빅 데이터 분석을 위한 알고리즘 활용

빅 데이터분석을 위한 알고리즘 소개 및 Python을 활용한 실습

SUNGHEE KUK

DATA Analysis

2025
프로그래밍

CONTENTS

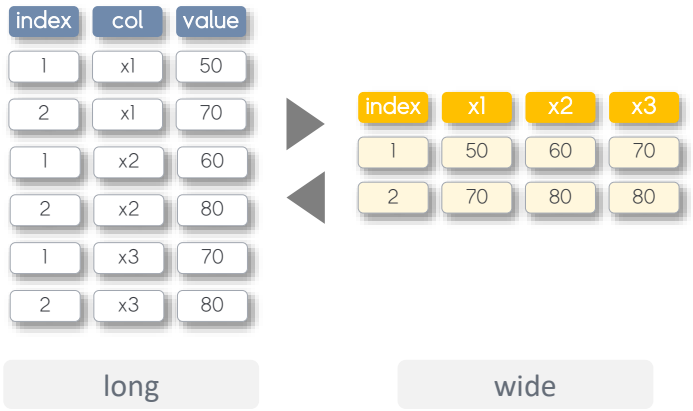
3.1 데이터 재구조화 (데이터 변형)

- 현업에서는 분석가가 수집한 데이터가 원하는 형태로 수집되지 않아서 따로 변형 시켜야하는 경우가 빈번이 발생됩니다.
- 또한 각 그룹에 따른 통계량을 보고싶은 경우에도 코딩을 통한 데이터의 변형이 필요하게됩니다.

Pivot

- 데이터 재구조화

• 수집한 데이터를 데이터에 활용하기 위해서 원하는 형태로 변형해야 하는 일은 빈번히 발생하며, 주로 사용되는 형태 변환은 long → wide, wide → long의 형태로 변형하는 경우가 많음



실습

```
[9] # 데이터 재구조화 (long -> wide) ver1
sample_wide = sample2.pivot(index = 'index', columns = 'col', values = 'value')
sample_wide
```

index	x1	x2	x3
1	50	60	70
2	70	80	80

```
[10] # 데이터 재구조화 (long -> wide) ver2
sample_wide2 = pd.pivot_table(sample2, index = 'index', columns = 'col', values = 'value')
sample_wide2
```

index	x1	x2	x3
1	50	60	70
2	70	80	80

```
[12] # 데이터 재구조화 (wide -> long)
sample_long = pd.melt(sample_wide2, id_vars = 'index')
sample_long
```

	index	col	value
0	1	x1	50
1	2	x1	70
2	1	x2	60
3	2	x2	80
4	1	x3	70

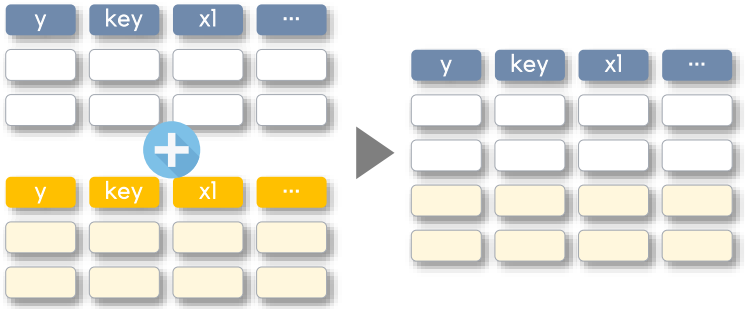
3.2 데이터 결합

- 데이터 전처리에 사용되는 여러가지 활동 중 여러 “데이터를 결합”하는 방법과 “불필요한 데이터를 제거"하는 방법에 대해 파이썬을 활용한 실습을 통해 알아보도록 하겠습니다.
- 데이터를 결합하는 방법에는 “Union”과 “Left Join”에 대해 알아보도록 하겠습니다.

데이터 결합 (Union)

- 데이터 결합 (Union)

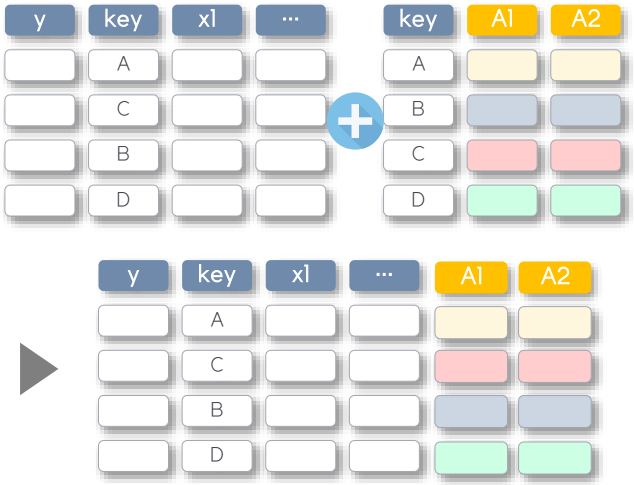
• Union은 데이터 결합의 방법 중 하나로 2개 이상의 데이터를 가로 방향으로 기존 데이터 아래 결합시키는 방법을 의미함
• 기존 데이터 아래로 결합시키는 경우 동일한 컬럼을 기준으로 결합시킴



데이터 결합 (Left Join)

- 데이터 결합 (Left Join)

• Left Join은 데이터 결합의 방법 중 하나로 왼쪽 데이터를 기준으로 오른쪽 데이터를 옆으로 결합시키는 방법을 의미함
• 왼쪽 데이터에 오른쪽 데이터를 결합시키는 경우 “key” 컬럼이라고 불리우는 컬럼을 기준으로 결합시킴



3.2 데이터 결합


```
[22] import pandas as pd
     import numpy as np
     from IPython.core.interactiveshell import InteractiveShell
     InteractiveShell.ast_node_interactivity = "all"
```

```
[23] # 필요한 데이터 읽기
     tb_order_04 = pd.read_csv('tb_order_202105.csv')
     tb_order_05 = pd.read_csv('tb_order_202104.csv')
     tb_order_04.head(10)
     tb_order_04.shape
     tb_order_05.head(10)
     tb_order_05.shape
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2
1	86276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9
5	92005874	104	C00574154	80	58592863	2021-05-01 11:00	2021-05-01 11:59	0	46240	2
6	8243996	171	C79994479	28	16376941	2021-05-01 11:00	2021-05-01 11:43	0	20640	2
7	62015909	191	C41716522	69	77974780	2021-05-01 11:00	2021-05-01 11:54	0	27320	2
8	23354107	108	C04237650	26	18054006	2021-05-01 11:00	2021-05-01 11:50	0	29870	2
9	82121011	123	C95413238	22	11628886	2021-05-01 11:00	2021-05-01 11:10	0	27910	2

(241142, 10)

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	34104383	11	C65806632	57	61573513	2021-04-01 11:00	2021-04-01 11:39	1	28270	1
1	70652318	59	C09760173	37	54068709	2021-04-01 11:00	2021-04-01 11:34	0	28270	2
2	71640388	195	C61227084	17	93678366	2021-04-01 11:00	2021-04-01 11:54	0	26470	9
3	75673365	127	C64119972	17	5287952	2021-04-01 11:00	2021-04-01 11:17	0	23080	2
4	9077529	174	C10231192	18	18248867	2021-04-01 11:00	2021-04-01 11:35	0	46920	2
5	86102793	167	C06298599	21	70395221	2021-04-01 11:00	2021-04-01 11:59	1	37420	1
6	49394078	187	C99985130	97	31854449	2021-04-01 11:00	2021-04-01 11:31	0	38650	9
7	10290387	30	C17281363	75	81945254	2021-04-01 11:00	2021-04-01 11:16	1	22380	1
8	54821099	9	C16681192	90	9773815	2021-04-01 11:00	2021-04-01 11:35	0	23080	2
9	42415936	156	C00944252	84	82103008	2021-04-01 11:00	2021-04-01 11:30	1	23120	1

(233262, 10)

3.2 데이터 결합

1.1. 데이터 결합 (유니온) 하기

```
[24] # 데이터 결합하기
tb_order_all = pd.concat([tb_order_04, tb_order_05])
tb_order_all.head()
tb_order_all.tail()
tb_order_all.shape
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
233257	61665702	103	C51797758	48	77989403	2021-04-30 21:58	2021-04-30 22:36	0	35300	2
233258	31151690	119	C04883863	78	16130893	2021-04-30 21:58	2021-04-30 22:35	0	32340	2
233259	15955692	175	C87594637	96	11457934	2021-04-30 21:58	2021-04-30 22:32	0	36170	9
233260	73251339	145	C93839111	92	64743537	2021-04-30 21:58	2021-04-30 22:46	0	36170	2
233261	42442108	117	C44570215	54	2304995	2021-04-30 21:58	2021-04-30 22:37	1	18990	1

(474404, 10)

```
[25] # 중복 제거 하기
tb_order_all.drop_duplicates(inplace = True)
```

```
# index 초기화 하기
tb_order_all.reset_index(drop = True, inplace = True)
tb_order_all.head()
tb_order_all.tail()
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
474399	61665702	103	C51797758	48	77989403	2021-04-30 21:58	2021-04-30 22:36	0	35300	2
474400	31151690	119	C04883863	78	16130893	2021-04-30 21:58	2021-04-30 22:35	0	32340	2
474401	15955692	175	C87594637	96	11457934	2021-04-30 21:58	2021-04-30 22:32	0	36170	9
474402	73251339	145	C93839111	92	64743537	2021-04-30 21:58	2021-04-30 22:46	0	36170	2
474403	42442108	117	C44570215	54	2304995	2021-04-30 21:58	2021-04-30 22:37	1	18990	1

3.2 데이터 결합

1.2. 데이터 결합(Left Join)하기

```
[16] # 결합시킬 데이터 불러오기
a_store = pd.read_csv('a_store.csv')
a_store.head()

# 온주에 가공이 된 데이터
tb_order_all.head()
```

	store_id	store_name	area_cd
0	1	삼일대로점	SL
1	2	세종대로점	SL
2	3	우고로점	SL
3	4	덕수궁길점	SL
4	5	서소문로점	SL

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9

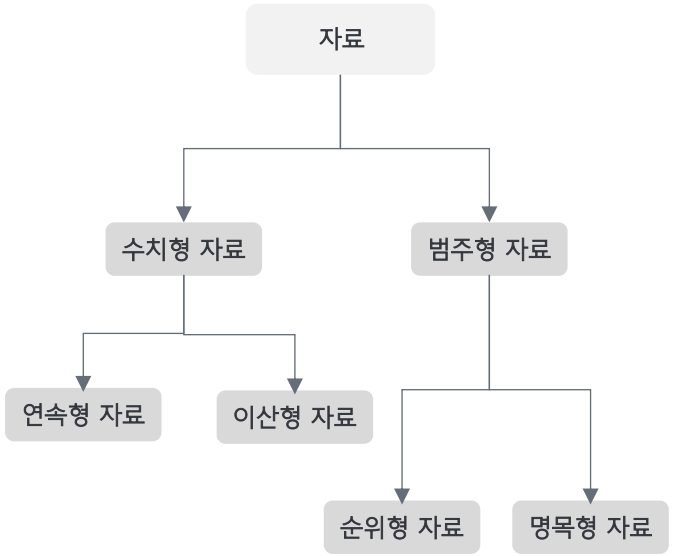
```
# 데이터 결합 (Left Join)
# key 컬럼 : store_id
order_df = pd.merge(tb_order_all, a_store, on ='store_id', how = 'left')
order_df.head()
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status	store_name	area_cd
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2	삼봉로점	SL
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2	광산로점	GJ
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2	연제로점	BS
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2	황양대로2점	BS
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9	당산로2점	SL

3.3 자료 형태에 따른 그래프

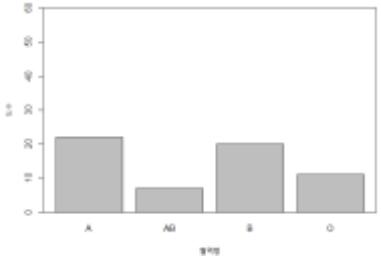
- 데이터 분석에서 전처리 및 시각화 단계는 데이터 분석을 통틀어서 가장 많은 비중을 차지하는 단계로 데이터 분석 단계에서 가장 중요한 단계라고 할 수 있습니다.
- 데이터 전처리와 시각화는 기본적으로는 전처리가 우선으로 수행되지만 분석 과정 내에서 반복적으로 수행됩니다.

자료의 형태



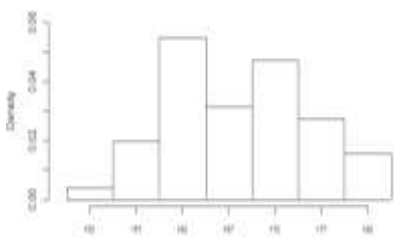
자료의 형태에 따른 그래프

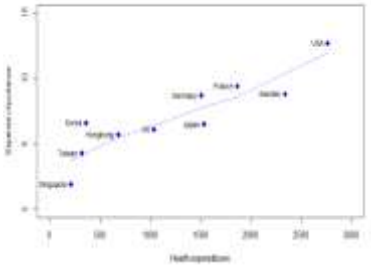
- 범주형 변수



범주	도수	상대 도수
A	22	22/60
B	20	20/60
AB	7	7/60
O	11	11/60
합	60	1

- 수치형 변수





3.4 데이터 시각화

- 데이터 내 수치형 변수들 사이의 관계를 파악하기 위한 가장 대표적인 방법 중 하나는 “상관성”의 정도를 의미하는 “상관계수“와 함께 산점도를 확인하는 것입니다.
- 상관계수라는 통계량에서 중요한 것은 “선형성”을 의미하는 수치를 의미한다는 것이 유념해야 할 부분이기 때문에 그 외의 관계를 확인하기 위해 산점도를 함께 확인하는 것이 좋습니다.

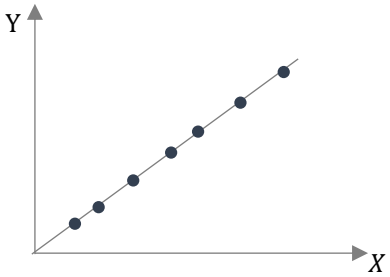
상관계수

상관성

- 두 수치형 변수 사이의 점의 선형성의 강도를 통해서 변수들 끼리의 관계를 파악하고자 하는 특성

상관계수

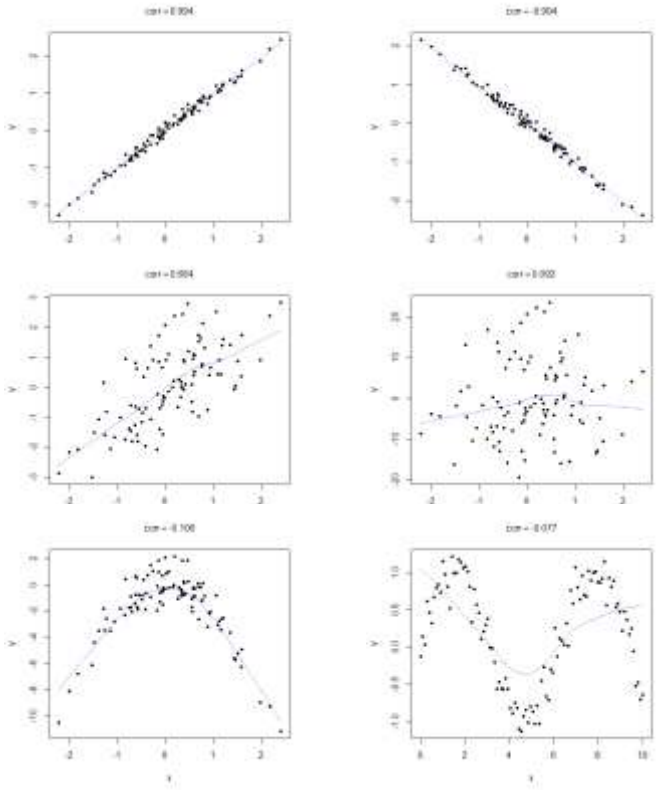
- 상관계수는 항상 -1과 1사이에 있음
- 상관계수의 절댓값의 크기는 선형(직선)관계에 가까운 정도를 나타내고 상관계수의 부호는 선형(직선)관계의 방향을 나타냄
- 상관계수의 단위는 없기 때문에 단위가 다른 여러 쌍의 변수의 직선관계정도를 비교할 수 있음



(2,2) (3,3) (5,5) (7,7)
(8,8) (11,11) (13,13)

$$r = \frac{S_{xy}}{\sqrt{S_{xx}} \cdot \sqrt{S_{yy}}}$$

상관계수 형태



3.4 데이터 시각화

- 실습 데이터는 총 3개의 데이터를 활용합니다.
- 서울의 지역(시군구)이름 및 코드 정보와, 서울 일별 지역별 날씨 관련 정보 및 눈병 환자 수 데이터, 서울 지역별 년도별 인구수 데이터를 활용합니다.

- 지역(시군구)이름 및 코드 정보

	SIGUNGU_CD	SIGUNGU_NM
0	11010	종로구
1	11020	중구
2	11030	용산구
3	11040	성동구
4	11050	광진구

- 지역(시군구)이름 및 코드 정보

	행정구역	2014년_총인구수	2015년_총인구수	2016년_총인구수
0	서울특별시 종로구 (1111000000)	156993	154986	152737
1	서울특별시 중구 (1114000000)	128065	125733	125249
2	서울특별시 용산구 (1117000000)	235951	233342	230241
3	서울특별시 성동구 (1120000000)	296086	297003	299259
4	서울특별시 광진구 (1121500000)	363354	360369	357215

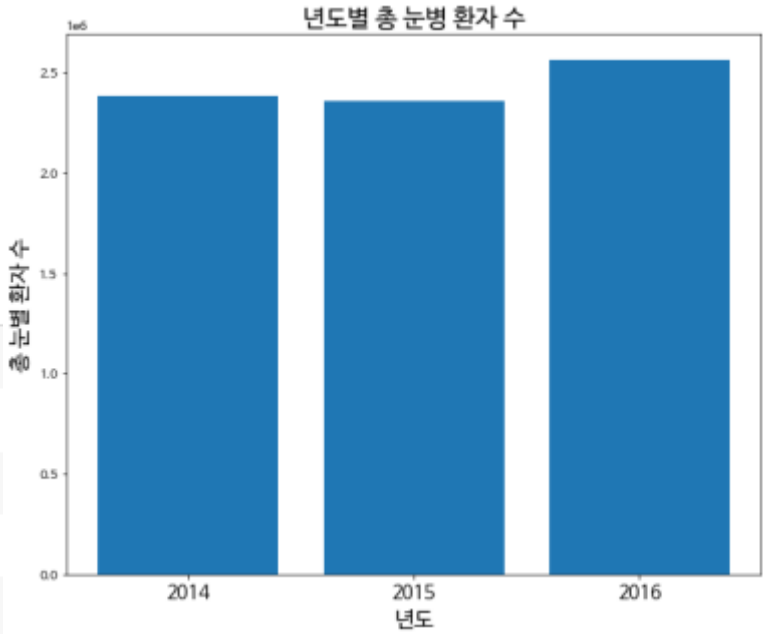
- 서울 일별 지역별 날씨 관련 정보 및 눈병 환자 수

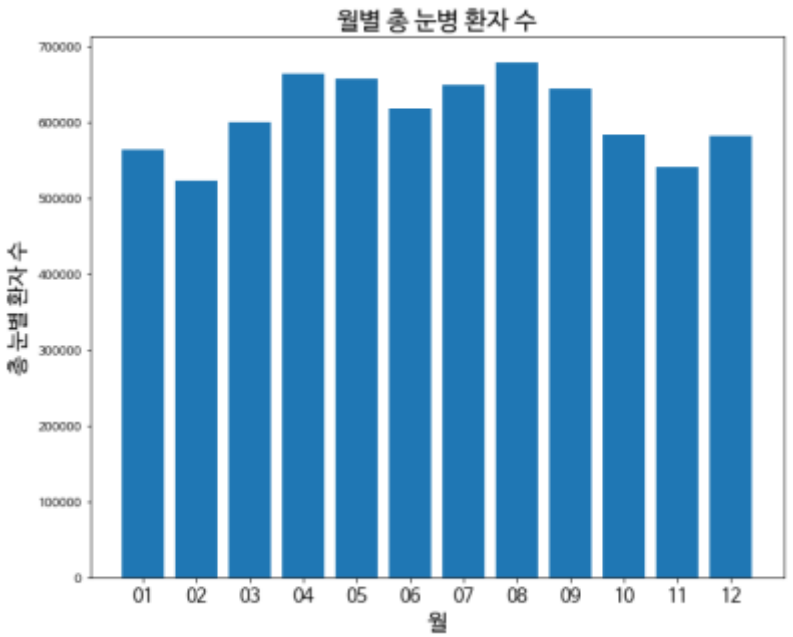
	date	sigungu_cd	sigungu	avg_temp	min_temp	min_temp_time	max_temp	max_temp_time	rain_per_h	rain_per_time	...	평균습도	최저습도	최고습도	평균풍속	최대풍속	강수량	탁도	pH	잔류염소	eye
0	2014-01-01	11110	종로구	1.8	-1.4	348.0	5.2	1432.0	0.0	NaN	...	60.3	35.7	72.9	1.2	4.9	0.0	0.07	7.1	0.31	2
1	2014-01-01	11140	중구	3.2	0.6	433.0	7.4	1310.0	0.0	NaN	...	60.8	36.0	75.2	2.5	7.9	0.0	0.07	7.2	0.31	1
2	2014-01-01	11170	용산구	5.0	1.9	610.0	8.4	1426.0	0.0	NaN	...	60.1	33.0	71.8	2.1	7.6	0.0	0.07	7.2	0.32	7
3	2014-01-01	11200	성동구	5.3	2.3	658.0	8.7	1445.0	0.0	NaN	...	54.4	29.0	69.9	2.7	7.6	0.0	0.06	7.2	0.32	9
4	2014-01-01	11215	광진구	5.3	0.9	710.0	8.8	1428.0	0.0	NaN	...	60.9	34.1	74.8	1.5	6.4	0.0	0.07	7.1	0.31	6

- 범주형 데이터에서의 막대 그래프

	year	eye
0	2014	2387303
1	2015	2359266
2	2016	2566345

	month	eye
0	01	564596
1	02	523718
2	03	599872
3	04	665179
4	05	658147
5	06	618440
6	07	649612
7	08	679932
8	09	644325
9	10	584600
10	11	542152
11	12	582341





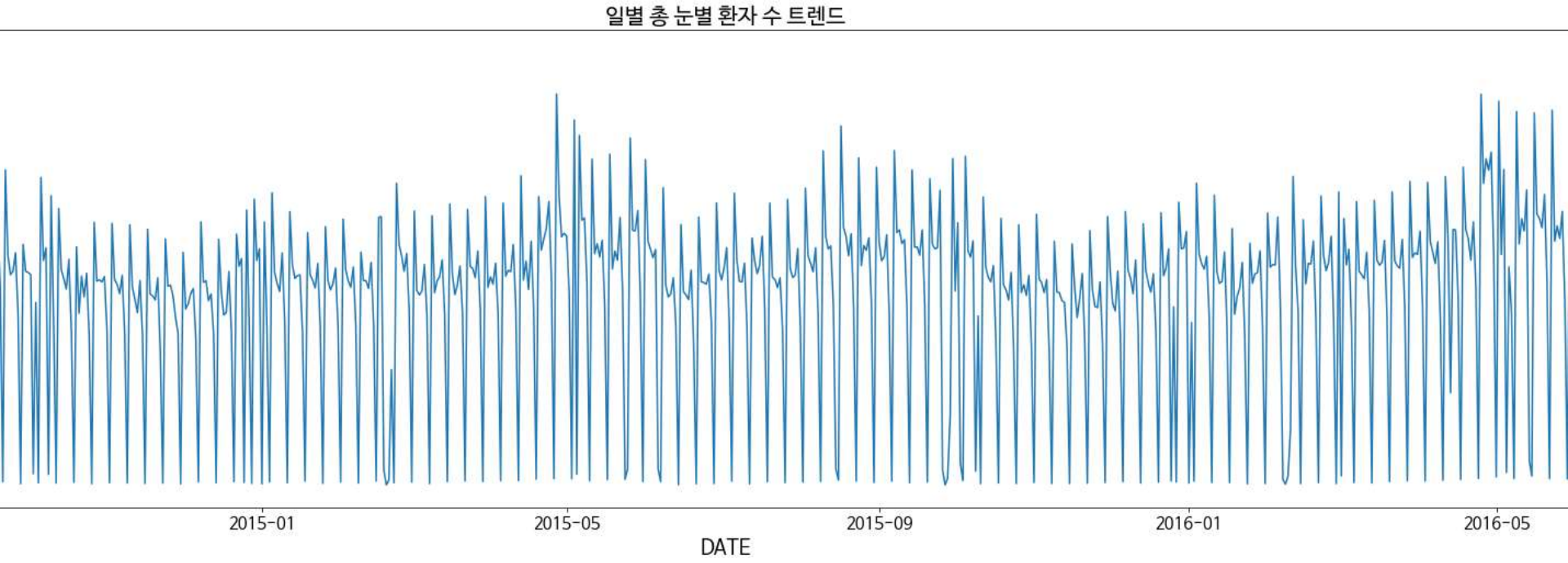

```
# 막대그래프를 활용한 시각화
# 패키지
import matplotlib.pyplot as plt
from matplotlib import gridspec
import seaborn as sns
import os
plt.rc('font', family='NanumBarunGothic')
```

```
plt.figure(figsize=(10,8))

plt.bar(year_eye['year'], year_eye['eye'])
plt.title('년도별 총 눈병 환자 수', fontsize = 20)
plt.xlabel('년도', fontsize = 18)
plt.ylabel('총 눈병 환자 수', fontsize = 18)
plt.xticks(fontsize = 15)
plt.show()
```

```
plt.figure(figsize=(10,8))

plt.bar(month_eye['month'], month_eye['eye'])
plt.title('월별 총 눈병 환자 수', fontsize = 20)
plt.xlabel('월', fontsize = 18)
plt.ylabel('총 눈병 환자 수', fontsize = 18)
plt.xticks(fontsize = 15)
plt.show()
```

3.4 데이터 시각화

- 수치형 데이터에서의 트렌드 그래프

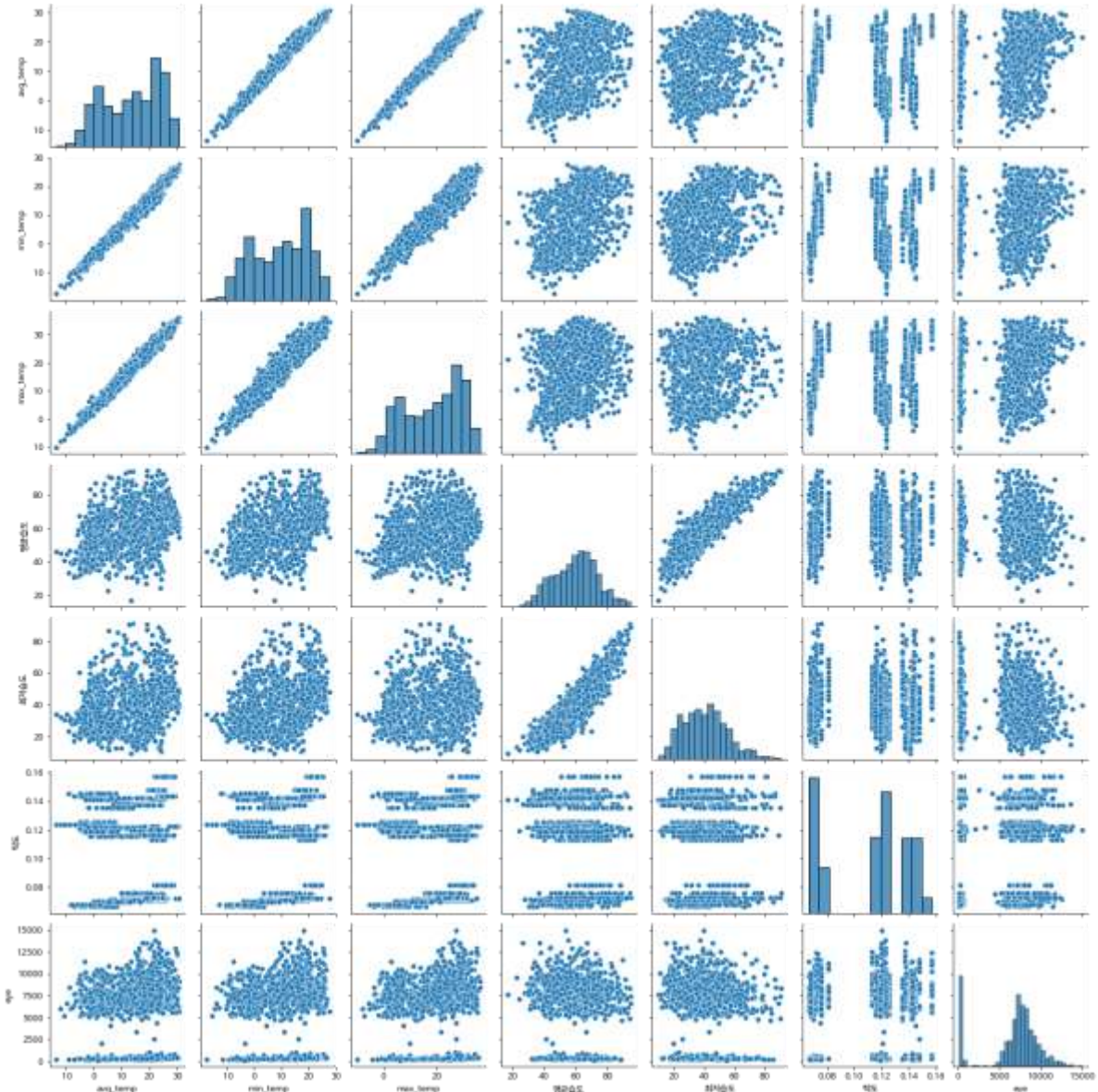


	date	avg_temp	min_temp	max_temp	평균습도	최저습도	탁도	eye
0	2014-01-01	4.468	0.724	8.112	59.756	35.060	0.0676	247
1	2014-01-02	1.920	-1.076	6.076	40.672	21.608	0.0676	8811
2	2014-01-03	2.656	-1.032	7.372	56.164	39.972	0.0676	7829
3	2014-01-04	0.224	-3.156	4.908	52.112	42.680	0.0676	6350
4	2014-01-05	-0.364	-3.716	3.812	51.544	32.912	0.0676	234

```
plt.figure(figsize=(50,8))
plt.plot(tot_df2['date'], tot_df2['eye'])
plt.title('일별 총 눈병 환자 수 트렌드', fontsize = 20)
plt.xlabel('DATE', fontsize = 18)
plt.ylabel('총 눈병 수', fontsize = 18)
plt.xticks(fontsize = 15)

plt.show()
```

3.4 데이터 시각화

- 수치형 데이터에서의 상관관계 시각화 (산점도 그래프 / pair plot)



3.4 데이터 시각화

- 수치형 데이터에서의 상관관계 시각화 (Heatmap)