2장 통계 분석과 시각화

1. 데이터 시각화
   1) 데이터 시각화 개요
   2) Mapplotlib 차트 그리기
   3) Seaborn 차트 그리기
2. 기술 통계 분석과 시각화
3. 상관 분석 + 히트맵 시각화

1 데이터 시각화 : 1) 데이터 시각화 개요

데이터의 중요성

- 하루에 생산되는 데이터 양
  - 약 5억 개의 트윗이 전송(전세계 인구 중 6%가 하루에 1번씩 트윗 전송)
  - 약 2940억 개의 이메일이 발송된(전세계 인구 1명당 하루에 37개의 메일을 발송)
  - 약 4페타 바이트(Petabyte,PB) 데이터가 페이스북에 생성
  - 약 50억 건의 검색이 검색엔진을 통해 이루어짐
  - 약 100억 건의 메시지가 카카오톡에서 송수신

데이터 활용의 이점

- 성공적인 의사결정을 하는데 도움
- 의사소통 과정에 강력한 무기
- 눈이 보이지 않던 사실 발견

데이터 기억용량 단위		
바이트	1Byte	8bit
킬로바이트	1KiloByte	1024Byte
메가바이트	1MegaByte	1024KB
기가바이트	1GigaByte	1024MB
테라바이트	1TeraByte	1024GB
페타파이트	1PetaByte	1024TB
엑사바이트	1ExaByte	1024PB
제타바이트	1ZetaByte	1024EB
요타바이트	1YottaByte	1024ZB

1 데이터 시각화 : 1) 데이터 시각화 개요

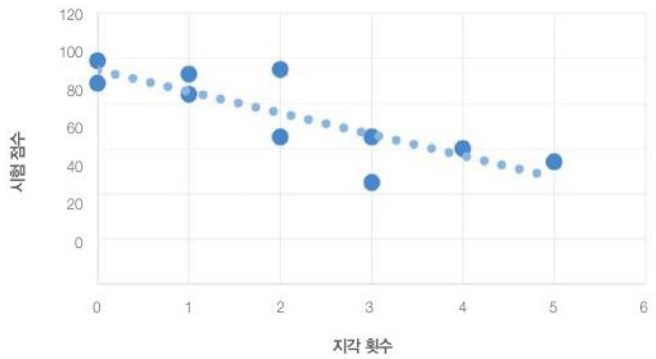
- **데이터 시각화의 정의와 목적**
  - 긴 글보다 짧게 요약된 내용 선호
  - 글자보다 임팩트 있는 시각 요소에 집중
- **데이터 시각화 이유**
  - 항목간의 관계 발견
  - 데이터에 숨어 있는 트렌드 발견
  - 핵심을 효과적으로 전달하는 데이터 스토리 텔링 가능
    - 스토리텔링 : Story + Telling : 의견을 이야기에 담아 전달하는 기법
    - 데이터 스토리텔링 : 데이터에서 발견한 중요한 메시지에 이야기를 담아 듣는 사람에게 메시지를 깊게 심어주는 기법
  - 감정을 자극해 깊은 인상을 남김
    - 메시지에 감정을 표현 : 기쁨, 행복(좋은 감정), 두려움, 슬픔(나쁜 감점) 등을 시각화

1 데이터 시각화 : 1) 데이터 시각화 개요

□ 데이터 시각화 예

| 학생별 지각 횟수와 시험 점수

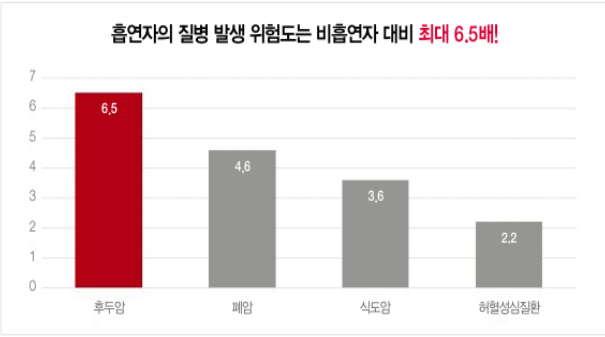
이름	지각 횟수	시험 점수
황나길	0	99
노우람	1	93
임한결	3	65
주한길	5	54
전민음	4	60
배으뜸	2	95
류다운	1	84
노별빛	3	45
박하늘	2	65
이은샘	0	89



항목간의 관계 발견

| 비흡연자 대비 흡연자의 각종 질병 발생 위험도

질병	비흡연자 대비 질병 발생 위험도(단위: 배)
후두암	6.5
폐암	4.6
식도암	3.6
허혈성심질환	2.2

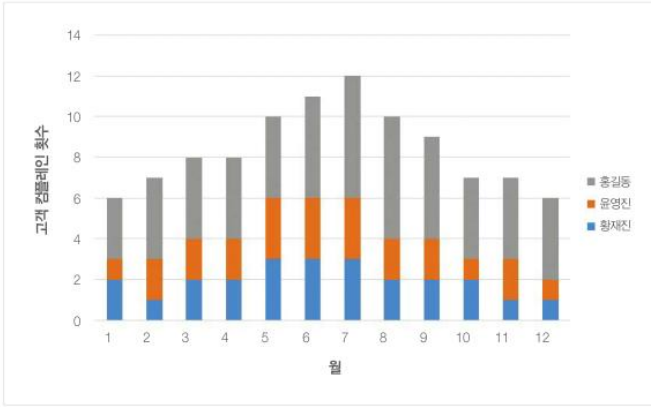


| 흡연자의 질병 발생 위험도를 보여 주는 막대 차트

데이터 스토리텔링 예

| 월별-사원별 컴플레인 현황

월	황재진	윤영진	홍길동
1	2	1	3
2	1	2	4
3	2	2	4
4	2	2	4
5	3	3	4
6	3	3	5
7	3	3	6
8	2	2	6
9	2	2	5
10	2	1	4
11	1	2	4
12	1	1	4



| 사원별 컴플레인 현황을 색으로 구분한 누적 막대 차트

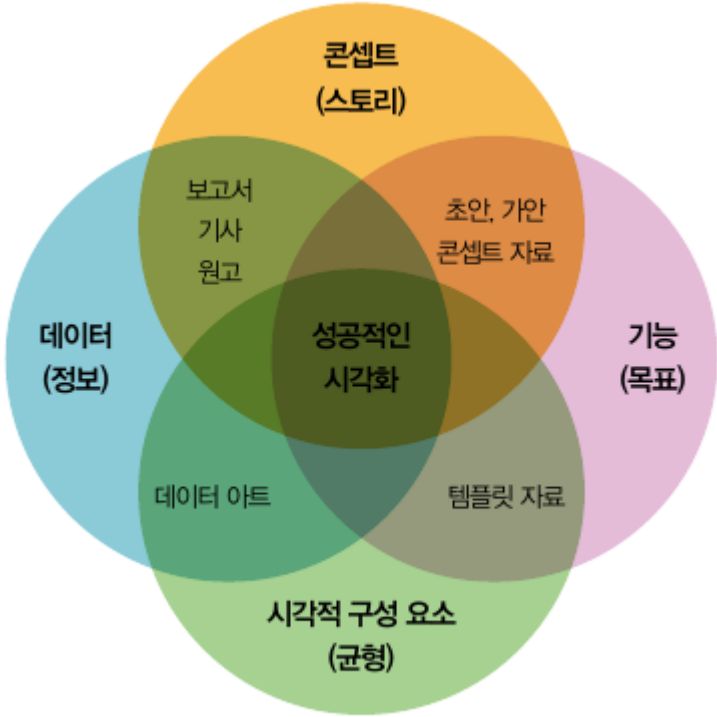
데이터의 트렌드 발견

| 여의도 한강공원의 유동인구 증가율을 보여 주는 선 차트

감정 자극 예

1 데이터 시각화 : 1) 데이터 시각화 개요

- **좋은 시각화의 정의**
  - 데이터를 선택하고 콘셉트를 담아 중요한 데이터를 시각적 구성요소로 강조해 읽기 쉽게 기능적으로 시각화한 것
  - 데이터 정보 : 중요 데이터 강조
  - 기능(목표) : 읽기 쉽게
  - 콘셉트(스토리) : 효과적으로 메시지 전달
  - 시각적 구성 요소 : 색상과 기호를 적절하게 사용



| 성공적인 데이터 시각화에 필요한 4가지 원칙

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- **python의 기본 데이터분석 라이브러리**
  - 넘파이(NumPy)
    - Python 데이터분석의 기본적인 기능들을 제공
    - 벡터 및 행렬 연산과 관련된 편리한 기능들 제공
  - 판다스(Pandas)
    - Series, DataFrame 등의 자료 구조를 활용하여 데이터 분석에 우수한 성능 발휘
    - 대량의 데이터를 빠른 속도로 처리 가능
  - 맷플롯립(Matplotlib)
    - 데이터 분석 결과에 대한 시각화를 빠르고 직관적으로 수행

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- **차트 유형별 데이터 시각화 기법**
  - 막대 차트 : 비교 분석 중심의 시각화
  - 선 차트 : 시간에 다른 변화 중심 시각화
  - 파이 차트 : 비율 분석 중심의 시각화
  - 분산형 차트 : 관계분석 중심의 시각화

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

Numpy 배열

```
1  #넘파이 모듈 import
2  import numpy as np
```

```
1  #리스트
2  data1=[1,2,3,4,5]
3  #리스트를 넘파일 배열로 변환
4  arr1=np.array(data1)
5  print(arr1)
6  print(type(arr1))
7  print(arr1.shape) #배열의 차원
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
(5,)
```

1행 5열

x=np.array([1,2,3,4])(O)
x=np.array(1,2,3,4) (X)

###
data1.mean() # 에러발생
arr1.mean() #가능

```
1  data2=[[1,2,3],[4,5,6]]
2  arr2=np.array(data2)
3  print(arr2)
4  print(arr2.shape) # (2,3) 2god 3duf 2ckdhjs godduf
```

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

동일
data2=[1,2,3,4,5,6]
arr2=np.array(data2).reshape(2,3)

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

Pandas의 Series

```
# pandas 사용하기
import numpy as np
import import pandas as pd
```

```
# Series 정의하기
x = pd.Series([4, 3, 5, 8])
print(x)
print(type(x))

# Series의 값만 확인하기
print(x.values)

# Series의 인덱스만 확인하기
print(x.index)

# Series의 자료형 확인하기
print(x.dtype)
```

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

Pandas의 Dataframe(2차원 배열)

```
# Data Frame 정의하기
# 이전에 DataFrame에 들어갈 데이터를 정의해주어야 하는데,
# 이는 python의 dictionary 또는 numpy의 array로 정의할 수 있다.
data = {'names': ['민준', '현우', '서연', '동현', '지현'],
        'years': [2013, 2014, 2015, 2016, 2015],
        'points': [1.5, 1.7, 3.6, 2.4, 2.9]}
df = pd.DataFrame(data)
df

## 행과 열의 구조를 가진 데이터가 생긴다.
```

	names	years	points
0	민준	2013	1.5
1	현우	2014	1.7
2	서연	2015	3.6
3	동현	2016	2.4
4	지현	2015	2.9

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- **matplotlib**

```
import numpy as np
import pandas as pd
import matplotlib
```

```
#50개의 난수 생성
data=np.random.rand(50)
print(type(data))
print(data)
```

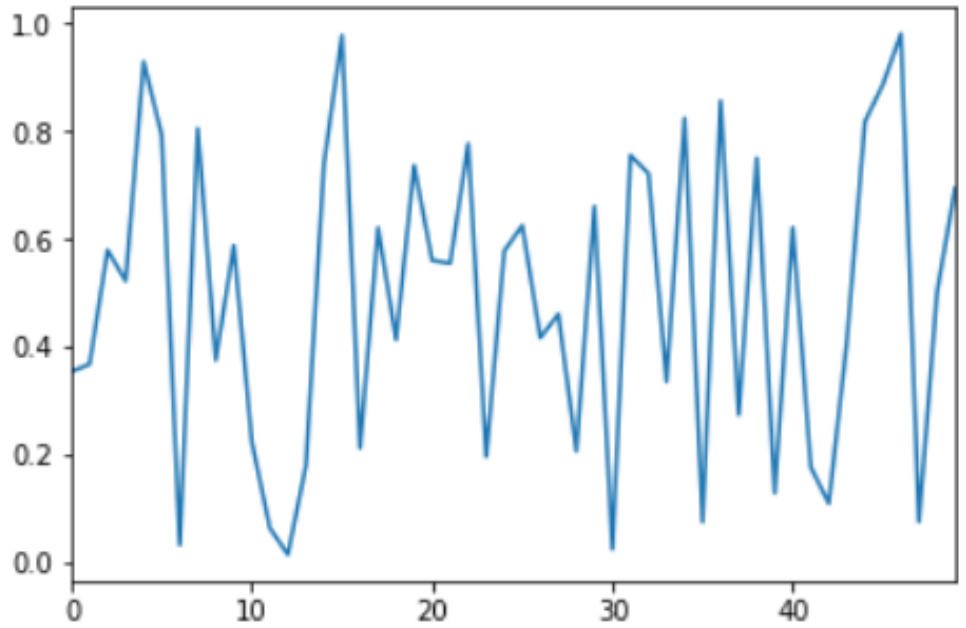
```
# 넘파이 배열을 판다스의 시리즈 자료형으로 변환(인덱스,데이터의 조합)
seri=pd.Series(data)
print(type(seri))
print(seri)
```

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- **matplotlib**

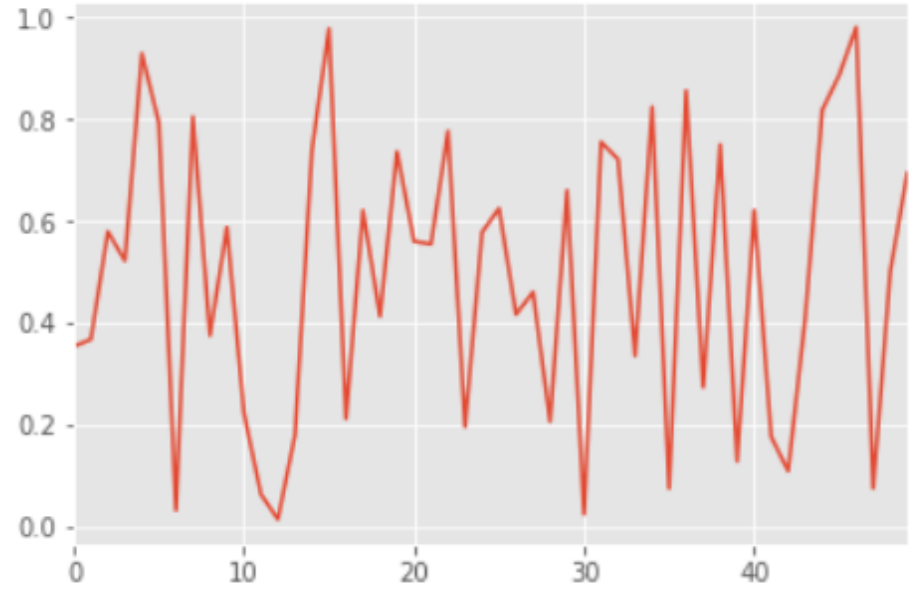
```
1  # x축 인덱스 값, y축 랜덤 값
2  seri.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10d242390>
```




```
1  #그래프 스타일 변경
2  matplotlib.style.use("ggplot")
3
4  # x축 인덱스 값, y축 랜덤 값
5  seri.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10cec6160>
```



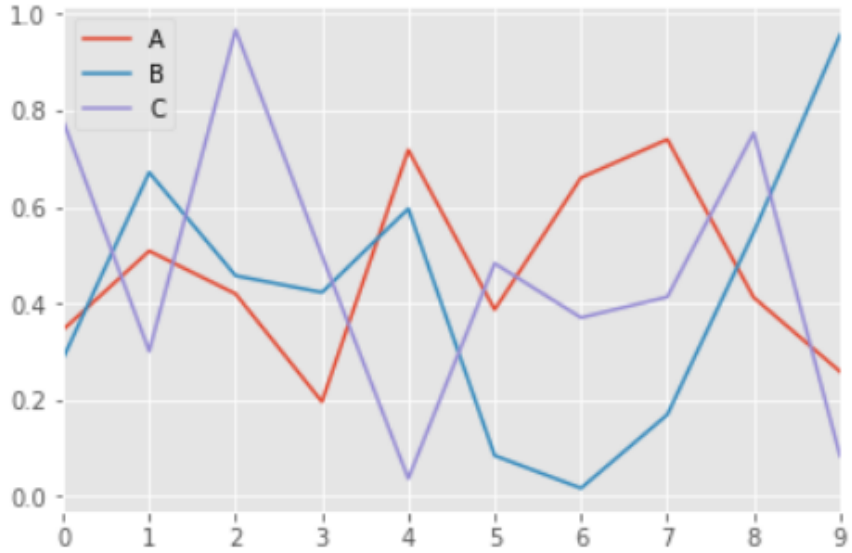
1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- **실습 예제**

```
#넘파이 행렬을 10행 3열로 만든 후 판다스의 데이터 프레임으로 변환
data_set=np.random.rand(10,3)
df=pd.DataFrame(data_set,columns=['A','B','C'])
df
```

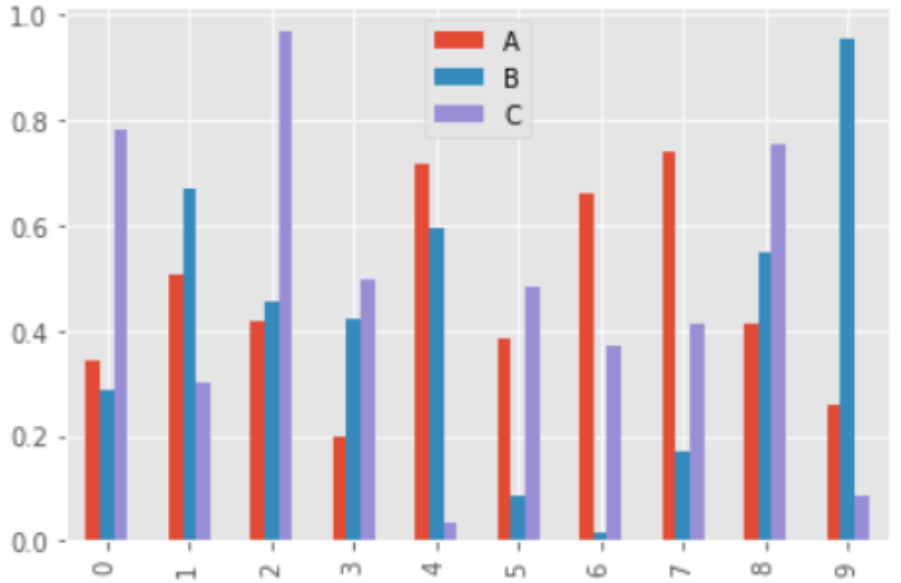
```
1  #데이터 프레임 자료르 그래프로 출력
2  df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10d92bcf8>
```




```
1  df.plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10d9d6710>
```

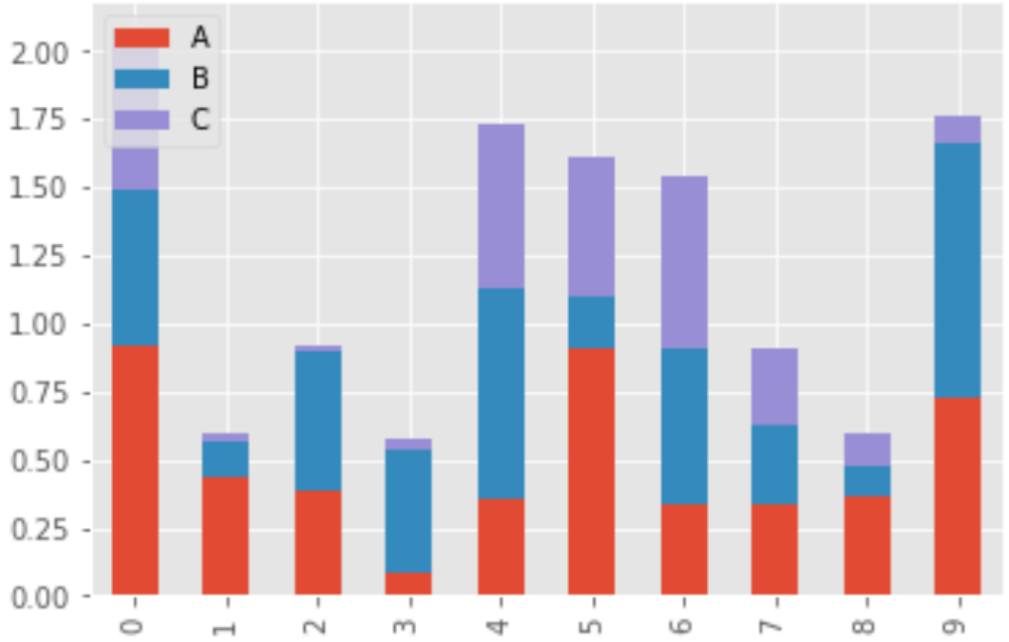


1. Matplotlib 차트 그리기

실습 예제

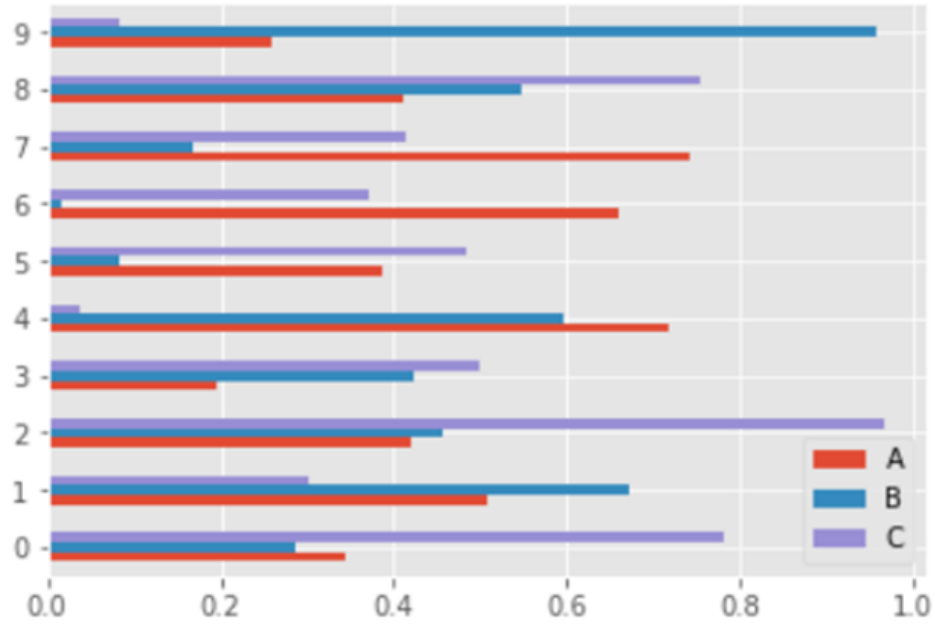
```
df.plot(kind='bar',stacked=True)
```

```
<AxesSubplot:>
```




```
1 df.plot(kind="barh")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10da6a198>
```

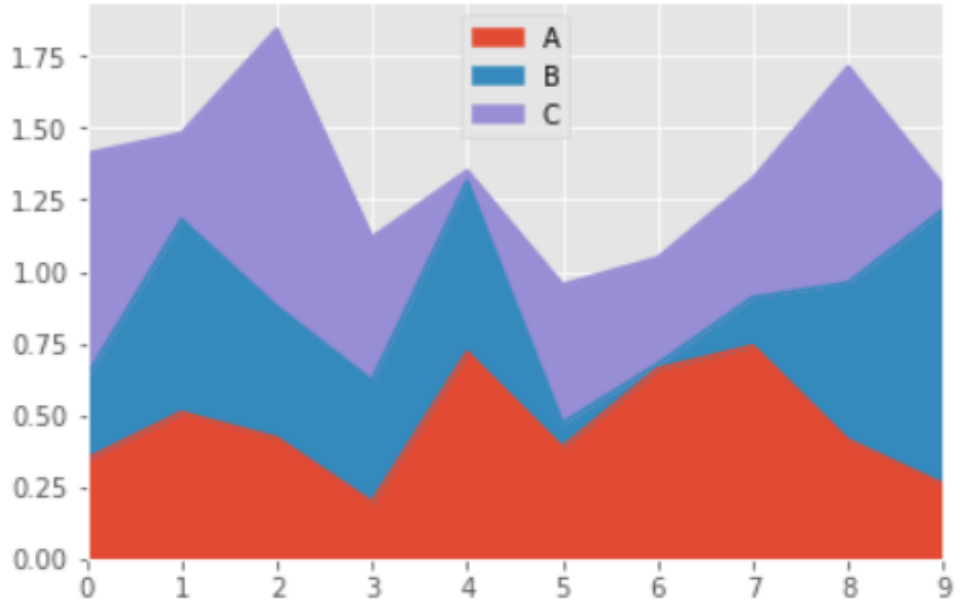


1. 데이터 시각화 : 2) Matplotlib 차트 그리기

1. Matplotlib 차트 그리기

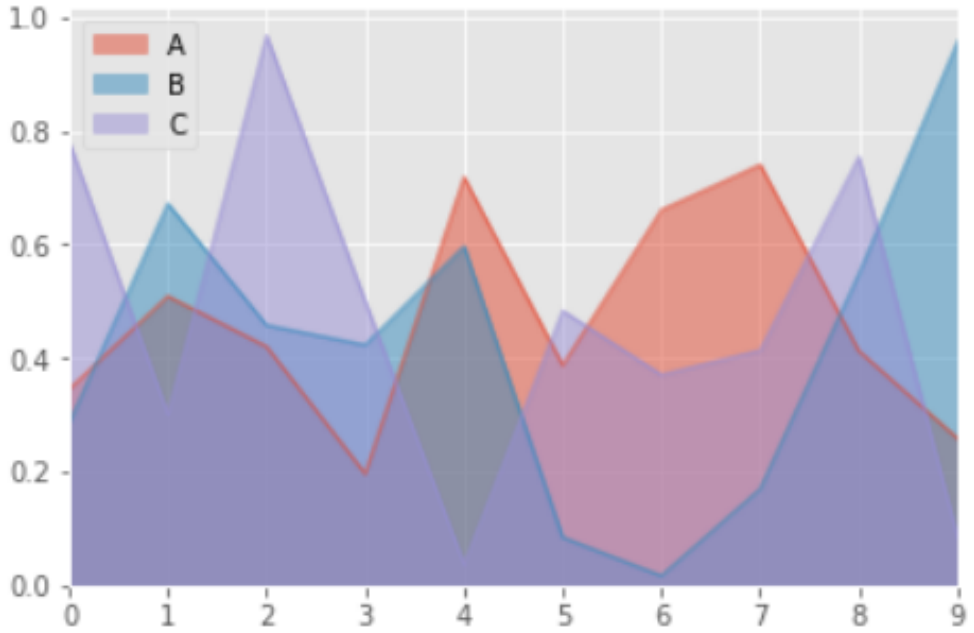
```
1 df.plot(kind="area")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10db1a908>
```




```
1 df.plot(kind="area",stacked=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10dbc92b0>
```

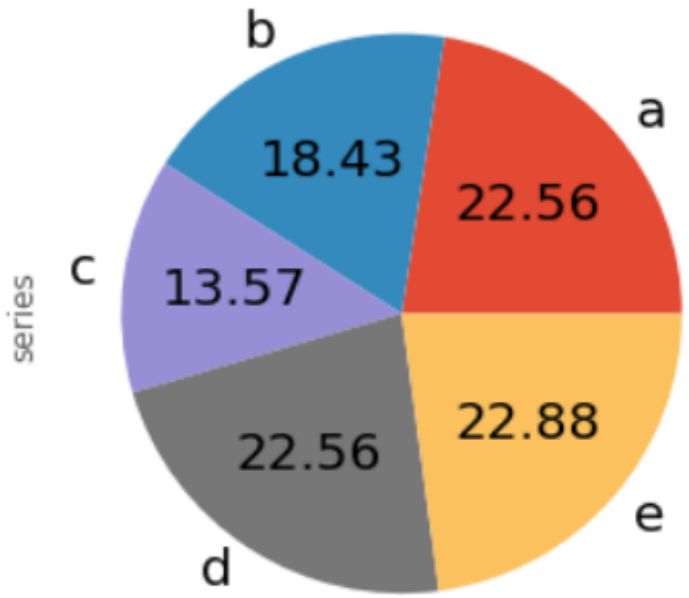


1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- 실습

```
1  #넘파이의 런덤값 5개->판다스의 시리즈 자료형 변환
2  seri2=pd.Series(np.random.rand(5),index=["a","b","c","d","e"],name="series")
3  seri2.plot(kind="pie",autopct="%.2f",fontsize=20,figsize=(8,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10ecf2b00>
```



1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- **그래프에 한글 및 (-)기호 표시**

```
from matplotlib import rc, font_manager
# 한글 처리를 위해 폰트 설정
font_name=font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font',family=font_name)
df2=pd.DataFrame(np.random.rand(6,4), index=['one','two','three','four','five','six'],
                 columns=pd.Index(['A','B','C','D'],name="종류"))
```

```
plt.rcParams['axes.unicode_minus']=False
df2.plot(kind='barh',stacked=True)
```

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- **히스토그램**

```
# 히스토그램 : x변수가 가질 수 있는 값의 구간, 개수를 막대 형태로 출력
# normal() 정규분포
s3=pd.Series(np.random.normal(0,100,size=200))
s3.head()
```

```
# 음수 부호가 깨지지 않도록 설정
plt.rcParams['axes.unicode_minus']=False
# 축 구간은 기본값으로 10개로 설정 됨
s3.hist()
# 축 구간은 기본값으로 50개로 설정 됨
s3.hist(bins=50)
# normed 정규분포
s3.hist(bins=100, normed=True)
```

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

- **scatter**

```
# 산점도 : 서로 다른 독립변수 x1, x2의 관계를 파악할 때 사용
# normal(mu, sigma, size=(rows, cols)) , #mu :중앙값(median),sigma : 표준편차
x1=np.random.normal(1,1,size=(100,1))
x2=np.random.normal(-2,4,size=(100,1))
# concatenate : 열방향 연결, axis=1 가로 방향 ,axis=0 세로방향
x=np.concatenate((x1,x2),axis=1)
print(x[:5]) # 첫 5행 출력
```

```
df3=pd.DataFrame(x,columns=["x1","x2"])
df3.head()
```

```
# x1과 x2의 상관관계
# x1(x축), x2(y축)
# 양의 상관관계 : 산점도가 좌측하단에서 우측 상단으로
# 음의 상관관계 : 산점도가 좌측상단에서 우측하단으로

plt.scatter(df3["x1"],df3["x2"]) # 이 그래프에서 상관관계를 찾을 수 없음
```

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

```
import csv
from datetime import datetime
from matplotlib import pyplot as plt
filename='d:/data/temp/temperature_2014.csv'
with open(filename) as f:
    reader=csv.reader(f)
    header_row=next(reader)
    dates, highs,lows=[],[],[]
    for row in reader:
        try:
            # strftime : 날짜를 문자열로, strptime : 문자열을 날짜로
            current_date=datetime.strptime(row[0],"%Y-%m-%d")
            high=int(row[1]) # 최고 온도
            low=int(row[3]) # 최저 온도
        except ValueError:
            print(current_date,'missing date')
        else: # 예외가 발생하지 않으면 리스트 추가
            dates.append(current_date)
            highs.append(high)
            lows.append(low)
```

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

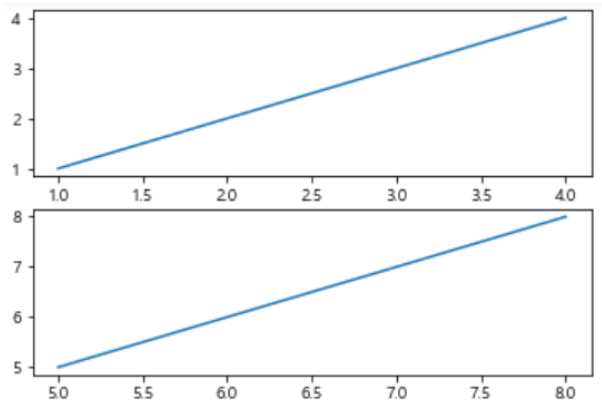
```
# 그래프 생성
fig=plt.figure(dpi=128,figsize=(10,6))
#  x축 dates, y축 highs, 색상 red
# 투명도 0.5, 범위 :0.0~1.0
plt.plot(dates,highs, c='red',alpha=0.5)
plt.plot(dates,lows, c='blue',alpha=0.5)
# 두개의 선 사이(highs~lows)를 색칠함, alpha=0.1 투명도 0.1
plt.fill_between(dates,highs,lows,facecolor='blue',alpha=0.1)
# 그래프 스타일
plt.title('온도 통계',fontsize=20)
plt.xlabel("기간",fontsize=16)
plt.ylabel("온도",fontsize=16)
plt.tick_params(axis='both',labelsize=16) # 눈금 표시
plt.show()  # 이미지 보여주기

# 그래프를 이미지로 저장,dpi 해성도, tight 그림을 둘러산 여백 제거
plt.savefig('data/images/temperature_plot.png',dpi=400,bbox_inches='tight')
```

1. 데이터 시각화 : 2) Matplotlib 차트 그리기

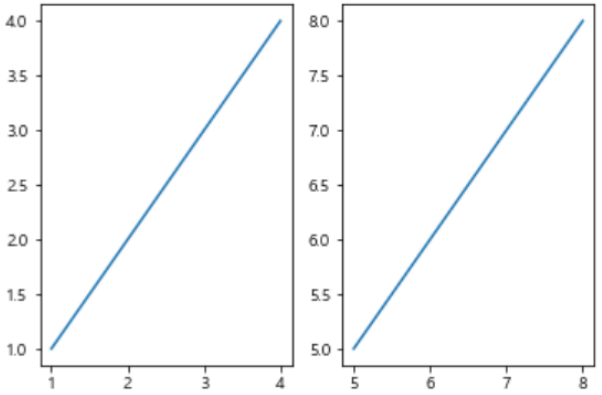
- **한 화면에 여러 개 그래프 그리기**

```
plt.figure()
plt.subplot(2,1,1)
plt.plot([1,2,3,4], [1,2,3,4])
plt.subplot(2,1,2))
plt.plot([5,6,7,8],[5,6,7,8])
plt.show()
```



figure()
- 하나의 캔버스 생성
subplot(m,n,idx)
- 캔버스에 여러 개 그림 넣기
m : 행수
n : 열수
idx : 위치

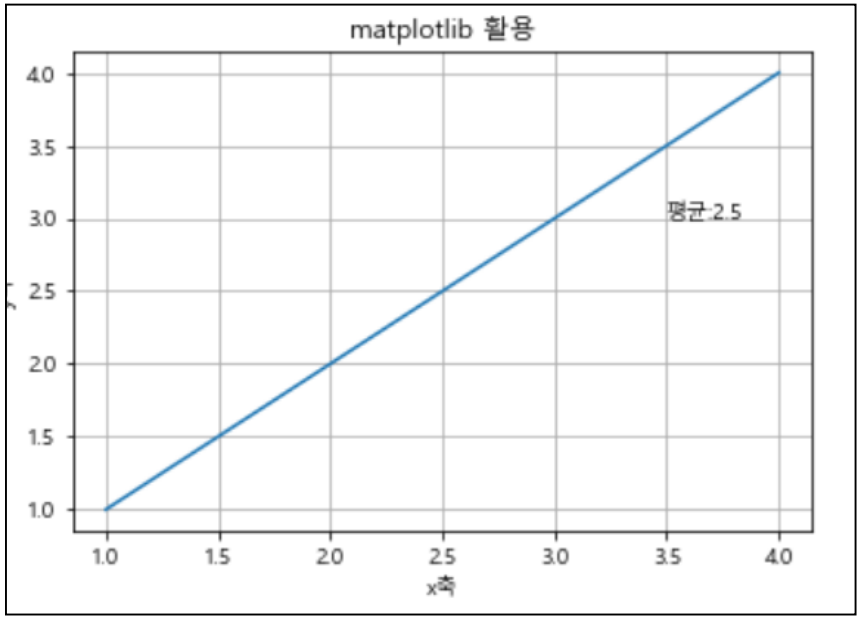
```
plt.figure()
plt.subplot(1, 2, 1)
plt.plot([1,2,3,4], [1,2,3,4])
plt.subplot(1,2,2)
plt.plot([5,6,7,8],[5,6,7,8])
plt.show()
```



1. 데이터 시각화 : 2) Matplotlib 차트 그리기

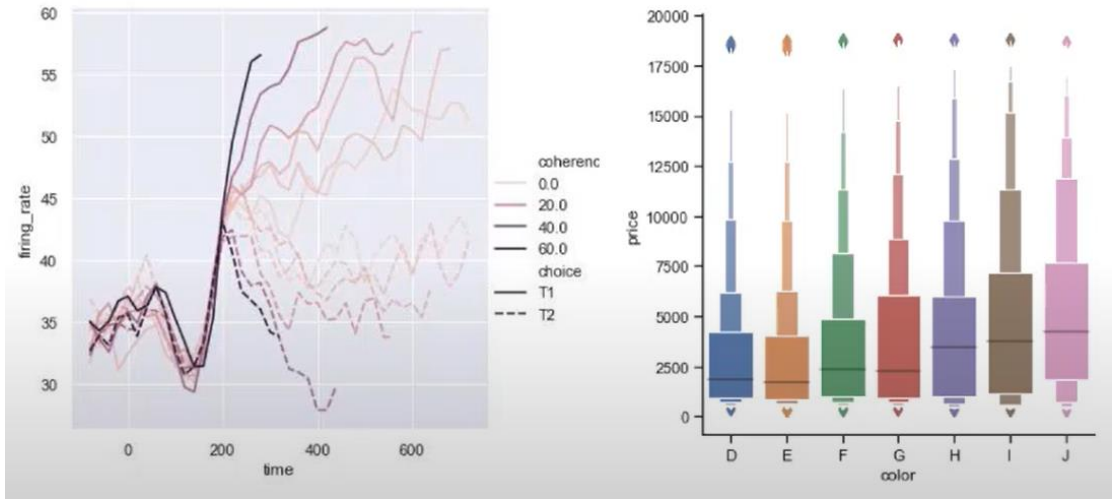
- 그래프에 문자 삽입

```
plt.plot([1,2,3,4], [1,2,3,4])
plt.xlabel('x축')
plt.ylabel('y축')
plt.title('matplotlib 활용')  # 그래프 제목
plt.text(3.5, 3.0, '평균:2.5') #그래프의 지정위치에 문자열 표시
plt.grid(True) # 인덱스마다 격자를 나타냄
plt.show()
```



1. 데이터 시각화 : 3) Seaborn

- **Seaborn**
  - 참조 사이트 : https://seaborn.pydata.org/
  - matplotlib을 기반으로 하는 Python 데이터 시각화 라이브러리
  - 매력적이고 유익한 통계 그래픽을 그리기 위한 고급 인터페이스를 제공
  - 설치 : conda install seaborn (관리자 모드에서 설치)
  - 표현 가능 그래프 : 관계형, 카테고리, 분포, 회귀분석, 멀티-플롯, 스타일, 색상 등



1. 데이터 시각화 : 3) Seaborn

- **Seaborn 장점**
  - seaborn에서만 제공되는 통계 기반 plot
  - 특별하게 꾸미지 않아도 깔끔하게 구현되는 기본 color 제공
  - palette 기능을 활용하여 더 아름답게 그래프 구현이 가능
  - pandas 데이터프레임과 높은 호환성
  - hue 옵션으로 bar 구분이 가능하며, xtick, ytick, xlabel, ylabel, legend 등이 추가적인 코딩 작업없이 자동으로 세팅.

1. 데이터 시각화 : 3) Seaborn

- **Count plot**
  - 항목별 갯수를 세어주 줌, 해당 column을 구성하고 있는 value들을 구분하여 보여 줌.
- **Relplot**
  - 두 column 사이의 상관관계를 보여줌
- **histplot**
  - matplotlib의 hist 그래프와 kdeplot을 통합한 그래프 이며, 분포와 밀도를 확인할 수 있다.
    - kde : 분포 라인, rugplot : 데이터 정밀 분포 위치 표시
- **Heat map**
  - 색상으로 표현할 수 있는 다양한 정보를 일정한 이미지위에 열분포 형태의 비쥬얼한 그래픽으로 출력
  - 다중 상관관계 표시할 때 유용

- **Cat plot : 범주를 나누어서 그려줌**
  - 참고사이트 : https://seaborn.pydata.org/generated/seaborn.catplot.html
  - 종류 : swarm, violin, box, boxen, Point, bar 등
- **Pair plot**
  - 그리드(grid) 형태로 각 집합의 조합에 대해 히스토그램과 분포도를 표시, 숫자형 column에 대해서만 그려준다.
- **lm plot**
  - column 간의 선형관계를 확인하기에 용이한 차트
- **Joint plot**
  - scatter(산점도)와 histogram(분포)을 동시에 그려주며 숫자형 데이터만 표현 가능

1. 데이터 시각화 : 3) Seaborn

- **seaborn 차트 실습**

```
# 라이브러리 로드
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style='whitegrid') # 스타일 지정
```

```
# 차트에 사용할 한글 설정
from matplotlib import rc, font_manager
# 한글 처리를 위해 폰트 설정
font_name=font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font',family=font_name)

# 차트 축에 마이너스 기호(-) 깨짐 방지
plt.rcParams['axes.unicode_minus']=False

#차트 그림 크기설정
plt.rcParams['figure.figsize']=(12,9)
```

1. 데이터 시각화 : 3) Seaborn

seaborn 차트 실습

```
# seaborn의 데이터 셋 확인
sns.get_dataset_names()

# titanic 데이터 로드
titanic=sns.load_dataset('titanic')
print(titanic.shape)
titanic.head()

# tips 데이터 셋 load
tips=sns.load_dataset('tips')
print(tips.shape)
tips.head()
```

 # titanic field
- survived: 생존여부
- pclass: 좌석등급
- sex: 성별
- age: 나이
- sibsp: 형제자매 + 배우자 숫자
- parch: 부모자식 숫자
- fare: 요금
- embarked: 탑승 항구
- class: 좌석등급 (영문)
- who: 사람 구분
- deck: 데크
- embark_town: 탑승 항구 (영문)
- alive: 생존여부 (영문)
- alone: 혼자인지 여부

 # tips 데이터 셋 필드
- total_bill: 총 합계 요금표
- tip: 팁
- sex: 성별
- smoker: 흡연자 여부
- day: 요일
- time: 식사 시간
- size: 식사 인원

1. 데이터 시각화 : 3) Seaborn

seaborn 차트 실습

countplot()

- 항목의 갯수를 세어줌, 알아서 column을 구성하고 있는 value들을 구분하여 보여줌

```
sns.countplot(x="class", data=titanic)
plt.show()

# hue로 카테고리 분류
sns.countplot(x="class", hue='who', data=titanic)
plt.show()

# 가로 bar countplot
sns.countplot(y="class", hue='who', data=titanic)
plt.show()

# palette로 차트 색상 변경
sns.countplot(x="class", hue='who', palette='YlGn', data=titanic)
plt.show()
```

1. 데이터 시각화 : 3) Seaborn

- **seaborn color palette 사용**

```
#참조사이트 : https://www.geeksforgeeks.org/seaborn-color-palette/
current_palette = sns.color_palette()
#sns.palplot(current_palette)

#sns.palplot(sns.color_palette("Greys"))
sns.palplot(sns.color_palette("terrain_r", 7))
#sns.palplot(sns.color_palette("deep", 10))
#sns.palplot(sns.color_palette("muted", 10))
#sns.palplot(sns.color_palette("bright", 10))
#sns.palplot(sns.color_palette("dark", 10))
color = ["Blue", "White", "Red", "Yellow", "Green", "Grey"]
sns.set_palette('deep')
#sns.palplot(sns.color_palette())
```

1. 데이터 시각화 : 3) Seaborn

- **relplot() : 두 column간의 상관관계를 보여줌**

```
sns.relplot(x='x축으로 사용할 필드',  y='y축으로 사용할 필드', data=dataFrame,
                hue='범주형 데이터 다른 색으로 구분하여 표시',
                size='수치형 데이터 값에 따라 마크의 크기 설정',
                style='마크의 모양 구분',
                 row='행으로 나타낼 필드', cols='열로 나타낼 필드')
print(tips.total_bill.corr(tips.tip))
sns.relplot(x='total_bill', y='tip', data=tips)
sns.scatterplot(x='total_bill', y='tip', data=tips)
sns.scatterplot(x='total_bill', y='tip', data=tips, hue='smoker')
sns.relplot(x="total_bill", y="tip", hue="day", data=tips)
sns.relplot(x='total_bill', y='tip', data=tips, hue='size')
sns.relplot(x='total_bill', y='tip', data=tips, style='smoker', hue='size')
sns.relplot(x='total_bill', y='tip', data=tips, palette='husl', hue='size')
sns.relplot(x='total_bill', y='tip', data=tips, palette='terrain_r',  size='size' , hue='size')
sns.relplot(x='total_bill', y='tip', hue='smoker', col='day', row='sex', data=tips)
sns.relplot(x='total_bill', y='tip', hue='smoker', col='day', col_wrap=2, data=tips)
plt.show()
```

1. 데이터 시각화 : 3) Seaborn

- **relplot()**

```
# line 차트 그리기
df=pd.DataFrame(dict(time=np.arange(500), value=np.random.randn(500).cumsum()))
sns.relplot(x='time',y='value',data=df, kind='line')
sns.lineplot(x='time', y='value', data=df)

fmri=sns.load_dataset('fmri')

sns.relplot(x=' timepoint ' ,y=' signal ' , data=르갸, kind=' line ' )
sns.relplot(x=' timepoint ' ,y=' signal ' , data=르갸, kind=' line', ci=90)
sns.relplot(x=' timepoint ' ,y=' signal ' , data=르갸, kind=' line', ci=None)
sns.lineplot(x='time', y='value', data=df, ci=None)
sns.lineplot(x='time', y='value', data=df, ci='sd')
sns.lineplot(x='time', y='value', data=df, estimator=None)
plt.show()

#ci: 신뢰구간, estimator : 추정회귀선
```

1. 데이터 시각화 : 3) Seaborn

- sns.histplot(): 히스토그램, 라인그래프, kde 그래프

```
sns.histplot(data=x, kde=True)
sns.rugplot(x=x)
plt.show()
```

- heatmap
  - 데이터 값을 색상으로 강조하여 표현

```
data=np.random.rand(5,6)
sns.heatmap(data,annot=True)

pivot=tips.pivot_table(index='day', columns='size', values='tip')
sns.heatmap(pivot, cmap='Reds', annot=True)

titanic_corr=titanic.corr()
sns.heatmap(titanic_corr, annot=True, cmap='GnBu')
```

1. 데이터 시각화 : 3) Seaborn

- **catplot**
  - 범주형 차트 그리기
  - 참사이트 : https://seaborn.pydata.org/generated/seaborn.catplot.html

```
sns.catplot(x='day', y='total_bill', data=tips, hue='sex')
sns.catplot(x='day', y='total_bill', data=tips, kind='swarm')
sns.catplot(x='day', y='total_bill', data=tips, kind='swarm', hue='sex')
sns.catplot(x='day', y='total_bill', data=tips, kind='violin')
sns.catplot(x='day', y='total_bill', data=tips, kind='violin', hue='sex')
sns.catplot(x='day', y='total_bill', data=tips, kind='violin', hue='sex', inner='stick', split=True)

plt.show()
```

1. 데이터 시각화 : 3) Seaborn

- **Boxplot**
  - 최댓값, 최소값, 분위수 표기함

```
sns.catplot(x='day', y='total_bill', data=tips, kind='box')
sns.catplot(x='day', y='total_bill', data=tips, kind='boxen')
```

- **barplot**

```
sns.catplot(x='day', y='total_bill', data=tips, hue='smoker',kind='bar')
sns.catplot(x='day', data=tips, hue='sex', kind='count')

titanic=sns.load_dataset('titanic')
sns.catplot(x='class', y='survived', data=titanic, col='sex', kind='bar')
```

1. 데이터 시각화 : 3) Seaborn

- **pairplot**
  - 그리드 형태로 각 집합으 조합에 대해 히스토그램과 분포도를 표시
  - 숫자형 column에 대해서만 그려줌

```
sns.pairplot(tips)
plt.show()

sns.pairplot(tips, hue='size')
plt.show()
```

1. 데이터 시각화 : 3) Seaborn

- **lmplot**
  - lmplot: column간의 선형관계를 확인할 수 있다
  - outlier 짐작가능

```
sns.lmplot(x='total_bill',y='tip', hue='smoker',data=tips, col='day', col_wrap=3)
plt.show()
```

- **joinplot**
  - scatter(산점도)와 histogram(분포)을 동시에 그려주며 숫자형 데이터만 표현 가능

```
sns.jointplot(x='total_bill', y='tip', data=tips, kind='reg')
plt.show()
```

3. 기술 통계 분석과 시각화

핵심 개념 이해

기술 통계(요약 통계)

- 데이터의 특성을 나타내는 수치를 이용해 분석하는 기본적인 통계 방법
- 평균, 중앙값, 최빈값 등을 구할 수 있음

회귀 분석

- 독립 변수 x와 종속변수, y간의 상호 연관성 정도를 파악하기 위한 분석기법
- 하나의 변수가 변함에 따라 대응되는 변수가 어떻게 변하는지를 측정하는 것
- 변수 간의 인과관계를 분석 할 때 많이 사용
- 독립 변수가 한 개이면 단순 회귀 분석, 두개이면 다중 회귀분석
- 독립변수와 종속 변수의 관계에 따라 선형 회귀 분석과 비선형 회귀분석으로 나눠짐
- 선형 회귀분석 식 : $y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$

t-검정

- 데이터에서 찾은 평균으로 두 그룹에 차이가 있는지 확인하는 방법
- 예 : A와인의 품질이 1등급인지 2등급인지에 따라 가격에 차이가 있는지를 확인할 때 사용

히스토그램

- 데이터 값의 범위를 몇 개 구간으로 나누고 각 구간에 해당하는 값의 숫자나 상대적인 빈도를 차트로 나타내는 것

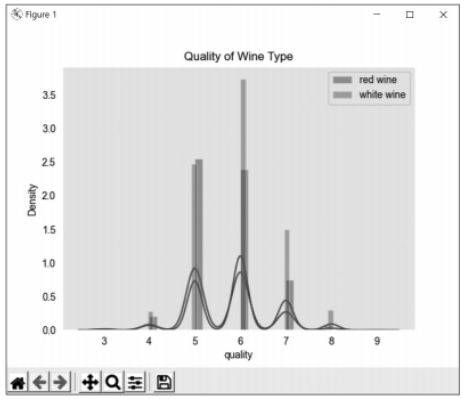
3. 기술 통계 분석과 시각화

와인 품질 예측하기(실습)

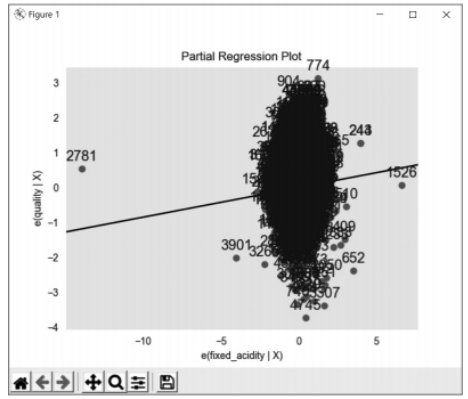
- 목표: 와인의 속성을 분석한 뒤 품질 등급을 예측하는 것
- 데이터의 기술 통계를 구함
- 레드 와인과 화이트 와인 그룹의 품질에 대한 t-검정을 수행
- 와인 속성을 독립 변수로, 품질 등급을 종속 변수로 선형 회귀 분석을 수행

와인 품질 등급 예측하기	
목표	와인 속성을 분석하여 품질 등급을 예측한다.
핵심 개념	기술 통계, 회귀 분석, t-검정, 히스토그램
데이터 수집	레드 와인/화이트 와인 데이터셋: 캘리포니아 어바인 대학의 머신러닝 저장소에서 다운로드
데이터 준비	수집한 데이터 파일 병합
데이터 탐색	1. 정보 확인: info() 2. 기술 통계 확인: describe(), unique(), value_counts()
데이터 모델링	1. 데이터를 두 그룹으로 비교 분석 • 그룹별 기술 통계 분석: describe() • t-검정: scipy 패키지의 ttest_ind() • 회귀 분석: statsmodels.formula.api 패키지의 ols() 2. 품질 등급 예측 • 샘플을 독립 변수(x)로 지정 → 회귀 분석 모델 적용 → 종속 변수(y)인 품질(quality) 예측
결과 시각화	

1. 히스토그램을 이용한 시각화



2. 부분 회귀 플롯을 이용한 시각화



3. 기술 통계 분석과 시각화

- **데이터 수집**
  - 다운로드 경로 : https://archive.ics.uci.edu/dataset/186/wine+quality
  - 다운로드버튼 클릭 -> wine+quality.zip파일 다운로드 후 압축을 풀고 data폴더에 'winequality-red.csv, winequality-white.csv 파일을 복사

```
import pandas as pd
red_df=pd.read_csv('data/winequality-red.csv', sep=';')
white_df=pd.read_csv('data/winequality-white.csv', sep=';')
red_df.to_csv('data/winequality-red1.csv', index=False)
white_df.to_csv('data/winequality-white1.csv', index=False)
red_df.info()
white_df.info()
```

3. 기술 통계 분석과 시각화

□ 데이터 통합

```
#type 필드 추가
red_df.insert(0,column='type', value='red')
white_df.insert(0, column='type',value='white')
print(red_df.head()) ; print(white_df.head())

#dataframe 합치
wine=pd.concat([red_df, white_df]) ; wine.info()
wine.to_csv('data/wine.csv',index=False)
wine_df=pd.read_csv('data/wine.csv') ; wine.head()

#필드명 변경
wine.columns=wine.columns.str.replace(' ','_')
wine.head()
```

wine 데이터셋 필드 정보
- type : 와인 타입(red, white)
- fixed acidity : 결합산 - 주로 타르타산(tartaric), 사과산(malic)으로 구성, 완인의 산도 제어
- volatile acidity : 휘발산 - 와인의 향과 연관
- citric acid : 구연산 - 와인의 신선함을 올려주는 역할, 산성화에 연관을 미침
- residual sugar : 잔여 설탕 - 화인의 단맛을 나타냄
- chlorides : 염화물 - 화인의 짠맛의 원인이며 와인의 신맛을 좌우하는 성분
- free sulfur dioxide : 이산화 황 활성
- total sulfur dioxide : 이산화 황 총량
- sulphates : 황산염
- ** 황 화합물 : 황 화합물은 원하지 않는 박테리아와 효모를 죽여서 와인을 오래 보관하는 역할
  (free sulfur dioxide, total sulfur dioxide, sulphates)
- density : 밀도 - 바디의 높고 낮음을 표현하는 와인의 무게감
- pH : 산성도 - 와인의 신맛의 정도를 나타냄
- alcohol : 알코올 -와인의 알코올 도수
- quality

3. 기술 통계 분석과 시각화

기술통계(요약통계)

```
print(wine.describe()) # 요약통계 보기
print(sorted(wine['quality'].unique())) # quality 값의 종류 확인
print(wine['quality'].value_counts())  # quality 값별 개수 확인
wine.groupby('type')['quality'].describe() #type으로 그룹하여 quality 요약통계 보기
wine.groupby('type')['quality'].std() # type으로 그룹하여  quality별 표준편차 확인
wine.groupby('type')['quality'].agg(['mean','sum','std'])  # type으로 그룹하여 quality의 평균,합,표준편차 확인
wine.groupby('type').agg({'quality':['sum','mean'], 'alcohol':['std','median']}) #type별 quality의 합과 평균, alcolol의 표준편차와 중앙값 확인
```

1. 와인의 type과 quality로 groupby 하여 alcohol의 평균을 구하라.
2. 와인의 quality로 groupby 하여 alcohol의 평균과 표준편차를 구하라.
3. 와인의 quality로 groupby 하여 alcolol의 [평균, 분산(var)], pH의 [중앙값(median)과 합]을 구하라.
4. 와인의 type과 quality로 groupby하여 alcohol의 [평균과 중앙값]을 구하라.

T검정과 회귀분석

- t 검정을 위해서 scipy라이브러리 패키지 사용, 두 그룹간 차이 비교
- 회귀분석 : statsmodels 라이브러리 패키지 사용

```
! pip install statsmodels
```

```
from scipy import stats # T 검정에 필요
#회귀 분석에 필요한 statsmodels.formula.api 패키지의 ols, glm 함수를 로드
#ols: 최소승자법 OLS: Ordinary Least Squares)는 잔차제곱합(RSS: Residual Sum of Squares)를
# 최소화하는 가중치 벡터를 구하는 방법
from statsmodels.formula.api import ols, glm
```

T검정과 회귀분석

T검정

```
red_wine_quality=wine.loc[wine['type']=='red','quality']
white_wine_quality=wine.loc[wine['type']=='white','quality']
print(red_wine_quality[:10])
print(white_wine_quality[:10])

stats.ttest_ind(red_wine_quality, white_wine_quality,
equal_var=False)
```

T검정과 회귀분석

회귀분석

```
#회귀분석 식 :  y=a1x1+a2x2+....+b

Rformula = 'quality ~ fixed_acidity + volatile_acidity + citric_acid + ₩
      residual_sugar + chlorides + free_sulfur_dioxide + total_sulfur_dioxide + ₩
      density + pH + sulphates + alcohol'

regression_result=ols(Rformula, data=wine).fit()  # 훈련
regression_result.summary()   # 훈련결과 요약

sample1=wine[wine.columns.difference(['quality','type'])] ; sample1[:5]  # 회귀분석식 X값 추축
sample1_predict=regression_result.predict(sample1)  #예측
print(sample1_predict[0:5]) # 예측결과 확인
print(wine['quality'][0:5])  # 정답
```

분석결과 시각화(커널 밀도 추정(Kde)를 적용한 히스토그램 그리기

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('dark') # 히스토그램 차트 배경 스타일 설정
#distplot -> histplot
sns.histplot(data=red_wine_quality, kde=True, color='red', label='red_wine') #red_wine
sns.histplot(data=white_wine_quality, kde=True,color='blue', label='white_wine') #white_wine
plt.title('Quality of wine Type')
plt.legend()
plt.show()
```

부분 회귀 플롯으로 시각화하기

- 독립변수가 2개 이상인 경우 부분회귀플롯을 사용하여 하나의 독립변수가 종속변수에 미치는
- 영향력을 시각화하여 분석할 수 있음
- **plot_partregress(endog, exog_i, exog_others, data=None, obs_labels=True, ret_coords=False)**
  - endog: 종속변수 문자열
  - exog_i: 분석 대상이 되는 독립변수 문자열
  - exog_others: 나머지 독립변수 문자열의 리스트
  - data: 모든 데이터가 있는 데이터프레임
  - obs_labels: 데이터 라벨링 여부
  - ret_coords: 잔차 데이터 반환 여부

부분 회귀 플롯으로 시각화하기

```
import statsmodels.api as sm  # 부분회귀 계산을 위해 로드

# fixed_acidity가 종속변수 quality에 미치는 영향을 시각화하기
# 부분 회귀에 사용한 독립변수와 종속 변수를 제외한 나머지 변수 리스트를 others변수에 저장
others=list(set(wine.columns).difference(set(['quality', 'fixed_acidity'])))
p, resids=sm.graphics.plot_partregress('quality', 'fixed_acidity', others, data=wine, ret_coords=True,
                                       obs_labels=False) ; plt.show()
fig=plt.figure(figsize=(8,13))
sm.graphics.plot_partregress_grid(regression_result, fig=fig) #전체 항목에 대한 부분 회귀 플롯 작성
plt.show()
```

4. 상관관계 분석과 히트맵 시각화

핵심 개념 이해

상관 분석

- 두 변수가 어떤 선형적 관계에 있는지를 분석하는 방법
- 두 변수는 서로 독립적이거나 상관된 관계일 수 있는데, 두 변수의 관계의 강도를 상관관계 라고함
- 상관 분석에서는 상관관계의 정도를 나타내는 단위로 모상관 계수 ρ를 사용
- 상관 계수는 두 변수가 연관된 정도를 나타낼 뿐 인과 관계를 설명하지 않으므로 정확한 예측치를 계산할 수는 없음

단순 상관 분석

- 두 변수가 어느 정도 강한 관계에 있는지 측정

다중 상관 분석

- 세 개 이상의 변수 간 관계의 강도를 측정
- 편상관 분석: 다른 변수와의 관계를 고정하고 두 변수 간 관계의 강도를 나타내는 것

상관관계 실습

```
키=[100,120,130,140,150,160,170,180,190]
발크기=[200,205,210,220,230,250,270,280,285]
plt.scatter(키, 발크기)
plt.xlabel('키(cm)')
plt.ylabel('발크기(mm)')
plt.show()

df=pd.DataFrame({'키':키,'발크기':발크기})
sns.relplot(x='키', y='발크기',data=df)
print(df.키.corr(df.발크기))
plt.show()
```

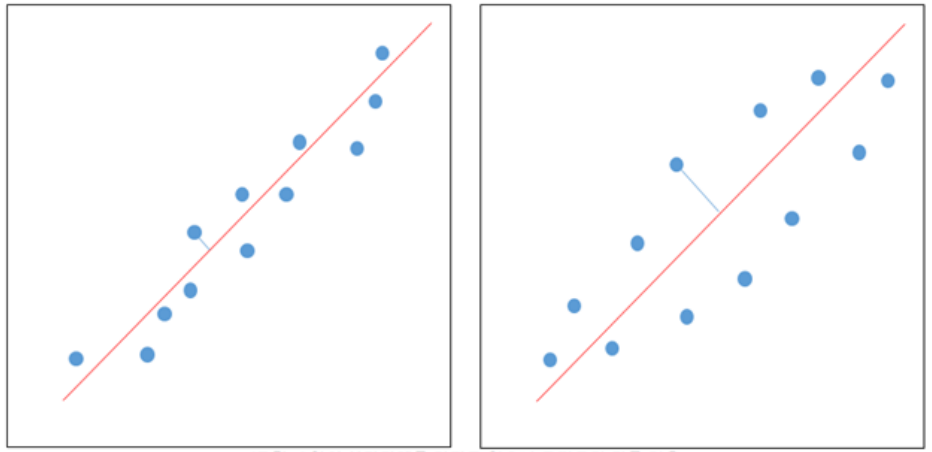
```
h=[100,200,300,400,500,600,700,800,900]
t=[18.0,17.5,17,16.5,15,13.5,13,12,11]
df=pd.DataFrame({'h':h, 't':t})
print(df.h.corr(df.t))
plt.scatter(h,t)
sns.relplot(x=h, y=t, data=df)
```

```
x=np.random.randint(0,100,50)
y=np.random.randint(0,100,50)
df=pd.DataFrame({'x':x, 'y':y})
print(df.x.corr(df.y))
sns.relplot(x=x, y=y, data=df)
```

3. 기술 통계 분석과 시각화

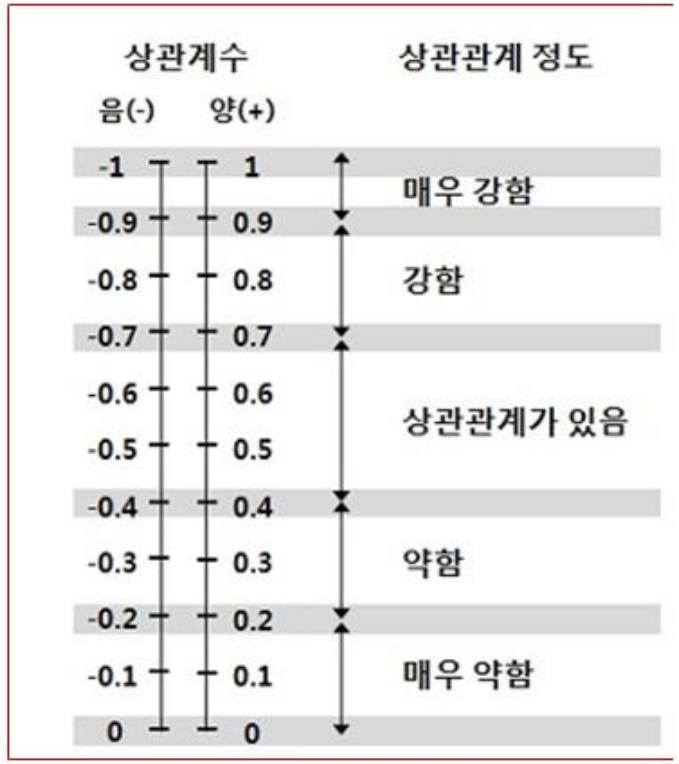
상관관계 분석과 상관 계수

- 상관관계분석: 두 변수사이의 상관정도를 분석하는 것
- 상관계수 : 상관관계 강도를 나타는 것
  - 변수 간 관계의 정도(0~1)와 방향(+, -)을 하나의 수치로 요약해주는 지수로 -1에서 +1 사이의 값을 가짐
  - 상관 계수가 + 이면 양의 상관관계이며 한 변수가 증가하면 다른 변수도 증가
  - 상관 계수가 -이면 음의 상관관계이며 한 변수가 증가할 때 다른 변수는 감소



[그림 4] 양의 상관관계를 가지고 있으나 근접도가 다른 경우

상관계수(기호 : r) - $-1 \le r \le 1$ 의 값을 가짐



상관관계 분석식

- 모집단 전체 상관계수 공식

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

$$= \frac{\mathbf{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

$$= \frac{\mathbf{E}(XY) - \mathbf{E}(X)\mathbf{E}(Y)}{\sigma_X \sigma_Y}$$

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E(X)^2}\sqrt{E(Y^2) - E(Y)^2}}.$$

- 모집단 전체 상관계수 공식

$$r = \frac{\sum_{i=1}^{n}(x-\bar{x})(y-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

```
import math

def correlation(x,y):
    n=len(x)
    x_sum=0.0
    y_sum=0.0
    x_sum_pow=0.0
    y_sum_pow=0.0
    mul_xy_sum=0.0

    for i in range(n):
        mul_xy_sum = mul_xy_sum + float(x[i]) * float(y[i])
        x_sum = x_sum + float(x[i])
        y_sum = y_sum + float(y[i])
        x_sum_pow=x_sum_pow + pow(float(x[i]),2)
        y_sum_pow=y_sum_pow + pow(float(y[i]),2)

        try:
          r=((n*mul_xy_sum)-(x_sum*y_sum)) / math.sqrt(((n*x_sum_pow)-pow(x_sum,2)) * ((n*y_sum_pow)-pow(y_sum,2)))
        except:
          r=0.0
    return r
```

상관관계 실습-2

```
tips=sns.load_dataset('tips')
print(tips.total_bill.corr(tips.tip))
sns.relplot(x='total_bill', y='tip', data=tips)
```

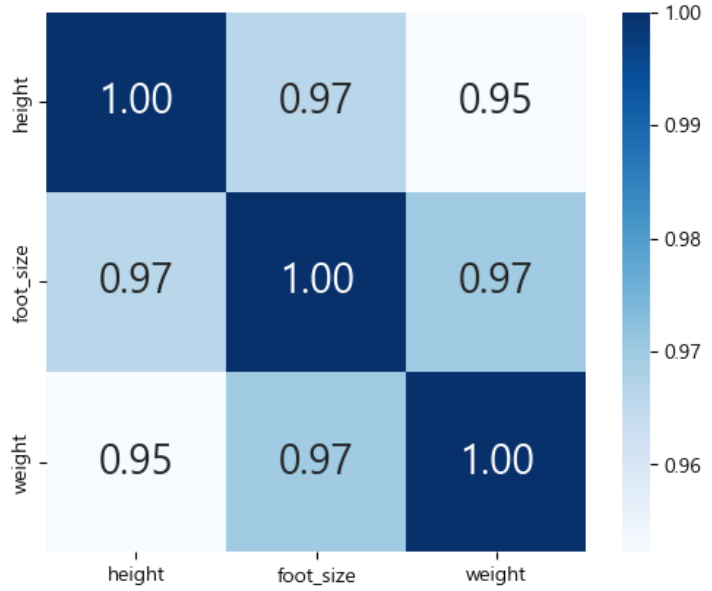
```
print('키와 발크기 상관관계 분석:', correlation(키, 발크기))
print('산의 높이와 기온사이 상관관계 분석:', correlation(h, t))
print('random 발생 값 상관관계 분석:', correlation(x, y))
```

```
height = [100, 120, 130, 140, 150, 160, 170, 180, 190]
foot_size = [200, 205, 210, 220, 230, 250, 270, 280, 285]
weight=[25, 30, 38, 35, 40, 45, 55, 68, 70]
list=[height,foot_size, weight]
column_names=['height','foot_size','weight']
df=pd.DataFrame(list).T
df.columns=column_names
print(df['height'].corr(df['foot_size']))
print(df['height'].corr(df['weight']))
print(df['weight'].corr(df['foot_size']))
corr=df.corr()
print(corr)
```

상관관계 시각화

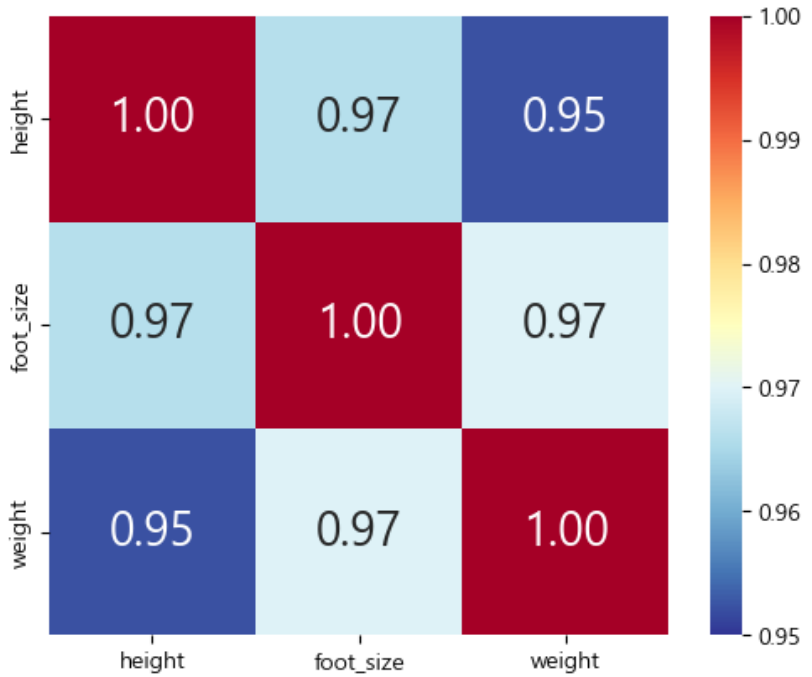
- scatter : 두 변수 사이의 상관관계 시각화
- relplot : 두 변수 사이의 상관관계 시각화
- heatmap : 데이터프레임 필드 사이의 상관관계 시각화

```
corr_heatmap=sns.heatmap(corr, # value
        cbar=True, # 사이브바 표시, defalut True
        annot=True, # 값 표시, defalut False
        annot_kws={'size':20}, #annot의 글자크기
        fmt='.2f',  # 숫자형식
        square=True, # 정사각형 여부
        cmap='Blues') # 색상
```



상관관계 시각화(heatmap)

```
corr_heatmap=sns.heatmap(corr, # value
                         cbar=True, # 사이드바 표시, defalut True
                         annot=True, # 값 표시, defalut False
                         annot_kws={'size':20}, #annot의 글자크기
                         fmt='.2f',  # 숫자형식
                         square=True, # 정사각형 여부
                         cmap='RdYlBu_r', #색상
                         vmin=0.95, #최소값
                         vmax=1 #최대값
                         )
```

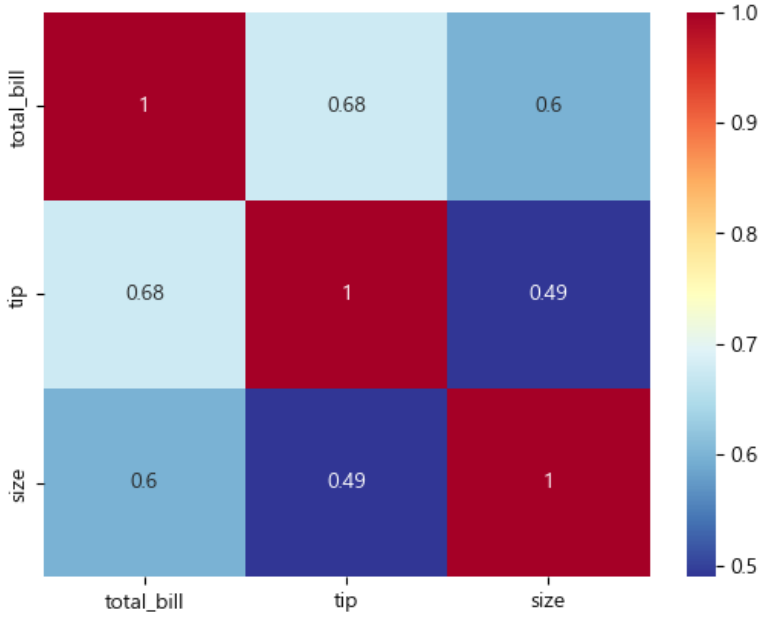


상관관계분석 시각화(heatmap)

```
tips=sns.load_dataset('tips')
tips_corr=tips.corr()
tips_corr


tips_corr_heatmap=sns.heatmap(tips_corr,
                              cbar=True,
                              annot=True,
                              cmap='RdYlBu_r')
```

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

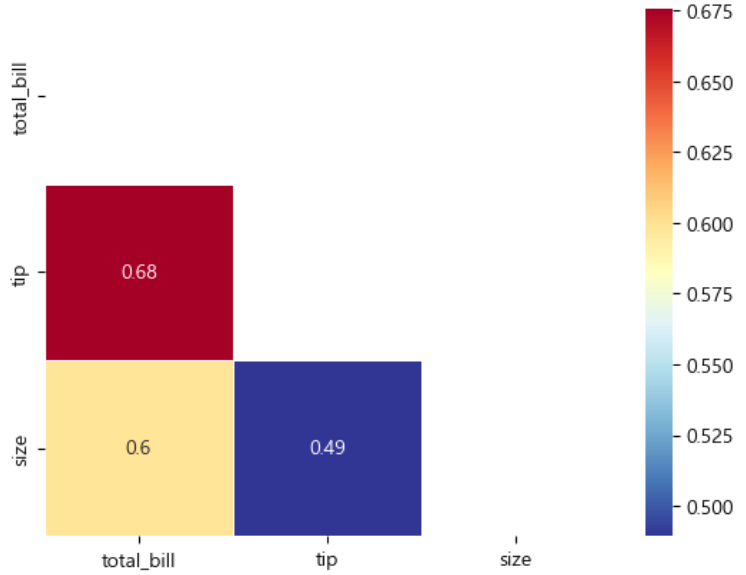


상관관계분석 시각화(heatmap)

```
import numpy as np
fig,ax=plt.subplots(figsize=(7,7))
mask=np.zeros_like(tips_corr, dtype=np.bool)
print(mask)
mask[np.triu_indices_from(mask)]=True
print(mask)

tips_corr_heatmap=sns.heatmap(tips_corr,
                              annot=True,
                              mask=mask,
                              linewidth=0.5,
                              cmap='RdYlBu_r')
```

```
[[False False False]
 [False False False]
 [False False False]]
[[ True  True  True]
 [False  True  True]
 [False False  True]]
```

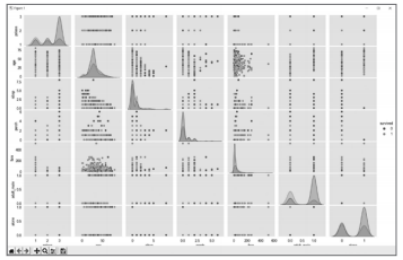


4. 상관관계 분석과 히트맵 시각화

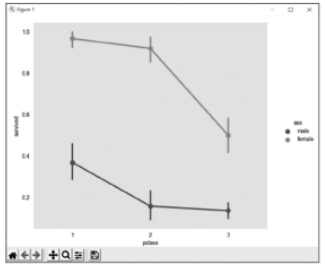
- **타이타닉호 생존율 분석하기(실습)**
  - 타이타닉호의 생존자와 관련된 변수의 상관관계를 찾아봄
  - 생존과 가장 상관도가 높은 변수는 무엇인지 분석
  - 상관 분석을 위해 피어슨 상관 계수를 사용
  - 변수 간의 상관관계는 시각화하여 분석

타이타닉호 생존율 분석하기	
목표	타이타닉호 승객 변수를 분석하여 생존율과의 상관관계를 찾는다.
핵심 개념	상관 분석, 상관 계수, 피어슨 상관 계수, 히트맵
데이터 수집	타이타닉 데이터: seaborn 내장 데이터셋
데이터 준비	결측치 치환: 중앙값 치환, 최빈값 치환
데이터 탐색	1. 정보 확인: info() 2. 차트를 통한 데이터 탐색: pie(), countplot()
데이터 모델링	1. 모든 변수 간 상관 계수 구하기 2. 지정한 두 변수 간 상관계수 구하기
결과 시각화	

1. 산점도를 이용한 시각화



2. 특정 변수 간 상관관계 시각화



3. 히트맵을 이용한 시각화