2024년 이젠컴퓨터학원

정 프로젝트

고령화에 따른 노인 운전자 사고

#1

주제선정

BrainStroming
주제 구체화 및 선정
프로젝트 일정

#2

프로젝트 일정 및 소개

프로젝트 일정
팀원소개
프로젝트시연

#3

데이터 가공

데이터 가공과정

#4

데이터 시각화

그래프 및 코드설명

Presenter – Jung Da Hyun

정 프로젝트

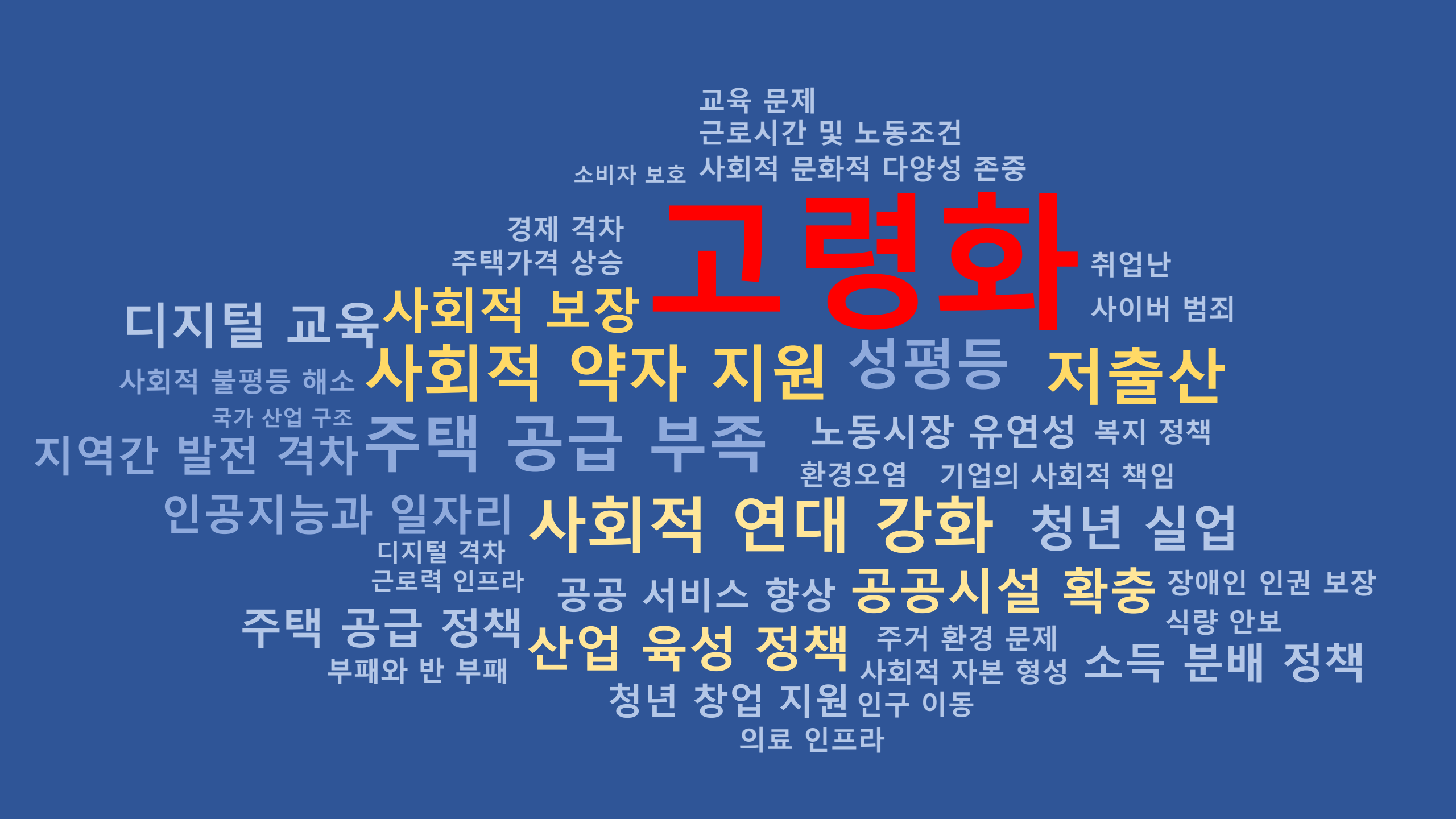
Team Member

Part :
맵, Sunburst ,
바 그래프, 파이그래프,
컨트롤러, 데이터 수집,
발표

Name : 정다현
Grade : 팀장

Part :
3D그래프, 선그래프,
맵, 컨트롤러,
프로젝트 디자인,
데이터 수집, 총괄

Name : 김성식
Grade : 팀원


교육 문제
근로시간 및 노동조건
소비자 보호
사회적 문화적 다양성 존중
경제 격차
고령화
주택가격 상승
취업난
디지털 교육
사회적 보장
사이버 범죄
사회적 약자 지원
성평등
저출산
사회적 불평등 해소
국가 산업 구조
지역간 발전 격차
주택 공급 부족
노동시장 유연성
복지 정책
환경오염
기업의 사회적 책임
인공지능과 일자리
사회적 연대 강화
청년 실업
디지털 격차
근로력 인프라
공공 서비스 향상
공공시설 확충
장애인 인권 보장
주택 공급 정책
식량 안보
산업 육성 정책
주거 환경 문제
부패와 반 부패
사회적 자본 형성
소득 분배 정책
청년 창업 지원
인구 이동
의료 인프라

저출산 고령화, 미래 경쟁력까지 위협… "2040년 R&D 인력 12만 감소'

파이낸셜뉴스 2024년 02월 14일 김동호 기자

저출산과 고령화의 영향으로 한국 경제 잠재성장률이 2040년 0.7%까지 떨어진다는 연구 결과가 나왔다. 특히 혁신 아이디어로 경제 성장을 견인할 연구인력이 2040년까지 12만명이 줄어들 것으로 전망돼 비상등이 켜졌다.

고령화 사회와 더불어 저출산 문제로 인구감소에서 이어진 연구인력감소

대한상공회의소 자본시장연구원(SGI)은 14일 '저출산 고령화의 성장률 완화를 위한 생산성 향상 방안' 보고서를 발표했다. 보고서에 따르면 2000년대 2%대 수준을 유지하던 국내 경제 잠재성장률은 코로나19를 거치며 지난해 1%대로 떨어진 것으로 추정되며 노동 성장 기여가 크게 줄며 2040년에는 0.7%까지 하락할 것으로 예측

경제 잠재성장률 0.7%대로 하락에 주목필요

보고서는 2018년 노벨상 수상자 폴 로머의 연구를 인용하며 "저출산 고령화 극복은 우리나라 혁신 역량에 달려있다"고 강조했다. 폴 로머는 "국가의 장기적 경제성장은 아이디어 축적에 달려있다"며 "혁신적 아이디어를 많이 쌓기 위해서는 연구인력 증가율과 연구자당 생산성이 중요하다"고 주장했다.

연구 인구증가 → 연구자당생상성 향상 필요

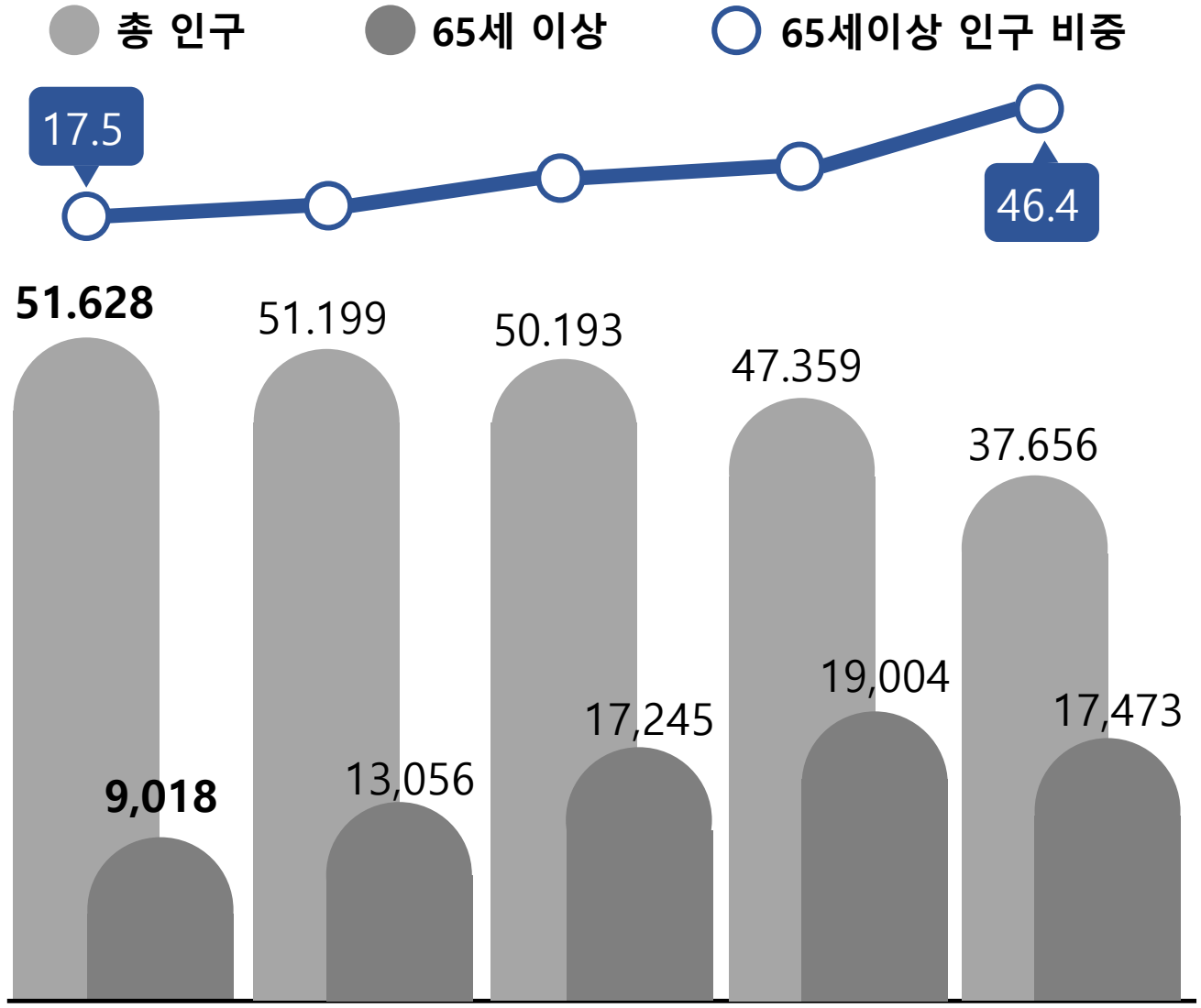
하지만 국내 연구인력 수를 보면 2020년 55만8000명에서 저출산 고령화 영향으로 2030년 51만2000명, 2040년 43만7000명으로 줄어드는 것으로 나타났다. 20년새 12만1000명이나 줄어드는 셈이다.

고령화 문제의 심각성 이제는 주목해야할 필요성

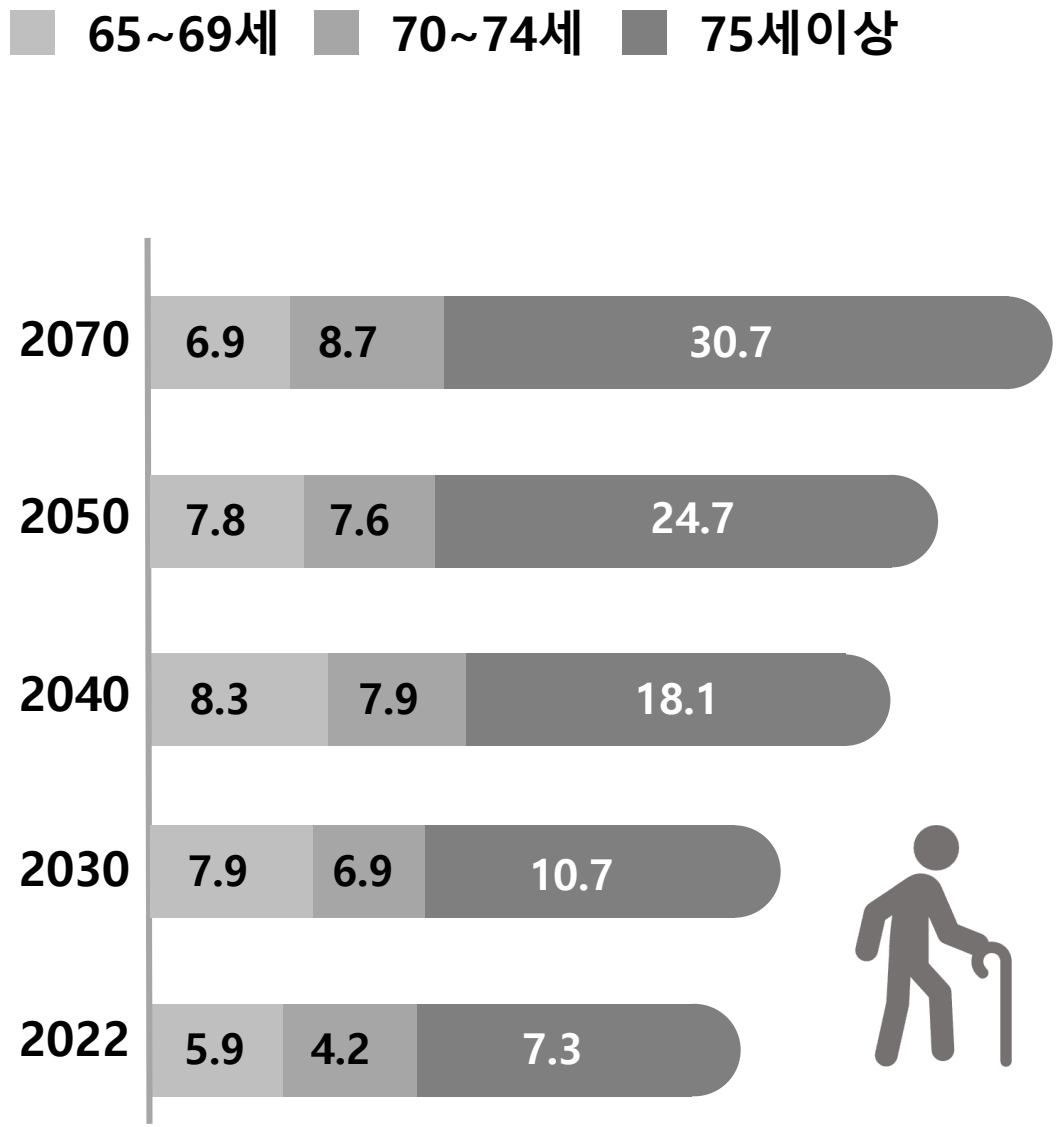
필요성을 강조했다. 대한상의 미치는 생산성 제고 효과가 약 43% 높다"며 "클린테크 등 신기술에 대해 인센티브 시스템을 마련할 정부의 금융 지원 강화 등이 필요하다"고 주장

출처 : https://www.fnnews.com/news/202402151337535647

고령 인구 및 비중

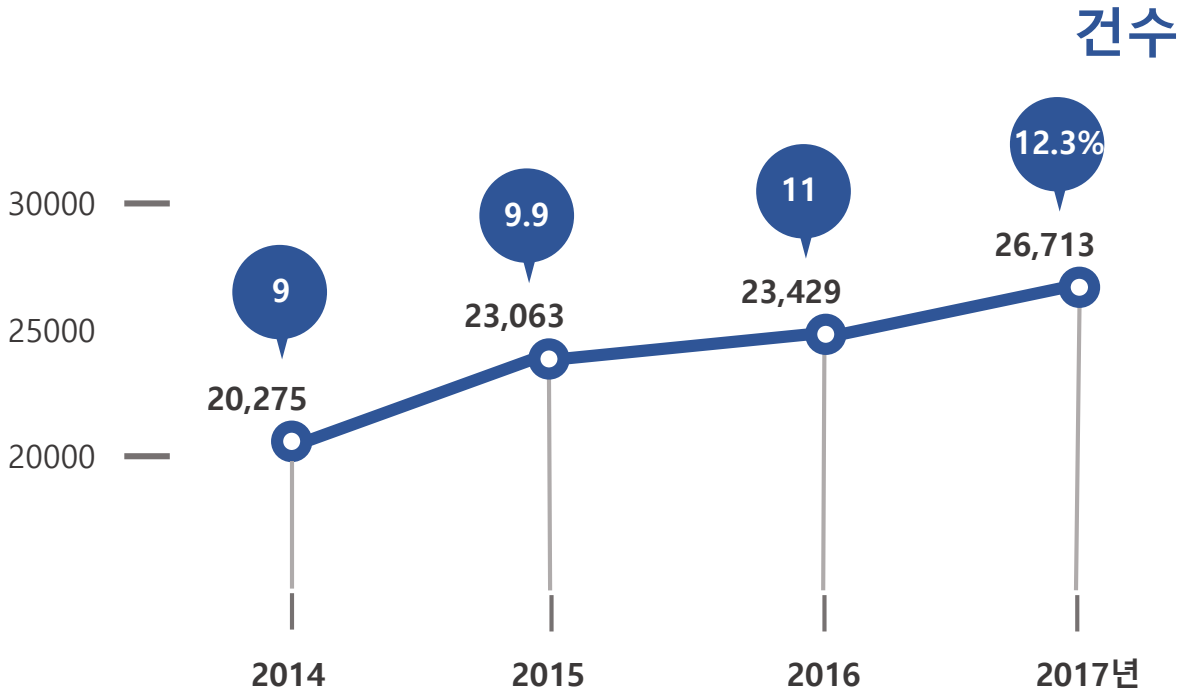


연령층별 고령인구 비중



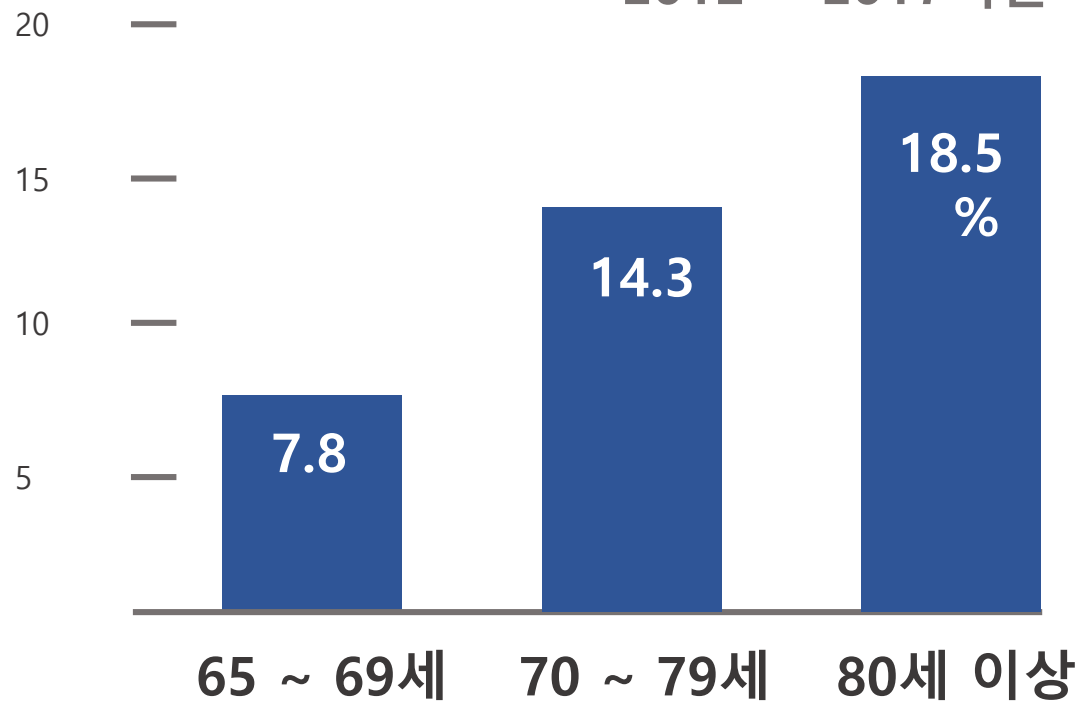
65세 이상 고령 운전자 교통사고 발생현황

전체 교통사고 중 비율

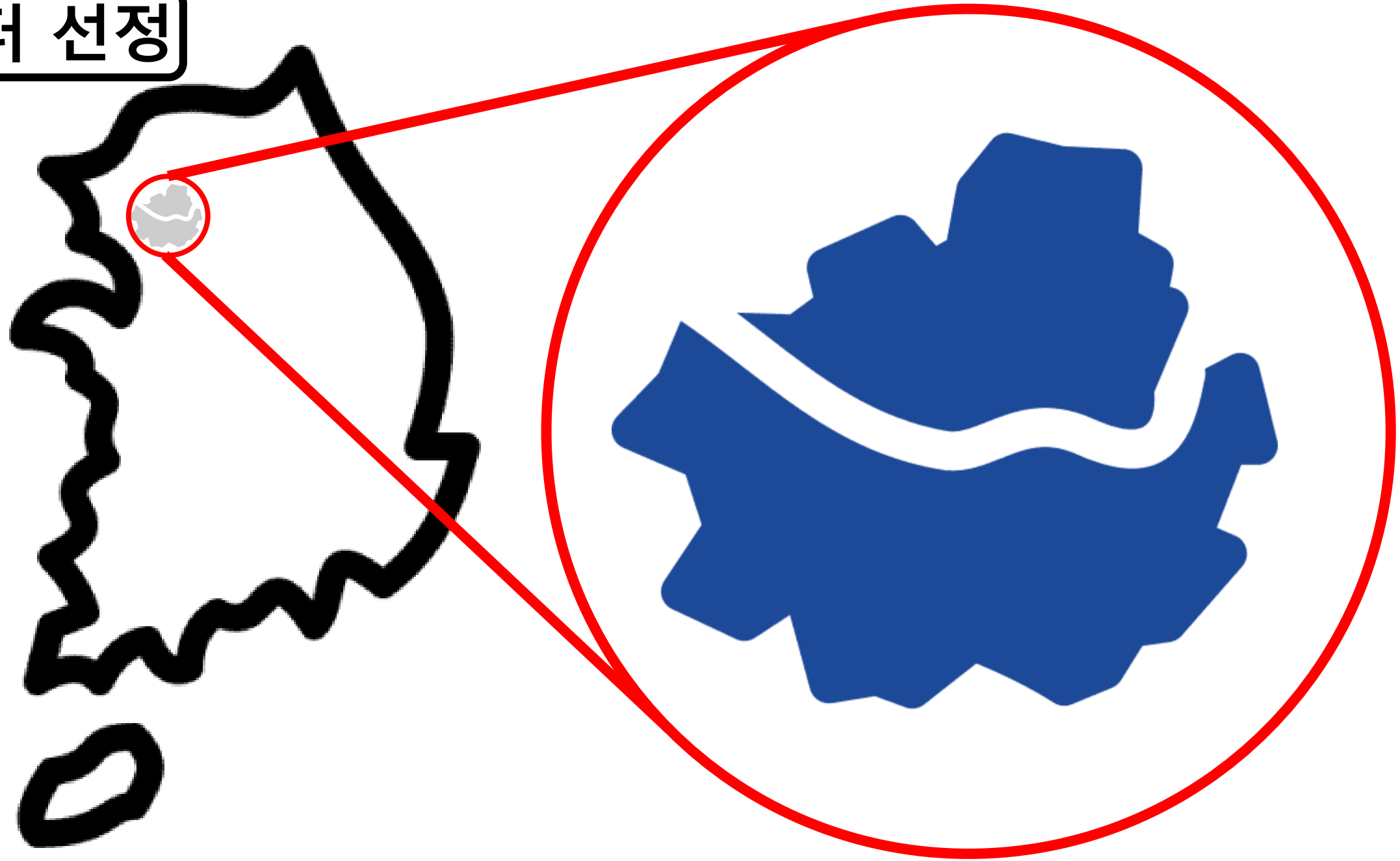


고령운전자 연령대별 교통사고 증가율

2012 ~ 2017기준

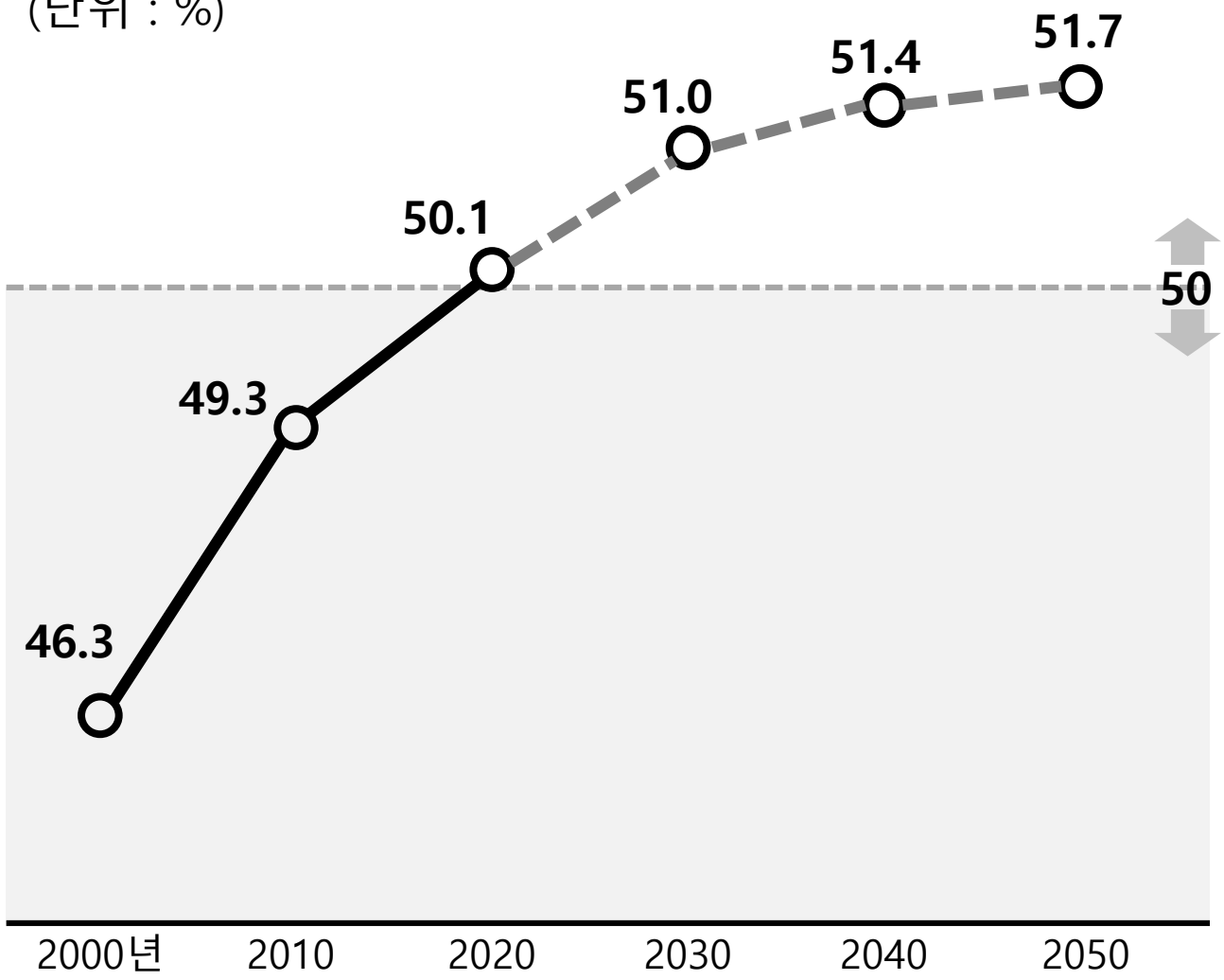


데이터 선정



수도권 인구 비중

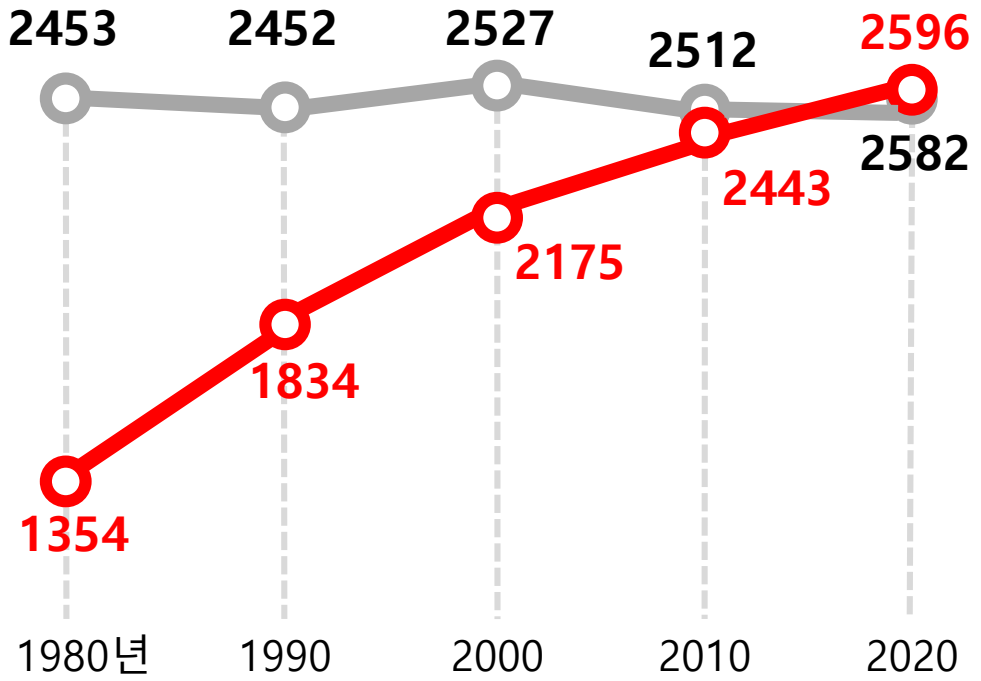
(단위 : %)



*2020년 이후는 전망치

자료 : 통계청

수도권,비수도권 인구 총 인구 추이(단위 : %)



* 비수도권 : 중부권, 호남권, 영남권

* 2020년 6월 29일 기준 추산치

자료 : 통계청

01 고령화
한국사회에서 중요하게 이슈화되고있는 인구감소와 관련한 주제 중 고령화를 선택

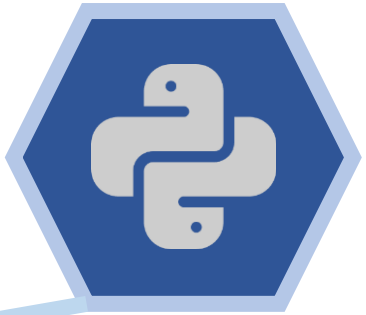
02 교통
프로젝트의 가장 큰 목표

03 노인교통사고
'고령화'와 '교통'이라는 키워드를 둘다 적용할 수 있는 주제 선정

08 수도권(서울)
데이터 범위를 설정중 대한민국에서 인구 비중이 가장 높은 서울로 지역선정

프로젝트

07 시각화
가공된 데이터로 그래프를 그려냅니다,



05 파이썬
파이썬 가상환경에서 프로그래밍 작업을 진행합니다

06 데이터
- 노인 공공데이터 가공
- 경계선 데이터 가공

프로젝트 일정 및 소개

2024.02.05
프로젝트 시작
데이터 수집
- 고속도로별 사고 건 데이터 수집
- 고속도로별 무인단속장비 데이터 수집
- 노인사고,노인운전자사고,노인보행자사고 데이터 수집

2024.02.06
~ 2024.02.08
그래프 그리기
막대그래프, 원 그래프 찍어보기
(서울시 구별, 동별)

2024.02.13
~ 2024.02.14
맵 그리기
-경계선 데이터를 가지고 서울시 구별 동별 경계선찍기
-구별 중심좌표 찾아 마커표시하고 그위에 원형 그래프 찍기

2024.02.15
~ 2024.02.17
-그래프 애니메이션 효과추가
-마무리

Geopandas
지도의 경계선을 찍어냄

Chart.JS
자바스크립트로 차트 그리기

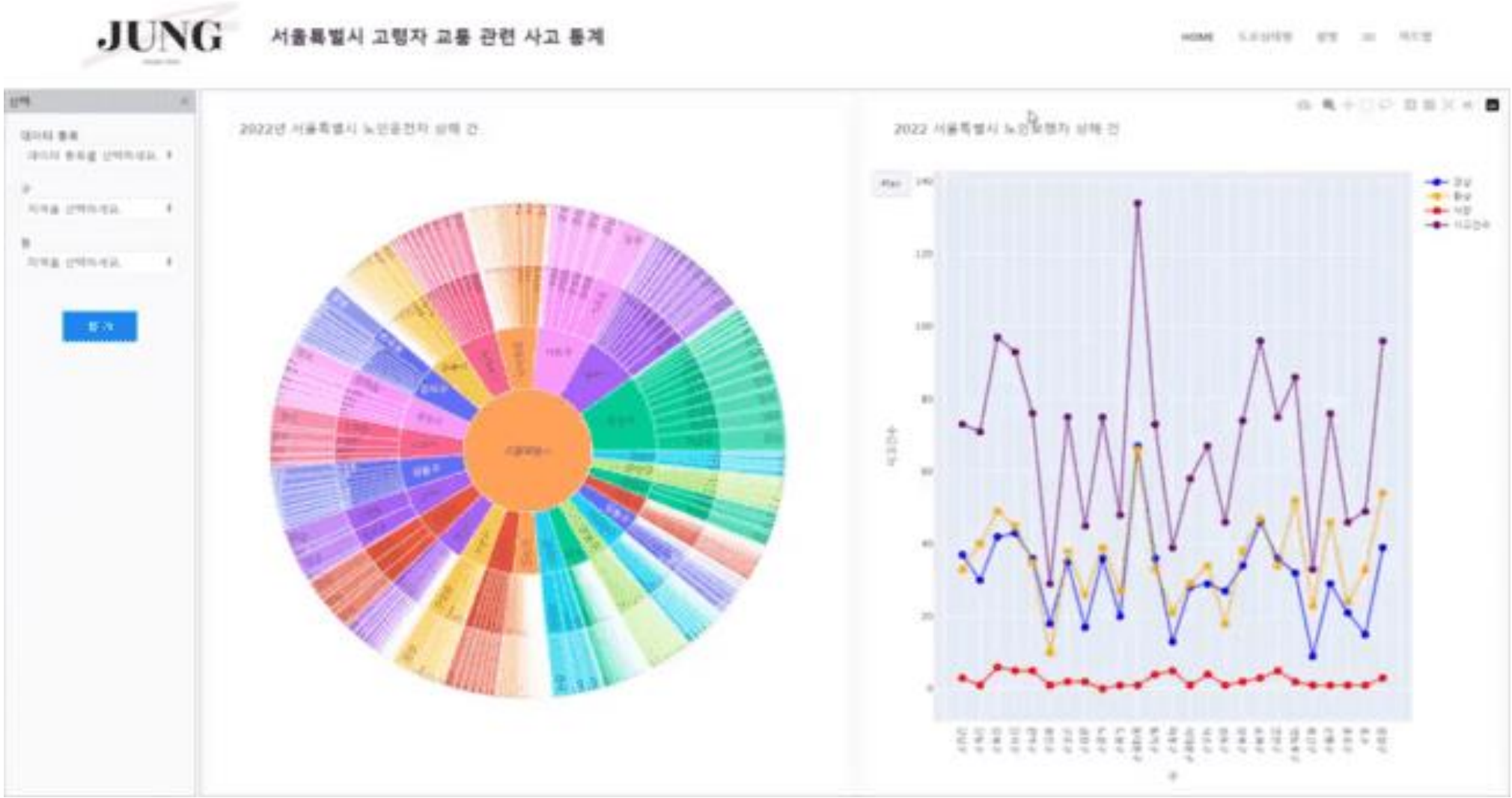
Pandas
데이터 가공

folium
지도 그리기

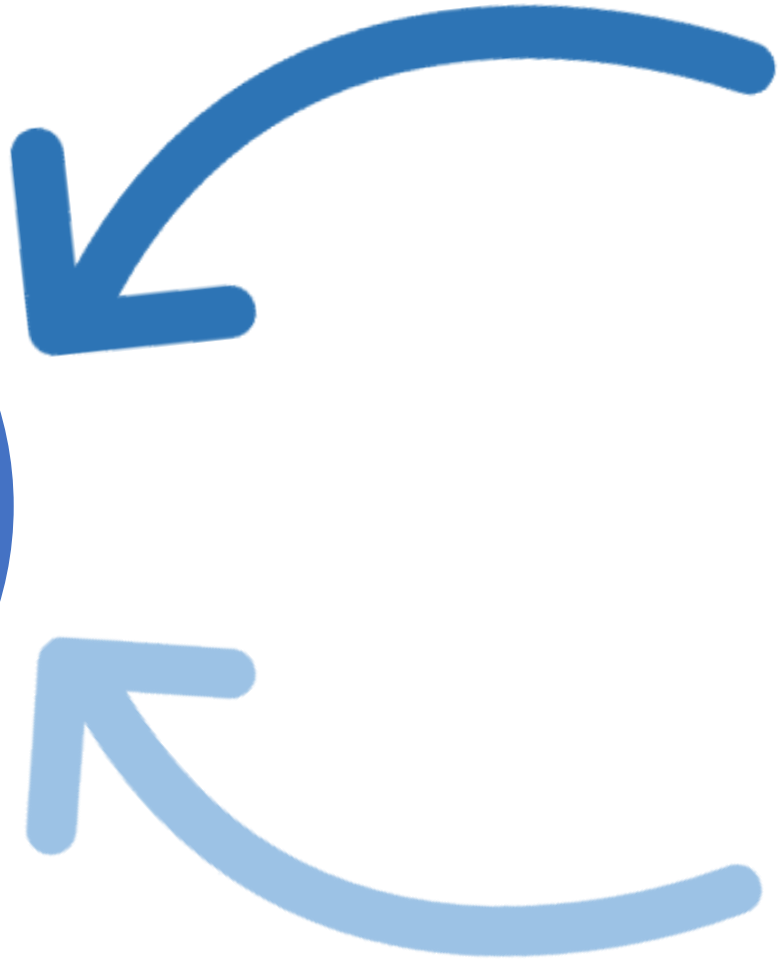
plotly
그래프 그리기

프로젝트 시연

프로젝트 시연영상



데이터 수집



KoROAD 도로교통공단

노인데이터

서울 열린데이터 광장

https://data.seoul.go.kr/dataList/OA-12074/F/1/datasetView.do

경계선 데이터

데이터 가공

원본 데이터 : 노인데이터.xlsx

- 6012행 16열

	A	B	C	D	E	F	G	H	I	J
1	사고번호	사고일시	요일	시군구	사고내용	사망자	중상자	경상자	부상신고자	사고유형
2	2022010100100002	2022년 1월 1일 00시	토요일	서울특별시 송파구 신천동	중상사고	0	2	0	0	차대사람 - 횡단중
3	2022010100100046	2022년 1월 1일 07시	토요일	서울특별시 용산구 한남동	경상사고	0	0	1	0	차대차 - 추돌
4	2022010100100062	2022년 1월 1일 09시	토요일	서울특별시 용산구 한남동	경상사고	0	0	1	0	차대차 - 추돌

가공된 데이터

-26행 9열

	A	B	C	D	E	F	G
1		구_숫자	경상	중상	사망	구_그룹명	사고건수
2	0	1	37	33	3	강남구	73
3	1	2	30	40	1	강동구	71
4	2	3	42	49	6	강북구	97
5	3	4	43	45	5	강서구	93

데이터 가공

```
<CODE>
df = pd.read_excel('노인데이터.xlsx', sheet_name='노인운전자사고')
 # 데이터 읽기

# '시군구' 열에서 '구'로 끝나는 단어만 추출하여 리스트에 저장
gu_list = []
for gu_str in df['시군구']:
    gu_words = gu_str.split()
    for word in gu_words:
        if word.endswith('구'):
            gu_list.append(word)

sorted_gu_list = sorted(set(gu_list))

for gu in sorted_gu_list:
    gu_df = df[df['시군구'].str.contains(gu)]
    gu_counts = gu_df['사고유형'].value_counts().reset_index()
    gu_counts.columns = ['사고유형', '건 수']
    print(gu_counts)
    break
이하 생략
```

	A	B	C
1		사고유형	건 수
2	0	차대차 - 기타	192
3	1	차대차 - 측면충	142
4	2	차대차 - 추돌	82
5	3	차대사람 - 기타	76

- 서울시 구 리스트를 gu_list에 저장한 후 gu_list를 조회하며 구별 사고 유형별 사건수를 계산해 gu_counts변수에 저장 하는 코드

- df['column'].value_counts() => column의 고유한 값이 각각 몇번씩 나오는지 세어주는 메서드입니다.

df.reset_index() => 데이터프레임의 인덱스(index)를 재설정하는 메서드입니다.

구별 코드

<CODE>

```
# '시군구' 열에서 '구'로 끝나는 단어만 추출하여 리스트에 저장
gu_list = []
for gu_str in df['시군구']:
    gu_words = gu_str.split()
    for word in gu_words:
        if word.endswith('구'):
            gu_list.append(word)

# 추출된 '구'로 끝나는 단어들을 가나다순으로 정렬
sorted_gu_list = sorted(set(gu_list))

# 각 구별로 경상, 중상, 사망 횟수 카운트
data = []
for gu in sorted_gu_list:
    counts = {'구분': gu, '경상': 0, '중상': 0, '사망': 0}
    for i, row in df.iterrows():
        if gu in row['시군구'] and row['피해운전자 상해정도'] in ['경상', '중상
, '사망']:
            counts[row['피해운전자 상해정도']] += 1
    data.append(counts)
df_counts = pd.DataFrame(data)
```

	A	B	C	D	E
1		구분	경상	중상	사망
2	0	강남구	37	33	3
3	1	강동구	30	40	1
4	2	강북구	42	49	6
5	3	강서구	43	45	5

-설명
gu_list에 서울시 구 리스트에 저장합니다

-설명
추출된 '구'로 끝나는 단어들을 가나다순으로 정렬

-설명
딕셔너리 counts를 생성를 생성하고 각 행을 반복해서 현재 행의 '시군구에 현재 구 이름(gu)이 포함되어 있고, '피해운전자 상해정도'가 '경상', 중상 ,'사망' 중 하나인 경우 피해운전자 상해정도에대한 row인 경상, 중상, 사망 중 하나 +1 합니다.

동별 코드

<CODE>

```
# '구' 그룹명을 호출받아 동/가 그룹을 생성
def create_dong_ga_group(gu_name):
    dong_ga_group = {}
    for gu_str in df['시군구']:
        if gu_name in gu_str:
            dong_ga_list = [word for word in gu_str.split() if
word.endswith('동') or word.endswith('가')]
            if dong_ga_list:
                dong_ga_name = dong_ga_list[0]
                if dong_ga_name not in dong_ga_group:
                    dong_ga_group[dong_ga_name] = []
    return dong_ga_group
# 구를 구하자!
result_set = set()
for index, row in map_data.iterrows():
    result_set.add(row["법정동명"].split(" ")[1])

for gu_name in result_set:
    # '구' 그룹명을 호출받아 동/가 그룹 생성
    dong_ga_group = create_dong_ga_group(gu_name)
```

	A	B	C	D	E
1		동별	경상	중상	사망
2	0	광희동2가	1	0	0
3	1	남대문로1가	0	1	0
4	2	남대문로3가	0	1	0
5	3	남대문로4가	5	0	0

설명

- '시군구' 항목의 구 별 그룹 뒤의 '동' 또는 '가'로 끝나는 단어를 하위 그룹화 (예시_서울특별시 영등포구 ' 영등포동2가 ')
- 구 별 그룹화된 그룹을 구 명(예시_영등포구)으로 호출하여 하위 그룹인 '동' 또는 '가'로 끝나는 단어 그룹 전체 호출
- '동'과 '가' 그룹을 가나다순으로 정렬
- '동'과 '가' 그룹에 해당하는 '피해운전자 상해정도' 항목의 '경상' '중상' '사망'에 대한 데이터만 카운트, 그 외 부상정도는 카운트하지 않음

동별 코드

<CODE>

```
# '구' 그룹별로 '동'과 '가' 그룹에 해당하는 데이터 추출
for i, row in df.iterrows():
    for dong_ga_name in dong_ga_group.keys():
        if dong_ga_name == row['시군구'].split(' ')[2]:
            if row['피해운전자 상해정도'] in ['경상', '중상', '사망']:
                # 해당 동이 포함된 구에서만 카운트
                if gu_name in row['시군구']:
                    dong_ga_group[dong_ga_name].append(row['피해
운전자 상해정도'])

# '구' 그룹별로 '동'과 '가' 그룹에 해당하는 데이터 카운트
data = []

# 그룹명을 가나다순으로 정렬
dong_ga_group_sorted = dict(sorted(dong_ga_group.items()))
for dong_ga_name, injuries in dong_ga_group_sorted.items():
    # 상해정도가 있는 동에 대해서만 데이터 추가
    if injuries:
        counts = {'동별': dong_ga_name, '경상': 0, '중상': 0, '사망':
0}
```

	A	B	C	D	E
1		동별	경상	중상	사망
2	0	광희동2가	1	0	0
3	1	남대문로1가	0	1	0
4	2	남대문로3가	0	1	0
5	3	남대문로4가	5	0	0

설명

- '시군구' 항목의 구 별 그룹 뒤의 '동' 또는 '가'로 끝나는 단어를 하위 그룹화 (예시_서울특별시 영등포구 ' 영등포동2가 ')
- 구 별 그룹화된 그룹을 구 명(예시_영등포구)으로 호출하여 하위 그룹인 '동' 또는 '가'로 끝나는 단어 그룹 전체 호출
- '동'과 '가' 그룹을 가나다순으로 정렬
- '동'과 '가' 그룹에 해당하는 '피해운전자 상해정도' 항목의 '경상' '중상' '사망'에 대한 데이터만 카운트, 그 외 부상정도는 카운트하지 않음

동별 코드

<CODE>

```
for injury in injuries:
        counts[injury] += 1
    data.append(counts)

# 데이터프레임 생성
df_counts = pd.DataFrame(data)
```

	A	B	C	D	E
1		동별	경상	중상	사망
2	0	광희동2가	1	0	0
3	1	남대문로1가	0	1	0
4	2	남대문로3가	0	1	0
5	3	남대문로4가	5	0	0

설명

- '시군구' 항목의 구 별 그룹 뒤의 '동' 또는 '가'로 끝나는 단어를 하위 그룹화 (예시_서울특별시 영등포구 ' 영등포동2가 ')
- 구 별 그룹화된 그룹을 구 명(예시_영등포구)으로 호출하여 하위 그룹인 '동' 또는 '가'로 끝나는 단어 그룹 전체 호출
- '동'과 '가' 그룹을 가나다순으로 정렬
- '동'과 '가' 그룹에 해당하는 '피해운전자 상해정도' 항목의 '경상' '중상' '사망'에 대한 데이터만 카운트, 그 외 부상정도는 카운트하지 않음

노인의 도로별 피해운전자 상해정도 코드

<CODE>

```
# 데이터 읽기
df = pd.read_excel('노인데이터.xlsx', sheet_name='노인운전자사고')

selected_data = df[df["피해운전자 상해정도"].isin(["사망", "중상", "경상"])]
# 건조, 젖음/습기, 서리/결빙 일 때 사망, 중상, 경상의 피해운전자 수 계산
severity_counts = selected_data.groupby(['피해운전자 상해정도', '노면상태']).size().unstack()

severity_counts.drop(columns=['기타'], inplace=True) # "기타" 열 제거
transposed_df = severity_counts.T # 행열 전환
transposed_df.fillna(0, inplace=True) # NaN을 0으로 변경

transposed_df_reset = transposed_df.reset_index()
```

	A	B	C	D	E
1		노면상태	경상	사망	중상
2	0	건조	3,597	19	957
3	1	서리/결빙	9	0	2
4	2	적설	3	0	0
5	3	젖음/습기	261	5	60

설명

df['column'].isin(['A', 'B', 'C']) => 자바의 list.contains()와 같은 기능

df.groupby(['column1', 'column2']).size() => 데이터프레임을 그룹화한 후, 각 그룹별로 레코드의 개수를 세어주는 메서드

df.unstack() => 다중 인덱스를 가진 데이터프레임을 풀어주는 메서드

df.drop(columns=[], inplace=True) => columns에 해당하는 열을 지워줍니다.

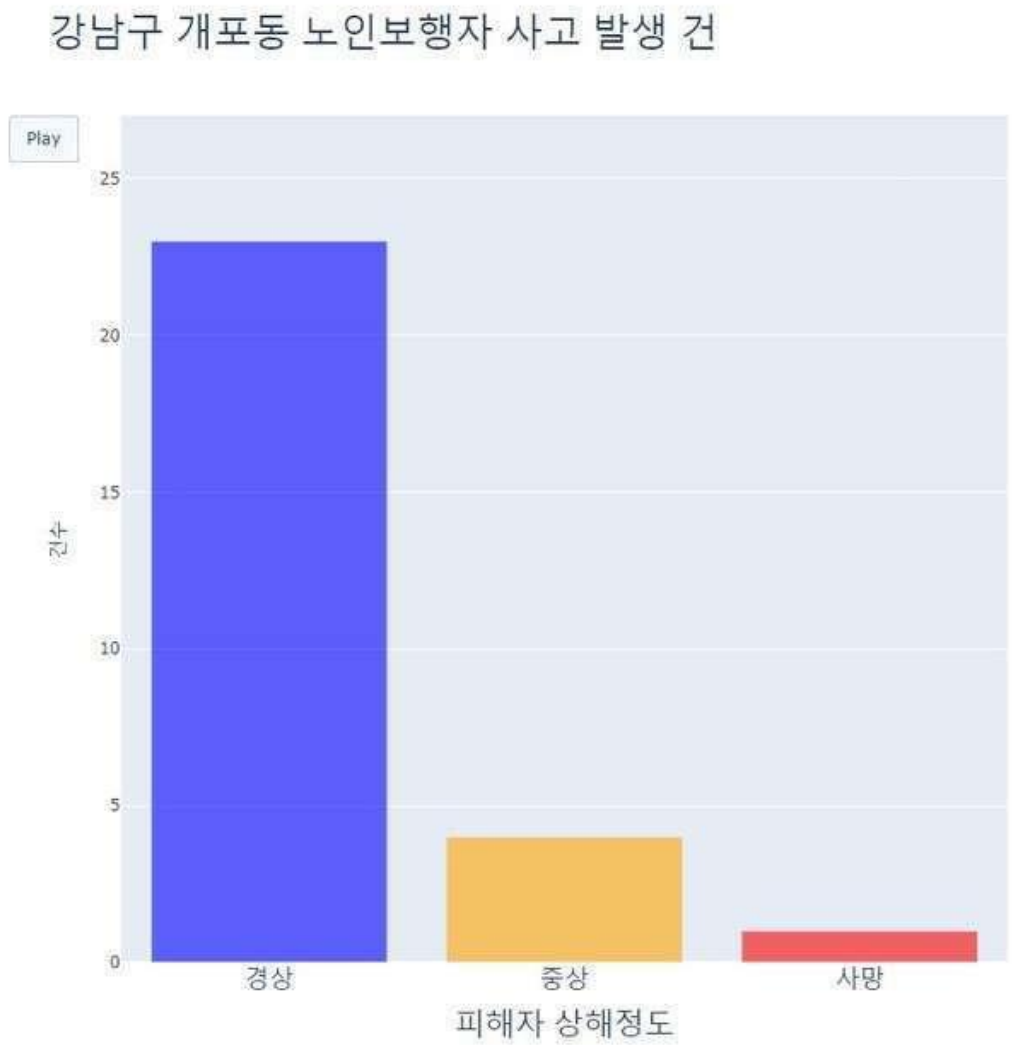
df.T => 행과 열을 전환해 줍니다.

df.fillna(value, inplace=True) => NaN인 값을 value로 채우기

df.reset_index() => 인덱스를 초기화해주는 메서드

inplace로 데이터를 직접 변경할지 여부를 boolean으로 확인

프로젝트 데이터 시각화 그래프



〈코드〉

```
# 각 프레임의 데이터 생성
        frames = []
        for i in range(51):
            frame_data = {
                'Year': data['Year'],
                'Value': hh(data['Value'], i, 50)
            }
```

- 설명

데이터를 조작하여 다양한 프레임을 생성합니다

```
            frame = go.Frame(data=[go.Bar(x=frame_data['Year'],
y=frame_data['Value'])])
            frames.append(frame)
```

- 설명

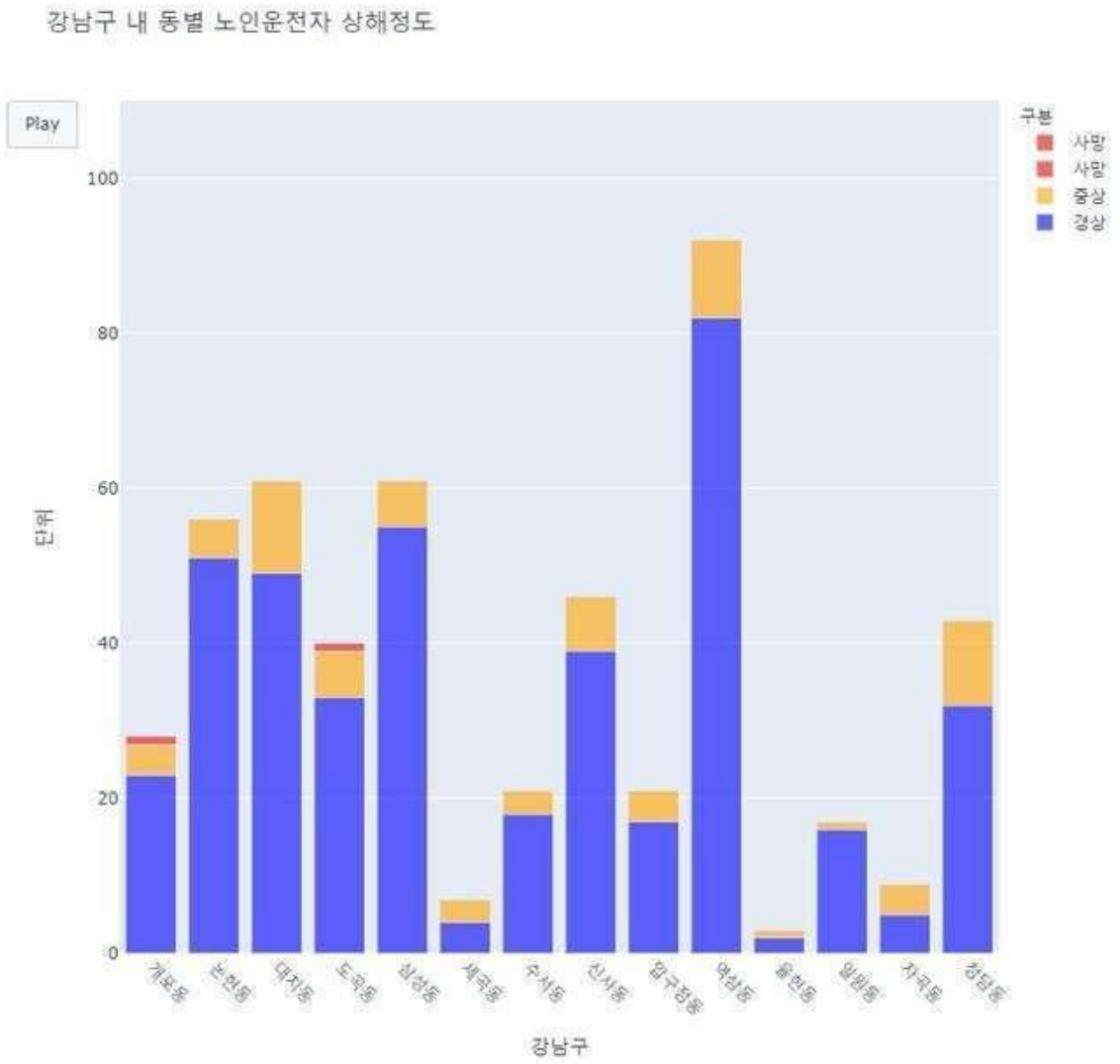
frame_data를 사용하여 막대 그래프를 생성하고

```
        fig = go.Figure(
            data=[go.Bar(x=data['Year'], y=[0] * len(data['Year']))],  # 초기 상태, 모든 값
은 0으로 설정
            frames=frames,
        )
```

- 설명

모든 프레임이 추가된 후, 이러한 프레임들을 사용하여 Figure 객체를 생성합니다

프로젝트 데이터 시각화 그래프



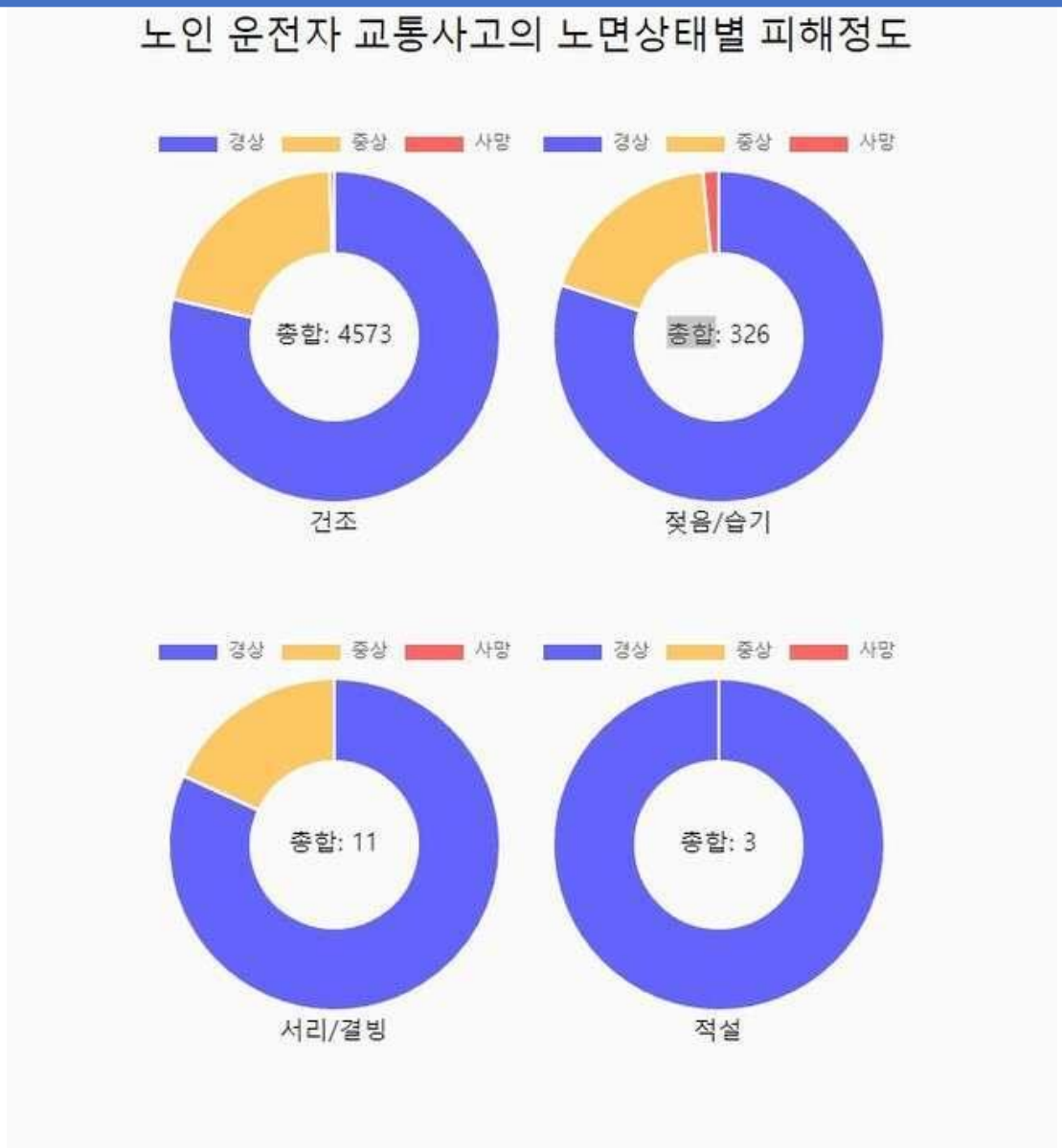
〈코드〉

```
# 프레임에 찍을 데이터 준비
  frames_data = []
  for i in range(1,51):
    nth_data = {
      '동별': df_counts['동별'],
      '경상': hh(df_counts['경상'],i,50),
      '중상': hh(df_counts['중상'],i,50),
      '사망': hh(df_counts['사망'],i,50)}
    frames_data.append(nth_data)
  # 프레임 만들고
  frames = []
  colors = {'경상':'blue','중상':'orange', '사망':'red'}
  for i in range(50):
    data = frames_data[i]
    frame_data = []
    # 각 상태별로 막대 그래프를 추가
    for status in ['경상', '중상', '사망']:
      frame_data.append(go.Bar(x=data['동별'],
y=data[status], name=status,
marker_color=colors[status]))
    frame = go.Frame(data=frame_data)
    frames.append(frame)
fig = go.Figure(
    data=[go.Bar(x=df_counts['동별'])],
    frames=frames)
```

```
# 각 상태별로 막대 그래프를 추가
  for status in ['경상', '중상', '사망']:
    fig.add_trace(go.Bar(x=df_counts['동별'],
y=init_data[status], name=status,
marker_color=colors[status]))
frames = []
        for i in range(51):
          frame_data = {
            'Year': data['Year'],
            'Value': hh(data['Value'], i, 50) }
          frame =
go.Frame(data=[go.Bar(x=frame_data['Year'],
y=frame_data['Value'])])
          frames.append(frame)
        fig = go.Figure(
          data=[go.Bar(x=data['Year'], y=[0]
* len(data['Year']))],  # 초기 상태, 모든 값은 0으
로 설정
          frames=frames, )
```

위를 통해 그래프의 프레임을 50분/1씩하여 데이터를 넣어 누적 막대를 만든다.

프로젝트 데이터 시각화 그래프

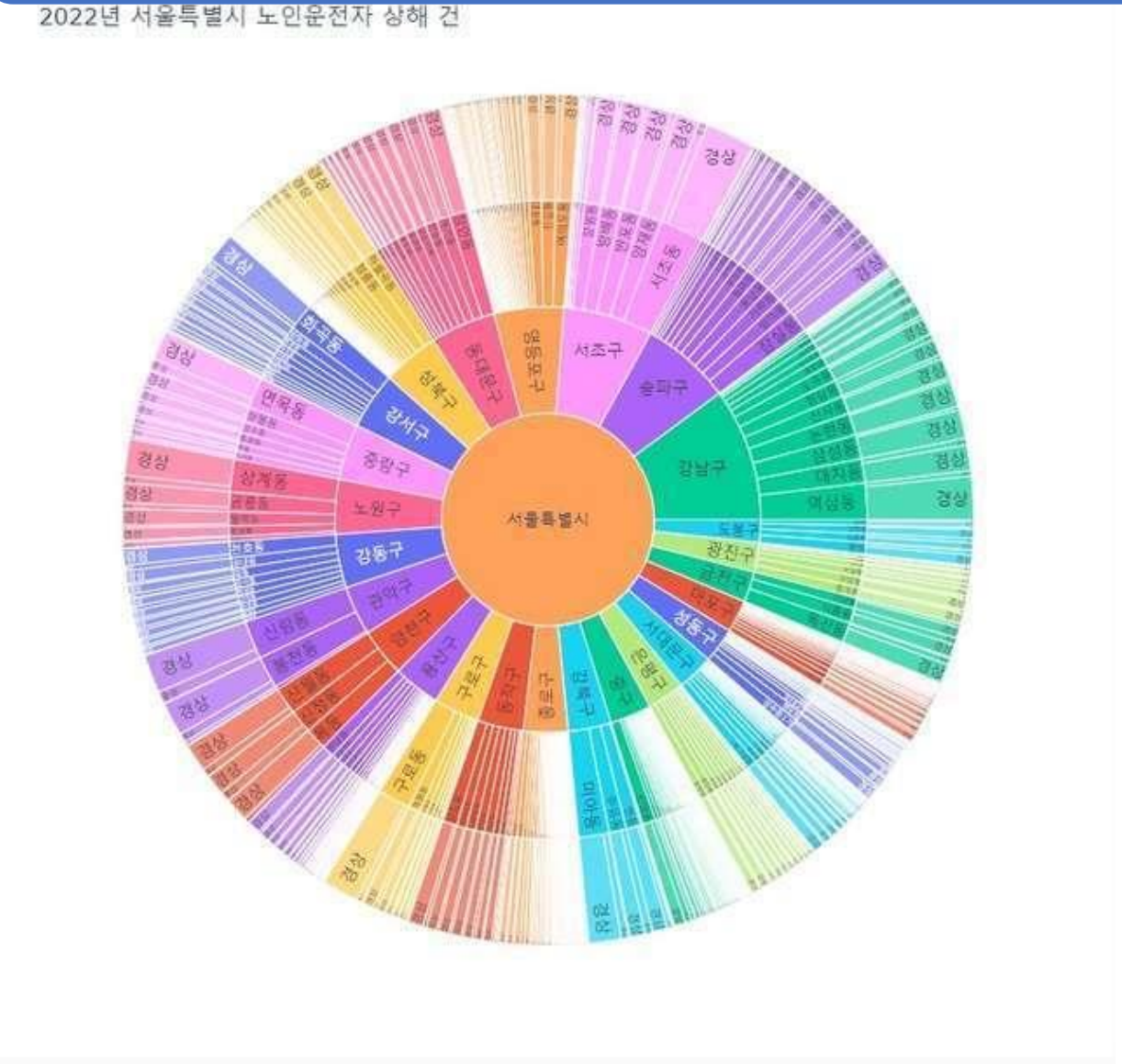


〈코드〉

```
var ctx1 = document.getElementById('donut-chart1').getContext('2d');
  var myDonutChart1 = new Chart(ctx1, {
    type: 'doughnut',
    data: data,
    options: {
      title: {
        display: true,
        text: '노인 운전자 교통사고 피해정도'
      },
      cutoutPercentage: 50 // 도넛 차트의 가운데 부분을 빈 공간으로 설정합니다.
    }
  });
  fetch('road_D_pie.json')
      .then(response => response.json())
      .then(datas => {
        data = datas[0]
        let tt = data['경상']+ data['중상'] + data['사망']
        // 데이터 업데이트
        myDonutChart1.data.datasets[0].data = Object.values(data);
        let total = document.querySelector('#total1')
        total.innerHTML=`총합: ${tt}`
        myDonutChart1.update();
```

위 코드를 통해 json파일 데이터 넣기

프로젝트 데이터 시각화 그래프



<코드>

```
fig = px.sunburst(df_filtered,
                  path=['그룹1', '그룹2', '그룹3', '피해운전자 상해 정도'],
                  title="2022년 서울특별시 노인운전자 상해 건",
                  color='그룹2 색상')
   # 색상 할당
   df_filtered['그룹2 색상'] = df_filtered[그룹2'].map(group2_color_mapping)
```

Px.sunburst를 통해 그래프 생성

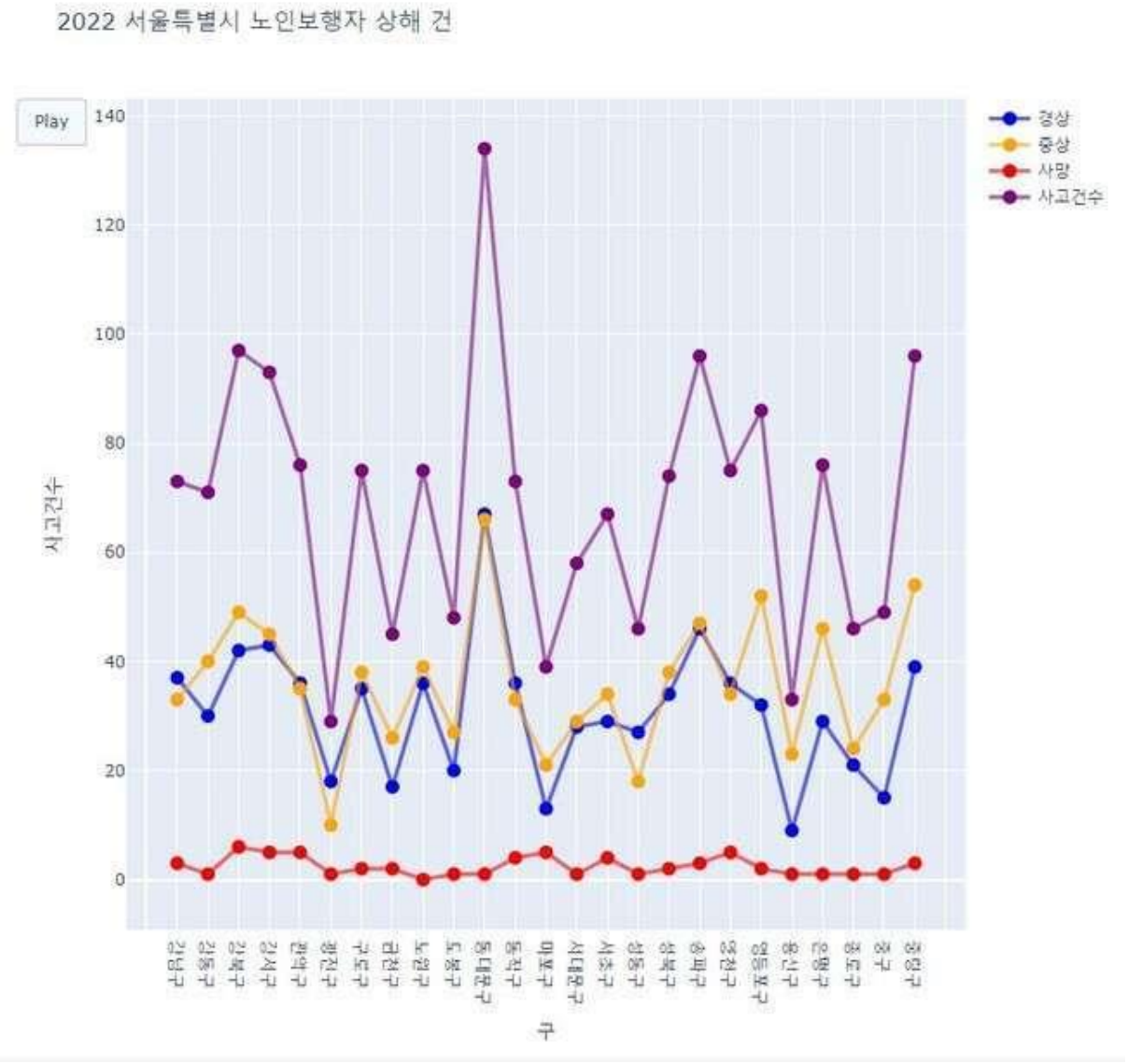
프로젝트 데이터 시각화 그래프

<코드>

```
# 선 그래프 생성
fig = px.line(all_counts, x='월_그룹명', y='건수', color='피해운전자 상해정도',
              title='월 별 노인운전자 상해정도', labels={'월_그룹명': '2022년', '건수': '상해 건수'},
              line_dash_sequence=['solid', 'dot', 'dash'],
              color_discrete_map={'경상': 'blue', '중상': 'orange', '사망': 'red'})
# Simple Slider Control 추가
fig.update_layout(
  xaxis=dict(title='2022년 서울특별시 노인보행자 사고 건', title_font=dict(size=18, family='Arial',
color='black'), tickfont=dict(size=14)),
  yaxis=dict(title='상해 건수', title_font=dict(size=18, family='Arial', color='black'),
tickfont=dict(size=14)),
  title=dict(text='월 별 노인보행자 상해정도', font=dict(size=20, family='Arial', color='black')),
  legend=dict(title='피해보행자 상해정도', font=dict(size=14, family='Arial', color='black')),
  sliders=[
    dict(
      active=0,
      currentvalue={"prefix": "노인보행자 상해정도: "},
      pad={"t": 50},
      steps=[
        dict(label='전체', method="update", args=[{"visible": [True, True, True]}]),
        dict(label='경상', method="update", args=[{"visible": [True, False, False]}]),
        dict(label='중상', method="update", args=[{"visible": [False, True, False]}]),
        dict(label='사망', method="update", args=[{"visible": [False, False, True]}])]
    )
  ]
)
```

Simple Slider Control 추가 코드를 통해 간단한 슬라이더 컨트롤을 추가하여 데이터의 일부분만 표시하거나, 필요에 따라 다른 정보를 보여줄 수 있다.

프로젝트 데이터 시각화 그래프



<코드>

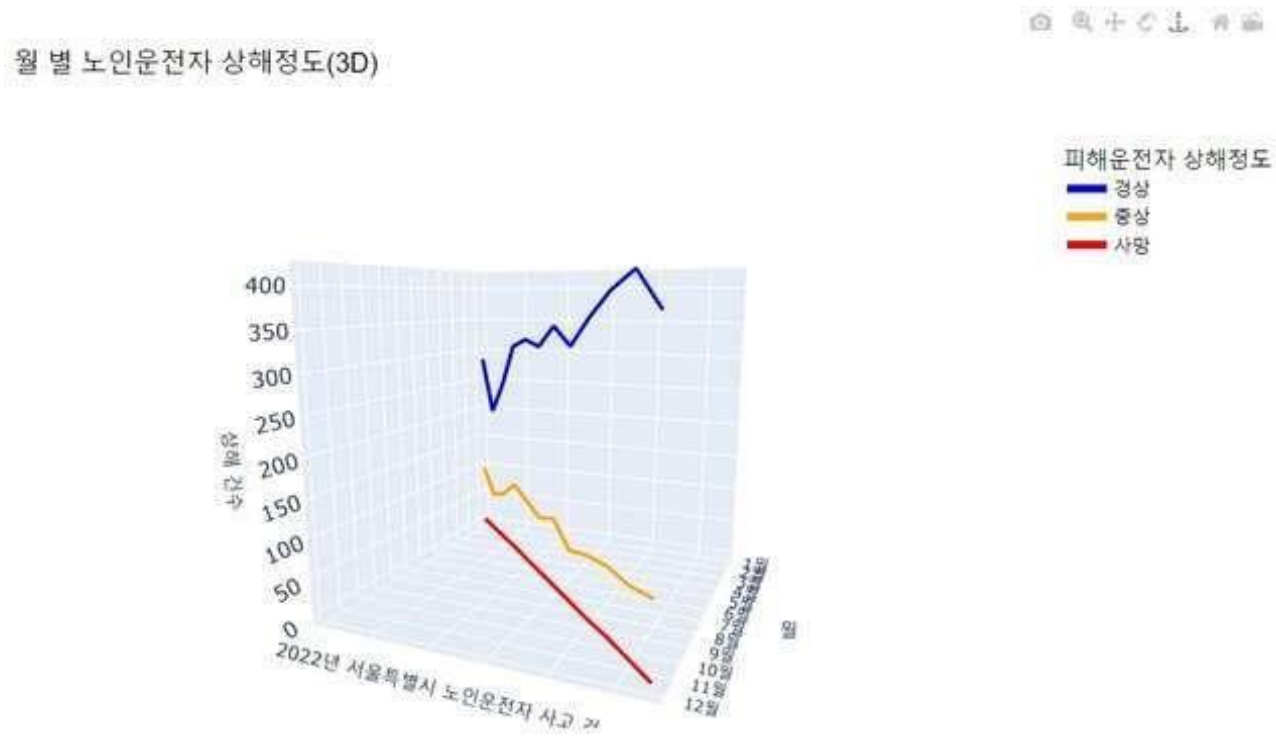
```
frames=[]
for i in range(26):
  frame_data = {'x': all_counts['구_숫자'][:i+1], 'y1': all_counts['경상'][:i+1], 'y2': all_counts['중상'][:i+1], 'y3': all_counts['사망'][:i+1], 'y4': all_counts['사고건수'][:i+1]}
 - 설명
 frame 데이터를 만들어 frame_data에 넣어준다.
  frame= go.Frame(
    data=[
      go.Scatter(x=frame['x'], y=frame['y1'], mode='lines+markers', line=dict(width=3,color='rgba(0,0,255,0.6)'), marker=dict(size=10,color='blue')),
      go.Scatter(x=frame['x'], y=frame['y2'], mode='lines+markers', line=dict(width=3,color='rgba(255,165,0,0.6)'), marker=dict(size=10,color='orange')),
      go.Scatter(x=frame['x'], y=frame['y3'], mode='lines+markers', line=dict(width=3,color='rgba(255,0,0,0.6)'), marker=dict(size=10,color='red')),
      go.Scatter(x=frame['x'], y=frame['y4'], mode='lines+markers', line=dict(width=3,color='rgba(128,0,128,0.6)'), marker=dict(size=10,color='purple')),
],
  )
-설명

frames.append(frame)

fig = go.Figure(
  data=[
    go.Scatter(x=[0], y=[0], mode='lines', name='경상'),
    go.Scatter(x=[0], y=[0], mode='lines', name='중상'),
    go.Scatter(x=[0], y=[0], mode='lines', name='사망'),
    go.Scatter(x=[0], y=[0], mode='lines', name='사고건수'),
  ],
 4개의 선 그래프에 데이터를 넣어준다

frames=frames
-설명
 선 그래프 만들어준다
```

프로젝트 데이터 시각화 그래프



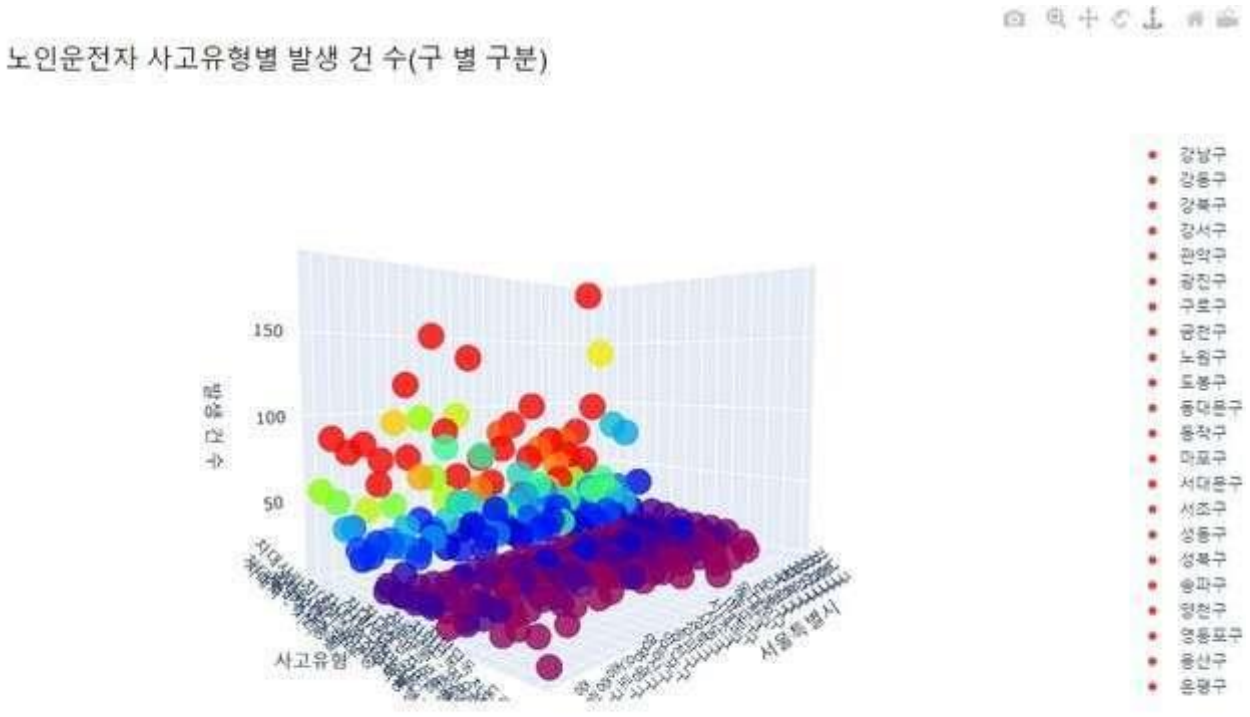
〈코드〉

```
fig.add_trace(go.Scatter3d(
      x=df_gyeongsang['월_숫자'],  # x축: 월 숫자
      y=df_gyeongsang['월_그룹명'],  # y축: 월별 그룹명
      z=df_gyeongsang['경상'],  # z축: 경상 건수
      mode='lines',  # 선으로 설정
      line=dict(width=6, color='blue'),  # 선의 너비와 색상 설정
      name='경상',  # 범례에 표시할 이름
   ))
```

〈요약 설명〉

fig.add_trace(go.Scatter3d 를 통해 3d line 그래프 생성하여
x,y,z축 설정하여 생성하고
Mode 지정으로 선 그래프 지정

프로젝트 데이터 시각화 그래프



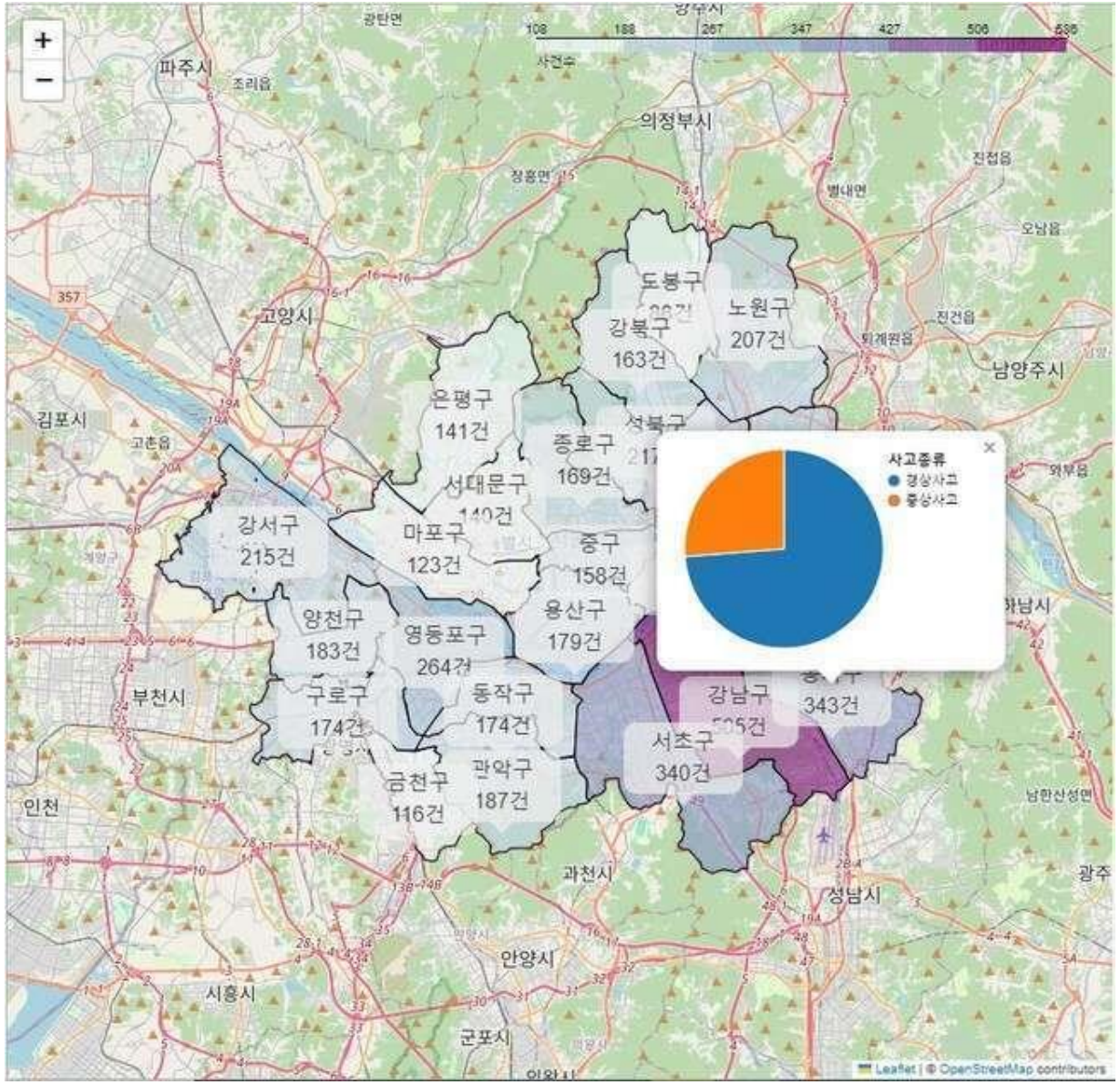
<코드>

```
fig = go.Figure()
  fig.add_trace(go.Scatter3d(
      x=[gu] * len(gu_counts),  # x축: 구별 그룹명
      y=gu_counts['사고유형'],    # y축: 사고유형
      z=gu_counts['건 수'],      # z축: 발생 건 수
      mode='markers',          # 마커 모드 설정
      marker=dict(
          size=6,              # 마커 크기 설정
          color=gu_counts['건 수'],  # 색상 설정
          colorscale='Rainbow',      # 색상 척도 설정
          opacity=0.8          # 투명도 설정
      ),
      name=gu                  # 범례에 표시할 이름
   ))
```

- 요약 설명

fig.add_trace(go.Scatter3d 를 통해 3d scatter 그래프 생성하여
x,y,z축 설정하여 생성하고
mode 지정으로 산점도 그래프 생성

프로젝트 데이터 시각화 그래프



〈코드〉

```
minx, miny, maxx, maxy = geo_data.total_bounds
center_x, center_y = (minx + maxx) / 2 , (miny + maxy) / 2 # 중심점찾기
m = folium.Map(location=[center_y, center_x], zoom_start=11) # 중심점으로 지도그리기
```

〈설명〉

geo_data를 읽어 geo_data의 Boundary에서 최대최소로 서울시 중심좌표를 찾아서 m의 중심점을 잡았다.

〈코드〉

```
for gu in result_set: # 서울시 구를 반복
    filtered_data = geo_data[geo_data["SIG_KOR_NM"] == gu]
    minx, miny, maxx, maxy = filtered_data.total_bounds
    center_x, center_y = (minx + maxx) / 2 , (miny + maxy) / 2 # 중심점 찾기
    marker = folium.Marker( # 마커표시
        location=[center_y, center_x],
        icon=DivIcon(icon_size=(w
idth, height), html="html코드가 들어갈 자리")
    ).add_to(m)
```

〈설명〉

마찬가지로 구에 해당하는 geo_data에서 Boundary의 중심값을 찾아 마커로 표시

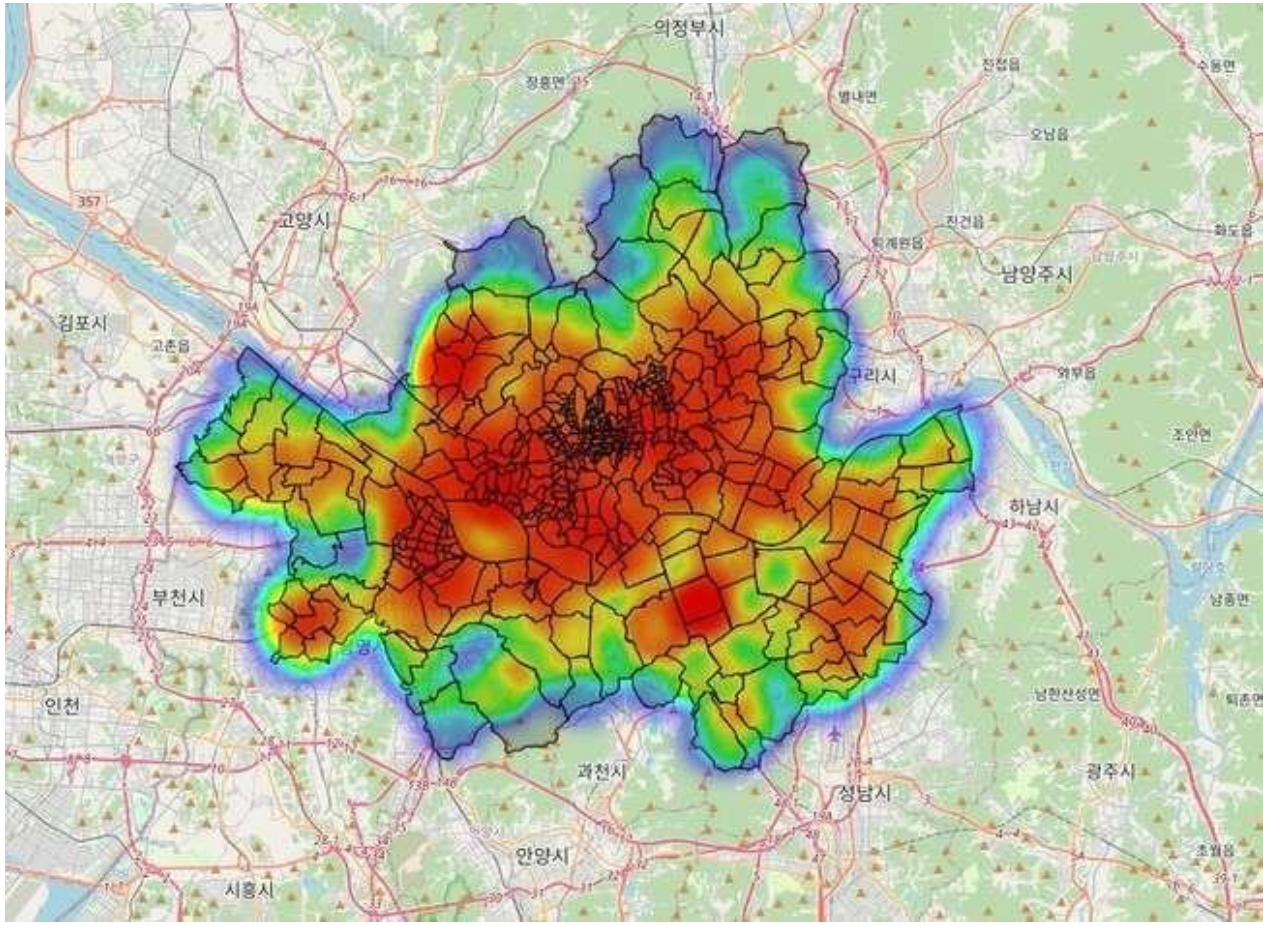
〈코드〉

```
for gu in result_set: # 서울시 구를 반복
    filtered_data = geo_data[geo_data["SIG_KOR_NM"] == gu]
    minx, miny, maxx, maxy = filtered_data.total_bounds
    center_x, center_y = (minx + maxx) / 2 , (miny + maxy) / 2 # 중심점 찾기
    marker = folium.Marker( # 마커표시
        location=[center_y, center_x],
        icon=DivIcon(icon_size=(width, height), html="html코드가 들어갈 자리")
    ).add_to(m)
```

〈설명〉

사건데이터와 지리적데이터를 연결하는 Choropleth를 지도에 추가

프로젝트 데이터 시각화 그래프



〈코드〉
```
for dong in seoul_emd['EMD_KOR_NM']:
    total_accidents = 0
    filtered_silver_data = silver_data[silver_data["시군구"].apply(lambda x: x.split()[2]) == dong]
    filtered_silver_data = filtered_silver_data[filtered_silver_data['피해운전자 상해정도'].notna()]
    filtered_silver_data = filtered_silver_data[filtered_silver_data['피해운전자 상해정도
'].str.contains('경상|중상|사망')]
    counts = filtered_silver_data['사고내용'].value_counts()
    row_num = filtered_silver_data.shape[0]
    total_accidents += row_num
    dong_data = seoul_emd[seoul_emd['EMD_KOR_NM'] == dong]
    if not dong_data.empty:
        latitude = dong_data['위도'].iloc[0]
        longitude = dong_data['경도'].iloc[0]
        # print(latitude, longitude)
        data['X'].append(longitude)
        data['Y'].append(latitude)
        data['빈도'].append(total_accidents)
```
를 통해 data 딕셔너리에 위에 for문으로 가공한 사고의 빈도를 계산하여 위도 경도 빈도를 넣어 데이터 처리하였다.

〈코드〉
```
m = folium.Map(location=[37.563383, 126.996039], zoom_start=12)
```
위도 37.563383, 경도 126.996039를 중심으로 하는 지도를 생성한다.

〈코드〉
```
heat_data = [(lat, lon, weight) for lat, lon, weight in zip(data['Y'], data['X'], data['빈도'])]
```
각각의 값을 튜플 형태로 묶어 heat_data 리스트에 저장합니다. 이때 zip 함수를 사용하여 각 리스트의 값을 순서대로 묶습니다.

〈코드〉
```
HeatMap(heat_data, radius=30).add_to(m)
```
를 통해 히트맵을 만들고 radius를 통해 나타나는 밝기의 부드러움을 표현한다.

감사합니다