

2022 데이터 청년 캠퍼스

스마트시티 AI·빅데이터 활용 과정

시각화(Python)

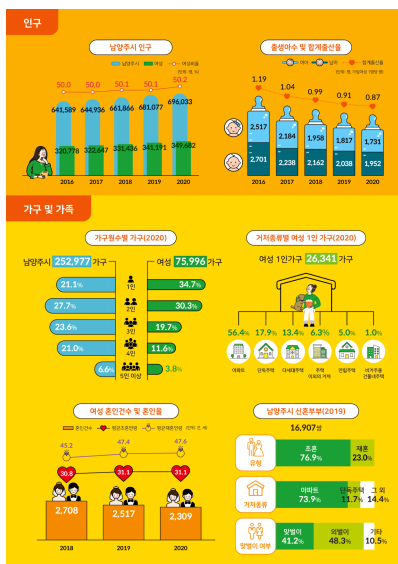
1 시각화

□ 데이터 시각화

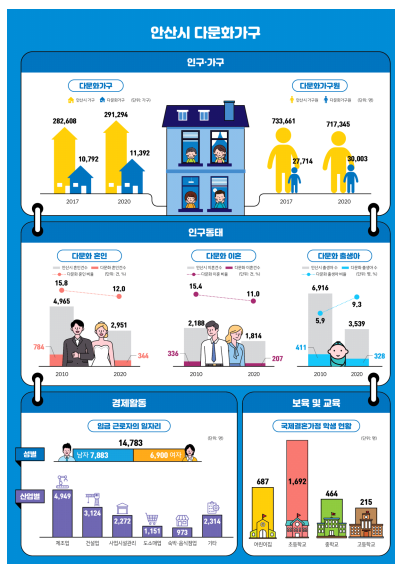
- 데이터 시각화(Data visualization)는 데이터 분석 결과를 쉽게 이해할 수 있도록 시각적으로 표현하고 전달되는 과정
- 목적은 커뮤니케이션(정보 전달, 대화)
- 데이터 시각화의 목적은 도표(graph)라는 수단을 통해 정보를 명확하고 효과적으로 전달하는 것
 - 방대한 양의 데이터를 탐색하거나 이해할 때 가장 좋은 방법
 - 데이터를 지식화하기 위한 과정

□ 인포그래픽

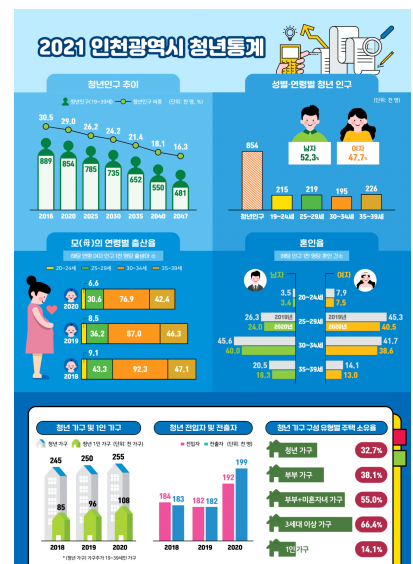
- 인포메이션 그래픽(Information graphics) 또는 인포그래픽(Infographics), 뉴스 그래픽(News graphics)은 정보를 빠르고 분명하게 표현하기 위해 정보, 자료, 지식을 그래픽 시각적으로 표현한 것
- 그래프의 추상적인 표현이나 디자인적인 요소를 활용하여 정보를 표현하는 방법
- 데이터를 조직화해서 특정 정보를 전달하려는 것이 목적
 - 경인지방통계청: http://kostat.go.kr/regional/gi/gi_ntc/2/index.board



남양주시 여성통계



안산시 다문화 외국인 가구통계



인천광역시 청년통계

□ 대규모 데이터 시각화

- 같은 범주 안에서 많은 양의 데이터에 의미를 부여함
- 공간에 배치된 숫자의 패턴을 인지하게 만든 것

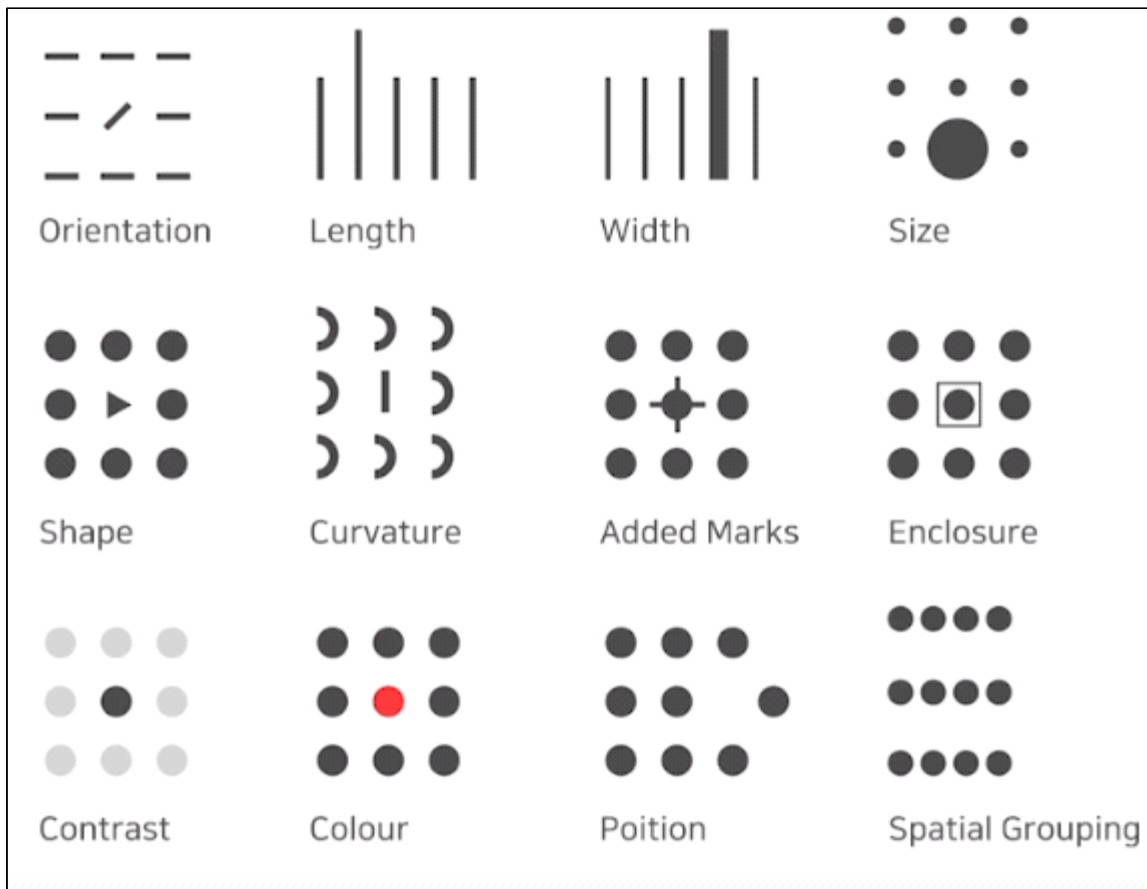
□ 시각화의 원리

- 하나의 시각화는 그 데이터에 표현하는 유일한 특성들만을 표현해야 한다.
- 가능한 한 소중한 정보만으로 최소화해야 한다.

■ 시각적 표현

특성	내용
크기	면적이나 도형 모양의 확대/축소를 이용하여, 사용자가 직관적으로 구별할 수 있기 때문에 가장 많이 쓰이는 표현
색상	자료가 많을 때 규칙성과 특이성을 구분하는 것에 효과적
위치	지도나 가상의 장소 데이터를 연결하여 나타냄으로써 관찰자가 자신의 정황을 시각화에 투영하여 해석하도록 함
네트워크	데이터 사이의 관계를 표현하며 각 데이터들은 노드(node)로 연결하는 것과 같으며, 사회적 관계를 시각화하거나 방대한 양의 데이터들 사이의 관계를 그룹지어 보여줄 때 유용함
시간	시간의 순서에 따라서 데이터를 나열하는 방법

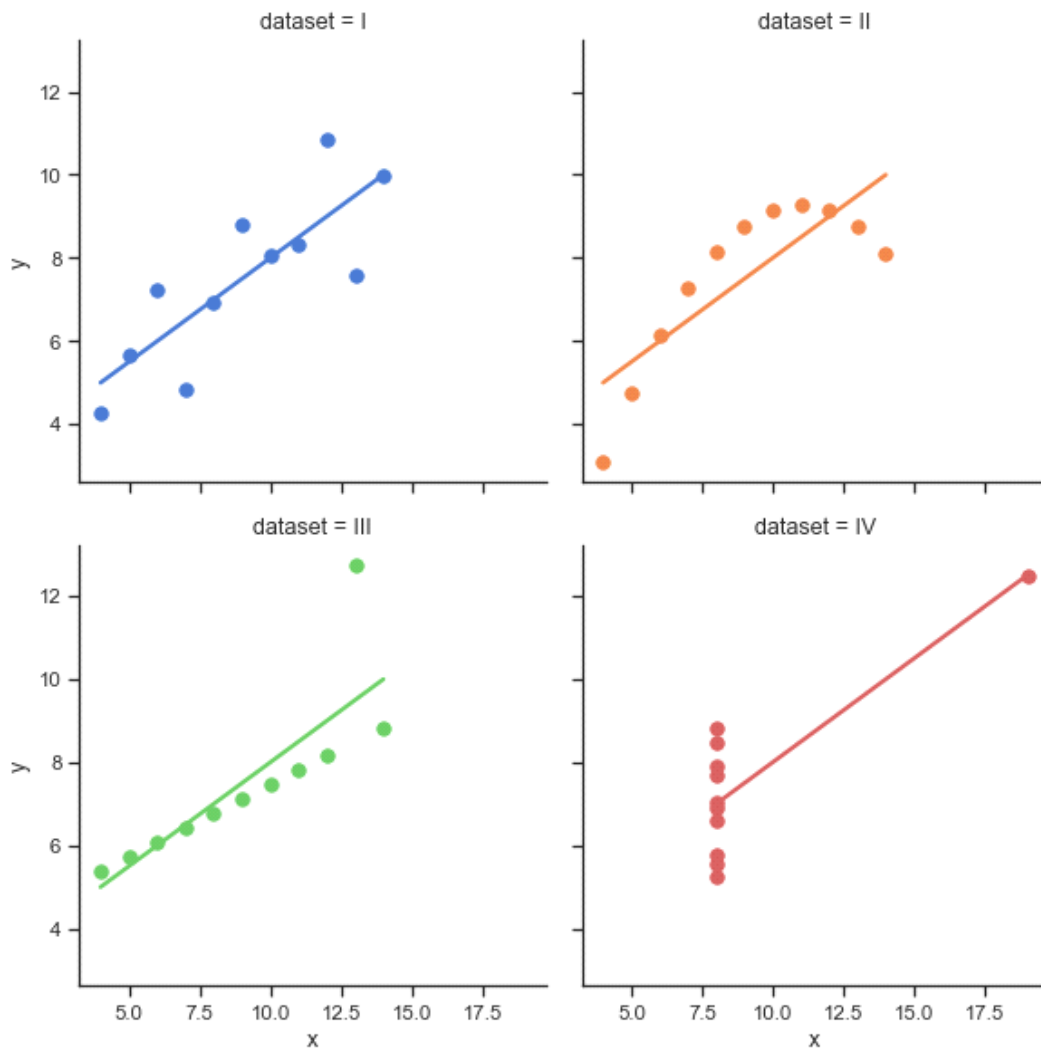
- 전주의적 속성 (Pre-attentive Attribute)



○ 앤스컴 콰르텟(Anscombe's quartet)

– 기술통계량은 유사하지만 분포나 그래프는 매우 다른 4개의 데이터셋

	I		II		III		IV	
	x	y	x	y	x	y	x	y
	10	8,04	10	9,14	10	7,46	8	6,58
	8	6,95	8	8,14	8	6,77	8	5,76
	13	7,58	13	8,74	13	12,74	8	7,71
	9	8,81	9	8,77	9	7,11	8	8,84
	11	8,33	11	9,26	11	7,81	8	8,47
	14	9,96	14	8,1	14	8,84	8	7,04
	6	7,24	6	6,13	6	6,08	8	5,25
	4	4,26	4	3,1	4	5,39	19	12,5
	12	10,84	12	9,13	12	8,15	8	5,56
	7	4,82	7	7,26	7	6,42	8	7,91
	5	5,68	5	4,74	5	5,73	8	6,89
SUM	99,00	82,51	99,00	82,51	99,00	82,50	99,00	82,51
AVG	9,00	7,50	9,00	7,50	9,00	7,50	9,00	7,50
STDEV	3,32	2,03	3,32	2,03	3,32	2,03	3,32	2,03



3	5	3	4	3	6	3	8	3	4	5	6	8	3	4	5
6	3	3	3	7	3	3	3	6	4	3	3	7	6	4	3
4	5	5	5	4	5	5	9	8	5	5	5	9	8	5	5
1	8	6	8	9	8	3	1	5	8	8	8	2	5	8	8
6	1	9	7	6	3	9	7	4	5	4	6	8	4	5	9
9	8	8	8	9	8	3	3	3	4	8	8	3	4	4	8
4	5	4	5	4	2	4	4	8	5	5	5	4	8	5	5
1	4	6	1	1	1	6	1	9	8	9	8	1	8	8	8
3	4	8	4	3	4	8	3	1	4	4	4	3	1	4	4
5	3	5	3	5	3	5	5	3	6	3	3	6	3	6	3
4	5	3	5	4	5	3	4	5	1	5	5	4	5	1	5
9	8	5	8	7	8	5	7	2	3	8	8	7	9	3	8
6	7	7	7	6	7	7	6	5	5	7	7	6	5	5	7
9	6	1	6	9	6	1	9	3	8	6	6	9	3	8	6
4	5	8	5	4	5	8	4	6	4	5	5	4	6	4	5
1	6	3	6	8	9	3	7	4	6	1	9	7	4	3	1

■ 시각화 방법

○ 차트와 통계 도구

- 시각화를 위한 모든 기능과 도구를 내장한 도구

○ 프로그래밍 환경

- 소프트웨어의 지원 범위를 벗어나는 기능이 필요한 경우
- 프로그래밍 언어와 구문을 활용하여 목적에 맞게 코딩하는 것
- 데이터 처리에 유리함

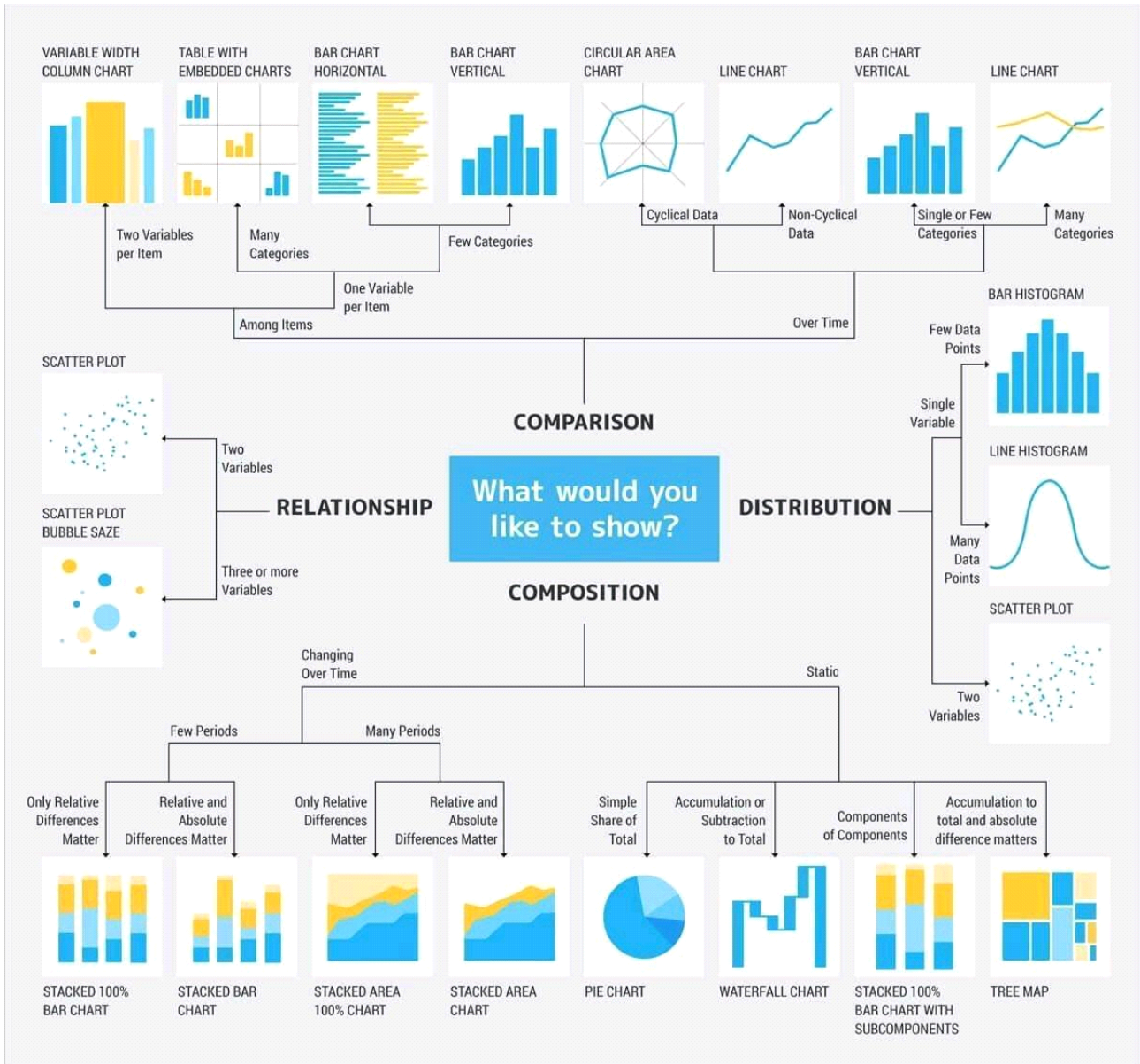
○ 지도

- 공간 데이터 시각화에 유리함
- 모바일 위치 정보 기반으로 데이터를 탐색하는 도구로 발전

□ 시각화의 구분

○ 그래프의 유형

- 비교(Comparison), 구성(Composition), 분포(Distribution), 관계(Relationship)



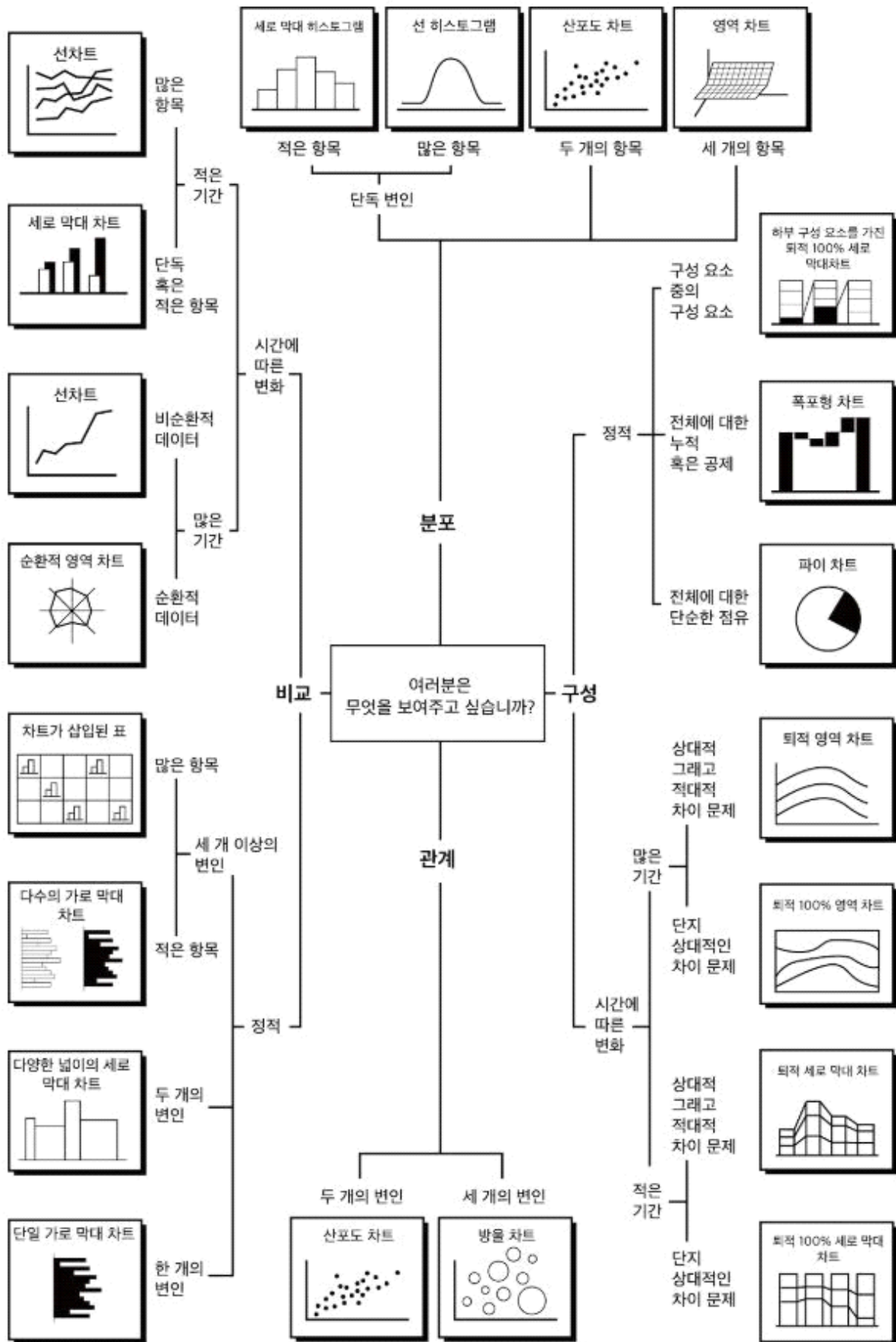


그림 1.20 앤드루 아벨라가 고안한 목적에 따른 차트 선택 방법

2 데이터 탐색

□ 자료

- 자료는 실험 또는 조사, 관찰 등을 통하여 수집
- 자료의 구분에 따라서 분석 방법이나 시각화가 달라짐

□ 자료의 구분

- 속성에 의한 구분
 - 수치형 자료(Numerical data), 정량적/양적 자료(Quantitative data)
 - : 숫자로 측정되는 자료
 - 범주형 자료(Categorical data), 정성적/질적 자료(Qualitative data)
 - : 숫자로 측정할 수 없는 자료

속성에 의한 구분	측정의 척도	설명
수치형 자료	비율척도(ratio scale)	절대 원점이 존재하는 수치형 척도
	구간척도(interval scale)	상대적 원점이 존재하는 구간에서 측정하는 수치형 척도
범주형 자료	순서척도(ordinal scale)	분류에 따른 서열 관계를 측정하는 척도
	명목척도(nominal scale)	분류에 사용하는 척도

- 형태에 의한 구분
 - 연속형 자료(Continuous data)
 - : 연속적인 구간에서 얻어지는 자료
 - 이산형 자료(Discrete data)
 - : 이산점으로 구성된 자료

속성에 의한 구분	형태에 의한 구분	측정의 척도	예
수치형 자료	연속형 자료	비율 척도	길이, 소득
		구간 척도	온도, 지능지수
범주형 자료	이산형 자료	비율/구간 척도	불량품의 수, 사고 건수
		순서 척도	만족도, 소득수준
		명목 척도	성별, 혈액형

○ 일반적인 데이터 시각화에서 다루는 변수의 유형

변수 유형	예시	적정 스케일	설명
수치형/연속형	1.3, 5.7, 83, 1.5×10^{-2}	연속형	임의의 숫자 값들. 정수, 유리수, 실수
범주형/이산형	1, 2, 3, 4	이산형	이산적인 단위의 숫자들. 대부분 정수지만 예외도 있다. 예를 들어 0.5, 1.0, 1.5 같은 숫자도 주어진 데이터셋에 중간값이 존재할 수 없으면 이산형 데이터로 처리할 수 있다.
비 순서형 범주	개, 고양이, 물고기	이산형	순서를 매길 수 없는 범주들. 이산적이며, 서열 속성이 없는 고유의 범주들이다. 이 변수를 '요인(factor)'이라고도 부른다.
순서형 범주	좋음, 보통, 나쁨	이산형	순서가 있는 범주들. 이산적이고 순서가 있는 범주들이다. 예를 들어 '보통'은 항상 '좋음'과 '나쁨' 사이에 위치한다. 이 변수들을 '순서 요인'이라고 부르기도 한다.
날짜 또는 시간	2021년 4월 13일 오전 7시 20분	연속형 또는 이산형	특정 날짜나 시간. 7월 4일이나 12월 25일 같이 연도가 특정되지 않은 날짜도 해당한다.
텍스트	논리적 사고를 훈련 하기 위해 프로그래 밍 언어를 배워야 한다	이산형 또는 해당 없음	자유 형식의 문자. 필요에 따라 범주형 데이터로 처리할 수도 있다.

▶ 자료

- 타슈 이용현황: 2020.csv

```
1 ### Packages
2 from easygui import fileopenbox
3 import pandas as pd
4 import numpy as np
```

```
1 ### 타슈 이용현황: 2020.csv
2 data_file = fileopenbox()
3 data_file
```

```
1 ### 타슈 이용현황: 2020.csv
2 data_file = "./data/타슈/2020.csv"
3 tashu_2020 = pd.read_csv(data_file, encoding = "cp949")
4 tashu_2020.head(3)
```

	대여스테이션	대여일시	반납스테이션	반납일시	이동거리	회원구분
0	174	20200101000100	224.0	20200101001137	640.0	2
1	174	20200101000109	224.0	20200101001210	640.0	2
2	117	20200101050735	115.0	20200101051922	1070.0	0

```
1 ### 자료형 확인
2 tashu_2020.dtypes
```

```
대여스테이션      int64
대여일시          int64
반납스테이션      float64
반납일시          int64
이동거리          float64
회원구분          int64
dtype: object
```

```
1 ### 정보 확인
2 tashu_2020.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 604446 entries, 0 to 604445
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   대여스테이션  604446 non-null  int64
1   대여일시     604446 non-null  int64
2   반납스테이션  595778 non-null  float64
3   반납일시     604446 non-null  int64
4   이동거리     595857 non-null  float64
5   회원구분     604446 non-null  int64
dtypes: float64(2), int64(4)
memory usage: 27.7 MB
```

▶ 자료형 변환

```
1 ### 자료형 변환: 실수 → 정수
2 tashu_2020[['반납스테이션', '이동거리']] = tashu_2020[['반납스테이션', '이동거리']].astype('Int64')
3 tashu_2020.head(3)
```

	대여스테이션	대여일시	반납스테이션	반납일시	이동거리	회원구분
0	174	20200101000100	224	20200101001137	640	2
1	174	20200101000109	224	20200101001210	640	2
2	117	20200101050735	115	20200101051922	1070	0

```
1 ### 자료형 변환: 정수 → 문자
2 tashu_2020[['대여스테이션', '대여일시', '반납스테이션', '반납일시', '회원구분']] = tashu_2020[['대여스테이션', '대여일시', '반납스테이션', '반납일시', '회원구분']].astype('string')
3 tashu_2020.head()
```

	대여스테이션	대여일시	반납스테이션	반납일시	이동거리	회원구분
0	174	20200101000100	224	20200101001137	640	2
1	174	20200101000109	224	20200101001210	640	2
2	117	20200101050735	115	20200101051922	1070	0
3	167	20200101051348	94	20200101054627	1490	2
4	203	20200101052002	203	20200101052058	0	1

```
1 ### 정보 확인
2 tashu_2020.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 604446 entries, 0 to 604445
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0   대여스테이션  604446 non-null  string
1   대여일시     604446 non-null  string
2   반납스테이션  595778 non-null  string
3   반납일시     604446 non-null  string
4   이동거리     595857 non-null  Int64
5   회원구분     604446 non-null  string
dtypes: Int64(1), string(5)
memory usage: 28.2 MB
```

```

1 ### 기술 통계량(데이터 요약)
2 tashu_2020.describe(include='all')

```

	대여스테이션	대여일시	반납스테이션	반납일시	이동거리	회원구분
count	604446	604446	595778	604446	595857.000000	604446
unique	262	590970	262	590517	NaN	3
top	3 20200926180015		3 20200905164703		NaN	2
freq	25476	4	27709	4	NaN	388611
mean	NaN	NaN	NaN	NaN	1654.610620	NaN
std	NaN	NaN	NaN	NaN	1446.016779	NaN
min	NaN	NaN	NaN	NaN	0.000000	NaN
25%	NaN	NaN	NaN	NaN	530.000000	NaN
50%	NaN	NaN	NaN	NaN	1290.000000	NaN
75%	NaN	NaN	NaN	NaN	2560.000000	NaN
max	NaN	NaN	NaN	NaN	5630.000000	NaN

■ 자료: 타슈 스테이션 현황

- 대전광역시_공영자전거(타슈) 위치(위경도) 현황_20200801.csv

```
1 ### 타슈 스테이션 현황: 대전광역시_공영자전거(타슈) 위치(위경도) 현황
  _20200801.csv
2 data_file = "./data/타슈/대전광역시_공영자전거(타슈) 위치(위경도) 현황
  _20200801.csv"
3 tashu_station = pd.read_csv(data_file, encoding = "cp949")
4 tashu_station.head(1)
```

연 번	Station 스테이션/성명	위 치	Latitude	Longitude	광역시 시도코드	광역시 시도명	시군 구코드	시군 구명	법정동코드	법정 동명	행정동코드	행정 동명	거 치 대
0	1	대전광역시유성구도룡동3-8 무역전 시관입 구(택시 승강장)	36.374708	127.389027	30	대전광역시	30200	유성구	3020012700	도룡동	3020055000	신성동	14

```
1 ### 정보 확인
2 tashu_station.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 262 entries, 0 to 261
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   연번                  262 non-null    int64
1   Station 스테이션/성명  262 non-null    object
2   위치                  262 non-null    object
3   Latitude              262 non-null    float64
4   Longitude              262 non-null    float64
5   광역시도코드          262 non-null    int64
6   광역시도명            262 non-null    object
7   시군구코드            262 non-null    int64
8   시군구명              262 non-null    object
9   법정동코드            262 non-null    int64
10  법정동명              262 non-null    object
11  행정동코드            262 non-null    int64
12  행정동명              262 non-null    object
13  거치대                262 non-null    int64
dtypes: float64(2), int64(6), object(6)
memory usage: 28.8+ KB
```

▶ 자료형 변환

- `연번` 열의 자료형을 정수 → 문자로 변경

```
1 ### 자료형 변환: 정수 → 문자
2 tashu_station['연번'] = tashu_station['연번'].astype('string')
```

```
1 ### 정보 확인
2 tashu_station.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 262 entries, 0 to 261
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   연번                  262 non-null   string
1   Station 스테이션/성명 262 non-null   object
2   위치                  262 non-null   object
3   Latitude              262 non-null   float64
4   Longitude              262 non-null   float64
5   광역시도코드          262 non-null   int64
6   광역시도명            262 non-null   object
7   시군구코드            262 non-null   int64
8   시군구명              262 non-null   object
9   법정동코드            262 non-null   int64
10  법정동명              262 non-null   object
11  행정동코드            262 non-null   int64
12  행정동명              262 non-null   object
dtypes: float64(2), int64(4), object(6), string(1)
memory usage: 26.7+ KB
```

```
1 ### 기술 통계량(데이터 요약)
2 tashu_station.describe(exclude=[np.number]) # 수치형 제외
```

	연 번	Station 스테이션/성 명	위치	광역시도 명	시군구 명	법정동 명	행정동 명
count	262	262	262	262	262	262	262
unique	262	262	258	1	5	87	70
top	87	송강초등학교	대전광역시 유성구 구성 동 23	대전광역 시	유성구	둔산동	신성동
freq	1	1	4	262	76	20	14

▶ 자료 합치기

- 타슈 이용현황: 2020.csv
- 대전광역시_공영자전거(타슈) 위치(위경도) 현황_20200801.csv
- 결과: 대여/반납스테이션 이름 추가

```
1 ### merge by values
2 df_right = tashu_station[['연번', 'Station 스테이션/성명']]
3 df_right.columns = ['대여스테이션', '대여스테이션_이름']
4 tashu_2020_station = tashu_2020.merge(df_right, left_on='대여스테이션',
right_on='대여스테이션')
```

```
1 ### merge by values - left
2 df_right = tashu_station[['연번', 'Station 스테이션/성명']]
3 df_right.columns = ['반납스테이션', '반납스테이션_이름']
4 tashu_2020_station = tashu_2020_station.merge(df_right, left_on='반납스테이션',
right_on='반납스테이션', how='left')
```

```
1 tashu_2020_station.head()
```

	대여스 테이션	대여일시	반납스 테이션	반납일시	이동거 리	회원 구분	대여스테이션_ 이름	반납스테이 션_이름
0	174	20200101000100	224	20200101001137	640	2	가수원네거리 (전통시장 입구)	정림삼거리
1	174	20200101000109	224	20200101001210	640	2	가수원네거리 (전통시장 입구)	정림삼거리
2	174	20200101233236	175	20200101234225	800	2	가수원네거리 (전통시장 입구)	건양대네거리
3	174	20200101233237	175	20200101234236	800	2	가수원네거리 (전통시장 입구)	건양대네거리
4	174	20200102202221	253	20200102205546	2360	2	가수원네거리 (전통시장 입구)	가수원파출 소(건너편)


```

1  ### 정보 확인
2  tashu_2020_station.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 604446 entries, 0 to 604445
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   대여스테이션          604446 non-null  string
1   대여일시              604446 non-null  string
2   반납스테이션          595778 non-null  string
3   반납일시              604446 non-null  string
4   이동거리              595857 non-null  Int64
5   회원구분              604446 non-null  string
6   대여스테이션_이름     604446 non-null  object
7   반납스테이션_이름     595778 non-null  object
dtypes: Int64(1), object(2), string(5)
memory usage: 42.1+ MB

```

▶ 자료 정렬

－ `대여일시`, `반납일시` 기준으로 오름차순 정렬(2차 정렬)

```
1 ### Sort by '대여일시', '반납일시' - ascending=[True, True]
2 tashu_2020_station = tashu_2020_station.sort_values(by=['대여일시', '반납일시'])
3 tashu_2020_station
```

	대여 스테이션	대여일시	반납 스테이션	반납일시	이동 거리	회 원 구 분	대여스테이 션_이름	반납스테이 션_이름
0	174	20200101000100	224	20200101001137	640	2	가수원네거리(전통시장 입구)	정림삼거리
1	174	20200101000109	224	20200101001210	640	2	가수원네거리(전통시장 입구)	정림삼거리
3455	117	20200101050735	115	20200101051922	1070	0	동부네거리	한전 대전세 종충남지역 본부(건너 편)
5304	167	20200101051348	94	20200101054627	1490	2	원동네거리	미리내아파트
6962	203	20200101052002	203	20200101052058	0	1	죽동 금성백 조	죽동 금성백 조
...
473879	87	20201231233850	118	20210101002139	0	2	쌍용예가아 파트	복합터미널
406942	182	20201231233856	182	20201231233937	0	2	한밭대 입구	한밭대 입구
484709	42	20201231234559	83	20210101001215	2470	0	갈마네거리	유평초등학교
463885	115	20201231235132	196	20210101001502	1730	0	한전 대전세 종충남지역 본부(건너 편)	대덕문화원
142794	60	20201231235603	108	20210101001237	1590	2	유성온천역 3번출구	목원대학교 입구

604446 rows × 8 columns

▶ 변수 생성

－ 회원구분(0 : 정회원 / 1 : 일반회원 / 2 : 비회원)

```
1 ### 회원구분 값
2 idx = tashu_2020_station['회원구분'].astype('int')
3 idx
```

```
0      2
1      2
3455    0
5304    2
6962    1
...
473879  2
406942  2
484709  0
463885  0
142794  2
Name: 회원구분, Length: 604446, dtype: int32
```

```
1 ### 회원구분 수준
2 col_levels = {0:'정회원', 1:'일반회원', 2:'비회원'}
3 col_levels[1]
```

'일반회원'

```
1 ### 변수 생성
2 tashu_2020_station['회원구분2'] = [col_levels[i] for i in idx]
3 tashu_2020_station.head(3)
```

	대여 스테이 이션	대여일시	반납 스테이 이션	반납일시	이동 거리	회 원 구 분	대여스테이 이션_이름	반납스테이 이션_이름	회 원 구 분2
0	174	20200101000100	224	20200101001137	640	2	가수원네 거리(전통 시장 입 구)	정림삼거리	비 회 원
1	174	20200101000109	224	20200101001210	640	2	가수원네 거리(전통 시장 입 구)	정림삼거리	비 회 원
3455	117	20200101050735	115	20200101051922	1070	0	동부네거 리	한전 대전 세종충남지 역본부(건 너편)	정 회 원

□ 기술통계량

○ 기술통계량

- 데이터의 요약
- 숫자 또는 그래프를 이용
 - 숫자 요약
 - : 평균, 중앙값, 표준편차, 범위 등
 - 그래프 요약
 - : 막대 그래프, 히스토그램, 상자 그림, 산점도, 선 그림 등

```
1 ### 기술통계량
2 tashu_2020_station.describe(include='all')
```

	대여스 테이션	대여일시	반납스 테이션	반납일시	이동거리	회원구 분	대여스 테이션 _이름	반납 테이 _이
count	604446	604446	595778	604446	595857.000000	604446	604446	595778
unique	262	590970	262	590517	NaN	3	262	262
top	3	20200926162941	3	20200905164703	NaN	2	한밭수 목원1	한밭 수목원
freq	25476	4	27709	4	NaN	388611	25476	27709
mean	NaN	NaN	NaN	NaN	1654.610620	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	1446.016779	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	0.000000	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	530.000000	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	1290.000000	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	2560.000000	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	5630.000000	NaN	NaN	NaN

3 그래프 패키지

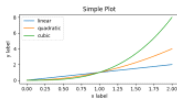
□ Python 패키지

○ matplotlib

- Python에서 정적, 애니메이션 및 대화형 시각화를 작성하기 위한 라이브러리
 - <https://matplotlib.org/>
- 튜토리얼
 - <https://matplotlib.org/tutorials/index.html>

Introductory

These tutorials cover the basics of creating visualizations with Matplotlib, as well as some best-practices in using the package effectively.



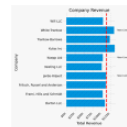
Basic Usage



Pyplot tutorial

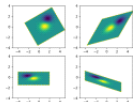


Image tutorial

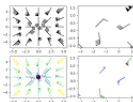


The Lifecycle of a Plot

Images, contours and fields



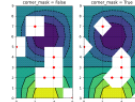
Affine transform of an image



Wind Barbs

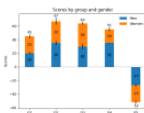


Barcode

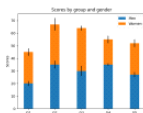


Contour Corner Mask

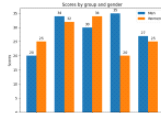
Lines, bars and markers



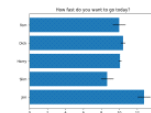
Bar Label Demo



Stacked bar chart

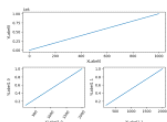


Grouped bar chart with labels

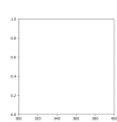


Horizontal bar chart

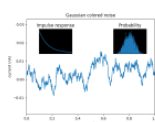
Subplots, axes and figures



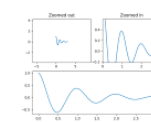
Aligning Labels



Axes box aspect

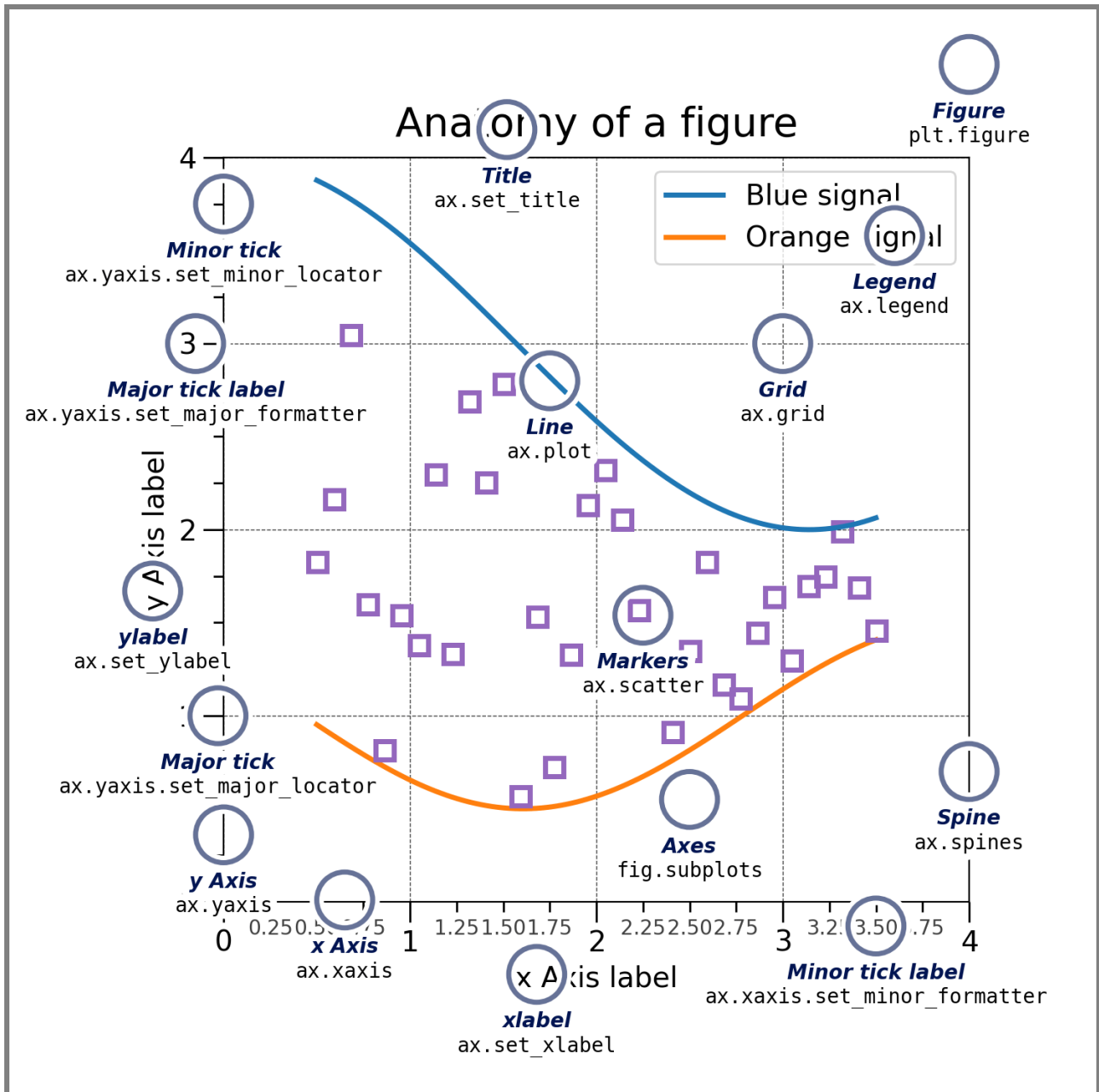


Axes Demo



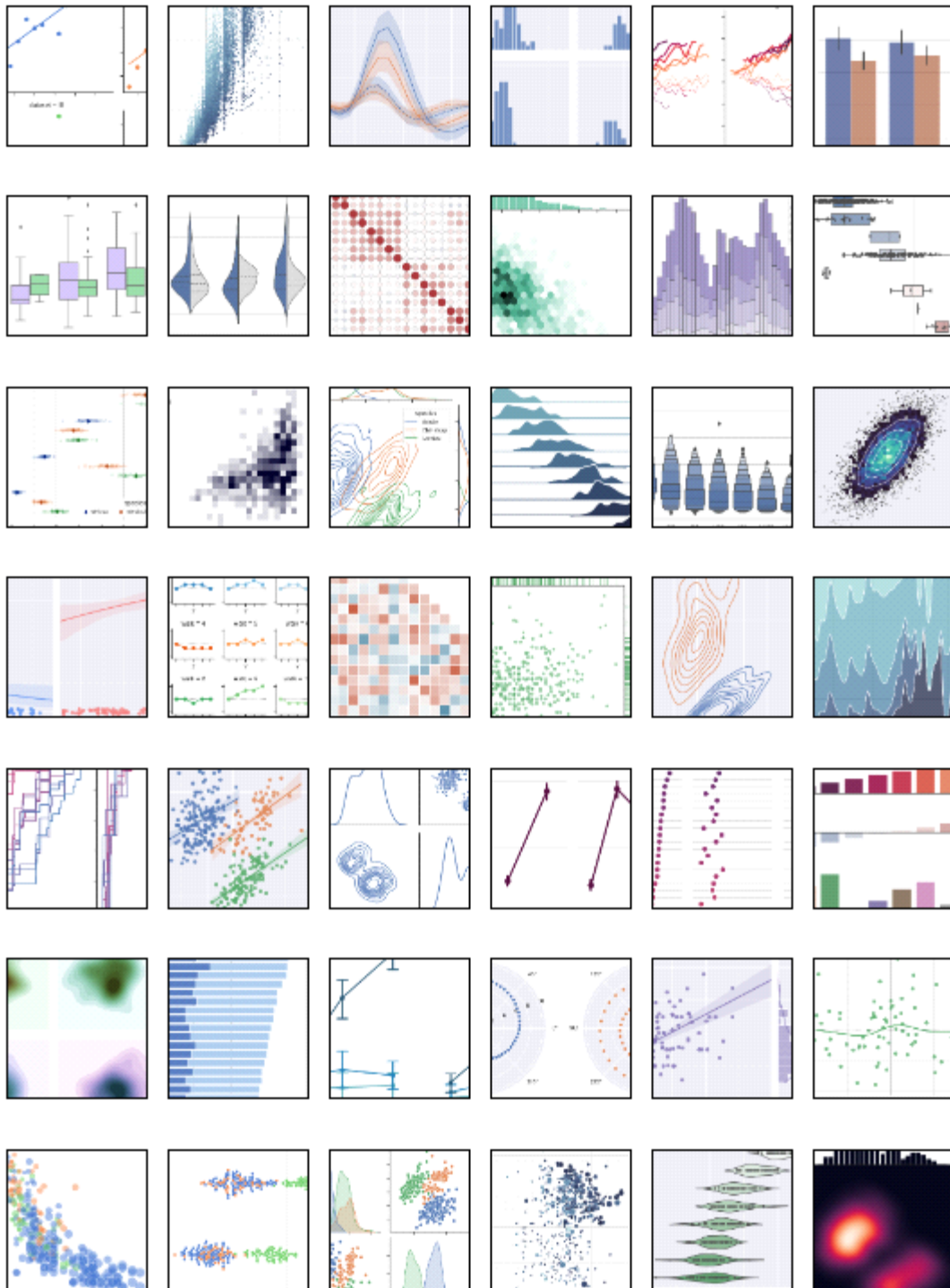
Controlling view limits using margins and sticky_edges

○ 그래프 명칭



- seaborn

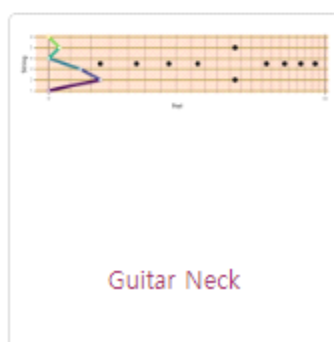
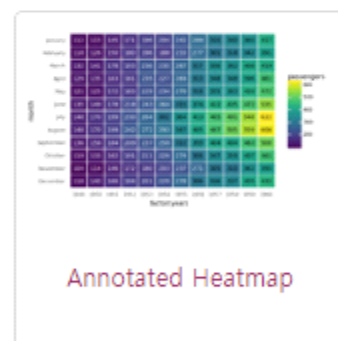
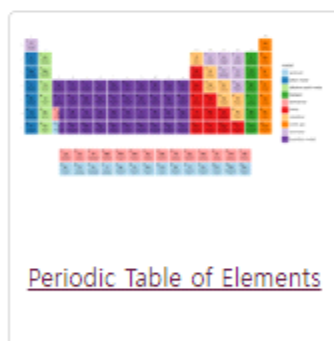
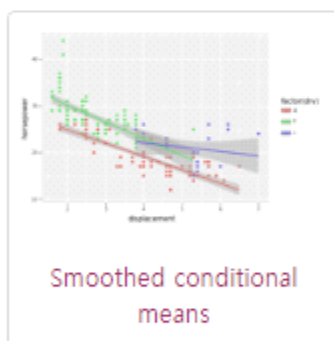
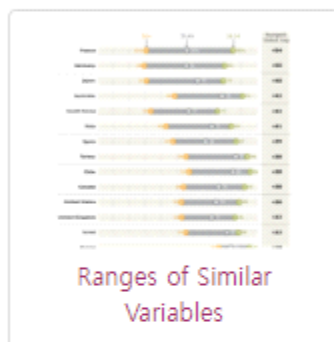
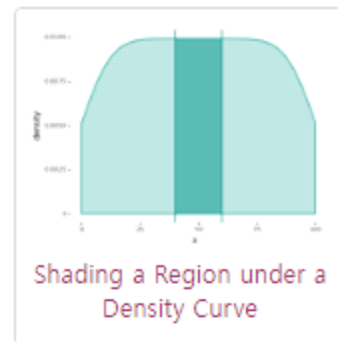
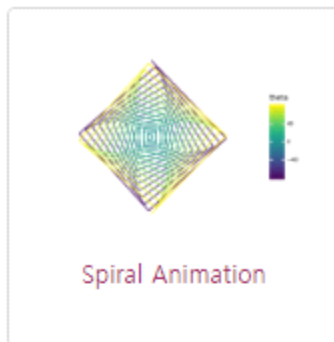
- matplotlib에 기반한 시각화 라이브러리
- 통계 그래픽을 위한 고급 인터페이스를 제공
 - <https://seaborn.pydata.org/>



○ plotnine









– R의 시각화 패키지인 ggplot2에 기반한 파이썬 그래픽 라이브러리

- <https://plotnine.readthedocs.io/en/stable/>



○ plotly

- JavaScript 라이브러리 위에 구축되어 jupyter notebook에 표시하거나, 독립적인 HTML 파일로 저장하여 대화형 웹 기반 시각화를 생성
- 오픈소스 지원을 통해 다양한 그래픽을 제공하는 라이브러리
- Graphing Libraries
 - Python: <https://plotly.com/python/>
 - R: <https://plotly.com/r/>
 - JavaScript: <https://plotly.com/javascript/>

 <p>Plotly Python Open Source Graphing Library</p> <p>Star 11,725</p>	 <p>Plotly R Open Source Graphing Library</p> <p>Star 2,213</p>	 <p>Plotly Julia Open Source Graphing Library</p> <p>Star 325</p>	 <p>Plotly Javascript Open Source Graphing Library</p> <p>Star 14,804</p>
 <p>Plotly ggplot2 Open Source Graphing Library</p> <p>Star 2,213</p>	 <p>Plotly F# Open Source Graphing Library</p> <p>Star 333</p>	 <p>Plotly MATLAB® Open Source Graphing Library</p> <p>Star 311</p>	 <p>Plotly Dash Open Source Analytical App Framework</p> <p>Star 16,751</p>

Fundamentals

More Fundamentals »



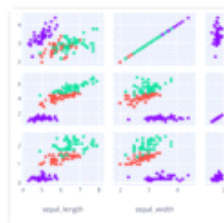
The Figure Data Structure



Creating and Updating Figures



Displaying Figures



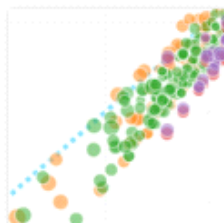
Plotly Express



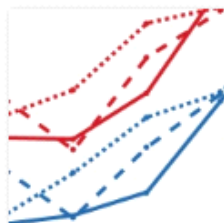
Analytical Apps with Dash

Basic Charts

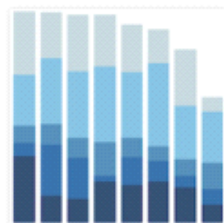
More Basic Charts »



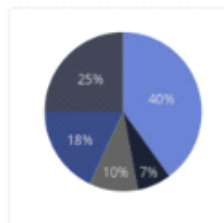
Scatter Plots



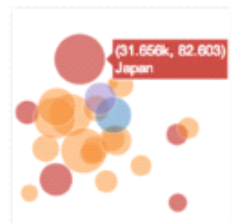
Line Charts



Bar Charts



Pie Charts



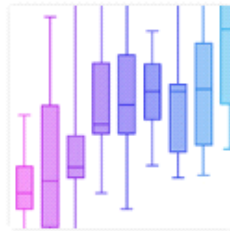
Bubble Charts

Statistical Charts

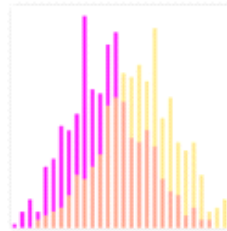
[More Statistical Charts »](#)



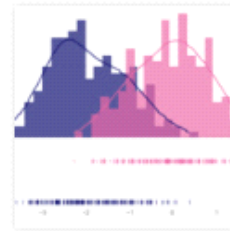
Error Bars



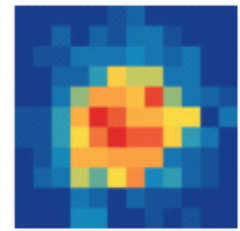
Box Plots



Histograms



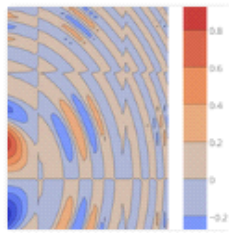
Distplots



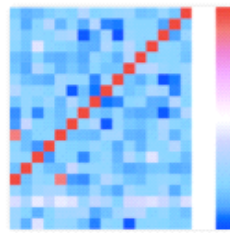
2D Histograms

Scientific Charts

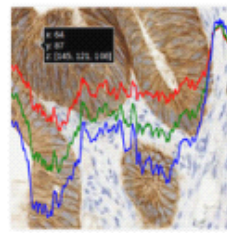
[More Scientific Charts »](#)



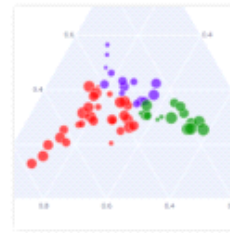
Contour Plots



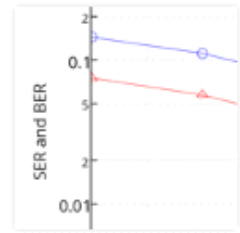
Heatmaps



Imshow



Ternary Plots



Log Plots

Financial Charts

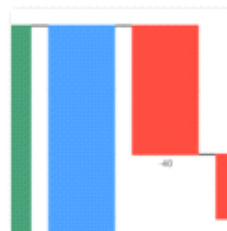
[More Financial Charts »](#)



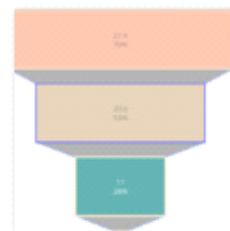
Time Series and Date Axes



Candlestick Charts



Waterfall Charts



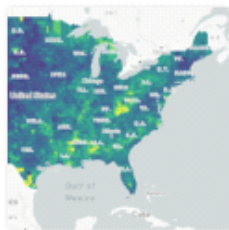
Funnel Chart



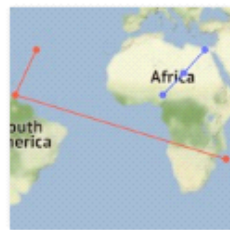
OHLC Charts

Maps

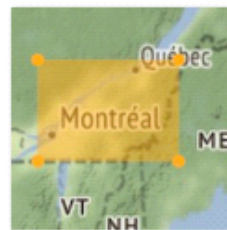
[More Maps »](#)



Mapbox Choropleth Maps



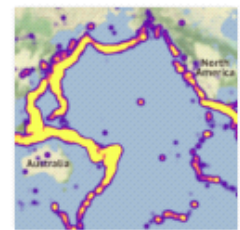
Lines on Mapbox



Filled Area on Maps



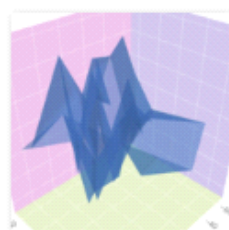
Bubble Maps



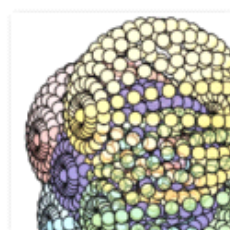
Mapbox Density Heatmap

3D Charts

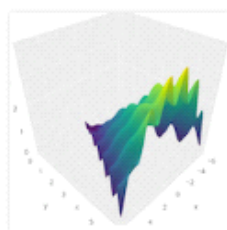
[More 3D Charts »](#)



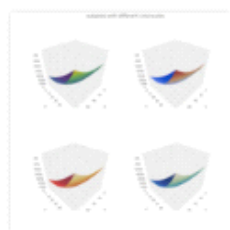
3D Axes



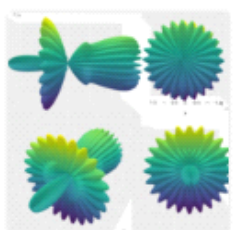
3D Scatter Plots



3D Surface Plots



3D Subplots



3D Camera Controls

4 그래프 요약

□ Python 패키지

▶ matplotlib

```
1 ### Package
2 from matplotlib import pyplot as plt
```

```
1 ### 그래프에 한글 표시를 위한 글꼴 설정
2 plt.rcParams['font.family'] = 'Gulim'      # 'AppleGothic' in mac
3 ### 스타일
4 plt.style.use('ggplot')
```

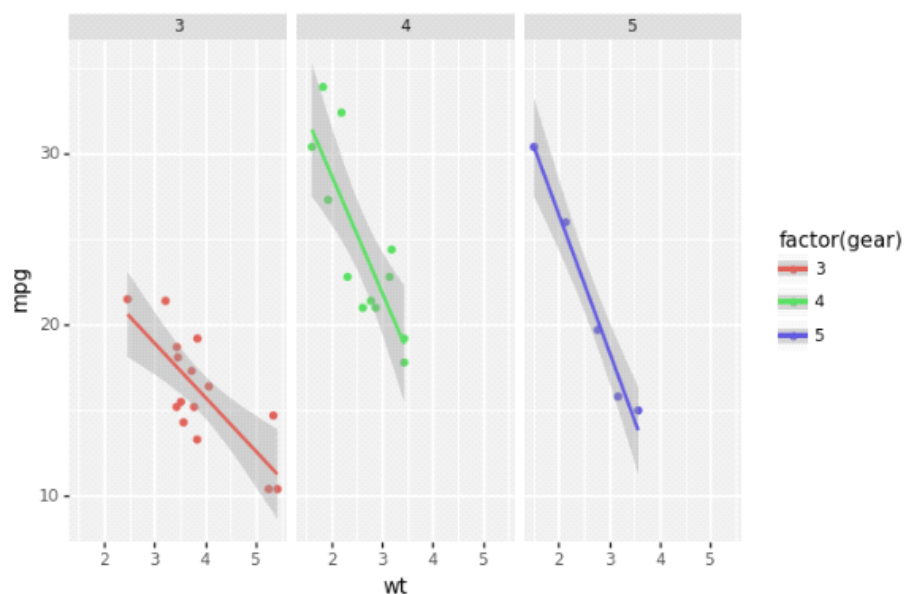
▶ seaborn

```
1 ### Package
2 import seaborn as sns
```

▶ plotnine

```
1 from plotnine import *
```

```
1 from plotnine.data import mtcars
2
3 (
4     ggplot(mtcars, aes('wt', 'mpg', color='factor(gear)'))
5     + geom_point()
6     + stat_smooth(method='lm')
7     + facet_wrap('~gear')
8 )
```



□ 단일 변수의 탐색

- 범주형/이산형 자료

```
1 ### 타슈 스테이션 현황: 대전광역시_공영자전거(타슈) 위치(위경도) 현황  
   _20200801.csv  
2 data_file = "./data/타슈/대전광역시_공영자전거(타슈) 위치(위경도) 현황  
   _20200801.csv"  
3 tashu_station = pd.read_csv(data_file, encoding = "cp949")
```

- 빈도표

- 값이 발생한 빈도나 횟수를 포함하고 있으며, 이러한 방식으로 표는 표본 값의 분포를 요약

```
1 ### 빈도표  
2 tb = pd.crosstab(tashu_station['시군구명'], 'Freq', colnames=[''])  
3 tb
```

Freq	
시군구명	
대덕구	38
동구	36
서구	73
유성구	76
중구	39

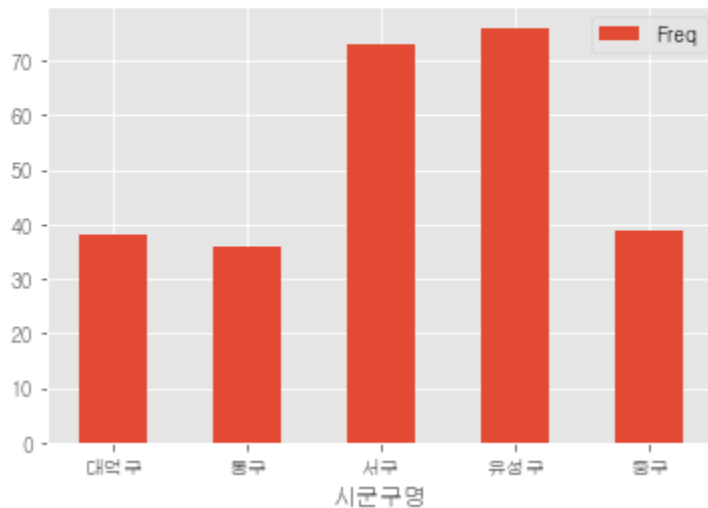
```
1 ### 빈도표 - 내림차순 정렬  
2 tb_sort = tb.sort_values('Freq', ascending=False)  
3 tb_sort
```

Freq	
시군구명	
유성구	76
서구	73
중구	39
대덕구	38
동구	36

- 막대 그래프(bar plot)

- 막대 그래프(-graph), 바 차트(bar chart), 바 그래프(bar graph)는 표현 값에 비례하여 높이와 길이를 지닌 직사각형 막대로 범주형 데이터를 표현하는 차트나 그래프
- 막대는 수직으로나 수평으로 그릴 수 있음

```
1 ### 막대 그래프 - 수직(기본)
2 ax = tb.plot.bar(rot=0)
```



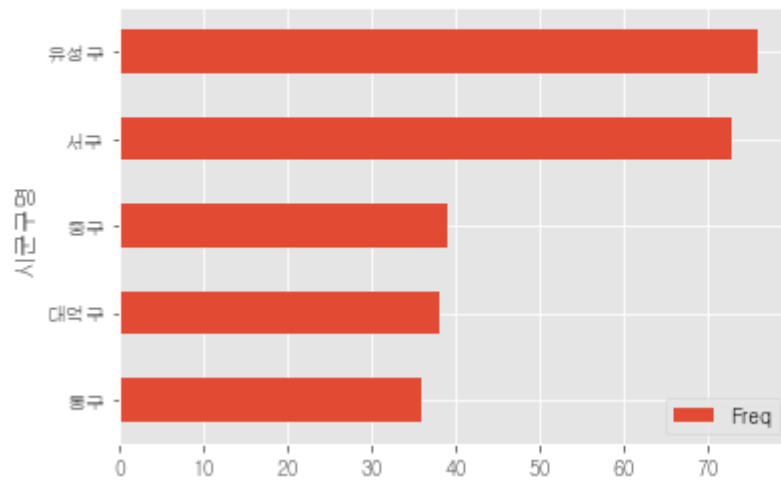
```
1 ### 막대 그래프 - 수직(디자인)
2 ax = tb_sort.plot.bar(rot=0, color="royalblue") # 색상
3 ax.set_title("타슈 스테이션 현황") # 제목
4 ax.set_xlabel("") # x축 이름
5 ax.set_ylabel("스테이션 수") # y축 이름
6 ax.legend().remove() # 범례 제거
7 ax.set_facecolor('white') # 배경색
8 ax.grid(True, axis='y', color="silver") # 눈금선(y축)
9 ax.tick_params(left=False, bottom=False) # 눈금 제거
10 ax.spines['bottom'].set_color('black') # x축 표시
11 plt.show()
```



```

1  ### 막대그래프 - 수평(기본)
2  tb_sort_asc = tb.sort_values('Freq', ascending=True)
3  ax = tb_sort_asc.plot.barh()

```



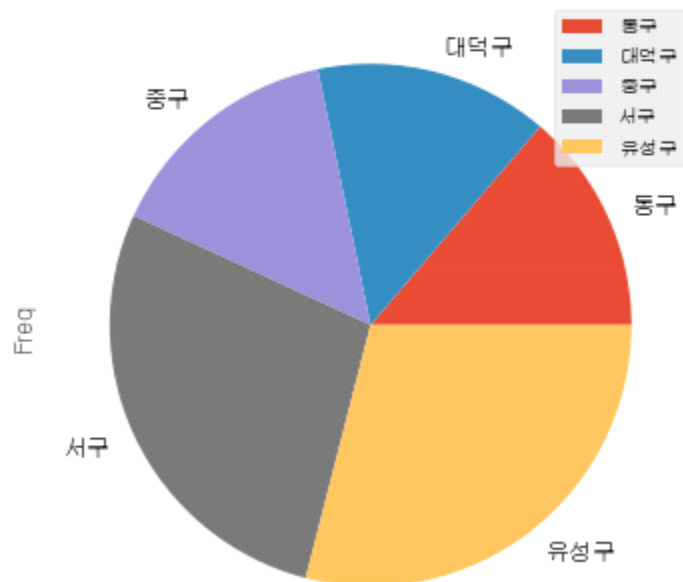
- 원 그래프(Pie chart)

- 전체에 대한 각 부분의 비율을 부채꼴 모양으로 나타낸 그래프

```

1  ### 원그래프
2  ax = tb_sort_asc.plot.pie(y = 'Freq', figsize = (6, 6), fontsize = 12)

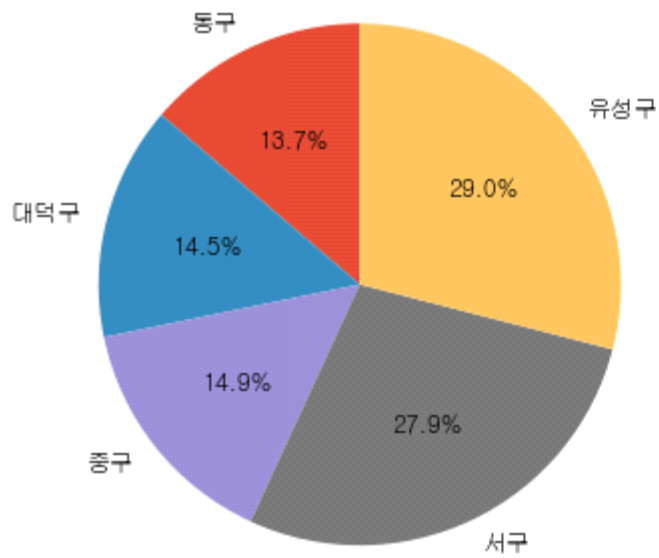
```



```

1  ### 원그래프
2  ax = tb_sort_asc.plot.pie(y = 'Freq', figsize = (6, 6), fontsize = 12,
3                           startangle=90, autopct = '%.1f%%')
4  ax.legend().remove()
5  ax.set_ylabel("")
6  plt.show()

```



○ 연속형 자료

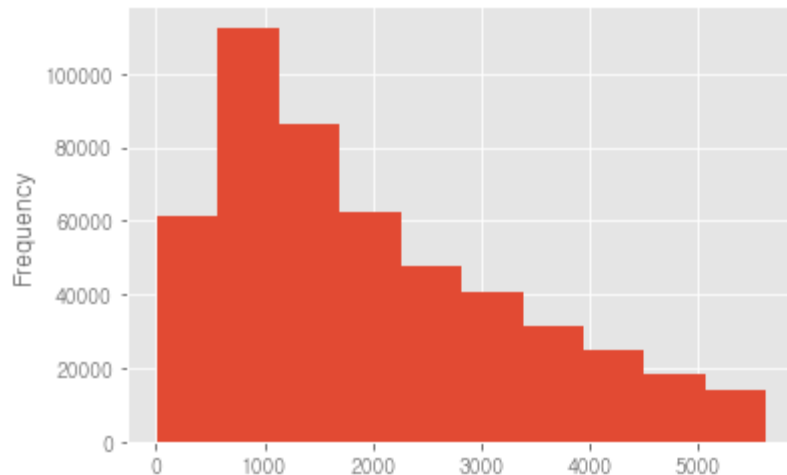
```
1 ### Drop missing values
2 df_use = tashu_2020_station.dropna(subset=['이동거리'])
3 ### Subset
4 df_use = df_use[df_use['이동거리'] > 0]
5 df_use.shape
```

(499409, 9)

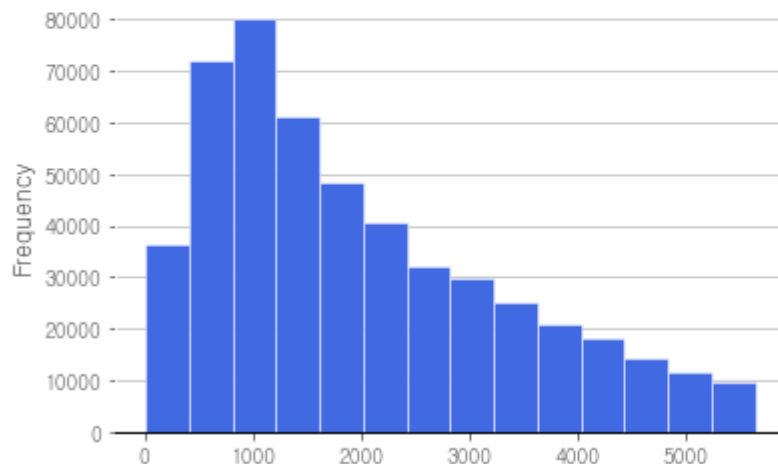
– 히스토그램(histogram)

- 표로 되어 있는 도수 분포를 정보 그림으로 나타낸 것
- 계급은 보통 변수의 구간이고, 서로 겹치지 않음

```
1 ### 히스토그램(기본)
2 ax = df_use['이동거리'].plot.hist()
```



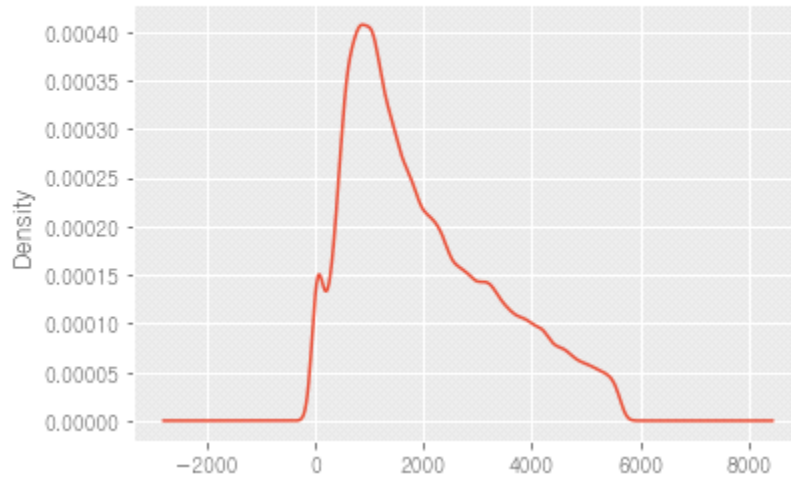
```
1 ### 히스토그램(디자인)
2 ax = df_use['이동거리'].plot.hist(bins=14, color="royalblue", edgecolor="w")
3 ax.set_facecolor('white') # 배경색
4 ax.grid(True, axis='y', color="silver") # 눈금선(y축)
5 ax.spines['bottom'].set_color('black') # x축 표시
```



- 분포(density)

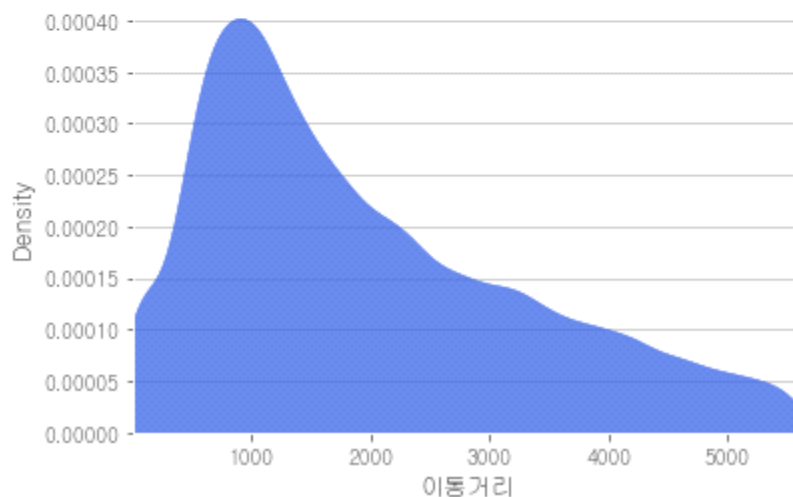
- 주어진 자료를 기반으로 밀도를 추정(kernel density estimate (KDE) plot)

```
1 ### 분포
2 ax = df_use['이동거리'].plot.density()
```



```
1 ### 기술 통계량
2 dist_stats = df_use['이동거리'].describe()
```

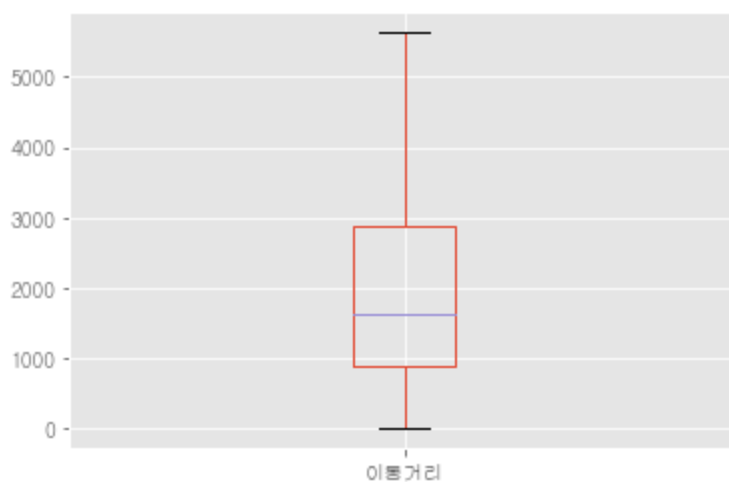
```
1 # KDE Plot with seaborn
2 data = df_use['이동거리'].astype('float')
3 ax = sns.kdeplot(data, color='royalblue', shade='True',
4                 bw_adjust=1.5, alpha=0.8)
5 ax.set_facecolor('white') # 배경색
6 ax.grid(True, axis='y', color="silver") # 눈금선(y축)
7 plt.xlim(dist_stats['min'], dist_stats['max']) # x축 범위 제한
8 plt.show()
```



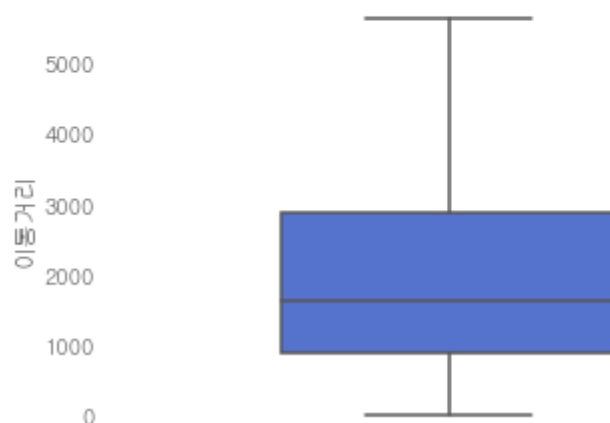
- 상자 그림(box plot)

- 자료로부터 얻어낸 통계량 최솟값, 제 1사분위(Q1), 제 2사분위(Q2), 제 3사분위(Q3), 최댓값을 이용한 그림
- 제 1사분위 수 (Q1) : 중앙값 기준으로 하위 50% 중의 중앙값, 전체 데이터 중 하위 25%에 해당하는 값
- 중위수 : 데이터의 정 가운데 순위에 해당하는 값
- 제 3사분위 수 (Q3) : 중앙값 기준으로 상위 50% 중의 중앙값, 전체 데이터 중 상위 25%에 해당하는 값
- 사분위 범위(IQR) : 데이터의 중간 50% ($Q3 - Q1$)

```
1 ### 상자 그림(기본)
2 ax = df_use['이동거리'].plot.box()
```



```
1 ### 상자 그림
2 ax = sns.boxplot(y=df_use['이동거리'].astype('float'),
3                 width=0.5, color='royalblue')
4 ax.set_facecolor('white') # 배경색
5 #ax.grid(True, axis='y', color='silver') # 눈금선(y축)
6 plt.show()
```

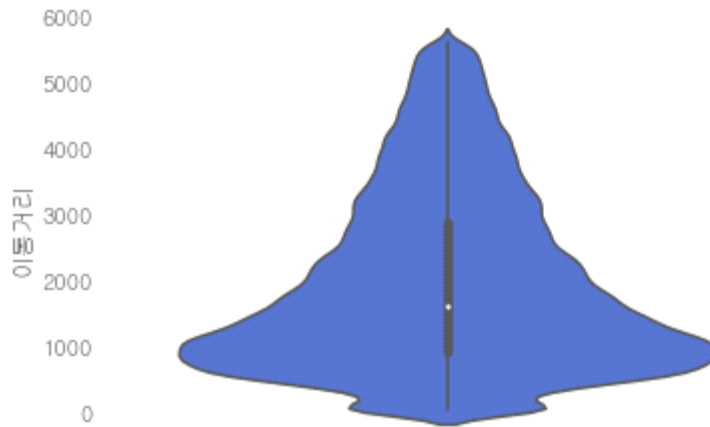


- 바이올린 그림(violin plot)
 - 상자 그림에 분포가 포함된 그래프

```

1  ### violin plot
2  ax = sns.violinplot(y = df_use['이동거리'].astype('float'),
3                      color='royalblue')
4  ax.set_facecolor('white')           # 배경색
5  plt.show()

```



- 선 그림(line plot)
 - 수량을 점으로 표시하고 그 점들을 선으로 이어 그린 그래프
 - 연속적인 값들의 변동을 파악하기 쉬움

```

1  ### 날짜형 변수 생성 - 대여일시, 반납일시
2  #df_use['대여일시'] = df_use['대여일시'].astype('str')
3  df_use['대여일시_datetime'] = pd.to_datetime(df_use['대여일시'])
4  #df_use['반납일시'] = df_use['반납일시'].astype('str')
5  df_use['반납일시_datetime'] = pd.to_datetime(df_use['반납일시'])

```

```

1  ### 대여일 변수 생성
2  df_use['대여일'] = df_use['대여일시_datetime'].dt.date

```

```

1  ### 대여시간 변수 생성
2  diff_dt = df_use['반납일시_datetime'] - df_use['대여일시_datetime']
3  df_use['대여시간_초'] = diff_dt.dt.total_seconds()

```

```
1 df_use.head(1)
```

대여일시	반납스테이션	반납일시	이동거리	회원구분	대여스테이션_이름	반납스테이션_이름	회원구분2	대여일시_datetime	반납일시_datetime	대여일	대여시간_초
20200101000100	224	20200101001137	640	2	가수원네거리(전통시장입구)	정림삼거리	비회원	2020-01-01 00:01:00	2020-01-01 00:11:37	2020-01-01	637.0

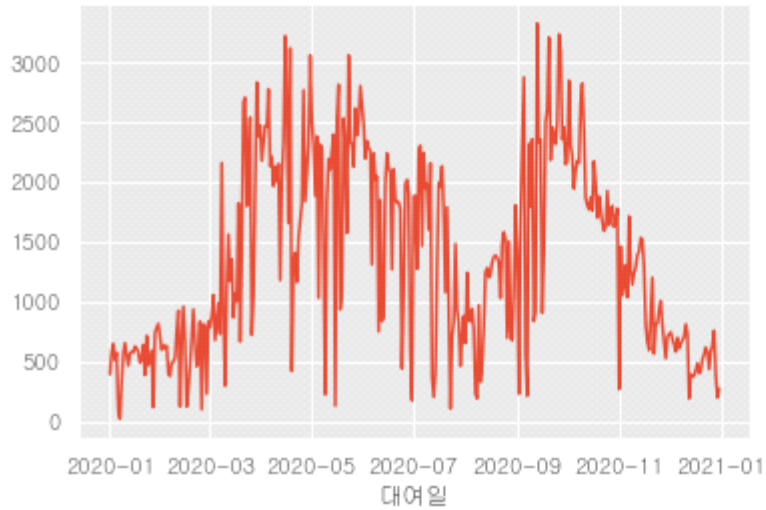
```
1 ### 대여일 별 대여수 & 대여시간(초)의 합
2 df_group_date = df_use.groupby(['대여일']).apply(
3     lambda x: pd.Series(
4         {'대여수': len(x),
5          '대여시간_합': sum(x.대여시간_초) / 60**2
6         })
7 df_group_date.head(3)
```

대여일	대여수	대여시간_합
2020-01-01	394.0	178.185833
2020-01-02	582.0	217.198056
2020-01-03	650.0	254.860278

```

1  ### 선 그림(기본)
2  ax = df_group_date['대여수'].plot.line()

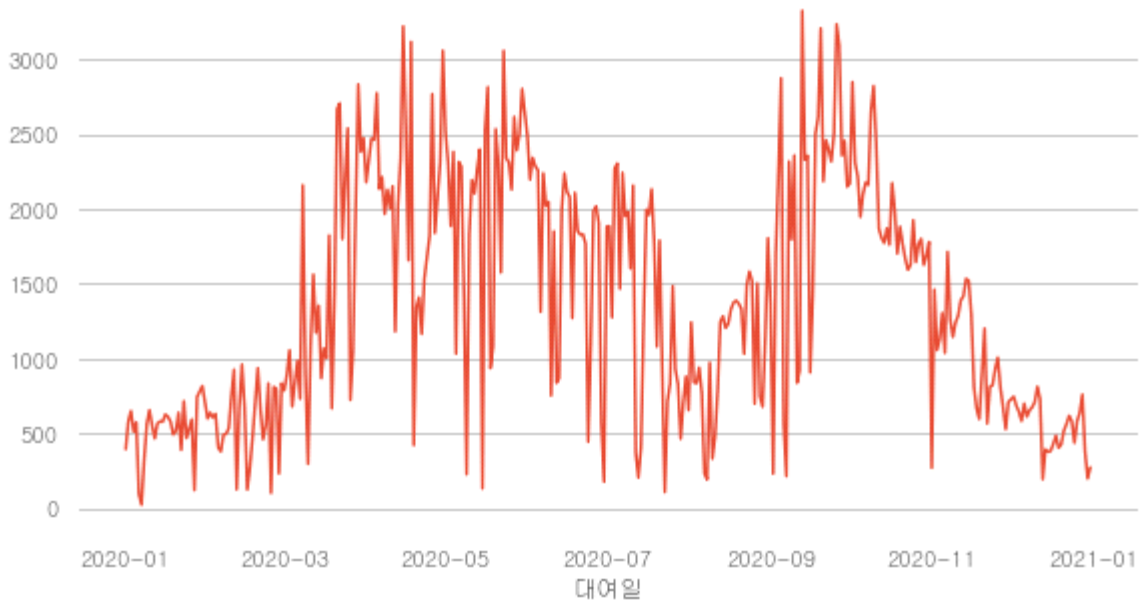
```



```

1  ### 선 그림
2  ax = df_group_date['대여수'].plot.line(figsize = (9.5, 5)) # [6.4, 4.8]
3  ax.set_facecolor('white') # 배경색
4  ax.grid(True, axis='y', color="silver") # 눈금선(y축)
5  plt.show()

```



□ 두 변수의 탐색

```

1  ### 자료 추출
2  cnd = (df_use['대여스테이션'].astype('Int64') <= 5) & (df_use['반납스테이션'].astype('Int64') <= 5)
3  df_sub = df_use[cnd]
4  df_sub.shape

```

(14740, 13)

○ 범주형 & 범주형 자료

– 막대그래프(barplot)

- 교차표를 이용하여 각 범주별 빈도를 확인

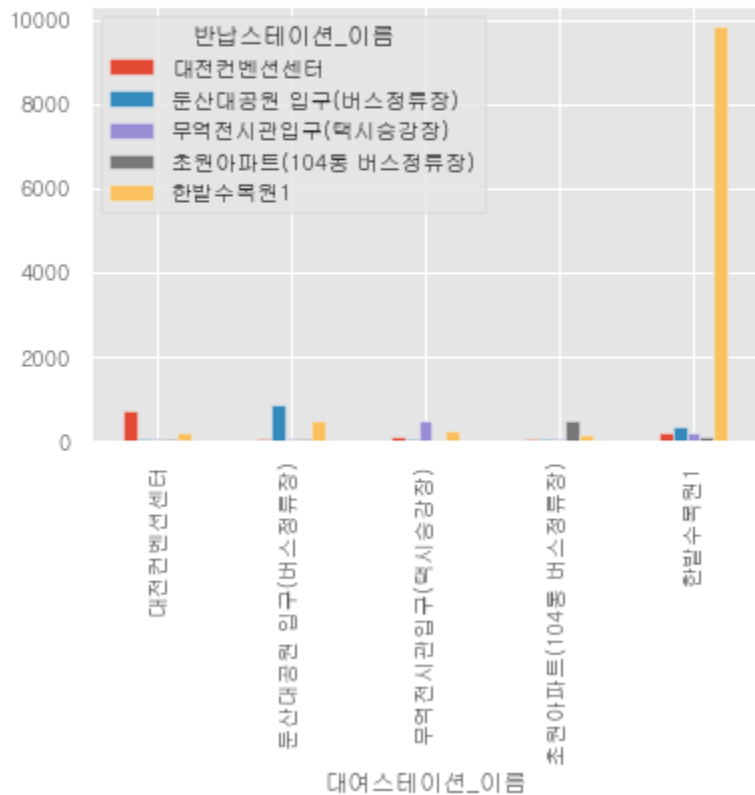
```

1  ### 교차표
2  tb = pd.crosstab(df_sub['대여스테이션_이름'], df_sub['반납스테이션_이름'])
3  tb

```

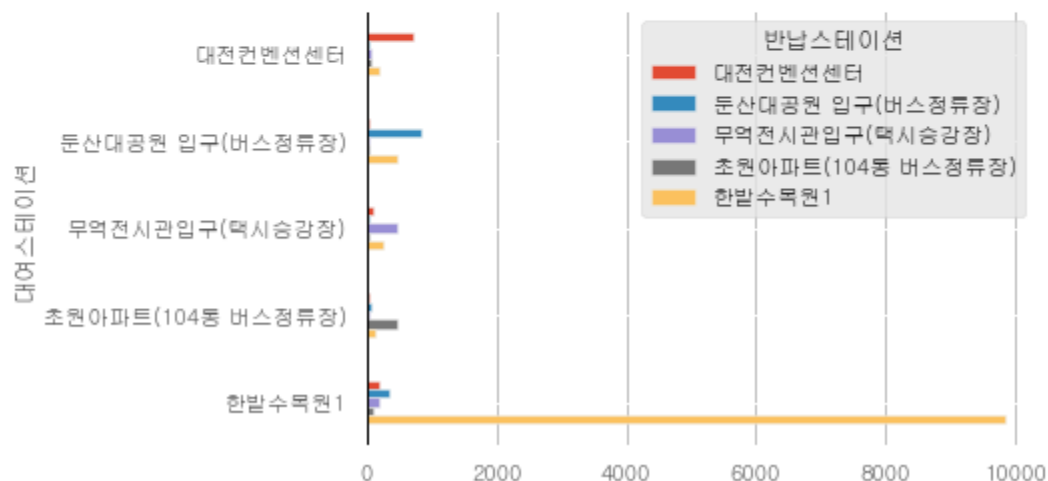
반납스테이션_이름	대전컨벤션센터	둔산대공원 입구 (버스정류장)	무역전시관입구 (택시승강장)	초원아파트(104동 버스정류장)	한밭수목원1
대여스테이션_이름					
대전컨벤션센터	714	41	61	58	175
둔산대공원 입구 (버스정류장)	43	847	32	35	459
무역전시관입구(택시승강장)	94	47	473	30	242
초원아파트(104동 버스정류장)	43	56	32	478	129
한밭수목원1	187	351	180	97	9836

```
1 ### 막대그래프
2 ax = tb.plot.bar()
```



```
1 ### 막대그래프 - 수평
2 ax = tb.plot.barh()
3 ax.set_facecolor('white')
4 ax.grid(True, axis='x', color="silver")
5 ax.spines['left'].set_color('black')
6 ax.set_ylabel("대여스테이션")
7 ax.legend(title='반납스테이션')
8 ax.invert_yaxis()
9 plt.show()
```

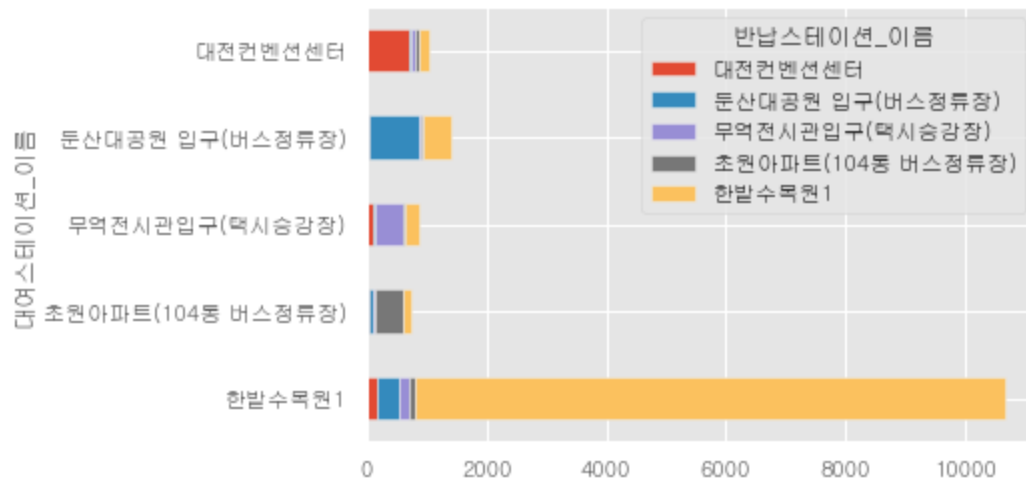
배경색
눈금선(x축)
y축 표시
y축 이름
범례 제목
y축 순서 반전



```

1  ### 막대그래프 - 누적
2  ax = tb.plot.barh(stacked = True)
3  ax.invert_yaxis()                                # y축 순서 반전
4  plt.show()

```



- 모자이크 도표(mosaic plot)

- 특수한 유형의 누적 막대 차트
- 막대의 가로길이는 가로축에 표시된 변수의 수준별 관측치 수에 비례하고, 세로 길이는 첫 번째 변수의 각 수준 내에서 두 번째 변수의 관측치 수에 비례

```

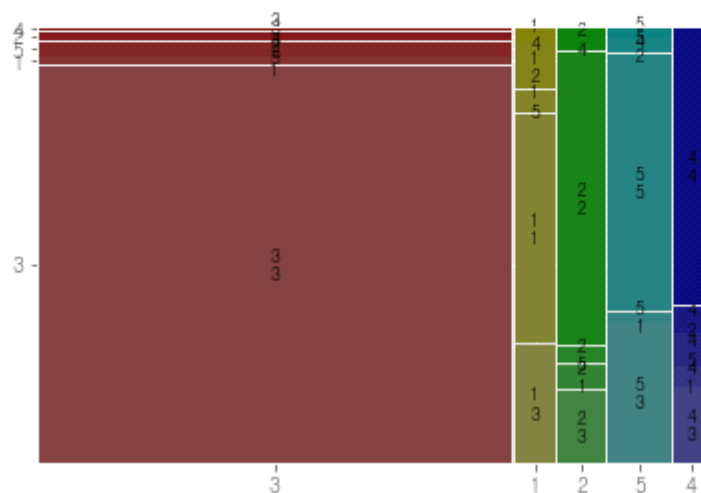
1  ### Packages
2  from statsmodels.graphics.mosaicplot import mosaic

```

```

1  ### mosaic plot(기본)
2  ax = mosaic(df_sub, ['대여스테이션', '반납스테이션'])

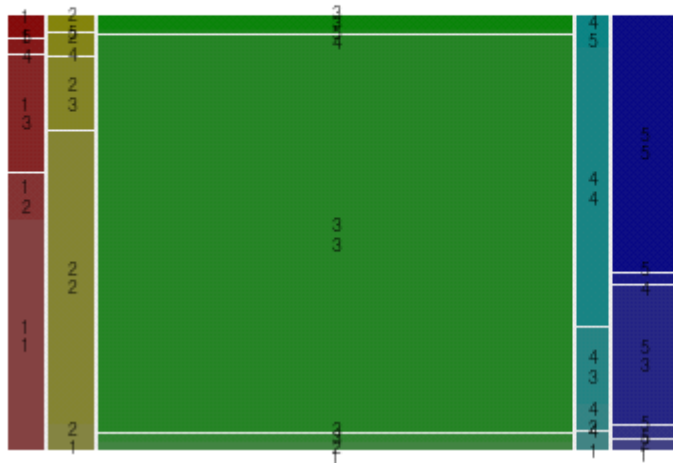
```




```

1  ### mosaic plot
2  ax = mosaic(df_sub.sort_values(['대여스테이션', '반납스테이션']),
3             ['대여스테이션', '반납스테이션'],
4             axes_label=False)

```



○ 범주형 & 연속형 자료

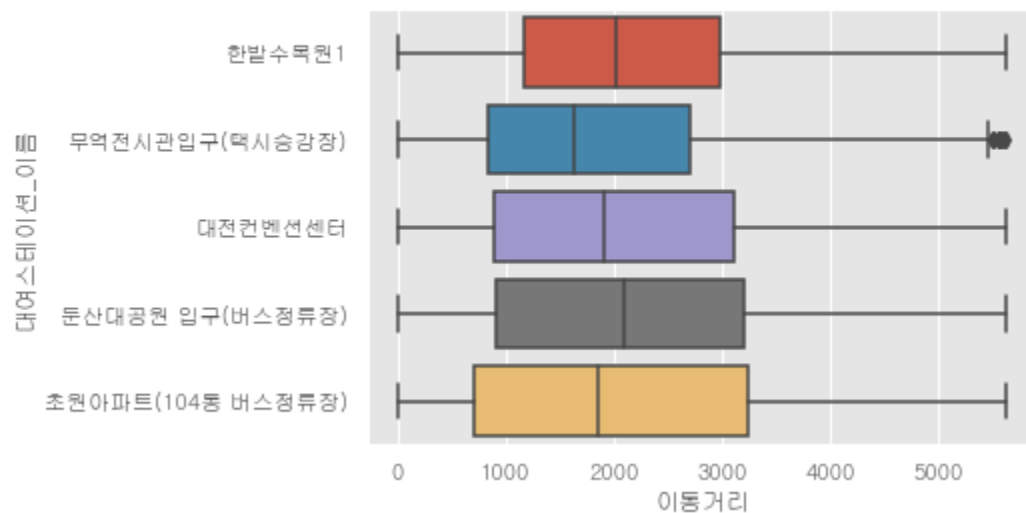
– 상자그림(boxplot)

- 범주형 변수의 값을 각 수준별로 나누어 연속형 변수에 대한 상자그림을 그림
- 집단별 분포를 비교

```

1  ### 상자그림 - 대여시간_초 by 대여스테이션_이름
2  ax = sns.boxplot(x="이동거리", y="대여스테이션_이름", data=df_sub)

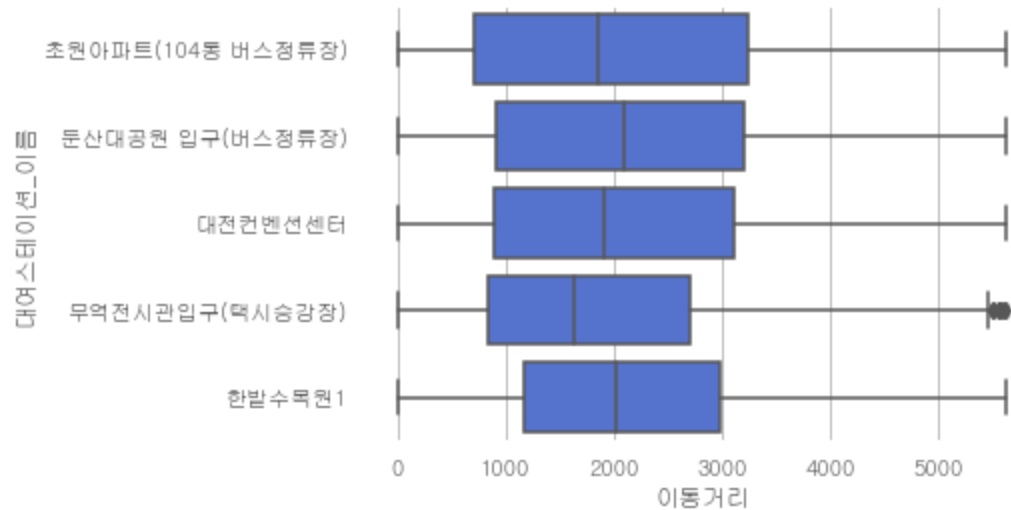
```



```

1  ### 상자그림 - 대여시간_초 by 대여스테이션_이름
2  ax = sns.boxplot(x="이동거리", y="대여스테이션_이름",
3                  data=df_sub, color="royalblue")
4  ax.set_facecolor('white') # 배경색
5  ax.grid(True, axis='x', color="silver") # 눈금선(x축)
6  ax.invert_yaxis() # y축 순서 반전
7  plt.show()

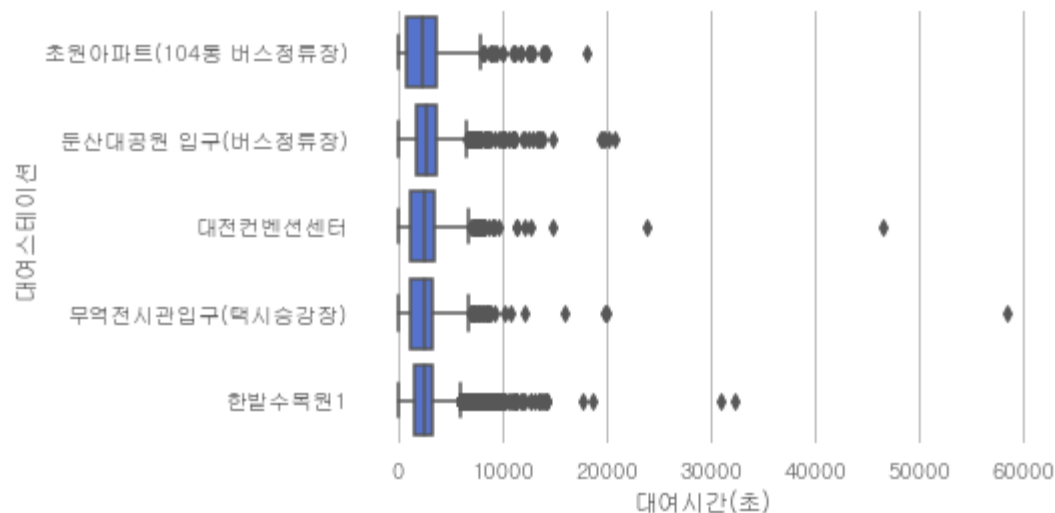
```



```

1  ### 상자그림 - 대여시간_초 by 대여스테이션_이름
2  ax = sns.boxplot(x="대여시간_초", y="대여스테이션_이름",
3                  data=df_sub, color="royalblue")
4  ax.set_xlabel("대여시간(초)") # x축 이름
5  ax.set_ylabel("대여스테이션") # y축 이름
6  ax.set_facecolor('white') # 배경색
7  ax.grid(True, axis='x', color="silver") # 눈금선(x축)
8  ax.invert_yaxis() # y축 순서 반전
9  plt.show()

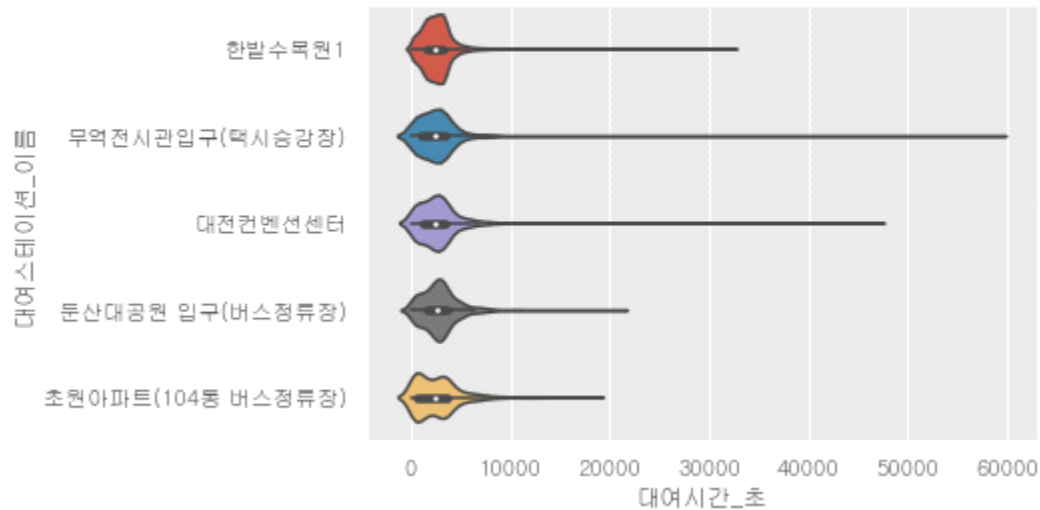
```



- 바이올린 그림(violin plot)

- 범주형 변수의 값을 각 수준별로 나누어 연속형 변수에 대한 바이올린 그림을 그림
- 집단별 분포를 비교

```
1 ### violin plot - 대여시간_초 by 대여스테이션_이름
2 ax = sns.violinplot(x="대여시간_초", y="대여스테이션_이름", data=df_sub)
```



```
1 ### violin plot - 대여시간_초 by 대여스테이션_이름
2 ax = sns.violinplot(x=df_sub.대여시간_초 / 60**2,
3                     y=df_sub.대여스테이션_이름,
4                     color="royalblue")
5 ax.set_xlabel("대여시간(시)")           # x축 이름
6 ax.set_ylabel("대여스테이션")          # y축 이름
7 ax.set_facecolor('white')              # 배경색
8 plt.xticks(range(0,18,2))               # x축 눈금
9 ax.grid(True, axis='x', color="silver") # 눈금선(x축)
10 plt.xlim(0, 20)                        # x축 범위 제한
11 ax.spines['left'].set_color('silver')   # y축 표시
12 plt.show()
```



○ 연속형 & 연속형 자료

```

1  ### 대여일 별 대여수 & 대여시간(초) & 이동거리 기초통계량
2  df_group_date = df_use.groupby(['대여일']).apply(
3      lambda x: pd.Series({
4          '대여수': len(x),
5          '대여시간_합': sum(x.대여시간_초),
6          '이동거리_합': sum(x.이동거리),
7          '이동거리_평균': np.mean(x.이동거리),
8          '이동거리_최솟값': np.min(x.이동거리),
9          '이동거리_중앙값': np.median(x.이동거리),
10         '이동거리_최댓값': np.max(x.이동거리)
11     })
12  df_group_date.head(3)

```

대여일	대여수	대여시간_합	이동거리_합	이동거리_평균	이동거리_최솟값	이동거리_중앙값	이동거리_최댓값
2020-01-01	394.0	641469.0	676070.0	1715.913706	10.0	1355.0	5370.0
2020-01-02	582.0	781913.0	913260.0	1569.175258	10.0	1210.0	5560.0
2020-01-03	650.0	917497.0	1011860.0	1556.707692	10.0	1245.0	5630.0

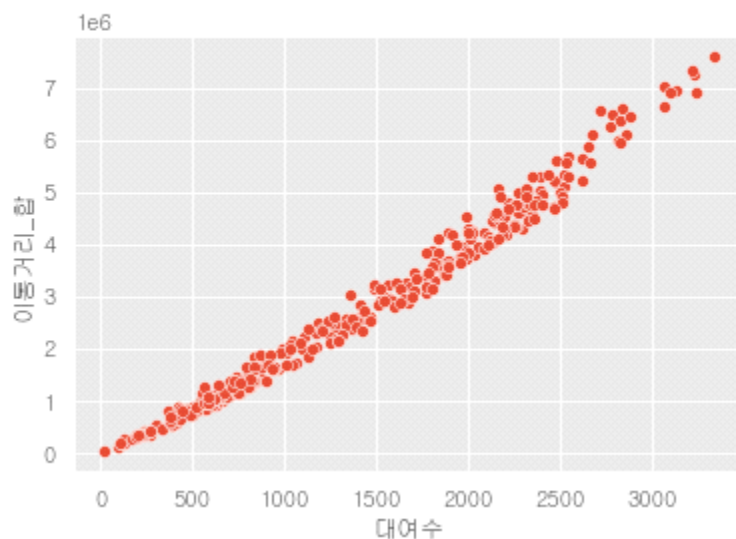
- 산점도(Scatter plot)

- 쌍(pair)으로 존재하는 연속형 자료를 좌표평면에 점으로 표시
- 두 개 변수 간의 관계를 나타내는 방법

```

1  ### 산점도(Scatter plot)
2  ax = sns.scatterplot(x = '대여수', y = '이동거리_합', data = df_group_date)

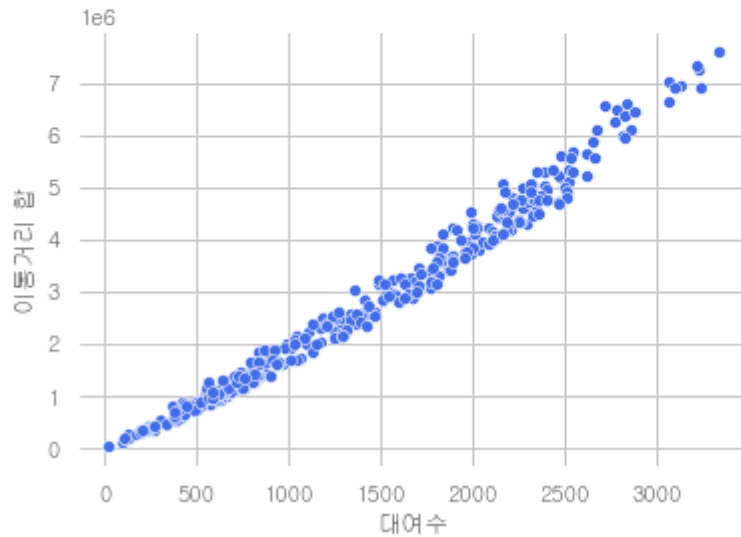
```



```

1  ### 산점도(Scatter plot)
2  ax = sns.scatterplot(x = '대여수', y = '이동거리_합',
3                      data = df_group_date, color="royalblue")
4  ax.set_facecolor('white') # 배경색
5  ax.set_ylabel("이동거리 합") # y축 이름
6  ax.grid(True, color="silver") # 눈금선
7  plt.show()

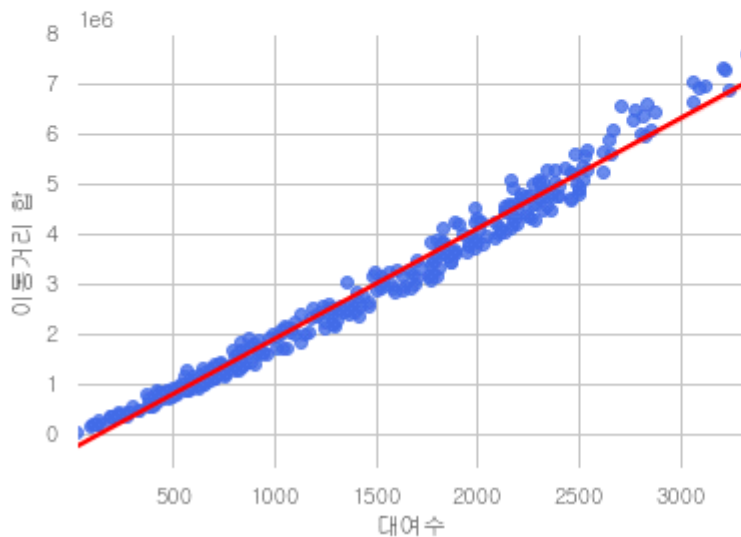
```



```

1  ### 산점도 - 회귀선 추가
2  ax = sns.regplot(x = '대여수', y = '이동거리_합',
3                  data = df_group_date, color="royalblue",
4                  line_kws={"color": "red"})
5  ax.set_facecolor('white') # 배경색
6  ax.set_ylabel("이동거리 합") # y축 이름
7  ax.grid(True, color="silver") # 눈금선
8  plt.show()

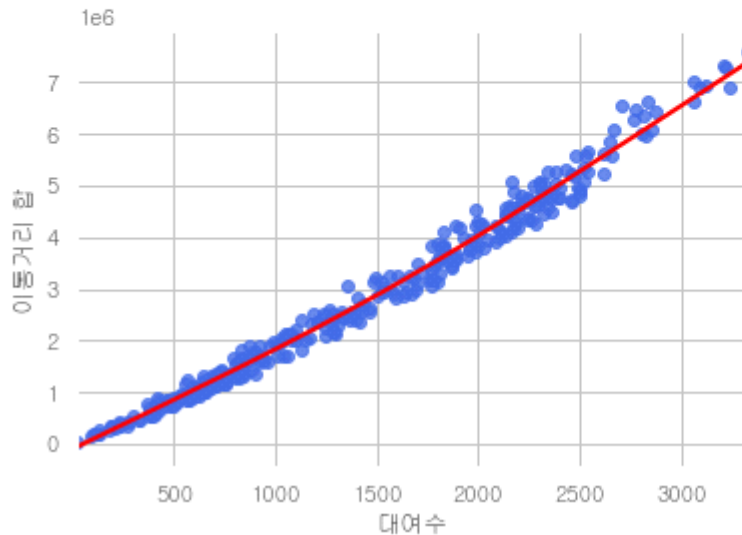
```



```

1  ### 산점도 - LOWESS (locally weighted scatterplot smoothing)
2  ax = sns.regplot(x = '대여수', y = '이동거리_합',
3                  data = df_group_date, color="royalblue",
4                  line_kws={"color": "red"}, lowess=True)
5  ax.set_facecolor('white') # 배경색
6  ax.set_ylabel("이동거리 합") # y축 이름
7  ax.grid(True, color="silver") # 눈금선
8  plt.show()

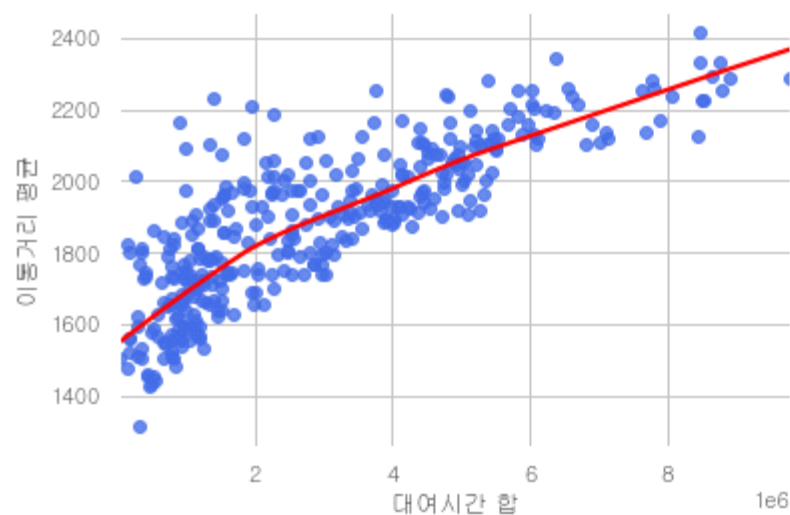
```



```

1  ### 산점도 - LOWESS (locally weighted scatterplot smoothing)
2  ax = sns.regplot(x = '대여시간_합', y = '이동거리_평균',
3                  data = df_group_date, color="royalblue",
4                  line_kws={"color": "red"}, lowess=True)
5  ax.set_facecolor('white') # 배경색
6  ax.set_xlabel("대여시간 합") # x축 이름
7  ax.set_ylabel("이동거리 평균") # y축 이름
8  ax.grid(True, color="silver") # 눈금선
9  plt.show()

```



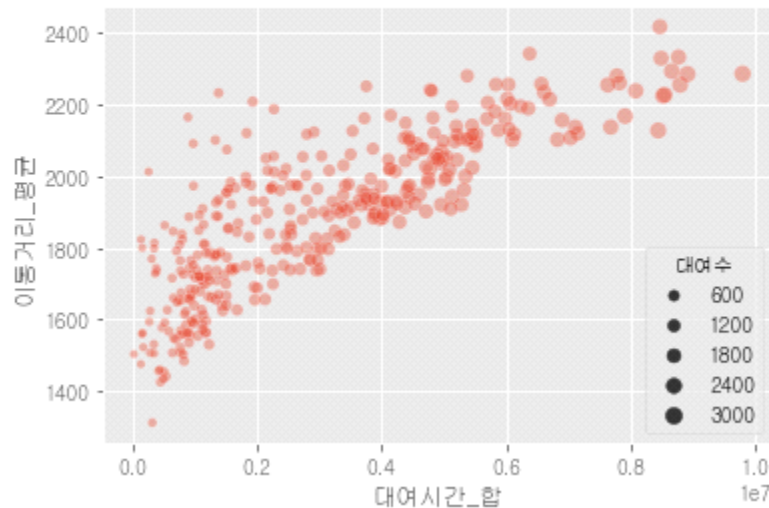
□ 다중 변수의 탐색

- 3개 이상의 변수를 표현한 그래프
 - 버블 차트(bubble chart)
 - 산점도 + 점의 크기

```

1  ### 버블 차트: 산점도 + 점의 크기
2  ax = sns.scatterplot(x = '대여시간_합', y = '이동거리_평균',
3                      data = df_group_date,
4                      size = df_group_date['대여수'], alpha=0.4)

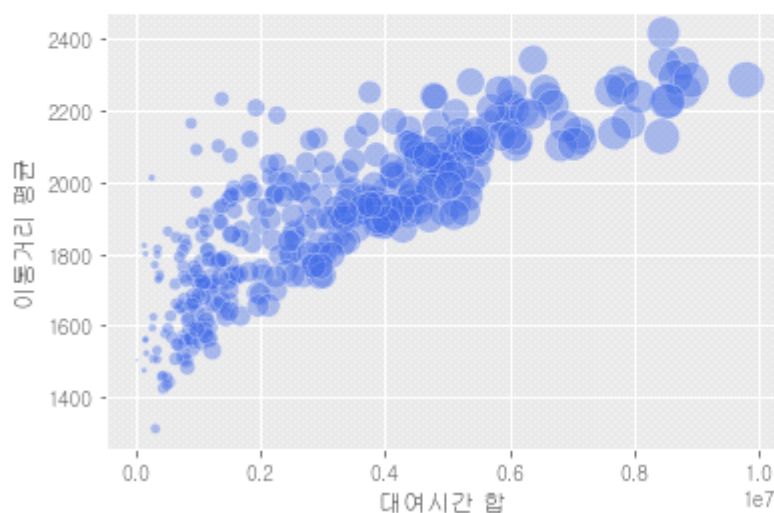
```



```

1  ### 버블 차트: 산점도 + 점의 크기
2  ax = sns.scatterplot(x = '대여시간_합', y = '이동거리_평균', color="royalblue",
3                      data = df_group_date,
4                      s = df_group_date['대여수']/10, alpha=0.4)
5  ax.set_xlabel('대여시간 합')
6  ax.set_ylabel('이동거리 평균')
7  plt.show()

```

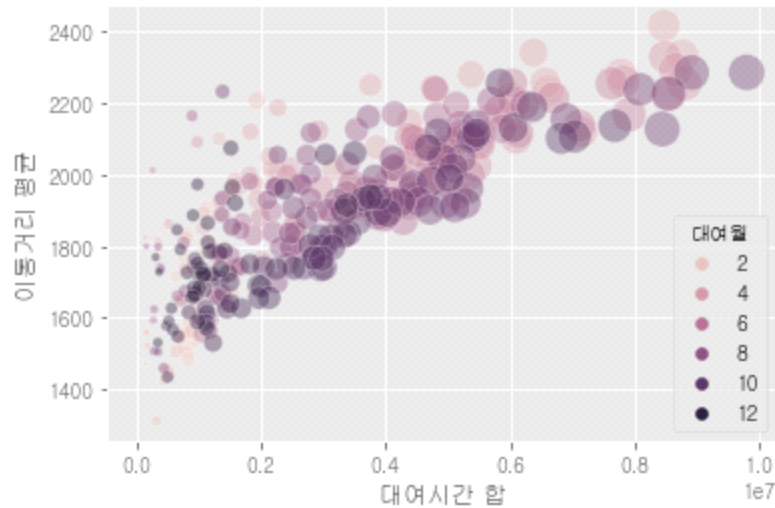


- 산점도 + 점의 크기 + 점의 색상

```

1  ### 버블 차트: 산점도 + 점의 크기 + 점의 색상
2  df_group_date['대여월'] = pd.to_datetime(df_group_date.index).month
3  ax = sns.scatterplot(x = '대여시간_합', y = '이동거리_평균',
4                      data = df_group_date,
5                      s = df_group_date['대여수']/10, alpha=0.4,
6                      hue = '대여월')
7  ax.set_xlabel('대여시간 합')
8  ax.set_ylabel('이동거리 평균')
9  plt.show()

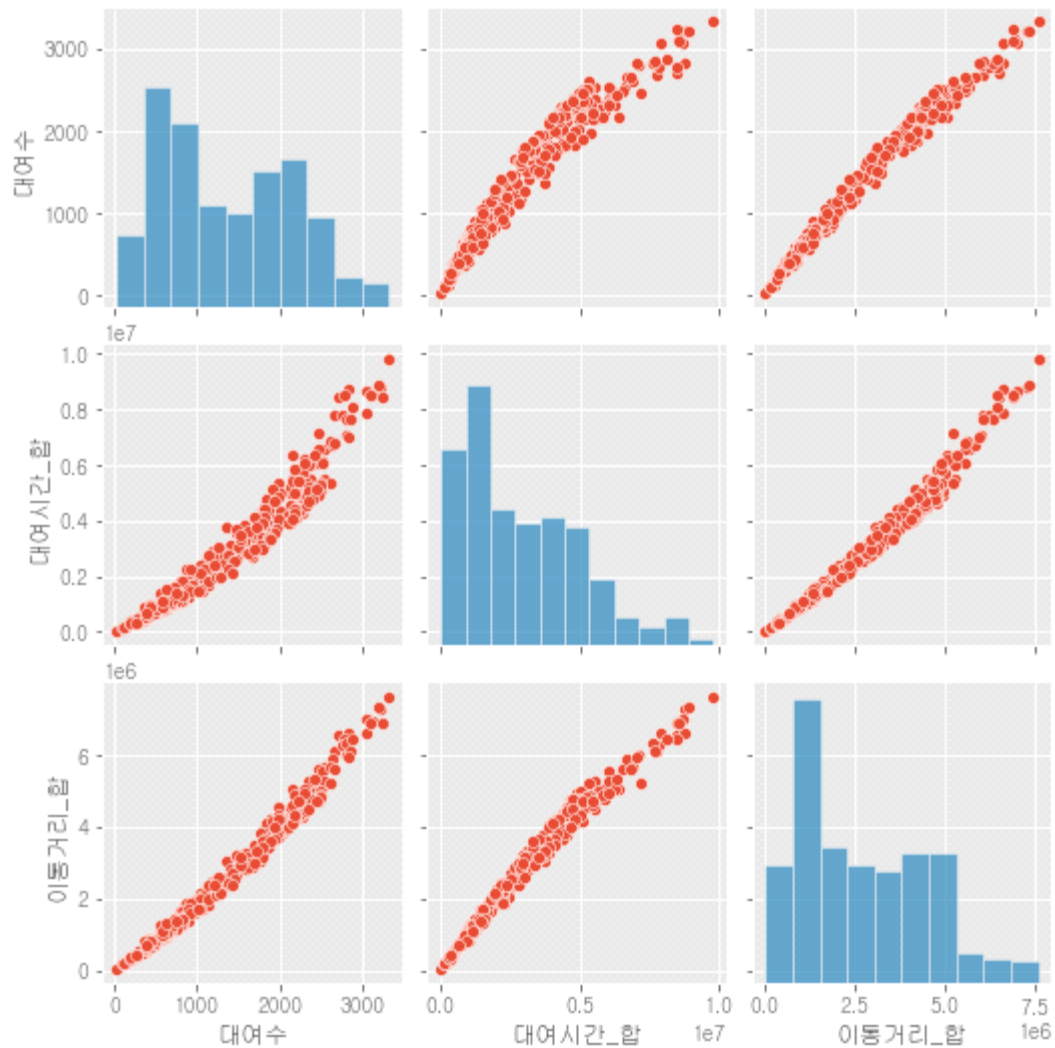
```



- 산점도 행렬(Scatter plot matrix)

- 3개 이상의 연속형 변수에 대한 모든 산점도를 행렬로 표현한 그림

```
1 ### 산점도 행렬  
2 ax = sns.pairplot(df_group_date.loc[:, '대여수':'이동거리_합'])
```

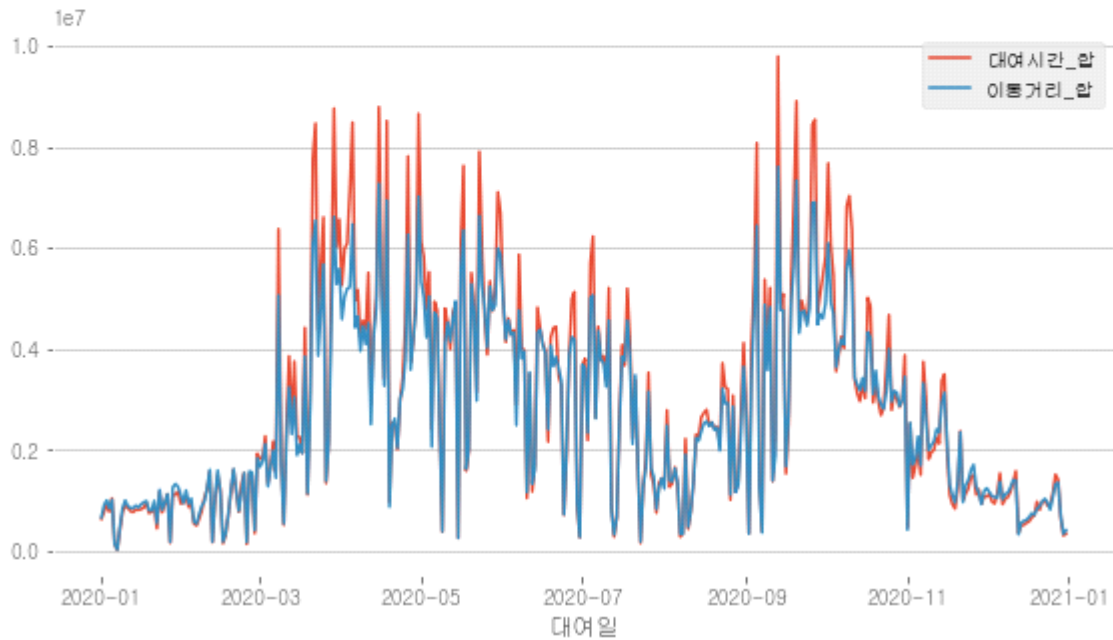


- 선 그림(line plot)
 - 연속형 변수 추가

```

1  ### 선 그림(line plot) - 연속형 변수 추가
2  ax = df_group_date.plot.line(y=['대여시간_합', '이동거리_합'],
3                                figsize = (9.5, 5))
4  ax.set_facecolor('white') # 배경색
5  ax.grid(True, axis='y', color="silver") # 눈금선(y축)
6  plt.show()

```



5 그래프 요약 - 대화형

□ plotly 패키지

▶ plotly 설치

```
1 !pip install plotly
```

▶ Package

○ Python API reference for plotly

- <https://plotly.com/python-api-reference/index.html>
- plotly.express
 - high-level interface for data visualization
- plotly.graph_objects
 - low-level interface to figures, traces and layout

```
1 ### Package
2 import plotly.express as px
```

▶ 자료

○ 타슈 스테이션 현황

- 대전광역시_공영자전거(타슈) 위치(위경도) 현황_20200801.csv

```
1 ### 타슈 스테이션 현황: 대전광역시_공영자전거(타슈) 위치(위경도) 현황
   _20200801.csv
2 data_file = "./data/타슈/대전광역시_공영자전거(타슈) 위치(위경도) 현황
   _20200801.csv"
3 tashu_station = pd.read_csv(data_file, encoding = "cp949")
```

□ 단일 변수의 탐색

○ 범주형/이산형 자료

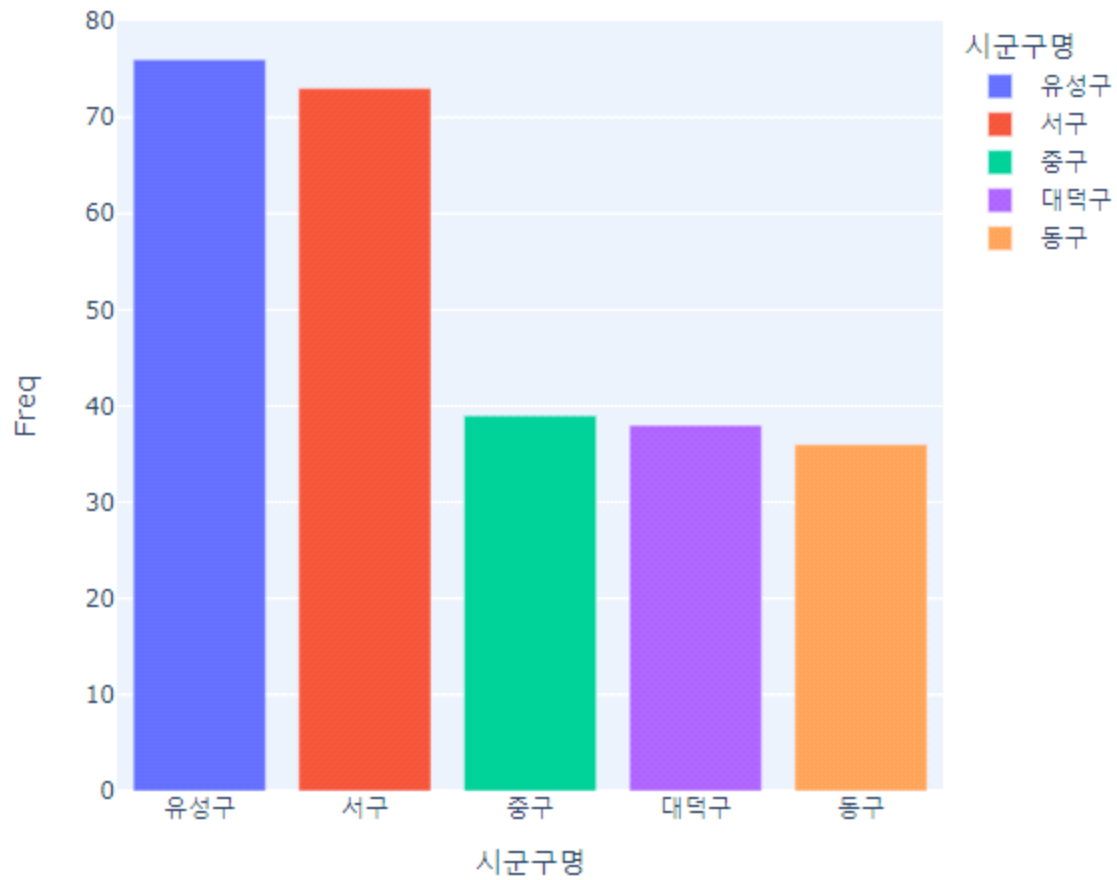
- 빈도표
 - 값이 발생한 빈도나 횟수를 포함하고 있으며, 이러한 방식으로 표는 표본 값의 분포를 요약

```
1 ### 빈도표
2 tb = pd.crosstab(tashu_station['시군구명'], 'Freq', colnames=[''])
3 tb_sort = tb.sort_values('Freq', ascending=False)
```

```

1  ### 막대 그래프 - 수직(기본)
2  fig = px.bar(tb_sort, x = tb_sort.index, y = 'Freq', color = tb_sort.index)
3  fig.show()

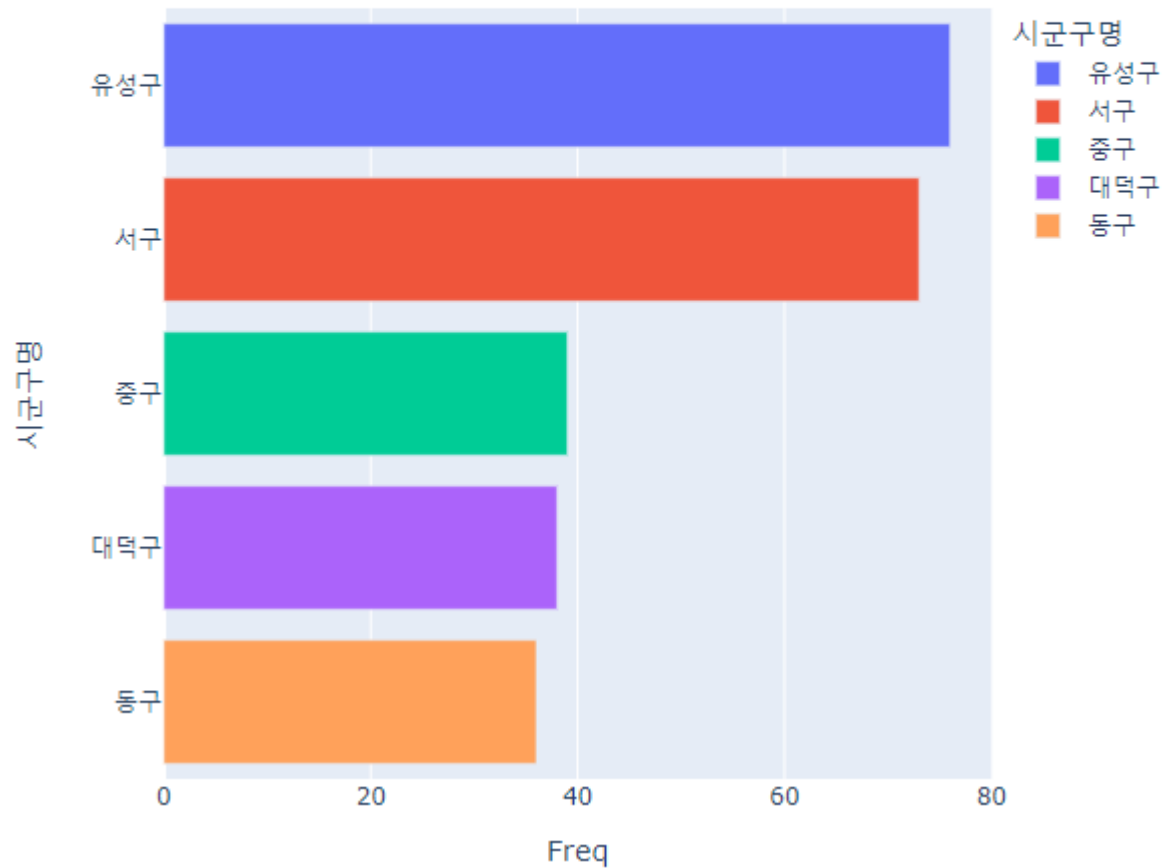
```



```

1  ### 막대그래프 - 수평
2  fig = px.bar(tb_sort, x = 'Freq',
3              y = tb_sort.index,
4              color = tb_sort.index)
5  fig.show()

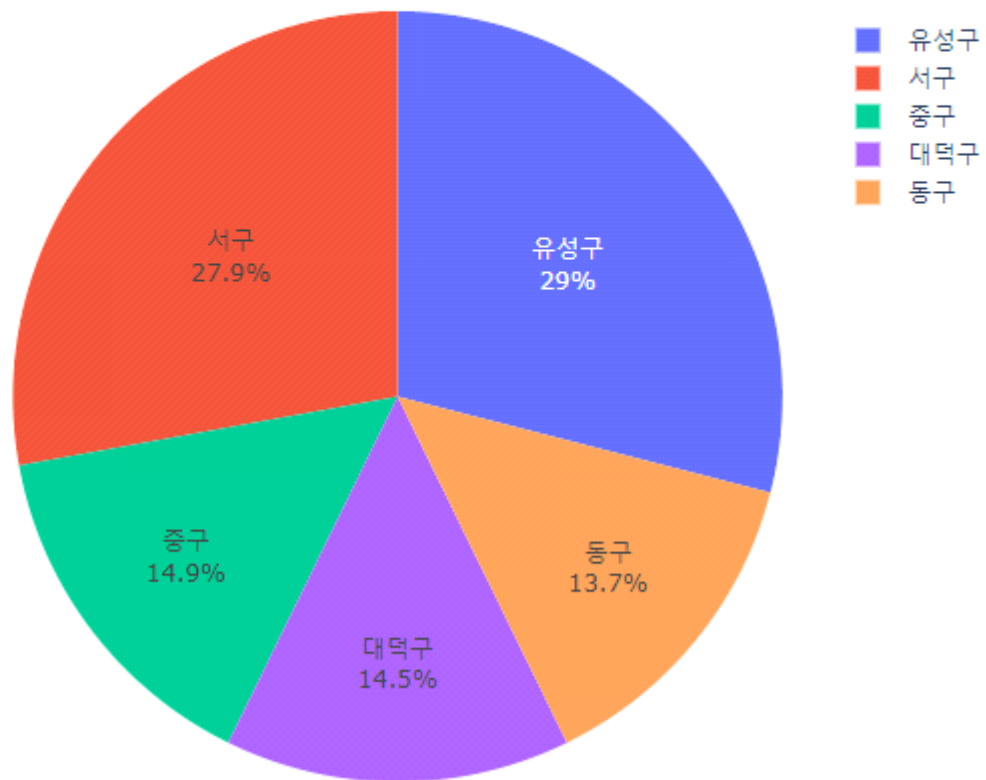
```



- 원 그래프(Pie chart)

- 전체에 대한 각 부분의 비율을 부채꼴 모양으로 나타낸 그래프

```
1 ### 원그래프
2 fig=px.pie(tb_sort, values='Freq', names=tb_sort.index)
3 fig.update_traces(textposition='inside', textinfo='percent+label')
4 fig.show()
```

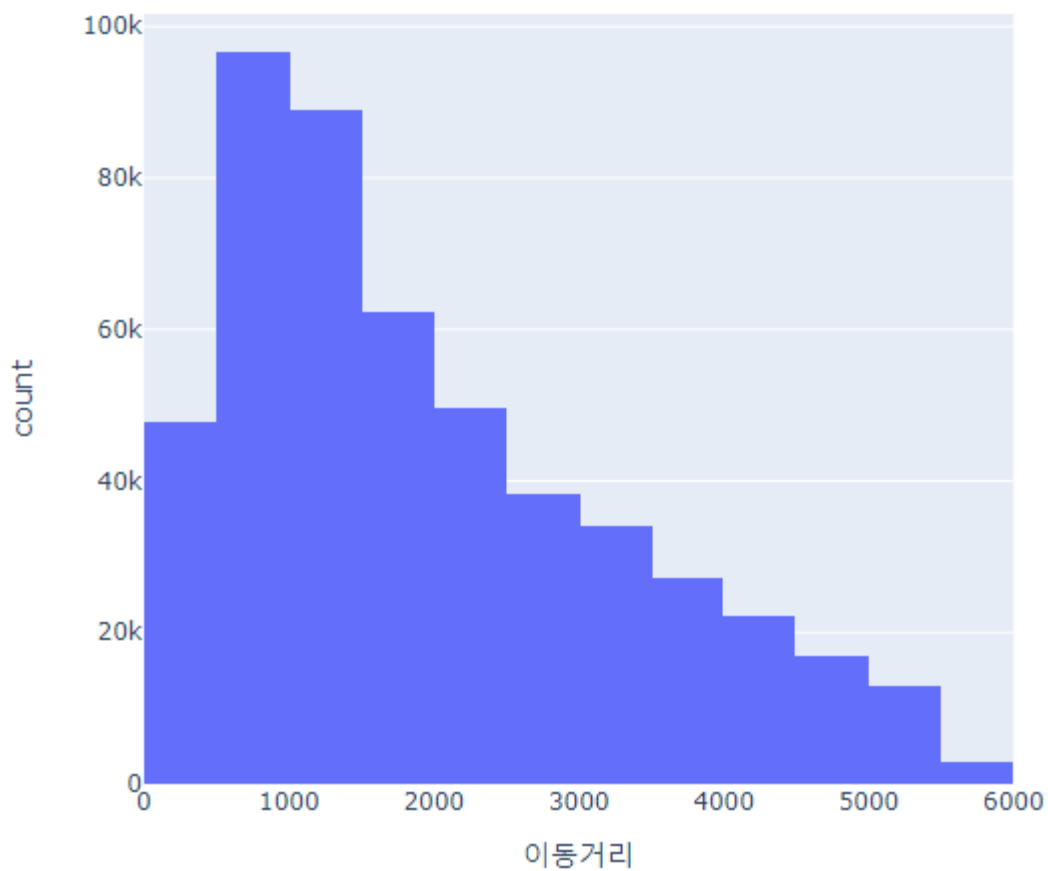


○ 연속형 자료

– 히스토그램(histogram)

- 표로 되어 있는 도수 분포를 정보 그림으로 나타낸 것
- 계급은 보통 변수의 구간이고, 서로 겹치지 않음

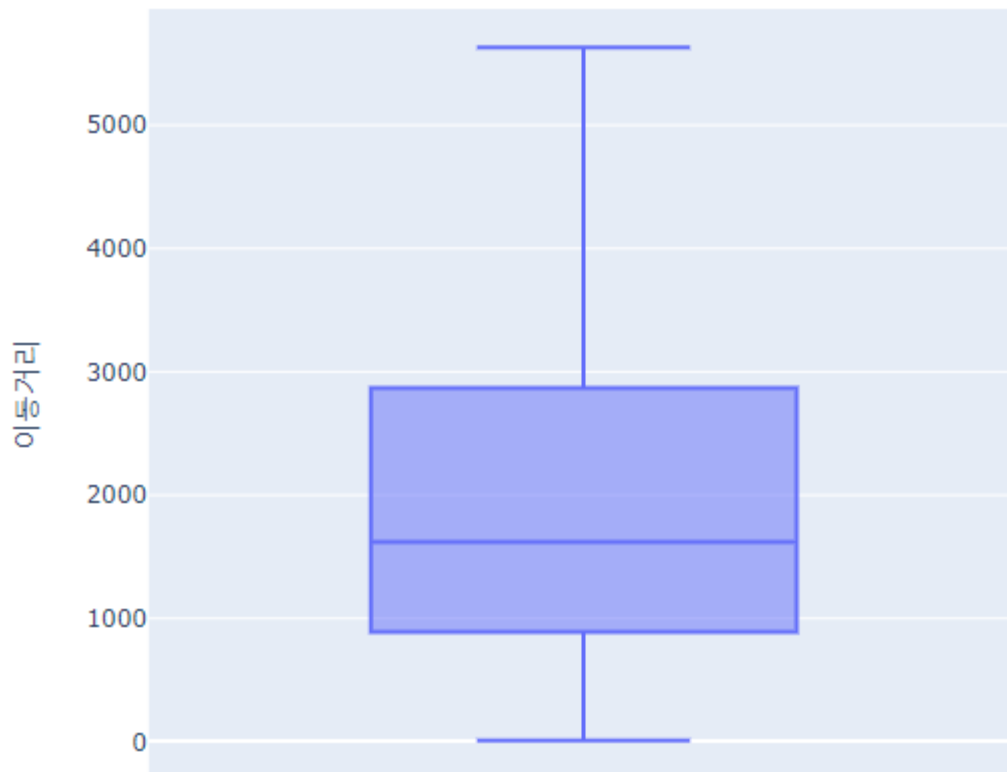
```
1 ### 히스토그램
2 fig = px.histogram(df_use, x = "이동거리", nbins = 14)
3 fig.show()
```



- 상자 그림(box plot)

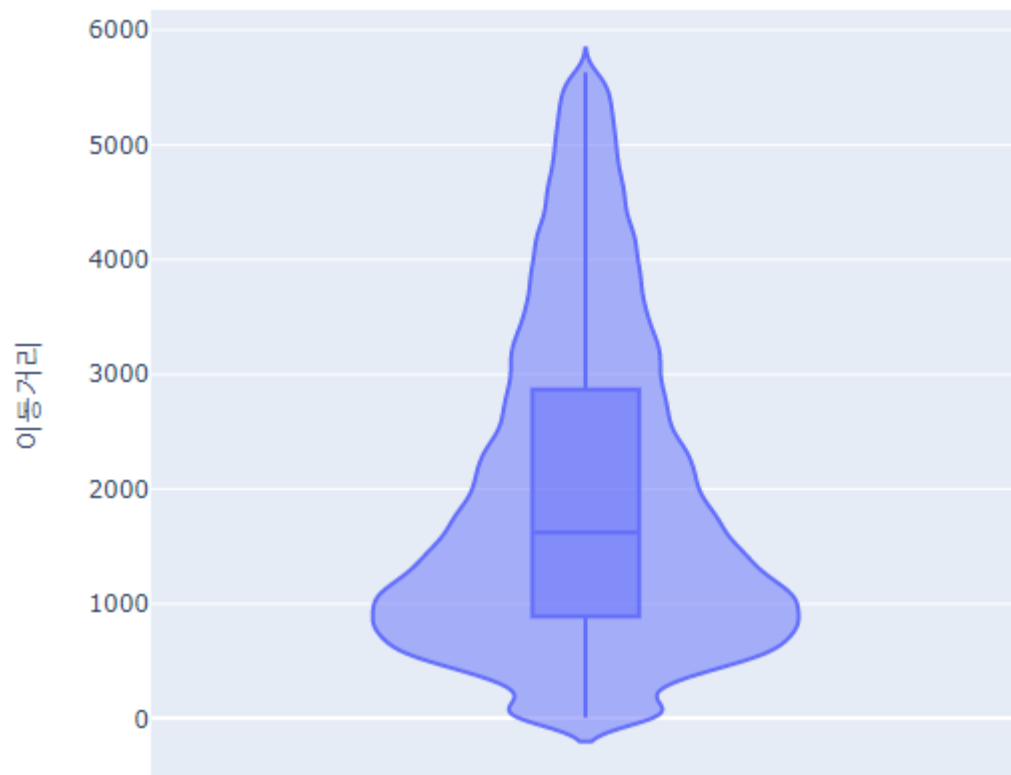
- 자료로부터 얻어낸 통계량 최솟값, 제 1사분위(Q1), 제 2사분위(Q2), 제 3사분위(Q3), 최댓값을 이용한 그림
- 제 1사분위 수 (Q1) : 중앙값 기준으로 하위 50% 중의 중앙값, 전체 데이터 중 하위 25%에 해당하는 값
- 중위수 : 데이터의 정 가운데 순위에 해당하는 값
- 제 3사분위 수 (Q3) : 중앙값 기준으로 상위 50% 중의 중앙값, 전체 데이터 중 상위 25%에 해당하는 값
- 사분위 범위(IQR) : 데이터의 중간 50% ($Q3 - Q1$)

```
1 ### 상자 그림
2 fig = px.box(df_use, y = "이동거리")
3 fig.show()
```



- 바이올린 그림(violin plot)
 - 상자 그림에 분포가 포함된 그래프

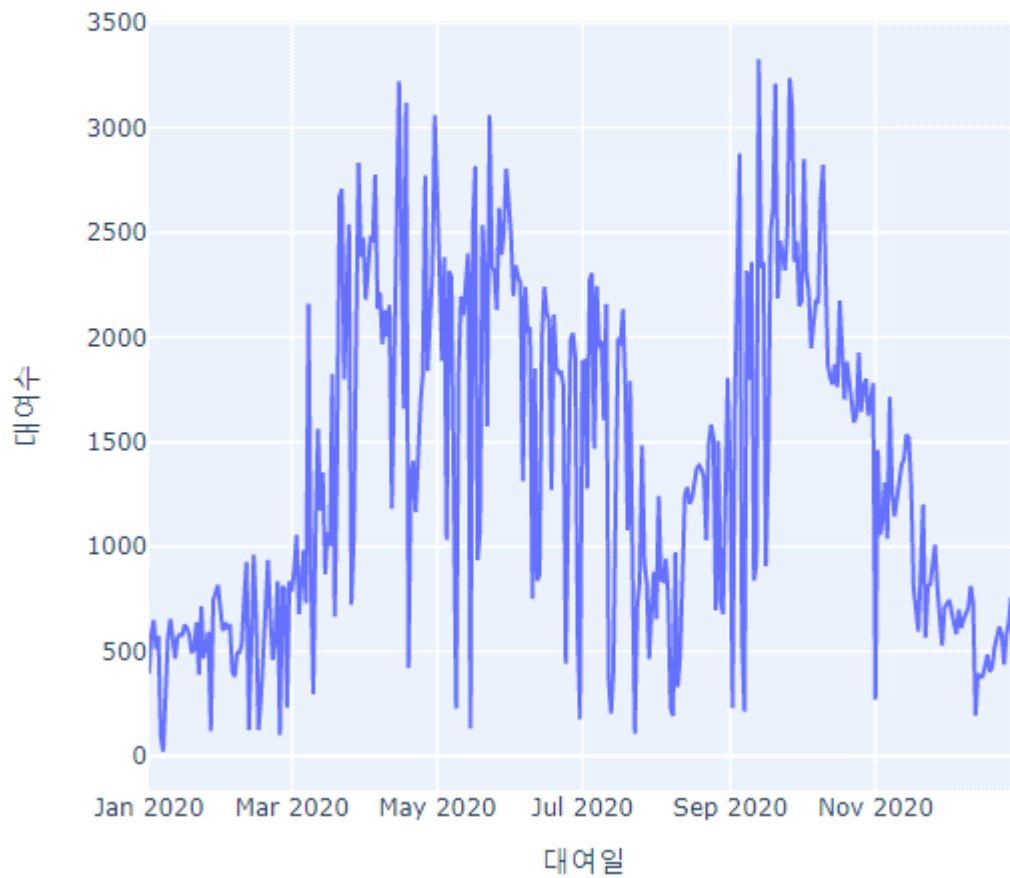
```
1 ### violin plot  
2 fig = px.violin(df_use, y = "이동거리", box = True) # points = 'all'  
3 fig.show()
```



- 선 그림(line plot)

- 수량을 점으로 표시하고 그 점들을 선으로 이어 그린 그래프
- 연속적인 값들의 변동을 파악하기 쉬움

```
1 ### 선 그림  
2 fig = px.line(df_group_date, y = "대여수")  
3 fig.show()
```



□ 두 변수의 탐색

- 범주형 & 범주형 자료
 - 막대그래프(barplot)
 - 빈도표를 이용하여 각 범주별 빈도를 확인

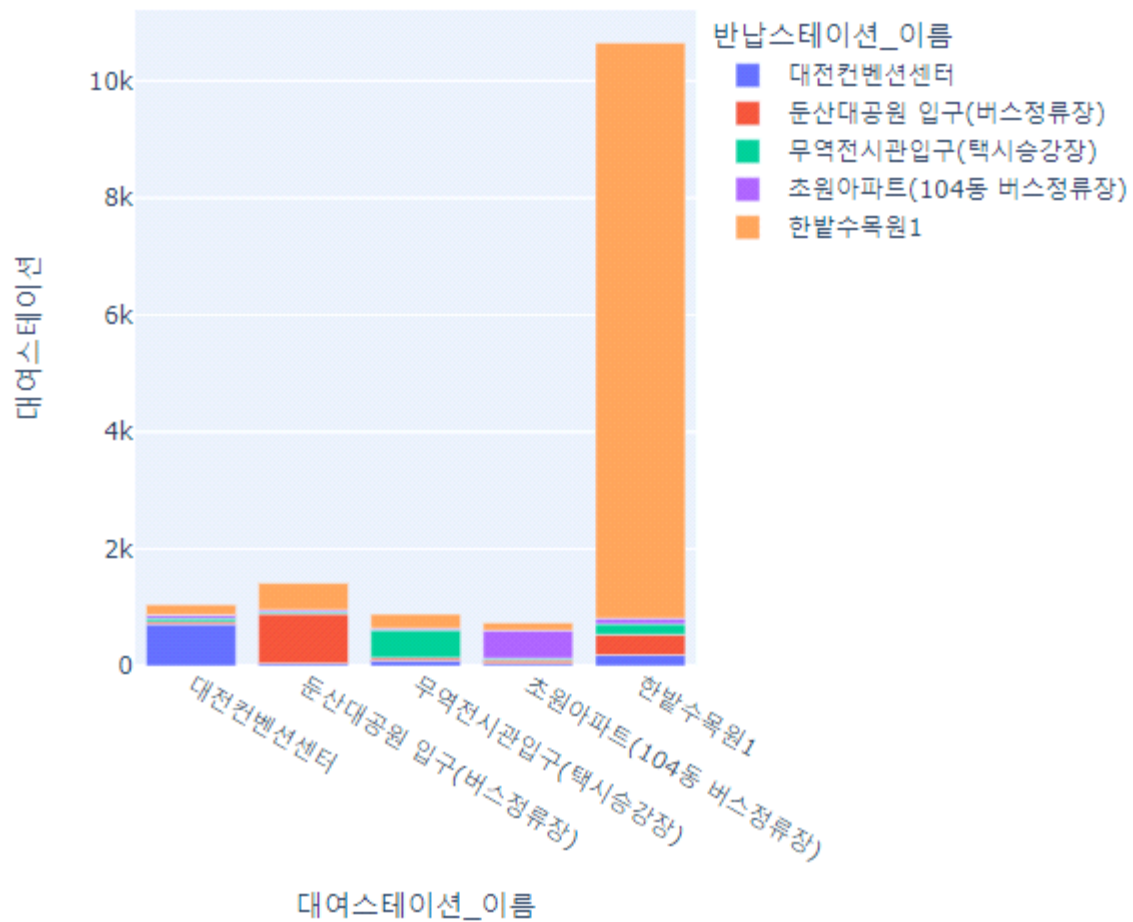
```
1 ### 빈도표
2 tb = df_sub.groupby(['대여스테이션_이름', '반납스테이션_이름'], as_index =
3 False)['대여스테이션'].count()
4 tb.head()
```

	대여스테이션_이름	반납스테이션_이름	대여스테이션
0	대전컨벤션센터	대전컨벤션센터	714
1	대전컨벤션센터	둔산대공원 입구(버스정류장)	41
2	대전컨벤션센터	무역전시관입구(택시승강장)	61
3	대전컨벤션센터	초원아파트(104동 버스정류장)	58
4	대전컨벤션센터	한밭수목원1	175

```

1 ### 막대그래프 - 누적
2 fig = px.bar(tb, x = "대여스테이션_이름", y = "대여스테이션", color = "반납스테이션_이름")
3 fig.show()

```

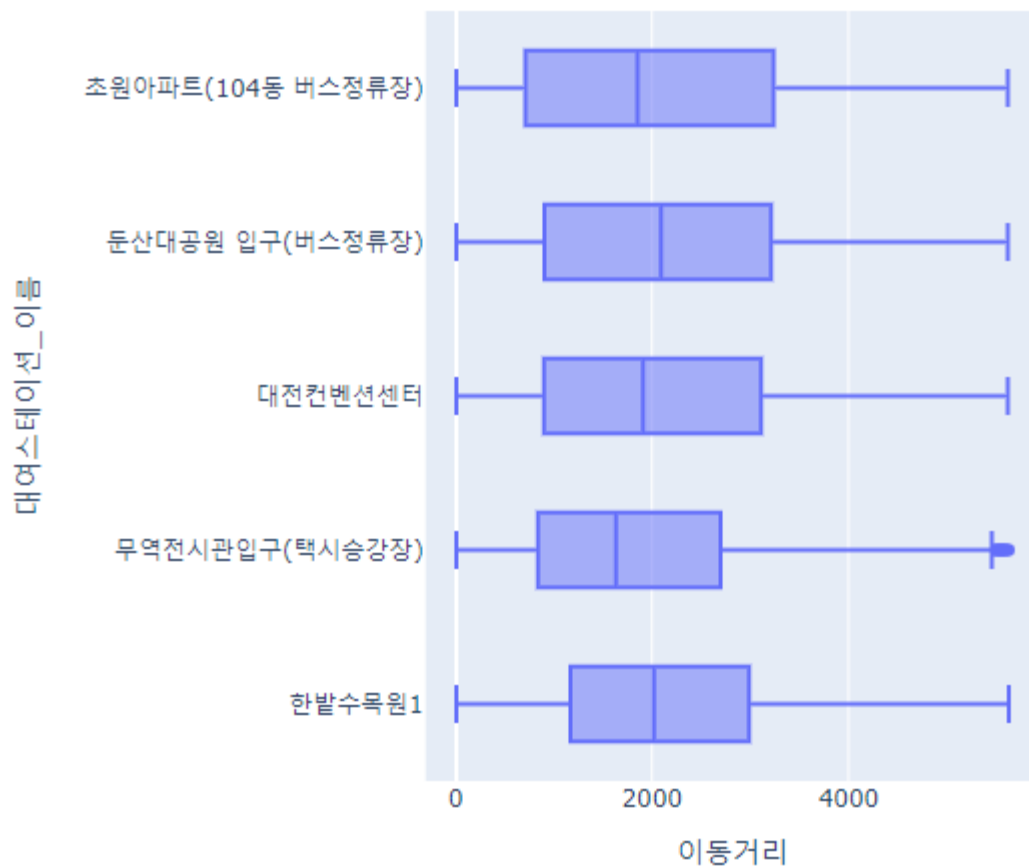


○ 범주형 & 연속형 자료

– 상자그림(boxplot)

- 범주형 변수의 값을 각 수준별로 나누어 연속형 변수에 대한 상자그림을 그림
- 집단별 분포를 비교

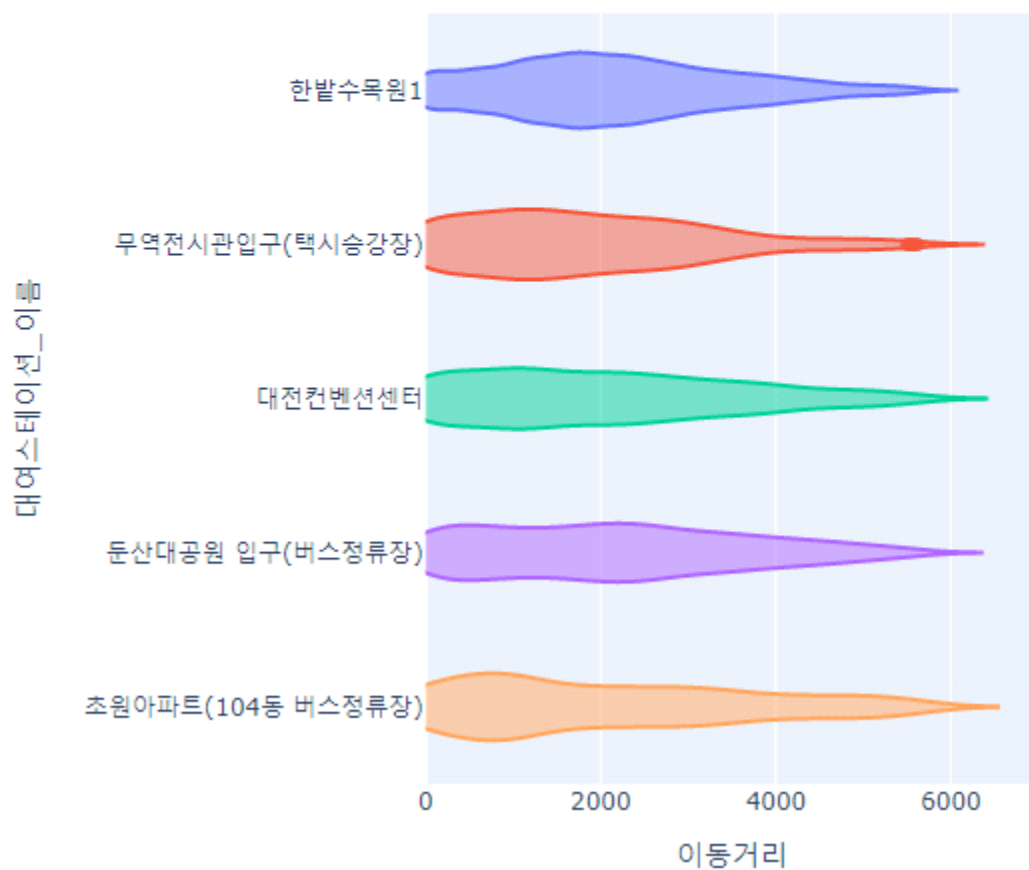
```
1 ### 상자그림 - 이동거리 by 대여스테이션_이름
2 fig = px.box(df_sub, x = "이동거리", y = "대여스테이션_이름")
3 fig.show()
```



- 바이올린 그림(violin plot)

- 범주형 변수의 값을 각 수준별로 나누어 연속형 변수에 대한 바이올린 그림을 그림
- 집단별 분포를 비교

```
1 ### violin plot - 이동거리 by 대여스테이션_이름
2 fig = px.violin(df_sub, x = "이동거리", y = "대여스테이션_이름",
3                 color = "대여스테이션_이름")
4 fig.update_layout(showlegend=False)
5 fig.update_xaxes(range=[0, 7000])
6 fig.show()
```



○ 연속형 & 연속형 자료

```

1  ### 대여일 별 대여수 & 대여시간(초) & 이동거리 기초통계량
2  df_group_date = df_use.groupby(['대여일']).apply(
3      lambda x: pd.Series({
4          '대여수': len(x),
5          '대여시간_합': sum(x.대여시간_초),
6          '이동거리_합': sum(x.이동거리),
7          '이동거리_평균': np.mean(x.이동거리),
8          '이동거리_최솟값': np.min(x.이동거리),
9          '이동거리_중앙값': np.median(x.이동거리),
10         '이동거리_최댓값': np.max(x.이동거리)
11     })
12 df_group_date.head(3)

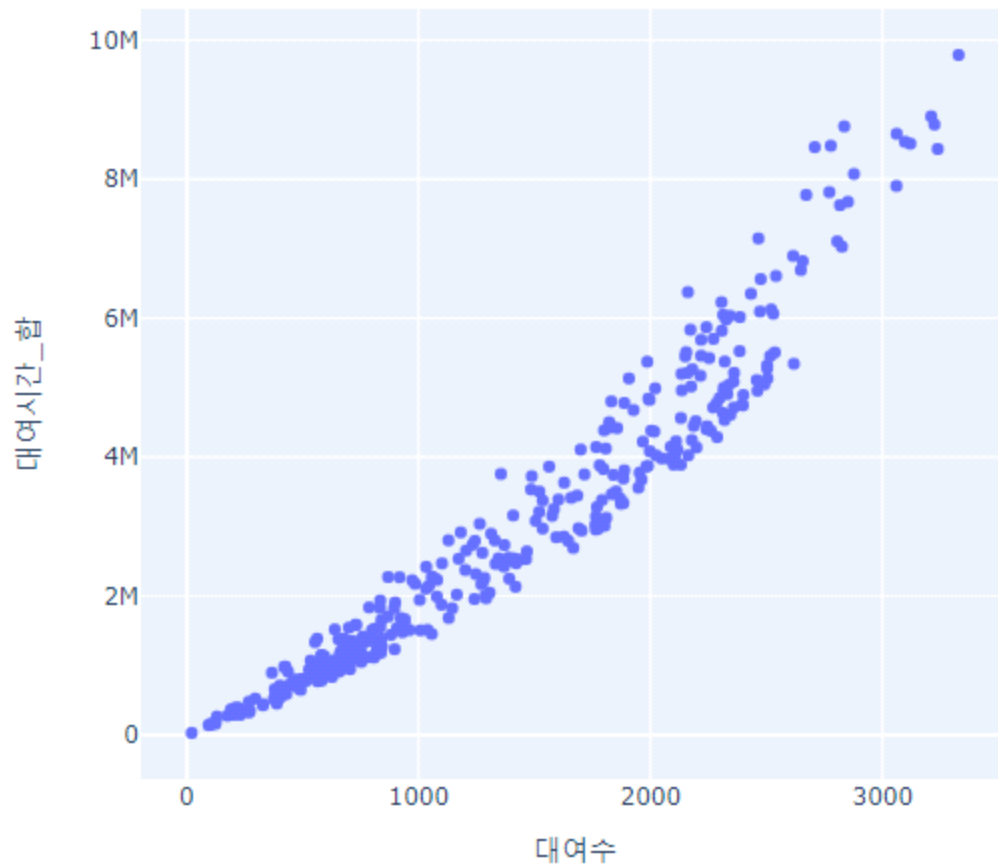
```

	대여 수	대여시간_ 합	이동거리_ 합	이동거리_평 균	이동거리_ 최솟값	이동거리_ 중앙값	이동거리_ 최댓값
대여일							
2020-01-01	394.0	641469.0	676070.0	1715.913706	10.0	1355.0	5370.0
2020-01-02	582.0	781913.0	913260.0	1569.175258	10.0	1210.0	5560.0
2020-01-03	650.0	917497.0	1011860.0	1556.707692	10.0	1245.0	5630.0

- 산점도(Scatter plot)

- 쌍(pair)으로 존재하는 연속형 자료를 좌표평면에 점으로 표시
- 두 개 변수 간의 관계를 나타내는 방법

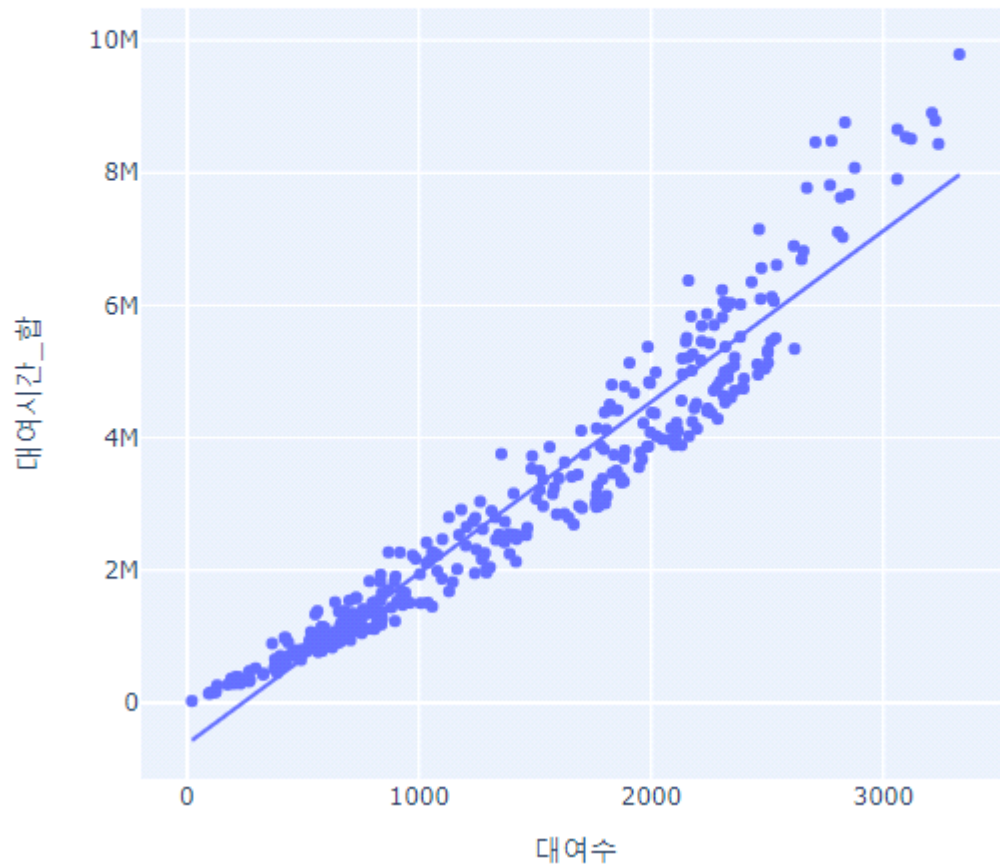
```
1 ### 산점도(Scatter plot)
2 fig = px.scatter(df_group_date, x = "대여수", y = "대여시간_합")
3 fig.show()
```




```

1  ### 산점도(Scatter plot) - 회귀선 추가
2  fig = px.scatter(df_group_date, x = "대여수", y = "대여시간_합",
3                  trendline = "ols")
4  fig.show()

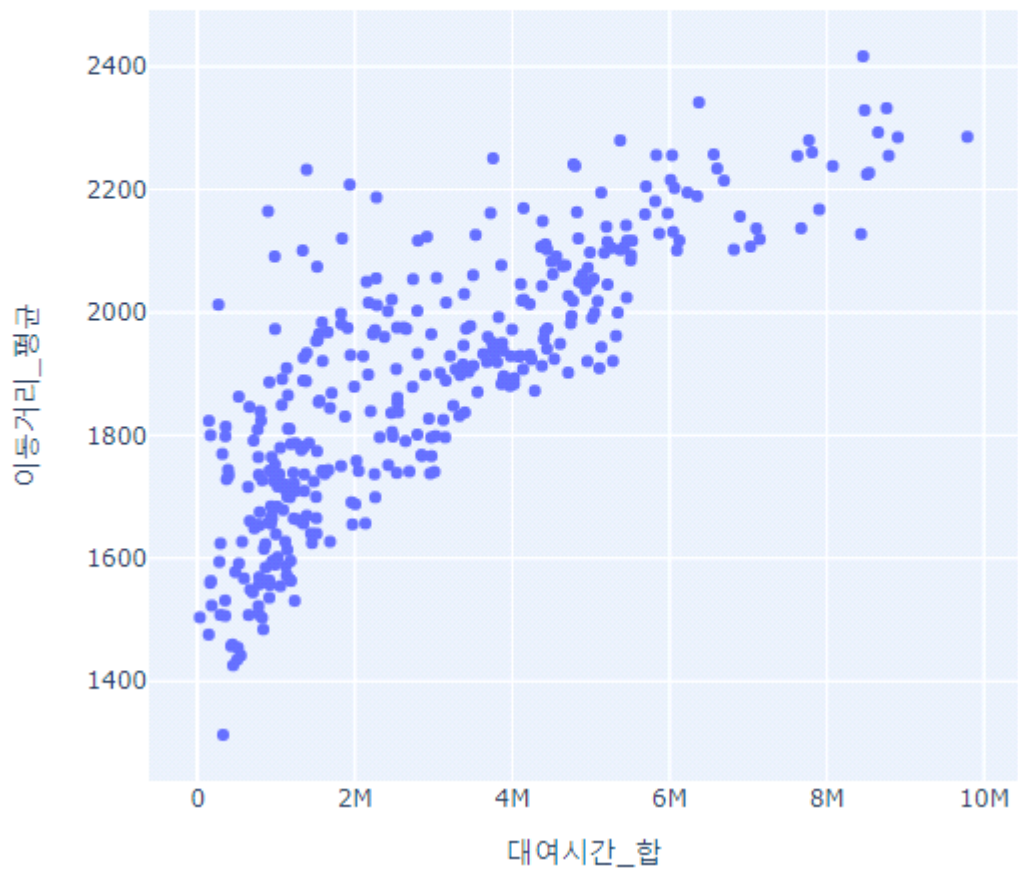
```



```

1 ### 산점도(Scatter plot)
2 fig = px.scatter(df_group_date, x = "대여시간_합", y = "이동거리_평균")
3 fig.show()

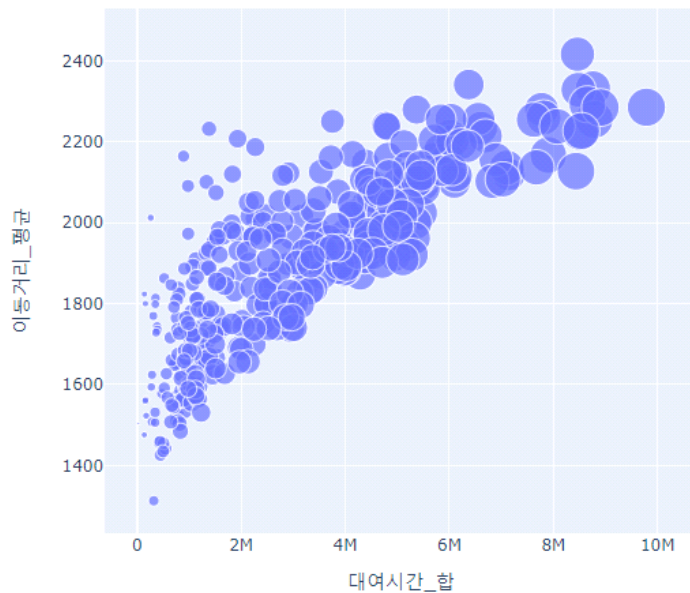
```



□ 다중 변수의 탐색

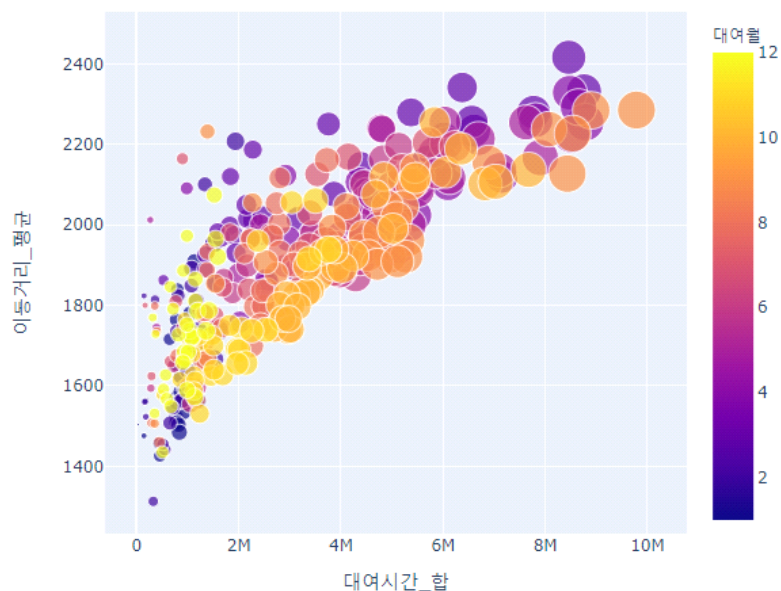
- 3개 이상의 변수를 표현한 그래프
 - 버블 차트(bubble chart)
 - 산점도 + 점의 크기

```
1 ### 버블 차트(bubble chart): 산점도 + 점의 크기
2 fig = px.scatter(df_group_date, x = "대여시간_합", y = "이동거리_평균",
3                 size = "대여수")
4 fig.show()
```



- 산점도 + 점의 크기 + 점의 색상

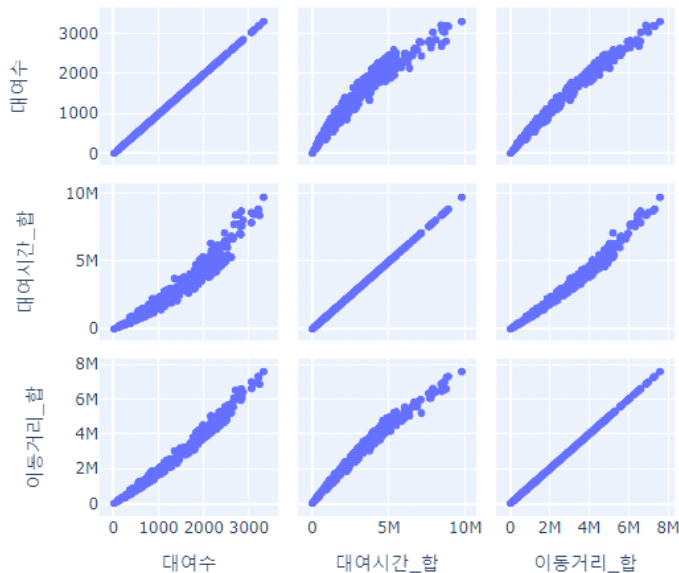
```
1 ### 버블 차트(bubble chart): 산점도 + 점의 크기 + 점의 색상
2 fig = px.scatter(df_group_date, x = "대여시간_합", y = "이동거리_평균",
3                 size = "대여수", color = "대여월")
4 fig.show()
```



– 산점도 행렬(Scatter plot matrix)

- 3개 이상의 연속형 변수에 대한 모든 산점도를 행렬로 표현한 그림

```
1 ### 산점도 행렬(Scatter plot matrix)
2 fig = px.scatter_matrix(df_group_date.loc[:, '대여수':'이동거리_합'])
3 fig.show()
```



– 선 그림(line plot)

- 연속형 변수 추가

```
1 ### 선 그림(line plot) - 연속형 변수 추가
2 fig = go.Figure()
3 fig.add_trace(go.Scatter(y = df_group_date["대여시간_합"],
4                           mode = 'lines', name = '대여시간_합'))
5 fig.add_trace(go.Scatter(y = df_group_date["이동거리_합"],
6                           mode = 'lines', name = '이동거리_합'))
7 fig.show()
```

