

2.2. 에너지 데이터 분석 실습



넘파이(Numpy)

NumPy(Numerical Python)는 데이터 분석을 포함해 수학과 과학연산을 위한 파이썬 기본 패키지로 고성능의 다차원 배열 객체와 다양한 객체에 대해 고속 연산을 가능하게 합니다.

```
data = np.array([1,2,3])
```

data

1
2
3

data

1
2
3

.max() = 3

data

1
2
3

.min() = 1

data

1
2
3

.sum() = 6

```
np.array([ [[1,2],[3,4]],  
          [[5,6],[7,8]] ])
```



	5	6
1	2	8
3	4	

```
np.random.random((4,3,2))
```

	0.3	0.6	0.8
0.2	0.5	0.3	0.8
0.7	0.6	0.1	0.5
0.4	0.5	0.5	0.3
0.1	0.1	0.4	

$$MeanSquareError = \frac{1}{n} \sum_{i=1}^n (Y_{prediction_i} - Y_i)^2 \rightarrow \text{error} = (1/n) * \text{np.sum}(\text{np.square}(\text{predictions} - \text{labels}))$$

넘파이(Numpy)

```
import numpy as np
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
arr = np.array(lst)
arr
```

```
[Out] array([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

```
arr.mean()
```

```
[Out] 5.5
```

```
print('Mean:', arr.mean())
```

```
print('Median:', np.median(arr))
```

```
print('Range (Max - min):', np.ptp(arr))
```

```
print('Standard deviation:', arr.std())
```

```
print('80th percentile:', np.percentile(arr, 80))
```

```
print('0.2-quantile:', np.quantile(arr, 0.2))
```

```
[Out]
```

```
Mean: 5.5
```

```
Median: 5.5
```

```
Range (Max - min): 9
```

```
Standard deviation: 2.87228132323
```

```
80th percentile: 8.2
```

```
0.2-quantile: 2.8000000000000003
```

판다스(Pandas)

Pandas는 데이터 분석을 위해 널리 사용되는 파이썬 라이브러리 패키지입니다.
행과 열로 이루어진 데이터프레임 자료구조를 사용하여, 데이터 분석에 있어 높은 수준의 성능을 발휘합니다.

■ 데이터프레임(DataFrame)

컬럼명(Column Names)

	power_usage	gas_usage	water_usage
2022-05-01	86	98	84
2022-05-02	70	31	88
2022-05-03	0	29	51
2022-05-04	32	77	75
2022-05-05	57	48	17

인덱스
(Index)

데이터

판다스(Pandas)

■ Pandas 라이브러리 импорт

```
import pandas as pd
```

■ 데이터프레임 생성

```
dates = pd.date_range('20220501', periods=31)
data = np.random.randint(0,100,(31,3))
df = pd.DataFrame(data=data, index=dates,
                  columns=['power_usage', 'gas_usage', 'water_usage'])
```

■ 데이터 확인

df.head()

	power_usage	gas_usage	water_usage
2022-05-01	86	98	84
2022-05-02	70	31	88
2022-05-03	0	29	51
2022-05-04	32	77	75
2022-05-05	57	48	17

df.tail()

	power_usage	gas_usage	water_usage
2022-05-27	90	84	63
2022-05-28	39	33	29
2022-05-29	5	77	72
2022-05-30	42	73	8
2022-05-31	54	27	3

판다스(Pandas)

■ 자료구조 파악

df.info()

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 31 entries, 2020-01-01 to 2020-01-31
Freq: D
Data columns (total 3 columns):
#   Column          Non-Null Count  Dtype
---  -
0   power_usage     31 non-null    int64
1   gas_usage       31 non-null    int64
2   water_usage     31 non-null    int64
dtypes: int64(3)
memory usage: 992.0 bytes
```

■ 데이터 타입 확인

df.dtypes

```
power_usage    int64
gas_usage      int64
water_usage    int64
dtype: object
```

■ Null 데이터 확인

df.isnull().sum()

```
power_usage    0
gas_usage      0
water_usage    0
dtype: int64
```

판다스(Pandas)

통계 정보

`df.describe()`

	power_usage	gas_usage	water_usage
count	31.000000	31.000000	31.000000
mean	45.161290	52.193548	52.161290
std	26.669704	31.679036	30.625803
min	3.000000	4.000000	0.000000
25%	29.000000	22.500000	18.500000
50%	41.000000	54.000000	57.000000
75%	61.500000	84.000000	76.000000
max	97.000000	99.000000	99.000000

데이터 상관관계 분석

`df.corr()`

	power_usage	gas_usage	water_usage
power_usage	1.000000	-0.187838	0.073957
gas_usage	-0.187838	1.000000	0.007285
water_usage	0.073957	0.007285	1.000000

판다스(Pandas)

■ 새 컬럼 생성하기

```
df['energy_usage'] = df['power_usage'] + df['gas_usage']  
df['date'] = pd.to_datetime(df.index, format= '%Y-%m-%d')  
df['day_of_week'] = df['date'].dt.day_name()  
print(df)
```

	power_usage	gas_usage	water_usage	energy_usage	date	day_of_week
2020-01-01	95	86	64	181	2020-01-01	Wednesday
2020-01-02	31	82	2	113	2020-01-02	Thursday
2020-01-03	77	16	55	93	2020-01-03	Friday
2020-01-04	43	4	0	47	2020-01-04	Saturday
2020-01-05	44	79	57	123	2020-01-05	Sunday
2020-01-06	27	63	52	90	2020-01-06	Monday
2020-01-07	45	33	16	78	2020-01-07	Tuesday

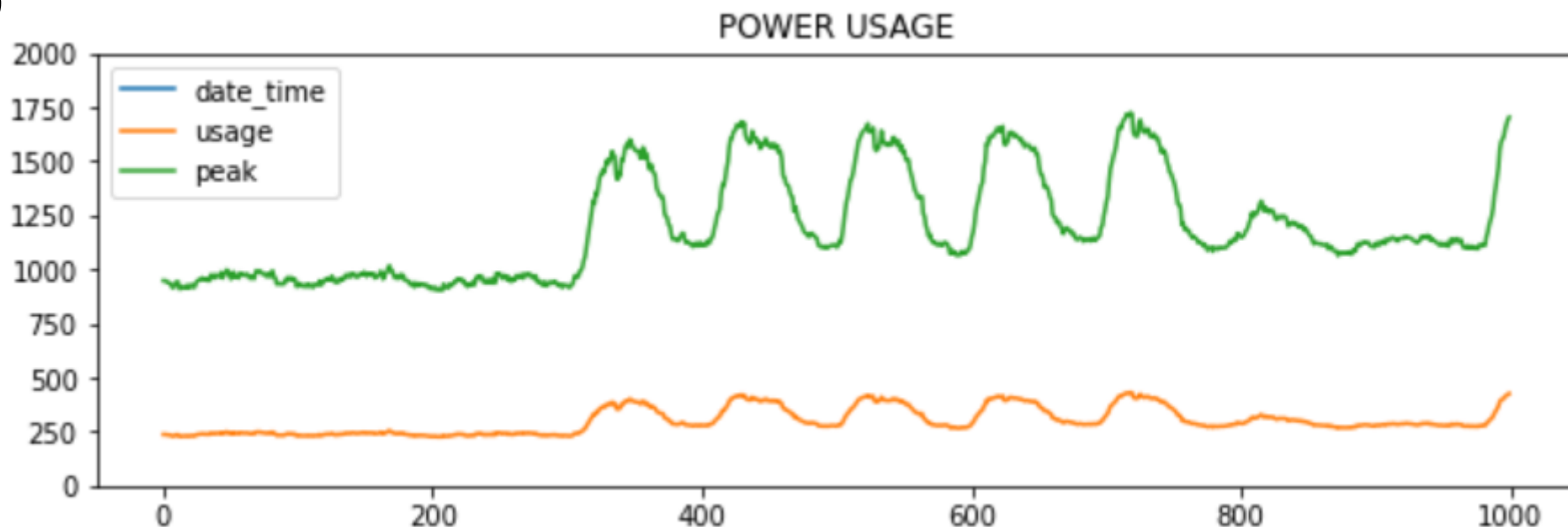
판다스(Pandas)

■ 파일에서 데이터를 로드하는 방법

```
# df_energy = pd.read_csv('data.csv')  
df_energy = pd.read_csv(  
    'https://raw.githubusercontent.com/kgpark88/energy-bigdata-analysis/master/data.csv')
```

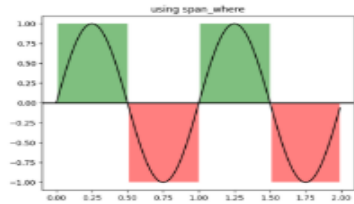
```
df_energy.plot(title='POWER USAGE', figsize=(10, 3), ylim=(0, 2000))
```

```
plt.show()
```

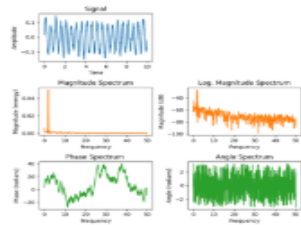


맷플롯립(Matplotlib)

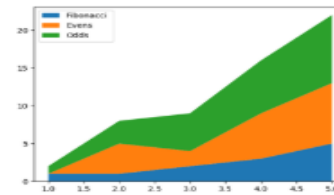
파이썬에서 데이터를 차트나 플롯(Plot)으로 그려주는 라이브러리 패키지로서
가장 많이 사용되는 데이터 시각화(Data Visualization) 패키지입니다.



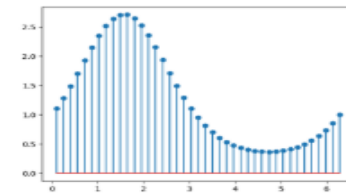
Using span_where



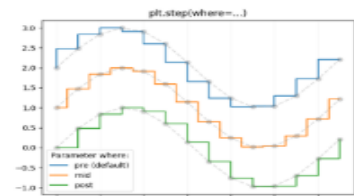
Spectrum Representations



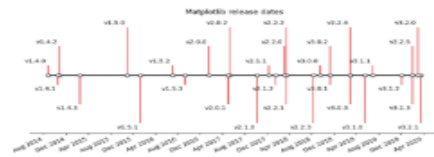
Stackplot Demo



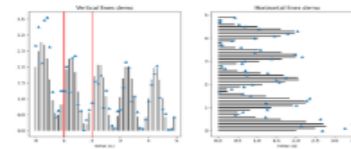
Stem Plot



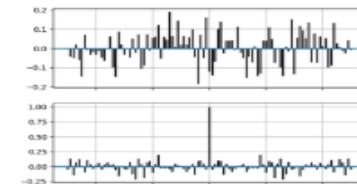
Step Demo



Creating a timeline with lines, dates, and text



hlines and vlines



Cross- and Auto-Correlation Demo

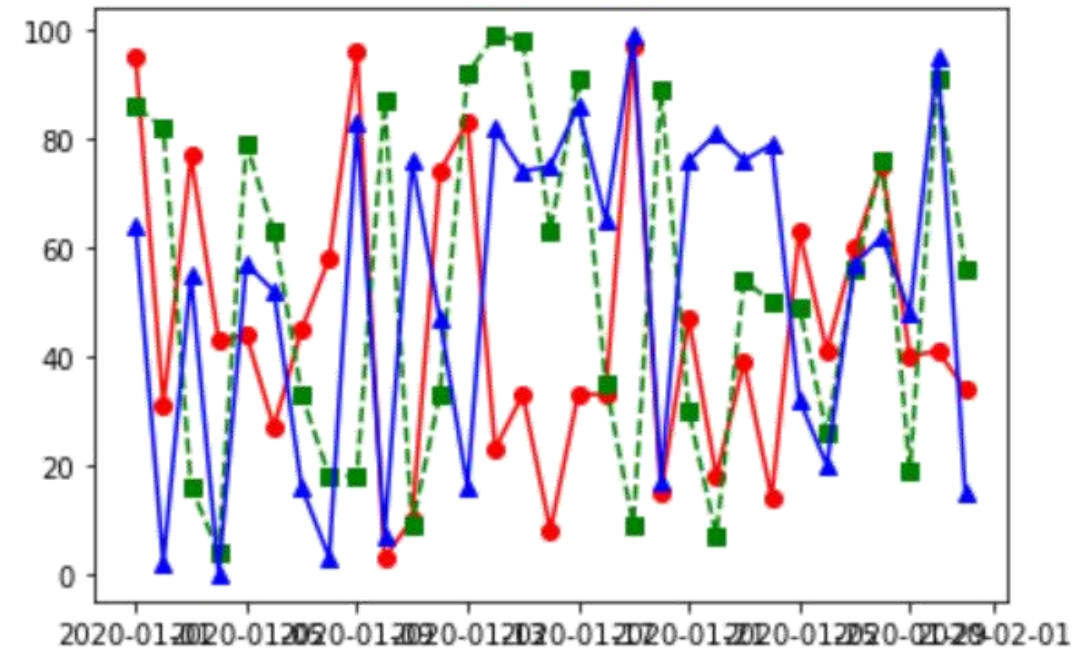
맷플롯립(Matplotlib)

라이브러리 импорт

```
import matplotlib.pyplot as plt
```

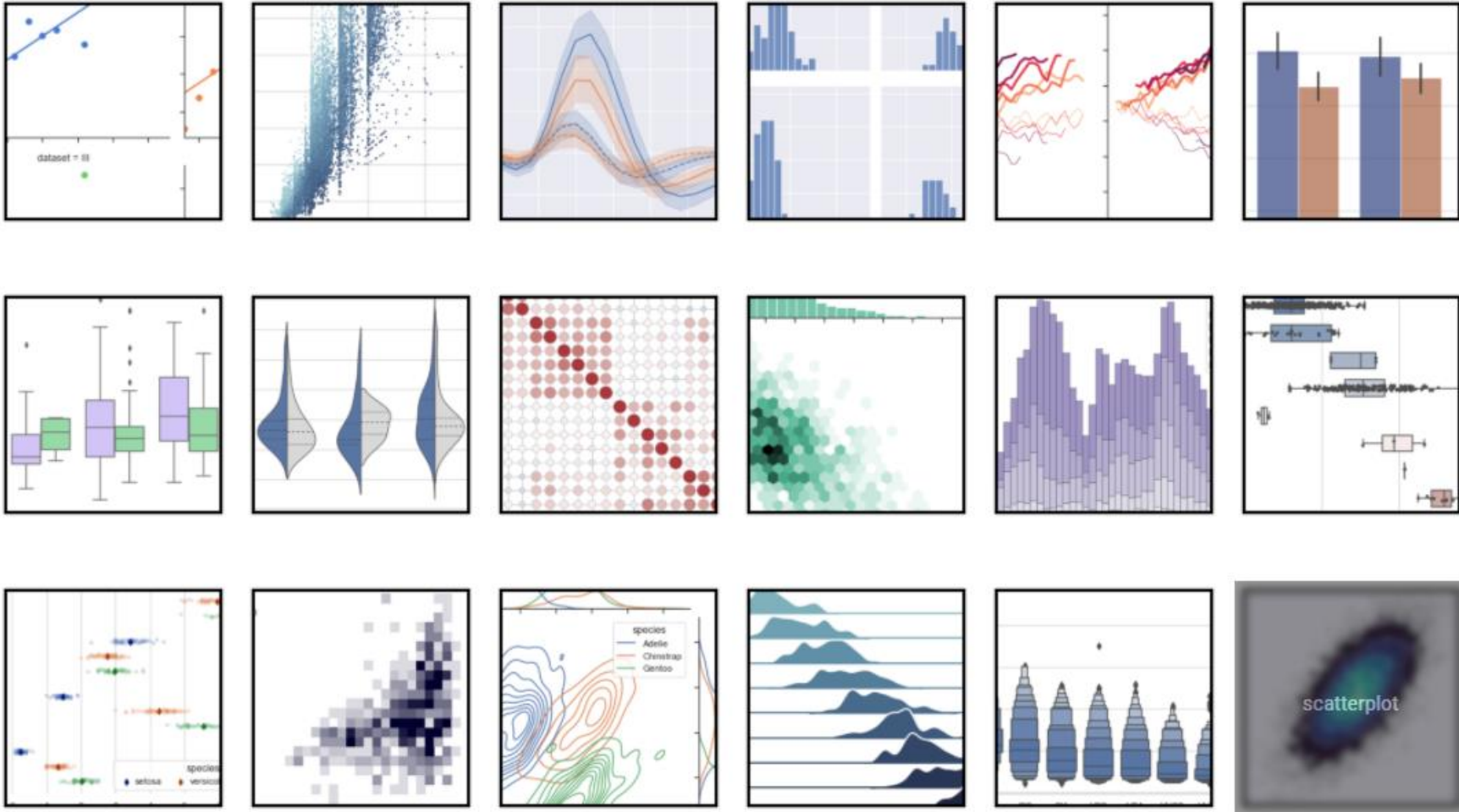
선 그래프

```
plt.plot(df['power_usage'], 'ro-',  
         df['gas_usage'], 'gs--',  
         df['water_usage'], 'b^-',  
         plt.show())
```



씨본(Seaborn)

Matplotlib을 기반으로 다양한 색상 테마와 통계용 차트 등의 기능을 추가한 시각화 패키지입니다.



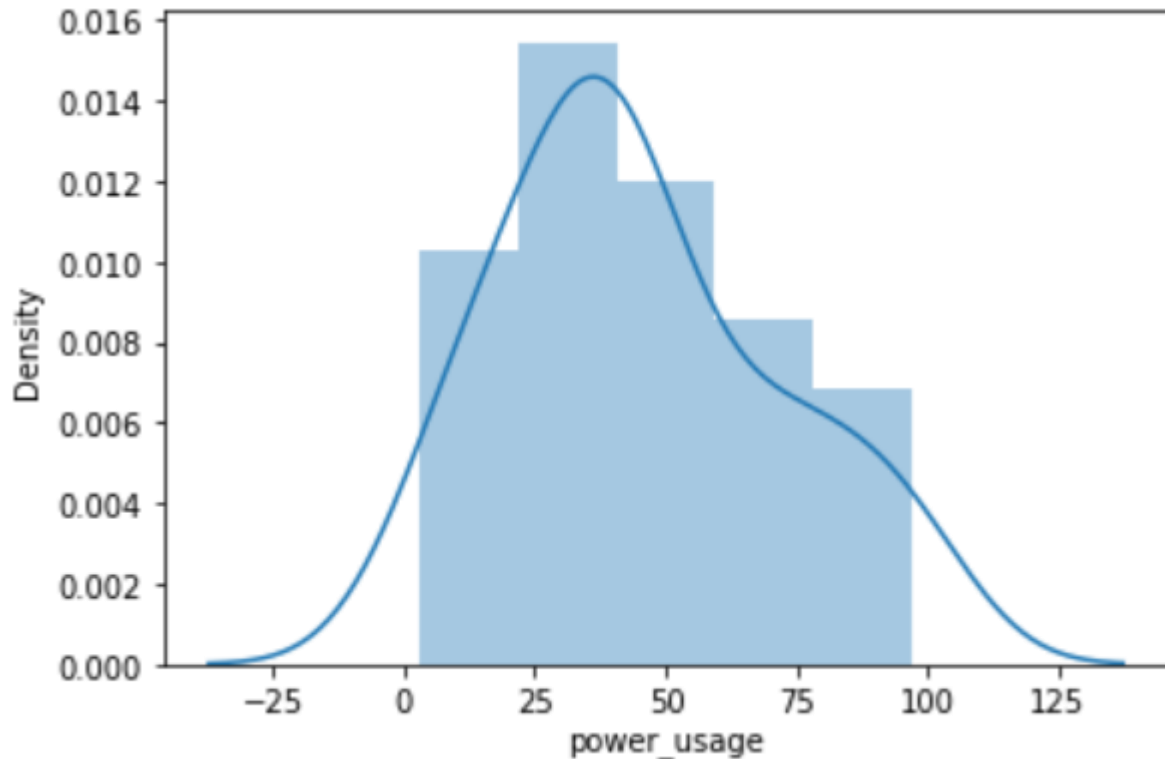
씨본(Seaborn)

라이브러리 импорт

```
import seaborn as sns
```

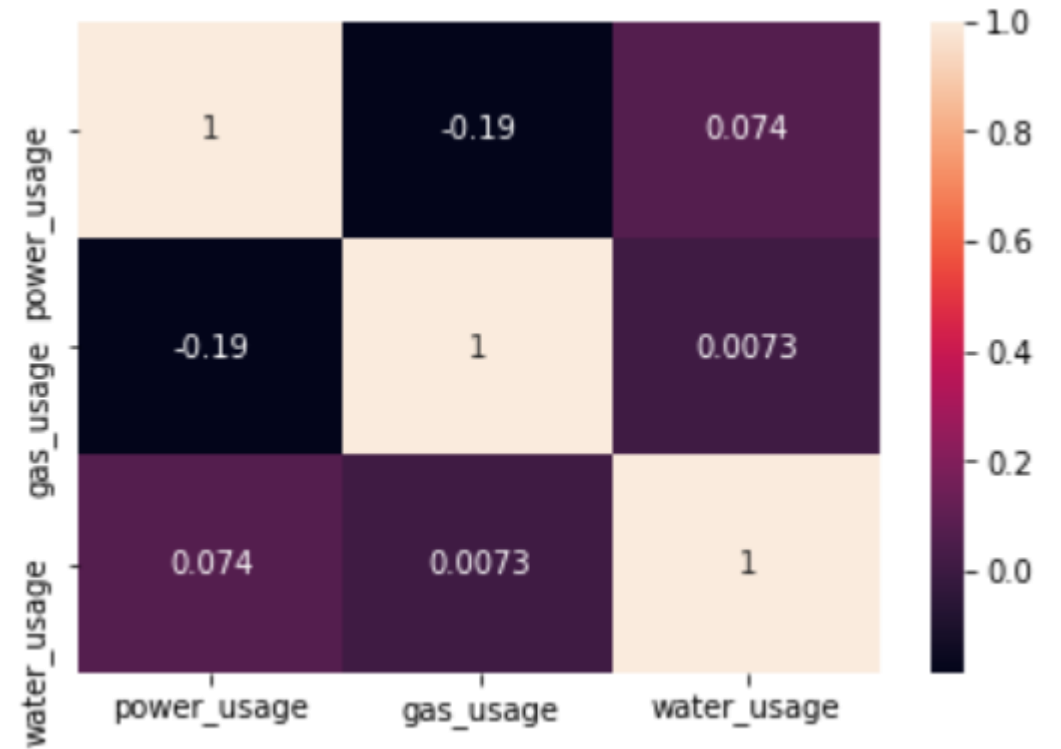
분포 플롯

```
sns.distplot(df['power usage'])
```



히트맵(상관관계)

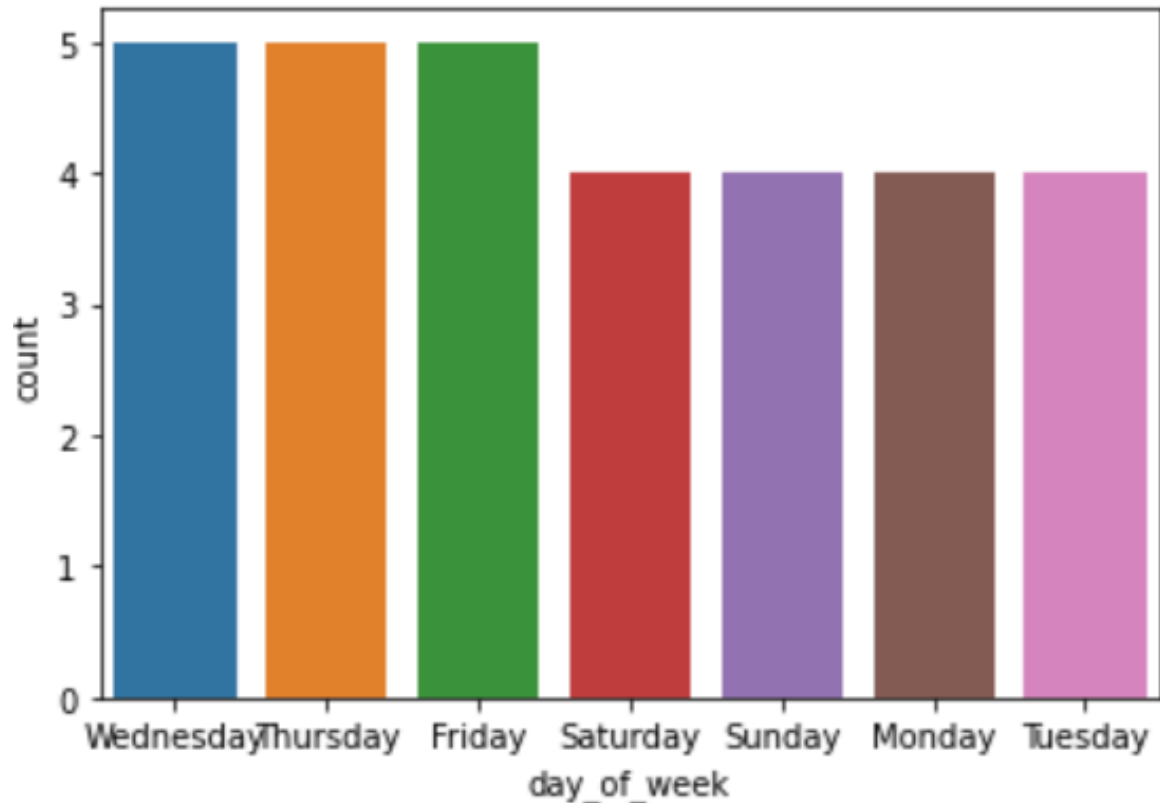
```
sns.heatmap(df.corr(), annot=True)
```



씨본(Seaborn)

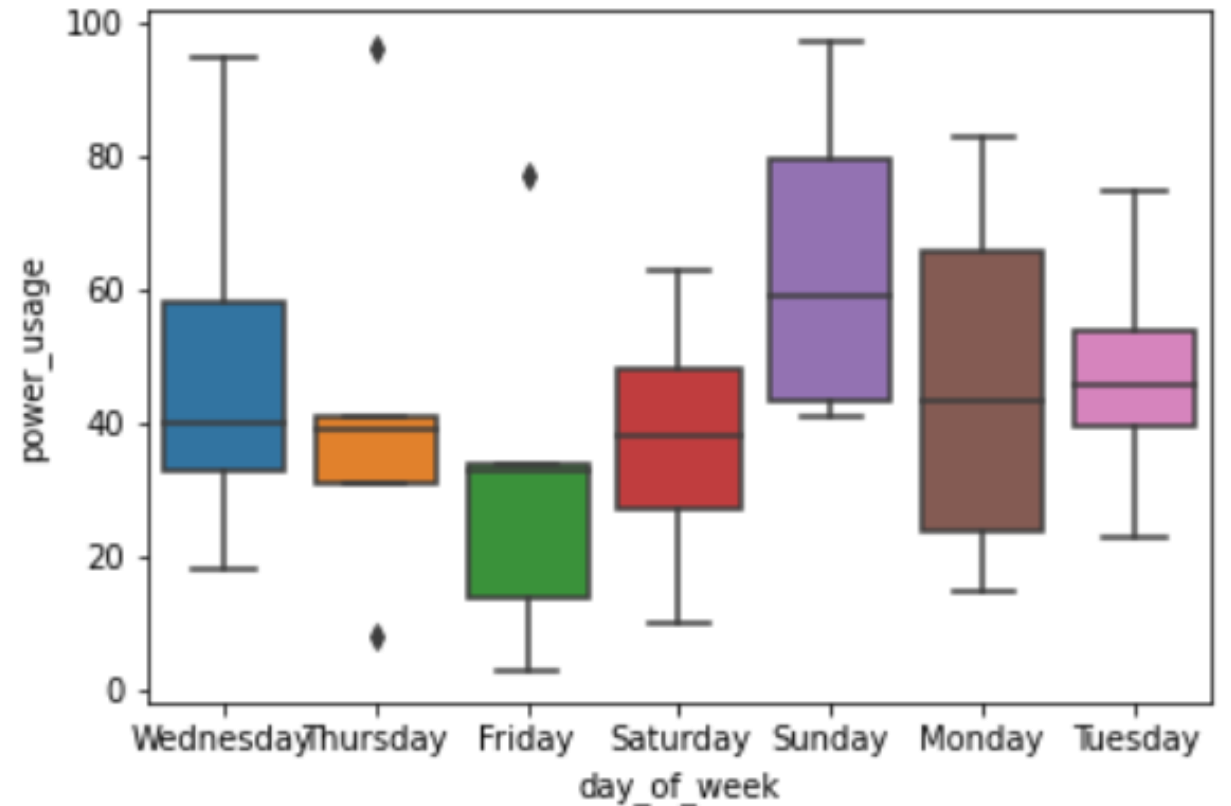
■ 카운트 플롯

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



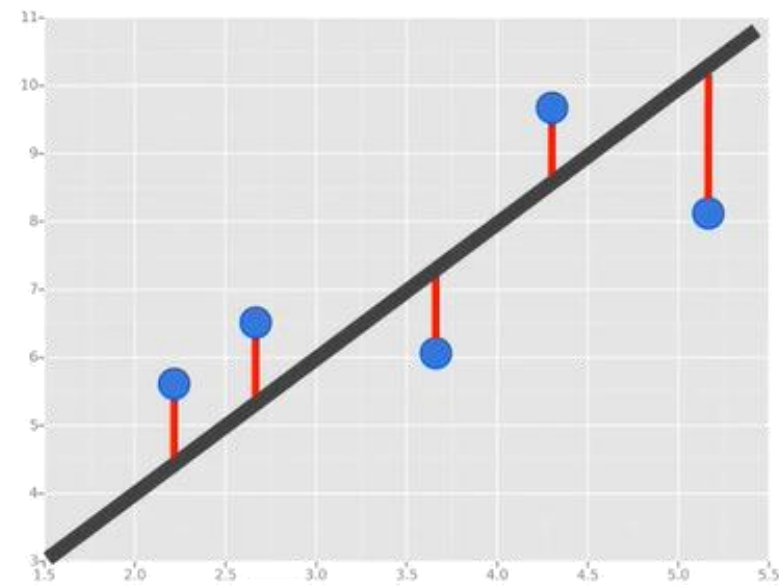
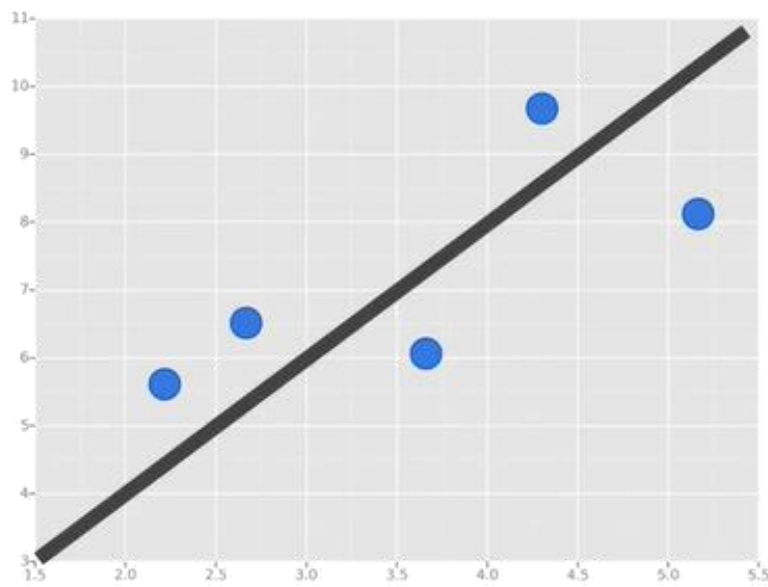
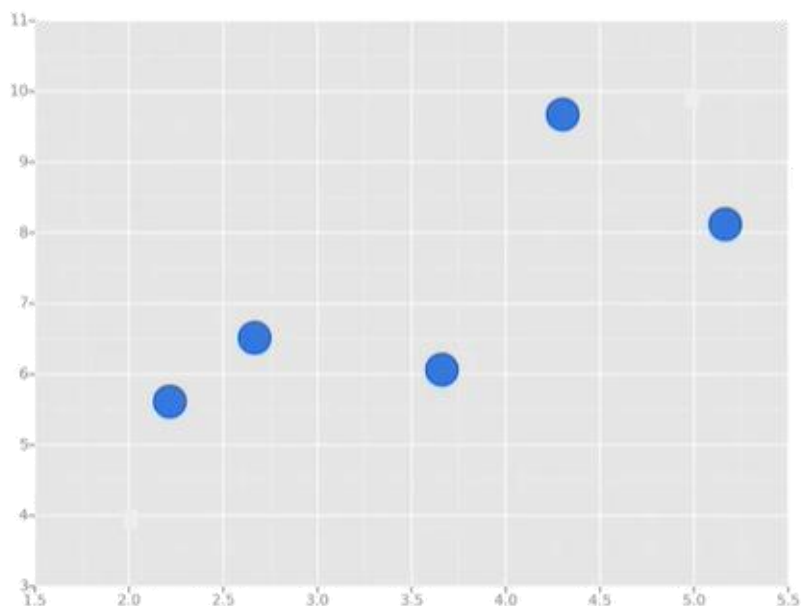
■ 박스 플롯

```
sns.boxplot(x='day_of_week',  
            y='power_usage', data=df)
```



선형 회귀

종속 변수 y 와 한 개 이상의 독립 변수 x 와의 선형 상관 관계를 모델링 하는 회귀분석 기법



선형 회귀

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

# 생산량
output = [110, 125, 140, 145, 160, 166, 179, 190, 200, 215, 230, 250]

# 전력사용량
power_usage = [98, 115, 120, 136, 140, 156, 160, 177, 185, 195, 210, 225]

# p-value : 유의 확률, 일반적으로 0.05 미만일 때 유의미
slope, intercept, r_value, p_value, stderr = stats.linregress(output, power_usage)
```


선형 회귀

생산량 134개일 때 전기사용량 예측

```
product = 134
```

```
print("기울기(slope) : ", slope)
```

```
print("절편(intercept) : ", intercept)
```

```
print("상관계수(r_value) : ", r_value)
```

```
print("유의확률(p_value) : ", p_value )
```

```
print("{}개 => 예측량 {}kWh".format(  
    product, product*slope + intercept))
```

```
plt.scatter(output, power_usage)
```

```
x = np.arange(0, 300)
```

```
y = [(slope*num + intercept) for num in x]
```

```
plt.plot(x, y, 'b', lw=1)
```

```
plt.xlabel("Output(EA)")
```

```
plt.ylabel("Power Usage(kWh)")
```

```
plt.show()
```

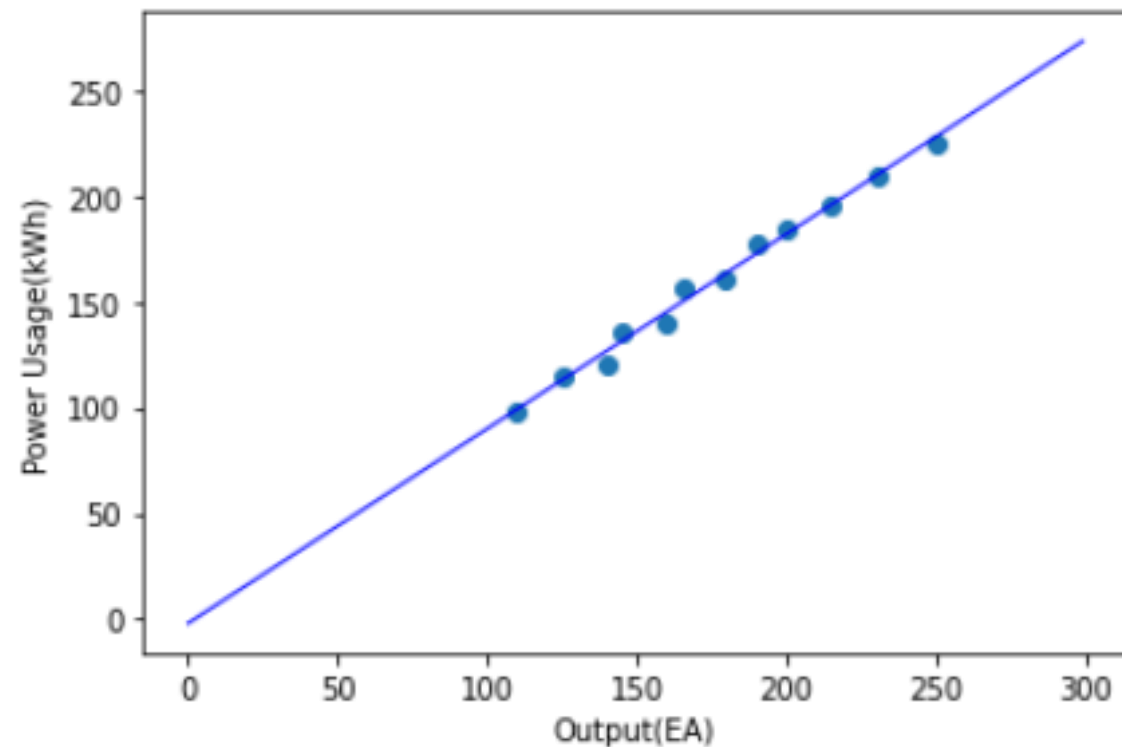
기울기(slope) : 0.9200457304535211

절편(intercept) : -2.024707604744151

상관계수(r_value) : 0.9950415352828844

유의확률(p_value) : 2.3409613797567155e-11

134개 => 예측량 121.26142027602768kWh



에너지 데이터 분석 실습



energy_data_analysis.ipynb

charts.ipynb

에너지 사용량 분석 실습

■ kaggle 가입 및 kgggle.json 다운로드

- kaggle 가입 : <https://www.kaggle.com/>
- kgggle.json 다운로드 : <https://www.kaggle.com/settings> → API

API

Using Kaggle's beta API, you can interact with Competitions and Datasets to download data, make submissions, and more via the command line. [Read the docs](#)

Create New Token

Expire Token

■ 데이터셋

- <https://www.kaggle.com/code/sudalairajkumar/simple-exploration-notebook-ashrae/data>
- train.csv
- test.csv
- building_metadata.csv
- weather_train.csv

에너지 사용량 분석 실습

■ kaggle 패키지 설치

```
!pip install kaggle
```

■ kaggle API 키 업로드

```
from google.colab import files
```

```
files.upload()
```

```
!mkdir ~/.kaggle
```

```
!cp kaggle.json ~/.kaggle/
```

```
!chmod 600 ~/.kaggle/kaggle.json
```

■ kaggle 패키지 설치

```
!kaggle competitions download -c ashrae-energy-prediction
```

에너지 사용량 분석 실습

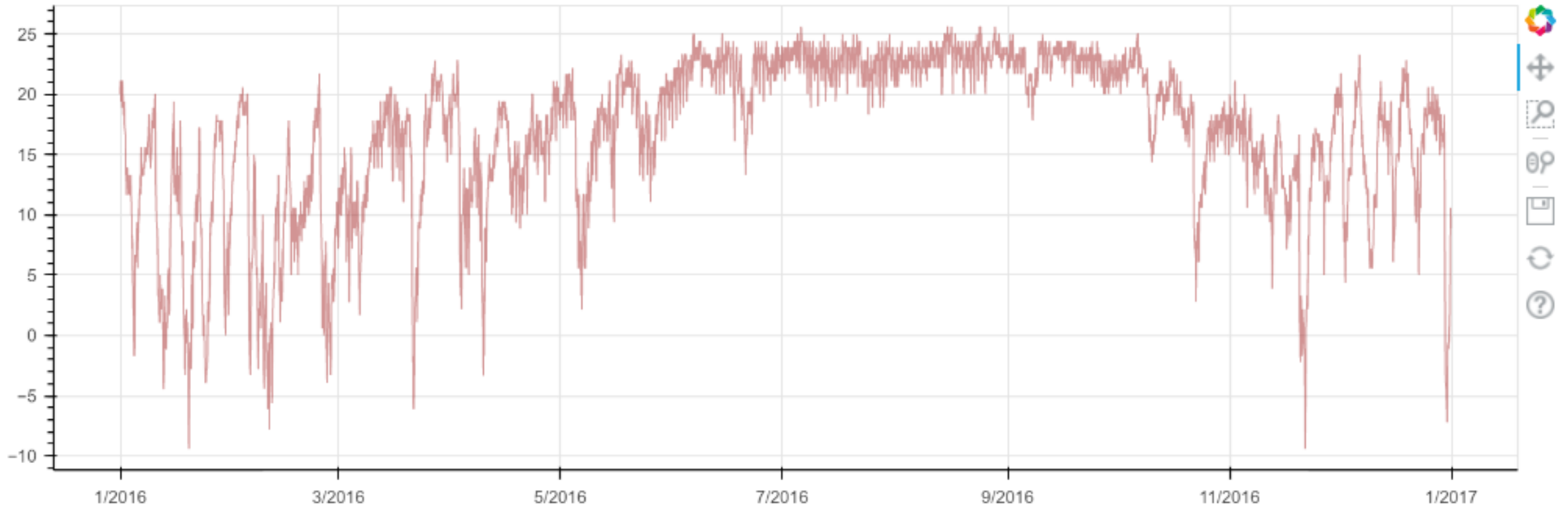


energy_data_exploration.ipynb

Dew Temperature Sea Level Pressure Wind Speed Cloud Coverage

Site:0 Site:1 Site:2 Site:3 Site:4 Site:5 Site:6 Site:7 Site:8 Site:9 Site:10 Site:11 Site:12 Site:13 Site:14 Site:15

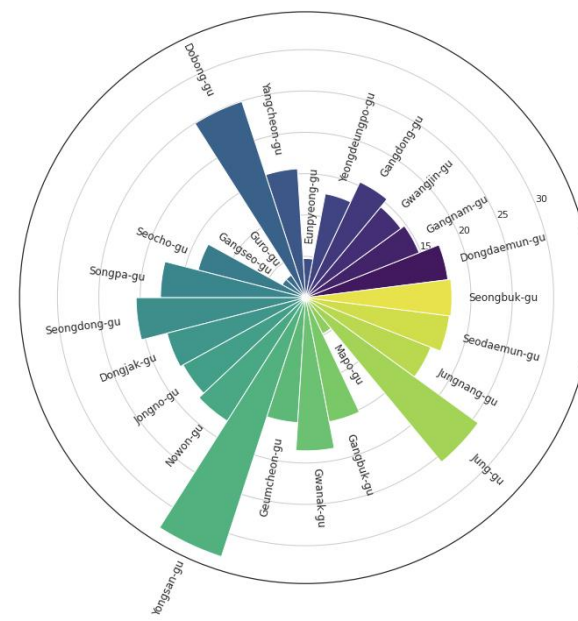
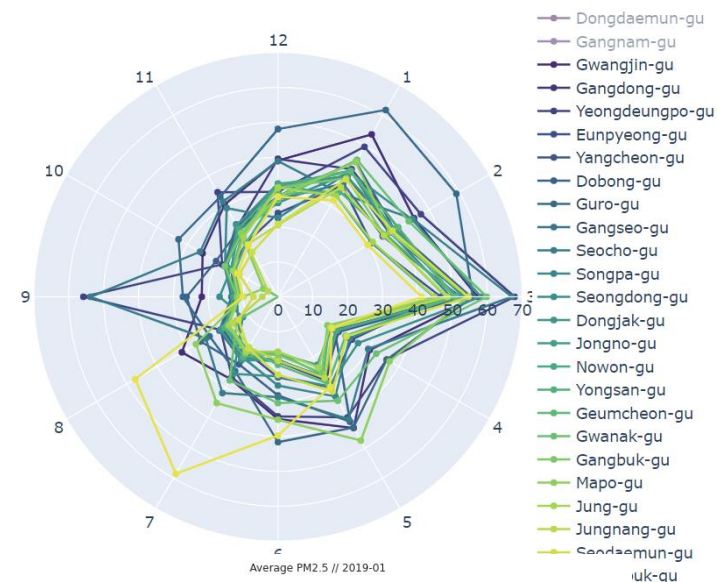
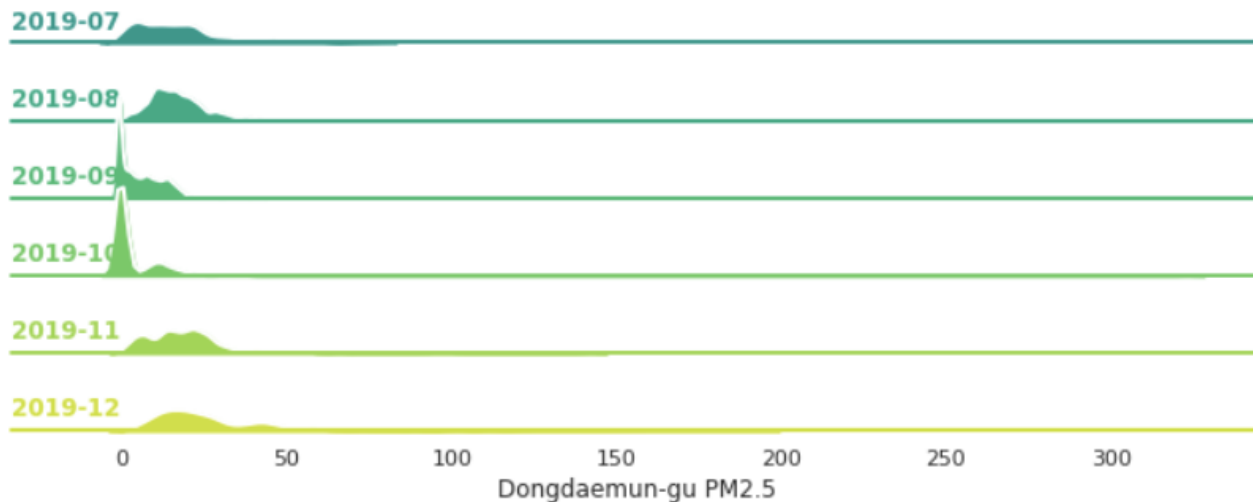
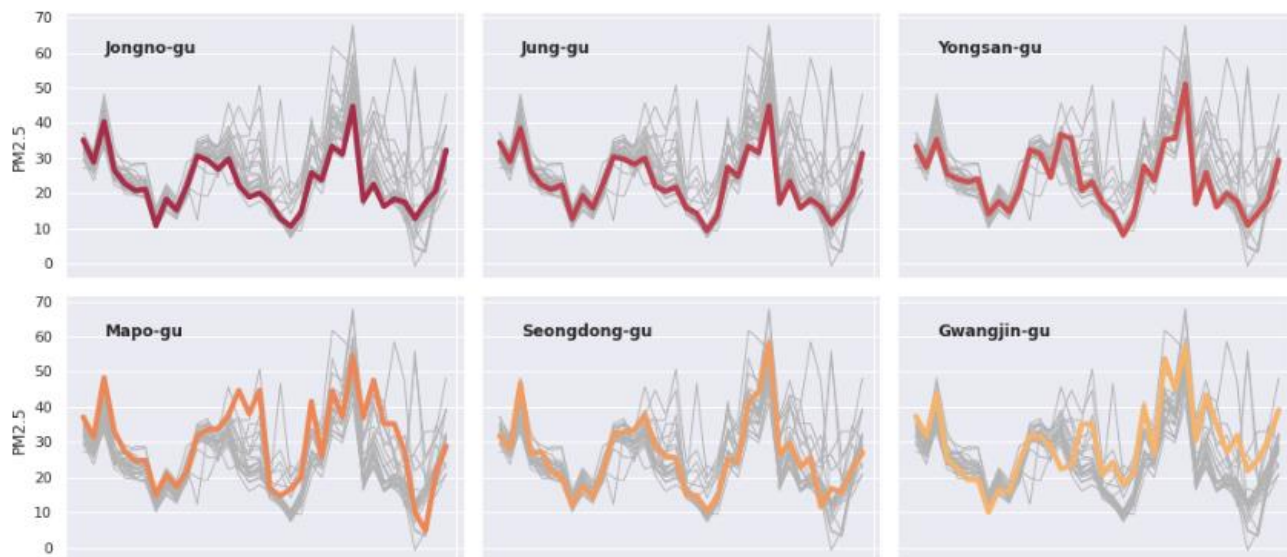
dew_temperature distribution over time



다중 시계열 데이터 분석 시각화 실습



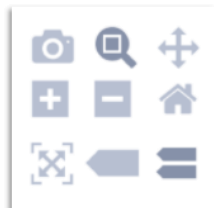
multiple_time_series_data_analysis.ipynb



자바스크립트 차트 패키지 - Plotly

<https://plot.ly/javascript/>

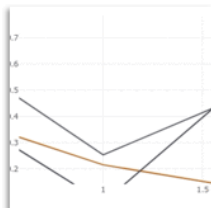
Fundamentals



Configuration Options



Responsive / Fluid Layouts



uirevision in Plotly.react



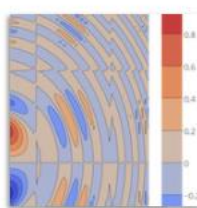
React Plotly.js



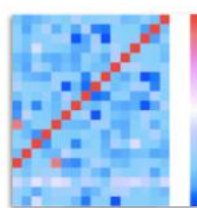
Analytical Apps with Dash

[More Fundamentals >](#)

Scientific Charts



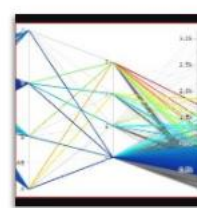
Contour Plots



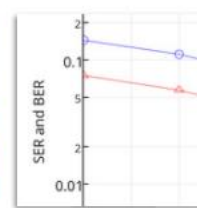
Heatmaps



Ternary Plots



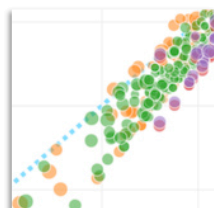
Parallel Coordinates Plot



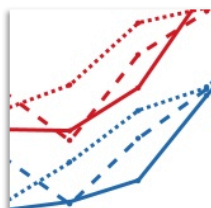
Log Plots

[More Scientific Charts >](#)

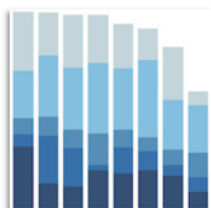
Basic Charts



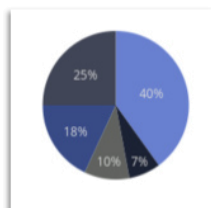
Scatter Plots



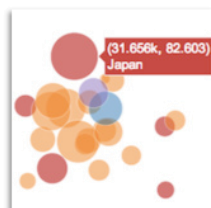
Line Charts



Bar Charts



Pie Charts



Bubble Charts

[More Basic Charts >](#)

Financial Charts



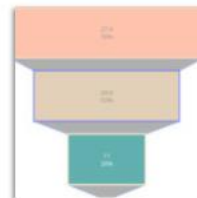
Waterfall Charts



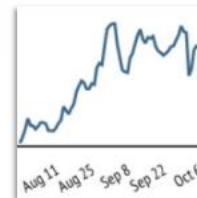
Indicators



Candlestick Charts



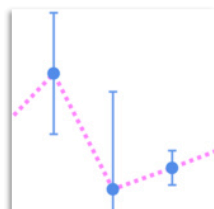
Funnel and Funnelarea Charts



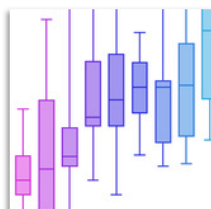
Time Series

[More Financial Charts >](#)

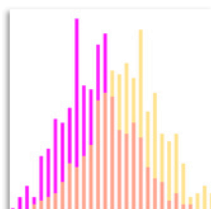
Statistical Charts



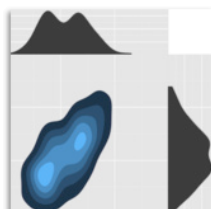
Error Bars



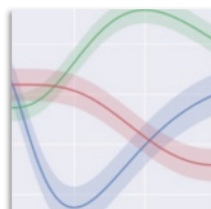
Box Plots



Histograms



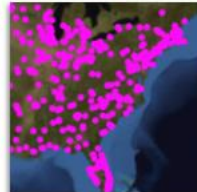
2d Density Plots



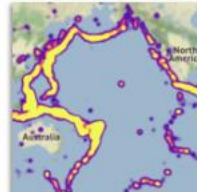
Continuous Error

[More Statistical Charts >](#)

Maps



Mapbox Map Layers



Mapbox Density



Choropleth Mapbox



Lines on Maps



Bubble Maps

[More Maps >](#)

자바스크립트 차트 패키지 - ECharts

<https://echarts.apache.org/examples/en/index.html>



THANK YOU

kgpark88@gmail.com