplot(x='class' , data = titanic)
2 plt.title('타이타닉 클래스별 승객수')
3 plt.show()
```



Seaborn

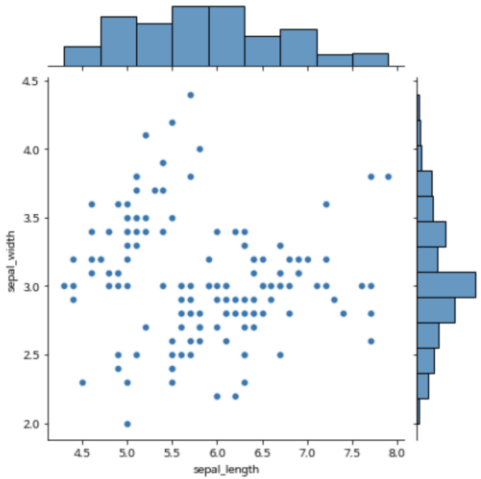
jointplot : 카테고리별 데이터 개수

- 산점도를 기본으로 표시하고 x,y축에 변수에 대한 히스토그램 표시
- 두 변수의 관계와 데이터가 분산되어 있는 정도 파악

Jointplot은 두 개의 수치형 변수 사이의 관계를 시각화하는 데 사용되는 그래프입니다.

두 변수 간의 산점도와 각 변수의 히스토그램을 함께 보여주어 두 변수 간의 관계를 쉽게 파악할 수 있습니다.

```
1 sns.jointplot(x='sepal_length',y='sepal_width',data=iris)
2 plt.show()
```



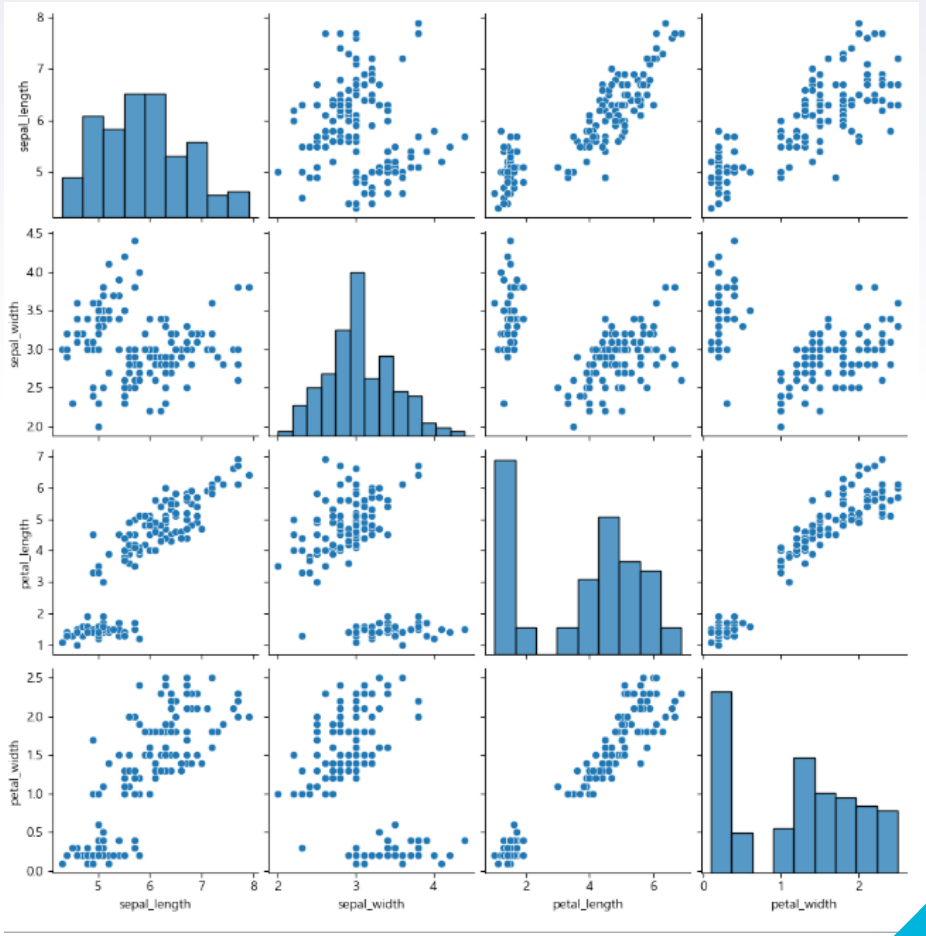
Seaborn

pairplot : 3차원 이상의 데이터 비교 분석

```
1 plt.figure(figsize=(6,6))
2 sns.pairplot(iris)
3 plt.show()
```

Pairplot 은 데이터셋의 모든 수치형 변수 쌍에 대한 산점도와, 각 변수의 히스토그램을 한번에 그려주는 그래프입니다. 이 그래프를 통해 변수 간의 관계를 한 눈에 파악할 수 있습니다.

Pairplot은 데이터셋 내 수치형 변수들의 관계를 시각화하는데 매우 유용합니다. 예를 들어, 데이터셋에서 특정 변수가 다른 변수와 어떤 관계를 가지는지, 변수 간의 상관관계가 있는지, 이상치가 존재하는지 등을 파악할 수 있습니다.



Seaborn

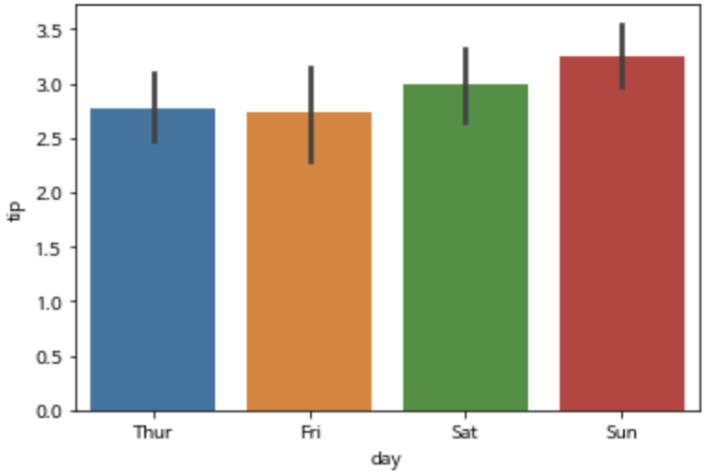
barplot

- 카테고리 값에 따른 실수 값의 평균과 편차를 표시
- 평균은 막대의 높이로, 편차는 에러바(error bar)로 표시

Bar plot은 범주형 변수의 각 카테고리별로 값을 시각화하는 데 사용되는 그래프입니다.

각 카테고리의 값들을 막대로 나타내어 각 값의 크기를 쉽게 비교할 수 있습니다.

```
1 sns.barplot(x='day',y='tip',data=tips)
2 plt.show()
```



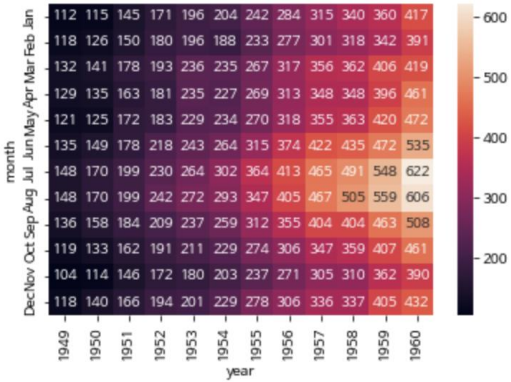
Seaborn

heatmap : 데이터의 값을 컬러로 변환시켜 시각적인 분석

Heatmap은 데이터셋의 각 변수 간 상관관계를 시각화하는 데 매우 유용한 그래프입니다.

변수 간의 상관관계를 색상으로 표현하여 한 눈에 쉽게 파악할 수 있습니다.

```
df = flights.pivot('month','year','passengers')
sns.heatmap(df,annot=True , fmt="d")
plt.show()
```



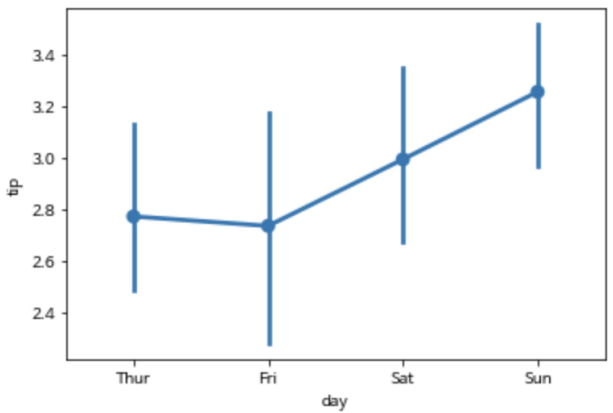
Seaborn

pointplot: 점 추정치 및 신뢰구간을 표시

Pointplot은 범주형 변수와 수치형 변수 사이의 관계를 시각화하는 데 사용되는 그래프입니다.

각 카테고리의 값들의 평균값을 점으로 나타내어 각 카테고리의 값들이 수치형 변수에 미치는 영향을 시각적으로 보여줍니다.

```
1 sns.pointplot(x='day',y='tip',data=tips)
2 plt.show()
```



Seaborn

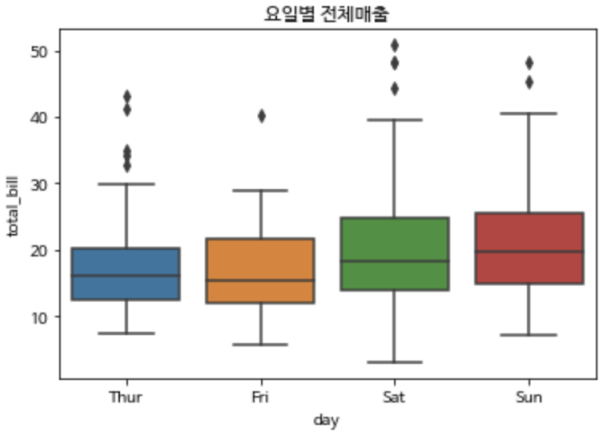
boxplot

- 박스-휘스커 플롯(Box-Whisker Plot) 혹은 간단히 박스 플롯이라 부르는 차트
- 박스와 박스 바깥의 선(whisker)으로 이루어짐

Boxplot은 수치형 변수의 분포와 이상치를 시각화하는 데 사용되는 그래프입니다.

데이터의 중앙값, 사분위수, 최소값, 최대값 등을 박스와 선으로 나타내어 데이터의 분포를 쉽게 파악할 수 있습니다.

```
1 plt.title('요일별 전체매출')
2 sns.boxplot(x='day',y='total_bill',data=tips)
3 plt.show()
```



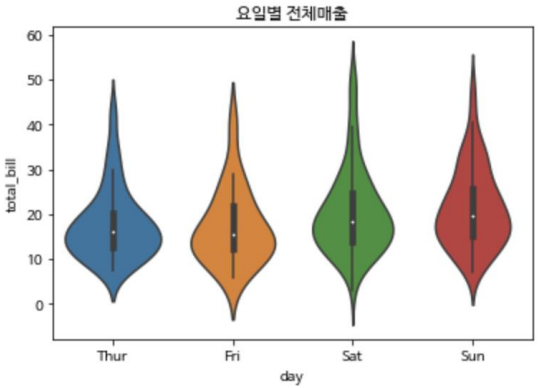
Seaborn

violinplot: 세로 방향으로 커널 밀도 히스토그램을 그림

Violinplot은 Boxplot과 유사하게 수치형 변수의 분포를 시각화하는 그래프입니다.

Boxplot과 달리 분포의 밀도를 곡선 형태로 나타내어 데이터의 분포를 더 자세하게 파악할 수 있습니다.

```
1 plt.title('요일별 전체매출')
2 sns.violinplot(x='day',y='total_bill',data=tips)
3 plt.show()
```



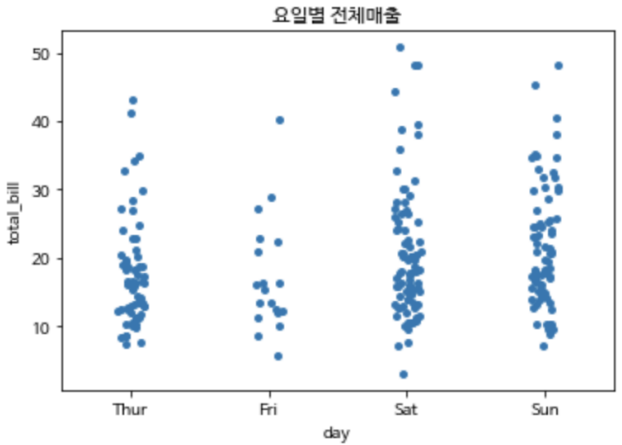
Seaborn

stripplot: 범주형 변수에 들어 있는 각 범주별 데이터의 분포 확인

Stripplot은 범주형 변수와 수치형 변수 사이의 관계를 시각화하는 데 사용되는 그래프입니다.

각 카테고리의 값들을 점으로 나타내어 각 카테고리별 데이터의 분포를 시각적으로 보여줍니다.

```
1 plt.title('요일별 전체매출')
2 sns.stripplot(x='day',y='total_bill',
3                 data=tips,jitter=True)
4 plt.show()
```



Seaborn

swarmplot: stripplot과 비슷하지만 데이터를 나타내는 점이 겹치지 않도록 옆으로 이동

Swarmplot은 범주형 변수와 수치형 변수 사이의 관계를 시각화하는 데 사용되는 그래프입니다.

각 카테고리의 값들을 점으로 나타내되, 각 카테고리별 데이터의 겹침을 최소화하여 데이터의 분포를 더 자세하게 보여줍니다.

```
1 plt.title('요일별 전체매출')
2 sns.swarmplot(x='day',y='total_bill',data=tips)
3 plt.show()
```