

## 03장~07장 & Appendix

### ▼ 03-1 불필요한 데이터 삭제하기



[주피터 노트북 뷰어로 보기](#)



[구글 코랩\(Colab\)에서 실행하기](#)

### ▼ 열 삭제하기

```
1 import gdown
2
3 gdown.download('https://bit.ly/3RhoNho', 'ns_202104.csv', quiet=False)
```

### ▼ 마운트 설정 : 구글코랩에서 구글드라이브의 내음을 사용

- 구글드라이브 홈 - 내드라이브 : 영구저장소

```
1 # 1. 마운트 설정
2 from google.colab import drive
3 drive.mount('/content/drive')

Mounted at /content/drive

1 # 2. 현재 작업디렉터리 연결
2 import os
3 os.chdir("drive/My Drive/tb_app/")

1 # 3. 작업디렉터리가 가진 목록을 화면에 표시
2 !ls

bearlist.txt  data  imgs  iris_model.joblib  mushroom.joblib  ns.lib.db  plots

1 # 작업디렉터리 파일을 읽는 예제 실행
2 with open('bearlist.txt', 'r') as f:
3     for line in f:
4         print(line) # print(line, end='')
```

## matplotlib 그래프 한글 깨짐 처리

### ▼ 아나콘다 주피터노트북용 한글깨짐 처리 - 로컬에 설치된

```
1 # 선수작업 1-1
2 # 주요 라이브러리 로드 : 라이브러리가 제공하는 함수 사용
3 import numpy as np # 고속 연산
4 import scipy as sp # 과학 계산
5 import pandas as pd # 데이터프레임을 다룸
```

```
1 # 선수작업 1-2
2 # windows, mac용 (matplotlib: 시각화 라이브러리)
3 import matplotlib as mpl # 글꼴이나 크기 설정
4 import matplotlib.pyplot as plt # 그래프를 플롯, 계산하여 그려줌
5 import platform
6
7 plt.rcParams['axes.unicode_minus'] = False
8
9 if platform.system() == 'Darwin':
10     mpl.rc('font', family='AppleGothic')
11 elif platform.system() == 'Windows':
12     path = "c:/Windows/Fonts/malgun.ttf"
13     font_name = mpl.font_manager.FontProperties(fname=path).get_name()
14     mpl.rc('font', family=font_name)
15 else: # 리눅스
16     print("Unknown System OS")
```

- 구글 코랩용 한글깨짐 처리 - 클라우드 기반 리눅스

```
1 # 선수작업2-1 : 주요 라이브러리 로드  
2 import numpy as np  
3 import scipy as sp  
4 import pandas as pd
```

```
1 # 선수작업2-2
2
3 %matplotlib inline
4
5 import matplotlib as mpl
6 import matplotlib.pyplot as plt
7 import matplotlib.font_manager as fm
8
9 mpl.rcParams['axes.unicode_minus'] = False # -기호 깨짐방지
```

```
1 # 선수작업2-3
2
3 # 나눔폰트 유무 확인
4 sys_font=fm.findSystemFonts()
5 print(f"sys_font number: {len(sys_font)}")
6 print(sys_font)
7
8 nanum_font = [f for f in sys_font if 'Nanum' in f]
9 print(f"nanum_font number: {len(nanum_font)})
```

# 1 # 선수작업2-4

## 2

### 3 nanum font

- 1 # 선수작업2-5
- 2 # 나눔글꼴 없는 경우 설치. 있는 경우 skip
- 3 !apt-get update -qq
- 4 !apt-get install fonts-nanum\* -qq

```
Selecting previously unselected package fonts-nanum.  
(Reading database ... 120880 files and directories currently installed.)  
Preparing to unpack .../fonts-nanum_20200506-1_all.deb ...  
Unpacking fonts-nanum (20200506-1) ...  
Selecting previously unselected package fonts-nanum-coding.
```

```
Preparing to unpack .../fonts-nanum-coding_2.5-3_all.deb ...
Unpacking fonts-nanum-coding (2.5-3) ...
Selecting previously unselected package fonts-nanum-eco.
Preparing to unpack .../fonts-nanum-eco_1.000-7_all.deb ...
Unpacking fonts-nanum-eco (1.000-7) ...
Selecting previously unselected package fonts-nanum-extra.
Preparing to unpack .../fonts-nanum-extra_20200506-1_all.deb ...
Unpacking fonts-nanum-extra (20200506-1) ...
Setting up fonts-nanum-extra (20200506-1) ...
Setting up fonts-nanum (20200506-1) ...
Setting up fonts-nanum-coding (2.5-3) ...
Setting up fonts-nanum-eco (1.000-7) ...
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) ...
```

```
1 # 선수작업2-5-1
2 # 이 셀 실행후 런타임 다시 시작
3 # Cleaning the matplotlib cache directory
4 !rm ~/.cache/matplotlib -rf
```

```
1 # 선수작업2-6
2 # 한글폰트 지정
3 fontpath = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
4 font = fm.FontProperties(fname=fontpath, size=10)
5 # fm._rebuild()
6
7 # 그래프에 retina display 적용
8
9 %config InlineBackend.figure_format = 'retina'
10
11 # Colab 의 한글 폰트 설정 – 전역설정
12
13 plt.rcParams['font.family'] = 'NanumBarunGothic'
```

```
1 # 그래프 한글 설정 확인1 : 2-7
2 # 파일 데이터 로드
3 df_tpop = pd.read_csv("data/20200521_202004_주민등록인구및세대현황.csv",
4                         encoding="cp949", engine="python")
5 df_pro = df_tpop
```

```
1 df_tpop
```

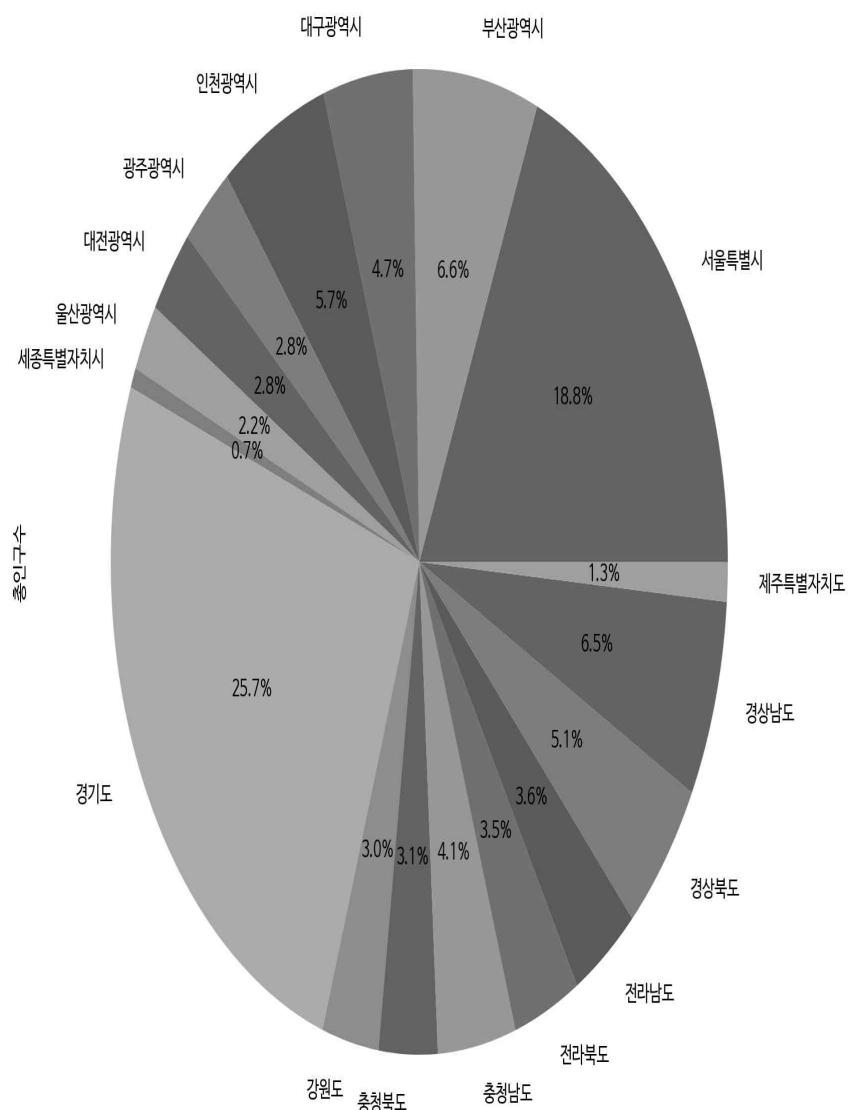
	행정구역	총인구수	세대수	세대당 인구수	남자인구 수	여자인구 수	남여비율
0	서울특별시	9726787	4361645	2.23	4737843	4988944	0.95

```

1 # 그래프 한글 설정 확인2 : 2-8
2 # 그래프 플롯 후 파일 저장
3 from google.colab import files
4
5 df_tpop.plot(kind='pie', y='총인구수', autopct='%1.1f%%',
6               labels=df_tpop['행정구역'], legend=False, figsize=(10, 10))
7 plt.title('2020년 04월 행정구역별 주민등록인구 현황')
8 plt.savefig('plots/2020_04_행정구역별_주민등록인구_원형.png') # 구글드라이브에 저장
9 files.download("plots/2020_04_행정구역별_주민등록인구_원형.png") # 저장된 그림파일 다운로드
10 plt.show()
11 plt.close()

```

2020년 04월 행정구역별 주민등록인구 현황



## ▼ 기본 코드 연습

```

1 list_1 = [5, 6, 7, 8, 9]
2 list_1[2:]

```

```
[7, 8, 9]
```

```
1 a1 = 5
2 a1
```

```
5
```

```
1 b1 = "자치구"
2 b1
```

```
'자치구'
```

```
1 자치구 = ["종로구", "중구", "용산구"]
2 자치구
```

```
['종로구', '중구', '용산구']
```

```
1 자치구[0:2]
```

```
['종로구', '중구']
```

```
1 자치구[-1]
```

```
'용산구'
```

```
1 자치구[1:]
```

```
['중구', '용산구']
```

```
1 자치구[:]
```

```
['종로구', '중구', '용산구']
```

## ▼ 실습: 실무데이터 다루기

```
1 df_ghgs = pd.read_csv("data/1999-2020_ghgs.csv", encoding="cp949")
2 df_ghgs
```

	지점	시간	CO2_ppm	CH4_ppm	N2O_ppm	CFC11_ppm	CFC12_ppm	CFC
0	안면도	1999-01-01	373.1	NaN	NaN	NaN	NaN	
1	안면도	1999-02-01	374.0	NaN	315.2	266.9	534.1	
2	안면도	1999-03-01	374.9	NaN	314.6	267.5	535.1	
3	안면도	1999-04-01	375.1	1869.0	314.2	266.7	534.7	
4	안면도	1999-05-01	374.0	1863.0	314.6	268.6	535.1	



```
1 df_ghgs[["시간", "CO2_ppm"]]
2
```

```
1 df_ghgs.head()
```

```
1 df_ghgs.tail()
```

```
1 df_ghgs.shape
```

```
(264, 9)
```

```
1 df_ghgs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264 entries, 0 to 263
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   지점        264 non-null    object  
 1   시간        264 non-null    object  
 2   CO2_ppm     259 non-null    float64 
 3   CH4_ppm     228 non-null    float64 
 4   N2O_ppm     193 non-null    float64 
 5   CFC11_ppm   244 non-null    float64 
 6   CFC12_ppm   239 non-null    float64 
 7   CFC113_ppm  150 non-null    float64 
 8   SF6_ppm     158 non-null    float64 
dtypes: float64(7), object(2)
memory usage: 18.7+ KB
```

```
1 df_ghgs.dtypes
```

지점	object
시간	object
CO2_ppm	float64
CH4_ppm	float64
N2O_ppm	float64
CFC11_ppm	float64
CFC12_ppm	float64
CFC113_ppm	float64
SF6_ppm	float64
dtype:	object

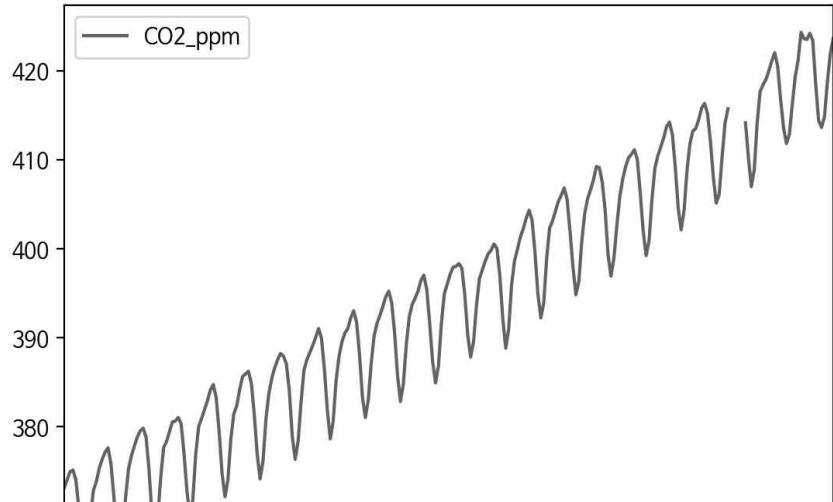
```
1 df_ghgs["시간"] = pd.to_datetime(df_ghgs["시간"])
2 df_ghgs["시간"]
```

0	1999-01-01
1	1999-02-01
2	1999-03-01
3	1999-04-01
4	1999-05-01
...	
259	2020-08-01
260	2020-09-01
261	2020-10-01
262	2020-11-01
263	2020-12-01

Name: 시간, Length: 264, dtype: datetime64[ns]

```
1 df_ghgs.plot.line(x="시간", y="CO2_ppm")
```

&lt;Axes: xlabel='시간'&gt;



```
1 df_ghgs.describe()
| *
|
```

```
1 type(df_ghgs)
```

```
pandas.core.frame.DataFrame
```

```
1 type(df_ghgs[["CO2_ppm", "CH4_ppm"]])
```

```
pandas.core.frame.DataFrame
```

```
1 df_ghgs.columns
```

```
Index(['지점', '시간', 'CO2_ppm', 'CH4_ppm', 'N2O_ppm', 'CFC11_ppm', 'CFC12_ppm',
       'CFC113_ppm', 'SF6_ppm'],
      dtype='object')
```

```
1 df_ghgs_corr = df_ghgs.iloc[:, 2: ].corr()
```

```
2 df_ghgs_corr
```

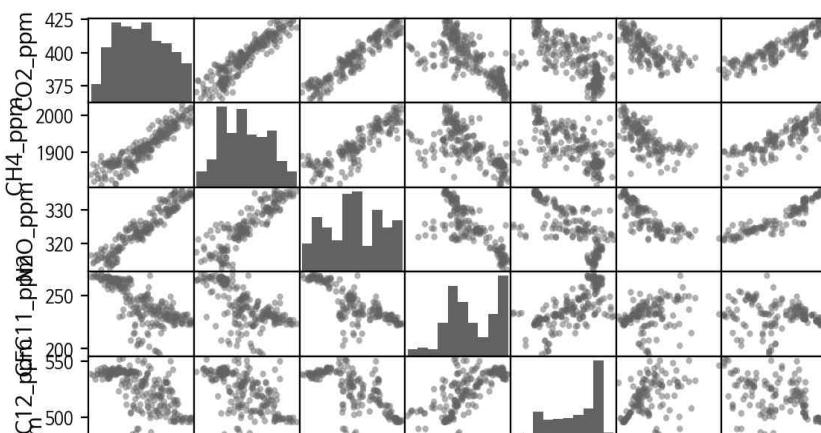
	CO2_ppm	CH4_ppm	N2O_ppm	CFC11_ppm	CFC12_ppm	CFC
CO2_ppm	1.000000	0.922095	0.949458	-0.736793	-0.671967	-
CH4_ppm	0.922095	1.000000	0.883395	-0.634872	-0.609726	-
N2O_ppm	0.949458	0.883395	1.000000	-0.791122	-0.700192	-
CFC11_ppm	-0.736793	-0.634872	-0.791122	1.000000	0.743244	
CFC12_ppm	-0.671967	-0.609726	-0.700192	0.743244	1.000000	
CFC113_ppm	-0.694139	-0.644629	-0.790626	0.310456	0.439440	
SF6_ppm	0.080050	0.027200	0.027052	0.122000	0.442601	

◀ ▶

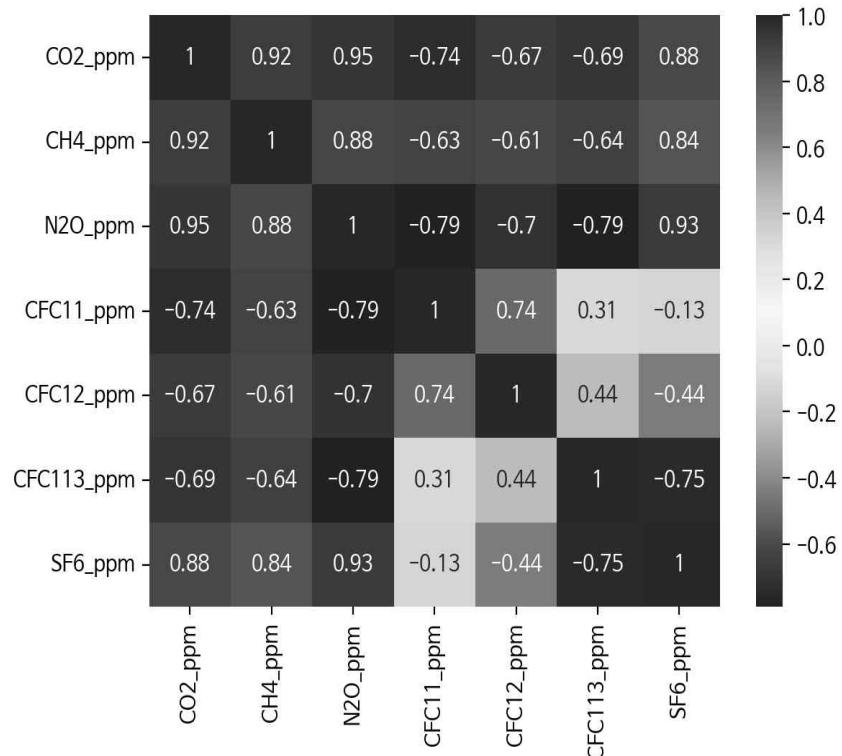
```
1 # 산점도행렬(산포행렬)
```

```
2 from pandas.plotting import scatter_matrix
3 scatter_matrix(df_ghgs.iloc[:, 2:])
```

```
array([[<Axes: xlabel='CO2_ppm', ylabel='CO2_ppm'>,
       <Axes: xlabel='CH4_ppm', ylabel='CO2_ppm'>,
       <Axes: xlabel='N2O_ppm', ylabel='CO2_ppm'>,
       <Axes: xlabel='CFC11_ppm', ylabel='CO2_ppm'>,
       <Axes: xlabel='CFC12_ppm', ylabel='CO2_ppm'>,
       <Axes: xlabel='CFC113_ppm', ylabel='CO2_ppm'>,
       <Axes: xlabel='SF6_ppm', ylabel='CO2_ppm'>],
      [<Axes: xlabel='CO2_ppm', ylabel='CH4_ppm'>,
       <Axes: xlabel='CH4_ppm', ylabel='CH4_ppm'>,
       <Axes: xlabel='N2O_ppm', ylabel='CH4_ppm'>,
       <Axes: xlabel='CFC11_ppm', ylabel='CH4_ppm'>,
       <Axes: xlabel='CFC12_ppm', ylabel='CH4_ppm'>,
       <Axes: xlabel='CFC113_ppm', ylabel='CH4_ppm'>,
       <Axes: xlabel='SF6_ppm', ylabel='CH4_ppm'>],
      [<Axes: xlabel='CO2_ppm', ylabel='N2O_ppm'>,
       <Axes: xlabel='CH4_ppm', ylabel='N2O_ppm'>,
       <Axes: xlabel='N2O_ppm', ylabel='N2O_ppm'>,
       <Axes: xlabel='CFC11_ppm', ylabel='N2O_ppm'>,
       <Axes: xlabel='CFC12_ppm', ylabel='N2O_ppm'>,
       <Axes: xlabel='CFC113_ppm', ylabel='N2O_ppm'>,
       <Axes: xlabel='SF6_ppm', ylabel='N2O_ppm'>],
      [<Axes: xlabel='CO2_ppm', ylabel='CFC11_ppm'>,
       <Axes: xlabel='CH4_ppm', ylabel='CFC11_ppm'>,
       <Axes: xlabel='N2O_ppm', ylabel='CFC11_ppm'>,
       <Axes: xlabel='CFC11_ppm', ylabel='CFC11_ppm'>,
       <Axes: xlabel='CFC12_ppm', ylabel='CFC11_ppm'>,
       <Axes: xlabel='CFC113_ppm', ylabel='CFC11_ppm'>,
       <Axes: xlabel='SF6_ppm', ylabel='CFC11_ppm'>],
      [<Axes: xlabel='CO2_ppm', ylabel='CFC12_ppm'>,
       <Axes: xlabel='CH4_ppm', ylabel='CFC12_ppm'>,
       <Axes: xlabel='N2O_ppm', ylabel='CFC12_ppm'>,
       <Axes: xlabel='CFC11_ppm', ylabel='CFC12_ppm'>,
       <Axes: xlabel='CFC12_ppm', ylabel='CFC12_ppm'>,
       <Axes: xlabel='CFC113_ppm', ylabel='CFC12_ppm'>,
       <Axes: xlabel='SF6_ppm', ylabel='CFC12_ppm'>],
      [<Axes: xlabel='CO2_ppm', ylabel='CFC113_ppm'>,
       <Axes: xlabel='CH4_ppm', ylabel='CFC113_ppm'>,
       <Axes: xlabel='N2O_ppm', ylabel='CFC113_ppm'>,
       <Axes: xlabel='CFC11_ppm', ylabel='CFC113_ppm'>,
       <Axes: xlabel='CFC12_ppm', ylabel='CFC113_ppm'>,
       <Axes: xlabel='CFC113_ppm', ylabel='CFC113_ppm'>,
       <Axes: xlabel='SF6_ppm', ylabel='CFC113_ppm'>],
      [<Axes: xlabel='CO2_ppm', ylabel='SF6_ppm'>,
       <Axes: xlabel='CH4_ppm', ylabel='SF6_ppm'>,
       <Axes: xlabel='N2O_ppm', ylabel='SF6_ppm'>,
       <Axes: xlabel='CFC11_ppm', ylabel='SF6_ppm'>,
       <Axes: xlabel='CFC12_ppm', ylabel='SF6_ppm'>,
       <Axes: xlabel='CFC113_ppm', ylabel='SF6_ppm'>,
       <Axes: xlabel='SF6_ppm', ylabel='SF6_ppm'>]], dtype=object)
```



```
1 # 상관계수 행렬
2 import seaborn as sns
3 # annot=True : 상관계수, cmap : 색상표
4 ax = sns.heatmap(df_ghgs_corr, annot=True, cmap="RdBu")
5 plt.show()
```



```
1 pd.read_excel("data/20200426_excel_test.xlsx")
```

	자치구	세대수	인구수	남자인구수	여자인구수
0	종로구	73668	164640	80173	84467
1	중구	60130	134174	66064	68110
2	df_co = pd.read_excel("data/20200426_excel_test.xlsx", sheet_name=1)				
3	df_co				
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					
33					
34					
35					
36					
37					
38					
39					
40					
41					
42					
43					
44					
45					
46					
47					
48					
49					
50					
51					
52					
53					
54					
55					
56					
57					
58					
59					
60					
61					
62					
63					
64					
65					
66					
67					
68					
69					
70					
71					
72					
73					
74					
75					
76					
77					
78					
79					
80					
81					
82					
83					
84					
85					
86					
87					
88					
89					
90					
91					
92					
93					
94					
95					
96					
97					
98					
99					
100					
101					
102					
103					
104					
105					
106					
107					
108					
109					
110					
111					
112					
113					
114					
115					
116					
117					
118					
119					
120					
121					
122					
123					
124					
125					
126					
127					
128					
129					
130					
131					
132					
133					
134					
135					
136					
137					
138					
139					
140					
141					
142					
143					
144					
145					
146					
147					
148					
149					
150					
151					
152					
153					
154					
155					
156					
157					
158					
159					
160					
161					
162					
163					
164					
165					
166					
167					
168					
169					
170					
171					
172					
173					
174					
175					
176					
177					
178					
179					
180					
181					
182					
183					
184					
185					
186					
187					
188					
189					
190					
191					
192					
193					
194					
195					
196					
197					
198					
199					
200					
201					
202					
203					
204					
205					
206					
207					
208					
209					
210					
211					
212					
213					
214					
215					
216					
217					
218					
219					
220					
221					
222					
223					
224					
225					
226					
227					
228					
229					
230					
231					
232					
233					
234					
235					
236					
237					
238					
239					
240					
241					
242					
243					
244					
245					
246					
247					
248					
249					
250					
251					
252					
253					
254					
255					
256					
257					
258					
259					
260					
261					
262					
263					
264					
265					
266					
267					
268					
269					
270					
271					
272					
273					
274					
275					
276					
277					
278					
279					
280					
281					
282					
283					
284					
285					
286					
287					
288					
289					
290					
291					
292					
293					
294					
295					
296					
297					
298					
299					
300					
301					
302					
303					
304					
305					
306					
307					
308					
309					
310					
311					
312					
313					
314					
315					
316					
317					
318					
319					
320					
321					
322					
323					
324					
325					
326					
327					
328					
329					
330					
331					
332					
333					
334					
335					
336					
337	</				

	0	1	2	3	4	5	6	7	8	9
발전 소명	화천 수력	춘천 수력	의암 수력	청평 수력	팔당 수력	강 릉 수 력	칠보 수력	청평 양수	삼랑 진양 수	무주 양수
용량 (MW)	108.0	62.28	48.0	140.1	120.0	82.0	35.4	400.0	600.0	600.0
2017	17.62	18.46	28.47	19.37	22.65	0.0	18.0	7.62	9.2	11.83

## ▼ 변수명 변경, 행인덱스를 재설정

-

### ▼ 1. 변수명변경

```
1 # 변수명변경 : df_ru_1.rename(inplace=True)
2 df_ru_1.rename(columns=df_ru_1.iloc[0], inplace=True)
3 df_ru_1
```

	화천 수력	춘천 수력	의암 수력	청평 수력	팔당 수력	강 릉 수 력	칠보 수력	청평 양수	삼랑 진양 수	무주 양수
발전 소명	화천 수력	춘천 수력	의암 수력	청평 수력	팔당 수력	강 릉 수 력	칠보 수력	청평 양수	삼랑 진양 수	무주 양수
용량 (MW)	108.0	62.28	48.0	140.1	120.0	82.0	35.4	400.0	600.0	600.0
2017 년	17.62	18.46	28.47	19.37	22.65	0.0	18.0	7.62	9.2	11.83



```
1 # 0행 제거- 발전소명 : 데이터변경
2 # 변수명 = 변경내용
3 # 데이터프레임.drop(데이터프레임.index[행번호])
4 df_ru_1 = df_ru_1.drop(df_ru_1.index[0])
5 df_ru_1
```

	화천 수력	춘천 수력	의암 수력	청평 수력	팔당 수력	강 릉 수 력	칠보 수력	청평 양수	삼랑 진양 수	무주 양수
발전 소명	화천 수력	춘천 수력	의암 수력	청평 수력	팔당 수력	강 릉 수 력	칠보 수력	청평 양수	삼랑 진양 수	무주 양수
용량 (MW)	108.0	62.28	48.0	140.1	120.0	82.0	35.4	400.0	600.0	600.0
2017 년	17.62	18.46	28.47	19.37	22.65	0.0	18.0	7.62	9.2	11.83
2018 년	16.64	18.79	31.2	22.84	32.48	0.0	29.2	7.83	5.7	10.23



```
1 # 0행 제외 - 용량(MW)행 제외
2 df_ru_1 = df_ru_1.iloc[1:, :]
3 df_ru_1
```

	화천 수력	춘천 수력	의암 수력	청평 수력	팔당 수력	강 릉 수 력	칠보 수력	청 평 양 수	삼랑 진양 수	무주 양수	산· 양·
2017 년	17.62	18.46	28.47	19.37	22.65	0.0	18.0	7.62	9.2	11.83	9.5

## ▼ 2. 인덱스 재설정

1 # 인덱스 복귀 : 인덱스 0부터 시작  
 2 # 데이터내용변경 적용: 변수명=변경내용  
 3 df\_ru\_1 = df\_ru\_1.reset\_index()  
 4 df\_ru\_1

index	화천 수력	춘천 수력	의암 수력	청평 수력	팔당 수력	강 릉 수 력	칠보 수력	청 평 양 수	삼랑 진양 수	무주 양수	
0	2017 년	17.62	18.46	28.47	19.37	22.65	0.0	18.0	7.62	9.2	11.83
1	2018 년	16.64	18.79	31.2	22.84	32.48	0.0	29.2	7.83	5.7	10.23
2	2019	12.52	12.49	32.12	15.89	36.05	0.0	24.14	8.16	8.8	9.27

1 # 변수명 변경: columns={"원래이름": "새이름"}  
 2 df\_ru\_1.rename(columns={"index": "년도"}, inplace=True)  
 3 df\_ru\_1

년도	화천 수력	춘천 수력	의암 수력	청평 수력	팔당 수력	강 릉 수 력	칠보 수력	청 평 양 수	삼랑 진양 수	무주 양수	
0	2017 년	17.62	18.46	28.47	19.37	22.65	0.0	18.0	7.62	9.2	11.83
1	2018 년	16.64	18.79	31.2	22.84	32.48	0.0	29.2	7.83	5.7	10.23
2	2019	12.52	12.49	32.12	15.89	36.05	0.0	24.14	8.16	8.8	9.27

## ▼ 년도 변수를 날짜타입으로 변경, df\_ru\_1데이터프레임을 파일로 저장

### ▼ 1. 년도 변수를 날짜타입으로 변경

1 df\_ru\_1.dtypes

1 # 년도변수 내용의 공백제거 : str.replace(" ", "")  
 2 df\_ru\_1["년도"] = df\_ru\_1["년도"].str.replace(" ", "")

1 # 년도를 표기 변경: 2017-01-01로 표기  
 2 df\_ru\_1["년도"] = df\_ru\_1["년도"].str.replace("년", "-01-01")  
 3 df\_ru\_1["년도"]

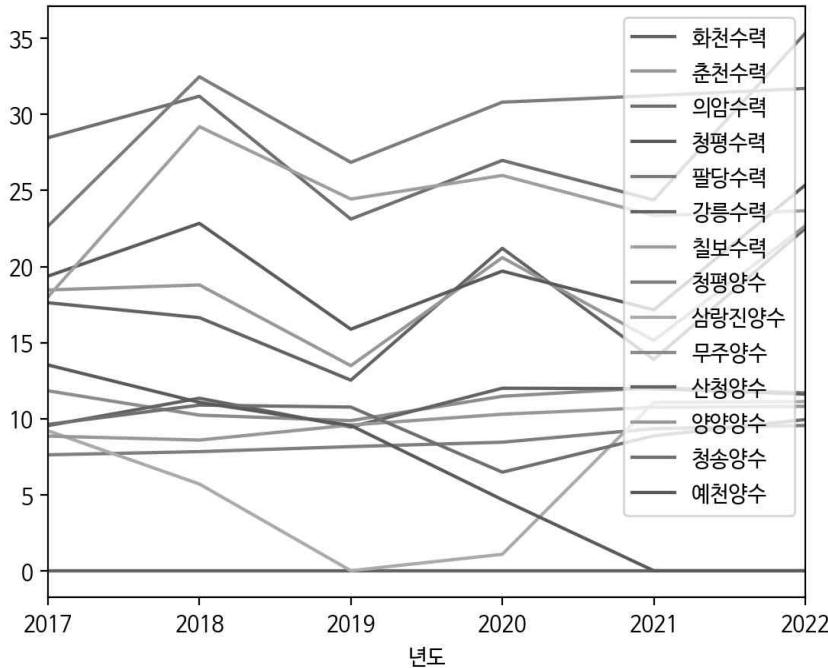
```
0    2017-01-01
1    2018-01-01
2    2019-01-01
3    2020-01-01
4    2021-01-01
```

5 2022-01-01  
Name: 년도, dtype: object

```
1 df_ru_1["년도"] = pd.to_datetime(df_ru_1["년도"])
2 df_ru_1["년도"]
```

```
1 df_ru_1.plot.line(x="년도")
```

<Axes: xlabel='년도'>



## ▼ 2. df\_ru\_1데이터프레임을 파일로 저장

-data/2017-2022\_한국수력원자력\_수력양수이용률\_transpose.csv로 저장

```
1 df_ru_1.to_csv("data/2017-2022_한국수력원자력_수력양수이용률_transpose.csv",
2 index=False)
```

```
1 df_ru_1 = pd.read_csv("data/2017-2022_한국수력원자력_수력양수이용률_transpose.csv")
2 df_ru_1
```

년도	화천수력	춘천수력	의암수력	청평수력	팔당수력	강릉수력	칠보수력	청평양수	삼랑진양수	무주양수
0 2017-01-01	17.62	18.46	28.47	19.37	22.65	0.0	18.00	7.62	9.20	11.83
1 2018-01-01	16.64	18.79	31.20	22.84	32.48	0.0	29.20	7.83	5.70	10.23
2 2019-01-01	12.52	12.12	22.12	15.00	26.25	0.0	21.44	8.16	8.00	8.87

```
1 df_ru_1.dtypes
```

```
1 df_ru_1["년도"] = pd.to_datetime(df_ru_1["년도"])
2 df_ru_1["년도"]
```

```
0 2017-01-01
1 2018-01-01
2 2019-01-01
3 2020-01-01
```

4 2021-01-01

5 2022-01-01

Name: 년도, dtype: datetime64[ns]

## 작업대상을 수력만으로 지정하고 산점도 행렬, 산점도 행렬에 회귀선 추가, 상관계수

### 행렬

#### A. 작업대상을 수력만으로 지정

```
1 # 모든 데이터값이 0인 강릉수력 변수 제외하고 df_ru_1에 저장
```

```
2 df_ru_1 = df_ru_1.iloc[:, df_ru_1.columns != "강릉수력"]
```

```
3 df_ru_1
```

년도	화천수력	춘천수력	의암수력	청평수력	팔당수력	칠보수력	청평양수	삼랑진양수	무주양수	산총양수
0 2017-01-01	17.62	18.46	28.47	19.37	22.65	18.00	7.62	9.20	11.83	9.54
1 2018-01-01	16.64	18.79	31.20	22.84	32.48	29.20	7.83	5.70	10.23	11.34
2 2019-01-01	12.53	13.48	23.12	15.88	26.85	24.44	8.00	8.27	8.44	8.44

```
1 # 작업대상을 수력만 - 년도와 수력이 포함 변수결
```

```
2 df_ru_11 = df_ru_1[df_ru_1.columns[0:7]]
```

```
3 df_ru_11
```

년도	화천수력	춘천수력	의암수력	청평수력	팔당수력	칠보수력
0 2017-01-01	17.62	18.46	28.47	19.37	22.65	18.00
1 2018-01-01	16.64	18.79	31.20	22.84	32.48	29.20
2 2019-01-01	12.53	13.48	23.12	15.88	26.85	24.44
3 2020-01-01	21.20	20.59	26.98	19.70	30.81	25.99

```
1 df_ru_11.to_csv("data/2017-2022_한국수력원자력_수력이용률.csv", index=False)
```

```
1 df_ru_11 = pd.read_csv("data/2017-2022_한국수력원자력_수력이용률.csv")
```

```
2 df_ru_11
```

년도	화천수력	춘천수력	의암수력	청평수력	팔당수력	칠보수력
0 2017-01-01	17.62	18.46	28.47	19.37	22.65	18.00
1 2018-01-01	16.64	18.79	31.20	22.84	32.48	29.20
2 2019-01-01	12.53	13.48	23.12	15.88	26.85	24.44
3 2020-01-01	21.20	20.59	26.98	19.70	30.81	25.99

```
1 df_ru_11.dtypes
```

```

년도      object
화천수력   float64
춘천수력   float64
의암수력   float64
청평수력   float64
팔당수력   float64
칠보수력   float64
dtype: object

```

```
1 df_ru_11.describe()
```

	화천수력	춘천수력	의암수력	청평수력	팔당수력	칠보수력
<b>count</b>	6.000000	6.000000	6.000000	6.000000	6.0000	6.000000
<b>mean</b>	17.398333	18.188333	28.248333	20.050000	29.2900	24.108333
<b>std</b>	3.929760	3.398526	4.514990	3.531685	3.7997	3.678442
<b>min</b>	12.530000	13.480000	23.120000	15.880000	22.6500	18.000000
<b>25%</b>	14.577500	15.970000	25.030000	17.705000	27.8400	23.430000
<b>50%</b>	17.130000	18.625000	27.725000	19.535000	31.0250	24.055000
<b>75%</b>	20.305000	20.140000	30.517500	22.055000	31.5925	25.602500
<b>max</b>	22.510000	22.670000	35.340000	25.360000	32.4800	29.200000

## ▼ B. 산점도 행렬, 산점도 행렬에 회귀선 추가, 상관계수 행렬

```

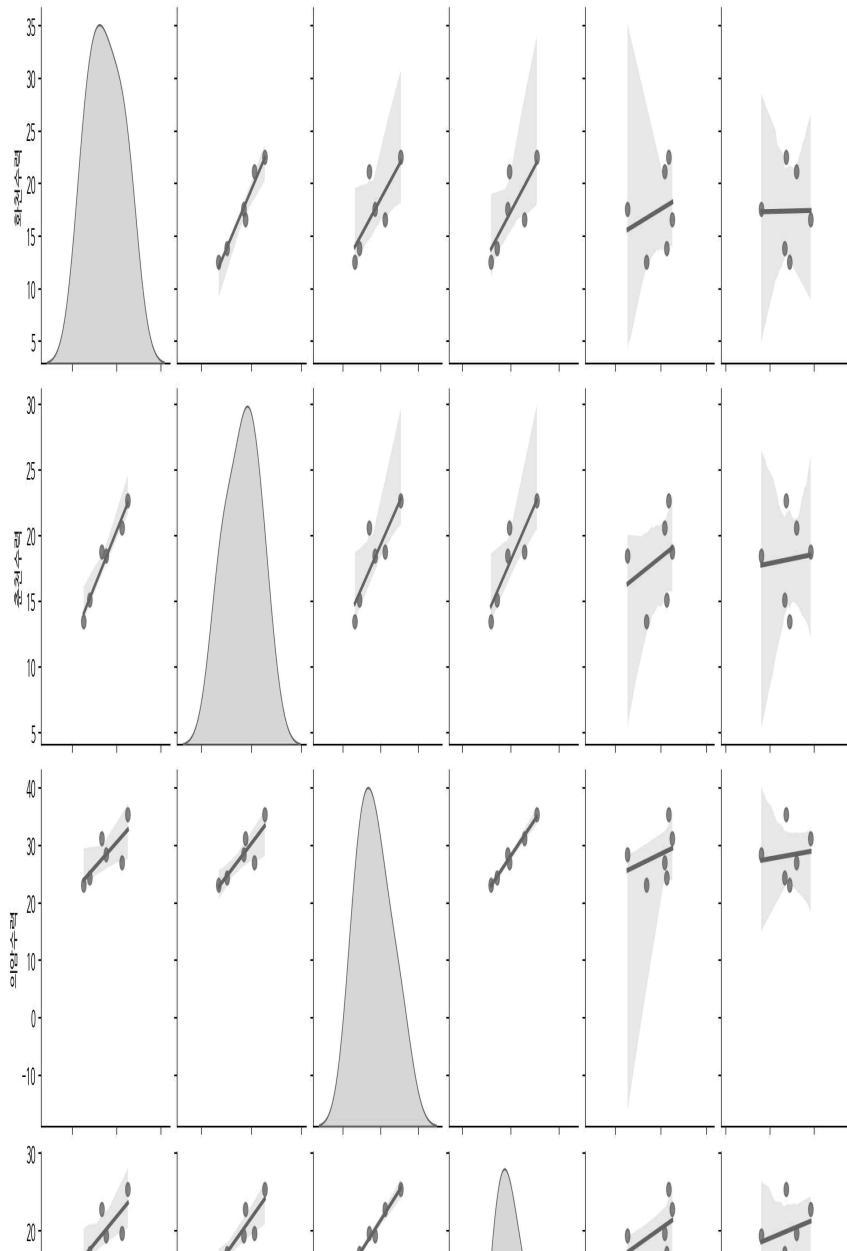
1 # 산점도행렬(산포행렬)
2 from pandas.plotting import scatter_matrix
3 scatter_matrix(df_ru_11)

```

```
array([[<Axes: xlabel='화천수력', ylabel='화천수력'>,
       <Axes: xlabel='춘천수력', ylabel='화천수력'>,
       <Axes: xlabel='의암수력', ylabel='화천수력'>,
       <Axes: xlabel='청평수력', ylabel='화천수력'>,
       <Axes: xlabel='팔당수력', ylabel='화천수력'>,
       <Axes: xlabel='칠보수력', ylabel='화천수력'>],
      [<Axes: xlabel='화천수력', ylabel='춘천수력'>,
       <Axes: xlabel='춘천수력', ylabel='춘천수력'>,
       <Axes: xlabel='의암수력', ylabel='춘천수력'>,
       <Axes: xlabel='청평수력', ylabel='춘천수력'>,
       <Axes: xlabel='팔당수력', ylabel='춘천수력'>,
       <Axes: xlabel='칠보수력', ylabel='춘천수력'>],
      [<Axes: xlabel='화천수력', ylabel='의암수력'>,
       <Axes: xlabel='춘천수력', ylabel='의암수력'>,
       <Axes: xlabel='의암수력', ylabel='의암수력'>,
       <Axes: xlabel='청평수력', ylabel='의암수력'>,
       <Axes: xlabel='팔당수력', ylabel='의암수력'>,
       <Axes: xlabel='칠보수력', ylabel='의암수력'>],
      [<Axes: xlabel='화천수력', ylabel='청평수력'>,
       <Axes: xlabel='춘천수력', ylabel='청평수력'>,
       <Axes: xlabel='의암수력', ylabel='청평수력'>,
       <Axes: xlabel='청평수력', ylabel='청평수력'>,
       <Axes: xlabel='팔당수력', ylabel='청평수력'>,
       <Axes: xlabel='칠보수력', ylabel='청평수력'>],
      [<Axes: xlabel='화천수력', ylabel='팔당수력'>,
       <Axes: xlabel='춘천수력', ylabel='팔당수력'>]]
```

```
1 # 산점도 행렬에 회귀선 추가 : seaborn라이브러리의 pairplot()사용
2 import seaborn as sns
3 sns.pairplot(df_ru_11,
4               diag_kind="kde", # 대각선이 kde(커널밀도추정)
5               kind="reg") # 산점도에 선형회귀선 추가
```

&lt;seaborn.axisgrid.PairGrid at 0x7dd478ca38b0&gt;



```

1 df_ru_11_corr = df_ru_11.iloc[:, 1: ].corr()
2 df_ru_11_corr

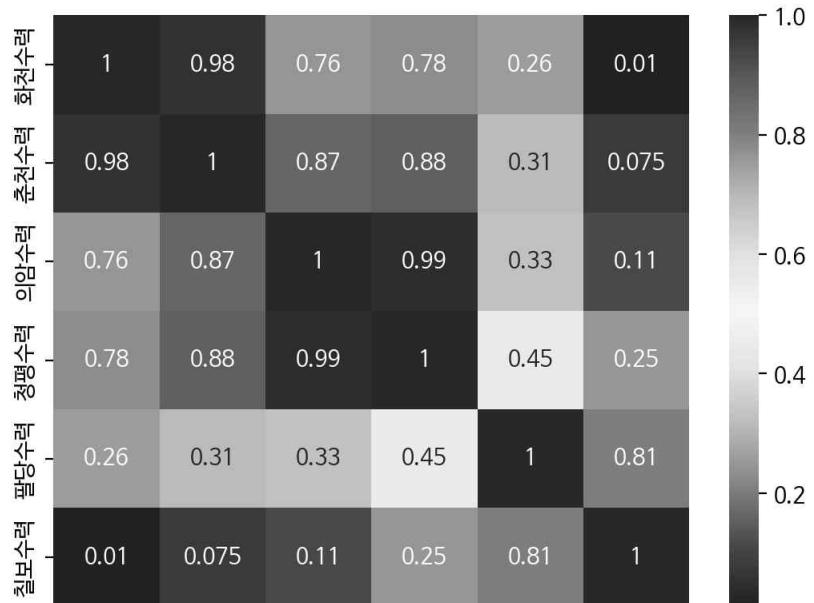
```

	화천수력	춘천수력	의암수력	청평수력	팔당수력	철보수력
화천 수력	1.000000	0.977274	0.763994	0.778475	0.258444	0.010287
춘천 수력	0.977274	1.000000	0.866906	0.881896	0.312168	0.075333
의암 수력	0.763994	0.866906	1.000000	0.988220	0.328147	0.113347
청평 수력	0.778475	0.881896	0.988220	1.000000	0.450891	0.249238

```

1 # 상관계수 행렬 히트맵
2 import seaborn as sns
3 # annot=True : 상관계수, cmap : 색상표
4 ax = sns.heatmap(df_ru_11_corr, annot=True, cmap="RdBu")
5 plt.savefig("plots/수력발전이용률_상관계수행렬_히트맵.png")
6 plt.show()

```



## ▼ 회귀분석 : 최소제곱법

- statsmodels.formula.api.ols("y ~ x", data=df)
 

```
model1 = ols("화천수력 ~ 춘천수력", data=df_ru_11)
result = model1.fit()
result.summary()
```
- statsmodels.api.OLS(y, X)
 

```
모델.fit()
```

```
1 import numpy as np
2 import pandas as pd
3 import statsmodels.api as sm
4 import statsmodels.formula.api as smf
5 from statsmodels.formula.api import ols
6 import matplotlib.pyplot as plt
7 import seaborn as sns
```

```
1 df_ru_11
```

→	년도	화천수력	춘천수력	의암수력	청평수력	팔당수력	칠보수력	☰
0	2017-01-01	17.62	18.46	28.47	19.37	22.65	18.00	☷
1	2018-01-01	16.64	18.79	31.20	22.84	32.48	29.20	☷
2	2019-01-01	12.53	13.48	23.12	15.88	26.85	24.44	☷
3	2020-01-01	21.20	20.59	26.98	19.70	30.81	25.99	☷
4	2021-01-01	13.89	15.14	24.38	17.15	31.24	23.35	☷
5	2022-01-01	22.51	22.67	35.34	25.36	31.71	23.67	☷

```
1 df_ru_11.columns
```

```
Index(['년도', '화천수력', '춘천수력', '의암수력', '청평수력', '팔당수력', '칠보수력'], dtype='object')
```

```
1 model1 = ols("화천수력 ~ 춘천수력", data=df_ru_11)
```

```
1 result = model1.fit()
```

```
1 result.summary()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/stats/stattools.py:74: ValueWarning: omni_normtest is not valid with less than 8 observations; %i "
warn("omni_normtest is not valid with less than 8 observations; %i "
      OLS Regression Results
Dep. Variable: 화천수력          R-squared:   0.955
Model: OLS                      Adj. R-squared:  0.944
Method: Least Squares           F-statistic:    85.02
Date: Thu, 23 Nov 2023 Prob (F-statistic): 0.000769
Time: 11:22:39                 Log-Likelihood: -6.8706
No. Observations: 6              AIC:        17.74
Df Residuals: 4                 BIC:        17.32
Df Model: 1
Covariance Type: nonrobust
            coef  std err      t      P>|t| [0.025  0.975]
Intercept -3.1551  2.261   -1.395  0.235 -9.434  3.123
춘천수력  1.1300  0.123   9.220  0.001  0.790  1.470
Omnibus:  nan     Durbin-Watson: 2.060
Prob(Omnibus): nan   Jarque-Bera (JB): 0.372
Skew:      -0.605   Prob(JB):    0.830
Kurtosis:  2.851    Cond. No.    110.
```

Notes:

```
1 df_ru_11["화천수력"].values
array([17.62, 16.64, 12.53, 21.2 , 13.89, 22.51])
```

```
1 X = df_ru_11["춘천수력"]
2 X = sm.add_constant(X)
3 y = df_ru_11["화천수력"].values
```

```
1 model = sm.OLS(y, X)
```

```
1 result = model.fit()
```

```
1 result.summary()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/stats/stattools.py:74: ValueWarning: omni_normtest is not valid with less than 8 observations; %i "
warn("omni_normtest is not valid with less than 8 observations; %i "
      OLS Regression Results
Dep. Variable: y          R-squared:   0.955
Model: OLS          Adj. R-squared:  0.944
Method: Least Squares   F-statistic:    85.02
Date: Thu, 23 Nov 2023 Prob (F-statistic): 0.000769
Time: 11:52:07          Log-Likelihood: -6.8706
No. Observations: 6          AIC:        17.74
Df Residuals: 4          BIC:        17.32
Df Model: 1
Covariance Type: nonrobust
            coef  std err      t      P>|t| [0.025  0.975]
const -3.1551  2.261   -1.395  0.235 -9.434  3.123
춘천수력  1.1300  0.123   9.220  0.001  0.790  1.470
Omnibus:  nan     Durbin-Watson: 2.060
Prob(Omnibus): nan   Jarque-Bera (JB): 0.372
Skew:      -0.605   Prob(JB):    0.830
Kurtosis:  2.851    Cond. No.    110.
```

Notes:

```

1 # 춘천수력의 이용률이 25일 경우 화천수력의 이용률 예측
2 X_new = np.array([[1, 25]])
3 result.predict(X_new)

array([25.09575436])

```

## ▼ 다중회귀분석

```
X = df_ru_11[["춘천수력", "의암수력", "청평수력"]]
```

```
X = sm.add_constant(X)
```

```
y = df_ru_11["화천수력"].values
```

```
model = sm.OLS(y, X)
```

```
result = model.fit()
```

```
result.summary()
```

```
X_new = np.array([[1, 25, 25, 25]])
```

```
result.predict(X_new)
```

```
1 X = df_ru_11[["춘천수력", "의암수력", "청평수력"]]
```

```
2 X = sm.add_constant(X)
```

```
3 y = df_ru_11["화천수력"].values
```

```
1 model = sm.OLS(y, X)
```

```
1 result = model.fit()
```

```
1 result.summary()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/stats/stattools.py:74: ValueWarning: omni_normtest is not valid with less than 8 observations; %i "
```

```
    OLS Regression Results
```

<b>Dep. Variable:</b>	y	<b>R-squared:</b>	0.986		
<b>Model:</b>	OLS	<b>Adj. R-squared:</b>	0.966		
<b>Method:</b>	Least Squares	<b>F-statistic:</b>	48.23		
<b>Date:</b>	Thu, 23 Nov 2023	<b>Prob (F-statistic):</b>	0.0204		
<b>Time:</b>	11:53:51	<b>Log-Likelihood:</b>	-3.2927		
<b>No. Observations:</b>	6	<b>AIC:</b>	14.59		
<b>Df Residuals:</b>	2	<b>BIC:</b>	13.75		
<b>Df Model:</b>	3				
<b>Covariance Type:</b> nonrobust					
	<b>coef</b>	<b>std err</b>	<b>t</b>	<b>P&gt; t </b>	[ <b>0.025</b> <b>0.975</b> ]
const	-1.8217	2.338	-0.779	0.517	-11.879 8.236
춘천수력	1.5133	0.203	7.457	0.018	0.640 2.386
의암수력	0.0264	0.471	0.056	0.960	-1.998 2.051
청평수력	-0.4515	0.636	-0.710	0.551	-3.188 2.285
<b>Omnibus:</b>	nan	<b>Durbin-Watson:</b>	1.563		
<b>Prob(Omnibus):</b>	nan	<b>Jarque-Bera (JB):</b>	0.593		
<b>Skew:</b>	0.104	<b>Prob(JB):</b>	0.743		
<b>Kurtosis:</b>	1.474	<b>Cond. No.</b>	318.		

Notes:

```

1 X_new = np.array([[1, 25, 25, 25]])
2 result.predict(X_new)

```

```
array([25.38593584])
```

## ▼ 병렬상자그림을 플롯해서 집단비교

1. 수력발전소를 1개의 변수로 생성

2. 1개의 변수내에서 집단비교

- 병렬상자그림 x: 범주형, y: 수량형

1

## ▼ 책 3장~부록까지 소스코드

```
1 import pandas as pd
2
3 ns_df = pd.read_csv('data/ns_202104.csv', low_memory=False)
4 ns_df.head()
```

```
1 import pandas as pd
2
3 ns_df = pd.read_csv('ns_202104.csv', low_memory=False)
4 ns_df.head()
```

1 ns\_df

```
1 ns_book = ns_df.loc[:, '번호':'등록일자']
2 # ns_book.head()
3 ns_book
```

1 print(ns\_df.columns)

1 print(ns\_df.columns[0])

1 ns\_df.columns != 'Unnamed: 13'

```
1 selected_columns = ns_df.columns != 'Unnamed: 13'
2 ns_book = ns_df.loc[:, selected_columns]
3
4 ns_book.head()
```

```
1 ns_book = ns_df.loc[:, ns_df.columns != 'Unnamed: 13']
2
3 ns_book
```

```
1 selected_columns = ns_df.columns != '부가기호'
2 ns_book = ns_df.loc[:, selected_columns]
3 ns_book.head()
```

```
1 ns_book = ns_df.drop('Unnamed: 13', axis=1)
2 ns_book.head()
```

```
1 ns_book = ns_df.drop(['부가기호', 'Unnamed: 13'], axis=1)
2 ns_book.head()
```

```
1 ns_book.drop('주제분류번호', axis=1, inplace=True)
2 ns_book.head()
```

```
1 ns_book = ns_df.dropna(axis=1)
2 ns_book.head()
```

```
1 ns_book = ns_df.dropna(axis=1, how='all')
2 ns_book.head()
```

## ▼ 행 삭제하기

```
1 ns_book2 = ns_book.drop([0,1])
2 ns_book2.head()
```

```
1 ns_book2 = ns_book[2:]
2 ns_book2.head()
```

```
1 ns_book2 = ns_book[0:2]
2 ns_book2.head()
```

```
1 selected_rows = ns_df['출판사'] == '한빛미디어'
2 ns_book2 = ns_book[selected_rows]
3 ns_book2.head()
```

```
1 ns_book2 = ns_book.loc[selected_rows]
2 ns_book2.head()
```

```
1 ns_book2 = ns_book[ns_book['대출건수'] > 1000]
2 ns_book2.head()
```

## ▼ 중복된 행 찾기

```
1 sum(ns_book.duplicated())
```

```
1 sum(ns_book.duplicated(subset=['도서명','저자','ISBN']))
```

```
1 dup_rows = ns_book.duplicated(subset=['도서명','저자','ISBN'], keep=False)
2 ns_book3 = ns_book[dup_rows]
3 ns_book3.head()
```

```
1 count_df = ns_book[['도서명','저자','ISBN','권','대출건수']]
```

```
1 group_df = count_df.groupby(by=['도서명','저자','ISBN','권'], dropna=False)
2 loan_count = group_df.sum()
```

```
1 loan_count = count_df.groupby(by=['도서명','저자','ISBN','권'], dropna=False).sum()
2 loan_count.head()
```

```
1 dup_rows = ns_book.duplicated(subset=['도서명','저자','ISBN','권'])
2 unique_rows = ~dup_rows
3 ns_book3 = ns_book[unique_rows].copy()
```

```
1 sum(ns_book3.duplicated(subset=['도서명','저자','ISBN','권']))
```

```
1 ns_book3.set_index(['도서명', '저자', 'ISBN', '권'], inplace=True)
2 ns_book3.head()
```

```
1 ns_book3.update(loan_count)
2 ns_book3.head()
```

```
1 ns_book4 = ns_book3.reset_index()
2 ns_book4.head()
```

```
1 sum(ns_book['대출건수']>100)
```

```
1 sum(ns_book4['대출건수']>100)
```

```
1 ns_book4 = ns_book4[ns_book.columns]
2 ns_book4.head()
```

```
1 ns_book4.to_csv('data/ns_book4.csv', index=False)
```

```
1 def data_cleaning(filename):
2     """
3     남산 도서관 장서 CSV 데이터 전처리 함수
4
5     :param filename: CSV 파일이름
6     """
7     # 파일을 데이터프레임으로 읽습니다.
8     ns_df = pd.read_csv(filename, low_memory=False)
9     # NaN인 열을 삭제합니다.
10    ns_book = ns_df.dropna(axis=1, how='all')
11
12    # 대출건수를 합치기 위해 필요한 행만 추출하여 count_df 데이터프레임을 만듭니다.
13    count_df = ns_book[['도서명', '저자', 'ISBN', '권', '대출건수']]
14    # 도서명, 저자, ISBN, 권을 기준으로 대출건수를 groupby합니다.
15    loan_count = count_df.groupby(by=['도서명', '저자', 'ISBN', '권'], dropna=False).sum()
16    # 원본 데이터프레임에서 중복된 행을 제외하고 고유한 행만 추출하여 복사합니다.
17    dup_rows = ns_book.duplicated(subset=['도서명', '저자', 'ISBN', '권'])
18    unique_rows = ~dup_rows
19    ns_book3 = ns_book[unique_rows].copy()
20    # 도서명, 저자, ISBN, 권을 인덱스로 설정합니다.
21    ns_book3.set_index(['도서명', '저자', 'ISBN', '권'], inplace=True)
22    # loan_count에 저장된 누적 대출건수를 업데이트합니다.
23    ns_book3.update(loan_count)
24
25    # 인덱스를 재설정합니다.
26    ns_book4 = ns_book3.reset_index()
27    # 원본 데이터프레임의 열 순서로 변경합니다.
28    ns_book4 = ns_book4[ns_book.columns]
29
30    return ns_book4
```

```
1 new_ns_book4 = data_cleaning('data/ns_202104.csv')
2
3 ns_book4.equals(new_ns_book4)
```

## ▼ 03-2 잘못된 데이터 수정하기



[주피터 노트북 뷰어로 보기](#)



[구글 코랩\(Colab\)에서 실행하기](#)

## ▼ 데이터프레임 정보 요약 확인하기

```

1 import pandas as pd
2
3 ns_book4 = pd.read_csv('data/ns_book4.csv', low_memory=False)
4 ns_book4.head()

1 import gdown
2
3 gdown.download('https://bit.ly/3GisL6J', 'ns_book4.csv', quiet=False)

1 import pandas as pd
2
3 ns_book4 = pd.read_csv('ns_book4.csv', low_memory=False)
4 ns_book4.head()

1 ns_book4.info()

1 ns_book4.info(memory_usage='deep')

```

## ▼ 누락된 값 처리하기

```

1 ns_book4.isna().sum()

1 ns_book4.loc[0, '도서권수'] = None
2 ns_book4['도서권수'].isna().sum()

1 ns_book4.head(2)

1 ns_book4.loc[0, '도서권수'] = 1
2 ns_book4 = ns_book4.astype({'도서권수':'int32', '대출건수': 'int32'})
3 ns_book4.head(2)

1 ns_book4.loc[0, '부가기호'] = None
2 ns_book4.head(2)

1 import numpy as np
2
3 ns_book4.loc[0, '부가기호'] = np.nan
4 ns_book4.head(2)

1 set_isbn_na_rows = ns_book4['세트 ISBN'].isna()
2 ns_book4.loc[set_isbn_na_rows, '세트 ISBN'] = ''
3
4 ns_book4['세트 ISBN'].isna().sum()

1 ns_book4.fillna('없음').isna().sum()

1 ns_book4['부가기호'].fillna('없음').isna().sum()

1 ns_book4.fillna({'부가기호':'없음'}).isna().sum()

1 ns_book4.replace(np.nan, '없음').isna().sum()

```

```

1 ns_book4.replace([np.nan, '2021'], ['없음', '21']).head(2)

1 ns_book4.replace({np.nan: '없음', '2021': '21'}).head(2)

1 ns_book4.replace({'부가기호': np.nan}, '없음').head(2)

1 ns_book4.replace({'부가기호': {np.nan: '없음'}, '발행년도': {'2021': '21'}}).head(2)

```

## ▼ 정규 표현식

```

1 ns_book4.replace({'발행년도': {'2021': '21'}})[100:102]

1 ns_book4.replace({'발행년도': {r'WdWd(WdWd)': r'W1'}}, regex=True)[100:102]

1 ns_book4.replace({'발행년도': {r'Wd{2}(Wd{2})': r'W1'}}, regex=True)[100:102]

1 ns_book4.replace({'저자': {r'(.*)WsW(지은이|W)(.*)WsW(옮긴이|W)': r'W1W2'},
2                     '발행년도': {r'Wd{2}(Wd{2})': r'W1'}}, regex=True)[100:102]

```

## ▼ 잘못된 값 바꾸기

```

1 # 아래 코드는 오류 발생
2 # ns_book4.astype({'발행년도': 'int32'})

1 ns_book4['발행년도'].str.contains('1988').sum()

1 invalid_number = ns_book4['발행년도'].str.contains('WD', na=True)
2 print(invalid_number.sum())
3 ns_book4[invalid_number].head()

1 ns_book5 = ns_book4.replace({'발행년도': r'.*(Wd{4}).*', r'W1', regex=True})
2 ns_book5[invalid_number].head()

1 unkown_year = ns_book5['발행년도'].str.contains('WD', na=True)
2 print(unkown_year.sum())
3 ns_book5[unkown_year].head()

1 ns_book5.loc[unkown_year, '발행년도'] = '-1'
2 ns_book5 = ns_book5.astype({'발행년도': 'int32'})

1 ns_book5['발행년도'].gt(4000).sum()

1 dangun_yy_rows = ns_book5['발행년도'].gt(4000)
2 ns_book5.loc[dangun_yy_rows, '발행년도'] = ns_book5.loc[dangun_yy_rows, '발행년도'] - 2333

1 dangun_year = ns_book5['발행년도'].gt(4000)
2 print(dangun_year.sum())
3 ns_book5[dangun_year].head(2)

1 ns_book5.loc[dangun_year, '발행년도'] = -1

```

```
1 old_books = ns_book5['발행년도'].gt(0) & ns_book5['발행년도'].lt(1900)
2 ns_book5[old_books]
```

```
1 ns_book5.loc[old_books, '발행년도'] = -1
```

```
1 ns_book5['발행년도'].eq(-1).sum()
```

## ▼ 누락된 정보 채우기

```
1 na_rows = ns_book5['도서명'].isna() | ns_book5['저자'].isna() | ns_book5['출판사'].isna() | ns_book5['발행년도'].eq(-1)
2 print(na_rows.sum())
3 ns_book5[na_rows].head(2)
```

```
1 # DH_KEY_TOO_SMALL 에러가 발생하는 경우 다음 코드의 주석을 제거하고 실행하세요.
2 # https://stackoverflow.com/questions/38015537/python-requests-exceptions-sslerror-dh-key-too-small
3 # import requests
4
5 # requests.packages.urllib3.util.ssl_.DEFAULT_CIPHERS += 'HIGH:!DH:!aNULL'
6 # try:
7 #     requests.packages.urllib3.contrib.pyopenssl.DEFAULT_SSL_CIPHER_LIST += 'HIGH:!DH:!aNULL'
8 # except AttributeError:
9 #     # no pyopenssl support used / needed / available
10 #    pass
```

```
1 import requests
2 from bs4 import BeautifulSoup
```

```
1 def get_book_title(isbn):
2     # Yes24 도서 검색 페이지 URL
3     url = 'http://www.yes24.com/Product/Search?domain=B00K&query={}'
4     # URL에 ISBN을 넣어 HTML 가져옵니다.
5     r = requests.get(url.format(isbn))
6     soup = BeautifulSoup(r.text, 'html.parser')    # HTML 파싱
7     # 클래스 이름이 'gd_name'인 a 태그의 텍스트를 가져옵니다.
8     title = soup.find('a', attrs={'class': 'gd_name'}) |
9             .get_text()
10    return title
```

```
1 get_book_title(9791191266054)
```

```

1 import re
2
3 def get_book_info(row):
4     title = row['도서명']
5     author = row['저자']
6     pub = row['출판사']
7     year = row['발행년도']
8     # Yes24 도서 검색 페이지 URL
9     url = 'http://www.yes24.com/Product/Search?domain=B00K&query={}'
10    # URL에 ISBN을 넣어 HTML 가져옵니다.
11    r = requests.get(url.format(row['ISBN']))
12    soup = BeautifulSoup(r.text, 'html.parser')    # HTML 파싱
13    try:
14        if pd.isna(title):
15            # 클래스 이름이 'gd_name'인 a 태그의 텍스트를 가져옵니다.
16            title = soup.find('a', attrs={'class':'gd_name'}) \
17                .get_text()
18    except AttributeError:
19        pass
20
21    try:
22        if pd.isna(author):
23            # 클래스 이름이 'info_auth'인 span 아래 a 태그의 텍스트를 가져옵니다.
24            authors = soup.find('span', attrs={'class':'info_auth'}) \
25                .find_all('a')
26            author_list = [auth.get_text() for auth in authors]
27            author = ','.join(author_list)
28    except AttributeError:
29        pass
30
31    try:
32        if pd.isna(pub):
33            # 클래스 이름이 'info_pub'인 span 아래 a 태그의 텍스트를 가져옵니다.
34            pub = soup.find('span', attrs={'class':'info_pub'}) \
35                .find('a') \
36                .get_text()
37    except AttributeError:
38        pass
39
40    try:
41        if year == -1:
42            # 클래스 이름이 'info_date'인 span 아래 텍스트를 가져옵니다.
43            year_str = soup.find('span', attrs={'class':'info_date'}) \
44                .get_text()
45            # 정규식으로 찾은 값 중에 첫 번째 것만 사용합니다.
46            year = re.findall(r'\d{4}', year_str)[0]
47    except AttributeError:
48        pass
49
50    return title, author, pub, year

```

```

1 updated_sample = ns_book5[na_rows].head(2).apply(get_book_info,
2      axis=1, result_type='expand')
3 updated_sample

```

아래 코드 셀은 실행하는데 시간이 오래 걸립니다. 편의를 위해 실행한 결과를 저장해 놓은 CSV 파일을 사용하겠습니다.

```

1 # ns_book5_update = ns_book5[na_rows].apply(get_book_info,
2 #      axis=1, result_type='expand')
3
4 # ns_book5_update.columns = ['도서명', '저자', '출판사', '발행년도']
5 # ns_book5_update.head()

```

```

1 import gdown
2 gdown.download('http://bit.ly/3UJZiHw', 'data/ns_book5_update.csv', quiet=False)
3
4 ns_book5_update = pd.read_csv('data/ns_book5_update.csv', index_col=0)
5 ns_book5_update.head()

1 ns_book5.update(ns_book5_update)
2
3 na_rows = ns_book5['도서명'].isna() | ns_book5['저자'].isna() |
4           | ns_book5['출판사'].isna() | ns_book5['발행년도'].eq(-1)
5 print(na_rows.sum())

1 ns_book5 = ns_book5.astype({'발행년도': 'int32'})

1 ns_book6 = ns_book5.dropna(subset=['도서명', '저자', '출판사'])
2 ns_book6 = ns_book6[ns_book6['발행년도'] != -1]
3 ns_book6.head()

1 ns_book6.to_csv('data/ns_book6.csv', index=False)

1 def data_fixing(ns_book4):
2     """
3         잘못된 값을 수정하거나 NaN 값을 채우는 함수
4
5         :param ns_book4: data_cleaning() 함수에서 전처리된 데이터프레임
6     """
7
8     # 도서권수와 대출건수를 int32로 바꿉니다.
9     ns_book4 = ns_book4.astype({'도서권수': 'int32', '대출건수': 'int32'})
10    # NaN인 세트 ISBN을 빈문자열로 바꿉니다.
11    set_isbn_na_rows = ns_book4['세트 ISBN'].isna()
12    ns_book4.loc[set_isbn_na_rows, '세트 ISBN'] = ''
13
14    # 발행년도 열에서 연도 네 자리를 추출하여 대체합니다. 나머지 발행년도는 -1로 바꿉니다.
15    ns_book5 = ns_book4.replace({'발행년도': '.*(\Wd{4}).*', r'\W1', regex=True})
16    unkown_year = ns_book5['발행년도'].str.contains('WD', na=True)
17    ns_book5.loc[unkown_year, '발행년도'] = '-1'
18
19    # 발행년도를 int32로 바꿉니다.
20    ns_book5 = ns_book5.astype({'발행년도': 'int32'})
21    # 4000년 이상인 경우 2333년을 뺍니다.
22    dangun_yy_rows = ns_book5['발행년도'].gt(4000)
23    ns_book5.loc[dangun_yy_rows, '발행년도'] = ns_book5.loc[dangun_yy_rows, '발행년도'] - 2333
24    # 여전히 4000년 이상인 경우 -1로 바꿉니다.
25    dangun_year = ns_book5['발행년도'].gt(4000)
26    ns_book5.loc[dangun_year, '발행년도'] = -1
27    # 0~1900년 사이의 발행년도는 -1로 바꿉니다.
28    old_books = ns_book5['발행년도'].gt(0) & ns_book5['발행년도'].lt(1900)
29    ns_book5.loc[old_books, '발행년도'] = -1
30
31    # 도서명, 저자, 출판사가 NaN이거나 발행년도가 -1인 행을 찾습니다.
32    na_rows = ns_book5['도서명'].isna() | ns_book5['저자'].isna() |
33           | ns_book5['출판사'].isna() | ns_book5['발행년도'].eq(-1)
34
35    # 교보문고 도서 상세 페이지에서 누락된 정보를 채웁니다.
36    updated_sample = ns_book5[na_rows].apply(get_book_info,
37                                         axis=1, result_type='expand')
38    updated_sample.columns = ['도서명', '저자', '출판사', '발행년도']
39    ns_book5.update(updated_sample)
40
41    # 도서명, 저자, 출판사가 NaN이거나 발행년도가 -1인 행을 삭제합니다.
42    ns_book6 = ns_book5.dropna(subset=['도서명', '저자', '출판사'])
43    ns_book6 = ns_book6[ns_book6['발행년도'] != -1]
44
45    return ns_book6

```

## 04장

### ▼ 04-1 통계로 요약하기


[주피터 노트북 뷰어로 보기](#)

[구글 콜랩\(Colab\)에서 실행하기](#)

### ▼ 기술통계 구하기

```

1 import pandas as pd
2
3 ns_book6 = pd.read_csv('data/ns_book6.csv', low_memory=False)
4 ns_book6.head()

1 import gdown
2
3 gdown.download('https://bit.ly/3736JW1', 'ns_book6.csv', quiet=False)

1 import pandas as pd
2
3 ns_book6 = pd.read_csv('ns_book6.csv', low_memory=False)
4 ns_book6.head()

1 ns_book6.describe()

1 sum(ns_book6['도서권수']==0)

1 ns_book7 = ns_book6[ns_book6['도서권수']>0]

1 ns_book7.describe(percentiles=[0.3, 0.6, 0.9])

1 ns_book7.describe(include='object')

```

### ▼ 평균

$$\text{평균} = \frac{a + b + c}{3}$$

$$\text{평균} = \frac{x_1 + x_2 + x_3}{3}$$

```

1 x = [10, 20, 30]
2 sum = 0
3 for i in range(3):
4     sum += x[i]
5 print("평균:", sum / len(x))

```

$$\text{평균} = \frac{x_1 + x_2 + x_3}{3} = \frac{\sum_{i=1}^3 x_i}{3}$$

$$\text{평균대출건수} = \frac{\sum_{i=1}^{376770} x_i}{376770}$$

```
1 ns_book7['대출건수'].mean()
```

## ▼ 중앙값

```
1 ns_book7['대출건수'].median()
```

```
1 temp_df = pd.DataFrame([1,2,3,4])
2 temp_df.median()
```

```
1 ns_book7['대출건수'].drop_duplicates().median()
```

## ▼ 최솟값, 최댓값

```
1 ns_book7['대출건수'].min()
```

```
1 ns_book7['대출건수'].max()
```

## ▼ 분위수

```
1 ns_book7['대출건수'].quantile(0.25)
```

```
1 ns_book7['대출건수'].quantile([0.25,0.5,0.75])
```

```
1 pd.Series([1,2,3,4,5]).quantile(0.9)
```

```
1 4 + (0.9-0.75)*(5-4)/(1.0-0.75)
```

```
1 pd.Series([1,2,3,4,5]).quantile(0.9, interpolation='midpoint')
```

```
1 pd.Series([1,2,3,4,5]).quantile(0.9, interpolation='nearest')
```

```
1 borrow_10_flag = ns_book7['대출건수'] < 10
```

```
1 borrow_10_flag.mean()
```

```
1 ns_book7['대출건수'].quantile(0.65)
```

## ▼ 분산

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}$$

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

```
1 ns_book7['대출건수'].var()
```

## ▼ 표준 편차

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

```
1 ns_book7['대출건수'].std()
```

$$\sqrt{4} = 2$$

```
1 import numpy as np
2
3 diff = ns_book7['대출건수'] - ns_book7['대출건수'].mean()
4
5 np.sqrt( np.sum(diff**2) / (len(ns_book7)-1) )
```

## ▼ 최빈값

```
1 ns_book7['도서명'].mode()
```

```
1 ns_book7['발행년도'].mode()
```

## ▼ 데이터프레임에서 기술통계 구하기

```
1 ns_book7.mean(numeric_only=True)
```

```
1 ns_book7.loc[:, '도서명'].mode()
```

```
1 ns_book7.to_csv('data/ns_book7.csv', index=False)
```

## ▼ 넘파이의 기술통계 함수

### ▼ 평균 구하기

```
1 import numpy as np
2
3 np.mean(ns_book7['대출건수'])
```

$$\frac{\text{국어점수} \times 2 + \text{수학점수}}{3}$$

$$\frac{\text{국어점수} \times \text{국어가중치} + \text{수학점수} \times \text{수학가중치}}{\text{국어가중치} + \text{수학가중치}}$$

$$\text{가중평균} = \frac{x_1 \times w_1 + x_2 \times w_2}{w_1 + w_2} = \frac{\sum_{i=1}^2 x_i \times w_i}{\sum_{i=1}^2 w_i}$$

```
1 np.average(ns_book7['대출건수'], weights=1/ns_book7['도서권수'])
```

```
1 np.mean(ns_book7['대출건수']/ns_book7['도서권수'])
```

```
1 ns_book7['대출건수'].sum()/ns_book7['도서권수'].sum()
```

#### ▼ 중앙값 구하기

```
1 np.median(ns_book7['대출건수'])
```

#### ▼ 최솟값, 최댓값 구하기

```
1 np.min(ns_book7['대출건수'])
```

```
1 np.max(ns_book7['대출건수'])
```

#### ▼ 분위수 구하기

```
1 # interpolation 매개변수가 numpy 1.22(python >= 3.8) 버전부터 method로 바뀜
2 np.quantile(ns_book7['대출건수'], [0.25, 0.5, 0.75])
```

#### ▼ 분산 구하기

```
1 np.var(ns_book7['대출건수'])
```

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}$$

```
1 ns_book7['대출건수'].var(ddof=0)
```

```
1 np.var(ns_book7['대출건수'], ddof=1)
```

#### ▼ 표준 편차 구하기

```
1 np.std(ns_book7['대출건수'])
```

#### ▼ 최빈값 구하기

```
1 values, counts = np.unique(ns_book7['도서명'], return_counts=True)
2 max_idx = np.argmax(counts)
3 values[max_idx]
```

#### ▼ 확인문제

## ▼ 4.

```
1 ns_book7[['출판사', '대출건수']].groupby('출판사').mean().sort_values('대출건수', ascending=False).head(10)
```

## ▼ 5.

```
1 target_range = np.array(ns_book7['대출건수'].quantile(q=[0.25,0.75]))
2 target_bool_idx = (ns_book7['대출건수'] >= target_range[0]) & (ns_book7['대출건수'] <= target_range[1])
3 target_bool_idx.sum()/len(ns_book7)*100
```

## ▼ 04-2 분포 요약하기


[주피터 노트북 뷰어로 보기](#)

[구글 콜랩\(Colab\)에서 실행하기](#)

## ▼ 산점도 그리기

```
1 import pandas as pd
2
3 ns_book7 = pd.read_csv('data/ns_book7.csv', low_memory=False)
4 ns_book7.head()
```

```
1 import gdown
2
3 gdown.download('https://bit.ly/3pK7iuu', 'ns_book7.csv', quiet=False)
```

```
1 import pandas as pd
2
3 ns_book7 = pd.read_csv('ns_book7.csv', low_memory=False)
4 ns_book7.head()
```

```
1 import matplotlib.pyplot as plt
2
3 plt.scatter([1,2,3,4], [1,2,3,4])
4 plt.show()
```

```
1 plt.scatter(ns_book7['번호'], ns_book7['대출건수'])
2 plt.show()
```

```
1 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'])
2 plt.show()
```

```
1 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
2 plt.show()
```

```
1 average_borrows = ns_book7['대출건수']/ns_book7['도서권수']
2 plt.scatter(average_borrows, ns_book7['대출건수'], alpha=0.1)
3 plt.show()
```

## ▼ 히스토그램 그리기

```
1 plt.hist([0,3,5,6,7,7,9,13], bins=5)
2 plt.show()
```

```
1 import numpy as np
2
3 np.histogram_bin_edges([0,3,5,6,7,7,9,13], bins=5)
```

```
1 np.random.seed(42)
2 random_samples = np.random.randn(1000)

1 print(np.mean(random_samples), np.std(random_samples))
```

```
1 plt.hist(random_samples)
2 plt.show()
```

```
1 plt.hist(ns_book7['대출건수'])
2 plt.show()
```

```
1 plt.hist(ns_book7['대출건수'])
2 plt.yscale('log')
3 plt.show()
```

```
1 plt.hist(ns_book7['대출건수'], log=True)
2 plt.show()
```

```
1 plt.hist(ns_book7['대출건수'], bins=100)
2 plt.yscale('log')
3 plt.show()
```

```
1 title_len = ns_book7['도서명'].apply(len)
2 plt.hist(title_len, bins=100)
3 plt.show()
```

```
1 plt.hist(title_len, bins=100)
2 plt.xscale('log')
3 plt.show()
```

## ▼ 상자 수염 그림 그리기

```
1 temp = ns_book7[['대출건수', '도서권수']]
```

```
1 plt.boxplot(temp)
2 plt.show()
```

```
1 plt.boxplot(ns_book7[['대출건수', '도서권수']])
2 plt.yscale('log')
3 plt.show()
```

```
1 plt.boxplot(ns_book7[['대출건수', '도서권수']], vert=False)
2 plt.xscale('log')
3 plt.show()
```

```
1 plt.boxplot(ns_book7[['대출건수', '도서권수']], whis=10)
2 plt.yscale('log')
3 plt.show()
```

```
1 plt.boxplot(ns_book7[['대출건수', '도서권수']], whis=(0,100))
2 plt.yscale('log')
3 plt.show()
```

## ▼ 판다스의 그래프 함수

### ▼ 산점도 그리기

```
1 ns_book7.plot.scatter('도서권수', '대출건수', alpha=0.1)
2 plt.show()
```

### ▼ 히스토그램 그리기

```
1 ns_book7['도서명'].apply(len).plot.hist(bins=100)
2 plt.show()
```

```
1 ns_book7['도서명'].apply(len).plot.hist(bins=100)
2 plt.show()
```

### ▼ 상자 수염 그림 그리기

```
1 ns_book7[['대출건수', '도서권수']].boxplot()
2 plt.yscale('log')
3 plt.show()
```

## ▼ 확인문제

### ▼ 4.

```
1 selected_rows = (1980 <= ns_book7['발행년도']) & (ns_book7['발행년도'] <= 2022)
2 plt.hist(ns_book7.loc[selected_rows, '발행년도'])
3 plt.show()
```

### ▼ 5.

```
1 plt.boxplot(ns_book7.loc[selected_rows, '발행년도'])
2 plt.show()
```

# 05장

## ▼ 05-1 맷플롯립 기본 요소 알아보기



[주피터 노트북 뷰어로 보기](#)



[구글 코랩\(Colab\)에서 실행하기](#)

## ▼ Figure 클래스

```

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('data/ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 import gdown
2
3 gdown.download('https://bit.ly/3pK7iuu', 'ns_book7.csv', quiet=False)

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 import matplotlib.pyplot as plt
2
3 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
4 plt.show()

1 print(plt.rcParams['figure.figsize'])

1 plt.figure(figsize=(9, 6))
2 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
3 plt.show()

1 print(plt.rcParams['figure.dpi'])

1 plt.figure(figsize=(900/72, 600/72))
2 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
3 plt.show()

1 %config InlineBackend.print_figure_kwarg = {'bbox_inches': None}
2 plt.figure(figsize=(900/72, 600/72))
3 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
4 plt.show()

1 %config InlineBackend.print_figure_kwarg = {'bbox_inches': 'tight'}

1 plt.figure(dpi=144)
2 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
3 plt.show()

```

## ▼ rcParams 객체

```

1 plt.rcParams['figure.dpi'] = 100

1 plt.rcParams['scatter.marker']

1 plt.rcParams['scatter.marker'] = '*'

1 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
2 plt.show()

```

```
1 plt.scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1, marker='+')
2 plt.show()
```

## ▼ 여러 개의 서브플롯 출력하기

```
1 fig, axs = plt.subplots(2)
2
3 axs[0].scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
4
5 axs[1].hist(ns_book7['대출건수'], bins=100)
6 axs[1].set_yscale('log')
7
8 fig.show()
```

```
1 fig, axs = plt.subplots(2, figsize=(6, 8))
2
3 axs[0].scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
4 axs[0].set_title('scatter plot')
5
6 axs[1].hist(ns_book7['대출건수'], bins=100)
7 axs[1].set_title('histogram')
8 axs[1].set_yscale('log')
9
10 fig.show()
```

```
1 fig, axs = plt.subplots(1, 2, figsize=(10, 4))
2
3 axs[0].scatter(ns_book7['도서권수'], ns_book7['대출건수'], alpha=0.1)
4 axs[0].set_title('scatter plot')
5 axs[0].set_xlabel('number of books')
6 axs[0].set_ylabel('borrow count')
7
8 axs[1].hist(ns_book7['대출건수'], bins=100)
9 axs[1].set_title('histogram')
10 axs[1].set_yscale('log')
11 axs[1].set_xlabel('borrow count')
12 axs[1].set_ylabel('frequency')
13
14 fig.show()
```

## ▼ 05-2 선, 막대 그래프 그리기



[주피터 노트북 뷰어로 보기](#)



[구글 코랩\(Colab\)에서 실행하기](#)

## ▼ 연도별 발행 도서 개수 구하기

```
1 import pandas as pd
2
3 ns_book7 = pd.read_csv('data/ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 import gdown
2
3 gdown.download('https://bit.ly/3pK7iuu', 'ns_book7.csv', quiet=False)
```

```

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('ns_book7.csv', low_memory=False)
4 ns_book7.head()

```

```

1 count_by_year = ns_book7['발행년도'].value_counts()
2 count_by_year

```

```

1 count_by_year = count_by_year.sort_index()
2 count_by_year

```

```

1 count_by_year = count_by_year[count_by_year.index <= 2030]
2 count_by_year

```

## ▼ 주제별 도서 개수 구하기

```

1 import numpy as np
2
3 def kdc_1st_char(no):
4     if no is np.nan:
5         return '-1'
6     else:
7         return no[0]
8
9 count_by_subject = ns_book7['주제분류번호'].apply(kdc_1st_char).value_counts()
10 count_by_subject

```

## ▼ 선 그래프 그리기

```

1 import matplotlib.pyplot as plt
2 plt.rcParams['figure.dpi'] = 100

```

```

1 plt.plot(count_by_year.index, count_by_year.values)
2 plt.title('Books by year')
3 plt.xlabel('year')
4 plt.ylabel('number of books')
5 plt.show()

```

```

1 plt.plot(count_by_year, marker='.', linestyle=':', color='red')
2 plt.title('Books by year')
3 plt.xlabel('year')
4 plt.ylabel('number of books')
5 plt.show()

```

```

1 plt.plot(count_by_year, '*-g')
2 plt.title('Books by year')
3 plt.xlabel('year')
4 plt.ylabel('number of books')
5 plt.show()

```

```

1 plt.plot(count_by_year, '*-g')
2 plt.title('Books by year')
3 plt.xlabel('year')
4 plt.ylabel('number of books')
5 plt.xticks(range(1947, 2030, 10)):
6 for idx, val in count_by_year[::-5].items():
7     plt.annotate(val, (idx, val))
8 plt.show()

```

```

1 plt.plot(count_by_year, '*-g')
2 plt.title('Books by year')
3 plt.xlabel('year')
4 plt.ylabel('number of books')
5 plt.xticks(range(1947, 2030, 10))
6 for idx, val in count_by_year[::5].items():
7     plt.annotate(val, (idx, val), xytext=(idx+1, val+10))
8 plt.show()

```

```

1 plt.plot(count_by_year, '*-g')
2 plt.title('Books by year')
3 plt.xlabel('year')
4 plt.ylabel('number of books')
5 plt.xticks(range(1947, 2030, 10))
6 for idx, val in count_by_year[::5].items():
7     plt.annotate(val, (idx, val), xytext=(2, 2), textcoords='offset points')
8 plt.show()

```

## ▼ 막대 그래프 그리기

```

1 plt.bar(count_by_subject.index, count_by_subject.values)
2 plt.title('Books by subject')
3 plt.xlabel('subject')
4 plt.ylabel('number of books')
5 for idx, val in count_by_subject.items():
6     plt.annotate(val, (idx, val), xytext=(0, 2), textcoords='offset points')
7 plt.show()

```

```

1 plt.bar(count_by_subject.index, count_by_subject.values, width=0.7, color='blue')
2 plt.title('Books by subject')
3 plt.xlabel('subject')
4 plt.ylabel('number of books')
5 for idx, val in count_by_subject.items():
6     plt.annotate(val, (idx, val), xytext=(0, 2), textcoords='offset points',
7                  fontsize=8, ha='center', color='green')
8 plt.show()

```

```

1 plt.bart(count_by_subject.index, count_by_subject.values, height=0.7, color='blue')
2 plt.title('Books by subject')
3 plt.xlabel('number of books')
4 plt.ylabel('subject')
5 for idx, val in count_by_subject.items():
6     plt.annotate(val, (val, idx), xytext=(2, 0), textcoords='offset points',
7                  fontsize=8, va='center', color='green')
8 plt.show()

```

## ▼ 이미지 출력하고 저장하기

```

1 # 노트북이 코랩에서 실행 중인지 체크합니다.
2 import sys
3 if 'google.colab' in sys.modules:
4     # 샘플 이미지를 다운로드합니다.
5     !wget https://bit.ly/3wrj4xf -O jupiter.png

```

```

1 img = plt.imread('imgs/jupiter.png')
2 img.shape

```

```
1 plt.imshow(img)
2 plt.show()
```

```
1 plt.figure(figsize=(8, 6))
2 plt.imshow(img)
3 plt.axis('off')
4 plt.show()
```

```
1 from PIL import Image
2
3 pil_img = Image.open('imgs/jupiter.png')
4 plt.figure(figsize=(8, 6))
5 plt.imshow(pil_img)
6 plt.axis('off')
7 plt.show()
```

```
1 import numpy as np
2
3 arr_img = np.array(pil_img)
4 arr_img.shape

1 plt.imsave('imgs/jupiter.jpg', arr_img)
```

## ▼ 그래프를 이미지로 저장하기

```
1 plt.rcParams['savefig.dpi']
```

```
1 plt.bart(count_by_subject.index, count_by_subject.values, height=0.7, color='blue')
2 plt.title('Books by subject')
3 plt.xlabel('number of books')
4 plt.ylabel('subject')
5 for idx, val in count_by_subject.items():
6     plt.annotate(val, (val, idx), xytext=(2, 0), textcoords='offset points',
7                  fontsize=8, va='center', color='green')
8 plt.savefig('plots/books_by_subject.png')
9 plt.show()
```

```
1 pil_img = Image.open('plots/books_by_subject.png')
2
3 plt.figure(figsize=(8, 6))
4 plt.imshow(pil_img)
5 plt.axis('off')
6 plt.show()
```

# 06장

## ▼ 06-1 객체지향 API로 그래프 꾸미기



[주피터 노트북 뷰어로 보기](#)



[구글 콜랩\(Colab\)에서 실행하기](#)

## ▼ pyplot 방식과 객체지향 API 방식

```

1 import matplotlib.pyplot as plt
2
3 plt.rcParams['figure.dpi'] = 100

1 plt.plot([1, 4, 9, 16])
2 plt.title('simple line graph')
3 plt.show()

```

```

1 fig, ax = plt.subplots()
2 ax.plot([1, 4, 9, 16])
3 ax.set_title('simple line graph')
4 fig.show()

```

## ▼ 그래프에 한글 출력하기

이 노트북은 맷플롯립 그래프에 한글을 쓰기 위해 나눔 폰트를 사용합니다. 코랩의 경우 다음 셀에서 나눔 폰트를 직접 설치합니다.

```

1 # 노트북이 코랩에서 실행 중인지 체크합니다.
2 import sys
3 if 'google.colab' in sys.modules:
4     !echo 'debconf debconf/frontend select Noninteractive' | debconf-set-selections
5     # 나눔 폰트를 설치합니다.
6     !sudo apt-get -qq -y install fonts-nanum
7     import matplotlib.font_manager as fm
8     font_files = fm.findSystemFonts(fontpaths=['/usr/share/fonts/truetype/nanum'])
9     for fpath in font_files:
10         fm.fontManager.addfont(fpath)

```

```
1 plt.rcParams['figure.dpi'] = 100
```

```
1 plt.rcParams['font.family']
```

```

1 # 나눔고딕 폰트를 사용합니다.
2 plt.rcParams['font.family'] = 'NanumGothic'

```

```

1 # 위와 동일하지만 이번에는 나눔바른고딕 폰트로 설정합니다.
2 plt.rc('font', family='NanumBarunGothic')

```

```
1 plt.rc('font', family='NanumBarunGothic', size=11)
```

```
1 print(plt.rcParams['font.family'], plt.rcParams['font.size'])
```

```

1 from matplotlib.font_manager import findSystemFonts
2 findSystemFonts()

```

```

1 plt.plot([1, 4, 9, 16])
2 plt.title('간단한 선 그래프')
3 plt.show()

```

```
1 plt.rc('font', size=10)
```

## ▼ 출판사별 발행 도서 개수 산점도 그리기

```

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('data/ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 import gdown
2
3 gdown.download('https://bit.ly/3pK7iuu', 'ns_book7.csv', quiet=False)

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 top30_pubs = ns_book7['출판사'].value_counts()[:30]
2 top30_pubs

1 top30_pubs_idx = ns_book7['출판사'].isin(top30_pubs.index)
2 top30_pubs_idx

1 top30_pubs_idx.sum()

1 ns_book8 = ns_book7[top30_pubs_idx].sample(1000, random_state=42)
2 ns_book8.head()

1 fig, ax = plt.subplots(figsize=(10, 8))
2 ax.scatter(ns_book8['발행년도'], ns_book8['출판사'])
3 ax.set_title('출판사별 발행도서')
4 fig.show()

1 plt.rcParams['lines.markersize']

1 fig, ax = plt.subplots(figsize=(10, 8))
2 ax.scatter(ns_book8['발행년도'], ns_book8['출판사'], s=ns_book8['대출건수'])
3 ax.set_title('출판사별 대출도서')
4 fig.show()

1 fig, ax = plt.subplots(figsize=(10, 8))
2 ax.scatter(ns_book8['발행년도'], ns_book8['출판사'],
3            linewidths=0.5, edgecolors='k', alpha=0.3,
4            s=ns_book8['대출건수']*2, c=ns_book8['대출건수'])
5 ax.set_title('출판사별 대출도서')
6 fig.show()

1 fig, ax = plt.subplots(figsize=(10, 8))
2 sc = ax.scatter(ns_book8['발행년도'], ns_book8['출판사'],
3                 linewidths=0.5, edgecolors='k', alpha=0.3,
4                 s=ns_book8['대출건수']**1.3, c=ns_book8['대출건수'], cmap='jet')
5 ax.set_title('출판사별 대출도서')
6 fig.colorbar(sc)
7 fig.show()

```

## ▼ 06-2 맷플롯립의 고급 기능 배우기



[주피터 노트북 뷰어로 보기](#)



[구글 콜랩\(Colab\)에서 실행하기](#)

## ▼ 실습 준비하기

이 노트북은 맷플롯립 그래프에 한글을 쓰기 위해 나눔 폰트를 사용합니다. 코랩의 경우 다음 셀에서 나눔 폰트를 직접 설치합니다.

```

1 # 노트북이 코랩에서 실행 중인지 체크합니다.
2 import sys
3 if 'google.colab' in sys.modules:
4     !echo 'debconf debconf/frontend select Noninteractive' | debconf-set-selections
5     # 나눔 폰트를 설치합니다.
6     !sudo apt-get -qq -y install fonts-nanum
7     import matplotlib.font_manager as fm
8     font_files = fm.findSystemFonts(fontpaths=['/usr/share/fonts/truetype/nanum'])
9     for fpath in font_files:
10         fm.fontManager.addfont(fpath)

1 import matplotlib.pyplot as plt
2
3 # 나눔바른고딕 폰트로 설정합니다.
4 plt.rc('font', family='NanumBarunGothic')
5
6 # 그래프 DPI 기본값을 변경합니다.
7 plt.rcParams['figure.dpi'] = 100

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('data/ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 import gdown
2
3 gdown.download('https://bit.ly/3pK7iuu', 'ns_book7.csv', quiet=False)

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('ns_book7.csv', low_memory=False)
4 ns_book7.head()

```

## ▼ 하나의 피겨에 여러 개의 선 그래프 그리기

```

1 top30_pubs = ns_book7['출판사'].value_counts()[:30]
2 top30_pubs_idx = ns_book7['출판사'].isin(top30_pubs.index)

1 ns_book9 = ns_book7[top30_pubs_idx][['출판사', '발행년도', '대출건수']]
2 ns_book9 = ns_book9.groupby(by=['출판사', '발행년도']).sum()

1 ns_book9 = ns_book9.reset_index()
2 ns_book9[ns_book9['출판사'] == '황금가지'].head()

1 line1 = ns_book9[ns_book9['출판사'] == '황금가지']
2 line2 = ns_book9[ns_book9['출판사'] == '비룡소']

```

```

1 fig, ax = plt.subplots(figsize=(8, 6))
2 ax.plot(line1['발행년도'], line1['대출건수'])
3 ax.plot(line2['발행년도'], line2['대출건수'])
4 ax.set_title('년도별 대출건수')
5 fig.show()

1 fig, ax = plt.subplots(figsize=(8, 6))
2 ax.plot(line1['발행년도'], line1['대출건수'], label='황금가지')
3 ax.plot(line2['발행년도'], line2['대출건수'], label='비룡소')
4 ax.set_title('년도별 대출건수')
5 ax.legend()
6 fig.show()

1 fig, ax = plt.subplots(figsize=(8, 6))
2 for pub in top30_pubs.index[:5]:
3     line = ns_book9[ns_book9['출판사'] == pub]
4     ax.plot(line['발행년도'], line['대출건수'], label=pub)
5 ax.set_title('년도별 대출건수')
6 ax.legend()
7 ax.set_xlim(1985, 2025)
8 fig.show()

```

```

1 ns_book10 = ns_book9.pivot_table(index='출판사', columns='발행년도')
2 ns_book10.head()

```

```
1 ns_book10.columns[:10]
```

```

1 top10_pubs = top30_pubs.index[:10]
2 year_cols = ns_book10.columns.get_level_values(1)

1 fig, ax = plt.subplots(figsize=(8, 6))
2 ax.stackplot(year_cols, ns_book10.loc[top10_pubs].fillna(0), labels=top10_pubs)
3 ax.set_title('년도별 대출건수')
4 ax.legend(loc='upper left')
5 ax.set_xlim(1985, 2025)
6 fig.show()

```

## ▼ 하나의 피겨에 여러 개의 막대 그래프 그리기

```

1 fig, ax = plt.subplots(figsize=(8, 6))
2 ax.bar(line1['발행년도'], line1['대출건수'], label='황금가지')
3 ax.bar(line2['발행년도'], line2['대출건수'], label='비룡소')
4 ax.set_title('년도별 대출건수')
5 ax.legend()
6 fig.show()

1 fig, ax = plt.subplots(figsize=(8, 6))
2 ax.bar(line1['발행년도']-0.2, line1['대출건수'], width=0.4, label='황금가지')
3 ax.bar(line2['발행년도']+0.2, line2['대출건수'], width=0.4, label='비룡소')
4 ax.set_title('년도별 대출건수')
5 ax.legend()
6 fig.show()

```

```

1 height1 = [5, 4, 7, 9, 8]
2 height2 = [3, 2, 4, 1, 2]
3
4 plt.bar(range(5), height1, width=0.5)
5 plt.bar(range(5), height2, bottom=height1, width=0.5)
6 plt.show()

```

```

1 height3 = [a + b for a, b in zip(height1, height2)]
2
3 plt.bar(range(5), height3, width=0.5)
4 plt.bar(range(5), height1, width=0.5)
5 plt.show()

```

## ▼ 데이터값 누적하여 그리기

```

1 ns_book10.loc[top10_pubs[:5], ('대출건수',2013):('대출건수',2020)]
2
1 ns_book10.loc[top10_pubs[:5], ('대출건수',2013):('대출건수',2020)].cumsum()
2
1 ns_book12 = ns_book10.loc[top10_pubs].cumsum()
2
1 fig, ax = plt.subplots(figsize=(8, 6))
2 for i in reversed(range(len(ns_book12))):
3     bar = ns_book12.iloc[i]      # 행 추출
4     label = ns_book12.index[i]  # 출판사 이름 추출
5     ax.bar(year_cols, bar, label=label)
6 ax.set_title('년도별 대출건수')
7 ax.legend(loc='upper left')
8 ax.set_xlim(1985, 2025)
9 fig.show()

```

## ▼ 원 그래프 그리기

```

1 data = top30_pubs[:10]
2 labels = top30_pubs.index[:10]
3
1 fig, ax = plt.subplots(figsize=(8, 6))
2 ax.pie(data, labels=labels)
3 ax.set_title('출판사 도서비율')
4 fig.show()
5
1 plt.pie([10,9], labels=['A제품', 'B제품'], startangle=90)
2 plt.title('제품의 매출비율')
3 plt.show()
4
1 fig, ax = plt.subplots(figsize=(8, 6))
2 ax.pie(data, labels=labels, startangle=90,
3         autopct='%.1f%%', explode=[0.1]+[0]*9)
4 ax.set_title('출판사 도서비율')
5 fig.show()

```

## ▼ 여러 종류의 그래프가 있는 서브플롯 그리기

```

1 fig, axes = plt.subplots(2, 2, figsize=(20, 16))
2
3 # 산점도
4 ns_book8 = ns_book7[top30_pubs_idx].sample(1000, random_state=42)
5 sc = axes[0, 0].scatter(ns_book8['발행년도'], ns_book8['출판사'],
6                         linewidths=0.5, edgecolors='k', alpha=0.3,
7                         s=ns_book8['대출건수'], c=ns_book8['대출건수'], cmap='jet')
8 axes[0, 0].set_title('출판사별 발행도서')
9 fig.colorbar(sc, ax=axes[0, 0])
10
11 # 스택 선 그래프
12 axes[0, 1].stackplot(year_cols, ns_book10.loc[top10_pubs].fillna(0),
13                       labels=top10_pubs)
14 axes[0, 1].set_title('년도별 대출건수')
15 axes[0, 1].legend(loc='upper left')
16 axes[0, 1].set_xlim(1985, 2025)
17
18 # 스택 막대 그래프
19 for i in reversed(range(len(ns_book12))):
20     bar = ns_book12.iloc[i]      # 행 추출
21     label = ns_book12.index[i]  # 출판사 이름 추출
22     axes[1, 0].bar(year_cols, bar, label=label)
23 axes[1, 0].set_title('년도별 대출건수')
24 axes[1, 0].legend(loc='upper left')
25 axes[1, 0].set_xlim(1985, 2025)
26
27 # 원 그래프
28 axes[1, 1].pie(data, labels=labels, startangle=90,
29                  autopct='%.1f%%', explode=[0.1]+[0]*9)
30 axes[1, 1].set_title('출판사 도서비율')
31
32 fig.savefig('plots/all_in_one.png')
33 fig.show()

```

## ▼ 판다스로 여러 개의 그래프 그리기

### ▼ 스택 영역 그래프 그리기

```

1 ns_book11 = ns_book9.pivot_table(index='발행년도', columns='출판사', values='대출건수')
2 ns_book11.loc[2000:2005]

```

```

1 import numpy as np
2
3 ns_book11 = ns_book7[top30_pubs_idx].pivot_table(
4     index='발행년도', columns='출판사',
5     values='대출건수', aggfunc=np.sum)
6 ns_book11.loc[2000:2005]

1 fig, ax = plt.subplots(figsize=(8, 6))
2 ns_book11[top10_pubs].plot.area(ax=ax, title='년도별 대출건수',
3                                 xlim=(1985, 2025))
4 ax.legend(loc='upper left')
5 fig.show()

```

### ▼ 스택 막대 그래프 그리기

```

1 fig, ax = plt.subplots(figsize=(8, 6))
2 ns_book11.loc[1985:2025, top10_pubs].plot.bar(
3     ax=ax, title='년도별 대출건수', stacked=True, width=0.8)
4 ax.legend(loc='upper left')
5 fig.show()

```

## 07장

### ▼ 07-1 통계적으로 추론하기



[주피터 노트북 뷰어로 보기](#)



[구글 코랩\(Colab\)에서 실행하기](#)

### ▼ 표준 점수 구하기

```

1 import numpy as np
2
3 x = [0, 3, 5, 7, 10]
4
5 s = np.std(x)
6 m = np.mean(x)
7 z = (7 - m) / s
8 print(z)

1 from scipy import stats
2
3 stats.zscore(x)

1 stats.norm.cdf(0)

1 stats.norm.cdf(1.0) - stats.norm.cdf(-1.0)

1 stats.norm.cdf(2.0) - stats.norm.cdf(-2.0)

1 stats.norm.ppf(0.9)

```

### ▼ 중심극한정리 알아보기

```

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('data/ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 import gdown
2
3 gdown.download('https://bit.ly/3pK7iuu', 'ns_book7.csv', quiet=False)

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('ns_book7.csv', low_memory=False)
4 ns_book7.head()

```

```

1 import matplotlib.pyplot as plt
2
3 plt.hist(ns_book7['대출건수'], bins=50)
4 plt.yscale('log')
5 plt.show()

1 np.random.seed(42)
2 sample_means = []
3 for _ in range(1000):
4     m = ns_book7['대출건수'].sample(30).mean()
5     sample_means.append(m)

1 plt.hist(sample_means, bins=30)
2 plt.show()

```

```

1 np.mean(sample_means)

1 ns_book7['대출건수'].mean()

1 np.random.seed(42)
2 sample_means = []
3 for _ in range(1000):
4     m = ns_book7['대출건수'].sample(20).mean()
5     sample_means.append(m)
6 np.mean(sample_means)

```

```

1 np.random.seed(42)
2 sample_means = []
3 for _ in range(1000):
4     m = ns_book7['대출건수'].sample(40).mean()
5     sample_means.append(m)
6 np.mean(sample_means)

```

```

1 np.std(sample_means)

1 np.std(ns_book7['대출건수']) / np.sqrt(40)

```

## ▼ 모집단의 평균 범위 추정하기: 신뢰구간

```

1 python_books_index = ns_book7['주제분류번호'].str.startswith('00') & \
2                 ns_book7['도서명'].str.contains('파이썬')
3 python_books = ns_book7[python_books_index]
4 python_books.head()

```

```

1 len(python_books)

1 python_mean = np.mean(python_books['대출건수'])
2 python_mean

```

```

1 python_std = np.std(python_books['대출건수'])
2 python_se = python_std / np.sqrt(len(python_books))
3 python_se

```

```

1 stats.norm.ppf(0.975)

1 stats.norm.ppf(0.025)

```

```
1 print(python_mean-1.96*python_se, python_mean+1.96*python_se)
```

## ▼ 통계적 의미 확인하기: 가설검정

```
1 cplus_books_index = ns_book7['주제분류번호'].str.startswith('00') & #  
2           ns_book7['도서명'].str.contains('C++', regex=False)  
3 cplus_books = ns_book7[cplus_books_index]  
4 cplus_books.head()
```

```
1 len(cplus_books)
```

```
1 cplus_mean = np.mean(cplus_books['대출건수'])  
2 cplus_mean
```

```
1 cplus_se = np.std(cplus_books['대출건수'])/ np.sqrt(len(cplus_books))  
2 cplus_se
```

```
1 (python_mean - cplus_mean) / np.sqrt(python_se**2 + cplus_se**2)
```

```
1 stats.norm.cdf(2.50)
```

```
1 p_value = (1-0.995)*2  
2 p_value
```

```
1 t, pvalue = stats.ttest_ind(python_books['대출건수'], cplus_books['대출건수'])  
2 print(t, pvalue)
```

## ▼ 정규분포가 아닐 때 가설 검증하기: 순열검정

```
1 def statistic(x, y):  
2     return np.mean(x) - np.mean(y)  
  
1 def permutation_test(x, y):  
2     # 표본의 평균 차이를 계산합니다.  
3     obs_diff = statistic(x, y)  
4     # 두 표본을 합칩니다.  
5     all = np.append(x, y)  
6     diffs = []  
7     np.random.seed(42)  
8     # 순열 검정을 1000번 반복합니다.  
9     for _ in range(1000):  
10         # 전체 인덱스를 섞습니다.  
11         idx = np.random.permutation(len(all))  
12         # 랜덤하게 두 그룹으로 나눈 다음 평균 차이를 계산합니다.  
13         x_ = all[idx[:len(x)]]  
14         y_ = all[idx[len(x):]]  
15         diffs.append(statistic(x_, y_))  
16     # 원본 표본보다 작거나 큰 경우의 p-값을 계산합니다.  
17     less_pvalue = np.sum(diffs < obs_diff)/1000  
18     greater_pvalue = np.sum(diffs > obs_diff)/1000  
19     # 둘 중 작은 p-값을 선택해 2를 곱하여 최종 p-값을 반환합니다.  
20     return obs_diff, np.minimum(less_pvalue, greater_pvalue) * 2
```

```
1 permutation_test(python_books['대출건수'], cplus_books['대출건수'])
```

```

1 # scipy 1.8 버전 이상에서만 실행됩니다.
2 # res = stats.permutation_test((python_books['대출건수'], cplus_books['대출건수']),
3 #                               statistic, random_state=42)
4 # 결과는 약 3.153 0.0258입니다.
5 # print(res.statistic, res.pvalue)

1 java_books_idx = ns_book7['주제분류번호'].str.startswith('00') & \
2     ns_book7['도서명'].str.contains('자바스크립트')
3 java_books = ns_book7[java_books_idx]
4 java_books.head()

1 print(len(java_books), np.mean(java_books['대출건수']))

```

```
1 permutation_test(python_books['대출건수'], java_books['대출건수'])
```

## ▼ 07-2 머신러닝으로 예측하기



[주피터 노트북 뷰어로 보기](#)



[구글 코랩\(Colab\)에서 실행하기](#)

## ▼ 모델 훈련하기

```

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('data/ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 import gdown
2
3 gdown.download('https://bit.ly/3pK7iuu', 'ns_book7.csv', quiet=False)

1 import pandas as pd
2
3 ns_book7 = pd.read_csv('ns_book7.csv', low_memory=False)
4 ns_book7.head()

1 from sklearn.model_selection import train_test_split
2
3 train_set, test_set = train_test_split(ns_book7, random_state=42)

1 print(len(train_set), len(test_set))

1 X_train = train_set[['도서권수']]
2 y_train = train_set['대출건수']
3
4 print(X_train.shape, y_train.shape)

1 from sklearn.linear_model import LinearRegression
2
3 lr = LinearRegression()
4 lr.fit(X_train, y_train)

```

## ▼ 훈련된 모델을 평가하기: 결정계수

```

1 X_test = test_set[['도서권수']]
2 y_test = test_set['대출건수']
3
4 lr.score(X_test, y_test)

```

```

1 lr.fit(y_train.to_frame(), y_train)
2 lr.score(y_test.to_frame(), y_test)

```

## ▼ 연속적인 값 예측하기: 선형 회귀

```
1 print(lr.coef_, lr.intercept_)
```

## ▼ 카테고리 예측하기: 로지스틱 회귀

```

1 borrow_mean = ns_book7['대출건수'].mean()
2 y_train_c = y_train > borrow_mean
3 y_test_c = y_test > borrow_mean

```

```

1 from sklearn.linear_model import LogisticRegression
2
3 logr = LogisticRegression()
4 logr.fit(X_train, y_train_c)
5 logr.score(X_test, y_test_c)

```

```
1 y_test_c.value_counts()
```

```

1 from sklearn.dummy import DummyClassifier
2
3 dc = DummyClassifier()
4 dc.fit(X_train, y_train_c)
5 dc.score(X_test, y_test_c)

```

## ▼ 평균제곱오차와 평균절댓값오차로 모델 평가하기

```
1 lr.fit(X_train, y_train)
```

```
1 y_pred = lr.predict(X_test)
```

```

1 from sklearn.metrics import mean_absolute_error
2
3 mean_absolute_error(y_test, y_pred)

```

```
1 y_test.mean()
```

## Appendix

### ▼ 데이터베이스에서 가져오기



[주피터 노트북 뷰어로 보기](#)



[구글 콜랩\(Colab\)에서 실행하기](#)

## ▼ 파일에서 SQL 사용하기: SQLite

1 # 최신 sqlalchemy는 판다스에서 에러를 일으킵니다. 1.4.\* 버전을 사용해 주세요. (<https://github.com/pandas-dev/pandas/issues/27010>)  
 2 !pip install -U sqlalchemy==1.4.46

```
1 import sqlite3

1 conn = sqlite3.connect('ns_lib.db')

1 import gdown
2
3 gdown.download('https://bit.ly/3RhoNho', 'ns_202104.csv', quiet=False)
```

**이전에 만든 nslib\_book 테이블이 있다면 먼저 먼저 삭제해 주세요.**

```
1 c = conn.cursor()
2
3 c.execute("CREATE TABLE nslib_book \
4           (name TEXT, author TEXT, borrow_count INTEGER)")

1 c.execute("CREATE TABLE IF NOT EXISTS nslib_book \
2           (name TEXT, author TEXT, borrow_count INTEGER)")

1 c.execute("DROP TABLE nslib_book")

1 c.execute("CREATE TABLE nslib_book \
2           (name TEXT, author TEXT, borrow_count INTEGER)")
```

## ▼ 데이터프레임 데이터를 테이블에 추가하기

```
1 import pandas as pd
2
3 ns_df = pd.read_csv('data/ns_202104.csv', low_memory=False)
4 ns_df.head()

1 import pandas as pd
2
3 ns_df = pd.read_csv('ns_202104.csv', low_memory=False)
4 ns_df.head()

1 for index, row in ns_df.iterrows():
2     c.execute("INSERT INTO nslib_book (name,author,borrow_count) \
3               VALUES (?, ?, ?)", (row['도서명'], row['저자'], row['대출건수']))

1 for index, row in ns_df.iterrows():
2     pass

1 book_df = ns_df[['도서명', '저자', '대출건수']]
2 book_df.head()

1 book_df.columns = ['name', 'author', 'borrow_count']
2 book_df.head()
```