



[LAB-06] 1. LinePlot



#00. 데이터 시각화 개요



1. 데이터의 패턴과 의미를 ‘눈으로 보게’ 만드는 과정

숫자·표 형태로는 보이지 않던 경향, 변화, 차이, 관계를 그래픽으로 변환해 직관적으로 이해하게 만드는 작업.

너무 화려하거나 복잡하면 오히려 데이터의 이해를 어렵게 한다.



2. 데이터 시각화의 목적

- **분석**: 데이터 속 패턴·이상치·분포·상관관계 파악
- **설명**: 분석 결과를 쉽게 전달
- **의사결정**: 인사이트를 바탕으로 더 나은 선택을 가능하게 함



3. 데이터 시각화 구분

분류	설명	대표 목적	주로 사용하는 seaborn 함수
시계열(Time-series)	시간의 흐름에 따라 값이 어떻게 변하는지 표현	추세/변동성/계절성 분석	lineplot()
분포(Distribution)	하나의 변수 값이 어떻게 펴져 있는지 표현	패턴, 이상치, 형태(정규성) 확인	boxplot(), histplot(), kdeplot(), violinplot(), ecdfplot()
범주형(Categorical)	그룹 간 차이를 비교하거나 범주의 빈도 확인	집단 비교, 비율 분석	barplot(), countplot(), violinplot(), swarmplot()
관계(Relationship)	두 변수(연속형/범주형)의 관계를 표현	상관, 패턴, 경향성 파악	scatterplot(), regplot(), lineplot(), jointplot()
행렬(Matrix)	여러 변수의 조합을 표 형태로 표현하거나 상관을 열지도 형태로 시각화	변수 상관성, 패턴 구조 파악	heatmap(), clustermap()
다변량(Multivariate)	3개 이상의 변수 간 관계를 한번에 확인	고차원 구조 탐색, 분포·관계 종합 분석	pairplot(), jointplot(), FacetGrid(), catplot(), relplot()



4. 데이터 시각화에 필요한 주요 객체

- 기본형 : matplotlib만 사용

- 응용형태 : matplotlib + pandas(DataFrame)
- 확장형 : matplotlib + seaborn

개념	비유	이유
Figure	큰 캔버스·화판	전체 그림을 담는 공간
Axes	개별 작업 구역(스케치북 페이지)	그래프가 실제로 그려지는 공간
matplotlib 기본 함수	기본 봇·연필	전부 직접 컨트롤, 자유도 높음
seaborn	고급 봇·전문 도구 세트	기본 스타일링·색감·레이아웃 자동화

visual

visual



#01. 준비작업



1. 라이브러리 참조

matplotlib, seaborn 패키지가 설치되어야 한다.

```
$ pip install --upgrade matplotlib seaborn
```

```
from hossam import load_data

# 글꼴을 시스템에 등록
from matplotlib import font_manager as fm

# 캔버스(figure)를 생성, 기본 그래픽 함수 제공
from matplotlib import pyplot as plt

# 고급 그래픽 기능 제공
import seaborn as sb
```



2. 시스템 전역 설정

아래의 코드는 컴퓨터마다 1회만 수행하면 된다.

```
font_path = "./NotoSansKR-Regular.ttf"          # 한글을 지원하는 폰트 파일
                                                # 경로
fm.fontManager.addfont(font_path)               # 폰트의 글꼴을 시스템에 등
                                                # 록함
```

```
font_prop = fm.FontProperties(fname=font_path) # 폰트의 속성을 읽어옴
font_name = font_prop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
font_name # 글꼴 이름 확인
```

'Noto Sans KR'



3. 그래프 설정

아래 코드는 ipynb 파일 하나당 한번만 수행하면 된다.

이와 같은 형식의 코드가 다시 실행되기 전까지 현재 소스파일의 모든 그래프에 전역으로 적용되는 설정.

theme 종류: whitegrid, darkgrid, dark, white(기본값)

```
my_dpi = 120 # 이미지 선명도를 결정하는 1인치
               # 당점(픽셀)의 수
my_font_name = "Noto Sans KR" # 시스템에 등록된 글꼴 이름
my_theme = "dark" # 그림 스타일 지정

sb.set_theme(style=my_theme) # seaborn 스타일 (화풍 설정하기)

plt.rcParams['font.family'] = my_font_name # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 16 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지(한글환경에서 필수)
```



#02. 그래프 기본 코드 구성

```
# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px = 1280 # 그래프 가로 크기
height_px = 760 # 그래프 세로 크기
rows = 1 # 도화지의 행 수
cols = 1 # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 그래프의 도화지 상태 확인용 테스트 코드
#print(ax)
```

```

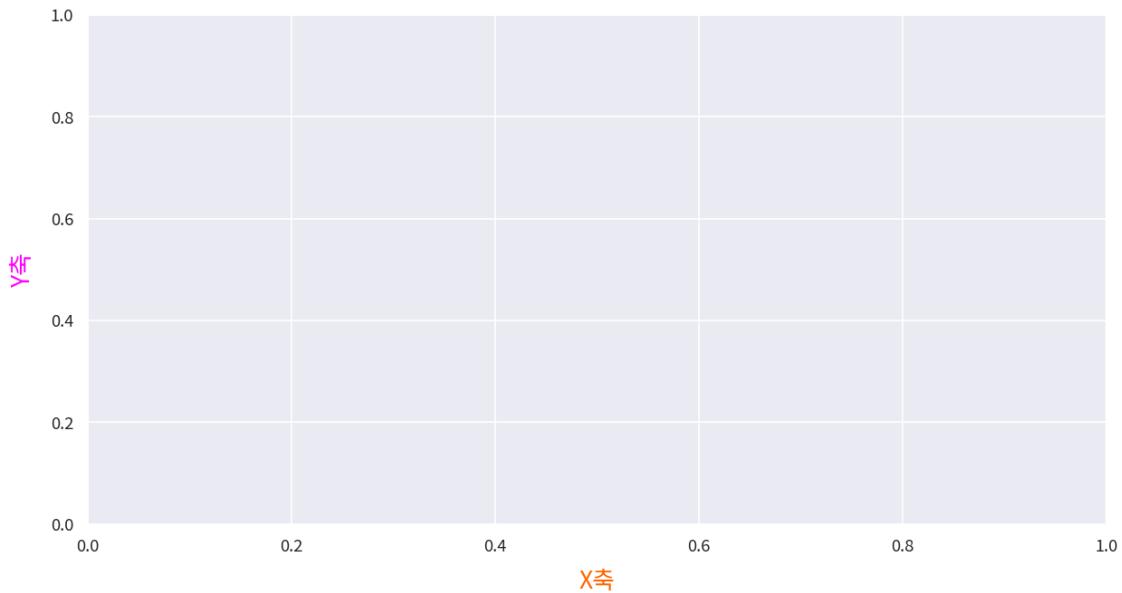
# 2) 그래프 그리기 -> seaborn 사용
# ...

# 3) 그래프 꾸미기 -> 도화지(ax)에 직접 적용
# color: 글자 색상 (기본값은 black)
# fontsize: 글자 크기 (기본값은 plt.rcParams['font.size'] 설정을 따름)
# pad, labelpad : 그래프와의 간격
# fontweight: 글자 굵기 (100~1000 사이 100단위 값). 글꼴이 지원하는 경우만 적용됨
ax.set_title("제목입니다", color="#0066ff", fontsize=22,
             fontweight=1000, pad=20)
ax.set_xlabel("X축", color="#ff6600", fontsize=16, labelpad=10)
ax.set_ylabel("Y축", color="#ff00ff", fontsize=16, labelpad=10)

# 4) 출력
plt.grid()                                     # 배경 격자 표시/숨김 (테마에 따라 다
                                               # 름)
plt.tight_layout()                             # 여백 제거
plt.savefig("myplot.png", dpi=my_dpi)          # 생략 가능
plt.show()                                     # 그래프 화면 출력
plt.close()                                     # 그래프 작업 종료

```

제목입니다



png



#03. Line Plot

라인 플롯은 하나의 변수가 시간의 흐름이나 순서 등에 따라 어떻게 변화하는지 보여주기 위해 사용한다.



1. 기본 그리기

그래프에 표시될 데이터를 리스트 등의 연속형 자료형으로 지정

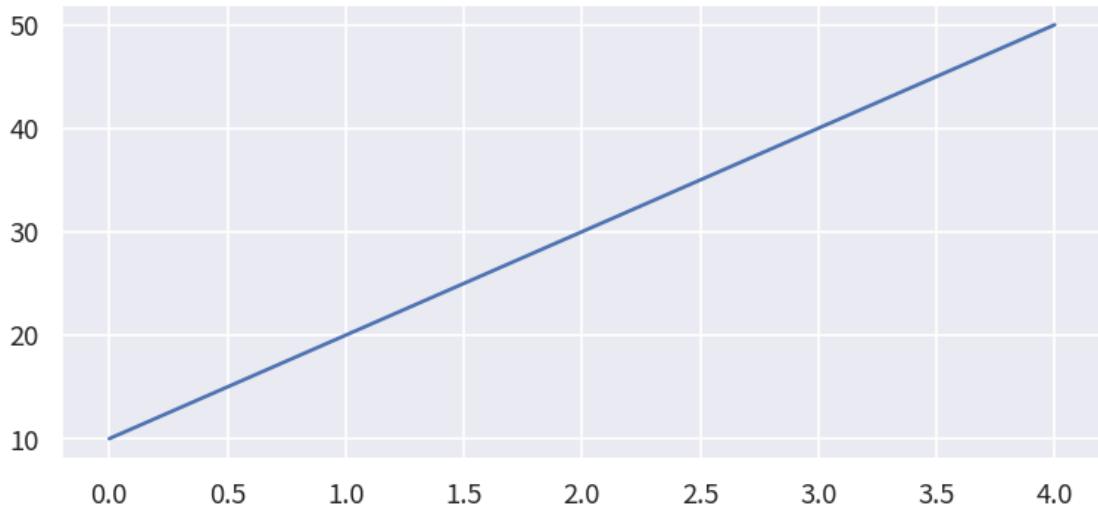
값은 y축이 되고, 인덱스는 x축이 된다.

```
# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px  = 800          # 그래프 가로 크기
height_px = 400          # 그래프 세로 크기
rows = 1                  # 도화지의 행 수
cols = 1                  # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot([10, 20, 30, 40, 50])

# 3) 그래프 꾸미기 -> 여기서는 생략

# 4) 출력
plt.grid()                # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()          # 여백 제거
plt.show()                 # 그래프 화면 출력
plt.close()                # 그래프 작업 종료
```



png



2. x, y축 및 선 모양 지정하기

x축과 y축에 모두 리스트 지정

파라미터 이름	파라미터 약자	의미
color	c	선 색깔
linestyle	ls	선 스타일
linewidth	lw	선 굵기
marker		마커 종류
markersize	ms	마커 크기
markerfacecolor	mfc	마커 내부 색깔
markeredgecolor	mec	마커 선 색깔
markeredgewidth	mew	마커 선 굵기

```

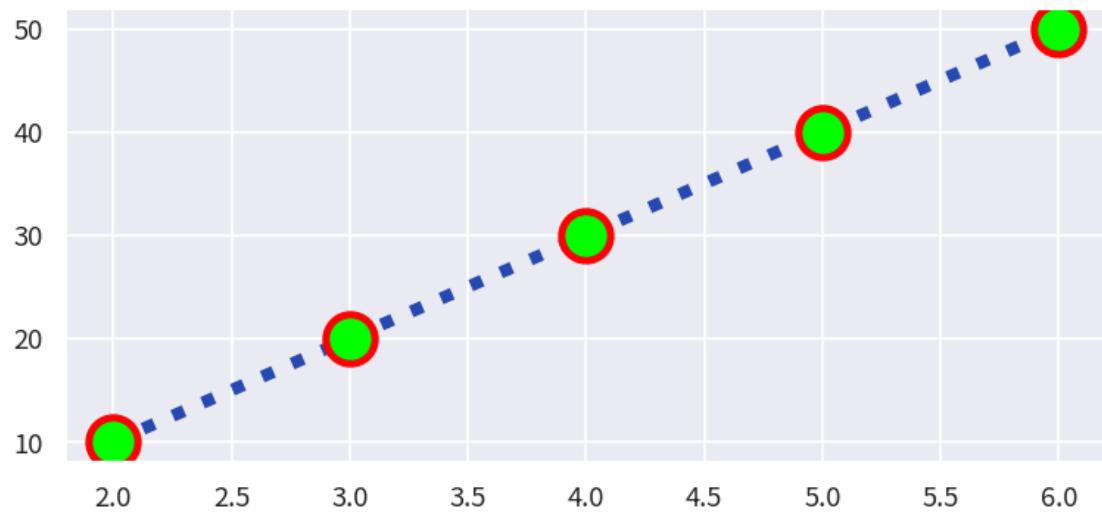
# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px = 800           # 그래프 가로 크기
height_px = 400          # 그래프 세로 크기
rows = 1                  # 도화지의 행 수
cols = 1                  # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot(x=[2, 3, 4, 5, 6], y=[10, 20, 30, 40, 50],
             c="#2548b1", linestyle=':', linewidth=5,
             marker="o", markersize=20, markerfacecolor="#00ff00",
             markeredgecolor="#ff0000", markeredgewidth=3)

# 3) 그래프 꾸미기 -> 여기서는 생략

# 4) 출력
plt.grid()                # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()         # 여백 제거
plt.show()                 # 그래프 화면 출력
plt.close()                # 그래프 작업 종료

```



png



[1] 선 스타일

선 스타일 문자열	의미
-	실선(solid)
--	대시선(dashed)
:	점선(dotted)
-.	대시-점선(dash-dit)



[2] 마커

데이터 위치를 나타내는 기호를 마커(marker)라고 한다. 마커의 종류는 다음과 같다.

마커 문자열	의미
.	point marker
,	pixel marker
o	circle marker
v	triangle_down marker
^	triangle_up marker
<	triangle_left marker
>	triangle_right marker
1	tri_down marker
2	tri_up marker
3	tri_left marker

4	tri_right marker
s	square marker
p	pentagon marker
*	star marker
h	hexagon1 marker
H	hexagon2 marker
+	plus marker
x	x marker
D	diamond marker
d	thin_diamond marker



3. 축 범위 설정

```

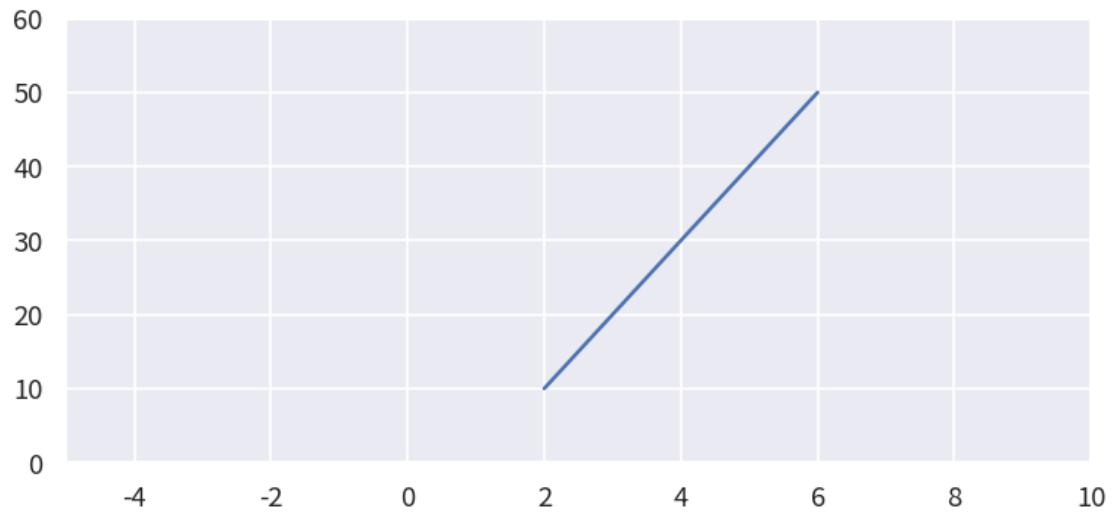
# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px = 800           # 그래프 가로 크기
height_px = 400          # 그래프 세로 크기
rows = 1                  # 도화지의 행 수
cols = 1                  # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot(x=[2, 3, 4, 5, 6], y=[10, 20, 30, 40, 50])

# 3) 그래프 꾸미기
ax.set_xlim([-5, 10])    # x축 범위
ax.set_ylim([0, 60])     # y축 범위

# 4) 출력
plt.grid()                # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()         # 여백 제거
plt.show()                 # 그래프 화면 출력
plt.close()                # 그래프 작업 종료

```



png



4. 각 축의 표시 내용 설정

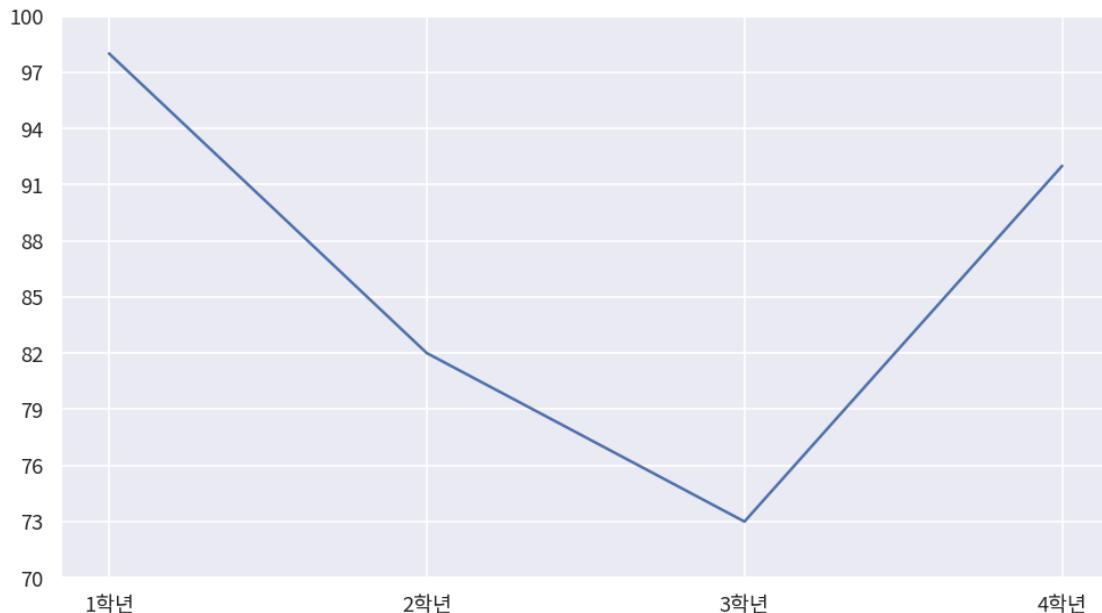
```
# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px    = 1024          # 그래프 가로 크기
height_px   = 640           # 그래프 세로 크기
rows = 1                  # 도화지의 행 수
cols = 1                  # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot(x=[1, 2, 3, 4], y=[98, 82, 73, 92])

# 3) 그래프 꾸미기
ax.set_title("철수의 학년별 평균 점수 변화", pad=15, fontsize=24)
# x축 좌표에 따른 표시할 문자열 지정
ax.set_xticks([1, 2, 3, 4], ['1학년', '2학년', '3학년', '4학년'])
ax.set_yticks(range(70, 101, 3), range(70, 101, 3))

# 4) 출력
plt.grid()                 # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료
```

철수의 학년별 평균 점수 변화



png

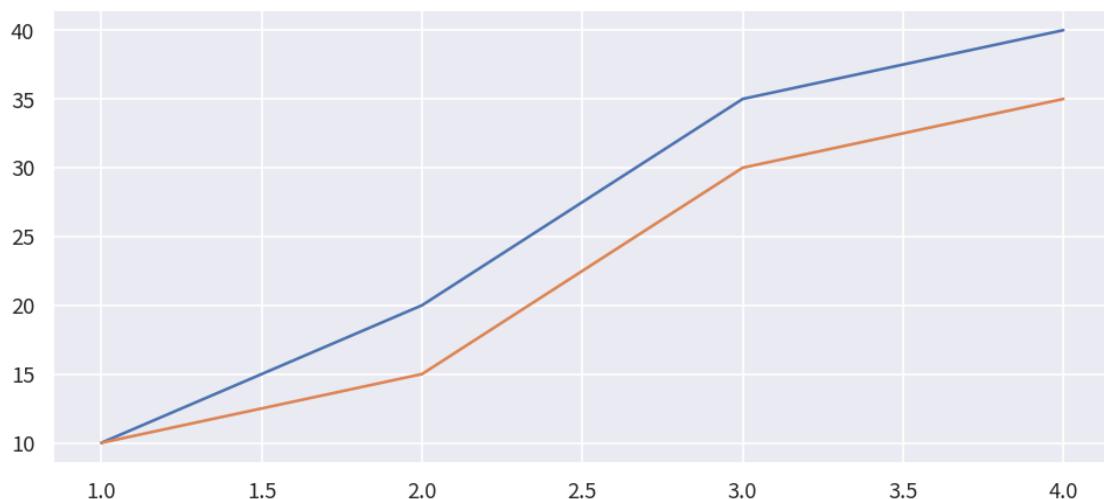
■ #04. 다중 선 그래프

```
# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px  = 1000           # 그래프 가로 크기
height_px = 480            # 그래프 세로 크기
rows      = 1               # 도화지의 행 수
cols      = 1               # 도화지의 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot(x=[1, 2, 3, 4], y=[10, 20, 35, 40])
sb.lineplot(x=[1, 2, 3, 4], y=[10, 15, 30, 35])

# 3) 그래프 꾸미기 (생략)

# 4) 출력
plt.grid()                 # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료
```



png

#05. 예제: 교통사고 발생건수 시각화



1. 데이터 가져오기

```
origin = load_data('traffic_acc')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab04/traffic_acc.xlsx
[94m[desc] [0m 2005년 1월부터 2018년 12월까지 월별 교통사고의 발생건수, 부상자
수, 사망자수 데이터(인덱스/메타데이터 없음, 출처: 공공데이터포털)
[91m[!] Cannot read metadata [0m
```

	년도	월	발생건수	사망자수	부상자수
0	2005	1	15494	504	25413
1	2005	2	13244	431	21635
2	2005	3	16580	477	25550
3	2005	4	17817	507	28131
4	2005	5	19085	571	29808
...
163	2018	8	18335	357	27749
164	2018	9	18371	348	27751
165	2018	10	19738	373	28836

166	2018	11	19029	298	28000
167	2018	12	18010	323	26463

168 rows × 5 columns



2. 데이터 전처리

```
df = origin.drop('월', axis=1).groupby('년도').mean()
df
```

	발생건수	사망자수	부상자수
년도			
2005	17847.583333	531.333333	28519.416667
2006	17812.083333	527.250000	28352.416667
2007	17638.500000	513.833333	27992.166667
2008	17985.166667	489.166667	28246.833333
2009	19332.500000	486.500000	30156.250000
2010	18906.500000	458.750000	29371.500000
2011	18475.916667	435.750000	28449.250000
2012	18638.000000	449.333333	28713.750000
2013	17946.166667	424.333333	27392.583333
2014	18629.333333	396.833333	28124.750000
2015	19336.250000	385.083333	29200.000000
2016	18409.750000	357.666667	27643.333333
2017	18027.916667	348.750000	26902.416667
2018	18095.666667	315.083333	26919.750000



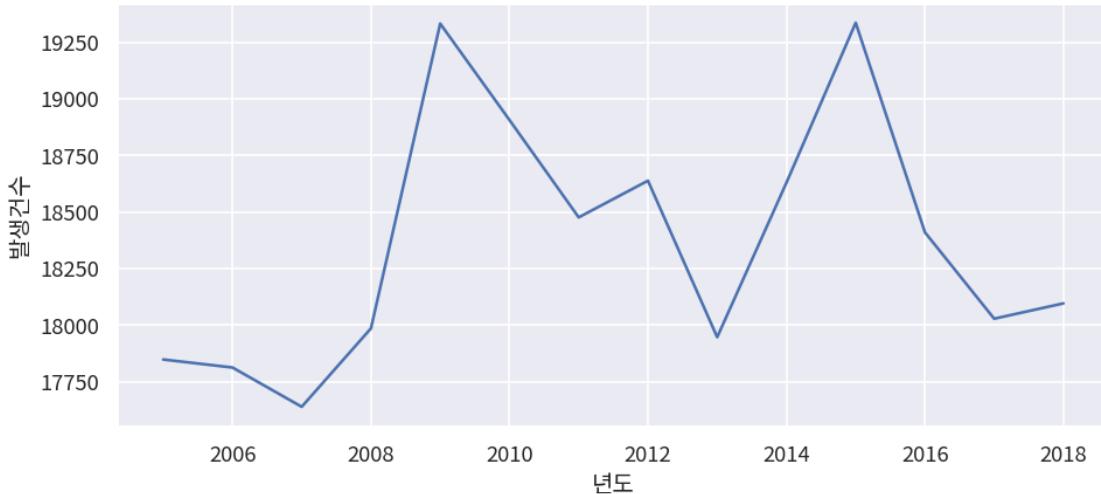
3. 교통사고 발생자 수 변화

```
# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px = 1000           # 그래프 가로 크기
height_px = 480            # 그래프 세로 크기
rows = 1                   # 도화지의 행 수
cols = 1                   # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

```
# 2) 그래프 그리기 -> seaborn 사용  
sb.lineplot(data=df, x=df.index, y='발생건수')
```

```
# 3) 그래프 꾸미기 (생략)
```

```
# 4) 출력  
plt.grid() # 배경 격자 표시/숨김 (테마에 따라 다름)  
plt.tight_layout() # 여백 제거  
plt.show() # 그래프 화면 출력  
plt.close() # 그래프 작업 종료
```



png



연습문제



1. Covid19 확진자수 변동 추이 시각화

covid19_active 데이터는 2022년 5월 1일부터 2023년 5월 31일까지 서울과 전국의 Covid19 일일 확진자 수를 기록한 데이터이다. 조사 기간동안 서울과 전국의 확진자 수가 어떻게 변화하고 있는지에 대한 추이를 시각화 하고 시각화 결과에서 얻을 수 있는 객관적 사실을 하나 이상 서술하시오.

단, x축에 표시되는 날짜는 30일 간격으로 표시한다.



2. 비트코인 시세 변동 추이 시각화

bitcoin 데이터는 2021년 06월 01일부터 2023년 06월 30일까지의 비트코인 시세 데이터의 일부이다.

이 데이터를 활용하여 날짜별 종가와 시가가 어떻게 변화하고 있는지 보여주고자 한다. 단, x축의 간격을 20일 간격으로 설정하여 시각화 하고 시각화 결과에서 얻을 수 있는 객관적 사실을 하나 이상 서술하시오.



[LAB-06] 11. 서브플롯 (1)

하나의 그래픽 영역을 나누어 두 개 이상의 시각화 결과물을 하나의 화면에서 표현할 수 있다.



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```



2. 그래프 전역 설정

```
my_dpi = 200
fpath = "./NotoSansKR-Regular.ttf"
fm.fontManager.addfont(fpath)
fprop = fm.FontProperties(fname=fpath)
fname = fprop.get_name()
    출
plt.rcParams['font.family'] = fname      # 이미지 선명도(100~300)
plt.rcParams['font.size'] = 6            # 한글을 지원하는 폰트 파일의 경로
plt.rcParams['axes.unicode_minus'] = False # 폰트의 글꼴을 시스템에 등록함
                                         # 폰트의 속성을 읽어옴
                                         # 읽어온 속성에서 폰트의 이름만 추
                                         # 그래프에 한글 폰트 적용
                                         # 기본 폰트 크기
                                         # 그래프에 마이너스 깨짐 방지
```



3. 데이터 가져오기

```
origin = load_data('traffic_acc')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab04/traffic_acc.xlsx
[94m[desc] [0m 2005년 1월부터 2018년 12월까지 월별 교통사고의 발생건수, 부상자
수, 사망자수 데이터(인덱스/메타데이터 없음, 출처: 공공데이터포털)
[91m[!] Cannot read metadata [0m
```

	년도	월	발생건수	사망자수	부상자수
--	----	---	------	------	------

0	2005	1	15494	504	25413
1	2005	2	13244	431	21635
2	2005	3	16580	477	25550
3	2005	4	17817	507	28131
4	2005	5	19085	571	29808
...
163	2018	8	18335	357	27749
164	2018	9	18371	348	27751
165	2018	10	19738	373	28836
166	2018	11	19029	298	28000
167	2018	12	18010	323	26463

168 rows × 5 columns



4. 데이터 전처리

각 변수를 년도별 평균값으로 전처리 한다.

```
df = origin.drop('월', axis=1).groupby('년도').mean()
df
```

	발생건수	사망자수	부상자수
년도			
2005	17847.583333	531.333333	28519.416667
2006	17812.083333	527.250000	28352.416667
2007	17638.500000	513.833333	27992.166667
2008	17985.166667	489.166667	28246.833333
2009	19332.500000	486.500000	30156.250000
2010	18906.500000	458.750000	29371.500000
2011	18475.916667	435.750000	28449.250000
2012	18638.000000	449.333333	28713.750000
2013	17946.166667	424.333333	27392.583333
2014	18629.333333	396.833333	28124.750000
2015	19336.250000	385.083333	29200.000000
2016	18409.750000	357.666667	27643.333333

2017	18027.916667	348.750000	26902.416667
2018	18095.666667	315.083333	26919.750000

#02. 서브플롯의 기본 사용

[1] 서브플롯 영역 나누기

2행 3열을 갖는 서브플롯 영역을 구성한다.

```
pyplot.subplots(행, 열 [, figsize=(가로크기, 세로크기)])
```

`plt.subplots()` 메서드에 의해 리턴되는 `fig`는 그래픽 처리 기능을 제공하는 객체이다.

`plt.subplots()` 메서드에 의해 리턴되는 `ax` 객체는 분할된 각 그래프 영역의 객체를 저장하고 있는 리스트이다.

```
# 1) 그래프 초기화
width_px    = 2000                      # 그래프 가로 크기
height_px   = 1000                       # 그래프 세로 크기
rows         = 2                          # 그래프 행 수
cols         = 3                          # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)

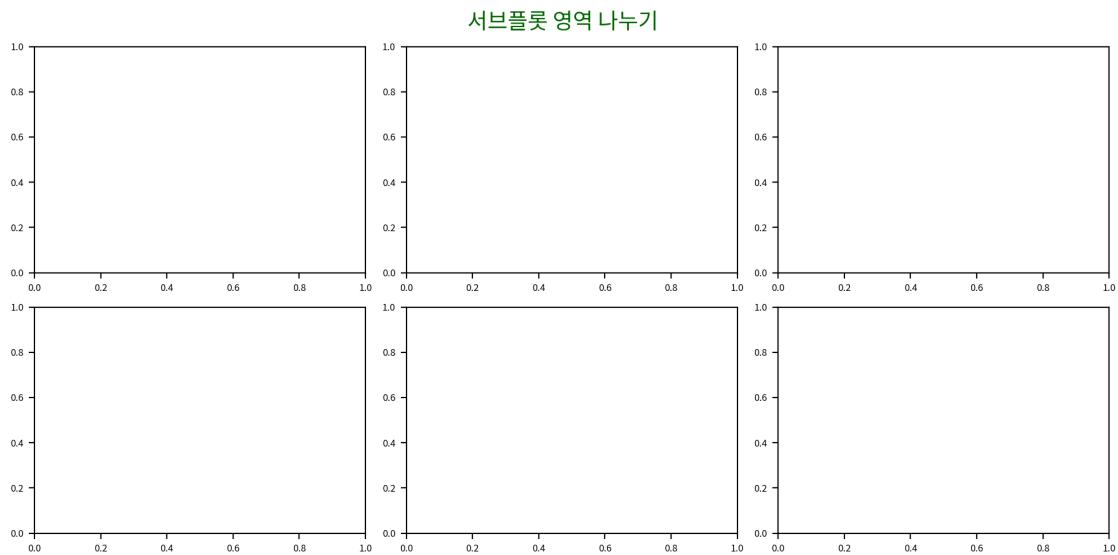
# ax 객체가 행, 열 수에 따라서 리스트가 된다.
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
# ...

# 3) 그래프 꾸미기
# 전체 제목
fig.suptitle('서브플롯 영역 나누기', fontsize=14, color='#006600')

# 각 그래프 간의 가로(wspace), 세로(hspace) 간격 지정
fig.subplots_adjust(wspace=0.2, hspace=0.2)

# 4) 출력
plt.tight_layout()                      # 여백 제거
plt.show()                             # 그래프 화면 출력
plt.close()                            # 그래프 작업 종료
```



png



[2] 서브플롯에 그래프 그리기

`plt.subplots()` 메서드의 결과로 `ax`에 반환되는 객체는 서브플롯의 행, 열에 대한 리스트이다.

```

# 1) 그래프 초기화
width_px    = 2400          # 그래프 가로 크기
height_px   = 1000          # 그래프 세로 크기
rows         = 2              # 그래프 행 수
cols         = 3              # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)
fig, ax      = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.boxplot(data=df, y='발생건수', ax=ax[0][0])
sb.histplot(data=df, x='사망자수', bins=5, ax=ax[0][1])
sb.kdeplot(data=df, x='부상자수', ax=ax[0][2])
sb.lineplot(data=df, x=df.index, y='발생건수', ax=ax[1][0])
sb.barplot(data=df, x=df.index, y='사망자수', estimator=np.sum,
           ax=ax[1][1])
sb.regplot(data=df, x='발생건수', y='사망자수', ax=ax[1][2])

# 3) 그래프 꾸미기
# 전체 제목
fig.suptitle('년도별 교통사고 집계', fontsize=14, color='#006600')

# 각 그래프 간의 가로(wspace), 세로(hspace) 간격 지정
fig.subplots_adjust(wspace=0.2, hspace=0.2)

# 첫 번째 영역 그래프의 제목, 글자크기, 색상, 격자

```

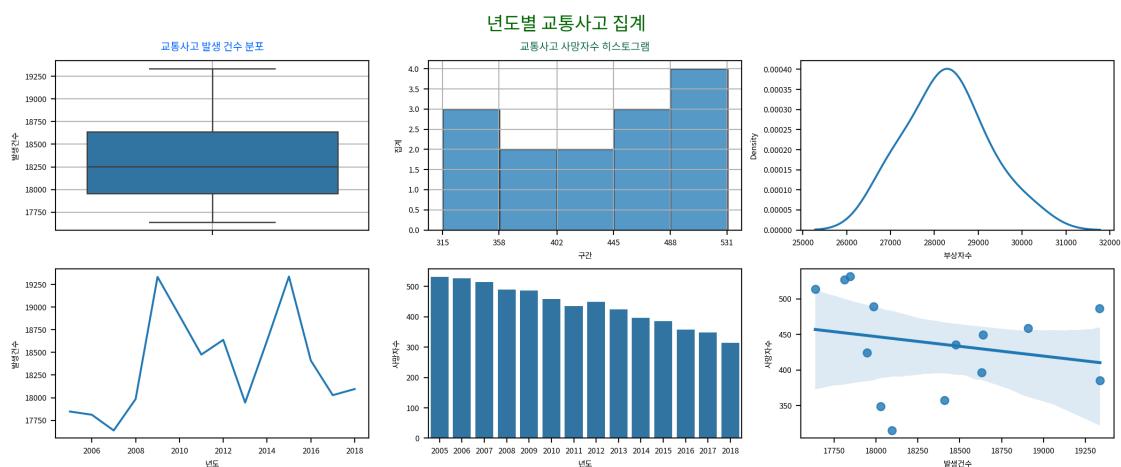
```

ax[0][0].set_title("교통사고 발생 건수 분포", color="#0066ff", fontsize=8,
                    pad=8)
ax[0][0].grid()

# 두 번째 영역 그래프의 x축 설정 및 x,y축 라벨 지정, 격자
hist, bins = np.histogram(df['사망자수'], bins=5)
bins = bins.round().astype("int")
ax[0][1].set_title("교통사고 사망자수 히스토그램", color="#0f6a46",
                    fontsize=8, pad=8)
ax[0][1].set_xticks(bins, bins)
ax[0][1].set_xlabel('구간')
ax[0][1].set_ylabel('집계')
ax[0][1].grid()

# 4) 출력
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료

```



png

#03. 두 개의 y축을 갖는 그래프



1. 샘플 데이터 만들기

1) x축 데이터 (공용)

```

x = np.arange(10)
x

```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

2) 첫 번째 y축 데이터

```
y1 = np.arange(10)  
y1
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

3) 두 번째 y축 데이터

```
y2 = x**2  
y2
```

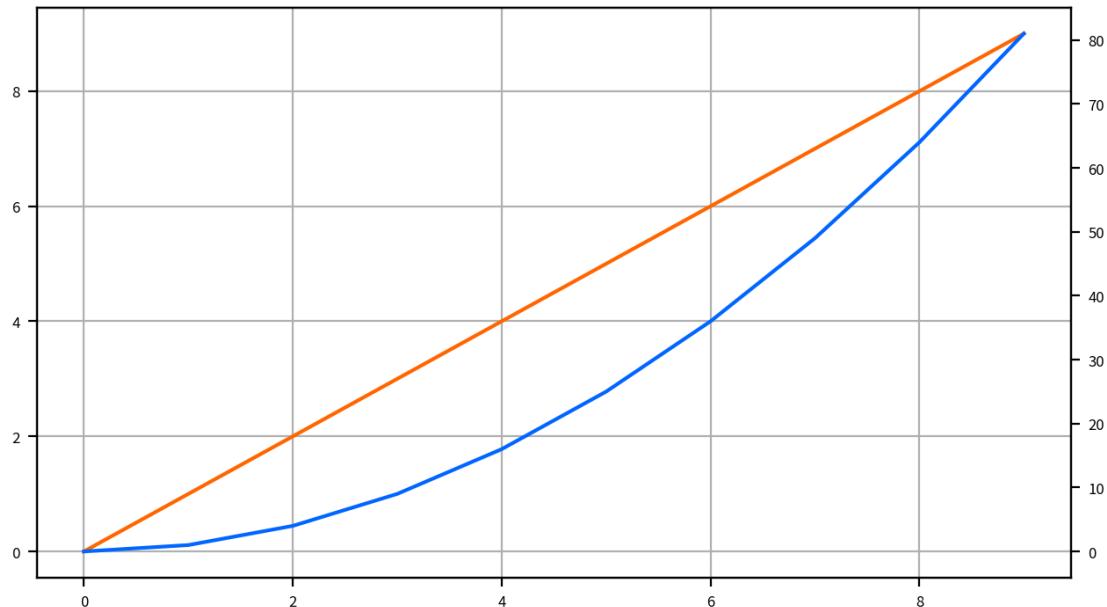
```
array([ 0, 1, 4, 9, 16, 25, 36, 49, 64, 81])
```



2. 서브플롯으로 2개의 y축을 갖는 그래프 구현

```
# 1) 그래프 초기화  
width_px = 1280 # 그래프 가로 크기  
height_px = 720 # 그래프 세로 크기  
rows = 1 # 그래프 행 수  
cols = 1 # 그래프 열 수  
figsize = (width_px / my_dpi, height_px / my_dpi)  
fig, ax1 = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)  
  
# ax1에 겹쳐지는 쌍둥이 서브플롯을 생성  
ax2 = ax1.twinx()  
  
# 2) LinePlot 그리기  
sb.lineplot(x=x, y=y1, color='#ff6600', ax=ax1)  
sb.lineplot(x=x, y=y2, color='#0066ff', ax=ax2)  
  
# 3) 그래프 꾸미기  
ax1.grid(True) # 그래프가 겹쳐지므로 격자는 하나만 표시해도 됨  
  
# 4) 출력  
plt.tight_layout() # 여백 제거
```

```
plt.show()          # 그래프 화면 출력  
plt.close()        # 그래프 작업 종료
```



png



04. 교통사고 발생건수와 사망자수 변화 시각화하기

우리나라는 2008년도에 자동차안전기준에 관한 규칙 일부개정령(안)을 개정한 이후 꾸준히 교통사고안전기준을 강화해 왔다.

이러한 노력이 교통사고 부상자수를 줄이는데 효과가 있었는지 알아보자.

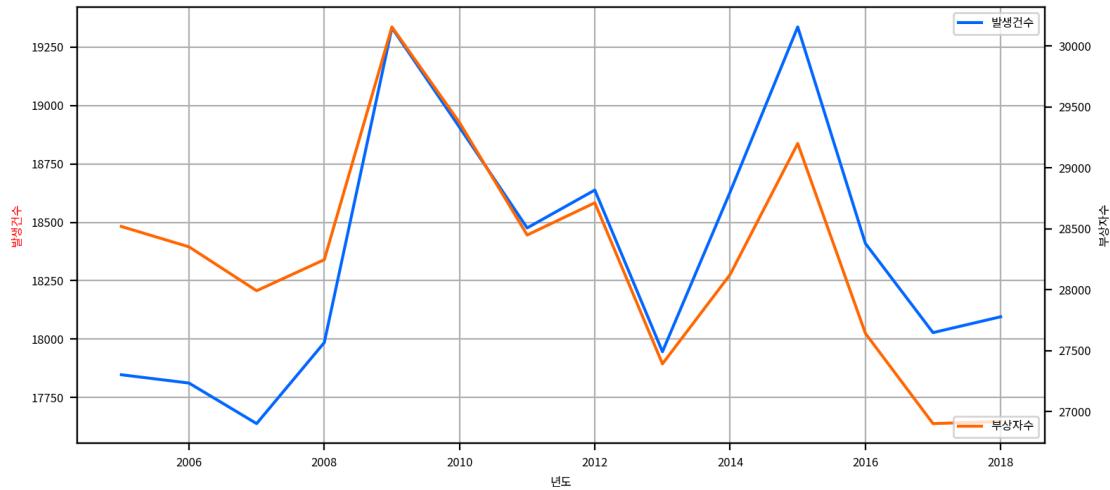
```
# 1) 그래프 초기화  
width_px    = 1600                      # 그래프 가로 크기  
height_px   = 720                       # 그래프 세로 크기  
rows         = 1                          # 그래프 행 수  
cols         = 1                          # 그래프 열 수  
figsize     = (width_px / my_dpi, height_px / my_dpi)  
fig, ax1 = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)  
  
# ax1에 겹쳐지는 쌍둥이 서브플롯을 생성  
ax2 = ax1.twinx()  
  
# 2) LinePlot 그리기  
sb.lineplot(data=df, x=df.index, y='발생건수', color='#0066ff',  
            ax=ax1, label='발생건수')  
sb.lineplot(data=df, x=df.index, y='부상자수', color='ffd600',  
            ax=ax2, label='부상자수')
```

```

# 3) 그래프 꾸미기
ax1.set_xlabel('년도')
ax1.set_ylabel('발생건수', color='ff0000')
ax1.grid(True)           # 그래프가 겹쳐지므로 격자는 하나만 표시해도 됨
ax1.legend(loc='upper right')
ax2.legend(loc='lower right')

# 4) 출력
plt.tight_layout()      # 여백 제거
plt.show()               # 그래프 화면 출력
plt.close()              # 그래프 작업 종료

```



png

해석

개정안이 시행되기 전(2008년)에는 교통사고 발생건수 대비 부상자 수의 비율이 더 많았지만 개정안이 시행된 후에는 교통사고 발생건수 대비 부상자수의 비율이 현저히 낮아졌다.



전국 실업률 분포 변화

unemployment_age 데이터는 2000년부터 2022년까지 행정구역(시도)/연령별 실업률을 담고 있다.

이 데이터를 토대로 지역별 실업률 어떻게 변화하고 있는지 확인해 보자.



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
from pandas import melt, pivot_table
import seaborn as sb
import numpy as np
```



[2] 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



[3] 데이터 가져오기

0번째 열과 1번째 열을 복수 인덱스로 지정

```
origin = load_data('unemployment_age')
origin.head()
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/
unemployment_age.xlsx
```

[94m[desc] [0m 2000년부터 2022년까지 행정구역(시도)/연령별 실업률 데이터 (출처:
국가통계포털)]

[91m[!] Cannot read metadata [0m]

	시도 별	연령계 층별	2000	2001	2002	2003	2004	2005	2006	2007	…	2013	2014	2015	2016
0	서울 특별 시	15-29 세	8.1	8.4	8.2	8.8	8.9	8.9	8.8	7.4	…	8.7	10.4	9.3	10.3
1	NaN	30-59 세	3.9	3.6	3.0	3.2	3.4	3.6	3.4	3.2	…	3.0	3.2	3.1	2.9
2	NaN	60세 이상	2.9	1.9	2.3	1.9	2.1	2.1	2.6	1.7	…	2.5	2.8	3.2	2.8
3	부산 광역 시	15-29 세	12.1	10.7	7.6	9.0	9.9	8.8	8.2	8.3	…	8.8	9.0	9.7	9.9
4	NaN	30-59 세	5.5	4.2	2.9	2.9	3.1	3.3	3.2	3.0	…	3.0	3.0	3.1	2.8

5 rows × 25 columns



#02. 데이터 전처리



[1] 시 이름에 대한 결측치 처리

```
df = origin.copy()
df['시도별'] = df['시도별'].ffill()
df.head(10)
```

	시 도 별	연령계 층별	2000	2001	2002	2003	2004	2005	2006	2007	…	2013	2014	2015	2016	20
0	서 울 특 별 시	15-29 세	8.1	8.4	8.2	8.8	8.9	8.9	8.8	7.4	…	8.7	10.4	9.3	10.3	10.
1	서 울 특	30-59 세	3.9	3.6	3.0	3.2	3.4	3.6	3.4	3.2	…	3.0	3.2	3.1	2.9	3.3

	별 시																				
2	서 울 특 별 시	60세 이상	2.9	1.9	2.3	1.9	2.1	2.1	2.6	1.7	...	2.5	2.8	3.2	2.8	3.6					
3	부 산 광 역 시	15-29 세	12.1	10.7	7.6	9.0	9.9	8.8	8.2	8.3	...	8.8	9.0	9.7	9.9	11.1					
4	부 산 광 역 시	30-59 세	5.5	4.2	2.9	2.9	3.1	3.3	3.2	3.0	...	3.0	3.0	3.1	2.8	3.2					
5	부 산 광 역 시	60세 이상	5.3	3.6	2.1	1.5	1.0	2.8	3.1	2.8	...	2.6	2.4	2.6	2.8	3.6					
6	대 구 광 역 시	15-29 세	9.1	9.8	9.2	9.8	8.6	8.7	9.5	8.9	...	9.9	11.5	10.1	12.0	11.1					
7	대 구 광 역 시	30-59 세	3.7	3.5	2.8	3.0	3.2	3.2	2.5	2.4	...	2.1	2.5	2.2	2.5	2.7					
8	대 구 광 역 시	60세 이상	0.7	1.2	1.4	2.2	1.6	2.5	1.6	2.0	...	2.2	2.1	2.5	3.8	3.4					
9	인 천 광 역 시	15-29 세	8.2	8.2	7.9	8.8	8.3	8.3	9.2	8.3	...	9.3	12.1	11.8	11.5	10.					

10 rows × 25 columns



#03. 시각화



[1] 연도에 따른 전국 평균 실업률 변화

(1) 데이터 전처리

데이터 재구조화

```
df2 = melt(df, id_vars=['시도별', '연령계층별'],
            var_name='년도', value_name='실업률')
df2.head(10)
```

	시도별	연령계층별	년도	실업률
0	서울특별시	15-29세	2000	8.1
1	서울특별시	30-59세	2000	3.9
2	서울특별시	60세이상	2000	2.9
3	부산광역시	15-29세	2000	12.1
4	부산광역시	30-59세	2000	5.5
5	부산광역시	60세이상	2000	5.3
6	대구광역시	15-29세	2000	9.1
7	대구광역시	30-59세	2000	3.7
8	대구광역시	60세이상	2000	0.7
9	인천광역시	15-29세	2000	8.2

시/도에 따른 연도별 평균 실업률

```
tdf1 = df2[['시도별', '년도', '실업률']].groupby(['시도별', '년도'],
                                                as_index=False).mean()
tdf1
```

	시도별	년도	실업률
0	강원도	2000	2.766667
1	강원도	2001	2.333333
2	강원도	2002	2.100000
3	강원도	2003	2.566667
4	강원도	2004	2.633333

...
386	충청북도	2018	3.400000
387	충청북도	2019	4.200000
388	충청북도	2020	4.466667
389	충청북도	2021	3.433333
390	충청북도	2022	3.200000

391 rows × 3 columns

전국에 대한 연도별 평균 실업률

```
tdf2 = tdf1[['년도', '실업률']].groupby('년도').mean()
tdf2
```

	실업률
연도	
2000	4.103922
2001	3.762745
2002	3.154902
2003	3.437255
2004	3.582353
2005	3.584314
2006	3.488235
2007	3.343137
2008	3.282353
2009	3.684314
2010	3.945098
2011	3.635294
2012	3.558824
2013	3.600000
2014	4.127451
2015	4.207843
2016	4.476471
2017	4.558824
2018	4.819608

2019	4.849020
2020	4.988235
2021	4.368627
2022	3.666667

(2) 데이터 시각화

```

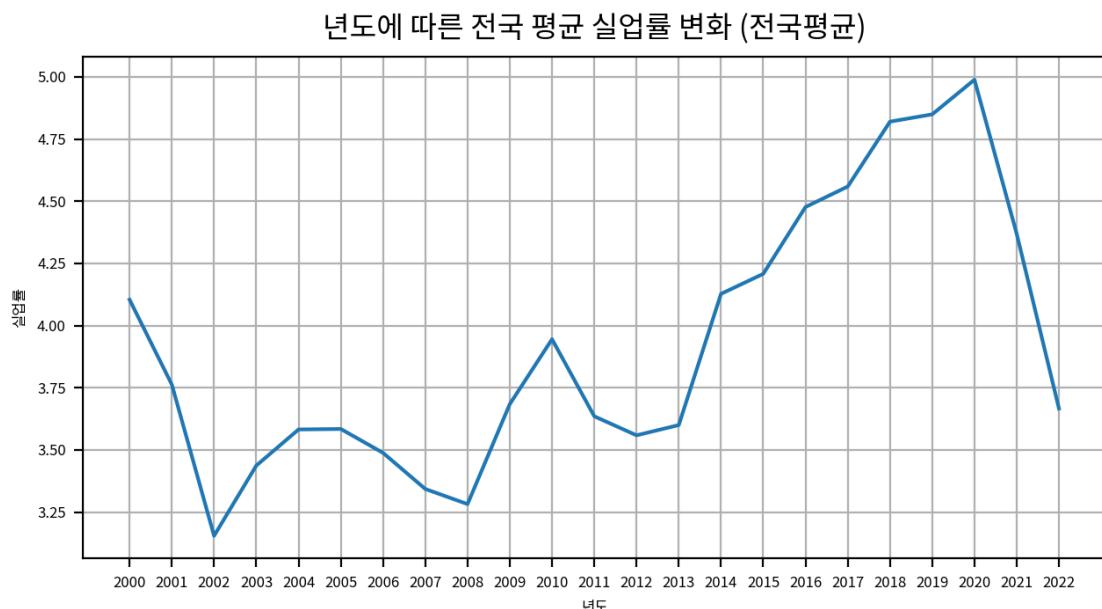
# 1) 그래프 초기화
width_px  = 1280                         # 그래프 가로 크기
height_px = 720                           # 그래프 세로 크기
rows      = 1                               # 그래프 행 수
cols      = 1                               # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax  = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) LinePlot 그리기
sb.lineplot(data=tdf2, x=tdf2.index, y="실업률")

# 3) 그래프 꾸미기
ax.set_title("년도에 따른 전국 평균 실업률 변화 (전국평균)", fontsize=12, pad=8)
ax.grid(True)                                # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                          # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)    # 그래프 화면 출력
plt.show()                                   # 그래프 작업 종료

```



png

- 2000년 이후 증감을 반복하는 모습을 보이지만 전체적으로 증가하는 추세.
- 2020년 이후 급격히 감소함



[2] 연도에 따른 연령대별 전국 평균 실업률 변화

(1) 데이터 전처리

앞에서 만든 데이터 재구조화 결과에 이어서 진행

데이터 그룹별 집계

```
gdf = df2.filter(['년도', '연령계층별', '실업률']).groupby(['년도', '연령계  
층별'], as_index=False).mean()  
gdf
```

	년도	연령계층별	실업률
0	2000	15-29세	7.735294
1	2000	30-59세	3.052941
2	2000	60세이상	1.523529
3	2001	15-29세	7.435294
4	2001	30-59세	2.605882
...
64	2021	30-59세	2.364706
65	2021	60세이상	3.570588
66	2022	15-29세	6.205882
67	2022	30-59세	1.982353
68	2022	60세이상	2.811765

69 rows × 3 columns

```
# 1) 그래프 초기화  
width_px  = 1280                      # 그래프 가로 크기  
height_px = 720                        # 그래프 세로 크기  
rows     = 1                            # 그래프 행 수  
cols     = 1                            # 그래프 열 수  
figsize  = (width_px / my_dpi, height_px / my_dpi)  
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

```

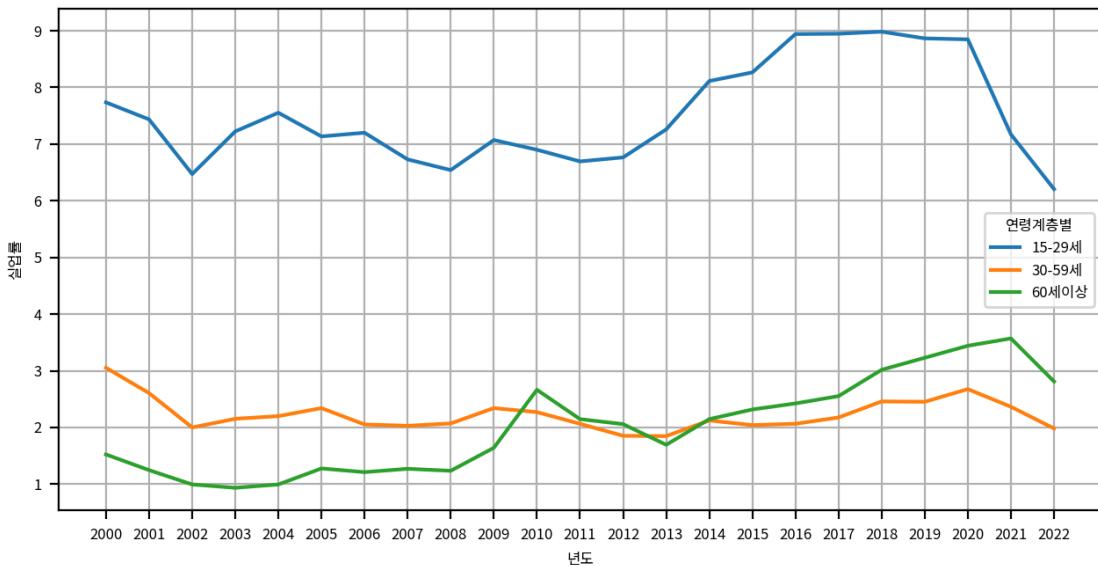
# 2) LinePlot 그리기
sb.lineplot(data=gdf, x='년도', y='실업률', hue='연령계층별')

# 3) 그래프 꾸미기
ax.set_title("년도에 따른 전국 평균 실업률 변화 (연령대별)", fontsize=12, pad=8)
ax.grid(True) # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout() # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```

년도에 따른 전국 평균 실업률 변화 (연령대별)



png

■ #04. 두 그래프를 서브플롯으로 구현

```

# 1) 그래프 초기화
width_px = 2400 # 그래프 가로 크기
height_px = 800 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 2 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)

# ax 객체가 행, 열 수에 따라서 리스트가 된다.
# [fig, [ax1, ax2]]]
fig, (ax1, ax2) = plt.subplots(rows, cols, figsize=figsize,
                               dpi=my_dpi)

```

```

# 2) 그래프 그리기
sb.lineplot(data=tdf2, x=tdf2.index, y="실업률", ax=ax1)
sb.lineplot(data=gdf, x='년도', y='실업률', hue='연령계층별', ax=ax2)

# 3) 그래프 꾸미기
# 전체 제목
fig.suptitle('년도에 따른 전국 평균 실업률 변화', fontsize=14,
             color="#006600")

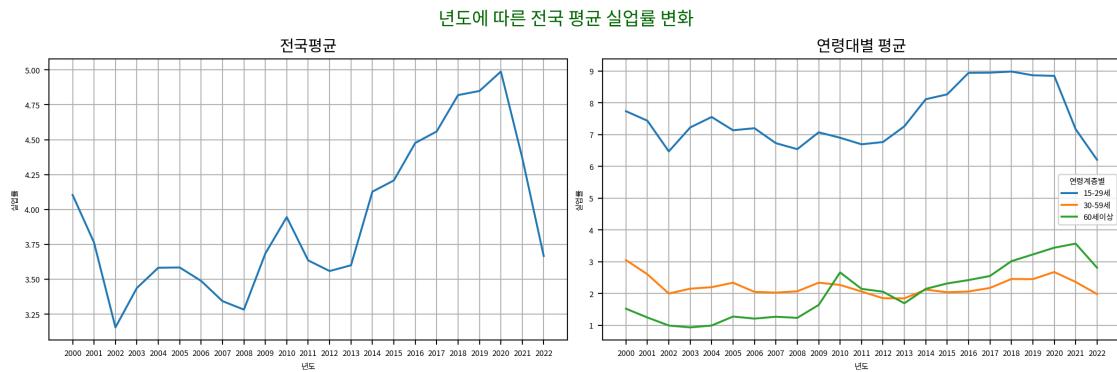
# 각 그래프 간의 가로(wspace), 세로(hspace) 간격 지정
fig.subplots_adjust(wspace=0.2, hspace=0.2)

ax1.title.set_text("전국평균")
ax1.title.set_fontsize(12)
ax1.grid()

ax2.title.set_text("연령대별 평균")
ax2.title.set_fontsize(12)
ax2.grid()

# 4) 출력
plt.tight_layout()      # 여백 제거
plt.show()               # 그래프 화면 출력
plt.close()              # 그래프 작업 종료

```



png



[LAB-06] 2. 데이터 분포 시각화 (1) - Boxplot, KDE



데이터 분포 시각화 종류

그래프 유형	seaborn 함수	설명 / 목적
박스플롯	boxplot()	사분위수 기반 분포 요약
KDE 곡선	kdeplot()	연속형 분포의 밀도
히스토그램	histplot()	연속형 변수의 분포 확인
히스토그램 + KDE	displot(kind="hist")/ histplot(kde=True)	분포 + 패턴
바이올린 플롯	violinplot()	분포 + 중앙값 + 퍼짐
스트립플롯	stripplot()	분포의 개별 관측값
스웜플롯	swarmplot()	겹치지 않는 스트립(마커) 분포
히트맵	heatmap()	복수 변수의 빈도/상관 분석



#01. 준비작업



1. 라이브러리 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```



2. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
font_path = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(font_path) # 폰트의 글꼴을 시스템에 등록
font_prop = fm.FontProperties(fname=font_path) # 폰트의 속성을 읽어옴
```

```

font_name = font_prop.get_name()          # 읽어온 속성에서 폰트의 이름
    만 추출
plt.rcParams['font.family'] = font_name   # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 10             # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지

```



3. 데이터 가져오기

```

origin = load_data("employee_data_40")
origin.head()

```

[94m[**data**] [0m https://data.hossam.kr/data/lab06/
employee_data_40.xlsx
[94m[**desc**] [0m 어느 기업의 직원 40명을 대상으로 성별과 결혼상태, 나이, 최종학력,
월수입을 조사한 가상의 데이터(인덱스, 메타데이터 없음)
[91m[!] Cannot read metadata [0m

	성별	결혼상태	나이	최종학력	월수입
0	남자	기혼	21	대학교	60
1	남자	기혼	22	대학원	100
2	남자	기혼	33	대학교	200
3	여자	미혼	33	대학교	120
4	남자	미혼	28	대학교	70



#02. Boxplot



1) 연속형 데이터의 분포를 사분위수 기반으로 확인

list, ndarray, Series 등 모든 연속형 객체를 data 파라미터에 지정한다.

orient로 방향을 설정할 수 있다.

- v : 세로 (기본값)
- h : 가로

```

# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px = 800                      # 그래프 가로 크기
height_px = 350                       # 그래프 세로 크기

```

```

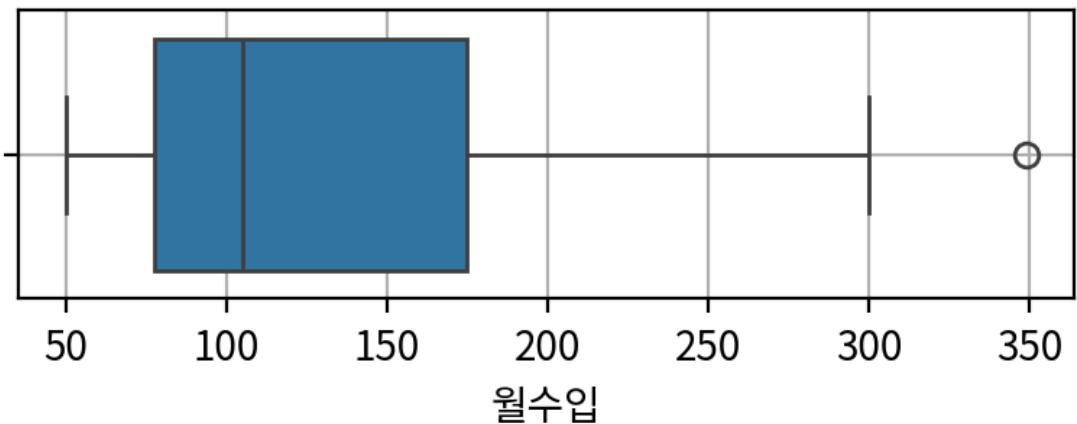
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.boxplot(data=origin['월수입'], orient="h")

# 3) 그래프 꾸미기
ax.grid(True) # 격자 표시

# 4) 출력
plt.tight_layout() # 여백 제거
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png



2. 데이터프레임을 통한 상자그림

data 파라미터에 데이터 프레임을 설정하고 y 파라미터에 표시하고자 하는 변수 이름을 문자열로 설정한다.

x 파라미터에 설정할 경우 가로 상자그림으로 표시된다.

```

# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px = 800 # 그래프 가로 크기
height_px = 350 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

```

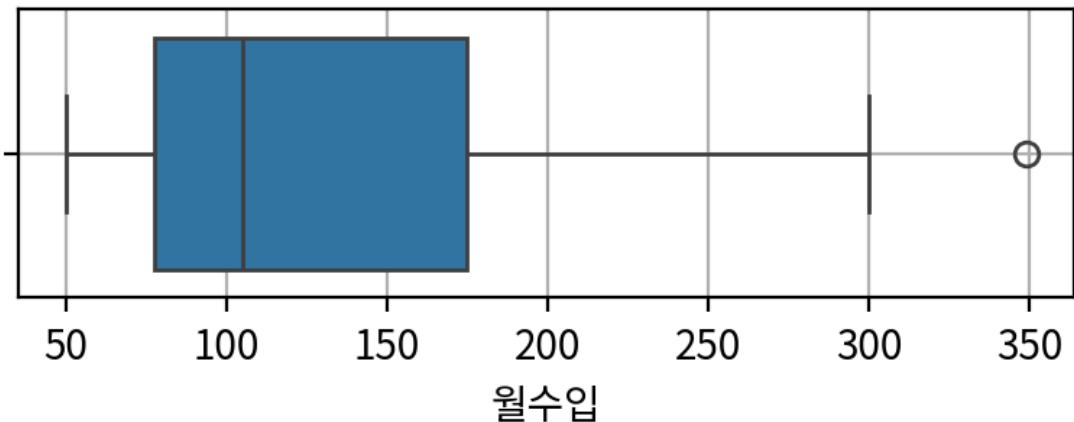
```

# 2) 그래프 그리기
sb.boxplot(data=origin, x='월수입')

# 3) 그래프 꾸미기
ax.grid(True) # 격자 표시

# 4) 출력
plt.tight_layout() # 여백 제거
#plt.savefig("myplot.png", dpi=my_dpi) # 생략 가능
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png



3) 복수 변수에 대한 처리

표시하고자 하는 변수를 필터링하여 data 파라미터에 설정한다.

```

# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px = 800 # 그래프 가로 크기
height_px = 500 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

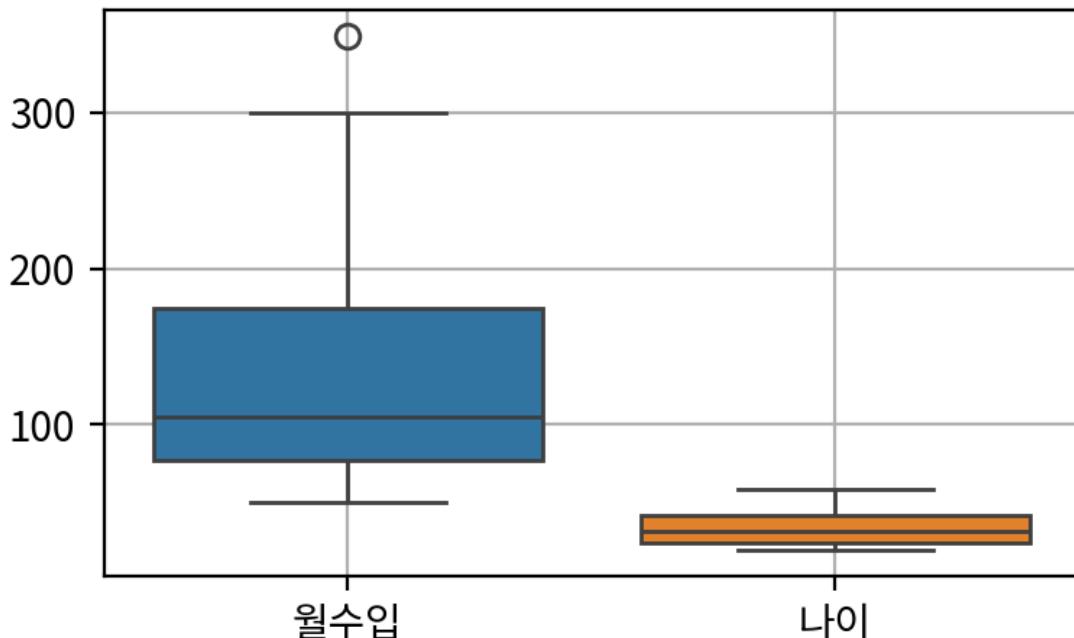
# 2) 그래프 그리기
sb.boxplot(data=origin[['월수입', '나이']])

# 3) 그래프 꾸미기
ax.grid(True) # 격자 표시

# 4) 출력

```

```
plt.tight_layout()          # 여백 제거  
plt.show()                # 그래프 화면 출력  
plt.close()               # 그래프 작업 종료
```



png

📘 #03. KDE(커널 밀도 추정) Plot

- 연속형 데이터의 분포를 부드러운 곡선 형태로 추정하는 비모수적 방법.
- 데이터의 전체적인 분포 모양, 봉우리(peak), 꼬리(tail) 등을 확인하는 데 유용함.



비모수적 방법이란?

- “데이터가 정규분포일 것이다” 같은 가정이 없음
- 대신 데이터 자체를 기반으로 분포의 모양을 추정함
- 필요한 것은 파라미터(평균·분산 등)보다 데이터의 구조와 패턴



1. 연속성 데이터 설정

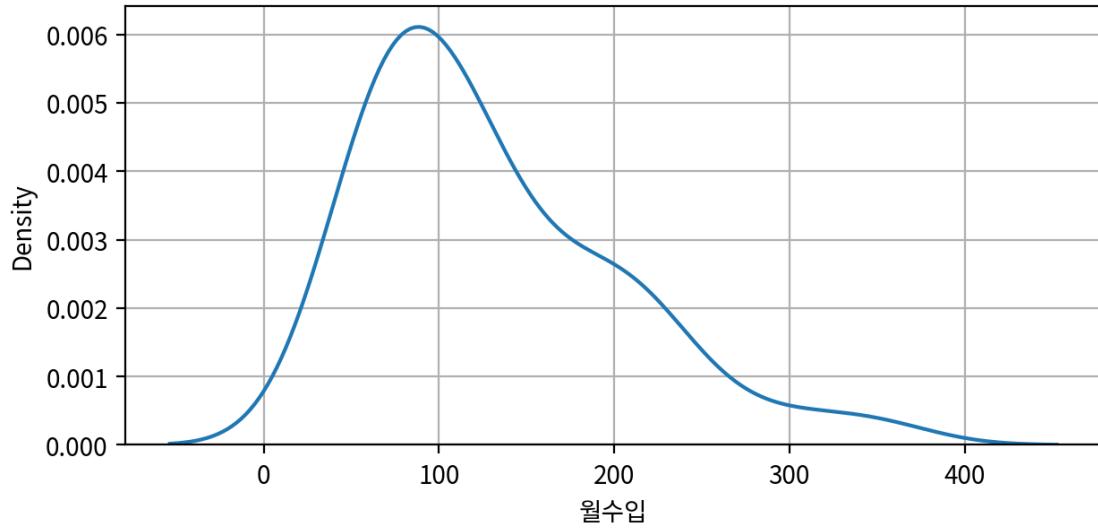
```
# 1) 그래프 초기화  
width_px  = 1280  
height_px = 640  
figsize = (width_px / my_dpi, height_px / my_dpi)  
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)
```

```

# 2) 그래프 그리기
sb.kdeplot(data=origin['월수입'])
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()

```



png

2. 데이터 프레임 자체를 적용하기

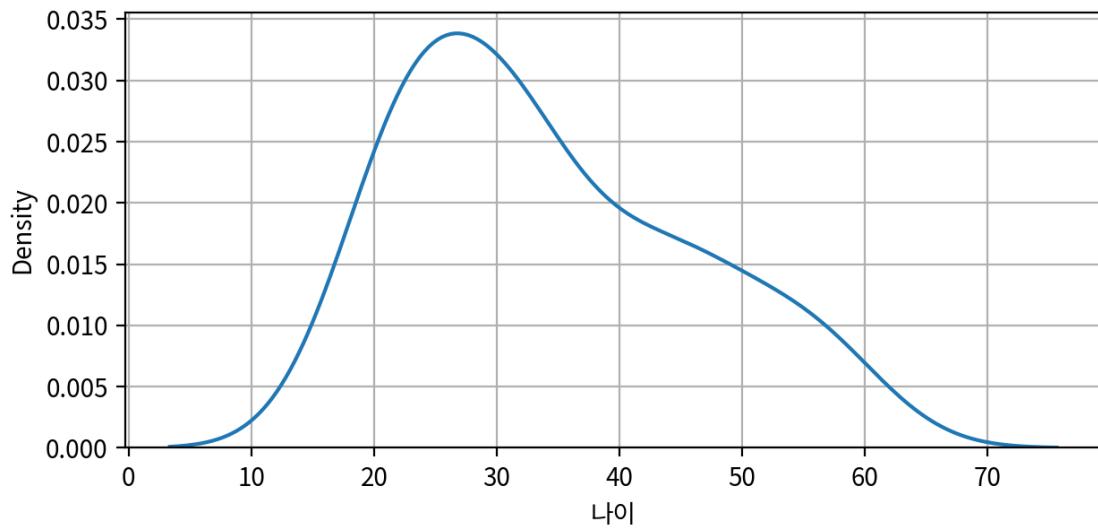
```

# 1) 그래프 초기화
width_px    = 1280
height_px   = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, x='나이')
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()

```



png



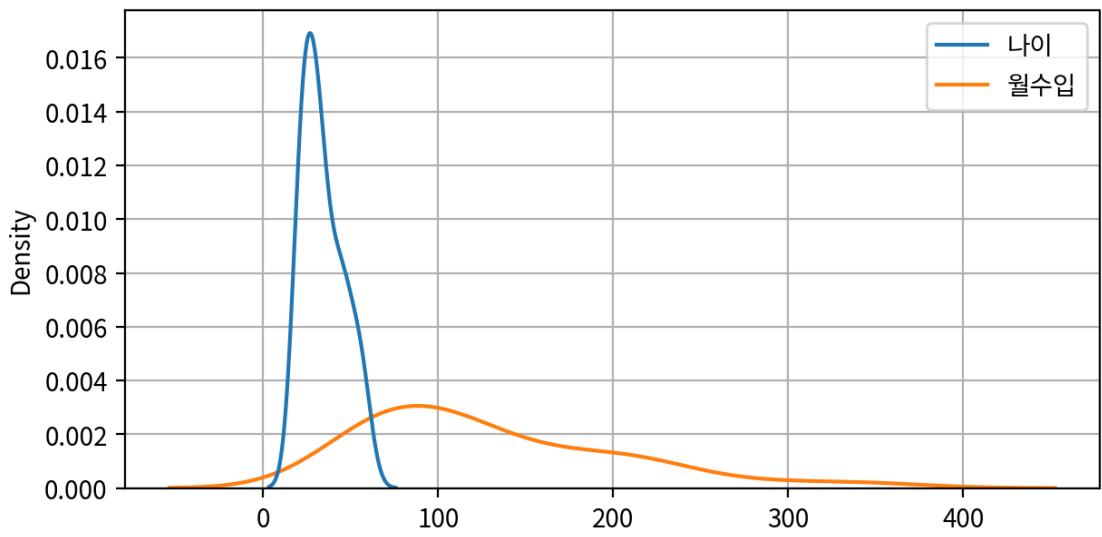
3. 다중 분포

데이터프레임을 data 파라미터에 적용하면서 x나 y파라미터를 지정하지 않으면 모든 연속형 변수에 대한 커널 밀도 곡선이 표현된다.

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin)
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



png



4. 색상 채우기

`fill=True` 파라미터를 설정하면 곡선 내부에 색상이 표시된다.

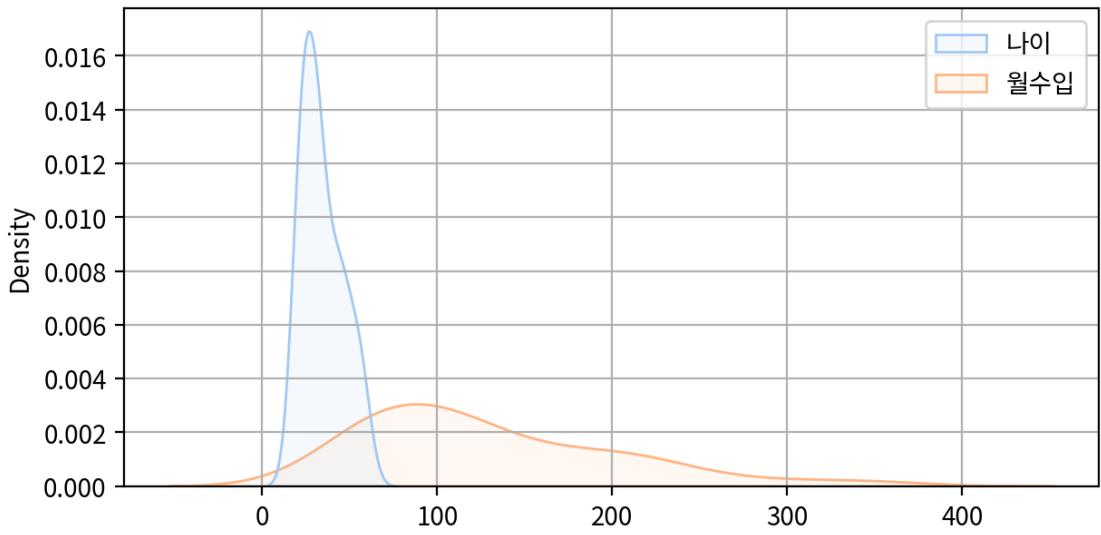
이 때, `alpha` 파라미터를 0~1사이의 값으로 설정하여 면의 투명도를 조절할 수 있다.

0=투명, 1=불투명

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, fill=True, alpha=0.1, palette="pastel")
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



png



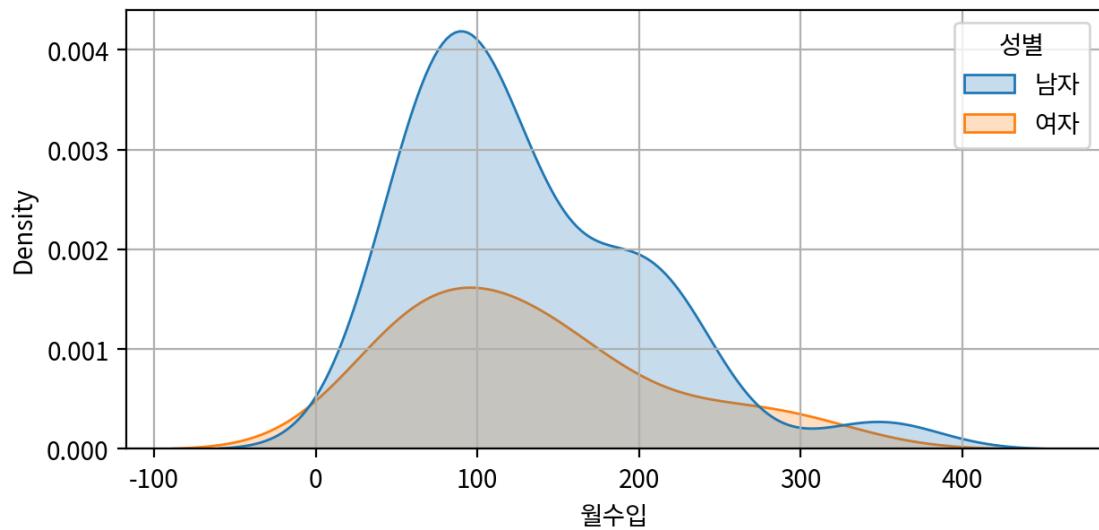
5. 범주에 따른 구분

`hue` 파라미터에 명목형 변수의 이름을 지정하면 범주에 따라 그래프를 분기한다.

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, x='월수입', hue='성별', fill=True)
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



png

📚 #04. Histogram

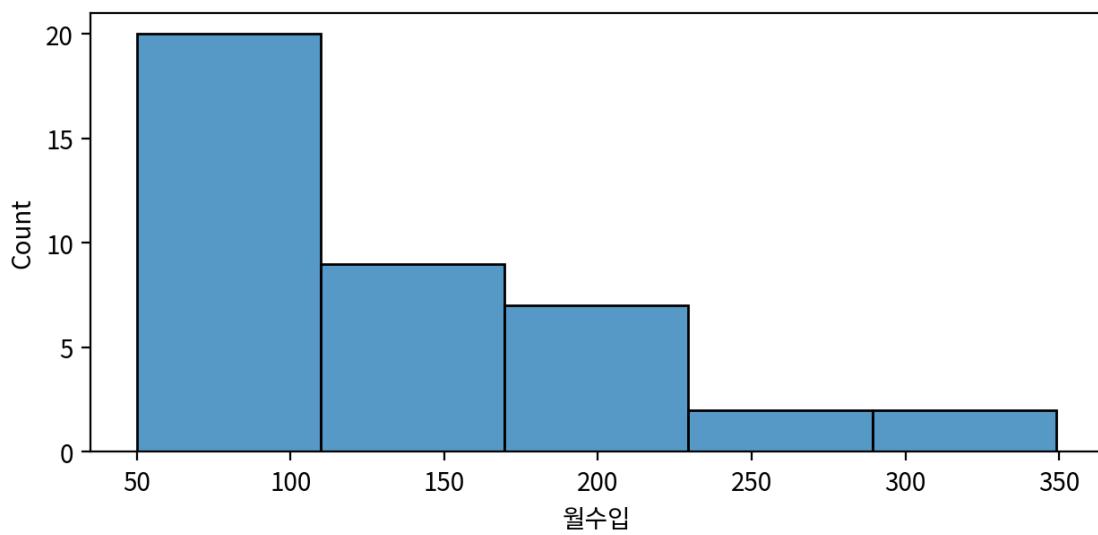
도수 분포표를 시각화 한 그래프

1. 구간 수 지정하기

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin['월수입'], bins=5)
# ax.grid(True) ← 비추

plt.tight_layout()
plt.show()
plt.close()
```



png



2. 구간을 표시하기

```
hist, bins = np.histogram(origin['월수입'], bins=5)
print(hist)
print(bins)
```

```
[20  9  7  2  2]
[ 50.  109.8 169.6 229.4 289.2 349.]
```

```
bins = bins.round().astype("int")
bins
```

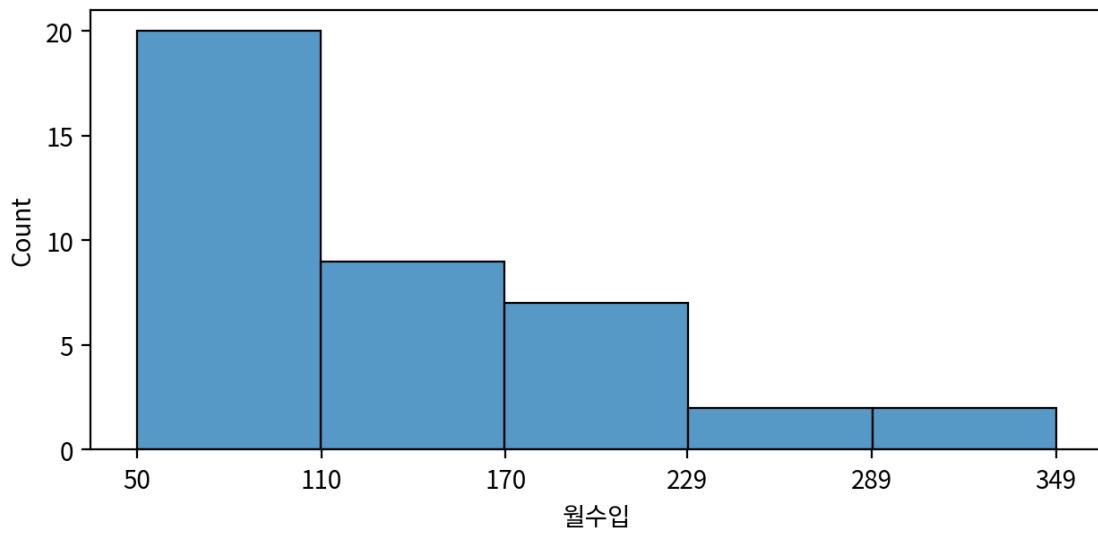
```
array([ 50, 110, 170, 229, 289, 349])
```

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin['월수입'], bins=5, edgecolor="#000000",
            linewidth=0.8)
ax.set_xticks(bins, bins)
# ax.grid(True) ← 비추

plt.tight_layout()
```

```
plt.show()  
plt.close()
```



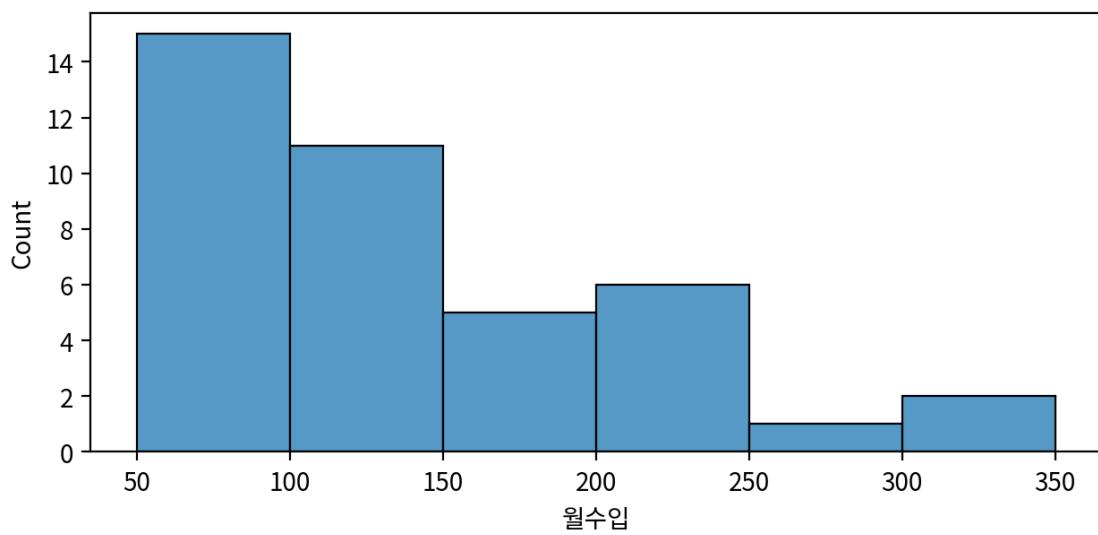
png



3. 구간을 직접 지정하기

bins 파라미터에 구간을 의미하는 리스트를 지정한다.

```
width_px  = 1280  
height_px = 640  
figsize = (width_px / my_dpi, height_px / my_dpi)  
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)  
  
sb.histplot(data=origin['월수입'], bins=[50, 100, 150, 200, 250, 300,  
    350],  
            edgecolor="#000000", linewidth=0.8)  
  
plt.tight_layout()  
plt.show()  
plt.close()
```



png

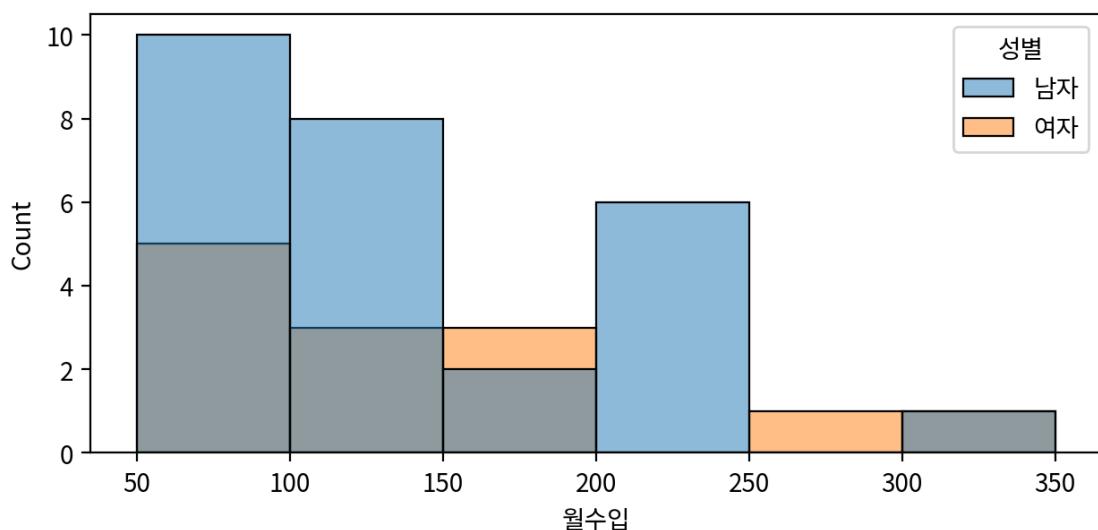


4. 범주에 따른 구분

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin, x='월수입', hue='성별',
            bins=[50, 100, 150, 200, 250, 300, 350],
            edgecolor="#000000", linewidth=0.8)

plt.tight_layout()
plt.show()
plt.close()
```



png

5. 히스토그램의 계급의 수와 간격의 크기

우리에게 주어지는 데이터는 종류도 다양하며 관측치의 개수도 다양하다.

또한 그 데이터의 최솟값과 최댓값 또한 다양하다. 따라서 데이터가 주어졌을 때 계급의 개수와 계급의 간격을 설정하는 것에 어려움이 있을 수 있다.

(1) 계급의 수를 구하는 방법

스터지스의 공식(Sturges' formula)

히스토그램의 계급의 수를 결정하는 공식

$$\text{계급구간수} = 1 + 3.3\log n$$

n = 관측치의 수

관측치의 수에 따른 계급구간 수 표 활용

통계학에서 일반적으로 활용하는 표

관측치의 수	계급구간의 수
50 미만	5 ~ 7
50 ~ 200	7 ~ 9
200 ~ 500	9 ~ 10
500 ~ 1,000	10 ~ 11
1,000 ~ 5,000	11 ~ 13
5,000 ~ 50,000	13 ~ 17
50,000 초과	17 ~ 20

(2) 간격의 크기를 구하는 방법

$$\text{계급구간의간격} = \frac{\text{관측치최대값} - \text{관측치최소값}}{\text{계급구간의수}}$$

여기서 중요한 지점은 계급구간의 첫 시작지점을 어떻게 잡아야 하는지이다.

이 때 필수적으로 알아야 하는 점은 첫 계급구간은 반드시 최소 관측치를 포함해야 한다는 점이다.

(3) 히스토그램의 모습

대칭성

정중앙 부분에 선을 수직으로 긋고 절반으로 접었을 때 일치하는 그래프

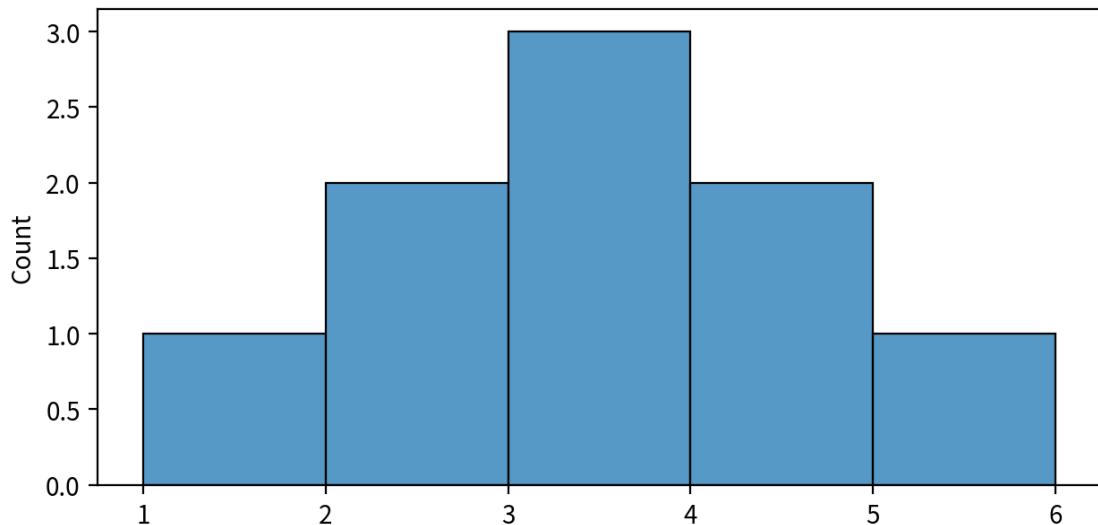
특히 정삼각형에 가까운 형태를 종모양이라고 하는데 이 그래프는 정규분포와 밀접한 관련이 있기 때문에 굉장히 중요한 히스토그램의 모습이라고 할 수 있다.

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 2, 2, 3, 3, 3, 4, 4, 5]

width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

plt.tight_layout()
plt.show()
plt.close()
```



png

비대칭성

그래프의 모양을 보면 점점 작아지는 히스토그램의 모양을 볼 수 있는데, 점점 작아지는 방향을 꼬리라고 칭하며 꼬리가 양의 방향으로 향해 있다면 양의 비대칭이며 꼬리가 음의 방향을 향해 있다면 음의 비대칭이라고 할 수 있다. 밑의 그림은 양과 음의 비대칭 히스토그램의 모습이다.

양의 비대칭은 평균이 중앙값보다 작으며 음의 비대칭은 평균이 중앙값보다 크다.

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 5]

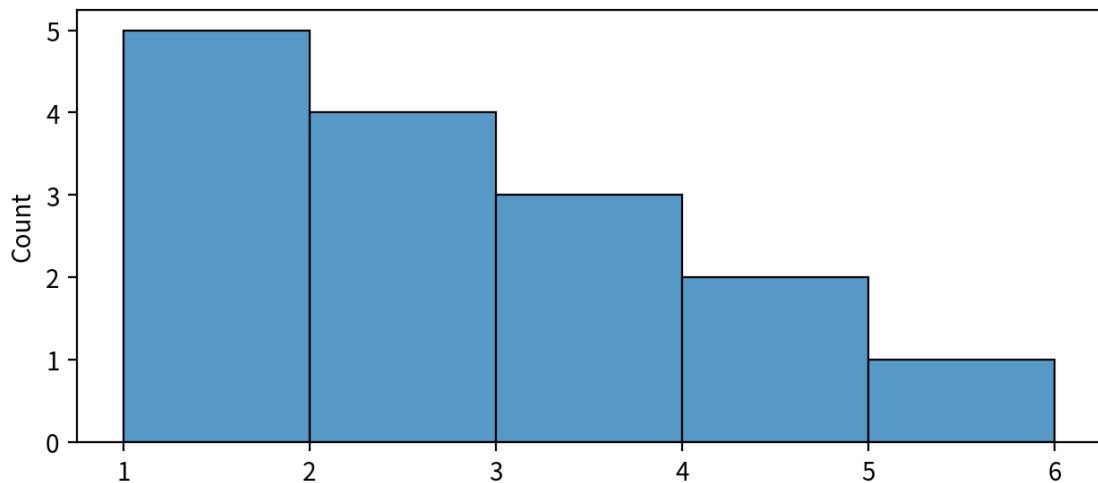
width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

ax.set_title("양의 비대칭", fontsize=15)

plt.tight_layout()
plt.show()
plt.close()
```

양의 비대칭



png

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

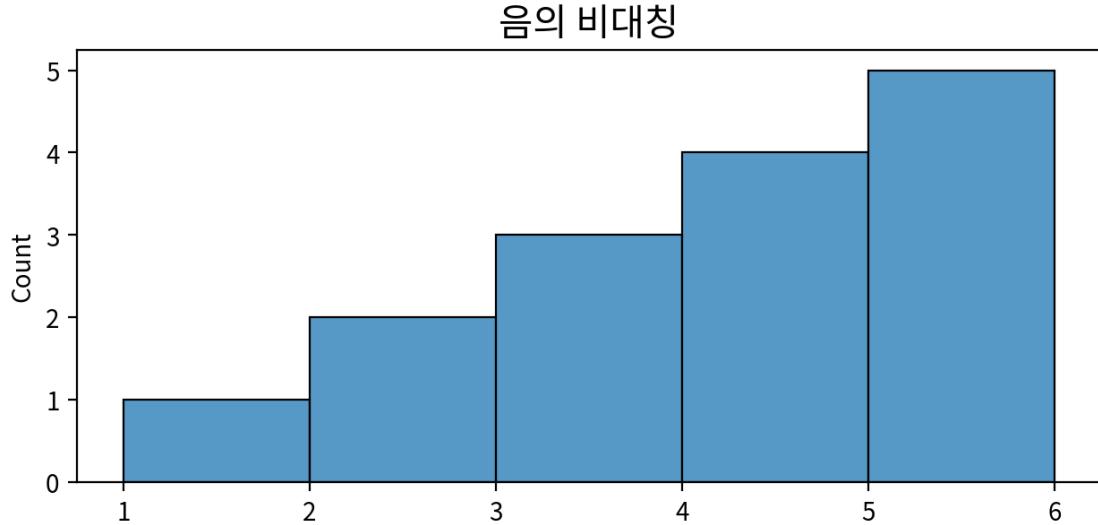
sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)
```

```

ax.set_title("음의 비대칭", fontsize=15)

plt.tight_layout()
plt.show()
plt.close()

```



png

봉우리 계급 구간의 수

히스토그램에서 가장 높은 도수를 나타내고 있는 수치를 최빈값(mode)이라고 부른다.

최빈계급이란 최대의 관측치 수를 가진 계급이다.

만일 최빈계급이 하나일 경우에는 단봉을 가진 히스토그램 이라고 불리며 최빈계급이 두 개일 경우에는 양봉을 가진 히스토그램이라고 불린다.

```

mybins = [1, 2, 3, 4, 5, 6]
data = [1, 1, 1, 2, 2, 3, 4, 4, 5, 5]

width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

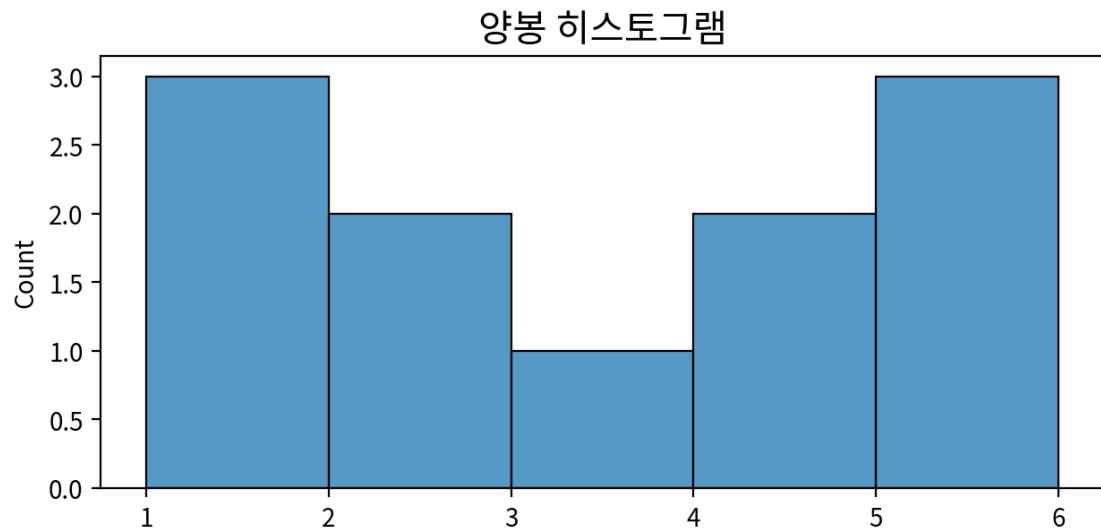
sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

ax.set_title("양봉 히스토그램", fontsize=15)

plt.tight_layout()

```

```
plt.show()  
plt.close()
```



png



[LAB-06] 3. 데이터 시각화 핵심 포인트



#01. 준비작업



1. 라이브러리 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
```



2. 데이터 불러오기

```
origin = load_data('traffic_acc')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab04/traffic_acc.xlsx
[94m[desc] [0m 2005년 1월부터 2018년 12월까지 월별 교통사고의 발생건수, 부상자
수, 사망자수 데이터(인덱스/메타데이터 없음, 출처: 공공데이터포털)
[91m[!] Cannot read metadata [0m
```

	년도	월	발생건수	사망자수	부상자수
0	2005	1	15494	504	25413
1	2005	2	13244	431	21635
2	2005	3	16580	477	25550
3	2005	4	17817	507	28131
4	2005	5	19085	571	29808
...
163	2018	8	18335	357	27749
164	2018	9	18371	348	27751
165	2018	10	19738	373	28836
166	2018	11	19029	298	28000
167	2018	12	18010	323	26463

168 rows × 5 columns

```
df = origin.drop('월', axis=1).groupby('년도').sum()
df
```

	발생건수	사망자수	부상자수
년도			
2005	214171	6376	342233
2006	213745	6327	340229
2007	211662	6166	335906
2008	215822	5870	338962
2009	231990	5838	361875
2010	226878	5505	352458
2011	221711	5229	341391
2012	223656	5392	344565
2013	215354	5092	328711
2014	223552	4762	337497
2015	232035	4621	350400
2016	220917	4292	331720
2017	216335	4185	322829
2018	217148	3781	323037



3. 그래프 초기화

```
my_dpi = 200
fpath = "./NotoSansKR-Regular.ttf"
fm.fontManager.addfont(fpath)
fprop = fm.FontProperties(fname=fpath)
fname = fprop.get_name()
    출
plt.rcParams['font.family'] = fname
plt.rcParams['font.size'] = 6
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
# 이미지 선명도(100~300)
# 한글을 지원하는 폰트 파일의 경로
# 폰트의 글꼴을 시스템에 등록함
# 폰트의 속성을 읽어옴
# 읽어온 속성에서 폰트의 이름만 추출
# 그래프에 한글 폰트 적용
# 기본 폰트 크기
# 그래프에 마이너스 깨짐 방지
```



#02. 시각화 기본 코드

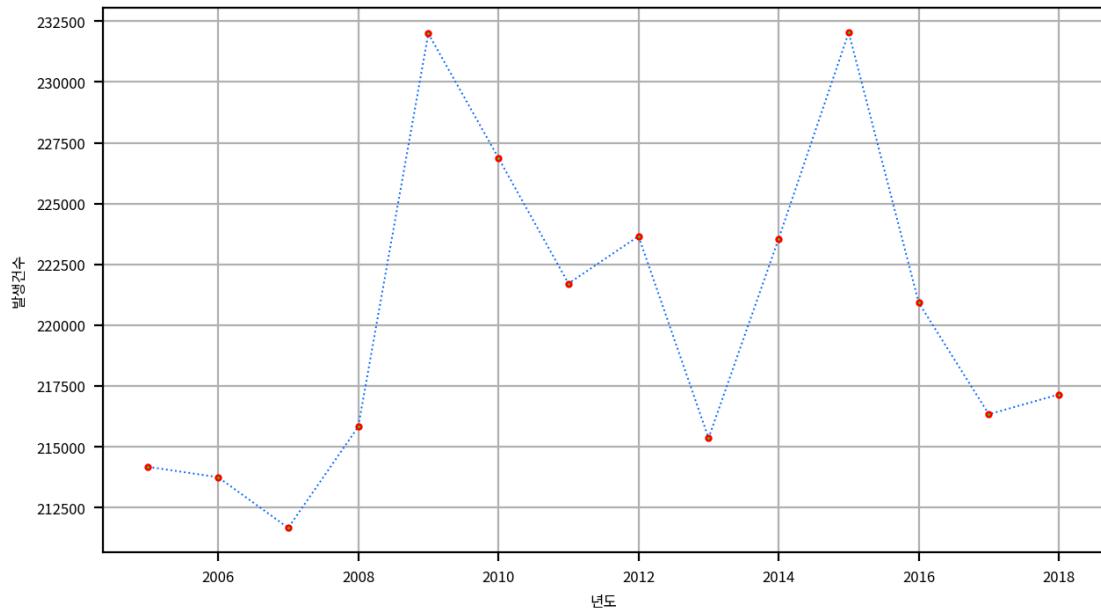
```
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax  = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2-1) BoxPlot 그리기
# sb.boxplot(data=df['발생건수'], orient="v")
# sb.boxplot(data=df['부상자수'], orient="v")
# sb.boxplot(data=df[['발생건수', '부상자수']], orient="v")

# 2-2) LinePlot
sb.lineplot(
    #df['발생건수'],
    #x=df.index, y=df['발생건수'],
    data=df, x=df.index, y='발생건수',
    color="#0066ff", linewidth=0.7, linestyle=':',
    marker="o", markersize=2,
    markerfacecolor="#00ff00",
    markeredgecolor="#ff0000", markeredgewidth=1)

# 3) 그래프 꾸미기
ax.grid(True)                         # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                    # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료
```



png

#03. 다중 그래프

```

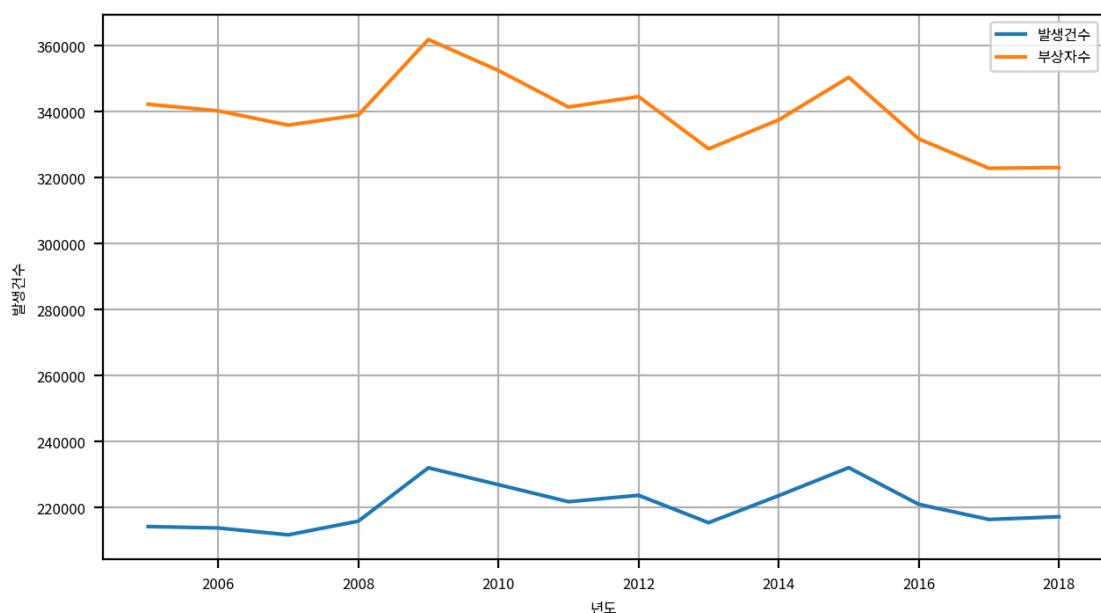
# 1) 그래프 초기화
width_px    = 1280          # 그래프 가로 크기
height_px   = 720           # 그래프 세로 크기
rows         = 1              # 그래프 행 수
cols         = 1              # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)
fig, ax      = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 선 그래프
sb.lineplot(data=df, x=df.index, y='발생건수', label='발생건수')
sb.lineplot(data=df, x=df.index, y='부상자수', label='부상자수')

# 3) 그래프 꾸미기
ax.grid(True)                # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()            # 여백 제거
plt.show()                    # 그래프 화면 출력
plt.close()                   # 그래프 작업 종료

```



png

#04. hue 파라미터 이해하기

1. 데이터 전처리 (melt)

melt를 적용할 경우 index 해제

```
df1 = df.reset_index()
df1
```

	년도	발생건수	사망자수	부상자수
0	2005	214171	6376	342233
1	2006	213745	6327	340229
2	2007	211662	6166	335906
3	2008	215822	5870	338962
4	2009	231990	5838	361875
5	2010	226878	5505	352458
6	2011	221711	5229	341391
7	2012	223656	5392	344565
8	2013	215354	5092	328711
9	2014	223552	4762	337497

10	2015	232035	4621	350400
11	2016	220917	4292	331720
12	2017	216335	4185	322829
13	2018	217148	3781	323037

```
df2 = df1.melt(id_vars='년도', value_vars=['발생건수', '사망자수', '부상자수'],
                 var_name='구분')
df2
```

	년도	구분	value
0	2005	발생건수	214171
1	2006	발생건수	213745
2	2007	발생건수	211662
3	2008	발생건수	215822
4	2009	발생건수	231990
5	2010	발생건수	226878
6	2011	발생건수	221711
7	2012	발생건수	223656
8	2013	발생건수	215354
9	2014	발생건수	223552
10	2015	발생건수	232035
11	2016	발생건수	220917
12	2017	발생건수	216335
13	2018	발생건수	217148
14	2005	사망자수	6376
15	2006	사망자수	6327
16	2007	사망자수	6166
17	2008	사망자수	5870
18	2009	사망자수	5838
19	2010	사망자수	5505
20	2011	사망자수	5229
21	2012	사망자수	5392
22	2013	사망자수	5092

23	2014	사망자수	4762
24	2015	사망자수	4621
25	2016	사망자수	4292
26	2017	사망자수	4185
27	2018	사망자수	3781
28	2005	부상자수	342233
29	2006	부상자수	340229
30	2007	부상자수	335906
31	2008	부상자수	338962
32	2009	부상자수	361875
33	2010	부상자수	352458
34	2011	부상자수	341391
35	2012	부상자수	344565
36	2013	부상자수	328711
37	2014	부상자수	337497
38	2015	부상자수	350400
39	2016	부상자수	331720
40	2017	부상자수	322829
41	2018	부상자수	323037



2. hue 파라미터를 사용한 시각화

```

# 1) 그래프 초기화
width_px    = 1280                         # 그래프 가로 크기
height_px   = 720                           # 그래프 세로 크기
rows         = 1                             # 그래프 행 수
cols         = 1                             # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)
fig, ax      = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

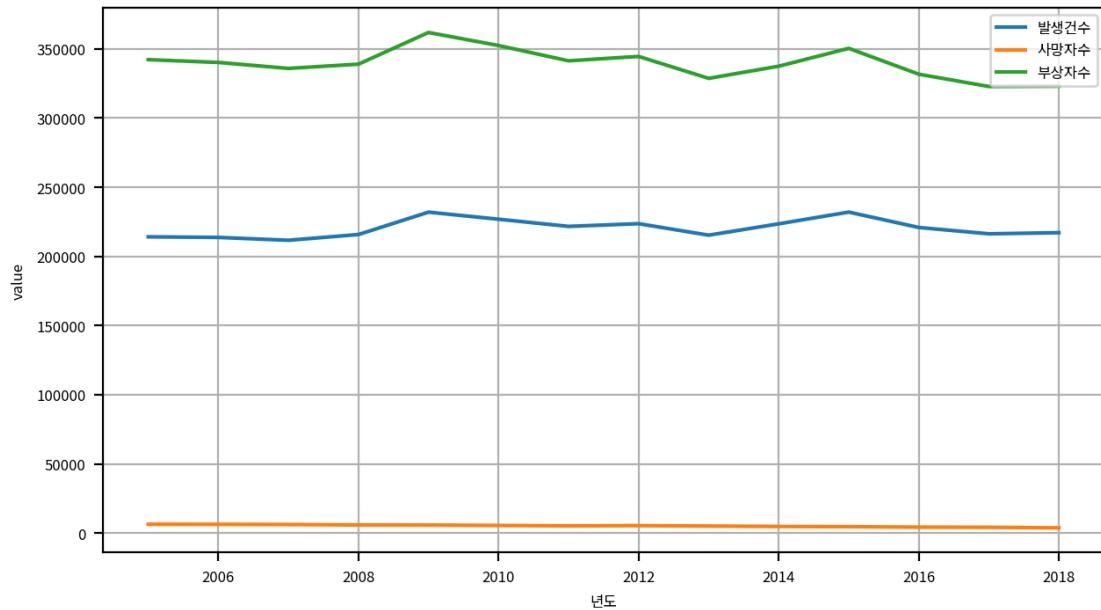
# 2) LinePlot 그리기
sb.lineplot(data=df2, x="년도", y="value", hue="구분")

# 3) 그래프 꾸미기
ax.grid(True)                                # 배경 격자 표시/숨김
ax.legend(loc="upper right")

# 4) 출력

```

```
plt.tight_layout()      # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show()              # 그래프 화면 출력
plt.close()             # 그래프 작업 종료
```



png



[LAB 06] 4. 데이터 분포 시각화 (2)



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
```



2. 데이터 불러오기

```
origin = load_data('employee_data_40')
origin.head()
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/
employee_data_40.xlsx
[94m[desc] [0m 어느 기업의 직원 40명을 대상으로 성별과 결혼상태, 나이, 최종학력,
월수입을 조사한 가상의 데이터(인덱스, 메타데이터 없음)
[91m[!] Cannot read metadata [0m
```

	성별	결혼상태	나이	최종학력	월수입
0	남자	기혼	21	대학교	60
1	남자	기혼	22	대학원	100
2	남자	기혼	33	대학교	200
3	여자	미혼	33	대학교	120
4	남자	미혼	28	대학교	70



3. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
```

```
fname = fprop.get_name()                      # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname            # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6                  # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False     # 그래프에 마이너스 깨짐 방지
```

#02. Histplot + KDE

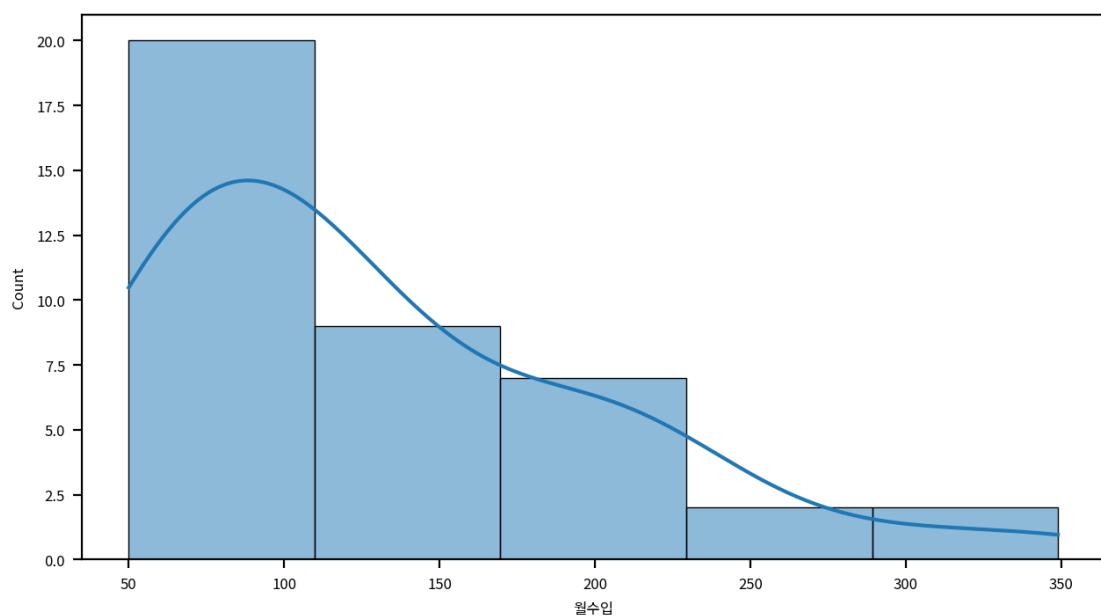
히스토그램에 커널밀도 그래프를 추가한 형태

histplot() 함수에 kde=True 파라미터를 추가한다.

```
# 1) 그래프 초기화
width_px = 1280                                # 그래프 가로 크기
height_px = 720                                 # 그래프 세로 크기
rows = 1                                         # 그래프 행 수
cols = 1                                         # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) Histogram 그리기
sb.histplot(data=origin, x="월수입", bins=5, edgecolor="#000000",
            linewidth=0.5, kde=True)

# 4) 출력
plt.tight_layout()                             # 여백 제거
plt.show()                                     # 그래프 화면 출력
plt.close()                                     # 그래프 작업 종료
```



png

#03. 그 밖의 데이터 분포 시각화 방법



1. 바이올린 플롯

- 데이터의 분포 형태(치우침, 뾰족함, 꼬리 등)를 한눈에 보여주는 그래프
- 박스플롯(boxplot)에 커널 밀도추정(KDE) 그래프를 결합한 형태
- 좌우 대칭의 “바이올린 모양” 폭이 해당 구간의 데이터 밀도를 의미
- 중앙에는 보통 중앙값, 사분위수(IQR) 등을 표시해 박스플롯 정보도 함께 제공
- 여러 그룹·카테고리의 분포 비교에 효과적
- 데이터의 세부적 분포 구조(멀티 모달, 꼬리 부분 등)를 박스플롯보다 더 잘 표현

1) 그래프 초기화

```
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows     = 1                            # 그래프 행 수
cols     = 1                            # 그래프 열 수
figsize  = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

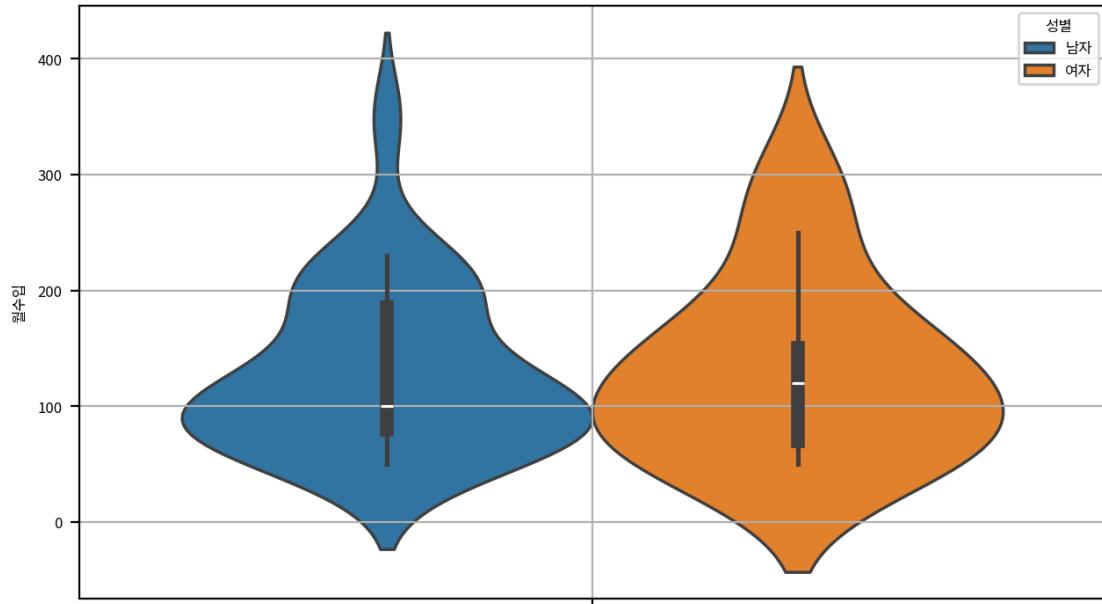
2) Histogram 그리기

```
sb.violinplot(data=origin, y='월수입', hue='성별')
```

3) 그래프 꾸미기

```
ax.grid(True)                           # 배경 격자 표시/숨김
```

```
# 4) 출력
plt.tight_layout()      # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show()                # 그래프 화면 출력
plt.close()                # 그래프 작업 종료
```



png



2. Strip Plot

- 개별 데이터를 점 하나씩 그대로 표시하는 분포 시각화
- x축(또는 y축) 기준으로 값들이 가로·세로로 흩뿌려진 형태
- 작은 표본이나 정확한 관측값 하나하나를 보여줄 때 효과적
- 범주형 그룹 간 원시 데이터 비교가 가능해 boxplot·violinplot보다 세밀한 확인 가능
- 여러 그룹 비교 시, swarmplot과 함께 많이 사용됨 (swarmplot은 겹침을 더 지능적으로 피함)

```
# 1) 그래프 초기화
width_px    = 1280                      # 그래프 가로 크기
height_px   = 720                        # 그래프 세로 크기
rows        = 1                           # 그래프 행 수
cols        = 1                           # 그래프 열 수
figsize     = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

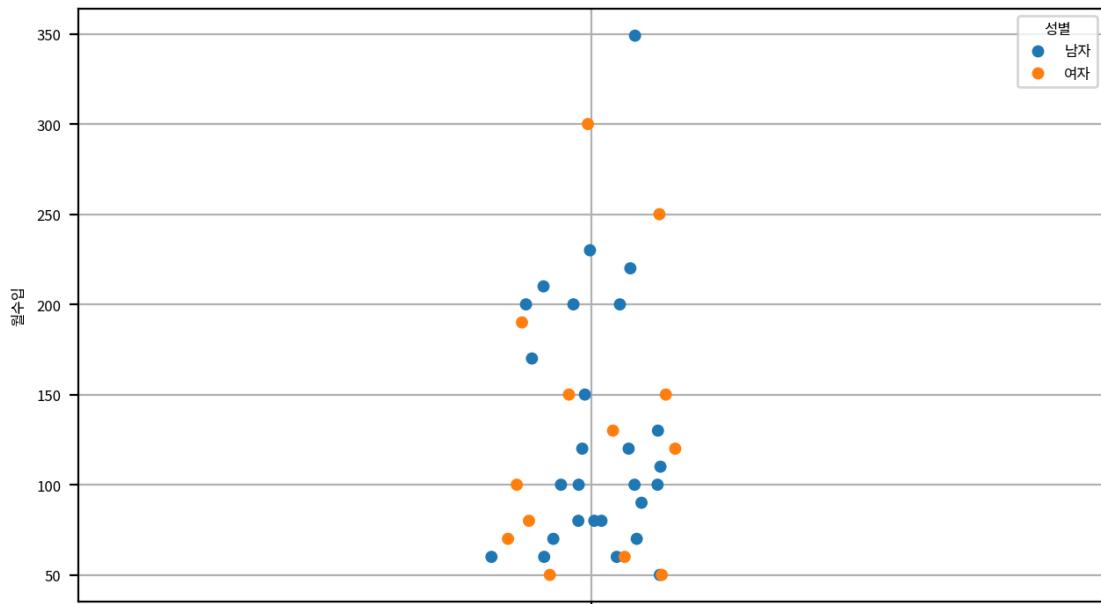
# 2) Histogram 그리기
sb.stripplot(data=origin, y='월수입', hue='성별')
```

```

# 3) 그래프 꾸미기
ax.grid(True)                      # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                 # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show()                          # 그래프 화면 출력
plt.close()                         # 그래프 작업 종료

```



png



3. Swarm Plot

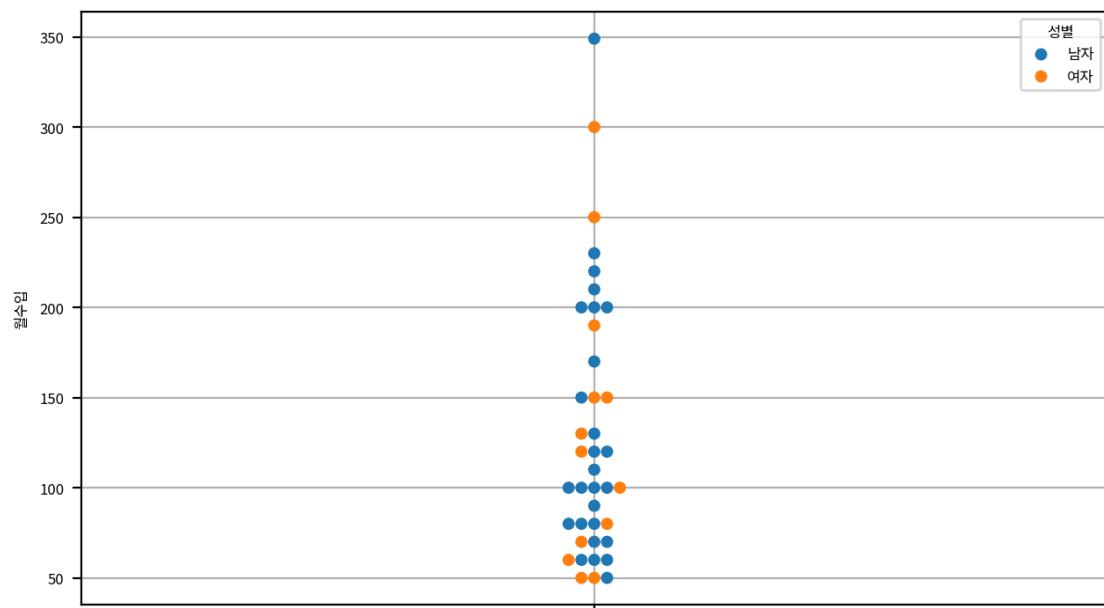
- stripplot처럼 개별 데이터 점을 그대로 표시하는 분포 시각화
- 단, 데이터 점들이 서로 겹치지 않도록 자동으로 배치해주는 알고리즘 사용
 - 이 점을 제외하면 stripplot과 차이가 없다.
- 관측값 하나하나를 보면서도 밀도와 구조를 직관적으로 파악 가능
- 표본 수가 많아도 stripplot보다 가독성이 훨씬 좋음
- 범주형 그룹별 데이터의 실제 분포 형태를 비교할 때 유용

```

# 1) 그래프 초기화
width_px   = 1280                      # 그래프 가로 크기
height_px  = 720                        # 그래프 세로 크기
rows        = 1                           # 그래프 행 수
cols        = 1                           # 그래프 열 수
figsize     = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

```

```
# 2) Histogram 그리기  
sb.swarmplot(data=origin, y='월수입', hue='성별')  
  
# 3) 그래프 꾸미기  
ax.grid(True) # 배경 격자 표시/숨김  
  
# 4) 출력  
plt.tight_layout() # 여백 제거  
plt.savefig("plot.png", dpi=my_dpi * 2)  
plt.show() # 그래프 화면 출력  
plt.close() # 그래프 작업 종료
```



png



[LAB-06] 5. 데이터 분포 시각화 (3)



#01. 준비작업



[1] 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
from pandas import pivot_table
import seaborn as sb
```



[2] 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



[3] 데이터 가져오기

```
origin = load_data('flights')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/flights.xlsx
[94m[desc] [0m 어느 항공사의 년/월별 국제선 탑승객 수(출처: seaborn 내장 데이터)
```

field	description
-----	-----
year	항공 승객 수가 집계된 연도
month	항공 승객 수가 집계된 월
passengers	해당 연도/월의 국제선 항공 승객 수

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
...
139	1960	August	606
140	1960	September	508
141	1960	October	461
142	1960	November	390
143	1960	December	432

144 rows × 3 columns



[4] 데이터 전처리

```
df = origin.copy()
df['month'] = df['month'].map({
    "January": 1, "February": 2, "March": 3, "April": 4,
    "May": 5, "June": 6, "July": 7, "August": 8,
    "September": 9, "October": 10, "November": 11, "December": 12
})

df.head()
```

	year	month	passengers
0	1949	1	112
1	1949	2	118
2	1949	3	132
3	1949	4	129
4	1949	5	121

```
df2 = pivot_table(df, index="year", columns="month",
                  values="passengers")
df2
```

month	1	2	3	4	5	6	7	8	9	10	11	12
year												
1949	112.0	118.0	132.0	129.0	121.0	135.0	148.0	148.0	136.0	119.0	104.0	118.0
1950	115.0	126.0	141.0	135.0	125.0	149.0	170.0	170.0	158.0	133.0	114.0	140.0
1951	145.0	150.0	178.0	163.0	172.0	178.0	199.0	199.0	184.0	162.0	146.0	166.0
1952	171.0	180.0	193.0	181.0	183.0	218.0	230.0	242.0	209.0	191.0	172.0	194.0
1953	196.0	196.0	236.0	235.0	229.0	243.0	264.0	272.0	237.0	211.0	180.0	201.0
1954	204.0	188.0	235.0	227.0	234.0	264.0	302.0	293.0	259.0	229.0	203.0	229.0
1955	242.0	233.0	267.0	269.0	270.0	315.0	364.0	347.0	312.0	274.0	237.0	278.0
1956	284.0	277.0	317.0	313.0	318.0	374.0	413.0	405.0	355.0	306.0	271.0	306.0
1957	315.0	301.0	356.0	348.0	355.0	422.0	465.0	467.0	404.0	347.0	305.0	336.0
1958	340.0	318.0	362.0	348.0	363.0	435.0	491.0	505.0	404.0	359.0	310.0	337.0
1959	360.0	342.0	406.0	396.0	420.0	472.0	548.0	559.0	463.0	407.0	362.0	405.0
1960	417.0	391.0	419.0	461.0	472.0	535.0	622.0	606.0	508.0	461.0	390.0	432.0

#02. Heatmap 시각화

데이터의 패턴, 특히 밀도와 분포를 빠르게 파악할 수 있게 해주는 시각적 도구.

행과 열을 가진 행렬 형태의 데이터를 색상으로 나타내어 각 셀의 값에 따라 색상이 변한다.



[1] 기본 사용 방법

성적표 데이터에 대한 점수 분포 시각화

```
# 1) 그래프 초기화
width_px    = 1000                      # 그래프 가로 크기
height_px   = 1000                      # 그래프 세로 크기
rows        = 1                          # 그래프 행 수
cols        = 1                          # 그래프 열 수
figsize     = (width_px / my_dpi, height_px / my_dpi)
fig, ax     = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

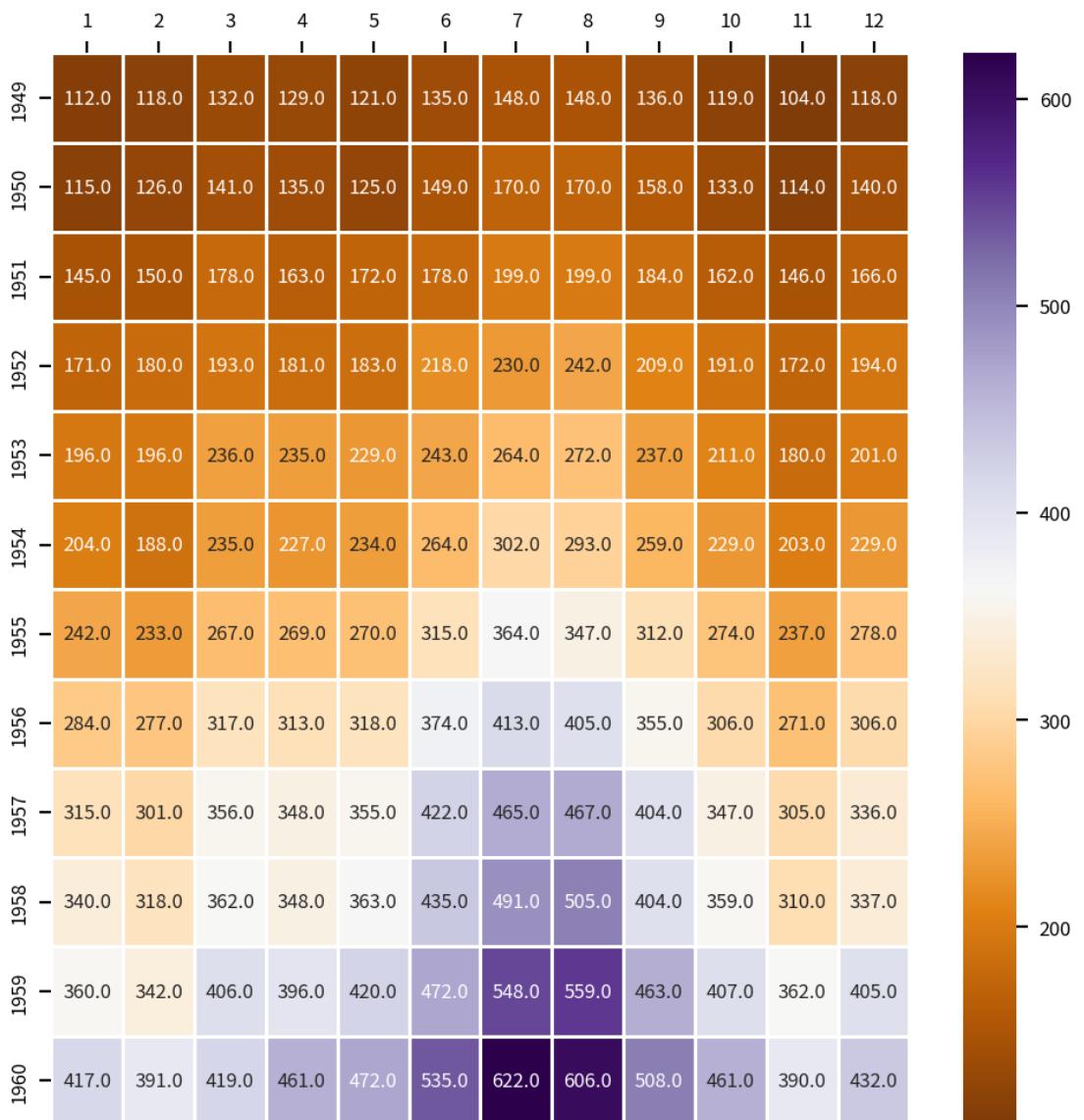
2) heatmap 그리기

```
# → annot=True : 수치값을 함께 표시함
# → fmt: annot=True가 설정된 경우 표시되는 수치값의 형식 지정
# → linewidth: 각 셀 사이의 선 굵기
```

```
# → cmap: 칼라맵 ('Greys', 'Purples', 'Blues', 'Greens', 'Oranges',
#                   'Reds',
#                   'YlOrBr', 'YlOrRd', 'OrRd', 'PuRd', 'RdPu', 'BuPu',
#                   'GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn', 'YlGn',
#                   'PiYG', 'PRGn', 'BrBG', 'PuOr', 'RdGy', 'RdBu',
#                   'RdYlBu',
#                   'RdYlGn', 'Spectral', 'coolwarm', 'bwr', 'seismic',
#                   'berlin', 'managua', 'viamo')
sb.heatmap(data=df2, annot=True, fmt="0.1f", linewidth=0.5,
            cmap="PuOr")

# 3) 그래프 꾸미기
ax.set_xlabel("")
ax.set_ylabel("")
ax.xaxis.tick_top()           # x축의 변수 이름을 상단으로 이동

# 4) 출력
plt.tight_layout()           # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show()                    # 그래프 화면 출력
plt.close()                   # 그래프 작업 종료
```



png

- 여름(6-8월)에 진한 색 → 승객 수가 가장 많음
- 연도가 증가할수록 전체 색 농도가 짙어짐 → 항공 운송 수요 증가 추세
- 연도별 패턴은 거의 동일하지만 크기만 증가 → 계절성 + 추세 구조가 혼합된 전형적 시계열 패턴



[LAB 06] 6. 집단별 요약 시각화



#01. 준비작업



1. 패키지 가져오기

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```



2. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



3. 데이터 가져오기

```
origin = load_data('penguins')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/penguins.xlsx
[94m[desc] [0m 남극 팔мер 군도의 펭귄 3종에 대해 신체 치수와 서식지 정보(출처:
seaborn 내장 데이터)
```

field	description
species	펭귄 종
island	서식지

bill_length_mm	부리 길이
bill_depth_mm	부리 두께
flipper_length_mm	날개 길이
body_mass_g	몸무게
sex	성별

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181	3750	MALE
1	Adelie	Torgersen	39.5	17.4	186	3800	FEMALE
2	Adelie	Torgersen	40.3	18.0	195	3250	FEMALE
3	Adelie	Torgersen	36.7	19.3	193	3450	FEMALE
4	Adelie	Torgersen	39.3	20.6	190	3650	MALE
...
329	Gentoo	Biscoe	47.2	13.7	214	4925	FEMALE
330	Gentoo	Biscoe	46.8	14.3	215	4850	FEMALE
331	Gentoo	Biscoe	50.4	15.7	222	5750	MALE
332	Gentoo	Biscoe	45.2	14.8	212	5200	FEMALE
333	Gentoo	Biscoe	49.9	16.1	213	5400	MALE

334 rows × 7 columns



#02. barplot

- 집단별 평균/합계/비율 등 요약 통계 시각화의 기본형
- estimator 파라미터로 평균(mean) 외에도 median, sum 등 자유롭게 설정 가능
- 범주형 분석에서 가장 표준적으로 쓰이는 그래프

```
# 1) 그래프 초기화
width_px = 1280 # 그래프 가로 크기
height_px = 720 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) barplot 그리기
sb.barplot(
    data=origin, # 사용할 데이터프레임
```

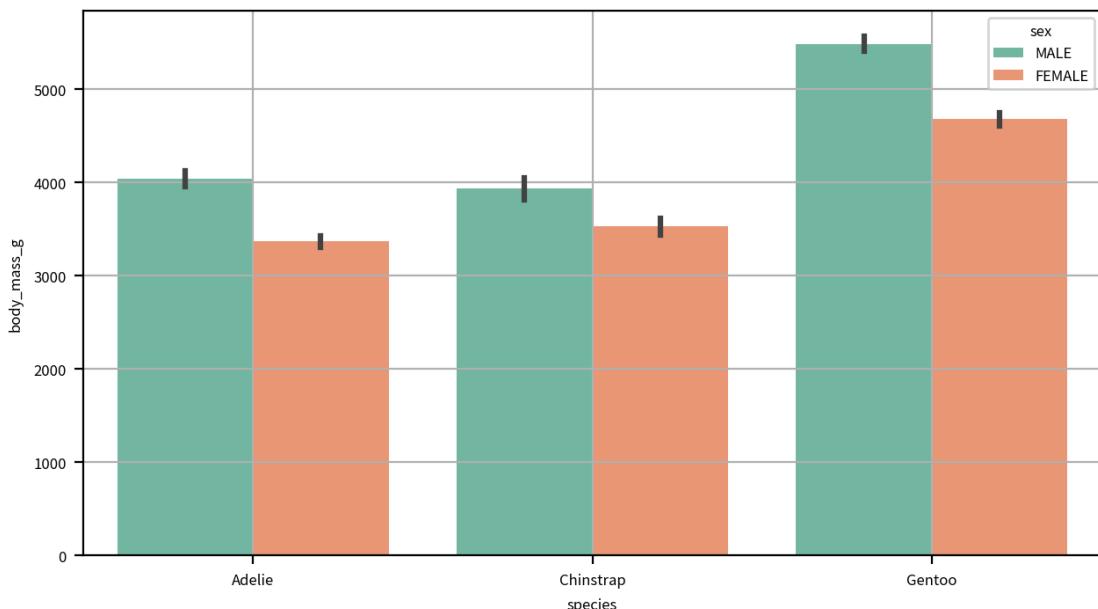
```

x="species",
y="body_mass_g",
hue="sex",
estimator=np.mean,
errorbar=("ci", 95),
준편차, None=없음
palette="Set2"
"bright", ...
)

# 3) 그래프 꾸미기
ax.grid(True)          # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()    # 여백 제거
plt.show()             # 그래프 화면 출력
plt.close()            # 그래프 작업 종료

```



png



#03. countplot (빈도 그래프)

- 범주형 빈도(Count)를 바로 보여주는 가장 단순하고 직관적인 요약 그래프
- 기술통계 보고서·EDA에서 거의 항상 등장

```

# 1) 그래프 초기화
width_px  = 1280           # 그래프 가로 크기
height_px = 720            # 그래프 세로 크기

```

```

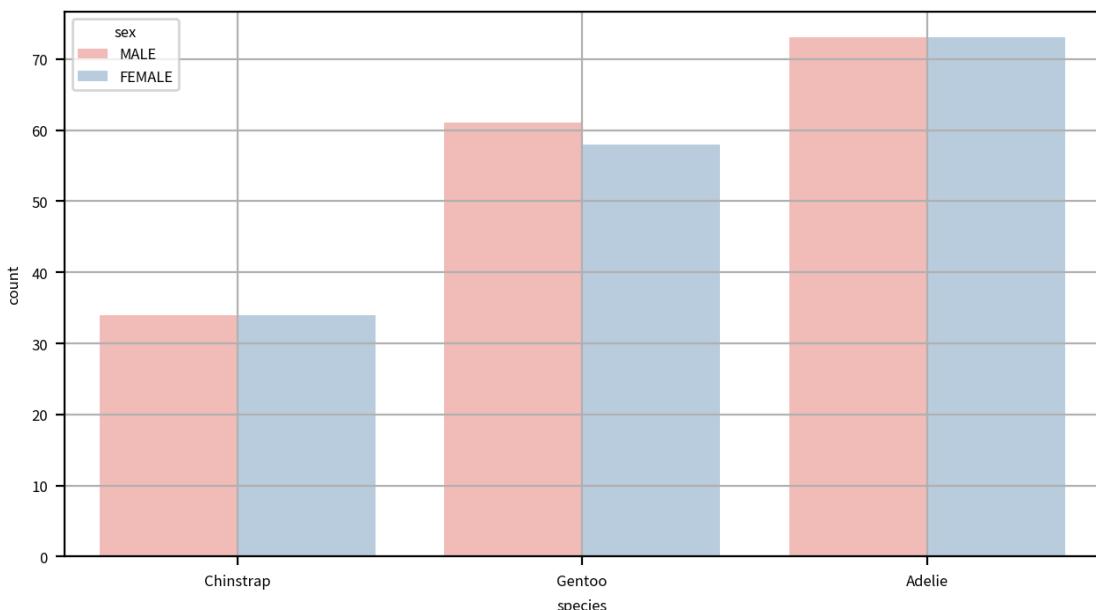
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) barplot 그리기
sb.countplot(
    data=origin, # 사용할 데이터프레임
    x="species", # 집계할 범주
    hue="sex", # 그룹 구분: None 가능
    order=['Chinstrap', 'Gentoo', 'Adelie'], # x축 범주 순서
    palette="Pastel1" # 색상 팔레트
)

# 3) 그래프 꾸미기
ax.grid(True) # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout() # 여백 제거
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png



#04. pointplot

- 점+오차막대 형태의 집단별 평균 + 신뢰구간 요약

- 여러 그룹(hue) 비교 시 가장 깔끔하고 해석성이 높음
- 회귀분석 전 EDA에서 추세 파악용으로 인기

```

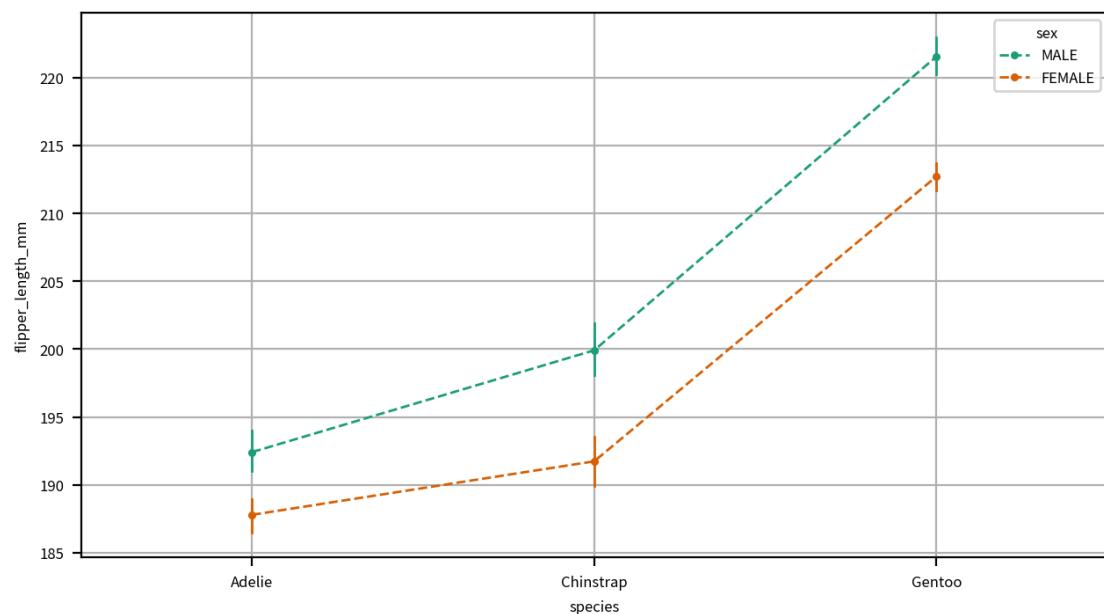
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) barplot 그리기
sb.pointplot(
    data=origin,                      # 사용할 데이터프레임
    x="species",
    y="flipper_length_mm",
    hue="sex",
    estimator=np.mean,                # 요약 방식
    errorbar=("ci", 95),             # 신뢰구간(기본 95%)
    linestyles="--",                 # "-", "--", ":" , "-."
    linewidth=1,
    markers="o",                     # 마커 종류: "o", "s", "D", "^", "v",
    "*",
    palette="Dark2"
)

# 3) 그래프 꾸미기
ax.grid(True)                         # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                   # 여백 제거
plt.show()                           # 그래프 화면 출력
plt.close()                          # 그래프 작업 종료

```



.png



[LAB-06] 7. 데이터간의 관계 시각화 (1) - Scatter Plot (산점도 그래프)



데이터 관계 시각화 그래프 4종

그래프	주요 목적	회귀선	그룹 비교	사용 상황
ScatterPlot	기본 관계 파악	X	X	기초 탐색
RegPlot	선형 경향 + 신뢰구간	O	X	관계 해석 강화
LmPlot	그룹별 선형관계 비교	O	O	비교 분석(성별/지역 등)
PairPlot	전체 변수 관계 한눈에 보기	변수쌍마다 단순 산점도	제한적(hue 지원)	EDA 초기 전체 스캔



Scatter Plot의 이해

- 두 연속형 변수의 관계를 시각화 하는 가장 기본적인 그래프
- 두 연속형 변수간의 영향력(분포와 상관 경향)을 점(point)으로 표시
- 패턴(직선적·곡선적), 군집 형태, 이상치를 쉽게 파악
- 통계적 모델은 포함하지 않으며 관계의 형태를 “그려만 준다”

언제 쓰나? → 변수 간 기본적인 관계가 있는지 빠르게 확인할 때



1. Scatter Plot 유형



2. Scatter Plot의 해석

마커들이 오밀조밀 뭉쳐 있으면 두 변수는 서로 관련성 정도가 높고 흩어져 있으면 관련성이 낮다.

이러한 관계를 상관관계라고 한다.

| 추론통계의 상관분석에서 좀 더 자세히 다룹니다.

예시

아래의 그래프에서 SAT에 응시한 학생 비율이 높을수록 수학 평균 점수는 낮아지는 경향이 있다고 볼 수 있다.



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```



2. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



3. 데이터 가져오기

```
origin = load_data("icecream")
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/icecream.xlsx
[94m[desc] [0m 기온과 아이스크림 판매량을 기록한 가상의 데이터 (메타데이터, 인덱스 없음)
[91m[!] Cannot read metadata [0m
```

	기온	판매량
0	23	431
1	36	593
2	30	512
3	25	474

4	26	476
5	31	523
6	29	491
7	32	526
8	33	550
9	24	456
10	34	566
11	35	581
12	27	480
13	28	487



#02. Scatter Plot 시작화

| hue 파라미터로 데이터 범주 구별이 가능함

```

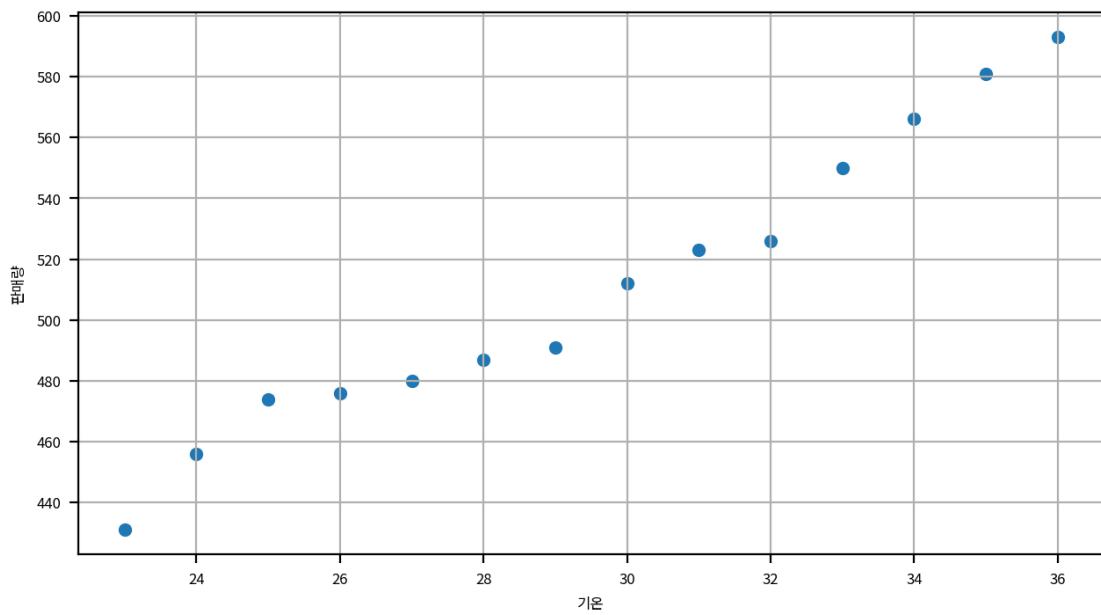
# 1) 그래프 초기화
width_px    = 1280                      # 그래프 가로 크기
height_px   = 720                        # 그래프 세로 크기
rows         = 1                          # 그래프 행 수
cols         = 1                          # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)
fig, ax      = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) BoxPlot 그리기
sb.scatterplot(data=origin, x='기온', y='판매량')

# 3) 그래프 꾸미기
ax.grid(True)                           # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                      # 여백 제거
plt.show()                             # 그래프 화면 출력
plt.close()                            # 그래프 작업 종료

```



png

알 수 있는 사실

기온에 따른 아이스크림 판매량을 산점도 그래프로 확인한 결과 기온이 상승할 수록 아이스크림 판매량도 증가하는 추세를 보이는 것으로 확인되었다.



#03. 추세선(회귀선) 그리기

주어진 데이터의 일반적인 경향이나 패턴을 나타내는 선

추세선은 주로 선형 회귀 분석을 통해 계산되며 이를 통해 데이터 간의 관계를 파악하고 예측 모델을 개발하는데 도움이 된다.

일반적으로 추세선을 그리기 위해서는 `scipy`나 `sklearn` 패키지를 통해 선형회귀 모델을 구현하여 회귀식을 도출해야 하지만 간단한 선형회귀의 경우 `numpy`를 통해서도 분석 모델을 도출할 수 있다.



1. 기울기(계수, 가중치)와 절편(상수항, 편향) 구하기

계수, 상수항 = `np.polyfit(x, y, 차수)`

통계학에서는 계수(기울기)를 가중치, 상수항(절편)를 편향이라고 하지만 `numpy`는 수학적 기능을 구현하고 있는 패키지이므로 `numpy`에서는 상수항과 계수라고 표현한다.

```
z = np.polyfit(origin['기온'], origin['판매량'], 1)
print("상수항:", z[0])
print("계수:", z[1])
```

상수항: 11.39780219780219
계수: 174.19340659340702



2. 회귀 분석 모형

상수항과 계수를 활용하여 $y = ax + b$ 에 해당하는 방정식을 확인한다.

```
expr = "y = %0.1f * x + %0.1f" % (z[0], z[1])
expr
```

'y = 11.4 * x + 174.2'



3. 분석 모형 객체 생성

$y = ax + b$ 에 해당하는 방정식 객체를 생성한다.

```
f = np.poly1d(z)
f
```

poly1d([11.3978022 , 174.19340659])



4. 분석모형을 활용한 판매량 예

방정식 $f(x)$ 에 x 값을 전달하면 그에 대한 결과를 확인할 수 있다.

```
x = 40
print("기온이 %d일 경우 아이스크림 판매량은 %f로 예상됩니다." % (x, f(x)))
```

기온이 40일 경우 아이스크림 판매량은 630.105495로 예상됩니다.



5. 전체 기온에 대한 예측 판매량 확

```
x = origin['기온']
y = f(x)
y
```

```
array([436.34285714, 584.51428571, 516.12747253, 459.13846154,
       470.53626374, 527.52527473, 504.72967033, 538.92307692,
       550.32087912, 447.74065934, 561.71868132, 573.11648352,
       481.93406593, 493.33186813])
```

6. 추세선을 포함하는 산점도 그래프

```
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

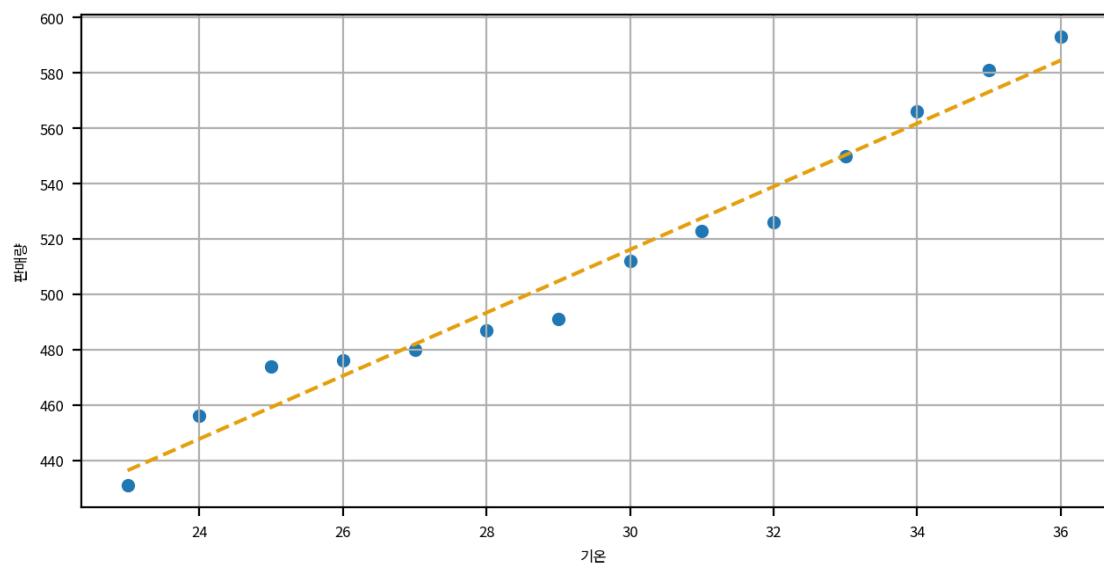
# 2-1) Scatter Plot 그리기
sb.scatterplot(data=origin, x='기온', y='판매량')

# 2-2) LinePlot 그리기
sb.lineplot(x=x, y=y, color="#e4a00c", linestyle="--")

# 3) 그래프 꾸미기
ax.set_title(expr, fontsize=14, pad=10)
ax.grid(True)                          # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                    # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료
```

$$y = 11.4 * x + 174.2$$



png



[LAB 06] 8. 데이터간의 관계 시각화 (2)



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
```



2. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



3. 데이터 가져오기

```
origin = load_data('penguins')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/penguins.xlsx
[94m[desc] [0m 남극 팔мер 군도의 펭귄 3종에 대해 신체 치수와 서식지 정보(출처:
seaborn 내장 데이터)
```

field	description
species	펭귄 종
island	서식지
bill_length_mm	부리 길이

bill_depth_mm	부리 두께
flipper_length_mm	날개 길이
body_mass_g	몸무게
sex	성별

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181	3750	MALE
1	Adelie	Torgersen	39.5	17.4	186	3800	FEMALE
2	Adelie	Torgersen	40.3	18.0	195	3250	FEMALE
3	Adelie	Torgersen	36.7	19.3	193	3450	FEMALE
4	Adelie	Torgersen	39.3	20.6	190	3650	MALE
...
329	Gentoo	Biscoe	47.2	13.7	214	4925	FEMALE
330	Gentoo	Biscoe	46.8	14.3	215	4850	FEMALE
331	Gentoo	Biscoe	50.4	15.7	222	5750	MALE
332	Gentoo	Biscoe	45.2	14.8	212	5200	FEMALE
333	Gentoo	Biscoe	49.9	16.1	213	5400	MALE

334 rows × 7 columns

4. 명목형 변수에 대한 전처리

```
df = origin.astype({"species": "category", "island": "category",
                    "sex": "category"})
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 334 entries, 0 to 333
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   species          334 non-null    category
 1   island            334 non-null    category
 2   bill_length_mm   334 non-null    float64
 3   bill_depth_mm   334 non-null    float64
 4   flipper_length_mm 334 non-null    int64  
 5   body_mass_g      334 non-null    int64  
 6   sex               333 non-null    category
```

```
dtypes: category(3), float64(2), int64(2)
memory usage: 11.9 KB
```

#02. RegPlot

- ScatterPlot에 단일 회귀선(Linear Regression Line)을 자동으로 추가
- 회귀선 주변의 신뢰구간(confidence interval)도 함께 표시
- 두 변수 사이의 선형적 관계가 어느 정도인지 직관적으로 해석 가능
- 단일 그래프에서만 사용하며(hue파라미터를 지원하지 않음), 세부 커스터마이징이 쉬움

언제 쓰나?

→ “상관이 있나?”뿐 아니라 “어느 정도 기울기인가?”도 보고 싶을 때

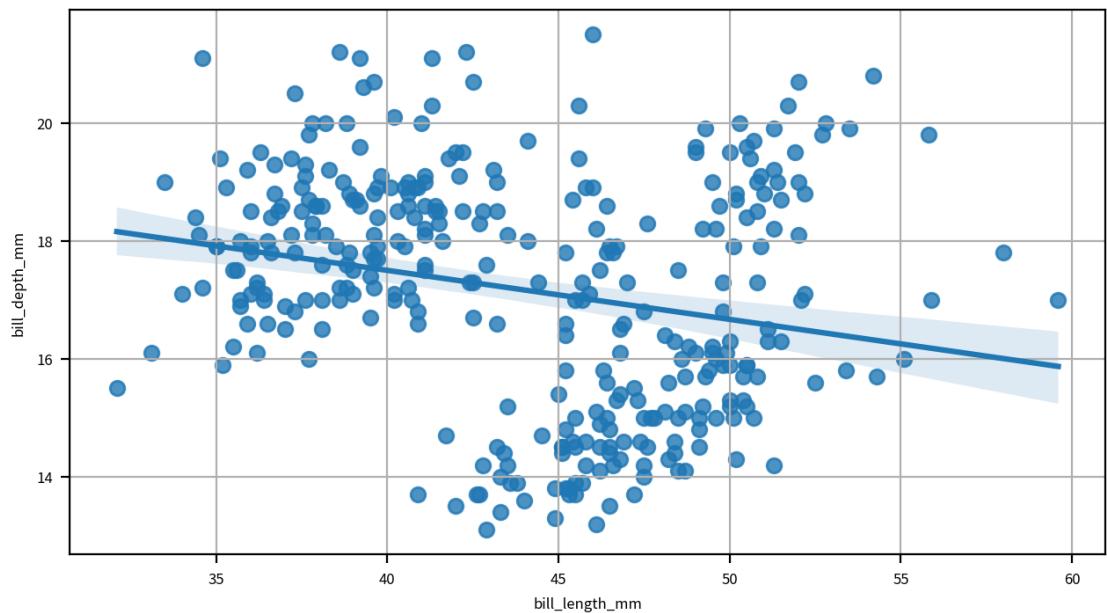
1. 부리 길이과 부리 두께의 관계

```
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) regplot 그리기
# `fit_reg=False` 파라미터를 추가할 경우 추세선이 표시되지 않는다. (scatterplot과
# 동일해짐)
# `scatter=False` 파라미터를 추가하면 산점도가 표시되지 않는다.(lineplot과 동일해
# 짐)
# `scatterplot()` 함수가 hue 파라미터를 적용하여 범주에 따라 구별할 수 있는 반면,
# RegPlot은 hue 파라미터를 적용할 수 없다.
sb.regplot(data=origin, x='bill_length_mm', y='bill_depth_mm')

# 3) 그래프 꾸미기
ax.grid(True)                          # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                     # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료
```



png



#03. LmPlot

- RegPlot의 기능을 확장한 형태 (scatterplot과 regplot의 결합)
- 여러 범주를 기준으로 행·열로 나눠(facet) 비교 가능
- 색(hue), 행(row), 열(col) 옵션을 통해 그룹 간 관계 차이를 시각화
- 관계 패턴을 다양한 하위집단으로 세분해 비교할 때 유용

언제 쓰나?

→ 성별/지역/연도 등 그룹별로 회귀선을 비교하고 싶을 때



1. 기본 사용 방법

산점도 그래프에 추세선을 추가함

기본 파라미터는 `regplot()` 메서드와 동일

`hue` 파라미터를 지원한다는 점에서 `regplot`과 차이를 보임

그래프 작성 코드가 기존 그래프와 다소 다르다

1) 그래프 초기화

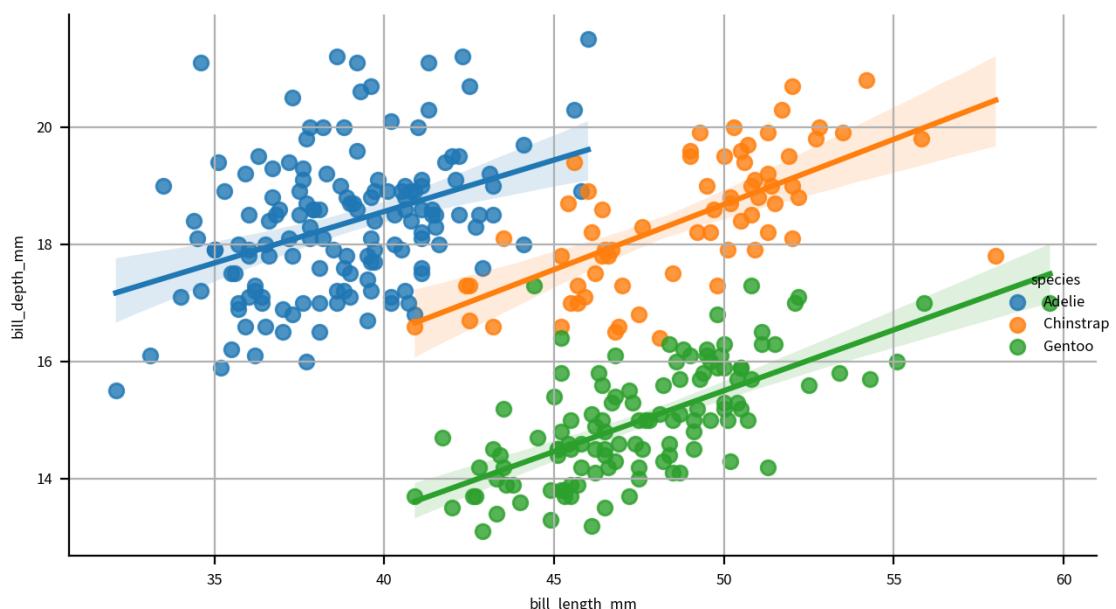
```
width_px = 1280 # 그래프 가로 크기
height_px = 720 # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)
```

```

# 2) LM Plot 그리기
# 그래프를 꾸미기 위해서 lmplot() 메서드로부터 리턴되는 객체(`g`)를 활용해야 함
g = sb.lmplot(data=df, x="bill_length_mm", y="bill_depth_mm",
               hue="species")
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
g.fig.set_figheight(figsize[1])
plt.grid()

# 3) 출력
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료

```



png



2. 조건별 병렬 시각화

범주에 따라 구분한 후 하위 변수를 사용하여 병렬 분할

```

# 1) 그래프 초기화
width_px   = 1280                      # 그래프 가로 크기
height_px  = 720                        # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) LM Plot 그리기
g = sb.lmplot(data=df, x="bill_length_mm", y="bill_depth_mm",
               hue="species", col='sex')
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])

```

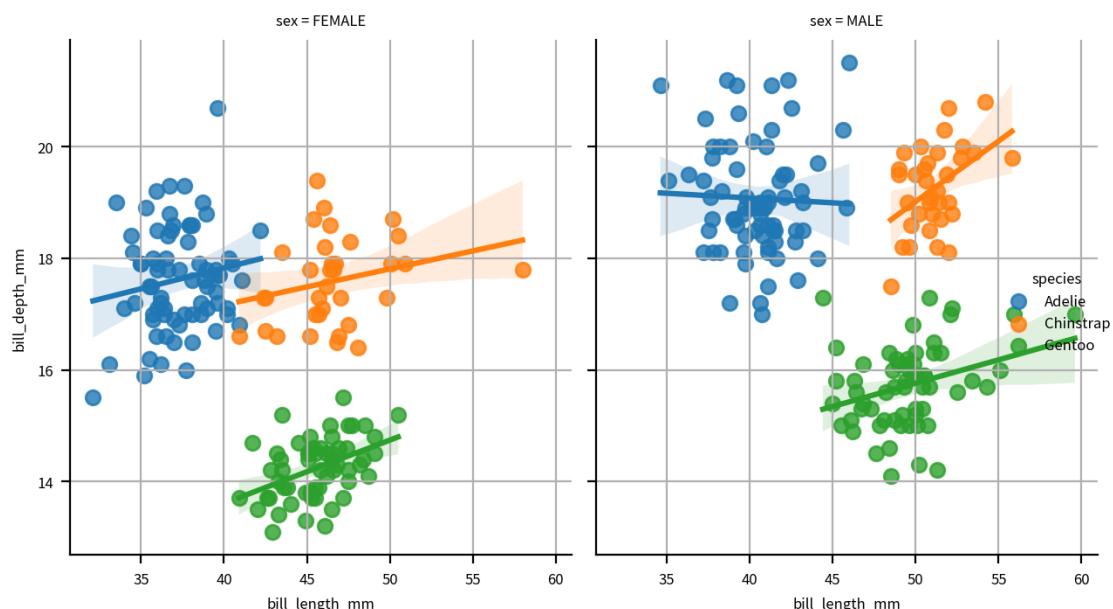
```

g.fig.set_figheight(figsize[1])

# 각 subplot에 grid 표시
for ax in g.axes.flatten():
    ax.grid(True)

# 3) 출력
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                  # 그래프 작업 종료

```



png



3. 모든 조건에 따라 행, 열로 분할

row, col 파라미터를 사용한다.

```

# 1) 그래프 초기화
width_px   = 1800                      # 그래프 가로 크기
height_px  = 900                        # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) LM Plot 그리기
g = sb.lmplot(data=df, x="bill_length_mm", y="bill_depth_mm",
               hue="species", col="species", row='sex')
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
g.fig.set_figheight(figsize[1])

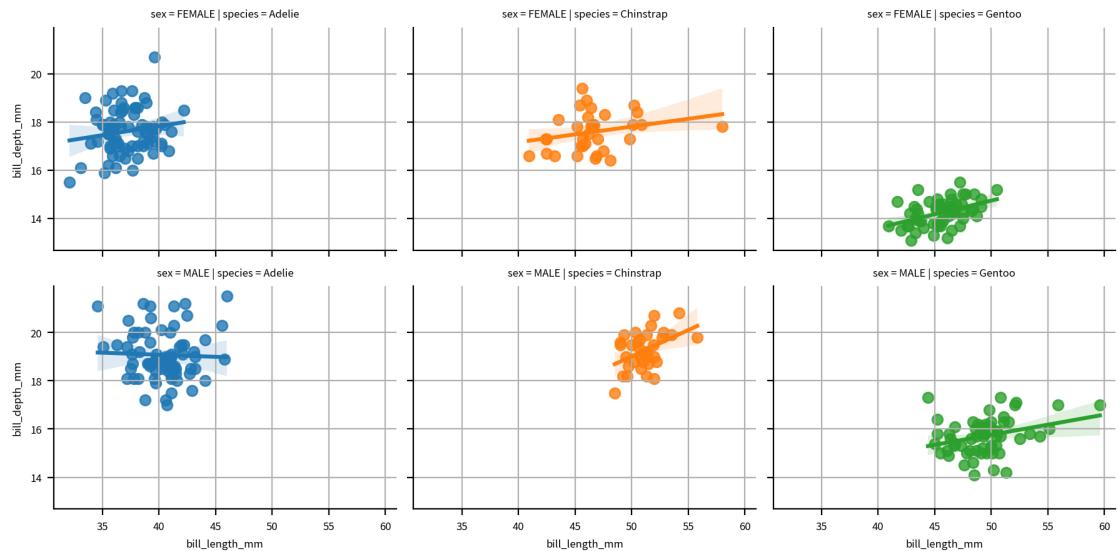
```

```

# 각 subplot에 grid 표시
for ax in g.axes.flatten():
    ax.grid(True)

# 3) 출력
plt.tight_layout()      # 여백 제거
plt.show()               # 그래프 화면 출력
plt.close()              # 그래프 작업 종료

```



png

#04. PairPlot (산점도 행)

- 모든 변수에 대한 교차 분석
- 전체 데이터의 구조와 변수 간 상관 관계를 한눈에 파악 가능
- 대각선 도표는 데이터의 주변 분포를 표시하기 위한 일변량 분포 도표(커널 밀도 곡선)이나 히스토그램이 그려진다.
- 탐색적 데이터 분석(EDA)에서 초기 전반 스캔용으로 매우 유용
- 다소 처리가 느리다.

언제 쓰나?

→ “전체 변수들이 서로 어떤 관계를 갖는지” 한 번에 파악할 때

1. 기본형

```

# 1) 그래프 초기화
width_px = 1200                      # 그래프 가로 크기

```

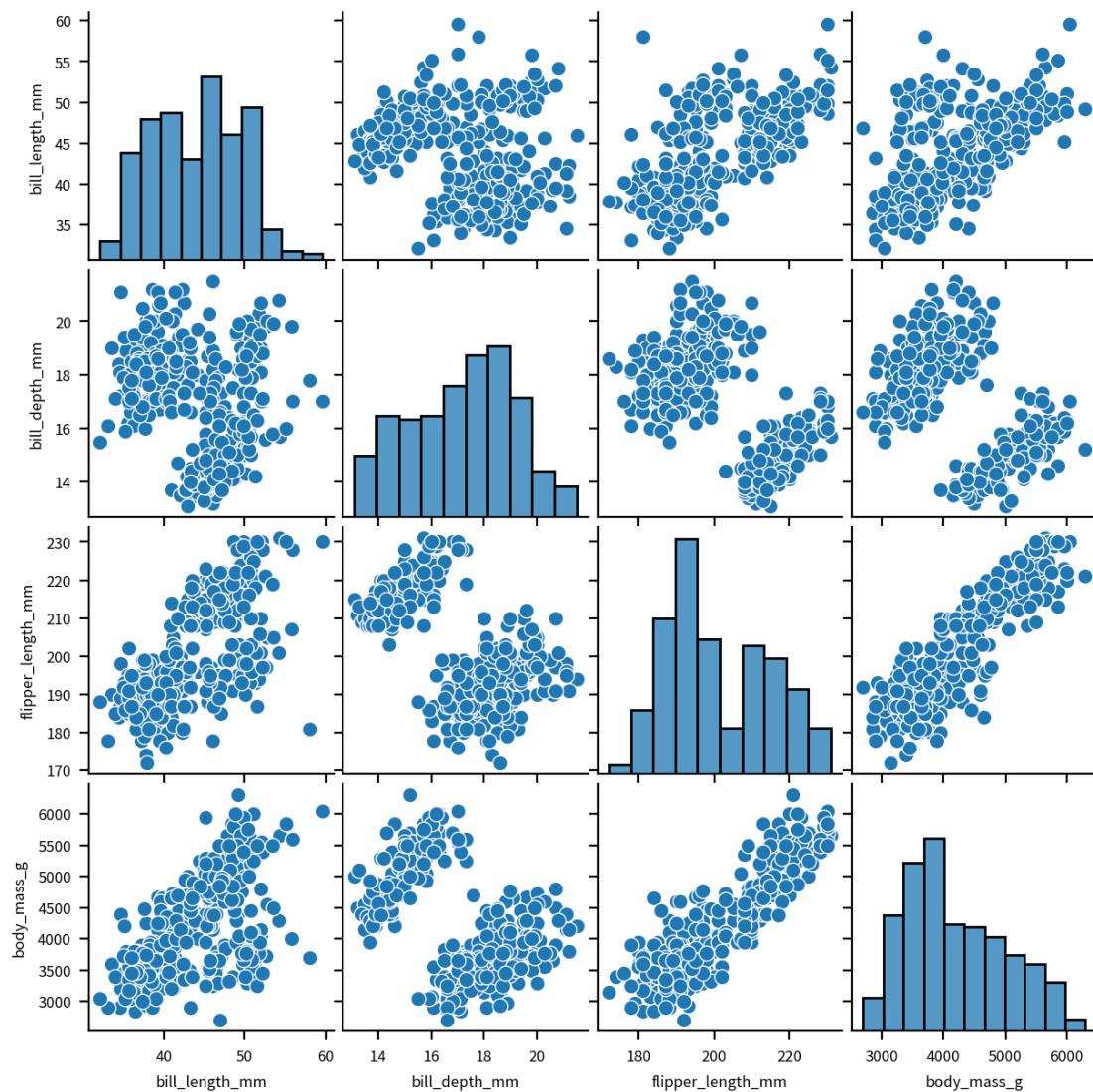
```

height_px = 1200 # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) Pair Plot 그리기
# `corner=True` 파라미터를 추가할 경우 아래쪽 삼각형만 플롯된다.
g = sb.pairplot(df)
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
g.fig.set_figheight(figsize[1])

# 3) 출력
plt.grid()
plt.show()
plt.close()

```



png



2. 범주별 구분

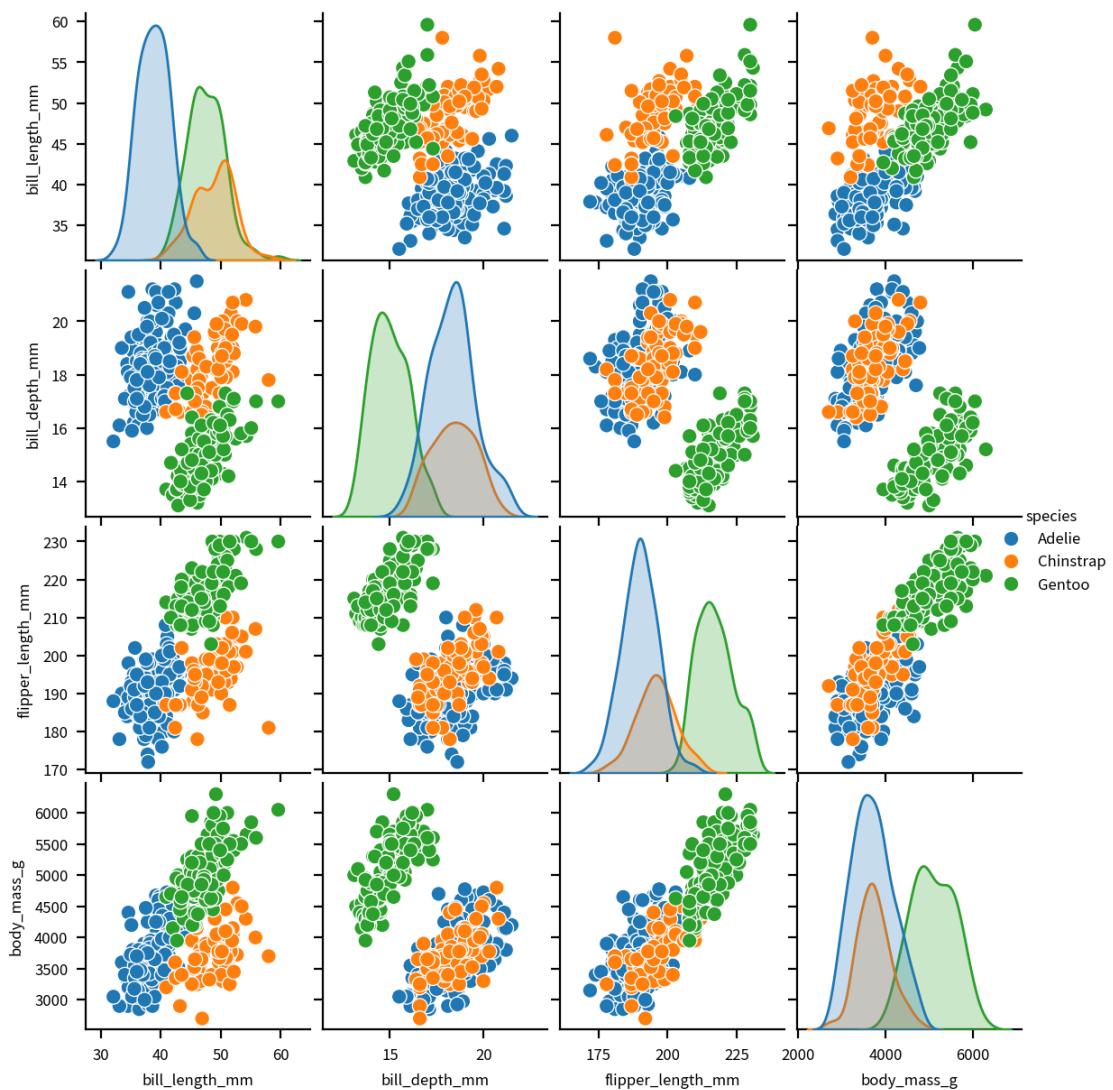
(1) hue 파라미터 적용

hue 파라미터에 변수를 할당하면 hue의 미론적 매핑이 추가되고 기본 주변 플롯이 계층화된 커널 밀도 추정(KDE)으로 변경된다.

```
# 1) 그래프 초기화
width_px  = 1200                                # 그래프 가로 크기
height_px = 1200                                 # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) Pair Plot 그리기
# `diag_kind` 파라미터에 `hist` 값을 적용한다.
# -> 적용 가능한 값: `auto`, `hist`, `kde`(기본값)
g = sb.pairplot(df, hue='species', diag_kind='kde')
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
g.fig.set_figheight(figsize[1])

# 3) 출력
plt.grid()
plt.show()
plt.close()
```



png



3. 선택적 변수 적용

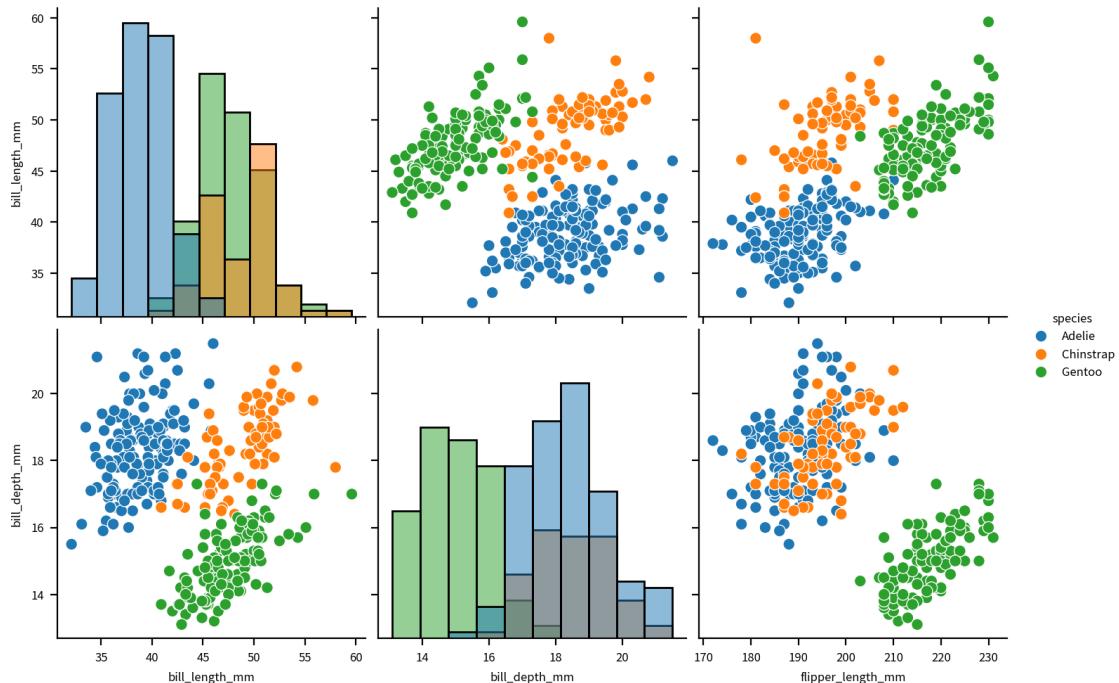
x_vars 파라미터와 y_vars 파라미터에 원하는 변수를 리스트 타입으로 지정한다.

```
# 1) 그래프 초기화
width_px  = 1600 # 그래프 가로 크기
height_px = 1000 # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) Pair Plot 그리기
g = sb.pairplot(df, hue='species', diag_kind='hist',
                 x_vars=["bill_length_mm", "bill_depth_mm",
                         "flipper_length_mm"],
                 y_vars=["bill_length_mm", "bill_depth_mm"])
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
```

```
g.fig.set_figheight(figsize[1])
```

```
# 3) 출력  
plt.grid()  
plt.show()  
plt.close()
```



png



4. 데이터를 그룹별로 묶어서 표시하기

pairplot() 메서드가 리턴하는 객체를 받아서 map_lower() 메서드를 호출한다.

map_lower() 메서드에 다른 종류의 함수 이름을 적용하면 대각선 기준으로 서로 다른 종류의 시각화 결과물을 표시할 수 있다.

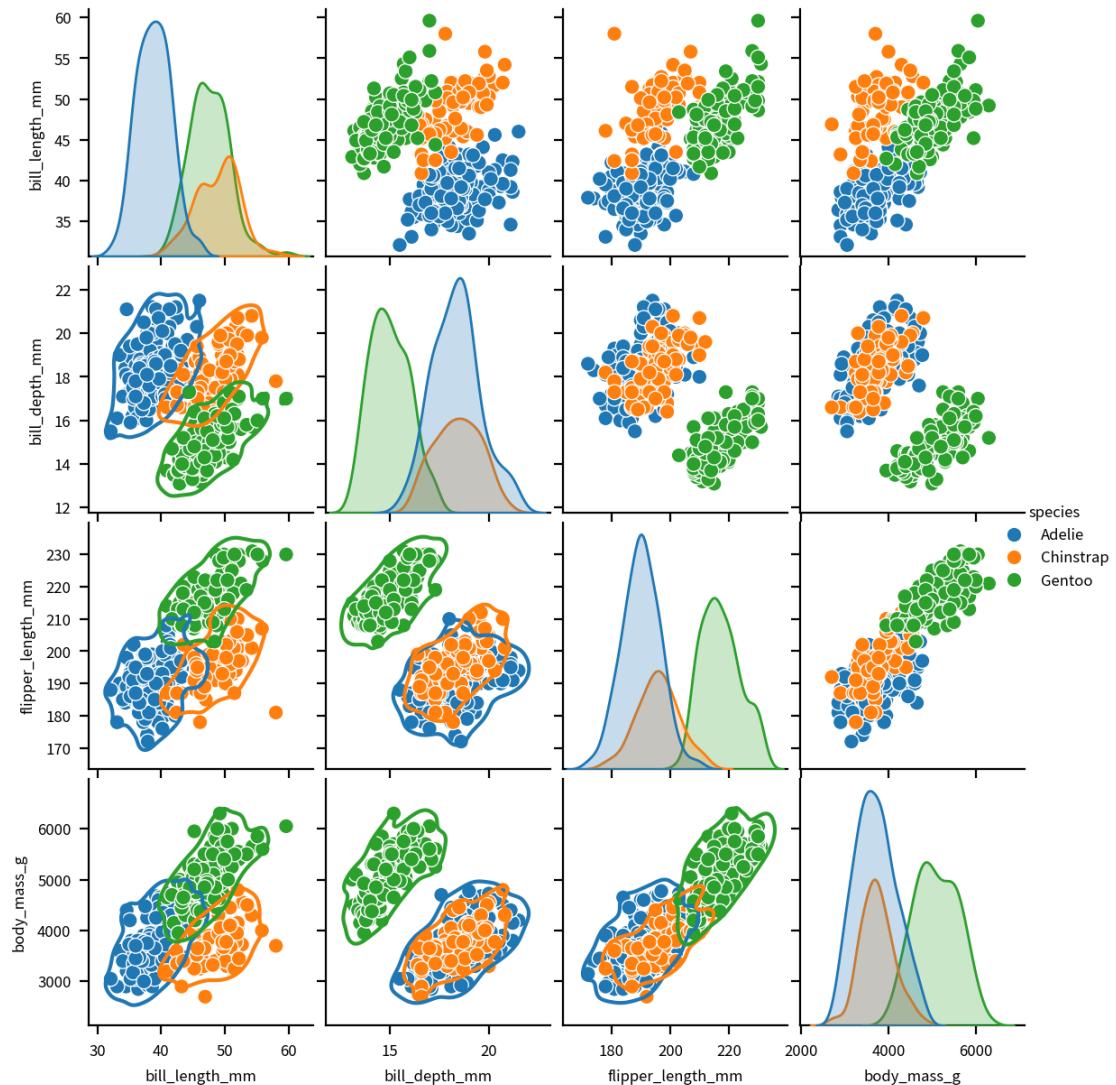
```
# 1) 그래프 초기화
```

```
width_px = 1200 # 그래프 가로 크기  
height_px = 1200 # 그래프 세로 크기  
figsize = (width_px / my_dpi, height_px / my_dpi)
```

```
# 2) Pair Plot 그리기
```

```
g = sb.pairplot(df, hue='species')  
g.map_lower(sb.kdeplot, levels=1, color=0.2)  
g.fig.set_dpi(my_dpi)  
g.fig.set_figwidth(figsize[0])  
g.fig.set_figheight(figsize[1])
```

```
# 3) 출력  
plt.grid()  
plt.show()  
plt.close()
```



png



[LAB-07] 1. 지도 시작화



#01. 준비작업



1. 라이브러리 참조

folium 패키지가 설치되어 있어야 한다.

```
from hossam import load_data
from os import path, mkdir
import folium
```



#02. 지도 표현하기



1. 지도 객체 생성

지도의 중심이 되는 위도와 경도를 설정

```
MY_PLACE = [37.4935982, 127.0327129]
```



2. 지도 불러오기

```
# zoom_start: 배율 1~22
map_osm1 = folium.Map(location=MY_PLACE, zoom_start=17)

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

# 지도가 표시된 웹 페이지 파일이 저장됨
# --> 결과 확인은 윈도우 폴더 창에서 직접 웹 페이지 파일 더블 클릭
map_osm1.save('output/map_osm1.html')
```



3. 지도 객체에 마커 추가

1) 일반 마커

아이콘 색상값 종류

'lightgreen', 'darkgreen', 'darkblue', 'cadetblue', 'orange', 'lightred', 'darkred', 'green', 'blue',
'black', 'lightblue', 'white', 'lightgray', 'red', 'pink', 'beige', 'gray', 'purple', 'darkpurple'

```
# 새로운 지도 객체 생성
map_osm2 = folium.Map(location=MY_PLACE, zoom_start=17)

# 마커 객체 생성
# -> 마커가 표시될 위치 [위도, 경도], 클리시 표시될 메시지, 아이콘 색상
my_marker = folium.Marker(MY_PLACE,
                           popup='아이티윌 교육센터',
                           icon=folium.Icon(color='darkred'))

my_marker.add_to(map_osm2) # 마커 객체를 지도에 추가함

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

# 지도가 표시된 웹 페이지 파일이 저장됨
# --> 결과 확인은 윈도우 폴더 창에서 직접 웹 페이지 파일 더블 클릭
map_osm2.save('output/map_osm2.html')
```

2) 사용자 지정 아이콘, HTML 팝업

```
# 새로운 지도 객체 생성
map_osm3 = folium.Map(location=MY_PLACE, zoom_start=17)

# HTML을 사용한 팝업
popup_html = folium.Popup("<div style='white-space: nowrap'><h3>아이티
    윌 교육센터</h3><img src='https://www.itwill.co.kr/css/wvtex/
    img/wvUser/logo.png' width='100%'><br/><p>tel: <a
    href='tel:02-6255-8002'>02-6255-8002</a></p><p>서울특별시 강남구
    역삼로 120 2층 (역삼동, 성보역삼빌딩)</p></div>",
    parse_html=False)

# 사용자 지정 아이콘 이미지 사용
# --> 온라인 상의 URL, 내 컴퓨터 상의 상대, 절대경로 모두 가능함
icon_img = folium.features.CustomIcon('https://data.hossam.kr/
    favicon.png', icon_size=(50, 50))
```

```

# 마커 객체 생성
custom_marker = folium.Marker(MY_PLACE,
                             popup=popup_html,
                             icon=icon_img)

custom_marker.add_to(map_osm3) # 마커 객체를 지도에 추가함

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

map_osm3.save('output/map_osm3.html') #파일이 저장될 위치

```

3) 원형 마커(범위지정)

```

# 새로운 지도 객체 생성
map_osm4 = folium.Map(location=MY_PLACE, zoom_start=17)

# 마커 객체 생성
my_marker = folium.Marker(MY_PLACE, icon=folium.Icon(color='orange'))

# 원형마커
circle_marker = folium.CircleMarker(MY_PLACE,
                                      radius=100,                      # 범위(m단위)
                                      color='#3186cc',                  # 선 색상
                                      fill_color='#3186cc')             # 면 색상

# 원형 마커 위에 아이콘 마커 올리기
circle_marker.add_to(map_osm4)
my_marker.add_to(map_osm4)

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

map_osm4.save('output/map_osm4.html') #파일이 저장될 위치

```



#03. (예제) 서울의 고등학교 분포 확인하기



1. 데이터 준비하기

전국 초,중,고 학교 위치 데이터 (데이터 로드에 다소 시간이 소요됨)

```
origin = load_data('schools')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab07/schools.xlsx
[94m[desc] [0m 전국 초,중,고 학교 위치 데이터 (출처: 공공데이터 포털)
[91m[!] Cannot read metadata [0m
```

	학교ID	학 교 명	학 교 급 구 분	설립일자	설 립 형 태	본 교 분 교 구 분	운 영 상 태	소재 지지 번주 소	소재 지도 로명 주소	시도교육청 코드	시 도 교 育 청 명	교 育 지 원 청 원 청	생성	
0	B000004204	한 울 초 등 학 교	초 등 학 교	2008-09-01	공 립	본 교	운 영	경 기 도 화 성 시 향 남 읍	경 기 도 화 성 시 향 남 읍 행 정 중 앙 1 로 25. 한 울 초 등 학 교 (향 남 읍)	7530000	경 기 도 교 育 청	7679000	경 기 도 화 성 오 산 교 育 지 원 청	2013-1
1	B000011476	수 원 농 생 명 과 학 고 등 학 교	고 등 학 교	1936-07-01	공 립	본 교	운 영	경 기 도 수 원 시 장 안 구 영 화 동 109	경 기 도 수 원 시 장 안 구 광 교 산 로 13 (영 화	7530000	경 기 도 교 育 청	7541000	경 기 도 수 원 교 育 지 원 청	2013-1

2	B000009647	녹양중학교	중학교	2008-03-01	공립	본교	운영	경기도의정부시녹양동191-5	경기도의정부체육로187.녹양중학교(녹양동)	7530000	경기도교육청	경기도의정부교육지원청	2013-1
3	B000005955	초락초등학교	초등학교	1959-04-02	공립	본교	운영	충청남도당진시석문면샛터말길35(석문면)	충청남도당진시석문면샛터말길35(석문면)	8140000	충청남도교육청	충청남도당진교육지원청	2013-1
4	B000005385	상봉초등학교	초등학교	1946-09-01	공립	본교	운영	충청북도청주시흥덕구오송읍상봉길9.상봉초등학교(오송읍)	충청북도청주시흥덕구오송읍상봉길9.상봉초등학교(오송읍)	8000000	충청북도교육청	충청북도청주교육지원청	2013-1

...
11983	B000003371	대전 옥계초등학교	초등학교	1982-12-09	공립	본교	운영	대전 광역시 중구 옥계동 65	대전 광역시 중구 모암로 35 (옥계동·대전 옥계초등학교)	7430000	대전 광역시 교육청	7441000	대전 광역시 동부교육지원청	2013-1	
11984	B000003345	성덕초등학교	초등학교	2011-03-11	공립	본교	운영	광주 광역시 광산구 풍영로 313 . 성덕초등학교 (장덕동) 1042	광주 광역시 광산구 풍영로 313 . 성덕초등학교 (장덕동)	7380000	광주 광역시 교육청	7401000	광주 광역시 서부교육지원청	2013-1	
11985	B000005441	이월초등학교	초등학교	1920-04-01	공립	본교	운영	충청북도 진천군 이월면 송림6길 26 . 이월초등학교 (이월면) 667	충청북도 진천군 이월면 송림6길 26 . 이월초등학교 (이월면)	8000000	충청북도 교육청	8081000	충청북도 진천교육지원청	2013-1	
11986	B000009875	해안중학교	중학교	1979-03-08	공립	본교	운영	강원도 양구군 해안면 현	강원도 양구군 해안면	7800000	강원도 교	7951000	강원도 양구교	2013-1	

								리 143	편치 볼로 1279 (해 안 면)		육 청		육 지 원 청
11987	B000011237	초 계 중 학 교	중 학 교	1952-06-01	공 립	본 교	운 영	경상 남도 합천 군 초계 면 초계 중앙 로 83. 초계 중학 교 (초 계 면. 초계 중학 교)	9010000	경 상 남 도 교 育 청	9221000	경 상 남 도 합 천 교 育 지 원 청	2013-1

11988 rows × 20 columns



2. 데이터 전처리

1) 사용할 필드만 추출

```
df = origin.filter(['학교명', '학교급구분', '소재지도로명주소', '위도', '경  
도'])  
df
```

	학교명	학교급구 분	소재지도로명주소	위도	경도
0	한울초등학교	초등학교	경기도 화성시 향남읍 행정중앙1로 25. 한울초등학교 (향 남읍)	37.126961	126.917854
1	수원농생명과학고 등학교	고등학교	경기도 수원시 장안구 광교산로 13 (영화동.농생명과학고 등학교)	37.295154	127.019450
2	녹양중학교	중학교	경기도 의정부시 체육로 187. 녹양중학교 (녹양동)	37.761864	127.028084
3	초락초등학교	초등학교	충청남도 당진시 석문면 샛터말길 35 (석문면)	36.993080	126.510472

4	상봉초등학교	초등학교	충청북도 청주시 흥덕구 오송읍 상봉길 9 . 상봉초등학교 (오송읍)	36.638251	127.286142
...
11983	대전옥계초등학교	초등학교	대전광역시 중구 모암로 35 (옥계동. 대전옥계초등학교)	36.301199	127.449039
11984	성덕초등학교	초등학교	광주광역시 광산구 풍영로 313 . 성덕초등학교 (장덕동)	35.199148	126.814301
11985	이월초등학교	초등학교	충청북도 진천군 이월면 송림6길 26 . 이월초등학교 (이월면)	36.931076	127.431922
11986	해안중학교	중학교	강원도 양구군 해안면 편치불로 1279 (해안면)	38.283771	128.135686
11987	초계중학교	중학교	경상남도 합천군 초계면 초계중앙로 83 . 초계중학교 (초계면. 초계중학교)	35.560066	128.270502

11988 rows × 5 columns

```
df_test = df.copy()
df_test
```

	학교명	학교급구 분	소재지도로명주소	위도	경도
0	한울초등학교	초등학교	경기도 화성시 향남읍 행정중앙1로 25 . 한울초등학교 (향남읍)	37.126961	126.917854
1	수원농생명과학고등학교	고등학교	경기도 수원시 장안구 광교산로 13 (영화동. 농생명과학고등학교)	37.295154	127.019450
2	녹양중학교	중학교	경기도 의정부시 체육로 187 . 녹양중학교 (녹양동)	37.761864	127.028084
3	초락초등학교	초등학교	충청남도 당진시 석문면 샛터말길 35 (석문면)	36.993080	126.510472
4	상봉초등학교	초등학교	충청북도 청주시 흥덕구 오송읍 상봉길 9 . 상봉초등학교 (오송읍)	36.638251	127.286142
...
11983	대전옥계초등학교	초등학교	대전광역시 중구 모암로 35 (옥계동. 대전옥계초등학교)	36.301199	127.449039
11984	성덕초등학교	초등학교	광주광역시 광산구 풍영로 313 . 성덕초등학교 (장덕동)	35.199148	126.814301
11985	이월초등학교	초등학교	충청북도 진천군 이월면 송림6길 26 . 이월초등학교 (이월면)	36.931076	127.431922
11986	해안중학교	중학교	강원도 양구군 해안면 편치불로 1279 (해안면)	38.283771	128.135686
11987	초계중학교	중학교	경상남도 합천군 초계면 초계중앙로 83 . 초계중학교 (초계면. 초계중학교)	35.560066	128.270502

11988 rows × 5 columns

2) 서울시의 고등학교만 추출

학교급구분 필드 값이 고등학교이고, 소재지도로명주소에 서울이라는 단어가 포함된 경우

LIKE 연산

컬럼이름.str.contains('검색어')

```
df2 = df.query("학교급구분 == '고등학교' and 소재지도로명주  
소.str.contains('서울')")
```

```
df2
```

	학교명	학교급 구분	소재지도로명주소	위도	경도
6	경기고등학교	고등학 교	서울특별시 강남구 영동대로 643 . 경기고등학교 (삼성동)	37.517565	127.056076
89	대일관광고등학 교	고등학 교	서울특별시 양천구 신정이펜1로 11 . 대일관광고등학교 (신정 동. 대일관광고등학교)	37.511414	126.834907
97	한광고등학교	고등학 교	서울특별시 강서구 등촌로13길 110 (화곡동)	37.538844	126.857922
98	상일미디어고등 학교	고등학 교	서울특별시 강동구 천호대로219길 61 . 상일미디어고등학교 (상일동)	37.549470	127.170767
118	선정국제관광고 등학교	고등학 교	서울특별시 은평구 서오릉로20길 19 . 선정국제관광고등학교 (갈현동)	37.618705	126.909032
...
11429	성수고등학교	고등학 교	서울특별시 성동구 서울숲길 18 . 성수고등학교 (성수동1가)	37.547342	127.038253
11451	대진여자고등학 교	고등학 교	서울특별시 노원구 공릉로 438 . 대진여자고등학교 (중계동)	37.646174	127.067197
11564	도선고등학교	고등학 교	서울특별시 성동구 마장로 156 (하왕십리동)	37.566844	127.026996
11600	금호고등학교	고등학 교	서울특별시 성동구 금호로 118 (금호동1가. 금호고등학교)	37.553621	127.023434
11929	서울인공지능고 등학교	고등학 교	서울특별시 송파구 양산로 21 (거여동)	37.491753	127.142331

320 rows × 5 columns



3. 데이터 시각화

```
# zoom_start: 배율 1~22 (여기서는 출력 안함)
map_osm5 = folium.Map(location=MY_PLACE, zoom_start=12)

html = "<font color='green' style='white-space: nowrap'><b>%s</b></font>"

# 데이터프레임의 행 수만큼 반복하면서 마커 생성
for i in df2.index:
    # 행 우선 접근 방식으로 값 추출하기
    name = df2.loc[i, '학교명']
    lat = df2.loc[i, '위도']
    lng = df2.loc[i, '경도']

    # 추출한 정보를 지도에 표시
    popup_html = folium.Popup(html % name, parse_html=False)
    marker = folium.Marker([lat,lng], popup=popup_html)
    marker.add_to(map_osm5)

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

map_osm5.save('output/map_osm5.html') #파일이 저장될 위치
```



[LAB-07] 2. SVG 지도 시각화 (서울)



SVG(Scalable Vector Graphics)

JPEG, PNG와 같은 그래픽 포맷(Graphic format)의 하나

벡터 기반이기 때문에 리사이징이 되어도 전혀 깨지지 않고 모든 해상도에서 자유자재로 활용할 수 있다.

SVG파일 포맷은 XML로 구성되어 있기 때문에 BeautifulSoup 패키지를 활용하여 HTML 파싱과 같은 구현과정을 통해 원하는 부분을 취득, 변형 할 수 있다.



#01. 준비작업



1. 패키지 참조

bs4 패키지가 필요하다.

| 웹 페이지 데이터 수집 단원을 통해 이미 설치되어 있음

jenksipy 패키지 설치가 필요하다 (연구과제용)

```
from hossam import load_data

# jupyter 상에서 SVG 이미지를 표시하기 위한 패키지(jupyter 기본 내장 패키지)
from IPython.display import SVG

# TAG로부터 원하는 내용을 추출하는 클래스 -> SVG 이미지의 핸들링을 위함
from bs4 import BeautifulSoup

# 연구과제용
import jenksipy
```



2. 데이터 가져오기

```
origin = load_data("senior_lsf")
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/senior_lsf.xlsx  
[94m[desc] [0m 서울시의 행정구역별 노인복지시설의 수를 조사한 가상의 데이터  
[91m[!] Cannot read metadata [0m
```

	지역명	복지시설
0	Jongno-gu	61
1	Jung-gu	53
2	Yongsan-gu	110
3	Seongdong-gu	155
4	Gwangjin-gu	103
5	Dongdaemun-gu	146
6	Jungnang-gu	128
7	Seongbuk-gu	158
8	Gangbuk-gu	111
9	Dobong-gu	139
10	Nowon-gu	252
11	Eunpyeong-gu	154
12	Seodaemun-gu	103
13	Mapo-gu	160
14	Yangcheon-gu	192
15	Gangseo-gu	215
16	Guro-gu	192
17	Geumcheon-gu	75
18	Yeongdeungpo-gu	208
19	Dongjak-gu	143
20	Gwanak-gu	127
21	Seocho-gu	129
22	Gangnam-gu	184
23	Songpa-gu	173
24	Gangdong-gu	140

#02. 지도 이미지 가져오기

위키미디어에서 Seoul districts.svg 키워드로 검색하여 서울 지도 이미지를 내려받아 map_seoul.svg라는 이름으로 작업 폴더에 추가하고 open() 함수로 파일을 읽어올 수 있다.

https://commons.wikimedia.org/wiki/File:Seoul_districts.svg?uselang=ko

혹은 아래의 URL에 접속한 후 Ctrl+S를 눌러서 파일을 저장한다.

https://data.hossam.kr/data/lab07/map_seoul.svg

저장된 파일은 /작업폴더/svg/map_seoul.svg 경로에 저장한다.



1. 지도 이미지 읽어오기

```
map_file_path = "svg/map_seoul.svg"

try:
    with open(map_file_path, 'r', encoding="utf-8") as f:
        map_svg = f.read()
    print("파일 읽어오기 완료")

    # 출력되는 내용이 많으므로 내용 확인 후 주석으로 막아두자.
    #print(map_svg)
except Exception as e:
    print("파일 읽어오기 에러:", e)
```

파일 읽어오기 완료



2. 이미지 확인

```
SVG(map_svg)
```



svg

#03. 데이터 시각화

1. 단계별 색상 팔레트 만들기

단계는 분석가가 임의로 정한다.

색상값을 1단계 ~ 높은단계 순으로 점점 진한 색상이 되도록 구성

사용할 색상값 (단계별로 6개 색상 준비)

```
colors = ['#F1EEF6', '#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77',  
         '#980043']
```



2. BeautifulSoup 객체 생성

svg파일의 내용을 BeautifulSoup객체로 변환

```
soup = BeautifulSoup(map_svg, features="xml")
# 출력되는 내용이 많으므로 내용 확인 후 주석으로 막아두자.
#soup
```



[3] 구 단위로 추출

id속성을 갖는 path 태그 가져오기

```
# soup.select() 메서드의 리턴값은 항상 리스트이다.
# → path 태그이면서 id속성을 갖는 요소를 리스트로 반환
#   <path id="???" ... >
path_list = soup.select('path[id]')
print("가져온 도형의 수: ", len(path_list))
```

가져온 도형의 수: 25



[3] 지도에서 확인한 지역명 수 만큼 반복

```
for p in path_list:
    #print(p)

    지역명 = p['id']
    #print(지역명)

    복지시설수 = origin.query('지역명 == @지역명')['복지시설'].values[0]
    print(지역명, " → ", 복지시설수)

    # 복지시설 수에 따라 단계값 설정 (단계는 색상값의 수에 따른)
    if 복지시설수 > 250:    color_index = 5
    elif 복지시설수 > 200:  color_index = 4
    elif 복지시설수 > 150:  color_index = 3
    elif 복지시설수 > 100:  color_index = 2
    elif 복지시설수 > 50:   color_index = 1
    else:                  color_index = 0

    # svg 이미지의 면 색상 변경
    p['fill'] = colors[color_index]
```

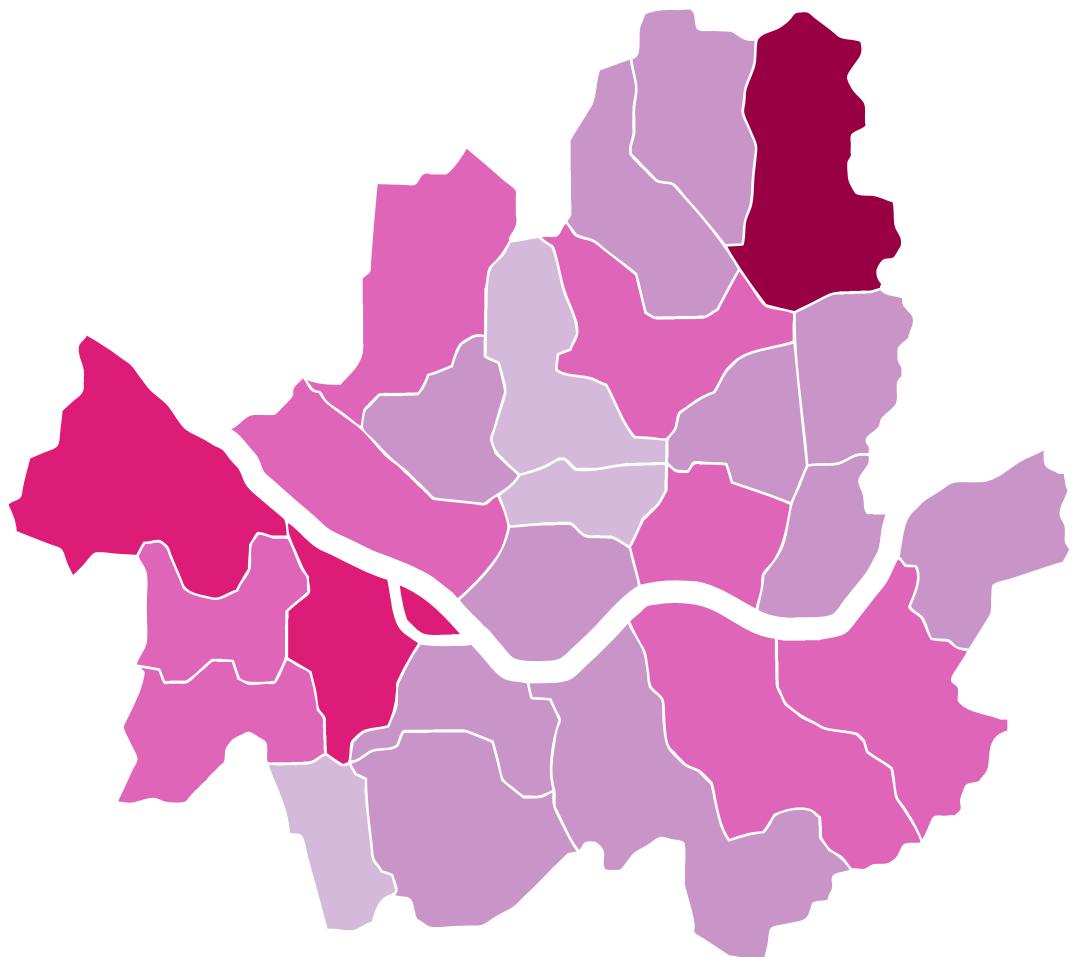
Dobong-gu → 139
Dongdaemun-gu → 146
Dongjak-gu → 143
Eunpyeong-gu → 154
Gangbuk-gu → 111
Gangdong-gu → 140
Gangseo-gu → 215
Geumcheon-gu → 75
Guro-gu → 192
Gwanak-gu → 127
Gwangjin-gu → 103
Gangnam-gu → 184
Jongno-gu → 61
Jung-gu → 53
Jungnang-gu → 128
Mapo-gu → 160
Nowon-gu → 252
Seocho-gu → 129
Seodaemun-gu → 103
Seongbuk-gu → 158
Seongdong-gu → 155
Songpa-gu → 173
Yangcheon-gu → 192
Yeongdeungpo-gu → 208
Yongsan-gu → 110



[4] 재구성된 내용을 토대로 새로운 svg 소스코드 얻기

```
# bs4 객체의 내용을 문자열로 리턴
new_seoul_svg = soup.prettify()

# jupyter에서 svg 이미지 표시하기
# 사용방법 -> SVG(소스문자열) 혹은 SVG(파일경로)
SVG(new_seoul_svg)
```



svg



[5] 생성된 이미지를 파일로 저장해야 하는 경우

```
# 저장된 파일은 윈도우 폴더창에서 직접 더블클릭 해서 웹 브라우저를 통해 확인해야 한다.  
with open('svg/new_seoul_svg.svg', 'w', encoding="utf-8") as f:  
    f.write(new_seoul_svg)
```



지도 시각화 연구과제

covid19_clinic 데이터는 2021년 05월 27일 기준 전국의 코로나 검사가 가능한 병원의 목록이다.

서울의 구 단위로 코로나 검사가 가능한 병원의 분포를 5단계로 구분하여 시각화 하시오.

색상 단계값은 아래의 리스트를 활용하세요.

```
colors = ['#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77', '#980043']
```



결과물 예시

img

img



#01. 데이터 불러오기

```
origin = load_data("covid19_clinic")
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab07/covid19_clinic.xlsx
[94m[desc] [0m 서울시의 Covid19 검진 가능 진료소 데이터 (출처: 공공데이터 포털)
[91m[!] Cannot read metadata [0m
```

	시도	시군구	의료기관명	주소
0	서울	강남구	강남구보건소	서울 강남구 삼성동(삼성2동) 8 강남구보건소
1	서울	강남구	삼성서울병원	서울 강남구 일원로81 삼성서울병원
2	서울	강남구	강남세브란스병원	서울 강남구 언주로211 강남세브란스병원
3	서울	강동구	강동구보건소	서울 강동구 성내동 541-2
4	서울	강동구	중앙보훈병원	서울 강동구 진황도로 61길 53
...
621	제주	제주시	한마음병원	연신로 52
622	제주	제주시	한국병원	서광로 193
623	제주	제주시	중앙병원	월랑로 91

624	제주	제주시	제주시(동부보건소)	제주 제주시 구좌읍 김녕리 1697-1 동부보건소 동부보건소
625	제주	제주시	제주시(서부보건소)	제주특별자치도 제주시 한림읍 한림리 966-1번지 (한림리)

626 rows × 4 columns

#02. 데이터 전처리



1. 필요한 변수만 추출

```
#seoul_df = origin.query('시도 == "서울"')
seoul_df = origin[origin["시도"] == '서울']
seoul_df
```

	시도	시군구	의료기관명	주소
0	서울	강남구	강남구보건소	서울 강남구 삼성동(삼성2동) 8 강남구보건소
1	서울	강남구	삼성서울병원	서울 강남구 일원로81 삼성서울병원
2	서울	강남구	강남세브란스병원	서울 강남구 언주로211 강남세브란스병원
3	서울	강동구	강동구보건소	서울 강동구 성내동 541-2
4	서울	강동구	중앙보훈병원	서울 강동구 진황도로 61길 53
...
66	서울	종로구	서울직십자병원	서울시 종로구 평동 164
67	서울	종로구	서울대학교병원	서울시 종로구 대학로 101(연건동)
68	서울	중랑구	중랑구보건소	서울 중랑구 신내2동 662 중랑구청
69	서울	중랑구	서울의료원	중랑구 신내로 156
70	서울	중랑구	녹색병원	중랑구 사가정로 49길 53

71 rows × 4 columns



2. 그룹별 집계

```
group_df = seoul_df.filter(["시군구", "의료기관명"]).groupby('시군구').count()
group_df.head()
```

	의료기관명
--	-------

시군구	
강남구	3
강동구	4
강북구	1
강서구	1
관악구	2



3. 구 이름에 대한 영문명 처리

앞 예제의 출력결과를 데이터셋의 구 이름과 직접 연결함

```
df = group_df.rename(
    columns={"의료기관명": "의료기관수"}, 
    index={
        "은평구": "Eunpyeong-gu",
        "영등포구": "Yeongdeungpo-gu",
        "동대문구": "Dongdaemun-gu",
        "강동구": "Gangdong-gu",
        "종로구": "Jongno-gu",
        "양천구": "Yangcheon-gu",
        "강남구": "Gangnam-gu",
        "중구": "Jung-gu",
        "송파구": "Songpa-gu",
        "성동구": "Seongdong-gu",
        "서초구": "Seocho-gu",
        "중랑구": "Jungnang-gu",
        "동작구": "Dongjak-gu",
        "노원구": "Nowon-gu",
        "구로구": "Guro-gu",
        "서대문구": "Seodaemun-gu",
        "도봉구": "Dobong-gu",
        "성북구": "Seongbuk-gu",
        "금천구": "Geumcheon-gu",
        "광진구": "Gwangjin-gu",
        "용산구": "Yongsan-gu",
        "관악구": "Gwanak-gu",
        "강서구": "Gangseo-gu",
        "강북구": "Gangbuk-gu",
        "마포구": "Mapo-gu"
    }
)
```

df

	의료기관수
시군구	
Gangnam-gu	3
Gangdong-gu	4
Gangbuk-gu	1
Gangseo-gu	1
Gwanak-gu	2
Gwangjin-gu	2
Guro-gu	3
Geumcheon-gu	2
Nowon-gu	3
Dobong-gu	2
Dongdaemun-gu	5
Dongjak-gu	3
Mapo-gu	1
Seodaemun-gu	2
Seocho-gu	3
Seongdong-gu	3
Seongbuk-gu	2
Songpa-gu	3
Yangcheon-gu	4
Yeongdeungpo-gu	5
Yongsan-gu	2
Eunpyeong-gu	5
Jongno-gu	4
Jung-gu	3
Jungnang-gu	3



Jenks Natural Breaks

GIS 분야에서 많이 사용하는 데이터의 구간을 나누는 방법.

데이터의 구간 수를 지정하면 값이 비슷한 항목끼리 그룹을 만드는 알고리즘으로 “시각화”라는 기준에서 봤을 때 사람의 눈이 가장 “자연스럽다”라고 느끼는 것에 초점을 두는 분류 방법이다.

1차원 데이터에 대한 k-means clustering 알고리즘과 같은 방식으로 작동한다.

1. 그룹의 갯수와 각 그룹의 중심값을 임의로 정함
2. 각 데이터 별로, 가장 가까운 중심값을 찾아 그룹을 할당
3. 할당된 결과를 가지고 그룹의 중심점을 재계산
4. 2~3을 반복

img

img

```
bins = jenks spy.jenks_breaks(df['의료기관수'], n_classes=5)
bins
```

```
[np.int64(1), np.int64(1), np.int64(2), np.int64(3), np.int64(4),
np.int64(5)]
```

#03. 지도 이미지 처리



1. 지도 불러오기

```
map_file_path = "svg/map_seoul.svg"

try:
    with open(map_file_path, 'r', encoding="utf-8") as f:
        map_svg = f.read()
    print("파일 읽어오기 완료")
except Exception as e:
    print("파일 읽어오기 에러:", e)

SVG(map_svg)
```

```
파일 읽어오기 완료
```



svg



2. 단계별 색상 팔레트 만들기

```
colors = ['#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77', '#980043']  
colors
```

```
[ '#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77', '#980043' ]
```



3. 지도에서 구 단위 추

```
soup = BeautifulSoup(map_svg)  
path_list = soup.select('path[id]')  
path_list[0]
```

```
/var/folders/wk/8tx_v8l94cqwt2b6dzgdc2h0000gn/T/
ipykernel_10929/342595318.py:1: XMLParsedAsHTMLWarning: It looks like
you're using an HTML parser to parse an XML document.
```

Assuming this really is an XML document, what you're doing might work, but you should know that using an XML parser will be more reliable. To parse this document as XML, make sure you have the Python package 'lxml' installed, and pass the keyword argument `features="xml"` into the BeautifulSoup constructor.

If you want or need to use an HTML parser on this document, you can make this warning go away by filtering it. To do that, run this code before calling the BeautifulSoup constructor:

```
from bs4 import XMLParsedAsHTMLWarning
import warnings

warnings.filterwarnings("ignore",
category=XMLParsedAsHTMLWarning)

soup = BeautifulSoup(map_svg)

<path clip-rule="evenodd" d="M964.064,164.667
c-1.447,9.018-0.285,18.105-2.002,27.506c-2.068,11.332-9.018,22.101-11.50
c-0.508,4.656-1.969,10.129-1.5,14.003c0.779,6.456,5.756,14.04,8.502,21.0
c0.539,4.856-0.953,11.628-1.502,17.504c-0.547,5.879-1.484,11.904-2,17.50
c-1.582,7.641-5.57,14.402-7.002,21.505c-1.725,8.558-1.271,18.438-3,27.50
c-14.793-19.111-31.705-39.509-48.51-58.013c-4.902-5.398-11.217-16.078-17
c-4.459-4.876-9.127-9.544-14.002-14.003c-0.148-1.02-1.354-0.98-1.502-2c-
c-2.484-9.723,2.434-16.186,3.5-24.005c1.156-1.678,0.176-5.493,0.5-8.001c
c-1.914-13.504,2.932-25.383,2.502-37.009c-0.459-12.384-5.236-23.798-6.00
c0.838-8.333,5.449-13.907,6.502-19.504c9.998-4.506,22.598-6.408,38.008-5
```

```
c3.451,0.612,7.951-0.803,10.502,0c9.178,2.887,3.551,20.857,10.002,25.005
```

```
c7.441,0.328,14.299,0.634,21.004,1C944.035,158.024,948.826,166.568,964.0  
fill="#C8C8C8" fill-rule="evenodd" id="Dobong-gu">></path>
```



4. 지도에서 확인된 지역명 만큼 반

```
for p in path_list:  
    지역명 = p['id']  
    #print(지역명)  
  
    의료기관수 = df.loc[지역명, '의료기관수']  
    # Dobong-gu → 2  
    #print(지역명, "-->", 의료기관수)  
  
    # 미리 생성한 구간만큼 반복  
    # [np.int64(1), np.int64(1), np.int64(2), np.int64(3),  
    # np.int64(4), np.int64(5)]  
    for i, v in enumerate(bins):  
        if i == 0:  
            continue  
        elif i + 1 < len(bins):  
            if 의료기관수 < v:  
                color_index = i - 1;  
                break  
        else:  
            if 의료기관수 ≤ v:  
                color_index = i - 1;  
                break  
                break  
  
    print(지역명, 의료기관수, color_index)  
    p['fill'] = colors[color_index]
```

```
Dobong-gu 2 2  
Dongdaemun-gu 5 4  
Dongjak-gu 3 3  
Eunpyeong-gu 5 4  
Gangbuk-gu 1 1  
Gangdong-gu 4 4  
Gangseo-gu 1 1  
Geumcheon-gu 2 2  
Guro-gu 3 3
```

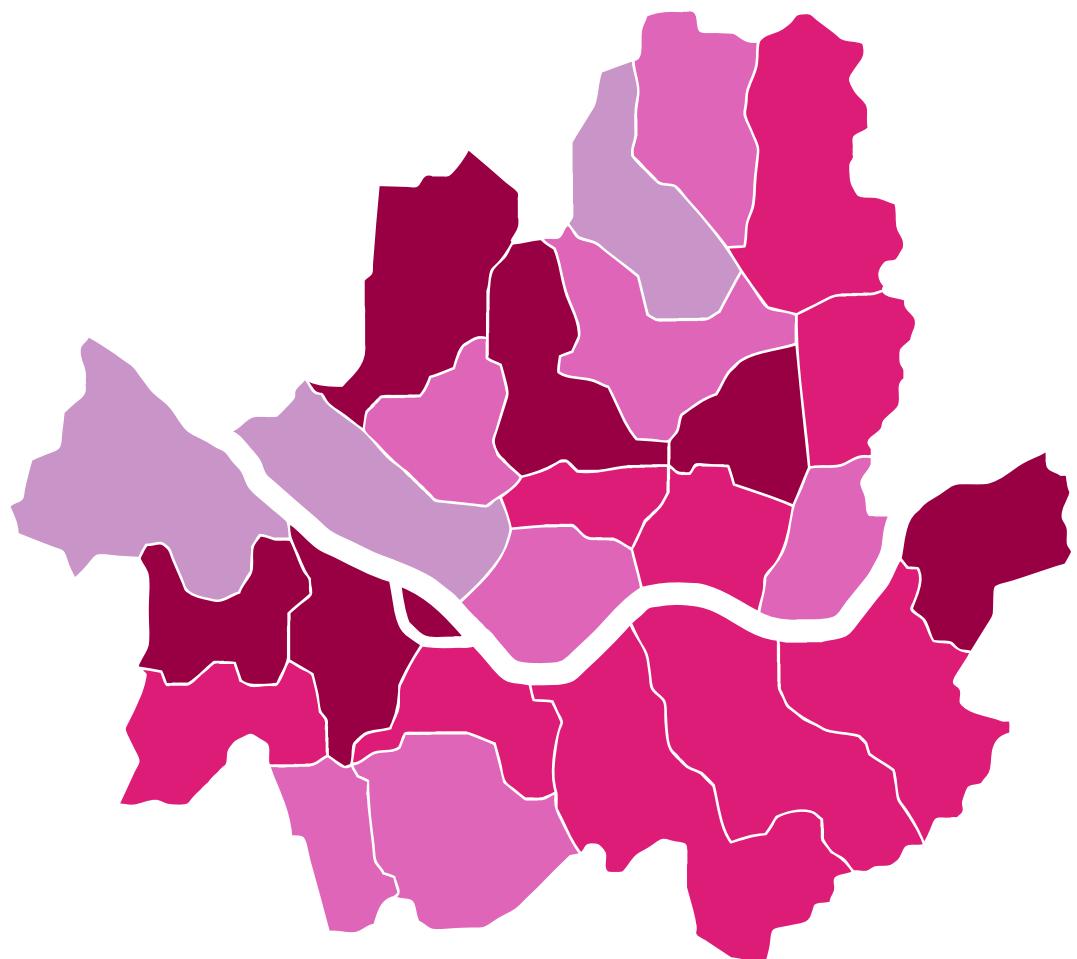
```
Gwanak-gu 2 2
Gwangjin-gu 2 2
Gangnam-gu 3 3
Jongno-gu 4 4
Jung-gu 3 3
Jungnang-gu 3 3
Mapo-gu 1 1
Nowon-gu 3 3
Seocho-gu 3 3
Seodaemun-gu 2 2
Seongbuk-gu 2 2
Seongdong-gu 3 3
Songpa-gu 3 3
Yangcheon-gu 4 4
Yeongdeungpo-gu 5 4
Yongsan-gu 2 2
```



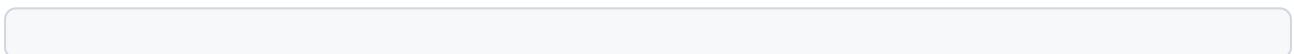
4. 재구성된 내용을 토대로 새로운 svg 생성

```
# bs4 객체의 내용을 문자열로 리턴
new_seoul_svg = soup.prettify()

# jupyter에서 svg 이미지 표시하기
# 사용방법 -> SVG(소스문자열) 혹은 SVG(파일경로)
SVG(new_seoul_svg)
```



svg





[LAB-07] 2. SVG 지도 시각화 (서울)



SVG(Scalable Vector Graphics)

JPEG, PNG와 같은 그래픽 포맷(Graphic format)의 하나

벡터 기반이기 때문에 리사이징이 되어도 전혀 깨지지 않고 모든 해상도에서 자유자재로 활용할 수 있다.

SVG파일 포맷은 XML로 구성되어 있기 때문에 BeautifulSoup 패키지를 활용하여 HTML 파싱과 같은 구현과정을 통해 원하는 부분을 취득, 변형 할 수 있다.



#01. 준비작업



1. 패키지 참조

bs4 패키지가 필요하다.

| 웹 페이지 데이터 수집 단원을 통해 이미 설치되어 있음

```
from hossam import load_data

# jupyter 상에서 SVG 이미지를 표시하기 위한 패키지(jupyter 기본 내장 패키지)
from IPython.display import SVG

# TAG로부터 원하는 내용을 추출하는 클래스 -> SVG 이미지의 핸들링을 위함
from bs4 import BeautifulSoup

from pandas import DataFrame
import numpy as np
import os
```



2. 데이터 가져오기

```
origin = load_data("senior_lsf")
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/senior_lsf.xlsx  
[94m[desc] [0m 서울시의 행정구역별 노인복지시설의 수를 조사한 가상의 데이터  
[91m[!] Cannot read metadata [0m
```

	지역명	복지시설
0	Jongno-gu	61
1	Jung-gu	53
2	Yongsan-gu	110
3	Seongdong-gu	155
4	Gwangjin-gu	103
5	Dongdaemun-gu	146
6	Jungnang-gu	128
7	Seongbuk-gu	158
8	Gangbuk-gu	111
9	Dobong-gu	139
10	Nowon-gu	252
11	Eunpyeong-gu	154
12	Seodaemun-gu	103
13	Mapo-gu	160
14	Yangcheon-gu	192
15	Gangseo-gu	215
16	Guro-gu	192
17	Geumcheon-gu	75
18	Yeongdeungpo-gu	208
19	Dongjak-gu	143
20	Gwanak-gu	127
21	Seocho-gu	129
22	Gangnam-gu	184
23	Songpa-gu	173
24	Gangdong-gu	140

#02. 지도 이미지 가져오기

위키미디어에서 Seoul districts.svg 키워드로 검색하여 서울 지도 이미지를 내려받아 map_seoul.svg라는 이름으로 작업 폴더에 추가하고 open() 함수로 파일을 읽어올 수 있다.

https://commons.wikimedia.org/wiki/File:Seoul_districts.svg?uselang=ko

혹은 아래의 URL에 접속한 후 Ctrl+S를 눌러서 파일을 저장한다.

https://data.hossam.kr/data/lab07/map_seoul.svg

저장된 파일은 /작업폴더/svg/map_seoul.svg 경로에 저장한다.



1. 지도 이미지 읽어오기

```
map_file_path = "svg/map_seoul.svg"

try:
    with open(map_file_path, 'r', encoding="utf-8") as f:
        map_svg = f.read()
    print("파일 읽어오기 완료")

    # 출력되는 내용이 많으므로 내용 확인 후 주석으로 막아두자.
    #print(map_svg)
except Exception as e:
    print("파일 읽어오기 에러:", e)
```

파일 읽어오기 완료



2. 이미지 확인

```
SVG(map_svg)
```



svg

#03. 데이터 시각화

1. 단계별 색상 팔레트 만들기

단계는 분석가가 임의로 정한다.

색상값을 1단계 ~ 높은단계 순으로 점점 진한 색상이 되도록 구성

사용할 색상값 (단계별로 6개 색상 준비)

```
colors = ['#F1EEF6', '#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77',  
         '#980043']
```



2. BeautifulSoup 객체 생성

svg파일의 내용을 BeautifulSoup객체로 변환

```
soup = BeautifulSoup(map_svg, features="xml")
# 출력되는 내용이 많으므로 내용 확인 후 주석으로 막아두자.
#soup
```



[3] 구 단위로 추출

id속성을 갖는 path 태그 가져오기

```
# soup.select() 메서드의 리턴값은 항상 리스트이다.
# → path 태그이면서 id속성을 갖는 요소를 리스트로 반환
#   <path id="???" ... >
path_list = soup.select('path[id]')
print("가져온 도형의 수: ", len(path_list))
```

가져온 도형의 수: 25



[3] 지도에서 확인한 지역명 수 만큼 반복

```
for p in path_list:
    #print(p)

    지역명 = p['id']
    #print(지역명)

    복지시설수 = origin.query('지역명 == @지역명')['복지시설'].values[0]
    print(지역명, " → ", 복지시설수)

    # 복지시설 수에 따라 단계값 설정 (단계는 색상값의 수에 따른)
    if 복지시설수 > 250:    color_index = 5
    elif 복지시설수 > 200:  color_index = 4
    elif 복지시설수 > 150:  color_index = 3
    elif 복지시설수 > 100:  color_index = 2
    elif 복지시설수 > 50:   color_index = 1
    else:                  color_index = 0

    # svg 이미지의 면 색상 변경
    p['fill'] = colors[color_index]
```

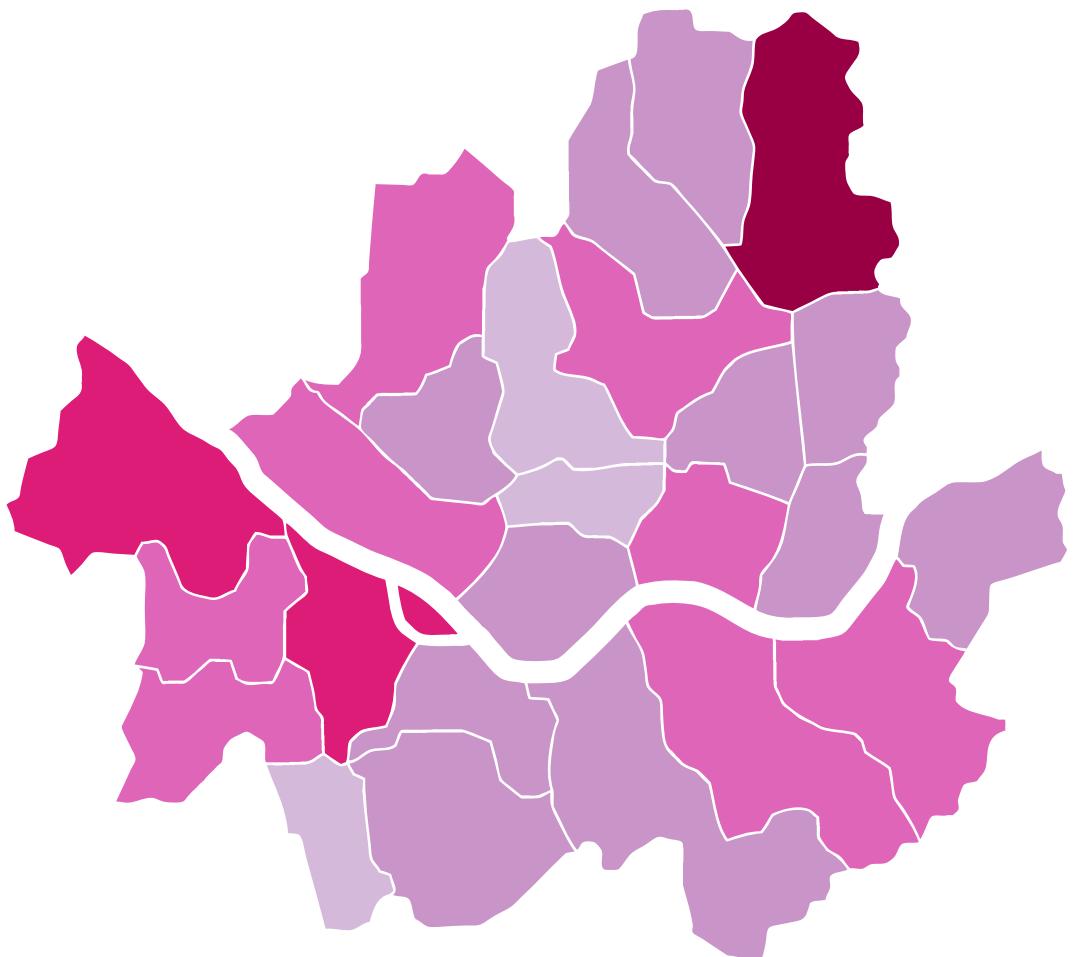
Dobong-gu → 139
Dongdaemun-gu → 146
Dongjak-gu → 143
Eunpyeong-gu → 154
Gangbuk-gu → 111
Gangdong-gu → 140
Gangseo-gu → 215
Geumcheon-gu → 75
Guro-gu → 192
Gwanak-gu → 127
Gwangjin-gu → 103
Gangnam-gu → 184
Jongno-gu → 61
Jung-gu → 53
Jungnang-gu → 128
Mapo-gu → 160
Nowon-gu → 252
Seocho-gu → 129
Seodaemun-gu → 103
Seongbuk-gu → 158
Seongdong-gu → 155
Songpa-gu → 173
Yangcheon-gu → 192
Yeongdeungpo-gu → 208
Yongsan-gu → 110



[4] 재구성된 내용을 토대로 새로운 svg 소스코드 얻기

```
# bs4 객체의 내용을 문자열로 리턴
new_seoul_svg = soup.prettify()

# jupyter에서 svg 이미지 표시하기
# 사용방법 -> SVG(소스문자열) 혹은 SVG(파일경로)
SVG(new_seoul_svg)
```



svg



[5] 생성된 이미지를 파일로 저장해야 하는 경우

```
# 저장된 파일은 윈도우 폴더창에서 직접 더블클릭 해서 웹 브라우저를 통해 확인해야 한다.  
with open('svg/new_seoul_svg.svg', 'w', encoding="utf-8") as f:  
    f.write(new_seoul_svg)
```

QGIS 활용 | 공간 데이터 분석

[LAB-10] GeoCoding

(강남구 지진 해일 대피소 위치 / 서울시 응급실 분포)





학습안내

이번 시간에 학습할 내용과 목표입니다.

학습 내용

1. 구글 스프레드시트 활용
2. 브이월드 Open API 활용

학습 목표

1. 지오코딩에 대해 이해하고 설명할 수 있다.
2. 구글 스프레드시트를 활용하여 지오코딩을 수행할 수 있다.
3. 브이월드 Open API를 활용하여 지오코딩을 수행할 수 있다.

1. 구글 스프레드시트 활용



Geo Coding (지오코딩) 개요

□ 고유명칭(주소나 산, 호수의 이름 등)을 가지고 위도와 경도의 좌표값을 얻는 것

- ◎ 쉐이프파일 등의 GIS 자료를 제공해 주는 기관의 경우 자료를 바로 QGIS 등의 GIS 플랫폼에서 열람하고 분석할 수 있지만, 대부분의 기관이나 업체는 데이터를 GIS 형태가 아닌 주소나 좌표 형태로 제공한다.
- ◎ 주소만 알고 있는 경우 자연어로 된 주소를 좌표로 바꾸는 작업인 지오코딩(geocoding)을 거쳐 쉐이프 파일을 만든 후, 이를 QGIS에서 불러오는 형식으로 분석을 진행할 수 있다.

□ 수업에서 제공되는 “강남구 지진해일대피소.xlsx” 파일의 예시

A	B	C	D	E	F	G	
1	지진해일대피소명	지진해일대피소구분	지진해일대피소유형	지진해일대피소유형구분	소재지도로명주소	수용가능면적	최대수용인원수
2	개원초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 29	12032	3646
3	구룡중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 103	7038	2133
4	개일초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 401	6600	2000
5	양전초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 513	7515	2277
6	개원중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 영동대로 101	10260	3109
7	개포고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 402	8882	2692
8	경기여자고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 삼성로 29	8292	2513
9	수도전기공업고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 410	20800	6303



구글 스프레드시트를 활용한 지오코딩 (1)

□ 구글 스프레드시트에 접속하여 빈 스프레드시트를 생성한다.

◎ <https://docs.google.com/spreadsheets/>

The screenshot shows the Google Sheets interface. At the top, there's a toolbar with various icons. Below it is a navigation bar with a back/forward button, a refresh icon, and a search bar containing the URL <https://docs.google.com/spreadsheets/u/0/>. To the right of the URL are several small icons. The main area is titled "스프레드시트" (Spreadsheet) and features a search bar. On the left, there's a sidebar with the title "새 스프레드시트 시작하기" (Start a new spreadsheet). It displays five template cards: "빈 스프레드시트" (Blank Spreadsheet), "캔트 차트 템플릿" (Canva Chart Template), "할 일 목록" (To-do list), "연간 가계부" (Annual household budget), and "월간 가계부" (Monthly household budget). The first card, "빈 스프레드시트", is highlighted with a red dashed border. At the bottom, there are buttons for "이전 7일" (Last 7 days), "모든 항목" (All items), and "내가 마지막으로 열어본 항목" (Items I last opened), along with sorting and filtering icons.



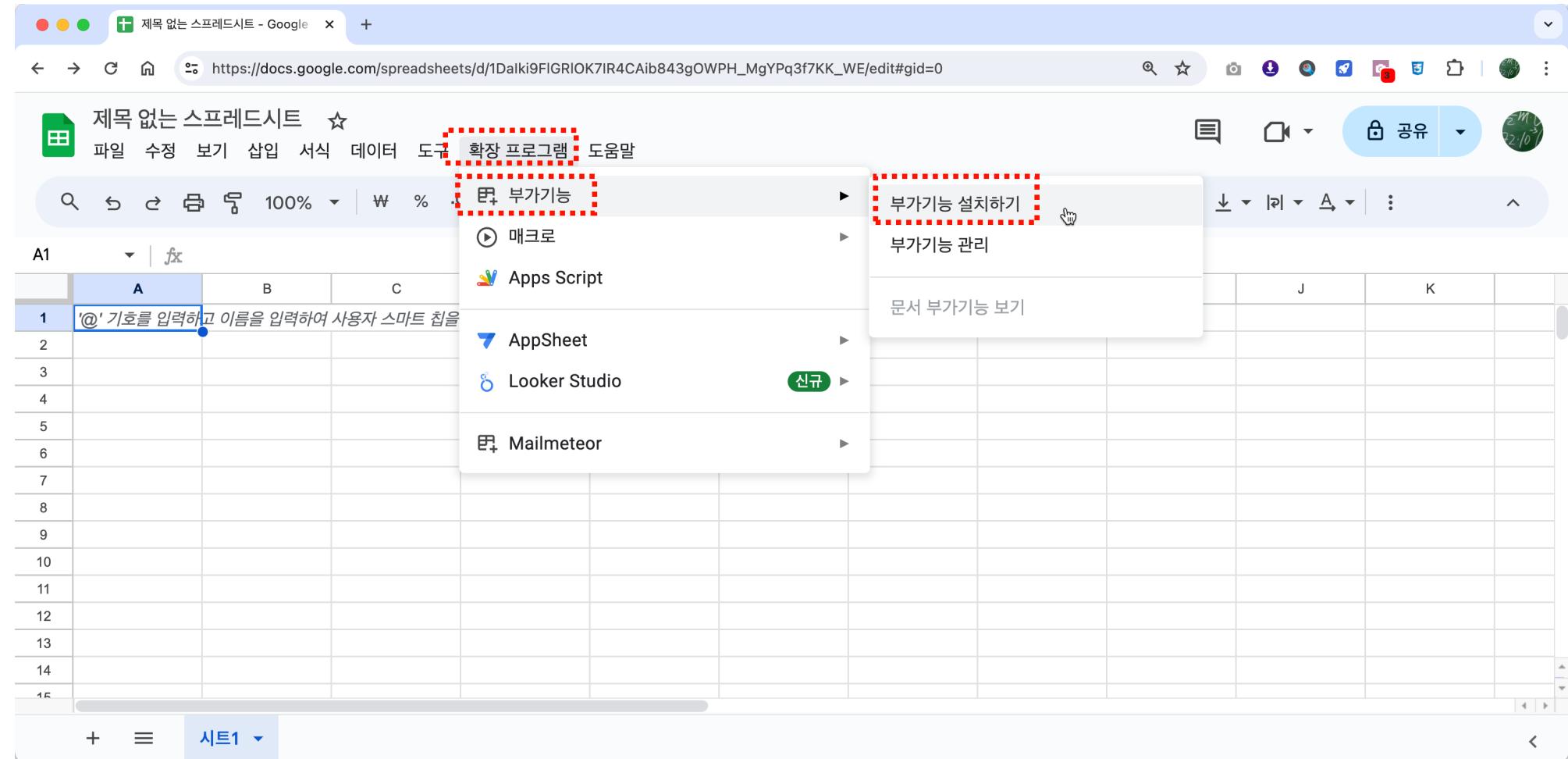
구글 스프레드시트를 활용한 지오코딩 (2)

□ 부가기능 설치하기

◎ “확장 프로그램 → 부가기능 → 부가기능 설치하기” 메뉴를 선택한다.

1. 구글스프레드시트활용

2. 브이월드 API 활용





구글 스프레드시트를 활용한 지오코딩 (3)

□ “Geocode by Awesome Table” 검색

◎ 검색 결과 중에서 설치하고자 하는 추가기능을 클릭한다.

1. 구글스프레드시트활용

2. 브이월드 API 활용

The screenshot shows the Google Sheets Marketplace interface. A search bar at the top right contains the text "Geocode by Awesome Table". A red box with the number "1" highlights this search bar. Below the search bar, a list of add-ons is displayed. The first item in the list, "GeoCode by Awesome Table" by Talarian, is highlighted with a red box and the number "2". This item has a blue icon featuring a map and a location pin. The description below the icon reads: "Geocode is a tool that helps you get latitudes & longitudes from addresses in a Google Sheet t...". The rating is 3.9 stars with 121 reviews. To the right of this are three other add-ons: "Awesome Table" (rating 4.5), "Geocode & Mapping Sheets" (rating 4.8), and "Sheets Mapper" (rating 4.9). The background shows a portion of a Google Sheets document with columns A and K visible.



구글 스프레드시트를 활용한 지오코딩 (4)

□ 프로그램 설치하기

- ◎ “설치” 버튼을 클릭하여 부가기능을 설치한다.
- ◎ 설치 과정 중에서 구글 로그인이 요구될 수 있다.

제목 없는 스프레드시트 - Google

https://docs.google.com/spreadsheets/d/1Dalki9FIGRIOK7IR4CAib843gOWPH_MgYPq3f7KK_WE/edit#gid=0

파일 수첩 보기 산인 서식 데이터 도구 화자 프로그램 도움말

Google Workspace Marketplace

Geocode by Aweso...

Geocode is a tool that helps you get latitudes & longitudes from addresses in a Google Sheet to display them on a map you can share.

개발자: Talarian

정보 업데이트: 2024년 4월 1일

호환 기기: 422 121만+

설치



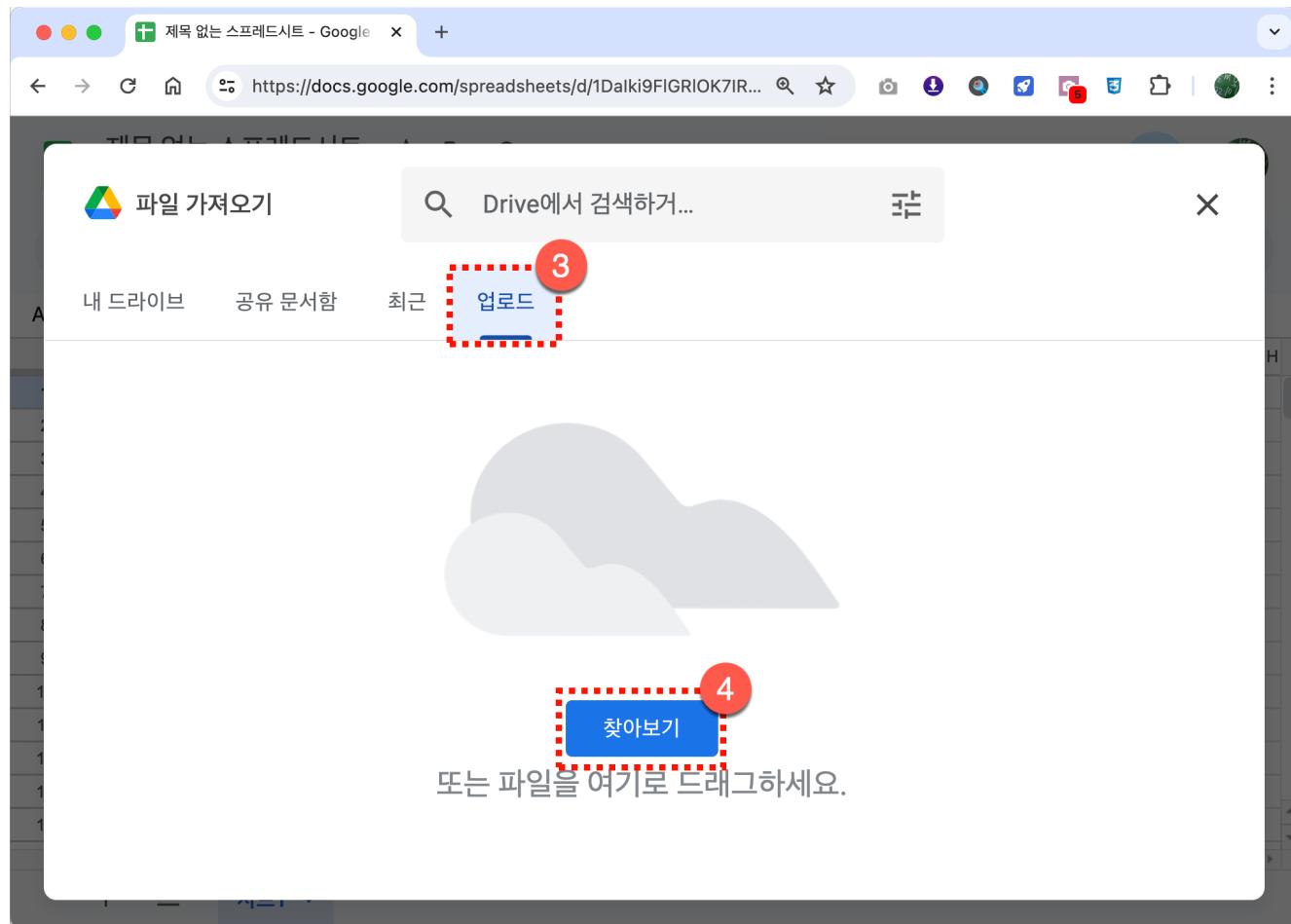
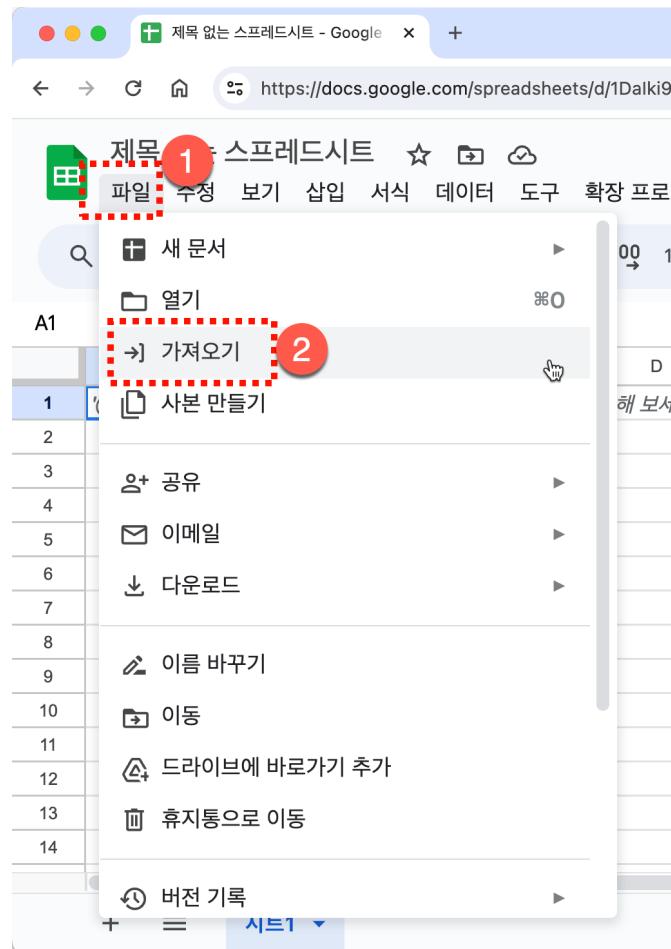
구글 스프레드시트를 활용한 지오코딩 (5)

□ 데이터 파일(엑셀) 업로드하기

◎ “강남구 지진해일대피소.xlsx” 파일을 업로드 한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용



또는 파일을 여기로 드래그하세요.



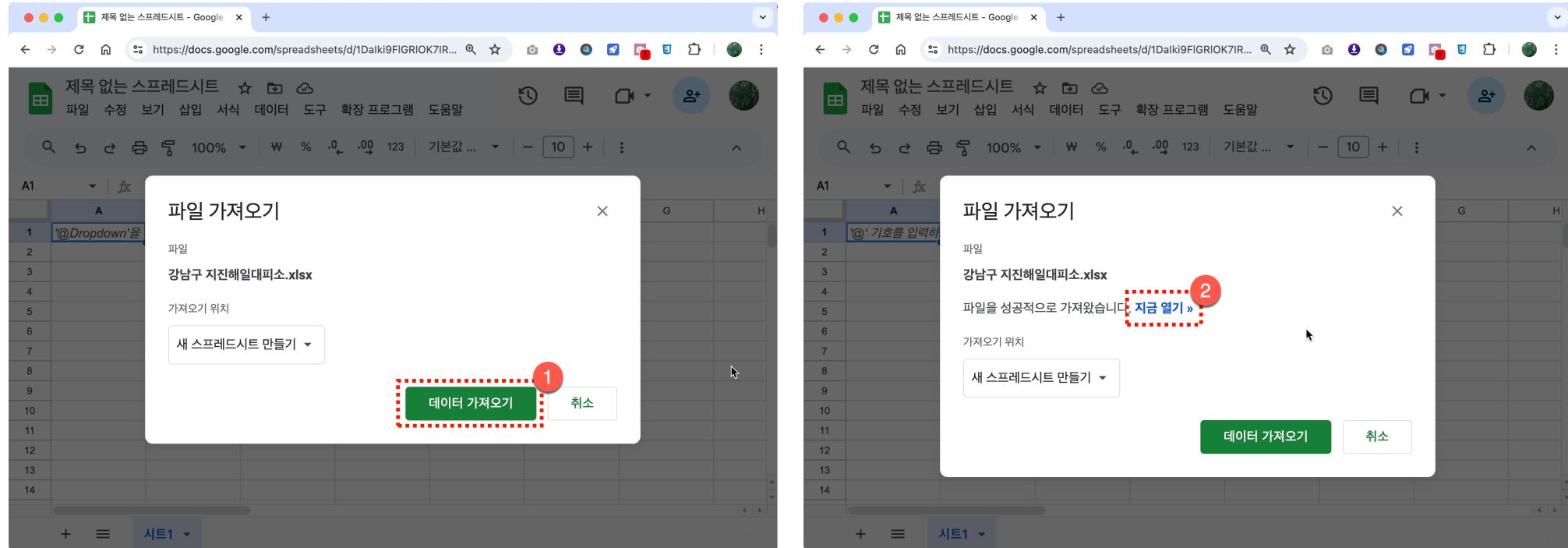
구글 스프레드시트를 활용한 지오코딩 (6)

□ 데이터 가져오기

◎ 업로드가 완료되면 “데이터 가져오기” 버튼을 클릭하여 파일을 연다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용





구글 스프레드시트를 활용한 지오코딩 (7)

□ 지오코딩 실행하기

- ◎ “확장 프로그램 → Geocode by Awesome Table → Start Geocoding” 메뉴를 실행하여 설치한 확장 프로그램을 시작한다.

The screenshot shows a Google Sheets document titled "강남구 지진해일대피소". A context menu is open over a table cell containing "지진대피소명". The menu is titled " 확장 프로그램" and includes options like "부가기능", "매크로", "Apps Script", "AppSheet", "Looker Studio", and two entries under "신규": "Geocode by Awesome Table" and "Mailmeteor". The "Geocode by Awesome Table" option is highlighted with a red dashed box. To its right, another red dashed box highlights the "Start Geocoding" option in a submenu. The main sheet contains a table of evacuation center information, and the sidebar shows a table of locations with their addresses, latitude, and longitude.

	A	B
1	지진해일대피소명	지진해일대피소구
2	개원초등학교 운동장	지진대피소
3	구룡중학교 운동장	지진대피소
4	개일초등학교 운동장	지진대피소
5	양전초등학교 운동장	지진대피소
6	개원중학교 운동장	지진대피소
7	개포고등학교 운동장	지진대피소
8	경기여자고등학교 운동장	지진대피소
9	수도전기공업고등학교 운동장	지진대피소
10	구룡초등학교 운동장	지진대피소
11	포이초등학교 운동장	지진대피소
12	국립국악고등학교 운동장	지진대피소
13	논현초등학교 운동장	지진대피소
14	학동초등학교 운동장	지진대피소
15	언복중학교 운동장	지진대피소
16	대치초등학교 운동장	지진대피소

E	F	G
소재지도로명주소	수용가능면적	최대수용인원수
서울특별시 강남구 선릉로 29	12032	3646
서울특별시 강남구 선릉로 103	7038	2133
서울특별시 강남구 개포로 401	6600	2000
	7515	2277
	10260	3109
	8882	2692
	8292	2513
	20800	6303
	12874	3901
	12279	3721
	1392	422
	4640	1406
	6136	1859
	6444	1953
	13435	4071



구글 스프레드시트를 활용한 지오코딩 (8)

□ 스크롤을 아래로 내려 파라미터 설정 구간으로 이동한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

The screenshot shows a Google Sheets document titled "강남구 지진해일대피소". The spreadsheet contains a table with 16 rows of data, each listing a location name, its category, type, specific category, and address. The first row is selected. A sidebar on the right is titled "Geocode" and contains a "Try new and improved Geocode in Awesome Table Connectors" section. This section includes a yellow banner stating "10k addresses per day limit for Google Workspace accounts", a red note about "구글 계정 하나당 하루 10000건의 처리 제한이 있음", and a list of features like automatic refreshes and data import from various sources. At the bottom of the sidebar, there's a "Try now for FREE" button and a red note "스크롤을 아래로 내린다." (scroll down) with a red dashed arrow pointing downwards.

	A	B	C	D	E
1	지진해일대피소명	지진해일대피소구분	지진해일대피소유형	지진해일대피소유형구분	소재지도로명주소
2	개원초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 29
3	구룡중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 103
4	개일초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 401
5	양전초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 513
6	개원중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 역동대로 101
7	개포고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 103
8	경기여자고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 삼성로 29
9	수도전기공업고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 410
10	구룡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 263
11	포이초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 22길 87
12	국립국악고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 22길 65
13	논현초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 강남대로 120길
14	학동초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 115길 42
15	언복중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 도산대로 38길 2
16	대치초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 양재천로 363

Geocode gets latitudes and longitudes from

+ Sheet1 ▾



구글 스프레드시트를 활용한 지오코딩 (9)

□ 파라미터 설정하기

◎ 시트 이름과 주소가 기입된 필드를 지정한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

	A	B	C	D	E
1	지진해일대피소명	지진해일대피소구분	지진해일대피소유형	지진해일대피소유형구분	소재지도로명주소
2	개원초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 29
3	구룡중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 103
4	개일초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 401
5	양전초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 513
6	개원중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 영동대로 101
7	개포고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 402
8	경기여자고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 삼성로 29
9	수도전기공업고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 410
10	구룡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 263
11	포이초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로22길 87
12	국립국악고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로22길 65
13	논현초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 강남대로120길
14	학동초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로115길 42
15	연북중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 도산대로38길
16	대치초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 양재천로 363



구글 스프레드시트를 활용한 지오코딩 (10)

□ 지오코딩 실행 확인

◎ 경도(Latitude), 위도(Longitude) 필드가 자동으로 추가되면서 지오코딩이 수행된다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

The screenshot shows a Google Sheets document titled "강남구 지진해일대피소". A sidebar titled "Geocode" is open, displaying progress: "45 out of 74 addresses". The main table has columns: 유형구분, 소재지도로명주소, Latitude, Longitude, 수용가능면적, 최대수용인원수. The "Latitude" and "Longitude" columns are highlighted with red dashed boxes, indicating they were automatically added during the geocoding process.

유형구분	소재지도로명주소	Latitude	Longitude	수용가능면적	최대수용인원수
1	서울특별시 강남구 선릉로 29	37.4813655	127.0590553	12032	3646
2	서울특별시 강남구 선릉로 103	37.4858999	127.0564272	7038	2133
3	서울특별시 강남구 개포로 401	37.4859589	127.0580194	6600	2000
4	서울특별시 강남구 개포로 513	37.4904255	127.07003	7515	2277
5	서울특별시 강남구 영동대로 101	37.4913769	127.0714248	10260	3109
6	서울특별시 강남구 개포로 402	37.484898	127.0591661	8882	2692
7	서울특별시 강남구 삼성로 29	37.4867111	127.0659079	8292	2513
8	서울특별시 강남구 개포로 410	37.4858037	127.0634758	20800	6303
9	서울특별시 강남구 개포로 263	37.4807968	127.0519292	12874	3901
10	서울특별시 강남구 개포로22길 87	37.4755995	127.0523238	12279	3721
11	서울특별시 강남구 개포로22길 65	37.4761552	127.0513959	1392	422
12	서울특별시 강남구 강남대로120길 33	37.5082167	127.0263166	4640	1406
13	서울특별시 강남구 선릉로115길 42	37.5120662	127.0398114	6136	1859
14	서울특별시 강남구 도산대로38길 27	37.5191189	127.0332439	6444	1953
15	서울특별시 강남구 양재천로 363	37.4909106	127.0622675	13435	4071
16					



구글 스프레드시트를 활용한 지오코딩 (11)

□ 처리 결과 다운로드

◎ “파일 → 다운로드 → Microsoft Excel(.xlsx)” 메뉴를 클릭하여 파일을 내려받는다.

1. 구글스프레드시트활용

2. 브이월드 API 활용

The screenshot shows a Google Sheets document titled "강남구 지진해일대피소". A sidebar on the right is titled "Geocode" and contains a section for "Awesome Table Connectors" with a "Try now for FREE" button. The main sheet displays a table of data with columns F, G, H, I, J, and K. The first row is a header: F (Latitude), G (Longitude), H (수용가능면적), I (최대수용인원수). Below the table, a dropdown menu is open under the "File" tab, showing options like "새 문서", "열기", "가져오기", "사본 만들기", "공유", "이메일", "다운로드", "이름 바꾸기", "이동", "드라이브에 바로가기 추가", "휴지통으로 이동", "버전 기록", and "오프라인 사용 설정". The "다운로드" option is highlighted with a red box. A sub-menu for "Microsoft Excel (.xlsx)" is also highlighted with a red box.

F	G	H	I	J	K
Latitude	Longitude	수용가능면적	최대수용인원수		
37.4813655	127.0590553	12032	3646		
37.4858999	127.0564272	7038	2133		
37.4859589	127.0580194	6600	2000		
37.4904255	127.07003	7515	2277		
		10260	3109		
		8882	2692		
		8292	2513		
		20800	6303		
		12874	3901		
		12279	3721		
		1392	422		
		4640	1406		
		6136	1859		
		37.5191189	127.0332439	6444	1953
		37.4909106	127.0622675	13435	4071



구글 스프레드시트를 활용한 지오코딩 - 결과 확인

□ 내려 받은 엑셀 파일 확인

◎ 경, 위도 값이 추가되어 있음을 확인할 수 있다.

1. 구글스프레드시트 활용

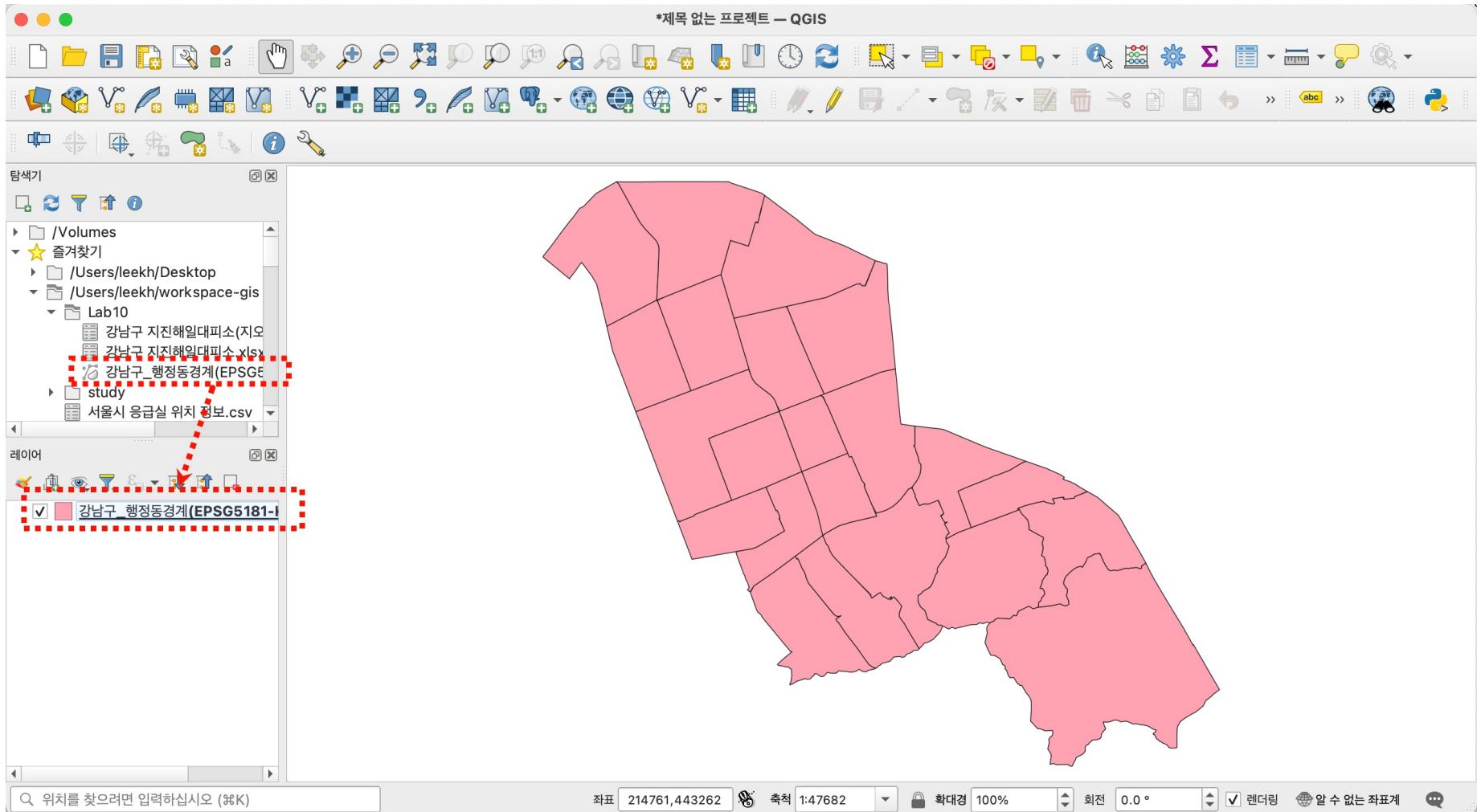
2. 브이월드 API 활용

	A	B	C	D	E	F	G	H	I
1	지진해일대피소명	지진해일대피소구분	지진해일대피소유형	지진해일대피소유형구분	소재지도로명주소	Latitude	Longitude	수용가능면적	최대수용인원수
2	개원초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 29	37.4813655	127.0590553	12032	3646
3	구룡중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 103	37.4858999	127.0564272	7038	2133
4	개일초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 401	37.4859589	127.0580194	6600	2000
5	양천초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 513	37.4904255	127.07003	7515	2277
6	개원중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 영동대로 101	37.4913769	127.0714248	10260	3109
7	개포고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 402	37.484898	127.0591661	8882	2692
8	경기여자고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 삼성로 29	37.4867111	127.0659079	8292	2513
9	수도전기공업고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 410	37.4858037	127.0634758	20800	6303
10	구룡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 263	37.4807968	127.0519292	12874	3901
11	포이초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로22길 87	37.4755995	127.0523238	12279	3721
12	국립국악고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로22길 65	37.4761552	127.0513959	1392	422
13	논현초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 강남대로120길 33	37.5082167	127.0263166	4640	1406
14	학동초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로115길 42	37.5120662	127.0398114	6136	1859
15	언북중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 도산대로38길 27	37.5191189	127.0332439	6444	1953
16	대치초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 양재천로 363	37.4909106	127.0622675	13435	4071
17	단대부속고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 도곡로64길 21	37.496529	127.0564676	7800	2364
18	대청중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 양재천로 321	37.4901314	127.0585502	8336	2526
19	대곡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 남부순환로 3022	37.4944062	127.0651765	7805	2365
20	대현초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 역삼로98길 16	37.5035728	127.0638597	8585	2602
21	대명중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 역삼로87길 26	37.5060559	127.061066	5400	1636
22	휘문중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 역삼로 541	37.505133	127.0621203	32455	9835
23	도곡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로64길 33	37.4990608	127.0546706	8080	2448



경,위도 좌표를 Point Layer로 추가하기 (1)

- 강남구 행정동 경계도를 작업 레이어에 추가
 - ◎ “강남구_행정동경계(EPSG5181-KGD2002,UTF8).shp” 파일을 추가한다.





경,위도 좌표를 Point Layer로 추가하기 (2)

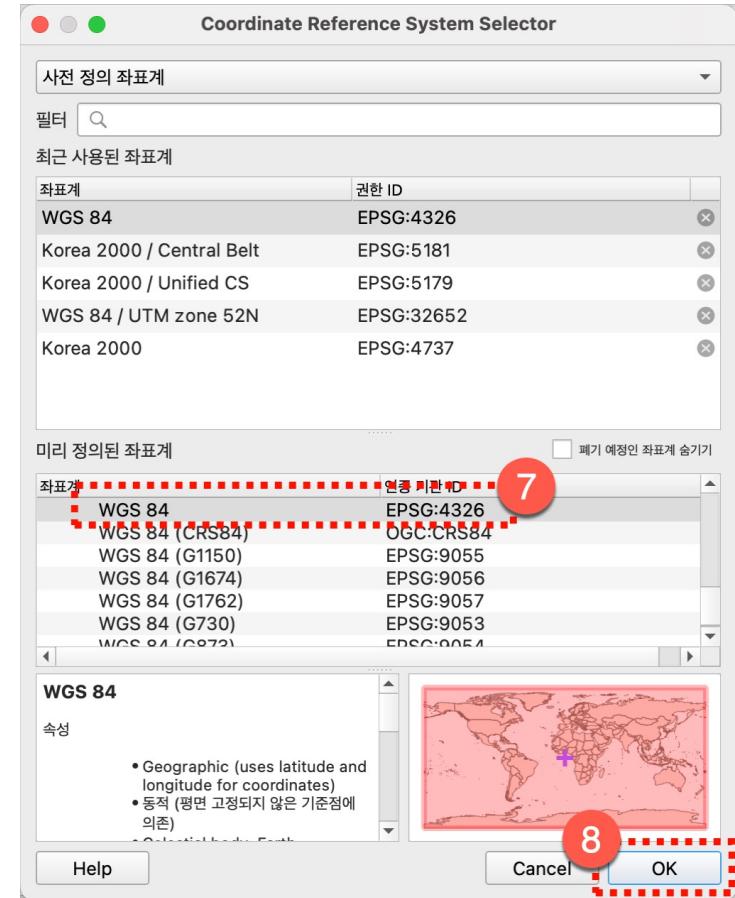
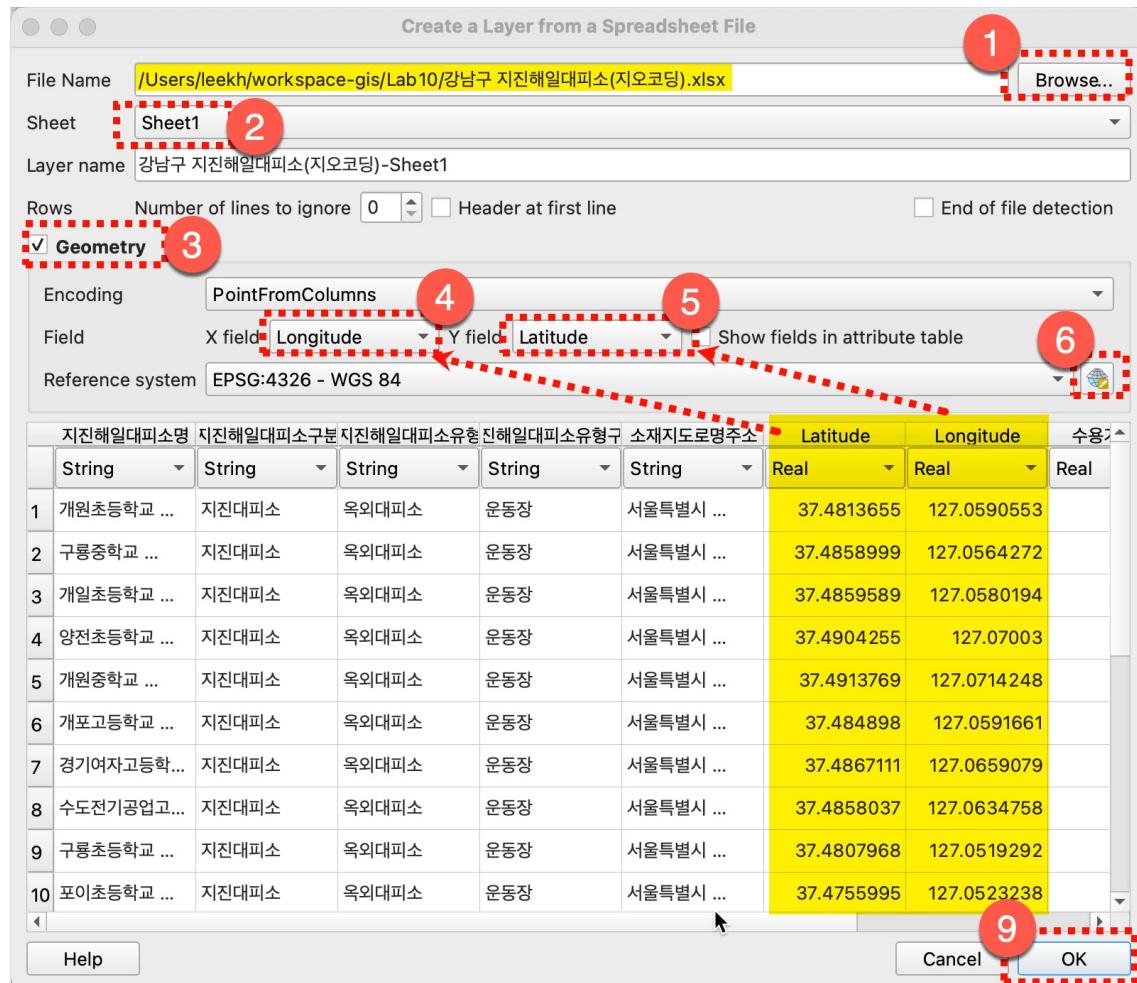
□ 지오코딩 결과가 포함된 엑셀 파일을 Point 레이어로 추가

◎ 툴바에서 “Add Spreadsheet Layer” 버튼을 클릭한다.



Add Spreadsheet Layer...

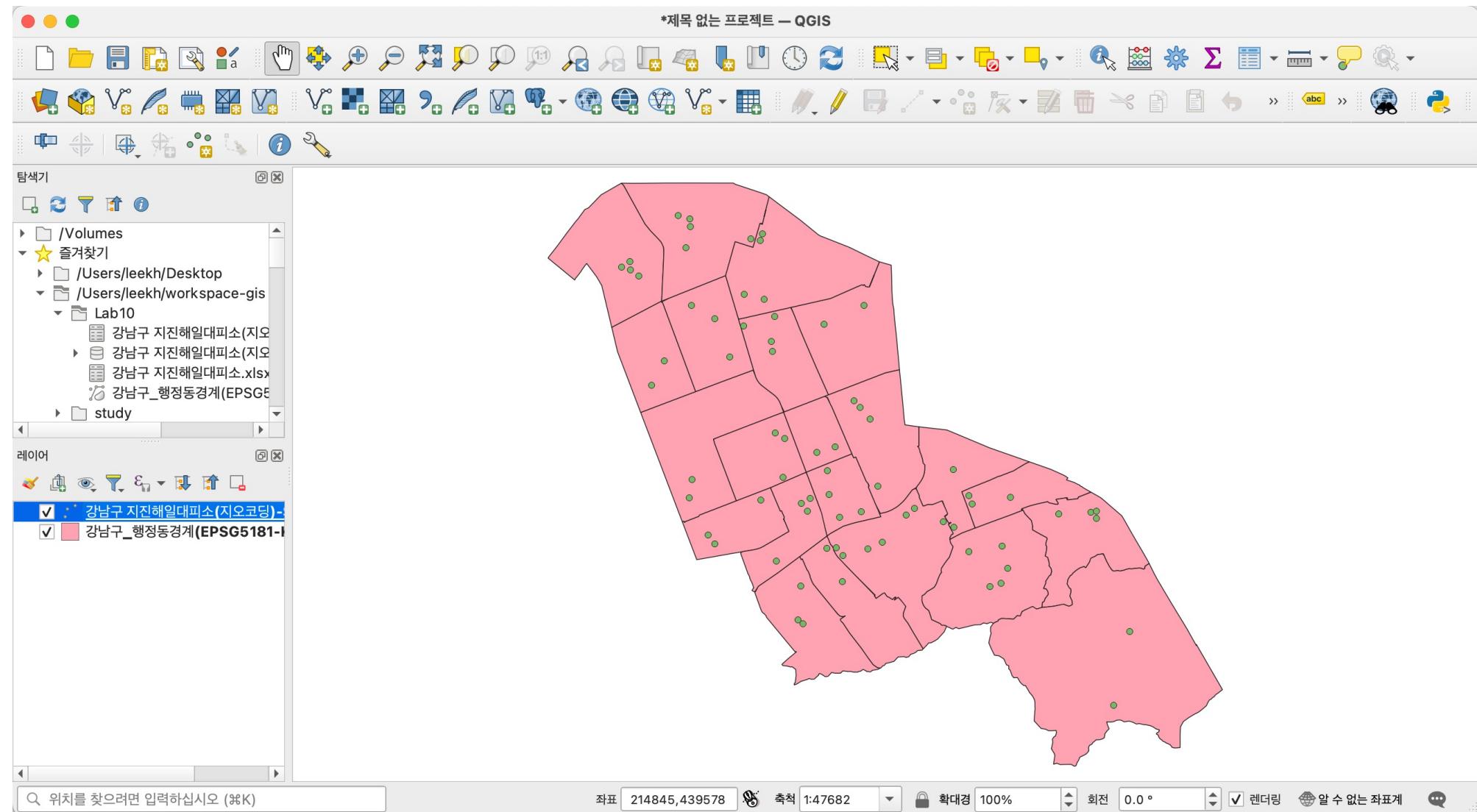
◎ 경위도 기반이므로 좌표계는 “EPSG 4326 - WGS84”를 지정해야 한다.





경,위도 좌표를 Point Layer로 추가하기 (3)

□ 지오코딩 처리된 좌표에 대한 포인트 확인

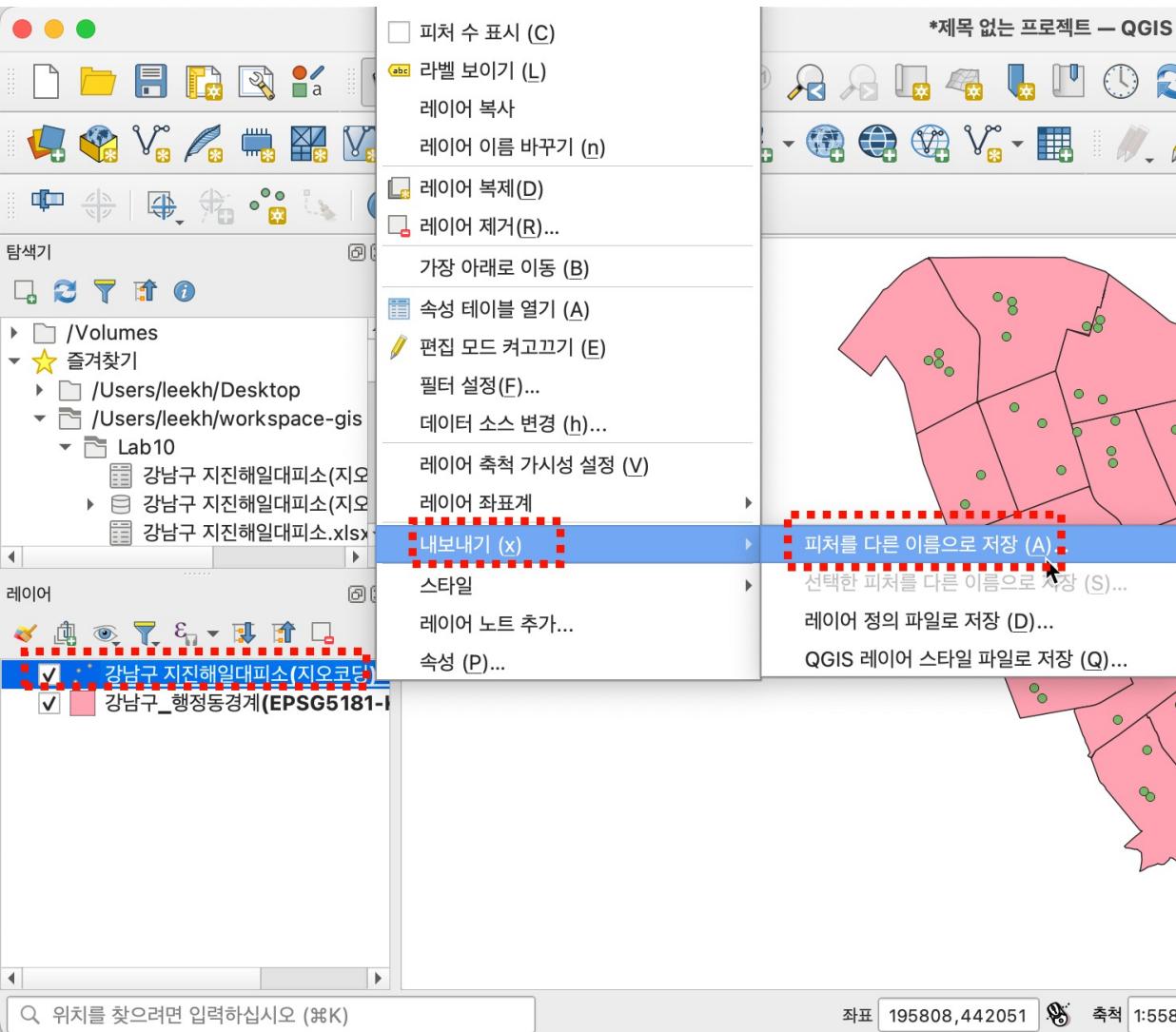




경,위도 좌표를 Point Layer로 추가하기 (4)

□ 포인트 레이어를 shape 파일로 저장하기

- ◎ 엑셀 파일을 Point Layer로 변경한 상태에서 shape 파일로 저장하면 작업시마다 변환해야 하는 번거로움을 피할 수 있다.



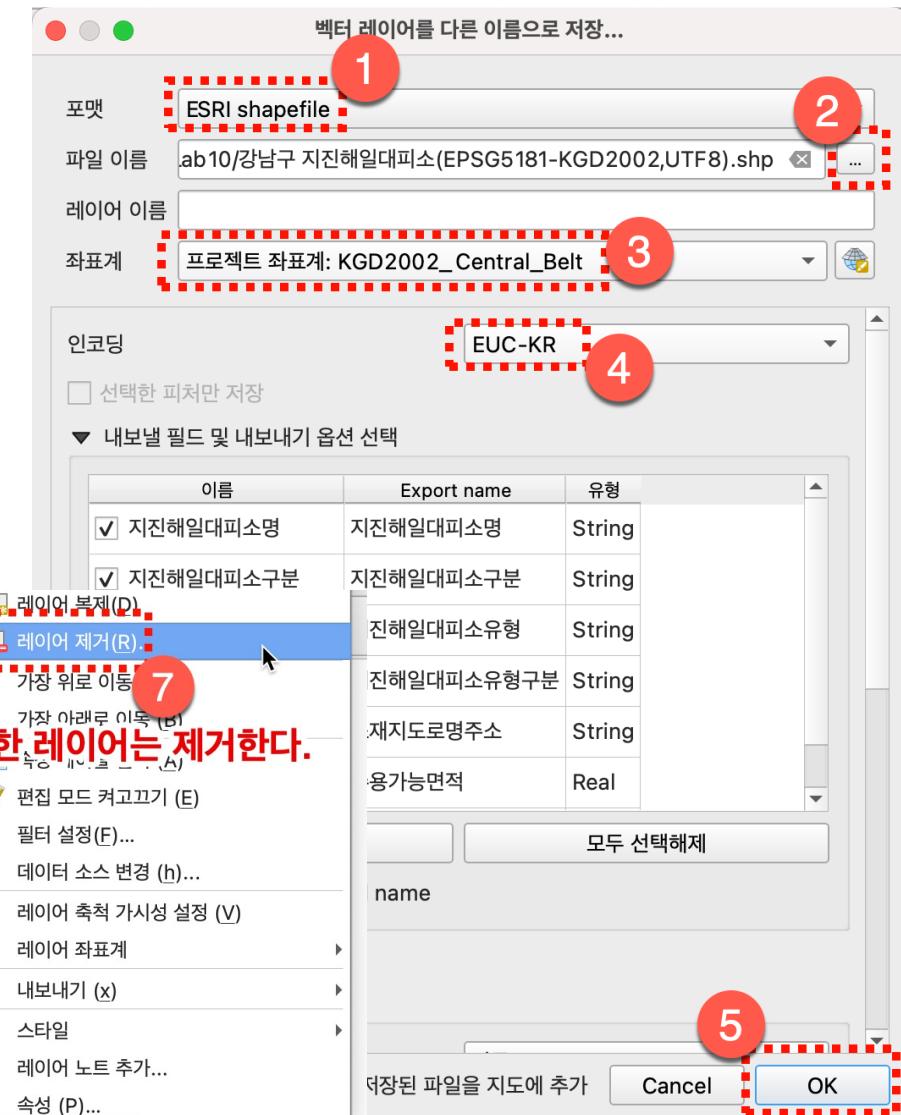
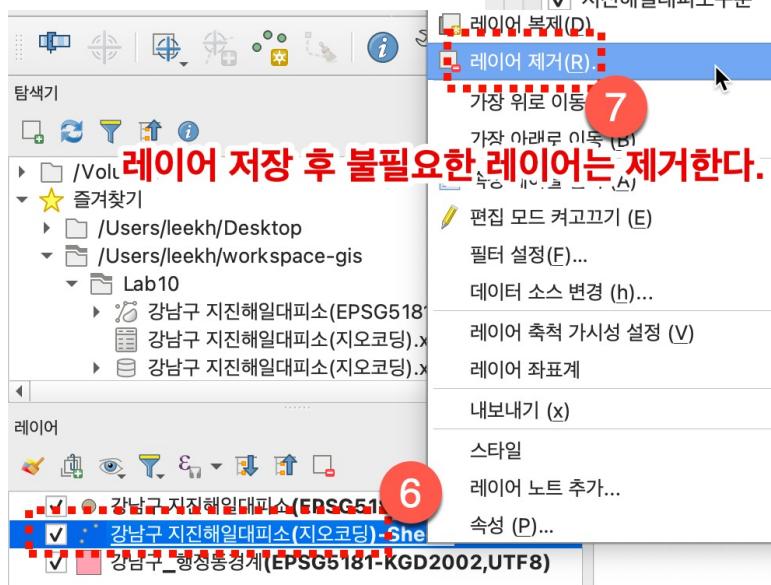


경,위도 좌표를 Point Layer로 추가하기 (5)

□ 저장 파라미터 설정

1. 구글스프레드시트 활용
2. 브이월드 API 활용

- ① 저장 포맷을 shape 파일로 설정
- ② 저장 경로 지정
- ③ 좌표계를 변경하고자 할 경우 설정
 - 여기서는 프로젝트 좌표계와 일치시킴
- ④ 한글 필드명이 있으므로 인코딩을 EUC-KR로 지정
 - UTF-8은 한글 필드명이 깨진다(QGIS 버그)
- ⑤ 설정 내용 실행





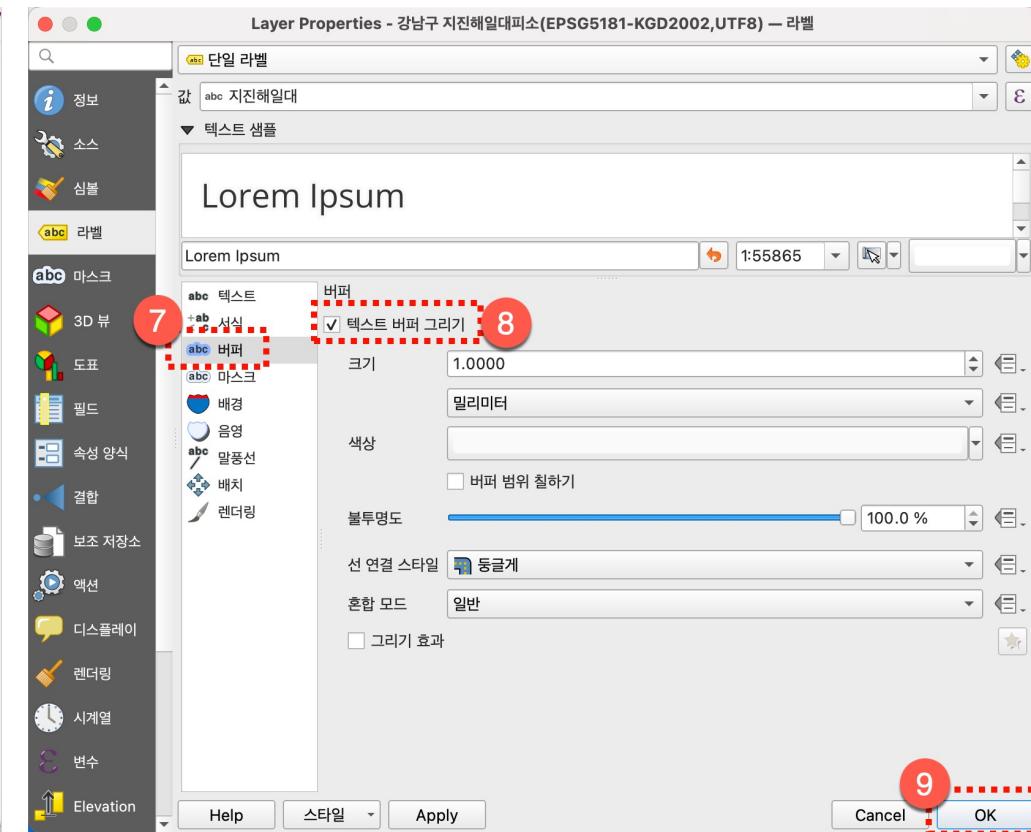
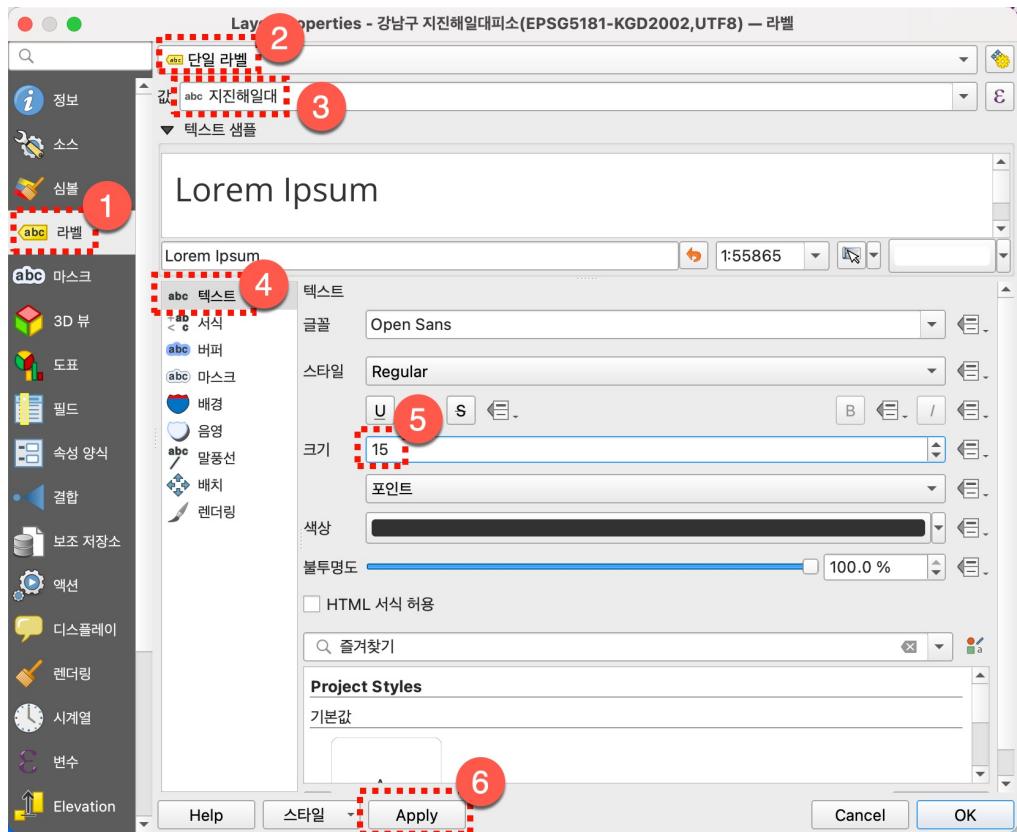
경,위도 좌표를 Point Layer로 추가하기 (6)

□ Point Layer의 라벨 속성 설정

◎ 대피소 명칭을 표시할 수 있도록 속성을 설정한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용



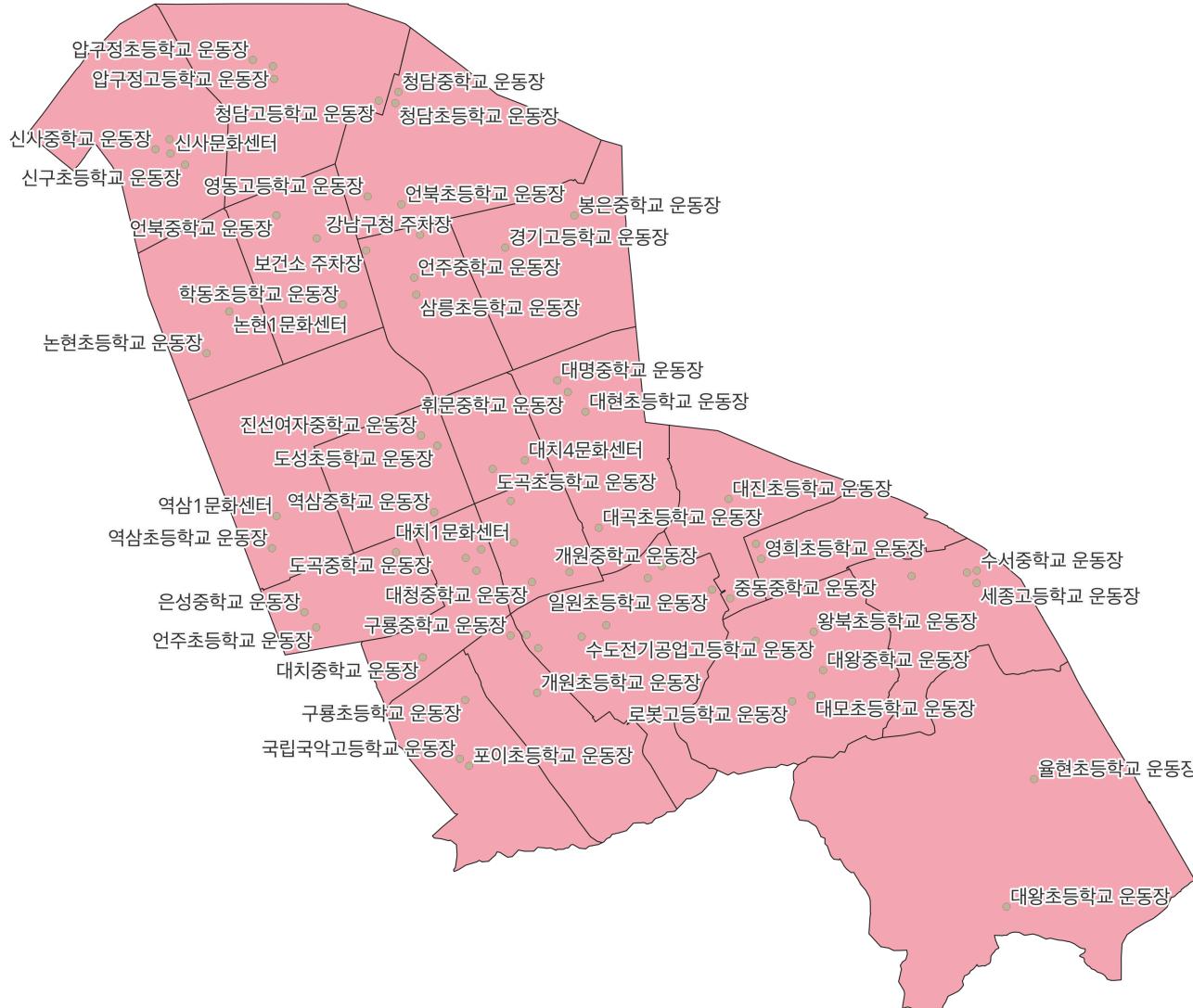


경,위도 좌표를 Point Layer로 추가하기 - 결과 확인

□ 강남구 지진해일 대피소에 대한 지오코딩 결과

1. 구글스프레드시트 활용

2. 브이월드 API 활용



1. 구글스프레드시트 활용

2. 브이월드 API 활용



실습하기

구글 스프레드시트 활용



2. 브이월드 Open API 활용



Open API의 이해

□ API와 Open API

API

- Application Programming Interface
- 응용 프로그램에서 데이터를 주고 받기 위한 방법

Open API

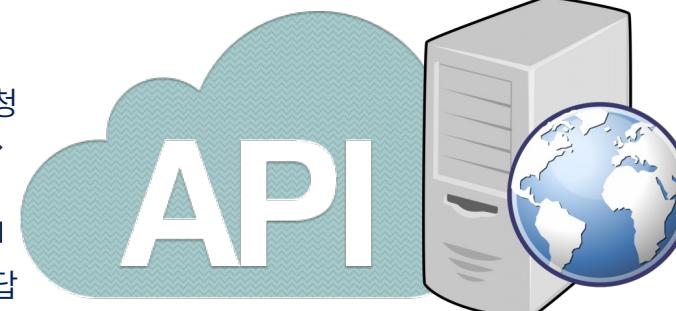
- 누구나 사용할 수 있도록 공개된 API
- 주로 공공 데이터에 대한 제공을 목적으로 관공서, 포털 등이 구축한다.
- 다양한 종류의 프로그램과 연동하기 유리하도록 웹 기반으로 구축한다.



CLIENT

여러가지 유형의 프로그램
Python, Java, C, JS 등으로 제작 가능

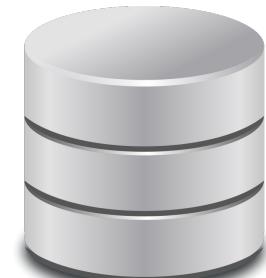
데이터 요청
데이터 응답



API

주로 웹 형태의 시스템으로 이루어짐

데이터 요청
데이터 응답



DATABASE

입력, 수정, 삭제, 조회
SQL 활용

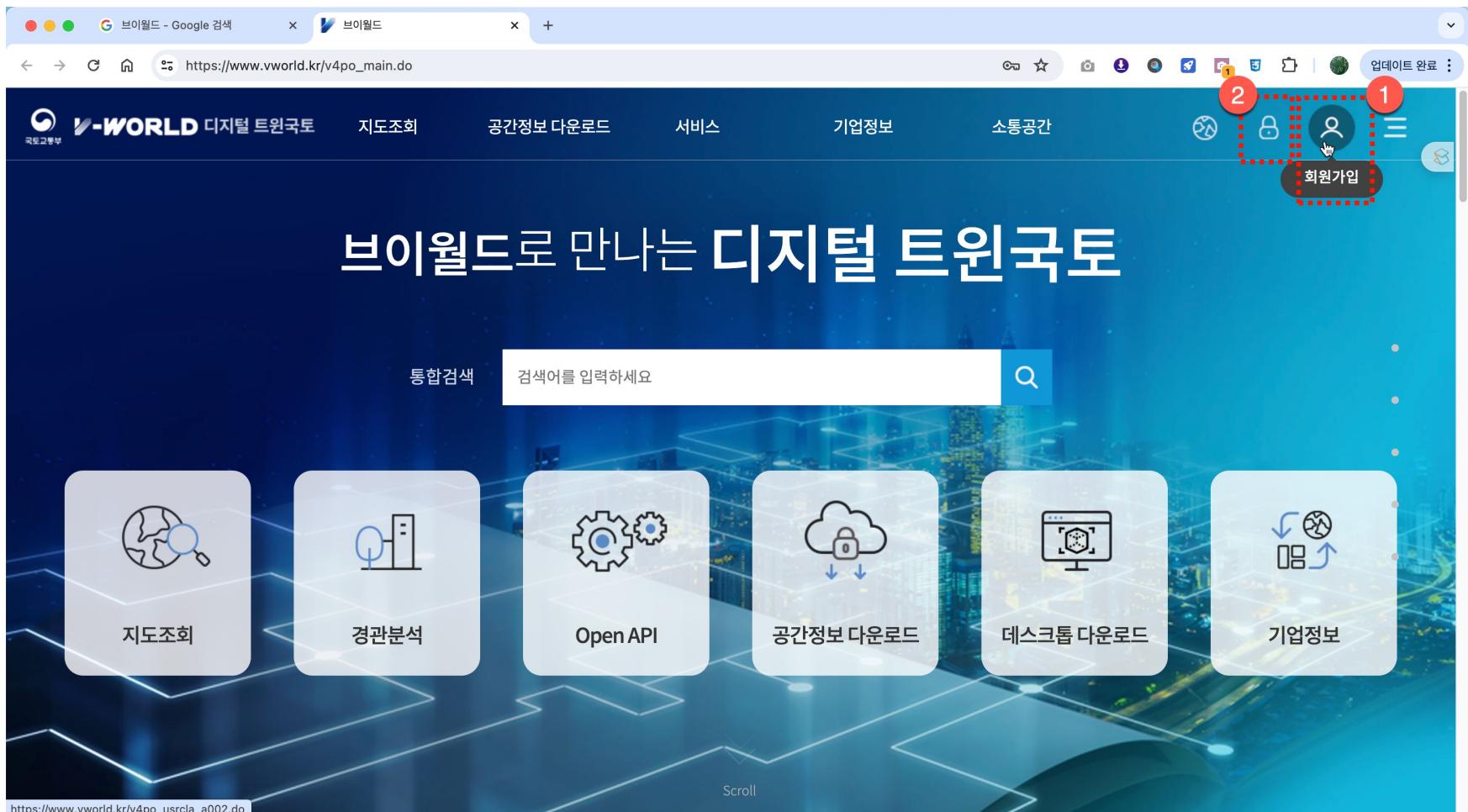
1. 구글스프레드시트 활용

2. 브이월드 API 활용



브이월드 Open API 인증키 발급받기 (1)

- 브이월드(https://www.vworld.kr/v4po_main.do) 회원가입 및 로그인
 - ◎ OpenAPI 연동을 위해서는 서비스를 제공하는 웹 사이트의 회원임을 인증하기 위한 인증키를 발급받아야 한다.



1. 구글스프레드시트 활용

2. 브이월드 API 활용



브이월드 Open API 인증키 발급받기 (2)

□ 인증키 발급을 위한 페이지 이동

The screenshot shows the V-World main page at https://www.vworld.kr/v4po_main.do. A red box labeled '1' highlights the '서비스' (Services) menu item. A red box labeled '2' highlights the '인증키' (Authentication Key) link under the 'Open API' section of the dropdown menu.

Branding: V-WORLD 디지털 트윈국토

Header: 브이월드

Header: https://www.vworld.kr/v4po_main.do

Header: 09:56 연장

Menu: 서비스 (1)

Sub-menu: Open API (2)

- 지도API
- 국가주체
- 인증키 (highlighted)
- 활용사례
- 오픈API 활용모델
- API인큐베이터

Sub-menu: 3D데스크톱

- 다운로드
- 사용방법안내

Sub-menu: 부동산 서비스

- 부동산개발업
- 부동산중개업
- 내토지찾기
- 조상땅찾기

Bottom: javascript:showLoginForm('/dev/v4dv_apikey_s001.do');

Bottom: Scroll

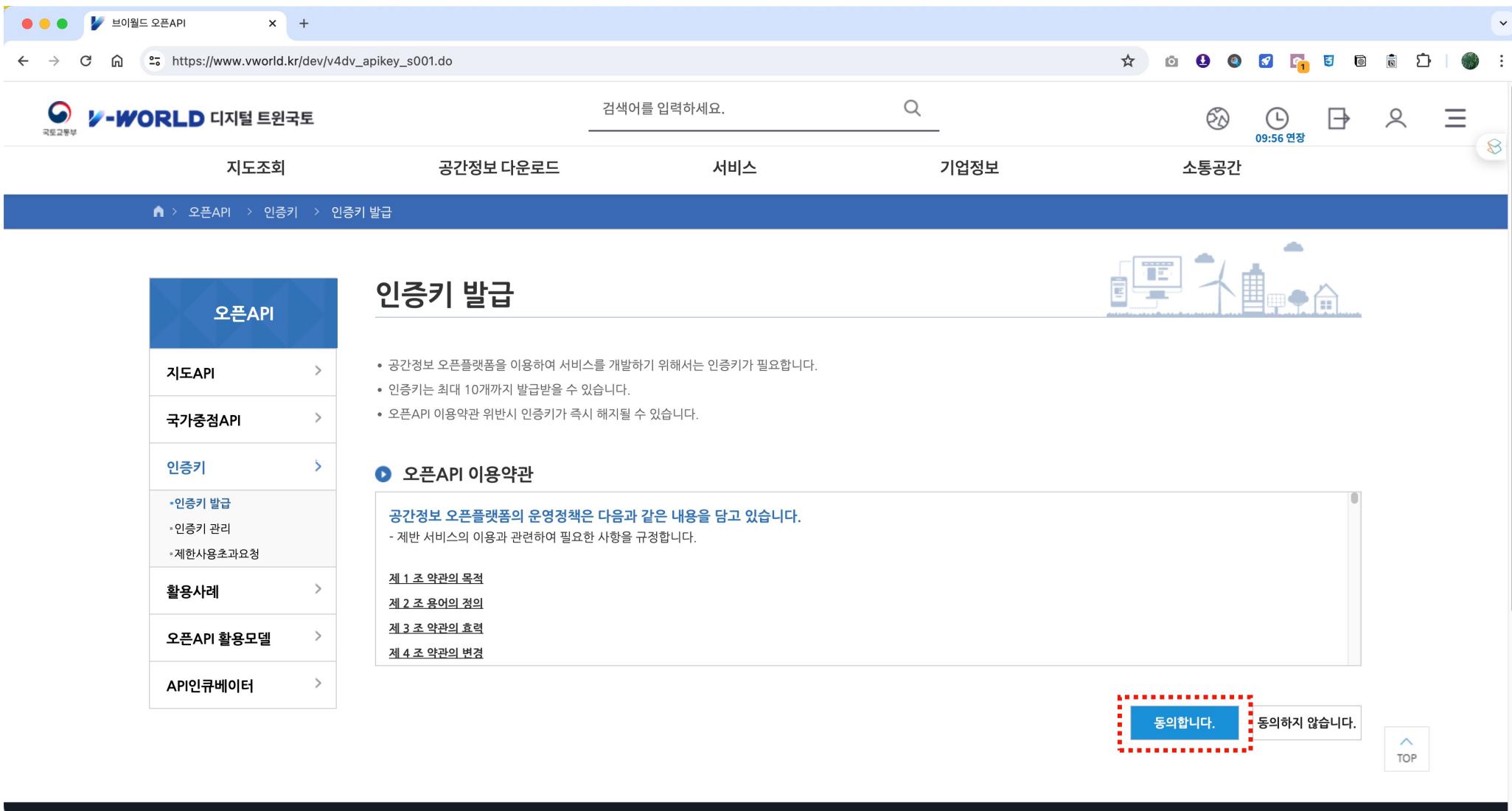
Bottom: ITWILL 이광호강사

1. 구글스프레드시트 활용

2. 브이월드 API 활용

브이월드 Open API 인증키 발급받기 (3)

□ 인증키 발급 약관 동의하기



The screenshot shows the V-World Open API landing page. On the left, a sidebar menu lists various API categories. The main content area is titled '인증키 발급' (API Key Issuance) and contains a summary of the requirements for using spatial information services. At the bottom right, there are two buttons: '동의합니다.' (Agree) and '동의하지 않습니다.' (Do not agree), with the 'Agree' button being highlighted by a red dashed box.

브이월드 오픈API

https://www.vworld.kr/dev/v4dv_apikey_s001.do

V-WORLD 디지털 트윈국토

지도조회 공간정보 다운로드 서비스 기업정보 소통공간

인증키 발급

• 공간정보 오픈플랫폼을 이용하여 서비스를 개발하기 위해서는 인증키가 필요합니다.
 • 인증기는 최대 10개까지 발급받을 수 있습니다.
 • 오픈API 이용약관 위반시 인증키가 즉시 해지될 수 있습니다.

● 오픈API 이용약관

제 1 조 약관의 목적
 제 2 조 용어의 정의
 제 3 조 약관의 효력
 제 4 조 약관의 변경

동의합니다. 동의하지 않습니다.

TOP

1. 구글스프레드시트 활용

2. 브이월드 API 활용

브이월드 Open API 인증키 발급받기 (4)

□ 인증키 발급 신청 양식 작성 – 필요 목적에 맞게 양식을 작성한다.

The screenshot shows the '인증키 발급' (API Key Application) page on the V-World website. A sidebar on the left lists various API categories. The main form has several fields and checkboxes highlighted with red boxes and numbers:

- 서비스명 ***: '개인학습용' (highlighted by a red box with number 1)
- 서비스분류 ***: '교육' (highlighted by a red box with number 2)
- 서비스유형 ***: '기타(QGIS, ArcGIS 등)' (highlighted by a red box with number 3)
- 서비스URL***: A text input field containing 'QGIS 학습용 키 발급' (highlighted by a red box with number 4)
- 서비스 설명 ***: A text area with placeholder text and character count (13 / 3000자) (highlighted by a red box with number 5)
- 브이월드 활용API ***: A group of checkboxes including '2D 지도 API' (unchecked), '2D 데이터 API' (unchecked), '2D 모바일 API' (unchecked), '지오코더 API' (checked), '배경지도 API' (unchecked), '검색 API' (unchecked), '이미지 API' (unchecked), '3D 모바일 API' (unchecked), '3D데스크톱 API' (unchecked), 'WMS/WFS API' (unchecked), 'WMTS/TMS API' (unchecked), and '국가중점 API' (unchecked). (highlighted by a red box with number 6)
- 사용기관 ***: Radio buttons for '민간' (checked), '공공' (unchecked), and '개인' (unchecked). (highlighted by a red box with number 7)
- Submit Button**: A button labeled '지도 인증키 발급' (highlighted by a red box with number 8)

1. 구글스프레드시트 활용

2. 브이월드 API 활용



브이월드 Open API 인증키 발급받기 (5)

□ 발급받은 키 보관하기

◎ 이 값을 OpenAPI 연동시 설정해야 하므로 별도로 복사해서 관리한다.

The screenshot shows a web browser window for the V-World API management system. The URL is https://www.vworld.kr/dev/v4dv_apikey_s002.do. The page title is "브이월드 오픈API". The main content area is titled "인증키 관리" (API Key Management). On the left, there is a sidebar menu under "오픈API" with options like "지도API", "국가중점API", "인증키" (which is expanded to show "인증키 발급" and "인증키 관리"), "활용사례", "오픈API 활용모델", and "API인큐베이터". The main content area contains a list of issued API keys, with one entry highlighted by a red dashed box. The highlighted entry shows the key value "개인학습용" (Personal Learning), its status "개발" (Development), and its creation date "2024-12-02". A red text overlay at the bottom right of the highlighted row says "이 값을 복사해서 보관한다." (Copy this value and save it).

서비스명	인증키	상태	만료일
개인학습용	[Red box]	개발	2024-12-02



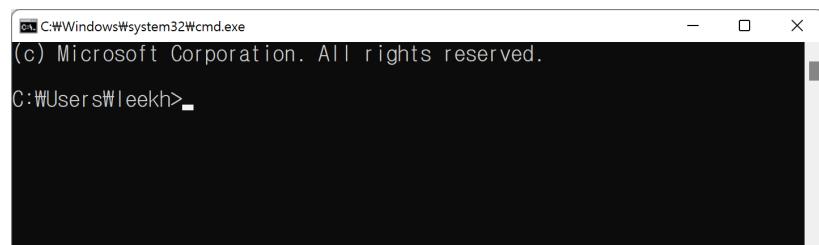
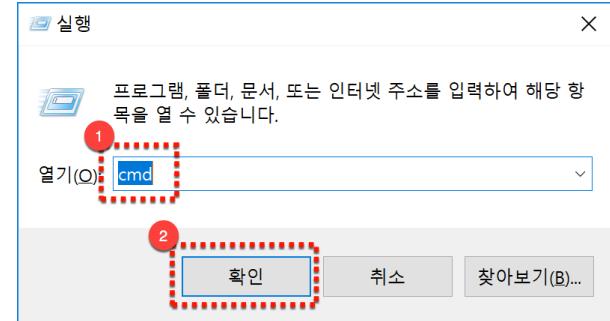
CLI 환경 실행하기

□ CLI (Command Line Interface) - 명령어 라인 인터페이스

- ◎ 사용자와 시스템의 상호작용을 위한 방식이 명령어 입력과 결과 메시지 출력으로 이루어지는 컴퓨팅 환경
- ◎ Windows는 명령프롬프트, Mac은 터미널이라는 프로그램으로 CLI 환경을 사용할 수 있다.

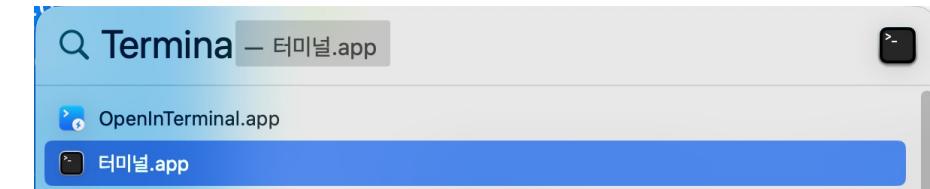
□ Window 환경 – 명령프롬프트 실행하기

WinKey + R > cmd (엔터)



□ Mac 환경 – 터미널 실행하기

⌘ + Space > terminal (엔터)





파이썬 설치 여부 확인하기 (1) - Window의 경우

□ 명령프롬프트에서 파이썬 버전 확인 명령어 수행하기

```
python --version
```

이 수업에서 파이썬을 직접적으로 코딩하는 것은 아니지만 사용하려는 지오코더 프로그램이 파이썬을 기반으로 하기 때문에 파이썬 환경 구성이 필요하다.

파이썬이 설치되어 있는 경우 (3.x 버전이면 가능, 3.11 이상 권장)

```
C:\Windows\system32\cmd + 
Microsoft Windows [Version 10.0.22000.2960]
(c) Microsoft Corporation. All rights reserved.

C:\Users\leekh>python --version
Python 3.11.2

C:\Users\leekh>
```

파이썬이 설치되어 있지 않은 경우 → 설치가 필요함

```
C:\Windows\system32\cmd + 
Microsoft Windows [Version 10.0.22000.2960]
(c) Microsoft Corporation. All rights reserved.

C:\Users\leekh>python --version
'python'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

C:\Users\leekh>
```

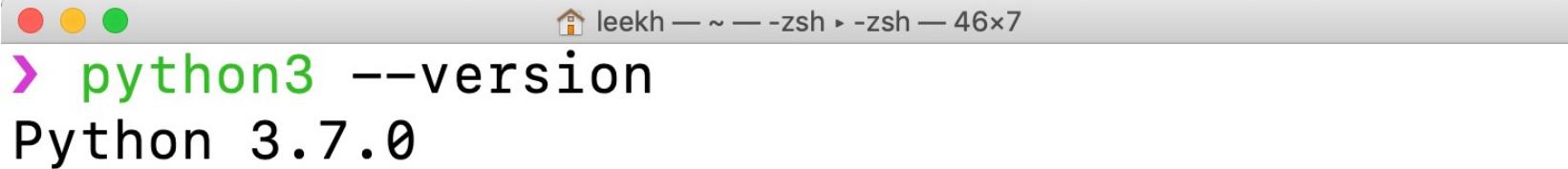


파이썬 설치 여부 확인하기 (2) - Mac의 경우

- 터미널에서 파이썬 버전 명령어 수행하기

```
python3 --version
```

파이썬이 설치되어 있는 경우



```
leekh ~ -- zsh -zsh -46x7
> python3 --version
Python 3.7.0
```

파이썬이 설치되어 있지 않은 경우 → 설치가 필요함



```
leekh ~ -- zsh -zsh -46x9
> python3 --version
zsh: command not found: python3
```



파이썬 설치 (1)

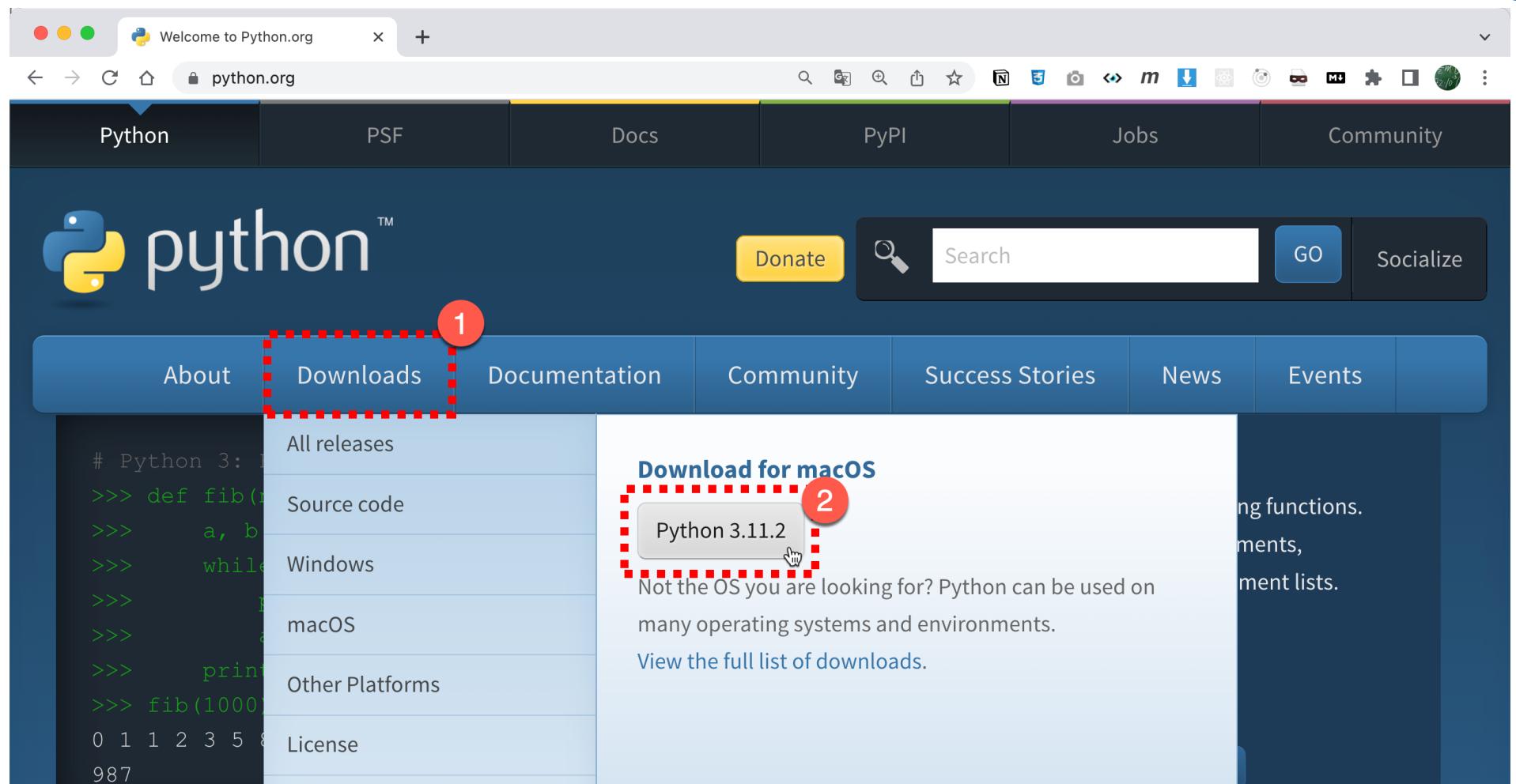
□ 파이썬 설치가 필요한 경우 설치 프로그램 다운로드

◎ 3.11 버전 혹은 그 이후 버전을 내려 받는다.

“<https://www.python.org>

1. 구글스프레드시트 활용

2. 브이월드 API 활용





1. 구글스프레드시트 활용
2. 브이월드 API 활용

□ Windows에서 파이썬 설치 시 주의사항

- ◎ 내려 받은 설치 프로그램을 실행하여 설치를 진행한다.
- ◎ 첫 화면의 맨 아래에 있는 “Add python.exe to PATH” 항목을 반드시 체크하고 다음으로 넘어간다. (Window에만 해당함)
- ◎ Mac의 경우는 특별한 이슈 없이 설치가 가능하다.





pip 업그레이드

1. 구글스프레드시트 활용

2. 브이월드 API 활용

□ PIP

- ◎ 파이썬의 패키지 관리 도구 → 패키지의 설치, 업그레이드, 삭제 기능을 수행한다.
 - 패키지는 파이썬에 추가할 수 있는 확장팩으로 이해하자.

□ PIP 업그레이드 명령어를 CLI 환경에서 수행

- ◎ window의 경우

```
python -m pip install --upgrade pip
```

- ◎ Mac의 경우

```
python3 -m pip install --upgrade pip
```

- ◎ Mac에서 위의 명령이 에러나는 경우

- 관리자 권한을 부여하기 위해 sudo 명령을 추가한다.

```
sudo python3 -m pip install --upgrade pip
```

1. 구글스프레드시트 활용

2. 브이월드 API 활용



py-geocoder 설치하기 (1)

- py-geocoder 소개 (<https://github.com/leekh4232/py-geocoder>)
 - ◎ Python 기반으로 작성된 오픈소스 지오코더 프로그램
 - ◎ 브이월드 OpenAPI와 연동된다.

The screenshot shows the GitHub repository page for 'py-geocoder' by user 'leekh4232'. The page includes the README file, which describes the project as a geocoding tool developed by Lee KwangHo and Ju YoungA, version 0.1.0, using Python and Pandas. It mentions that the tool is intended for a research project at Yonsei University's Department of Urban Planning and Design. The repository also includes sections for Packages (no packages published), Languages (Python 100.0%), and Suggested workflows (SLSA Generic generator, Pylint).

py-geocoder

Author Lee KwangHo Author Ju YoungA version 0.1.0 license MIT Python Pandas

이 자료는 연세대학교 도시공학과 주영아(j.purplerose@gmail.com)의 논문 프로젝트에 활용되기 위해 작성된 Geocoder 툴입니다.

브이월드 OpenAPI키를 발급받아 사용할 수 있으며, 브이월드 정책에 따라 개발키 적용시 1일 40,000건의 데이터를 처리할 수 있습니다.

비동기를 지원하기 때문에 빠른 처리가 가능합니다.

Installation

pip 명령으로 *.whl 파일을 설치합니다.

Packages

No packages published [Publish your first package](#)

Languages

Python 100.0%

Suggested workflows

Based on your tech stack

SLSA Generic generator [Configure](#)
Generate SLSA3 provenance for your existing release workflows

Pylint [Configure](#)
Lint a Python application with pylint.



py-geocoder 설치하기 (2)

□ py-geocoder 설치하기

◎ window의 경우

```
pip install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl
```

The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd'. The command entered is 'pip install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl'. The output shows the progress of the download and installation of various dependencies, including py-geocoder, pandas, tqdm, typer, and numpy.

```
C:\Windows\system32\cmd + - X
Microsoft Windows [Version 10.0.22000.2960]
(c) Microsoft Corporation. All rights reserved.

C:\Users\leekh>pip install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl
Collecting py-geocoder==0.1.0
  Downloading https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl (3.4 kB)
Collecting pandas<3.0.0,>=2.2.2
  Downloading pandas-2.2.2-cp311-cp311-win_amd64.whl (11.6 MB)
    11.6/11.6 MB 34.5 MB/s eta 0:00:00
Collecting tqdm<5.0.0,>=4.66.4
  Downloading tqdm-4.66.4-py3-none-any.whl (78 kB)
    78.3/78.3 kB ? eta 0:00:00
Collecting typer[all]<0.13.0,>=0.12.3
  Downloading typer-0.12.3-py3-none-any.whl (47 kB)
    47.2/47.2 kB ? eta 0:00:00
Collecting numpy>=1.23.2
  Downloading numpy-1.26.4-cp311-cp311-win_amd64.whl (15.8 MB)
    15.8/15.8 MB 54.4 MB/s eta 0:00:00
```

설치 명령을 <https://github.com/leekh4232/py-geocoder> 페이지에 접속하여 복사 후 명령프롬프트(터미널)에 붙여넣기 하여 설치하세요.



py-geocoder 설치하기 (3)

□ py-geocoder 설치하기

◎ Mac의 경우

```
pip3 install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl
```

◎ Mac에서 위의 명령이 에러나는 경우

- 관리자 권한을 부여하기 위해 sudo 명령을 추가한다.

```
sudu pip3 install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl
```



서울시 응급실 위치 정보 데이터 확인

- 1. 구글스프레드시트 활용
- 2. 브이월드 API 활용

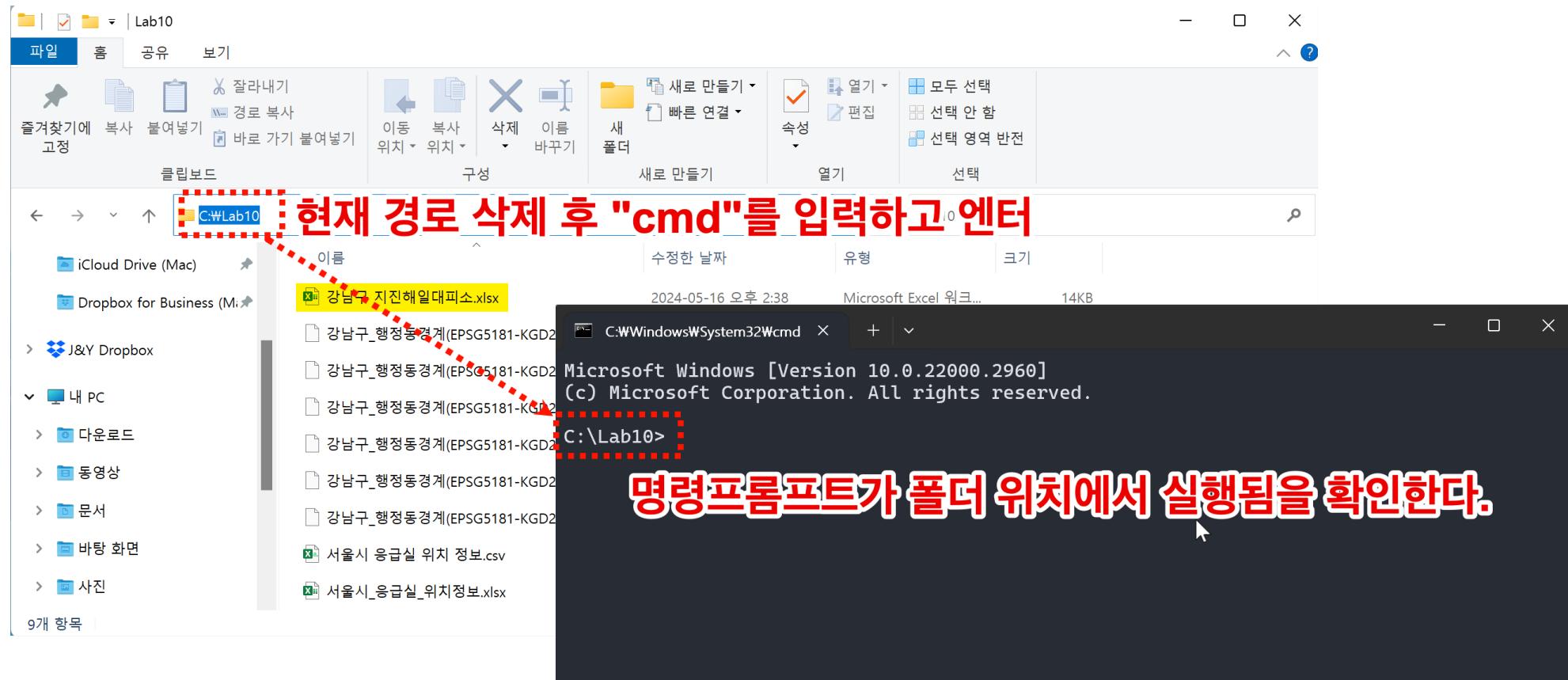
- “서울시_응급실_위치정보.xlsx” (엑셀파일)
 - ◎ 서울시에 위치한 응급실 이름, 주소, 구분, 연락처 등이 정리되어 있는 데이터 자료
 - ◎ 출처: 서울시 공공데이터 포털

기관ID	기관명	주소	병원분류	응급의료기관코드	응급의료기관코드명	대표전화	응급실전화
A1100011	가톨릭대학교 여의도성모병원	서울특별시 영등포구 63로 10, 여의도성모병원 (여의도동)	종합병원	G006	지역응급의료센터	1661-7575	02-3779-1188
A1121013	가톨릭대학교 은평성모병원	서울특별시 은평구 통일로 1021 (진관동)	종합병원	G006	지역응급의료센터	02-1811-7755	02-1811-7755
A1100047	강남고려병원	서울특별시 관악구 관악로 242 (봉천동)	병원	G009	응급의료기관 외의 의료기관(응급의료시설)	1661-8267	02-874-7227
A1100141	강남베드로병원	서울특별시 강남구 남부순환로 2649, 베드로병원 (도곡동)	종합병원	G009	응급의료기관 외의 의료기관(응급의료시설)	1544-7522	1544-7522
A1100076	강남힐병원	서울특별시 관악구 남부순환로 1449, 강남힐병원 (신림동)	병원	G009	응급의료기관 외의 의료기관(응급의료시설)	02-853-4600	02-853-9100
A1100043	강동경희대학교의대병원	서울특별시 강동구 도남로 892 (상일동)	종합병원	G001	권역응급의료센터	02-440-7114	02-440-7000
A1100006	강북삼성병원	서울특별시 종로구 새문안로 29 (평동)	종합병원	G006	지역응급의료센터	02-2001-2001	02-2001-1000
A1121842	강북으뜸병원	서울특별시 강북구 도봉로 187, 지하1층, 2층~5층 (미아동)	병원	G009	응급의료기관 외의 의료기관(응급의료시설)	02-986-0334	02-986-0334
A1100002	건국대학교병원	서울특별시 광진구 능동로 120-1 (화양동)	종합병원	G006	지역응급의료센터	1588-1533	02-2030-5555
A1100039	경찰병원	서울특별시 송파구 송이로 123, 국립경찰병원 (가락동)	종합병원	G007	지역응급의료기관	02-3400-1114	02-3400-1300
A1100001	경희대학교병원	서울특별시 동대문구 경희대로 23 (경기동)	종합병원	G006	지역응급의료센터	02-958-8114	02-958-8114
A1100014	고려대학교의과대학부속구로병원	서울특별시 구로구 구로동로 148, 고려대부속구로병원 (구로동)	종합병원	G001	권역응급의료센터	02-2626-1114	02-2626-1550
A1100026	구로성심병원	서울특별시 구로구 경인로 427, 구로성심병원 (고척동)	종합병원	G007	지역응급의료기관	02-2067-1500	02-2067-1515
A1100052	국립중앙의료원	서울특별시 종구 을지로 245, 국립중앙의료원 (을지로6가)	종합병원	G006	지역응급의료센터	02-2260-7114	02-2260-7414
A1100048	노원을지대학교병원	서울특별시 노원구 한글비석로 68, 을지병원 (하계동)	종합병원	G006	지역응급의료센터	02-970-8000	02-970-8282
A1100044	녹색병원	서울특별시 종량구 사가정로49길 53 (면목동)	종합병원	G007	지역응급의료기관	02-490-2000	02-490-2113
A1100037	대림성모병원	서울특별시 영등포구 시흥대로 657 (대림동, 대림성모병원)	종합병원	G007	지역응급의료기관	02-829-9000	02-829-9129
A1100024	명지성모병원	서울특별시 영등포구 도림로 156, 명지성모병원 (대림동)	종합병원	G007	지역응급의료기관	02-1899-1475	02-829-7800
A1100046	미즈메디병원	서울특별시 강서구 강서로 295 (내발산동)	종합병원	G009	응급의료기관 외의 의료기관(응급의료시설)	02-2007-1252	02-2007-1118
A1100036	부민병원	서울특별시 강서구 공항대로 389, 부민병원 (등촌동)	종합병원	G007	지역응급의료기관	1577-7582	02-2620-0119
A1100148	사랑의병원	서울특별시 관악구 남부순환로 1860, -1, 1, 3, 4, 5층 (봉천동,)	병원	G009	응급의료기관 외의 의료기관(응급의료시설)	02-880-0114	02-880-0119
A1100010	삼성서울병원	서울특별시 강남구 일원로 81 (일원동, 삼성의료원)	종합병원	G006	지역응급의료센터	02-3410-2114	02-3410-2060
A1100021	삼육서울병원	서울특별시 동대문구 망우로 82 (휘경동)	종합병원	G006	지역응급의료센터	02-2244-0191	02-2210-3566
A1100017	서울대학교병원	서울특별시 종로구 대학로 101 (연건동)	종합병원	G001	권역응급의료센터	1588-5700	02-2072-2475
A1100050	서울성심병원	서울특별시 동대문구 왕산로 259, 서울성심병원 (청량리동)	종합병원	G007	지역응급의료기관	02-966-1616	02-966-1616
A1100029	서울적십자병원	서울특별시 종로구 새문안로 9, 적십자병원 (평동)	종합병원	G007	지역응급의료기관	02-2002-8000	02-2002-8888
A1100022	서울특별시동부병원	서울특별시 동대문구 무학로 124 (용두동)	종합병원	G007	지역응급의료기관	02-920-9114	02-920-9119
A1100040	서울특별시보라매병원	서울특별시 동작구 보라매로5길 20 (신대방동)	종합병원	G006	지역응급의료센터	02-870-2114	02-870-2119

1. 구글스프레드시트 활용
2. 브이월드 API 활용

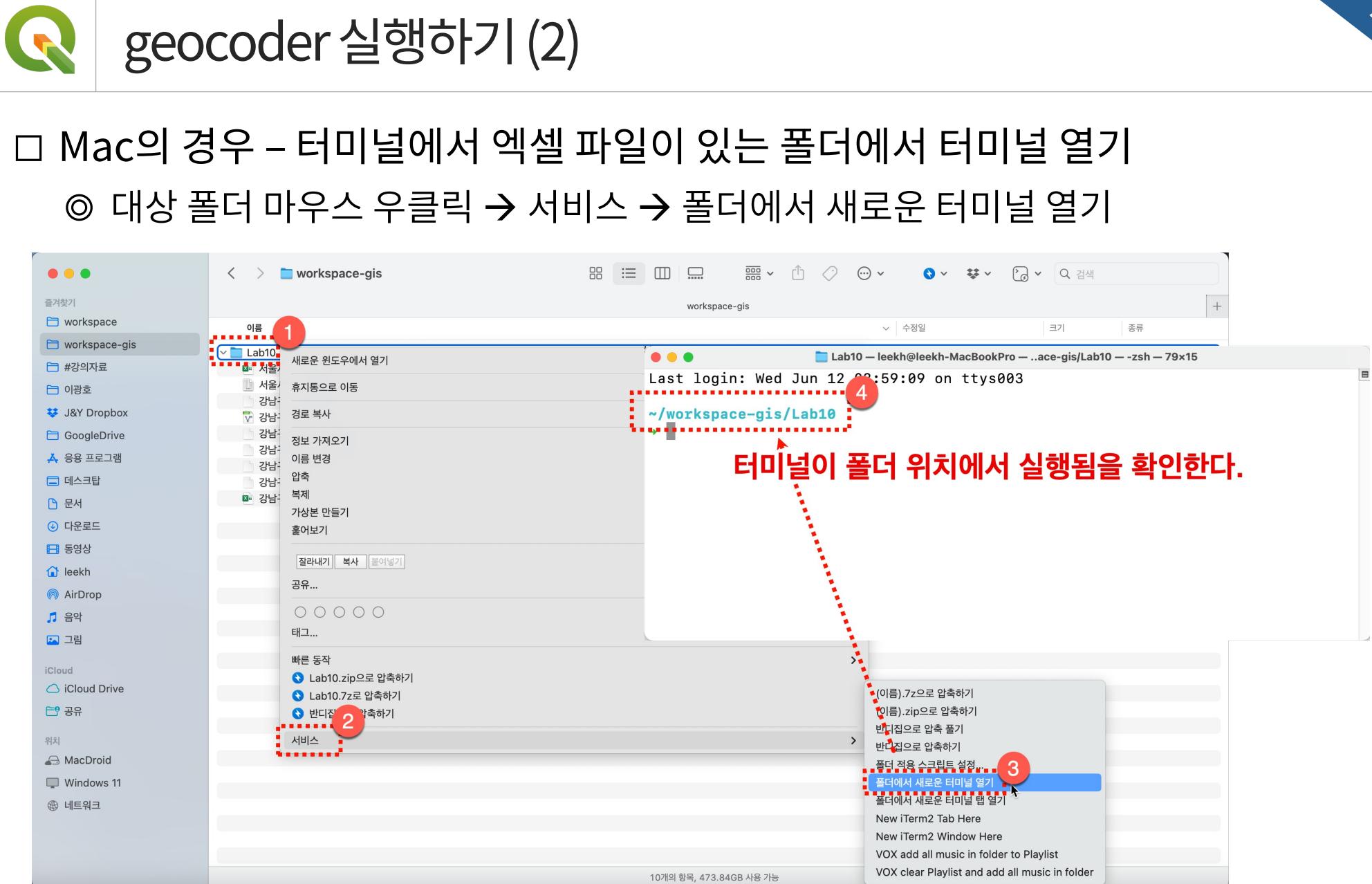
geocoder 실행하기 (1)

- Windows의 경우 - 엑셀 파일이 있는 위치에서 명령프롬프트 실행하기
 - ◎ 윈도우 탐색기를 엑셀 파일이 포함되어 있는 폴더의 위치로 이동하고, 경로 표시줄에 “cmd”를 입력후 엔터를 누른다.



1. 구글스프레드시트 활용

2. 브이월드 API 활용



The QGIS logo consists of a green stylized letter 'Q' with a 3D orange and yellow block extending from the top-left corner of its vertical stroke.

geocoder 실행하기 (3)

□ py-geocoder 실행하기 (Windows, Mac 공통)

◎ 아래의 명령어 형식에 맞춰 실행한다.

```
py-geocoder --key={브이월드OpenAPI키} --input={주소가작성된엑셀파일명} --addr={주소필드이름}
```

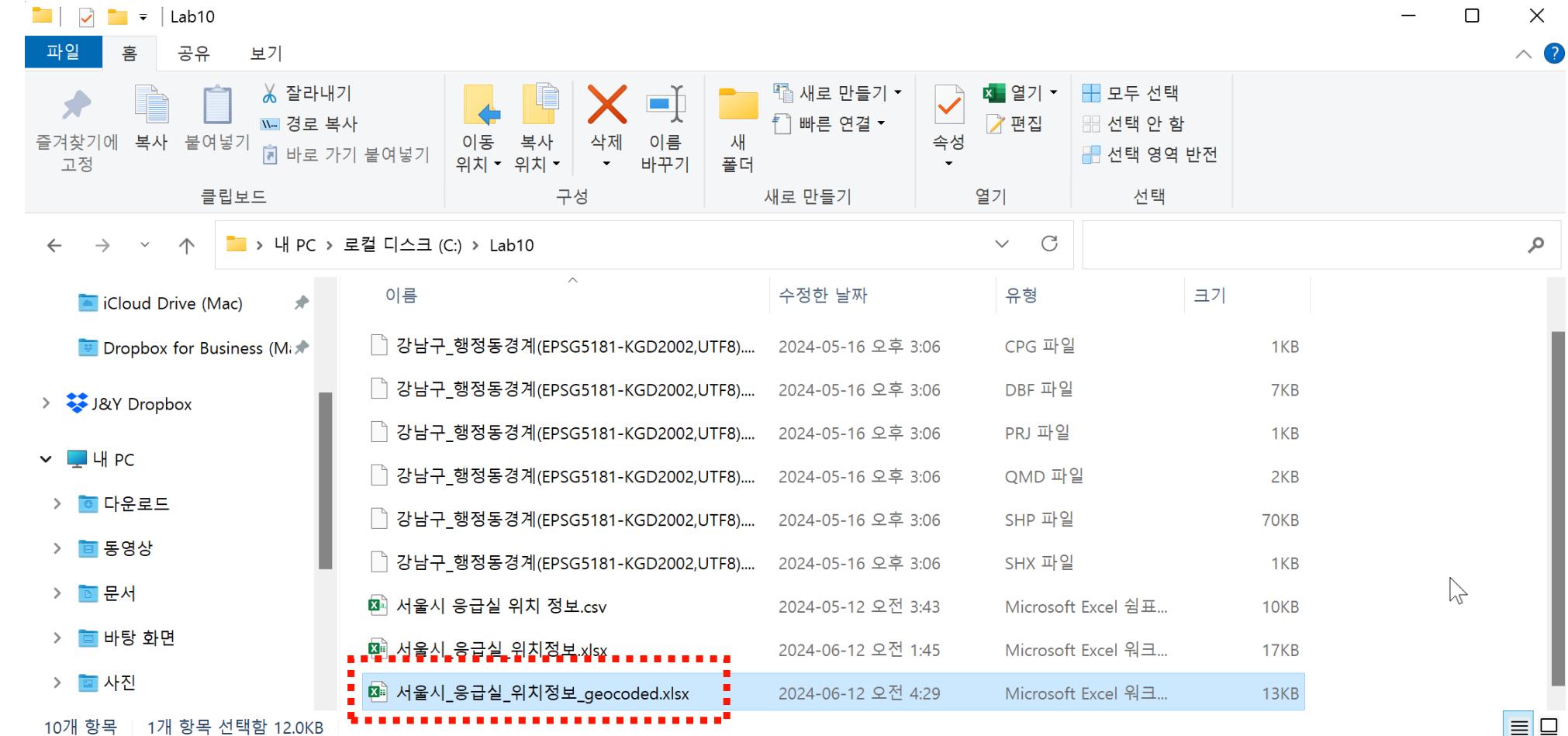
1. 구글스프레드시트 활용

2. 브이월드 API 활용

geocoder 실행하기 (4)

□ geocoder 결과 확인

◎ “[기존파일이름]_geocoded.xlsx” 파일이 생성됨을 확인할 수 있다.



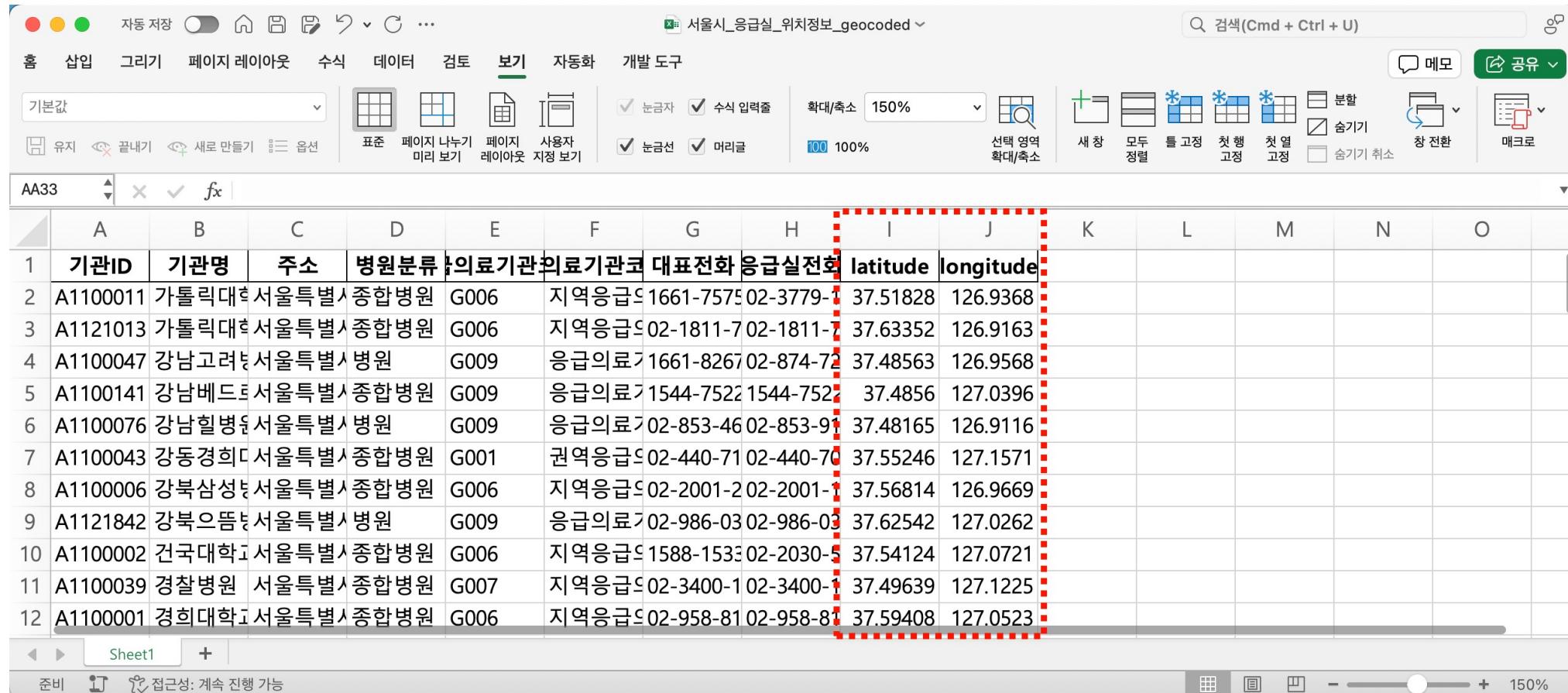
1. 구글스프레드시트 활용

2. 브이월드 API 활용

geocoder 실행하기 (5)

□ 주소 데이터에 대한 경,위도 확인

◎ 새로 생성된 엑셀 파일상에 경,위도 필드가 추가되어 있음을 확인한다.

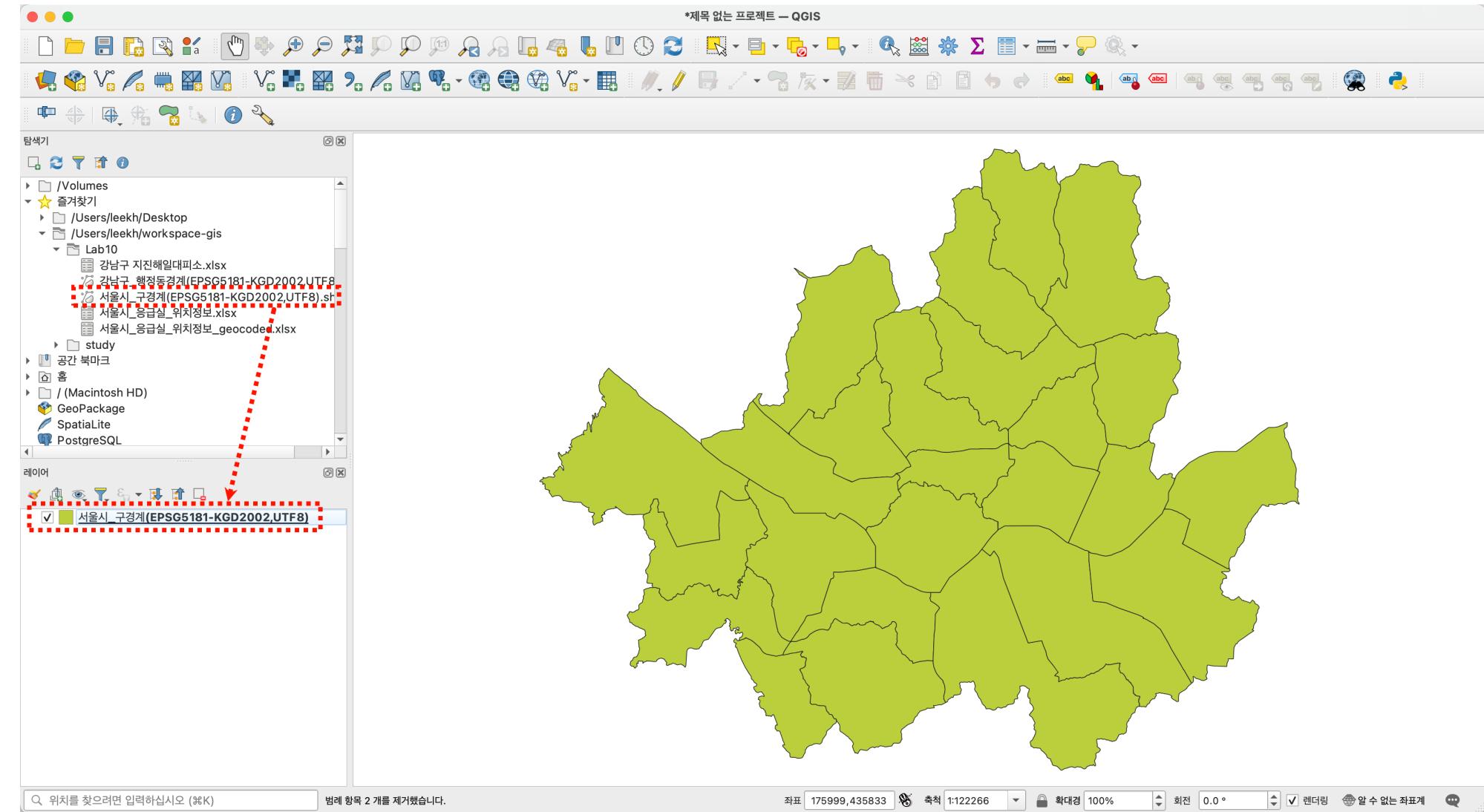


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	I
1	기관ID	기관명	주소	병원분류	의료기관	의료기관코드	대표전화	응급실전화	latitude	longitude						
2	A1100011	가톨릭대서울특별시 종합병원	G006	지역응급의료기관	1661-7575	02-3779-1700	37.51828	126.9368								
3	A1121013	가톨릭대학서울특별시 종합병원	G006	지역응급의료기관	02-1811-702-1811-70	02-1811-702-1811-70	37.63352	126.9163								
4	A1100047	강남고려한서울특별시 병원	G009	응급의료기관	1661-8267	02-874-7230	37.48563	126.9568								
5	A1100141	강남베드로서울특별시 종합병원	G009	응급의료기관	1544-7522	1544-7522	37.4856	127.0396								
6	A1100076	강남힐병원서울특별시 병원	G009	응급의료기관	02-853-4602	02-853-9130	37.48165	126.9116								
7	A1100043	강동경희대학교서울특별시 종합병원	G001	권역응급의료기관	02-440-7102	02-440-7000	37.55246	127.1571								
8	A1100006	강북삼성한서울특별시 종합병원	G006	지역응급의료기관	02-2001-2002	02-2001-1000	37.56814	126.9669								
9	A1121842	강북으뜸한서울특별시 병원	G009	응급의료기관	02-986-0302	02-986-0302	37.62542	127.0262								
10	A1100002	건국대학교서울특별시 종합병원	G006	지역응급의료기관	1588-1533	02-2030-5000	37.54124	127.0721								
11	A1100039	경찰병원서울특별시 종합병원	G007	지역응급의료기관	02-3400-1002	02-3400-1000	37.49639	127.1225								
12	A1100001	경희대학교서울특별시 종합병원	G006	지역응급의료기관	02-958-8102	02-958-8102	37.59408	127.0523								



서울시 구 단위 응급실 위치 지도 만들기 (1)

- “서울시_구경계(EPSG5181-KGD2002,UTF8).shp” 파일을 작업 레이어에 추가





서울시 구 단위 응급실 위치 지도 만들기 (2)

□ 지오코딩 결과가 포함된 엑셀 파일을 Point 레이어로 추가

◎ 툴바에서 “Add Spreadsheet Layer” 버튼을 클릭한다.



Add Spreadsheet Layer...

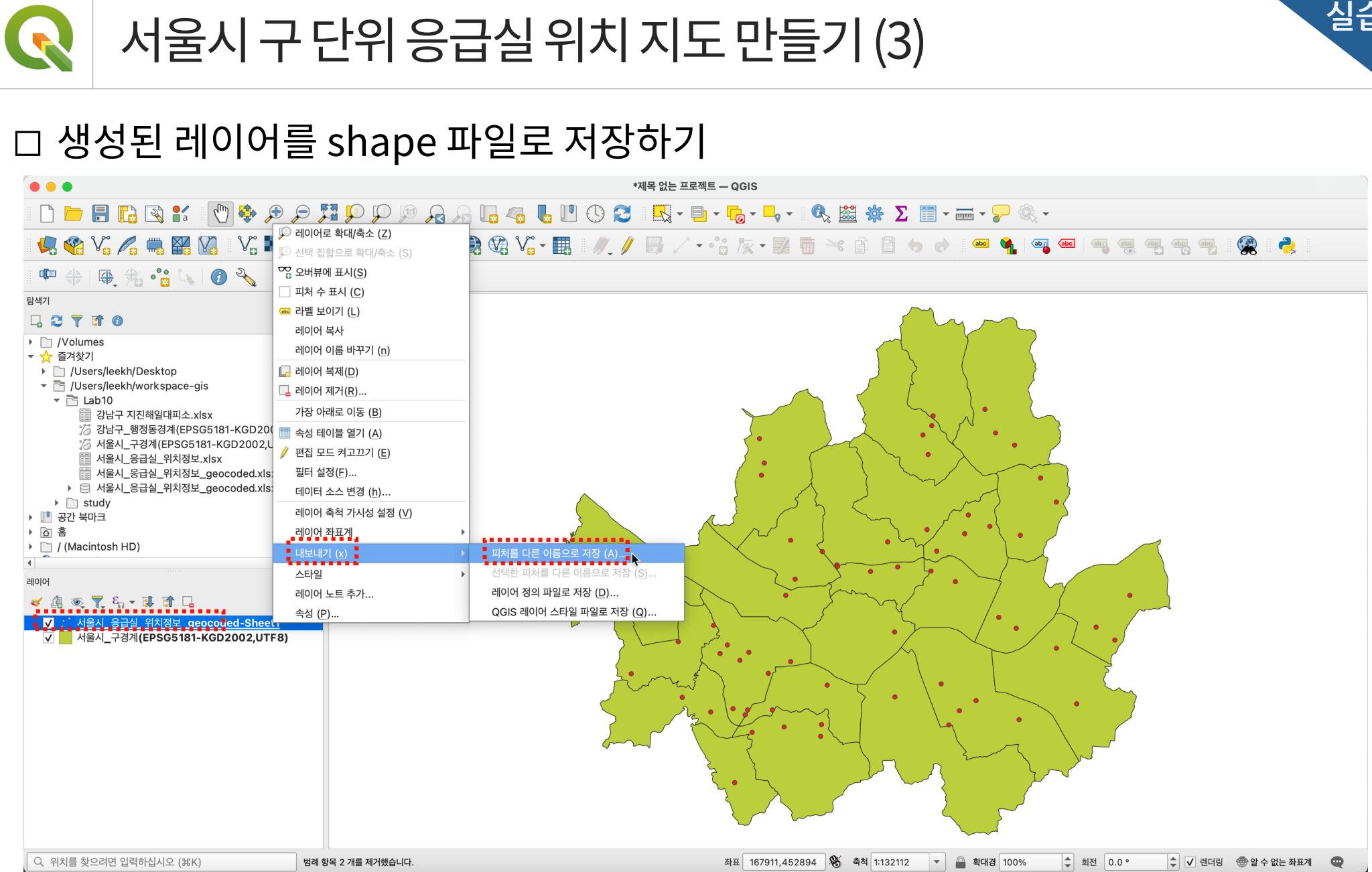
◎ 경위도 기반이므로 좌표계는 “EPSG 4326 - WGS84”를 지정해야 한다.

The screenshot shows two QGIS dialog boxes. The left dialog is 'Create a Layer from a Spreadsheet File' with the following steps numbered:

- File Name: /Users/leekh/workspace-gis/Lab10/서울시_응급실_위치정보_geocoded.xlsx (1)
- Sheet: Sheet1 (2)
- Geometry checkbox (3)
- Encoding: PointFromColumns (4)
- Field: X field: longitude, Y field: latitude (5)
- The right dialog is 'Coordinate Reference System Selector' with the following steps numbered:
- Search input: 사전 정의 좌표계 (6)
- Filter input: 4326 (7)
- Search button: 검색 (7)
- Result list: WGS 84 (8)
- Details panel: WGS 84 (9)

1. 구글스프레드시트 활용

2. 브이월드 API 활용



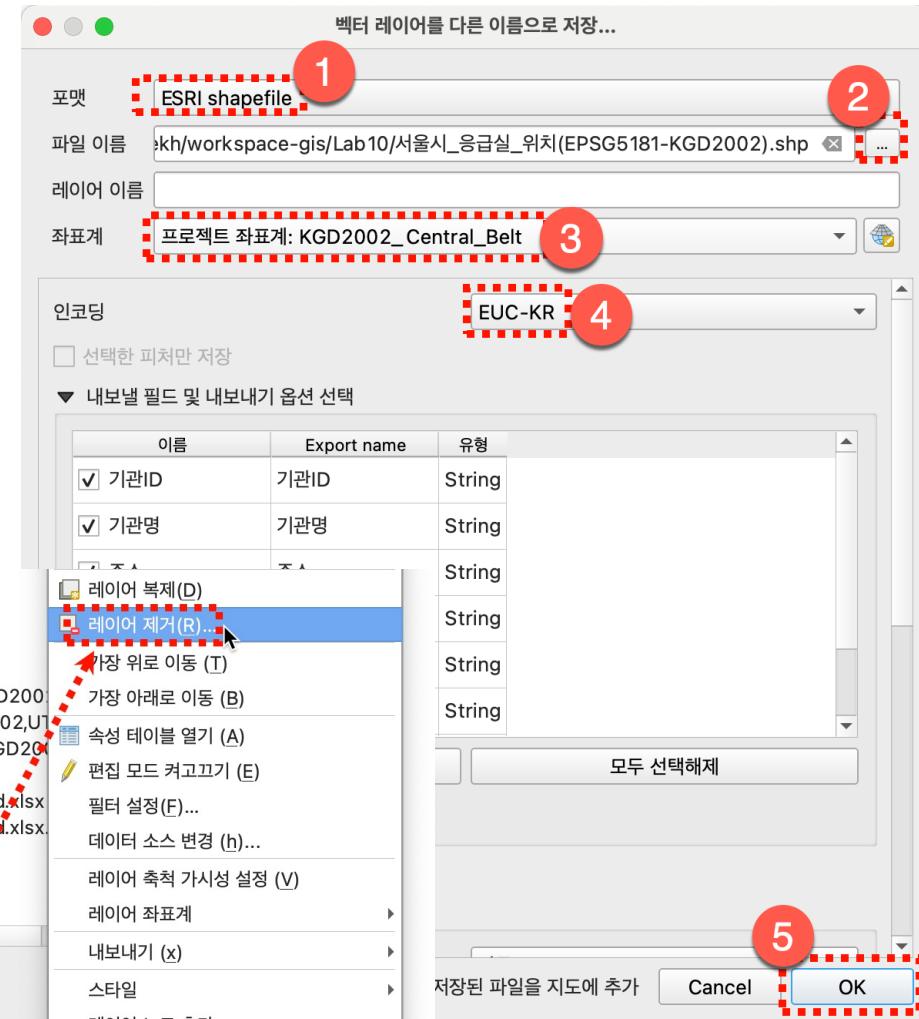
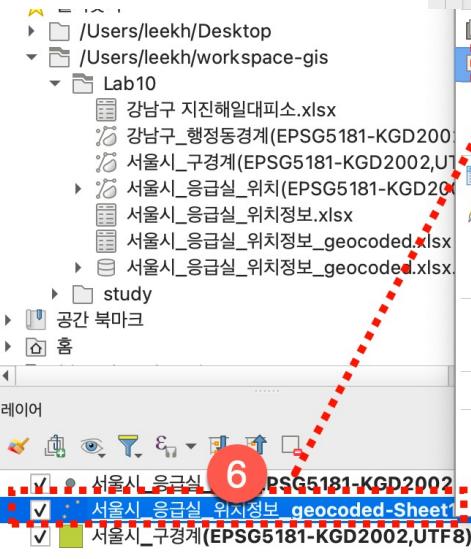


서울시 구 단위 응급실 위치 지도 만들기 (4)

1. 구글스프레드시트 활용
2. 브이월드 API 활용

□ 저장 파라미터 설정

- ① 저장 포맷을 shape 파일로 설정
- ② 저장 경로 지정
- ③ 좌표계를 변경하고자 할 경우 설정
 - 여기서는 프로젝트 좌표계와 일치시킴
- ④ 한글 필드명이 있으므로 인코딩을 EUC-KR로 지정
 - UTF-8은 한글 필드명이 깨진다(QGIS 버그)
- ⑤ 설정 내용 실행
- ⑥ 레이어 저장 후 불필요한 기존 레이어 "삭제"



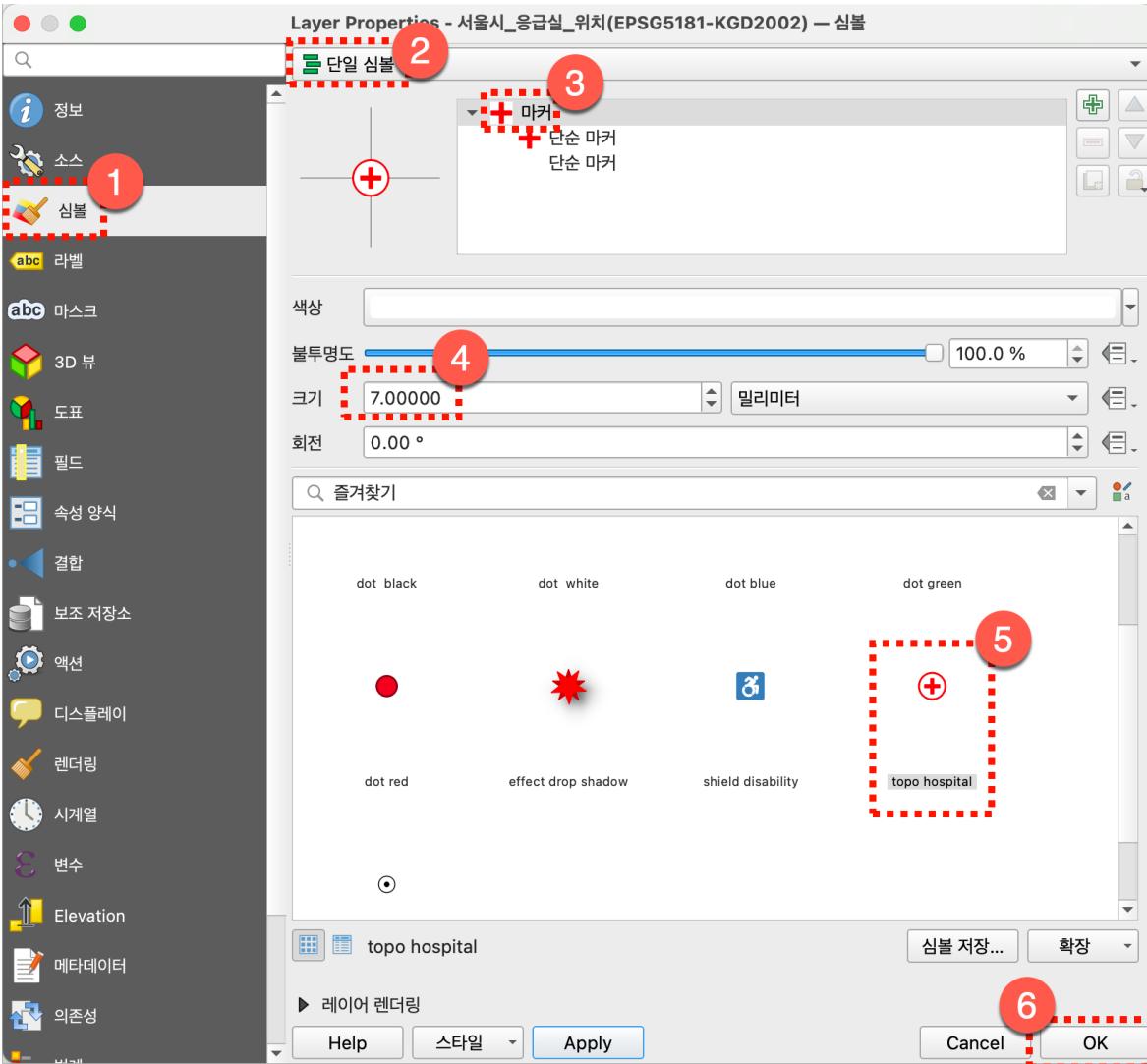
1. 구글스프레드시트 활용

2. 브이월드 API 활용

서울시 구 단위 응급실 위치 지도 만들기 (5)

□ 응급실 shape 레이어의 속성창에서 심볼을 설정한다.

- ① 심볼 선택
- ② 단일 심볼 선택
- ③ 마커 선택
- ④ 마커의 적정 크기 지정
- ⑤ 병원 아이콘 선택
- ⑥ 설정 내용 실행



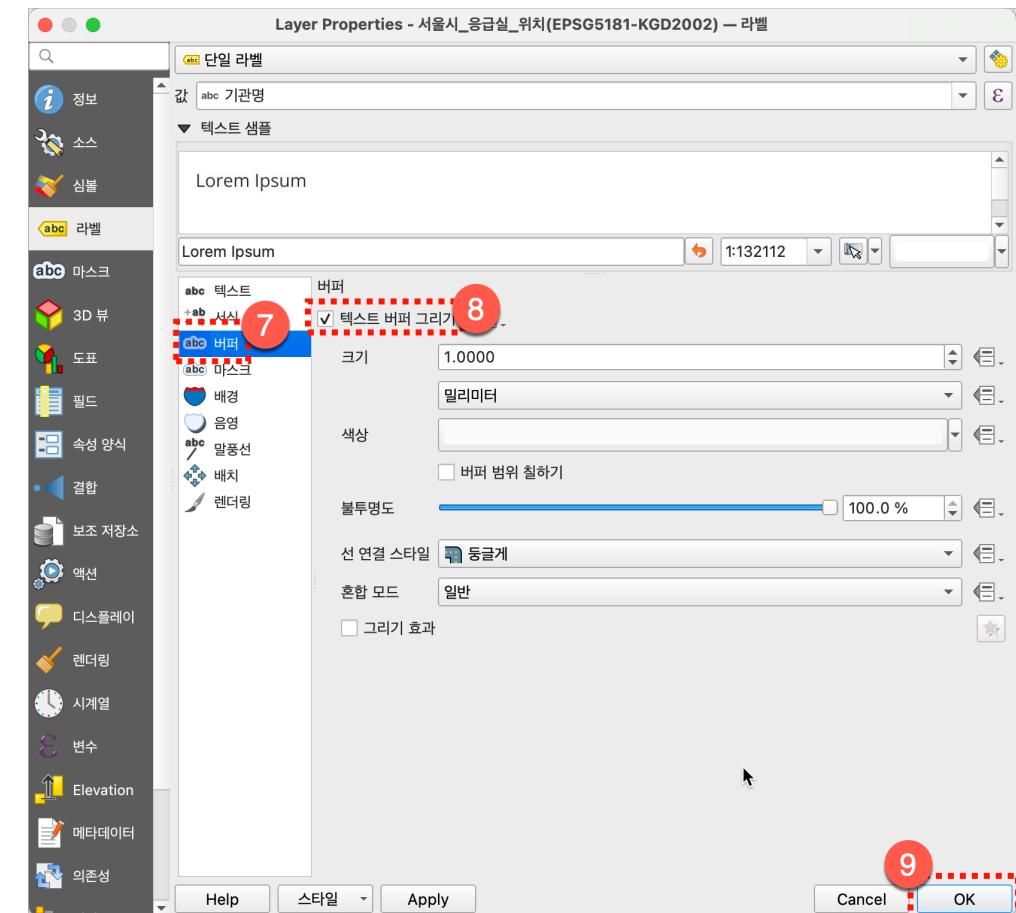
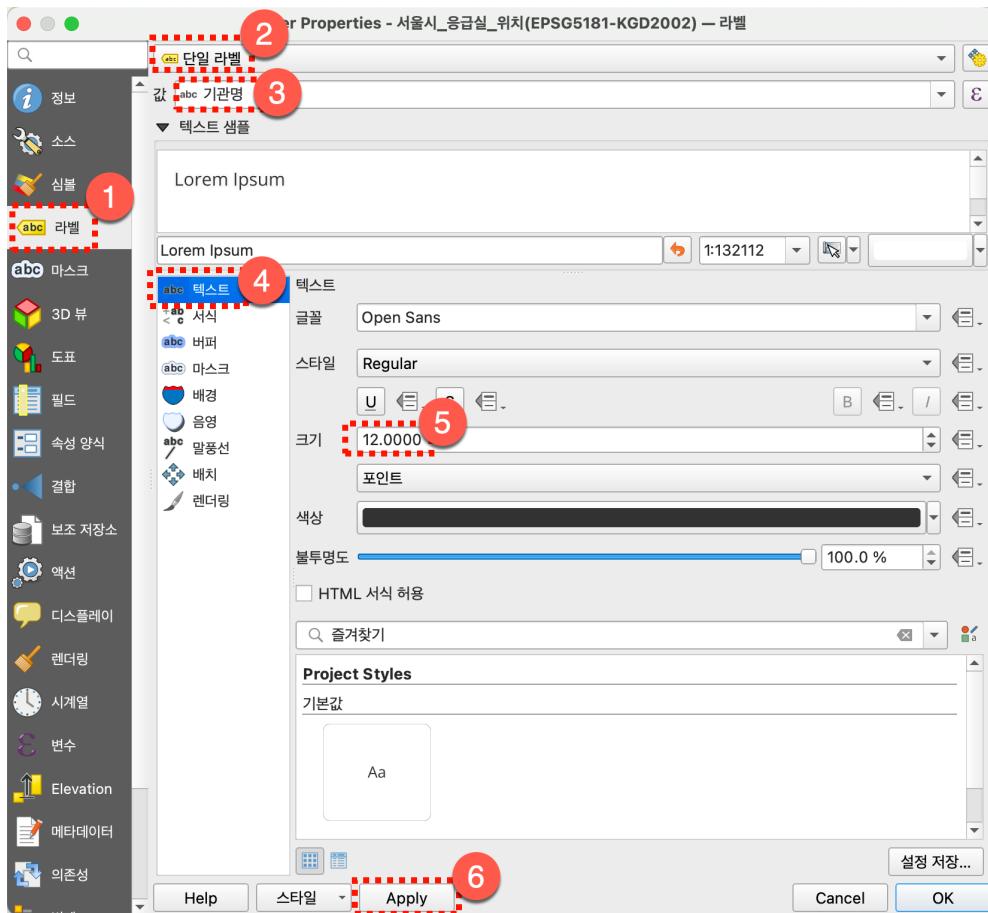


서울시 구 단위 응급실 위치 지도 만들기 (6)

□ 병원 이름에 대한 라벨 설정

1. 구글스프레드시트 활용

2. 브이월드 API 활용

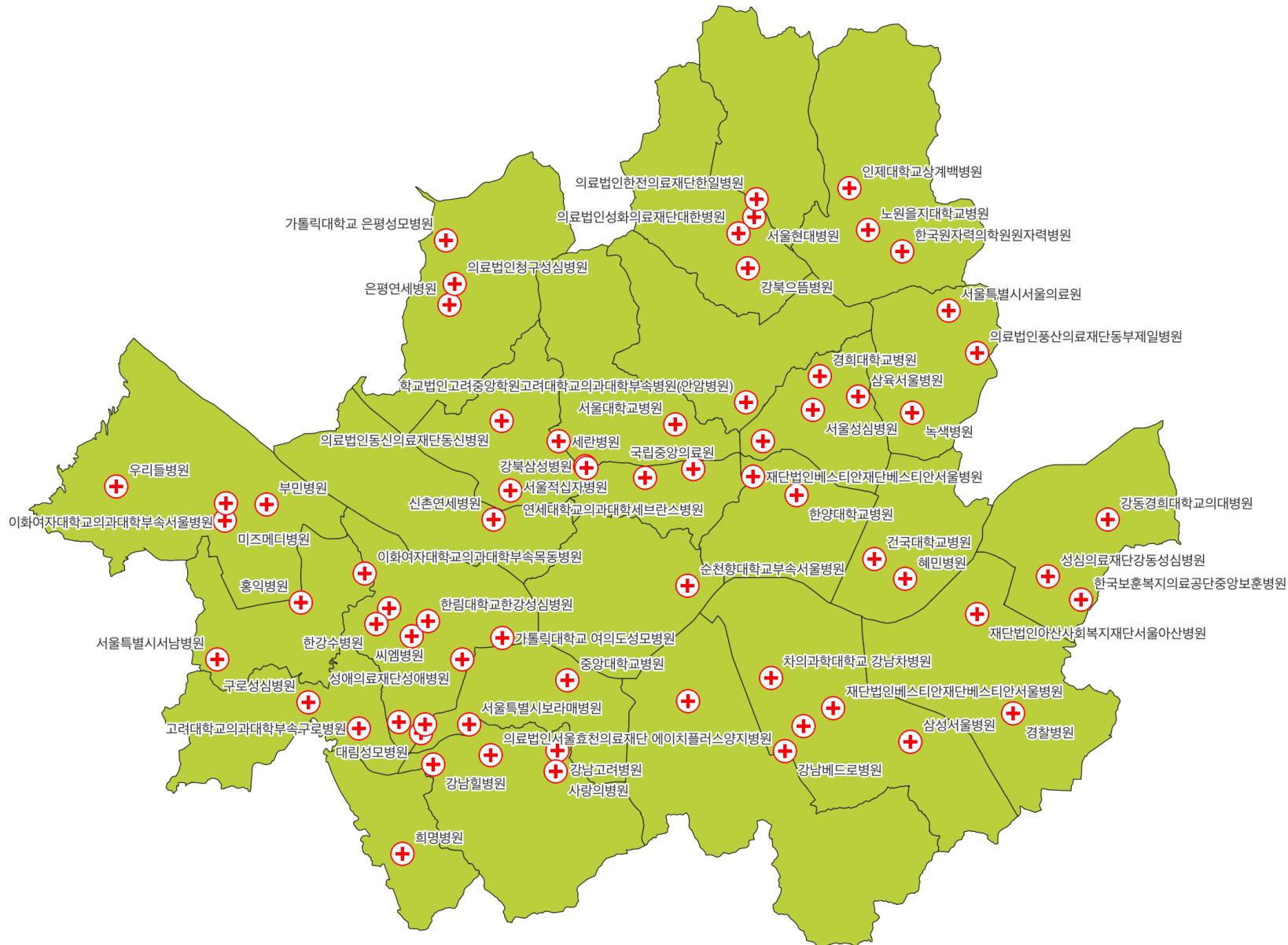




서울시 구 단위 응급실 위치 지도 만들기 (7) – 결과 확인

1. 구글스프레드시트 활용

2. 브이월드 API 활용





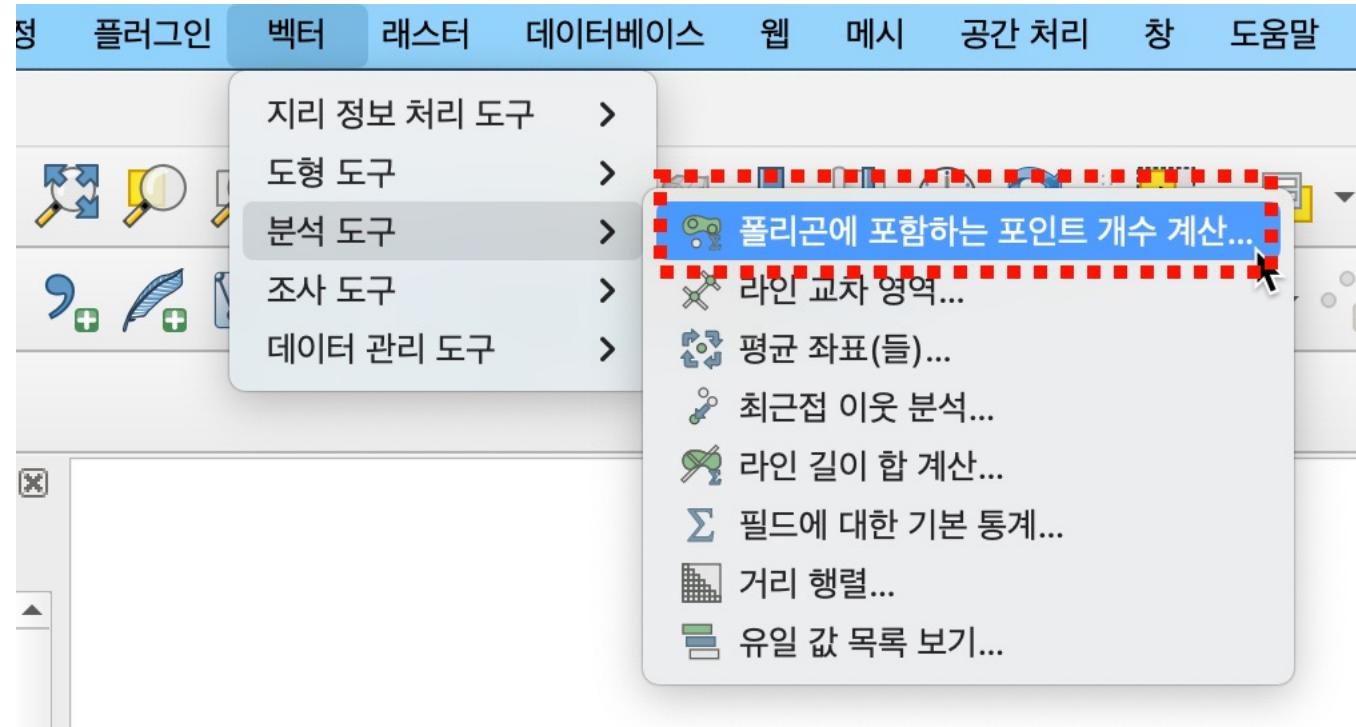
구 단위 응급실 분포 단계 구분도 만들기 (1)

□ 폴리곤에 포함되는 포인트(응급실)의 수 계산하기 (1)

- ◎ “벡터 → 분석 도구 → 폴리곤에 포함하는 포인트 개수 계산” 메뉴를 실행한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

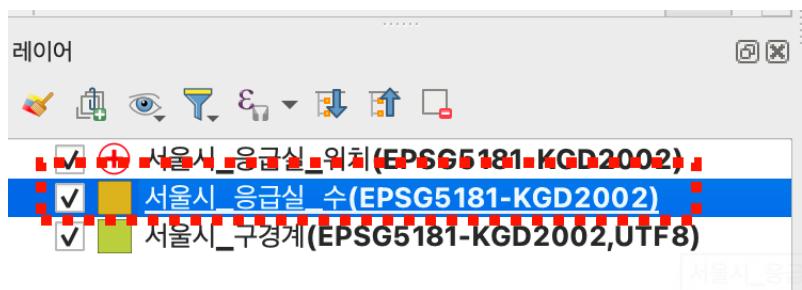




구 단위 응급실 분포 단계 구분도 만들기 (2)

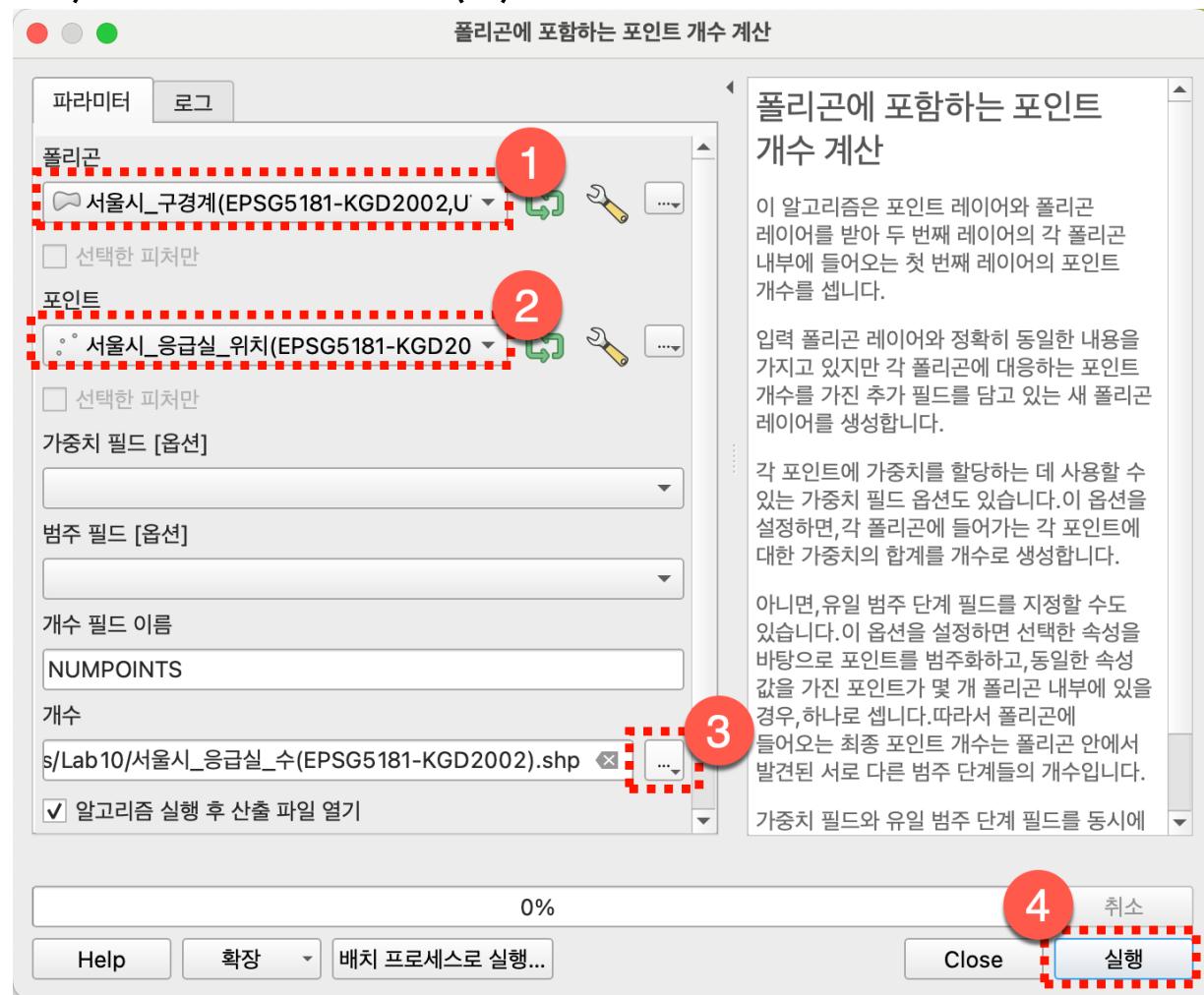
□ 폴리곤에 포함되는 포인트(응급실)의 수 계산하기 (2)

- ① 폴리곤 레이어 선택
→ 서울시 구 경계 지도
- ② 포인트 레이어 선택
→ 서울시 응급실 위치
- ③ 저장될 레이어 경로 지정
- ④ 설정 내용 실행



“서울시 응급실 수” 필드가 추가된 폴리곤 레이어가 생성된다.

이 레이어를 응급실 위치 포인트 레이어의 아래로 배치한다.



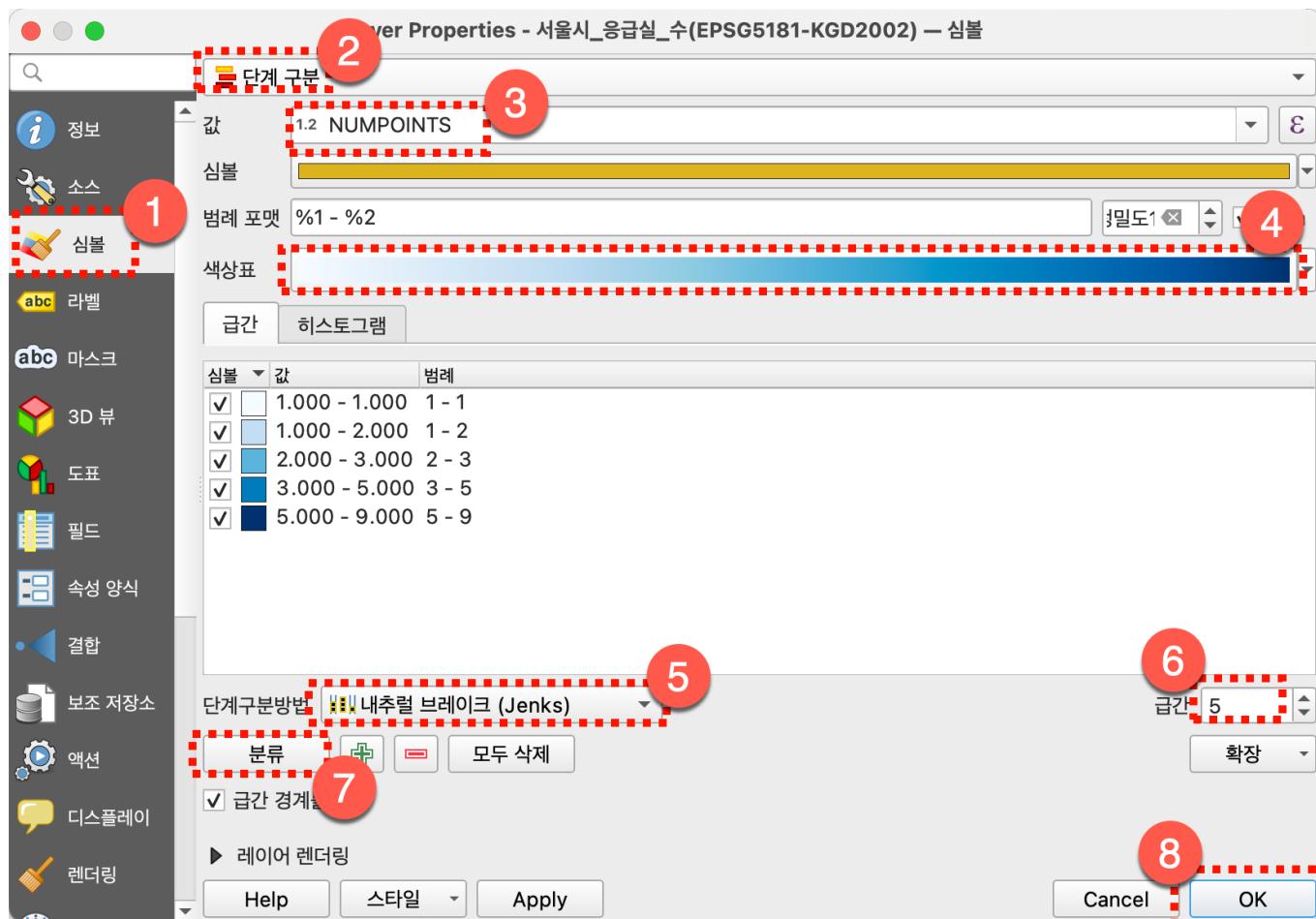


구 단위 응급실 분포 단계 구분도 만들기 (3)

1. 구글스프레드시트 활용
2. 브이월드 API 활용

□ 단계 구분 설정

- ◎ 새로 생성된 레이어에는 “NUMPOINTS”라는 필드가 생성된다.
- ◎ 이 필드는 폴리곤(서울시 구 경계)안에 포함되는 포인트(응급실)의 수를 저장하고 있다.
- ◎ 이 값을 사용하여 단계 구분도를 작성한다.





구 단위 응급실 분포 단계 구분도 만들기 - 결과 확인

□ 서울시 응급실 분포 및 위치 지도

1. 구글스프레드시트 활용

2. 브이월드 API 활용



1. 구글스프레드시트 활용

2. 브이월드 API 활용

실습하기

브이월드 API 활용



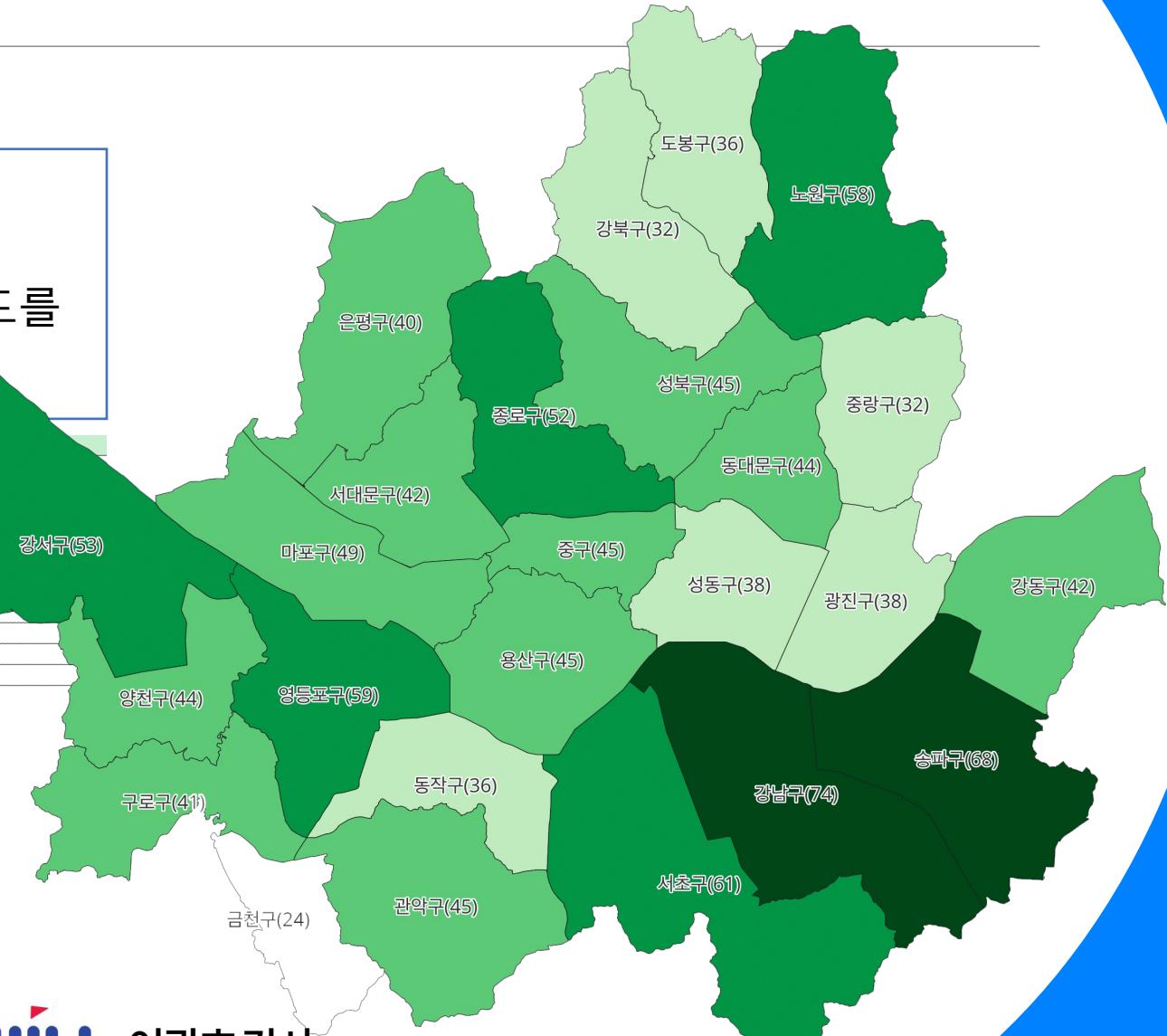


연구과제

□ 서울시 행정 기관 분포도

“서울_행정기관.xlsx”파일은 서울시에 위치한 행정기관의 주소를 담고 있다. 이 데이터를 사용하여 서울시의 구단위 행정기관의 분포도를 작성하시오.

기관유형	기관유형별분류	대표기관명	전체기관명
우정	우정_우체국	과학기술정보통신부	과학기술정보통신부 서울지방우정청 서울도봉우체국 서울우이동우편취급국
보훈	보훈부_국립묘지	국가보훈부	국가보훈부 국립4.19민주묘지관리소
지자체	8읍면동_동	서울특별시	서울특별시 강북구 우이동주민센터
지자체	8읍면동_동	서울특별시	서울특별시 강북구 인수동주민센터
지자체	소방_119_안전센터	서울특별시	서울특별시 강북소방서 우이119안전센터
우정	우정_우체국	과학기술정보통신부	과학기술정보통신부 서울지방우정청 서울도봉우체국 서울수유동우체국
지자체	8읍면동_동	서울특별시	서울특별시 강북구 수유2동주민센터
우정	우정_우체국	과학기술정보통신부	과학기술정보통신부 서울지방우정청 서울도봉우체국 서울수유3동우체국
지자체	8읍면동_동	서울특별시	서울특별시 강북구 번1동주민센터
지자체	5시군구_구	서울특별시	서울특별시 강북구청
지자체	8읍면동_동	서울특별시	서울특별시 강북구 수유3동주민센터
우정	우정_우체국	과학기술정보통신부	과학기술정보통신부 서울지방우정청 서울도봉우체국 서울인수동우편취급국





THANK YOU

수고하셨습니다.



[LAB-06] 1. LinePlot



#00. 데이터 시각화 개요



1. 데이터의 패턴과 의미를 ‘눈으로 보게’ 만드는 과정

숫자·표 형태로는 보이지 않던 경향, 변화, 차이, 관계를 그래픽으로 변환해 직관적으로 이해하게 만드는 작업.

너무 화려하거나 복잡하면 오히려 데이터의 이해를 어렵게 한다.



2. 데이터 시각화의 목적

- **분석**: 데이터 속 패턴·이상치·분포·상관관계 파악
- **설명**: 분석 결과를 쉽게 전달
- **의사결정**: 인사이트를 바탕으로 더 나은 선택을 가능하게 함



3. 데이터 시각화 구분

분류	설명	대표 목적	주로 사용하는 seaborn 함수
시계열(Time-series)	시간의 흐름에 따라 값이 어떻게 변하는지 표현	추세/변동성/계절성 분석	lineplot()
분포(Distribution)	하나의 변수 값이 어떻게 펴져 있는지 표현	패턴, 이상치, 형태(정규성) 확인	boxplot(), histplot(), kdeplot(), violinplot(), ecdfplot()
범주형(Categorical)	그룹 간 차이를 비교하거나 범주의 빈도 확인	집단 비교, 비율 분석	barplot(), countplot(), violinplot(), swarmplot()
관계(Relationship)	두 변수(연속형/범주형)의 관계를 표현	상관, 패턴, 경향성 파악	scatterplot(), regplot(), lineplot(), jointplot()
행렬(Matrix)	여러 변수의 조합을 표 형태로 표현하거나 상관을 열지도 형태로 시각화	변수 상관성, 패턴 구조 파악	heatmap(), clustermap()
다변량(Multivariate)	3개 이상의 변수 간 관계를 한번에 확인	고차원 구조 탐색, 분포·관계 종합 분석	pairplot(), jointplot(), FacetGrid(), catplot(), relplot()



4. 데이터 시각화에 필요한 주요 객체

- 기본형 : matplotlib만 사용

- 응용형태 : matplotlib + pandas(DataFrame)
- 확장형 : matplotlib + seaborn

개념	비유	이유
Figure	큰 캔버스·화판	전체 그림을 담는 공간
Axes	개별 작업 구역(스케치북 페이지)	그래프가 실제로 그려지는 공간
matplotlib 기본 함수	기본 봇·연필	전부 직접 컨트롤, 자유도 높음
seaborn	고급 봇·전문 도구 세트	기본 스타일링·색감·레이아웃 자동화

visual

visual



#01. 준비작업



1. 라이브러리 참조

matplotlib, seaborn 패키지가 설치되어야 한다.

```
$ pip install --upgrade matplotlib seaborn
```

```
from hossam import load_data

# 글꼴을 시스템에 등록
from matplotlib import font_manager as fm

# 캔버스(figure)를 생성, 기본 그래픽 함수 제공
from matplotlib import pyplot as plt

# 고급 그래픽 기능 제공
import seaborn as sb
```



2. 시스템 전역 설정

아래의 코드는 컴퓨터마다 1회만 수행하면 된다.

```
font_path = "./NotoSansKR-Regular.ttf"          # 한글을 지원하는 폰트 파일
                                                # 경로
fm.fontManager.addfont(font_path)               # 폰트의 글꼴을 시스템에 등
                                                # 록함
```

```
font_prop = fm.FontProperties(fname=font_path) # 폰트의 속성을 읽어옴
font_name = font_prop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
font_name # 글꼴 이름 확인
```

'Noto Sans KR'



3. 그래프 설정

아래 코드는 ipynb 파일 하나당 한번만 수행하면 된다.

이와 같은 형식의 코드가 다시 실행되기 전까지 현재 소스파일의 모든 그래프에 전역으로 적용되는 설정.

theme 종류: whitegrid, darkgrid, dark, white(기본값)

```
my_dpi = 120 # 이미지 선명도를 결정하는 1인치
               # 당점(픽셀)의 수
my_font_name = "Noto Sans KR" # 시스템에 등록된 글꼴 이름
my_theme = "dark" # 그림 스타일 지정

sb.set_theme(style=my_theme) # seaborn 스타일 (화풍 설정하기)

plt.rcParams['font.family'] = my_font_name # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 16 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지(한글환경에서 필수)
```



#02. 그래프 기본 코드 구성

```
# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px = 1280 # 그래프 가로 크기
height_px = 760 # 그래프 세로 크기
rows = 1 # 도화지의 행 수
cols = 1 # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 그래프의 도화지 상태 확인용 테스트 코드
#print(ax)
```

```

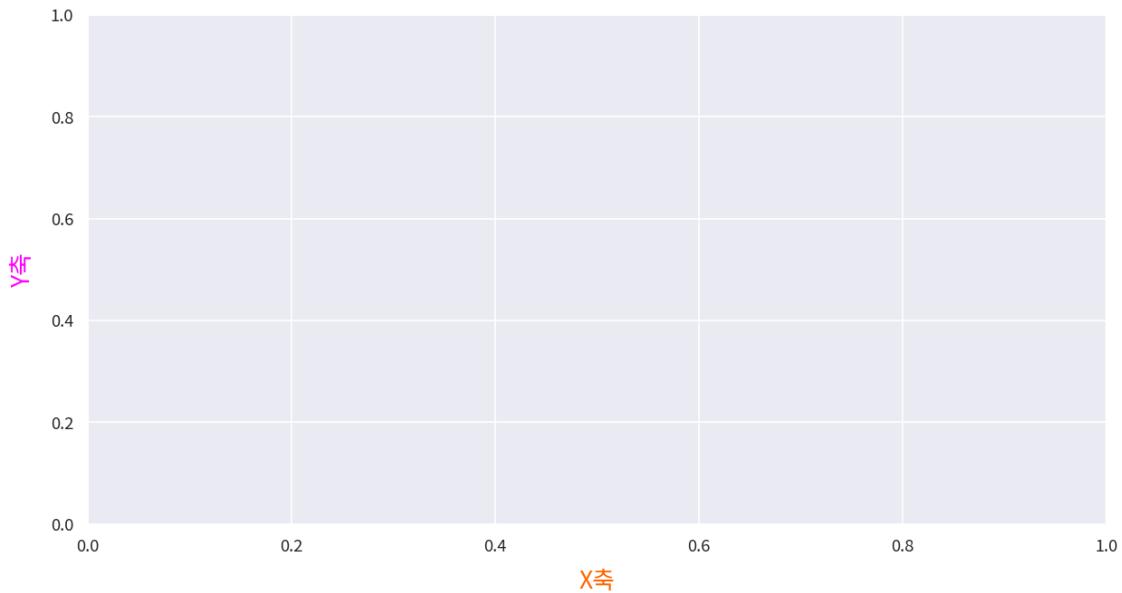
# 2) 그래프 그리기 -> seaborn 사용
# ...

# 3) 그래프 꾸미기 -> 도화지(ax)에 직접 적용
# color: 글자 색상 (기본값은 black)
# fontsize: 글자 크기 (기본값은 plt.rcParams['font.size'] 설정을 따름)
# pad, labelpad : 그래프와의 간격
# fontweight: 글자 굵기 (100~1000 사이 100단위 값). 글꼴이 지원하는 경우만 적용됨
ax.set_title("제목입니다", color="#0066ff", fontsize=22,
             fontweight=1000, pad=20)
ax.set_xlabel("X축", color="#ff6600", fontsize=16, labelpad=10)
ax.set_ylabel("Y축", color="#ff00ff", fontsize=16, labelpad=10)

# 4) 출력
plt.grid()                                     # 배경 격자 표시/숨김 (테마에 따라 다
                                              름)
plt.tight_layout()                             # 여백 제거
plt.savefig("myplot.png", dpi=my_dpi)          # 생략 가능
plt.show()                                     # 그래프 화면 출력
plt.close()                                     # 그래프 작업 종료

```

제목입니다



png



#03. Line Plot

라인 플롯은 하나의 변수가 시간의 흐름이나 순서 등에 따라 어떻게 변화하는지 보여주기 위해 사용한다.



1. 기본 그리기

그래프에 표시될 데이터를 리스트 등의 연속형 자료형으로 지정

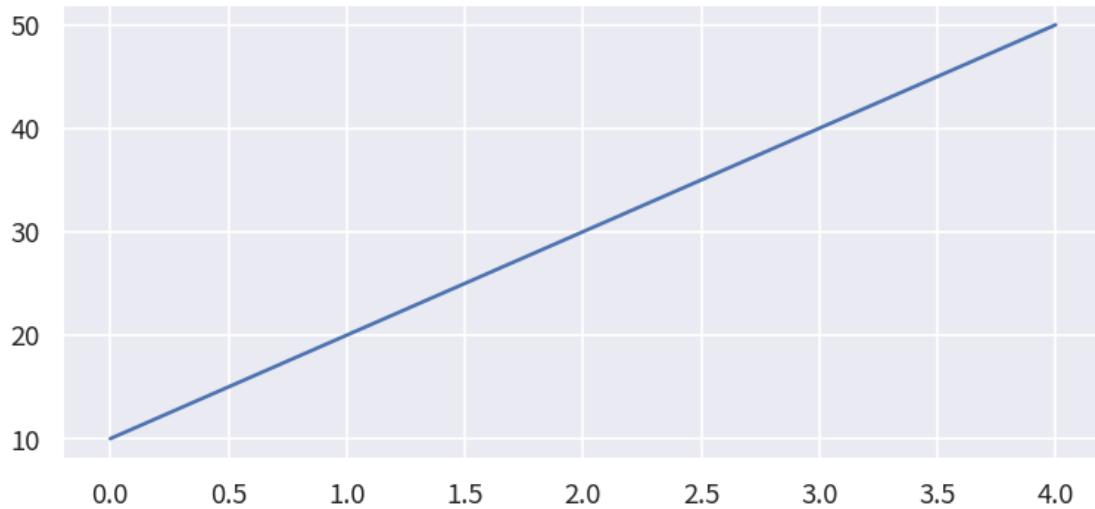
값은 y축이 되고, 인덱스는 x축이 된다.

```
# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px = 800           # 그래프 가로 크기
height_px = 400          # 그래프 세로 크기
rows = 1                  # 도화지의 행 수
cols = 1                  # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot([10, 20, 30, 40, 50])

# 3) 그래프 꾸미기 -> 여기서는 생략

# 4) 출력
plt.grid()                # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()         # 여백 제거
plt.show()                 # 그래프 화면 출력
plt.close()                # 그래프 작업 종료
```



png



2. x, y축 및 선 모양 지정하기

x축과 y축에 모두 리스트 지정

파라미터 이름	파라미터 약자	의미
color	c	선 색깔
linestyle	ls	선 스타일
linewidth	lw	선 굵기
marker		마커 종류
markersize	ms	마커 크기
markerfacecolor	mfc	마커 내부 색깔
markeredgecolor	mec	마커 선 색깔
markeredgewidth	mew	마커 선 굵기

```

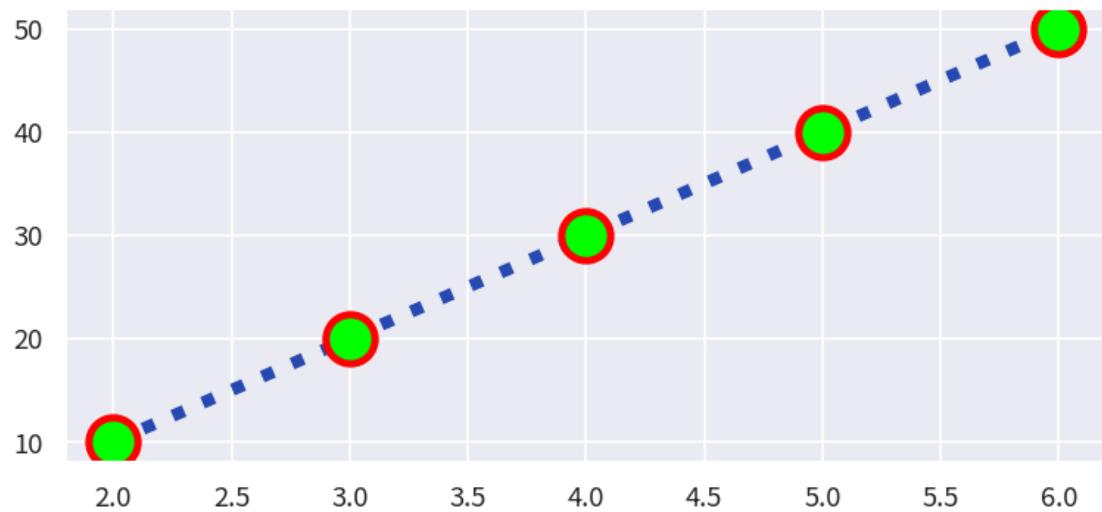
# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px = 800           # 그래프 가로 크기
height_px = 400          # 그래프 세로 크기
rows = 1                  # 도화지의 행 수
cols = 1                  # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot(x=[2, 3, 4, 5, 6], y=[10, 20, 30, 40, 50],
             c="#2548b1", linestyle=':', linewidth=5,
             marker="o", markersize=20, markerfacecolor="#00ff00",
             markeredgecolor="#ff0000", markeredgewidth=3)

# 3) 그래프 꾸미기 -> 여기서는 생략

# 4) 출력
plt.grid()                # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()         # 여백 제거
plt.show()                 # 그래프 화면 출력
plt.close()                # 그래프 작업 종료

```



png



[1] 선 스타일

선 스타일 문자열	의미
-	실선(solid)
--	대시선(dashed)
:	점선(dotted)
-.	대시-점선(dash-dit)



[2] 마커

데이터 위치를 나타내는 기호를 마커(marker)라고 한다. 마커의 종류는 다음과 같다.

마커 문자열	의미
.	point marker
,	pixel marker
o	circle marker
v	triangle_down marker
^	triangle_up marker
<	triangle_left marker
>	triangle_right marker
1	tri_down marker
2	tri_up marker
3	tri_left marker

4	tri_right marker
s	square marker
p	pentagon marker
*	star marker
h	hexagon1 marker
H	hexagon2 marker
+	plus marker
x	x marker
D	diamond marker
d	thin_diamond marker



3. 축 범위 설정

```

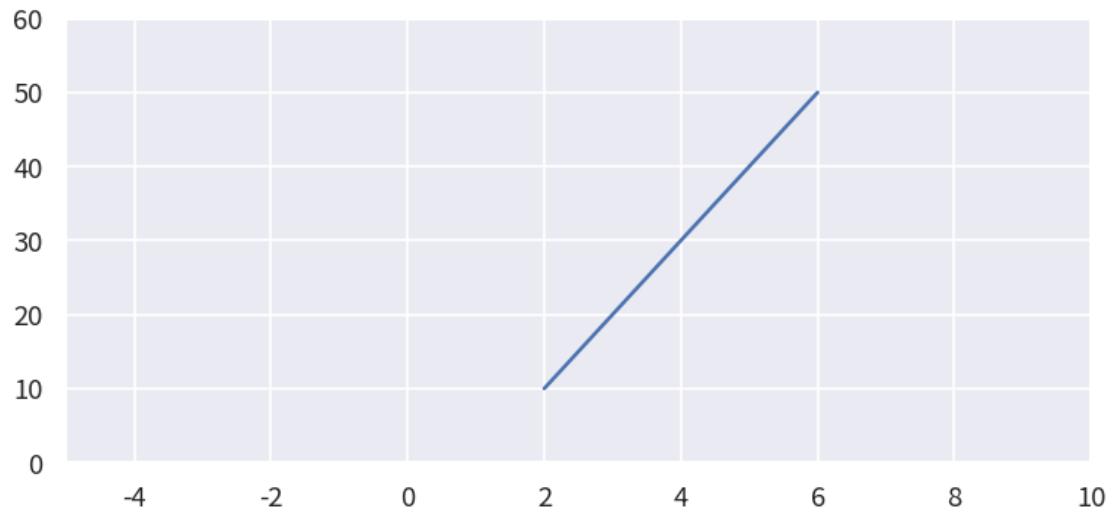
# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px = 800           # 그래프 가로 크기
height_px = 400          # 그래프 세로 크기
rows = 1                  # 도화지의 행 수
cols = 1                  # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot(x=[2, 3, 4, 5, 6], y=[10, 20, 30, 40, 50])

# 3) 그래프 꾸미기
ax.set_xlim([-5, 10])    # x축 범위
ax.set_ylim([0, 60])     # y축 범위

# 4) 출력
plt.grid()                # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()         # 여백 제거
plt.show()                 # 그래프 화면 출력
plt.close()                # 그래프 작업 종료

```



png



4. 각 축의 표시 내용 설정

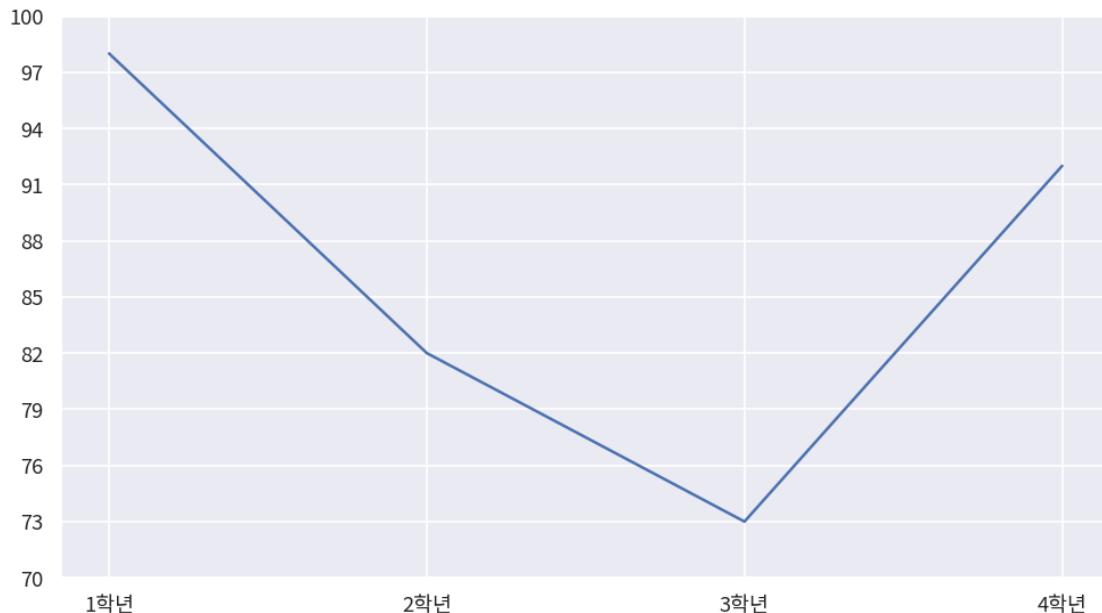
```
# 1) 그래프 초기화 (캔バス(fig)와 도화지(ax) 준비하기)
width_px    = 1024          # 그래프 가로 크기
height_px   = 640           # 그래프 세로 크기
rows = 1                  # 도화지의 행 수
cols = 1                  # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot(x=[1, 2, 3, 4], y=[98, 82, 73, 92])

# 3) 그래프 꾸미기
ax.set_title("철수의 학년별 평균 점수 변화", pad=15, fontsize=24)
# x축 좌표에 따른 표시할 문자열 지정
ax.set_xticks([1, 2, 3, 4], ['1학년', '2학년', '3학년', '4학년'])
ax.set_yticks(range(70, 101, 3), range(70, 101, 3))

# 4) 출력
plt.grid()                 # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료
```

철수의 학년별 평균 점수 변화



png

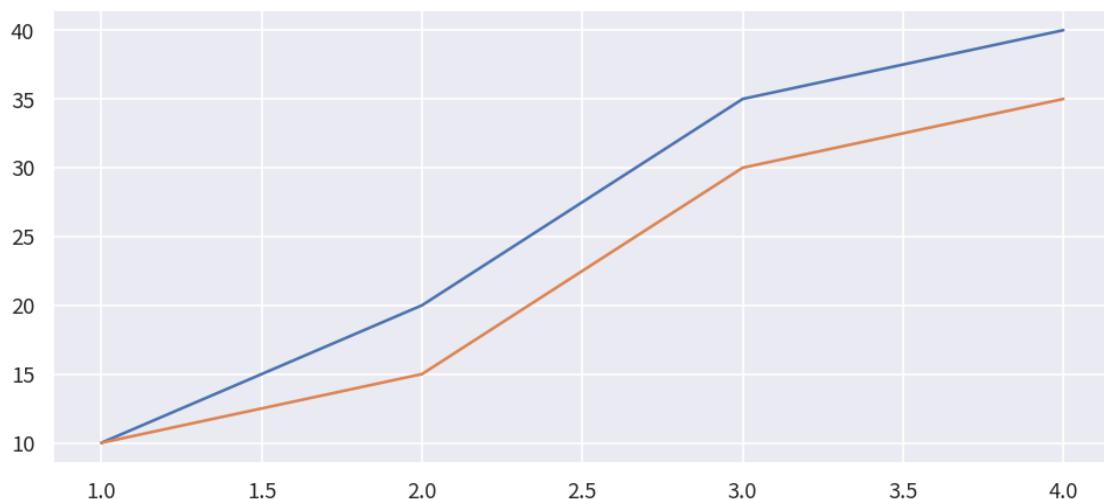
■ #04. 다중 선 그래프

```
# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px  = 1000          # 그래프 가로 크기
height_px = 480           # 그래프 세로 크기
rows      = 1              # 도화지의 행 수
cols      = 1              # 도화지의 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기 -> seaborn 사용
sb.lineplot(x=[1, 2, 3, 4], y=[10, 20, 35, 40])
sb.lineplot(x=[1, 2, 3, 4], y=[10, 15, 30, 35])

# 3) 그래프 꾸미기 (생략)

# 4) 출력
plt.grid()                 # 배경 격자 표시/숨김 (테마에 따라 다름)
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료
```



png

#05. 예제: 교통사고 발생건수 시각화



1. 데이터 가져오기

```
origin = load_data('traffic_acc')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab04/traffic_acc.xlsx
[94m[desc] [0m 2005년 1월부터 2018년 12월까지 월별 교통사고의 발생건수, 부상자
수, 사망자수 데이터(인덱스/메타데이터 없음, 출처: 공공데이터포털)
[91m[!] Cannot read metadata [0m
```

	년도	월	발생건수	사망자수	부상자수
0	2005	1	15494	504	25413
1	2005	2	13244	431	21635
2	2005	3	16580	477	25550
3	2005	4	17817	507	28131
4	2005	5	19085	571	29808
...
163	2018	8	18335	357	27749
164	2018	9	18371	348	27751
165	2018	10	19738	373	28836

166	2018	11	19029	298	28000
167	2018	12	18010	323	26463

168 rows × 5 columns



2. 데이터 전처리

```
df = origin.drop('월', axis=1).groupby('년도').mean()
df
```

	발생건수	사망자수	부상자수
년도			
2005	17847.583333	531.333333	28519.416667
2006	17812.083333	527.250000	28352.416667
2007	17638.500000	513.833333	27992.166667
2008	17985.166667	489.166667	28246.833333
2009	19332.500000	486.500000	30156.250000
2010	18906.500000	458.750000	29371.500000
2011	18475.916667	435.750000	28449.250000
2012	18638.000000	449.333333	28713.750000
2013	17946.166667	424.333333	27392.583333
2014	18629.333333	396.833333	28124.750000
2015	19336.250000	385.083333	29200.000000
2016	18409.750000	357.666667	27643.333333
2017	18027.916667	348.750000	26902.416667
2018	18095.666667	315.083333	26919.750000



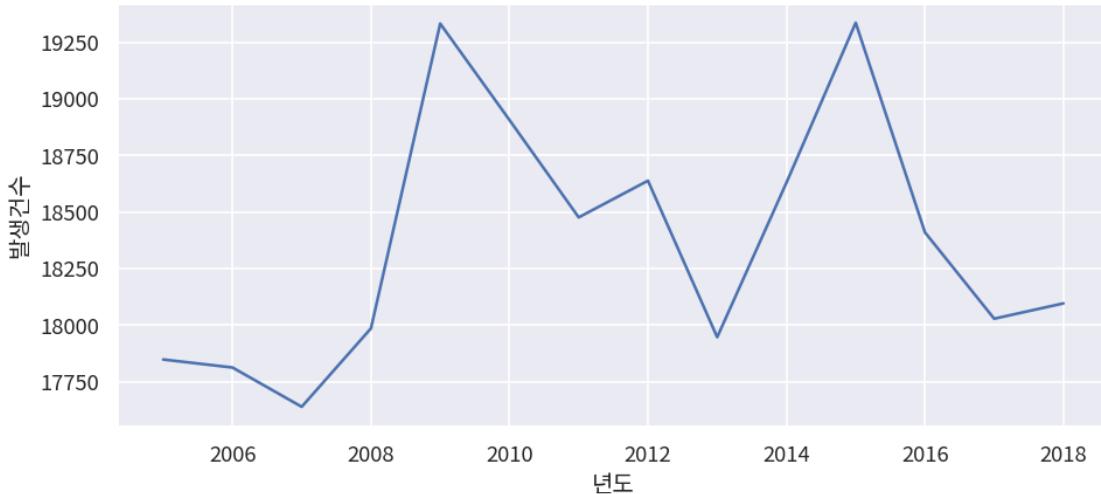
3. 교통사고 발생자 수 변화

```
# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px = 1000           # 그래프 가로 크기
height_px = 480            # 그래프 세로 크기
rows = 1                   # 도화지의 행 수
cols = 1                   # 도화지의 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

```
# 2) 그래프 그리기 -> seaborn 사용  
sb.lineplot(data=df, x=df.index, y='발생건수')
```

```
# 3) 그래프 꾸미기 (생략)
```

```
# 4) 출력  
plt.grid() # 배경 격자 표시/숨김 (테마에 따라 다름)  
plt.tight_layout() # 여백 제거  
plt.show() # 그래프 화면 출력  
plt.close() # 그래프 작업 종료
```



png



연습문제



1. Covid19 확진자수 변동 추이 시각화

covid19_active 데이터는 2022년 5월 1일부터 2023년 5월 31일까지 서울과 전국의 Covid19 일일 확진자 수를 기록한 데이터이다. 조사 기간동안 서울과 전국의 확진자 수가 어떻게 변화하고 있는지에 대한 추이를 시각화 하고 시각화 결과에서 얻을 수 있는 객관적 사실을 하나 이상 서술하시오.

단, x축에 표시되는 날짜는 30일 간격으로 표시한다.



2. 비트코인 시세 변동 추이 시각화

bitcoin 데이터는 2021년 06월 01일부터 2023년 06월 30일까지의 비트코인 시세 데이터의 일부이다.

이 데이터를 활용하여 날짜별 종가와 시가가 어떻게 변화하고 있는지 보여주고자 한다. 단, x축의 간격을 20일 간격으로 설정하여 시각화 하고 시각화 결과에서 얻을 수 있는 객관적 사실을 하나 이상 서술하시오.



[LAB-06] 11. 서브플롯 (1)

하나의 그래픽 영역을 나누어 두 개 이상의 시각화 결과물을 하나의 화면에서 표현할 수 있다.



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```



2. 그래프 전역 설정

```
my_dpi = 200
fpath = "./NotoSansKR-Regular.ttf"
fm.fontManager.addfont(fpath)
fprop = fm.FontProperties(fname=fpath)
fname = fprop.get_name()
    출
plt.rcParams['font.family'] = fname      # 이미지 선명도(100~300)
plt.rcParams['font.size'] = 6            # 한글을 지원하는 폰트 파일의 경로
plt.rcParams['axes.unicode_minus'] = False # 폰트의 글꼴을 시스템에 등록함
                                         # 폰트의 속성을 읽어옴
                                         # 읽어온 속성에서 폰트의 이름만 추
                                         # 그레프에 한글 폰트 적용
                                         # 기본 폰트 크기
                                         # 그레프에 마이너스 깨짐 방지
```



3. 데이터 가져오기

```
origin = load_data('traffic_acc')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab04/traffic_acc.xlsx
[94m[desc] [0m 2005년 1월부터 2018년 12월까지 월별 교통사고의 발생건수, 부상자
수, 사망자수 데이터(인덱스/메타데이터 없음, 출처: 공공데이터포털)
[91m[!] Cannot read metadata [0m
```

	년도	월	발생건수	사망자수	부상자수
--	----	---	------	------	------

0	2005	1	15494	504	25413
1	2005	2	13244	431	21635
2	2005	3	16580	477	25550
3	2005	4	17817	507	28131
4	2005	5	19085	571	29808
...
163	2018	8	18335	357	27749
164	2018	9	18371	348	27751
165	2018	10	19738	373	28836
166	2018	11	19029	298	28000
167	2018	12	18010	323	26463

168 rows × 5 columns



4. 데이터 전처리

각 변수를 년도별 평균값으로 전처리 한다.

```
df = origin.drop('월', axis=1).groupby('년도').mean()
df
```

	발생건수	사망자수	부상자수
년도			
2005	17847.583333	531.333333	28519.416667
2006	17812.083333	527.250000	28352.416667
2007	17638.500000	513.833333	27992.166667
2008	17985.166667	489.166667	28246.833333
2009	19332.500000	486.500000	30156.250000
2010	18906.500000	458.750000	29371.500000
2011	18475.916667	435.750000	28449.250000
2012	18638.000000	449.333333	28713.750000
2013	17946.166667	424.333333	27392.583333
2014	18629.333333	396.833333	28124.750000
2015	19336.250000	385.083333	29200.000000
2016	18409.750000	357.666667	27643.333333

2017	18027.916667	348.750000	26902.416667
2018	18095.666667	315.083333	26919.750000

#02. 서브플롯의 기본 사용

[1] 서브플롯 영역 나누기

2행 3열을 갖는 서브플롯 영역을 구성한다.

```
pyplot.subplots(행, 열 [, figsize=(가로크기, 세로크기)])
```

`plt.subplots()` 메서드에 의해 리턴되는 `fig`는 그래픽 처리 기능을 제공하는 객체이다.

`plt.subplots()` 메서드에 의해 리턴되는 `ax` 객체는 분할된 각 그래프 영역의 객체를 저장하고 있는 리스트이다.

```
# 1) 그래프 초기화
width_px    = 2000                      # 그래프 가로 크기
height_px   = 1000                       # 그래프 세로 크기
rows         = 2                          # 그래프 행 수
cols         = 3                          # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)

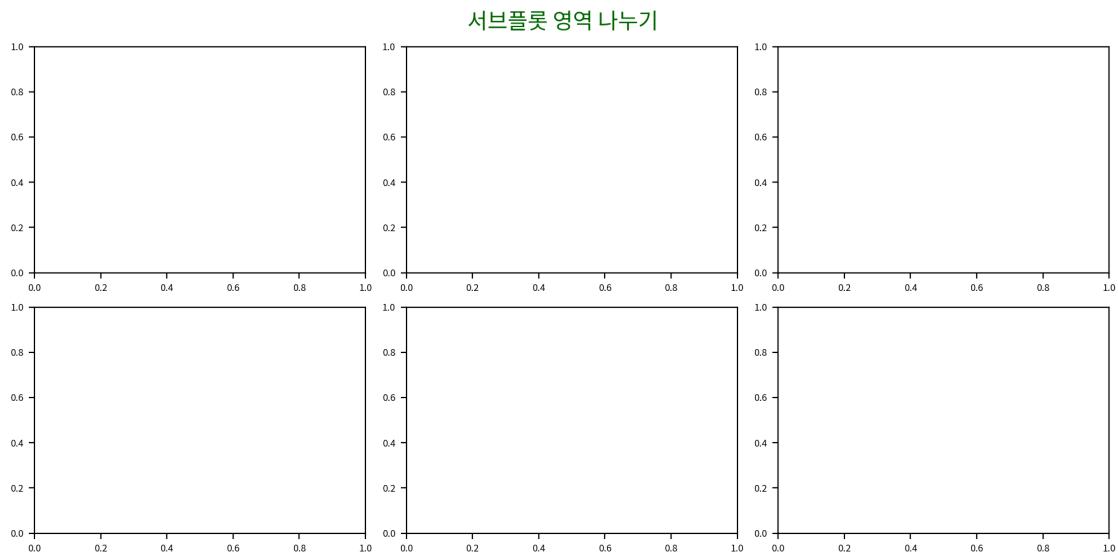
# ax 객체가 행, 열 수에 따라서 리스트가 된다.
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
# ...

# 3) 그래프 꾸미기
# 전체 제목
fig.suptitle('서브플롯 영역 나누기', fontsize=14, color='#006600')

# 각 그래프 간의 가로(wspace), 세로(hspace) 간격 지정
fig.subplots_adjust(wspace=0.2, hspace=0.2)

# 4) 출력
plt.tight_layout()                      # 여백 제거
plt.show()                             # 그래프 화면 출력
plt.close()                            # 그래프 작업 종료
```



png



[2] 서브플롯에 그래프 그리기

`plt.subplots()` 메서드의 결과로 `ax`에 반환되는 객체는 서브플롯의 행, 열에 대한 리스트이다.

```

# 1) 그래프 초기화
width_px    = 2400          # 그래프 가로 크기
height_px   = 1000          # 그래프 세로 크기
rows         = 2              # 그래프 행 수
cols         = 3              # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)
fig, ax      = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.boxplot(data=df, y='발생건수', ax=ax[0][0])
sb.histplot(data=df, x='사망자수', bins=5, ax=ax[0][1])
sb.kdeplot(data=df, x='부상자수', ax=ax[0][2])
sb.lineplot(data=df, x=df.index, y='발생건수', ax=ax[1][0])
sb.barplot(data=df, x=df.index, y='사망자수', estimator=np.sum,
           ax=ax[1][1])
sb.regplot(data=df, x='발생건수', y='사망자수', ax=ax[1][2])

# 3) 그래프 꾸미기
# 전체 제목
fig.suptitle('년도별 교통사고 집계', fontsize=14, color='#006600')

# 각 그래프 간의 가로(wspace), 세로(hspace) 간격 지정
fig.subplots_adjust(wspace=0.2, hspace=0.2)

# 첫 번째 영역 그래프의 제목, 글자크기, 색상, 격자

```

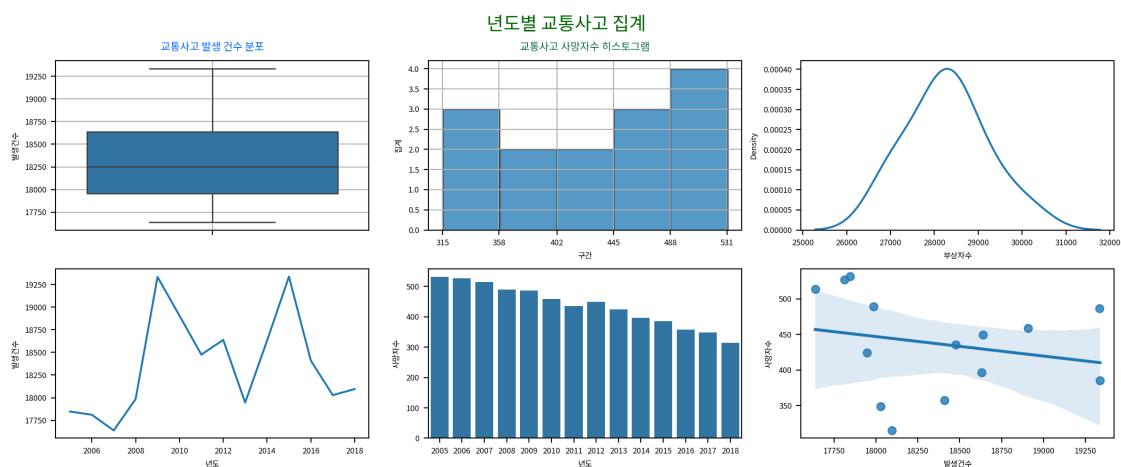
```

ax[0][0].set_title("교통사고 발생 건수 분포", color="#0066ff", fontsize=8,
                    pad=8)
ax[0][0].grid()

# 두 번째 영역 그래프의 x축 설정 및 x,y축 라벨 지정, 격자
hist, bins = np.histogram(df['사망자수'], bins=5)
bins = bins.round().astype("int")
ax[0][1].set_title("교통사고 사망자수 히스토그램", color="#0f6a46",
                    fontsize=8, pad=8)
ax[0][1].set_xticks(bins, bins)
ax[0][1].set_xlabel('구간')
ax[0][1].set_ylabel('집계')
ax[0][1].grid()

# 4) 출력
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료

```



png

#03. 두 개의 y축을 갖는 그래프



1. 샘플 데이터 만들기

1) x축 데이터 (공용)

```

x = np.arange(10)
x

```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

2) 첫 번째 y축 데이터

```
y1 = np.arange(10)  
y1
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

3) 두 번째 y축 데이터

```
y2 = x**2  
y2
```

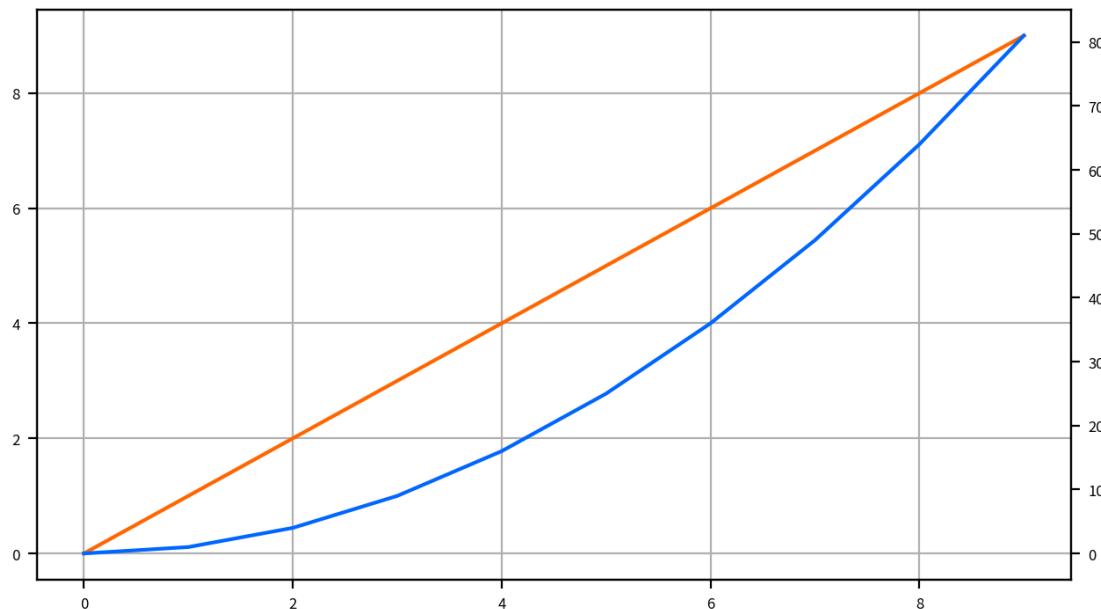
```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
```



2. 서브플롯으로 2개의 y축을 갖는 그래프 구현

```
# 1) 그래프 초기화  
width_px = 1280 # 그래프 가로 크기  
height_px = 720 # 그래프 세로 크기  
rows = 1 # 그래프 행 수  
cols = 1 # 그래프 열 수  
figsize = (width_px / my_dpi, height_px / my_dpi)  
fig, ax1 = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)  
  
# ax1에 겹쳐지는 쌍둥이 서브플롯을 생성  
ax2 = ax1.twinx()  
  
# 2) LinePlot 그리기  
sb.lineplot(x=x, y=y1, color='#ff6600', ax=ax1)  
sb.lineplot(x=x, y=y2, color='#0066ff', ax=ax2)  
  
# 3) 그래프 꾸미기  
ax1.grid(True) # 그래프가 겹쳐지므로 격자는 하나만 표시해도 됨  
  
# 4) 출력  
plt.tight_layout() # 여백 제거
```

```
plt.show()          # 그래프 화면 출력  
plt.close()        # 그래프 작업 종료
```



png



04. 교통사고 발생건수와 사망자수 변화 시각화하기

우리나라는 2008년도에 자동차안전기준에 관한 규칙 일부개정령(안)을 개정한 이후 꾸준히 교통사고안전기준을 강화해 왔다.

이러한 노력이 교통사고 부상자수를 줄이는데 효과가 있었는지 알아보자.

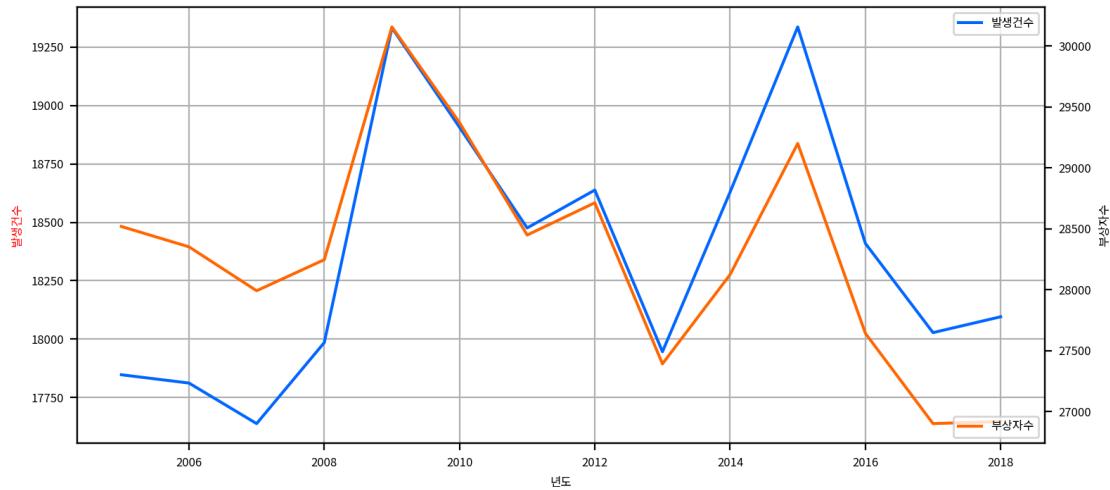
```
# 1) 그래프 초기화  
width_px  = 1600                      # 그래프 가로 크기  
height_px = 720                        # 그래프 세로 크기  
rows     = 1                            # 그래프 행 수  
cols     = 1                            # 그래프 열 수  
figsize  = (width_px / my_dpi, height_px / my_dpi)  
fig, ax1 = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)  
  
# ax1에 겹쳐지는 쌍둥이 서브플롯을 생성  
ax2 = ax1.twinx()  
  
# 2) LinePlot 그리기  
sb.lineplot(data=df, x=df.index, y='발생건수', color='#0066ff',  
            ax=ax1, label='발생건수')  
sb.lineplot(data=df, x=df.index, y='부상자수', color='ffd600',  
            ax=ax2, label='부상자수')
```

```

# 3) 그래프 꾸미기
ax1.set_xlabel('년도')
ax1.set_ylabel('발생건수', color='ff0000')
ax1.grid(True)           # 그래프가 겹쳐지므로 격자는 하나만 표시해도 됨
ax1.legend(loc='upper right')
ax2.legend(loc='lower right')

# 4) 출력
plt.tight_layout()      # 여백 제거
plt.show()               # 그래프 화면 출력
plt.close()              # 그래프 작업 종료

```



png

해석

개정안이 시행되기 전(2008년)에는 교통사고 발생건수 대비 부상자 수의 비율이 더 많았지만 개정안이 시행된 후에는 교통사고 발생건수 대비 부상자수의 비율이 현저히 낮아졌다.



전국 실업률 분포 변화

unemployment_age 데이터는 2000년부터 2022년까지 행정구역(시도)/연령별 실업률을 담고 있다.

이 데이터를 토대로 지역별 실업률 어떻게 변화하고 있는지 확인해 보자.



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
from pandas import melt, pivot_table
import seaborn as sb
import numpy as np
```



[2] 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



[3] 데이터 가져오기

0번째 열과 1번째 열을 복수 인덱스로 지정

```
origin = load_data('unemployment_age')
origin.head()
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/
unemployment_age.xlsx
```

[94m[desc] [0m 2000년부터 2022년까지 행정구역(시도)/연령별 실업률 데이터 (출처:
국가통계포털)]

[91m[!] Cannot read metadata [0m]

	시도 별	연령계 층별	2000	2001	2002	2003	2004	2005	2006	2007	…	2013	2014	2015	2016
0	서울 특별 시	15-29 세	8.1	8.4	8.2	8.8	8.9	8.9	8.8	7.4	…	8.7	10.4	9.3	10.3
1	NaN	30-59 세	3.9	3.6	3.0	3.2	3.4	3.6	3.4	3.2	…	3.0	3.2	3.1	2.9
2	NaN	60세 이상	2.9	1.9	2.3	1.9	2.1	2.1	2.6	1.7	…	2.5	2.8	3.2	2.8
3	부산 광역 시	15-29 세	12.1	10.7	7.6	9.0	9.9	8.8	8.2	8.3	…	8.8	9.0	9.7	9.9
4	NaN	30-59 세	5.5	4.2	2.9	2.9	3.1	3.3	3.2	3.0	…	3.0	3.0	3.1	2.8

5 rows × 25 columns



#02. 데이터 전처리



[1] 시 이름에 대한 결측치 처리

```
df = origin.copy()
df['시도별'] = df['시도별'].ffill()
df.head(10)
```

	시 도 별	연령계 층별	2000	2001	2002	2003	2004	2005	2006	2007	…	2013	2014	2015	2016	20
0	서 울 특 별 시	15-29 세	8.1	8.4	8.2	8.8	8.9	8.9	8.8	7.4	…	8.7	10.4	9.3	10.3	10.
1	서 울 특	30-59 세	3.9	3.6	3.0	3.2	3.4	3.6	3.4	3.2	…	3.0	3.2	3.1	2.9	3.3

	별 시																				
2	서 울 특 별 시	60세 이상	2.9	1.9	2.3	1.9	2.1	2.1	2.6	1.7	...	2.5	2.8	3.2	2.8	3.6					
3	부 산 광 역 시	15-29 세	12.1	10.7	7.6	9.0	9.9	8.8	8.2	8.3	...	8.8	9.0	9.7	9.9	11.1					
4	부 산 광 역 시	30-59 세	5.5	4.2	2.9	2.9	3.1	3.3	3.2	3.0	...	3.0	3.0	3.1	2.8	3.2					
5	부 산 광 역 시	60세 이상	5.3	3.6	2.1	1.5	1.0	2.8	3.1	2.8	...	2.6	2.4	2.6	2.8	3.6					
6	대 구 광 역 시	15-29 세	9.1	9.8	9.2	9.8	8.6	8.7	9.5	8.9	...	9.9	11.5	10.1	12.0	11.1					
7	대 구 광 역 시	30-59 세	3.7	3.5	2.8	3.0	3.2	3.2	2.5	2.4	...	2.1	2.5	2.2	2.5	2.7					
8	대 구 광 역 시	60세 이상	0.7	1.2	1.4	2.2	1.6	2.5	1.6	2.0	...	2.2	2.1	2.5	3.8	3.4					
9	인 천 광 역 시	15-29 세	8.2	8.2	7.9	8.8	8.3	8.3	9.2	8.3	...	9.3	12.1	11.8	11.5	10.					

10 rows × 25 columns



#03. 시각화



[1] 연도에 따른 전국 평균 실업률 변화

(1) 데이터 전처리

데이터 재구조화

```
df2 = melt(df, id_vars=['시도별', '연령계층별'],
            var_name='년도', value_name='실업률')
df2.head(10)
```

	시도별	연령계층별	년도	실업률
0	서울특별시	15-29세	2000	8.1
1	서울특별시	30-59세	2000	3.9
2	서울특별시	60세이상	2000	2.9
3	부산광역시	15-29세	2000	12.1
4	부산광역시	30-59세	2000	5.5
5	부산광역시	60세이상	2000	5.3
6	대구광역시	15-29세	2000	9.1
7	대구광역시	30-59세	2000	3.7
8	대구광역시	60세이상	2000	0.7
9	인천광역시	15-29세	2000	8.2

시/도에 따른 연도별 평균 실업률

```
tdf1 = df2[['시도별', '년도', '실업률']].groupby(['시도별', '년도'],
                                                as_index=False).mean()
tdf1
```

	시도별	년도	실업률
0	강원도	2000	2.766667
1	강원도	2001	2.333333
2	강원도	2002	2.100000
3	강원도	2003	2.566667
4	강원도	2004	2.633333

...
386	충청북도	2018	3.400000
387	충청북도	2019	4.200000
388	충청북도	2020	4.466667
389	충청북도	2021	3.433333
390	충청북도	2022	3.200000

391 rows × 3 columns

전국에 대한 연도별 평균 실업률

```
tdf2 = tdf1[['년도', '실업률']].groupby('년도').mean()
tdf2
```

	실업률
연도	
2000	4.103922
2001	3.762745
2002	3.154902
2003	3.437255
2004	3.582353
2005	3.584314
2006	3.488235
2007	3.343137
2008	3.282353
2009	3.684314
2010	3.945098
2011	3.635294
2012	3.558824
2013	3.600000
2014	4.127451
2015	4.207843
2016	4.476471
2017	4.558824
2018	4.819608

2019	4.849020
2020	4.988235
2021	4.368627
2022	3.666667

(2) 데이터 시각화

```

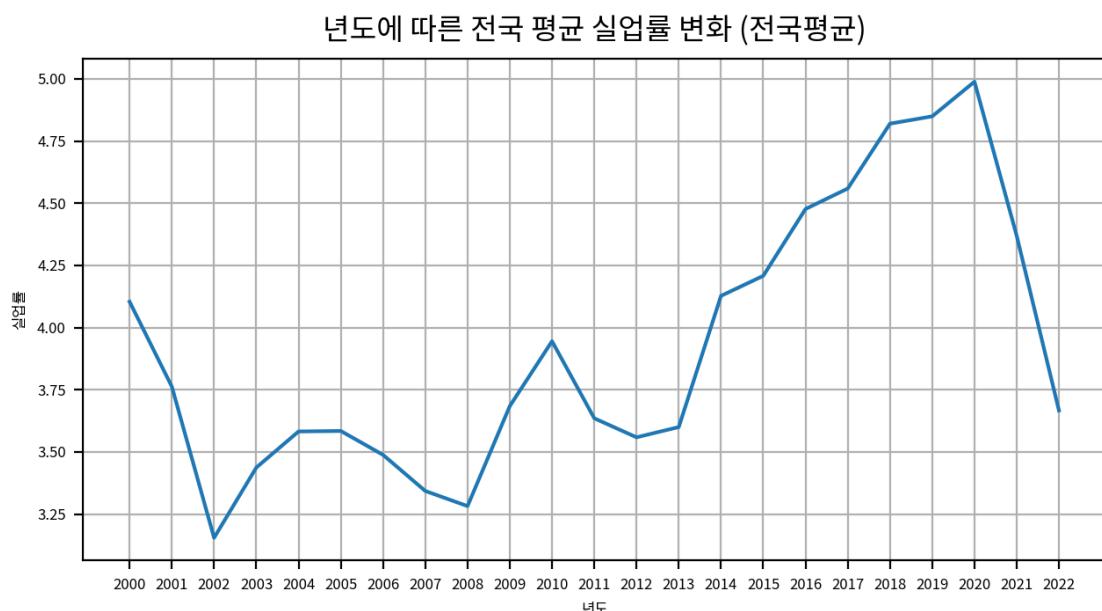
# 1) 그래프 초기화
width_px  = 1280                         # 그래프 가로 크기
height_px = 720                           # 그래프 세로 크기
rows      = 1                               # 그래프 행 수
cols      = 1                               # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax  = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) LinePlot 그리기
sb.lineplot(data=tdf2, x=tdf2.index, y="실업률")

# 3) 그래프 꾸미기
ax.set_title("년도에 따른 전국 평균 실업률 변화 (전국평균)", fontsize=12, pad=8)
ax.grid(True)                                # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                          # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)     # 그래프 화면 출력
plt.show()                                   # 그래프 작업 종료

```



png

- 2000년 이후 증감을 반복하는 모습을 보이지만 전체적으로 증가하는 추세.
- 2020년 이후 급격히 감소함



[2] 연도에 따른 연령대별 전국 평균 실업률 변화

(1) 데이터 전처리

앞에서 만든 데이터 재구조화 결과에 이어서 진행

데이터 그룹별 집계

```
gdf = df2.filter(['년도', '연령계층별', '실업률']).groupby(['년도', '연령계  
층별'], as_index=False).mean()  
gdf
```

	년도	연령계층별	실업률
0	2000	15-29세	7.735294
1	2000	30-59세	3.052941
2	2000	60세이상	1.523529
3	2001	15-29세	7.435294
4	2001	30-59세	2.605882
...
64	2021	30-59세	2.364706
65	2021	60세이상	3.570588
66	2022	15-29세	6.205882
67	2022	30-59세	1.982353
68	2022	60세이상	2.811765

69 rows × 3 columns

```
# 1) 그래프 초기화  
width_px = 1280 # 그래프 가로 크기  
height_px = 720 # 그래프 세로 크기  
rows = 1 # 그래프 행 수  
cols = 1 # 그래프 열 수  
figsize = (width_px / my_dpi, height_px / my_dpi)  
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

```

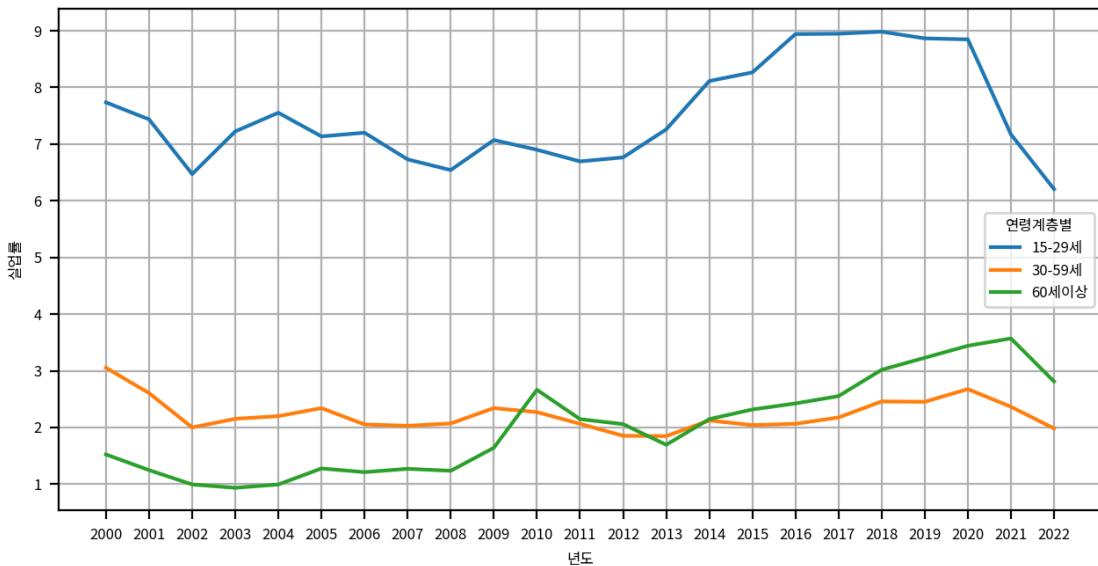
# 2) LinePlot 그리기
sb.lineplot(data=gdf, x='년도', y='실업률', hue='연령계층별')

# 3) 그래프 꾸미기
ax.set_title("년도에 따른 전국 평균 실업률 변화 (연령대별)", fontsize=12, pad=8)
ax.grid(True) # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout() # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```

년도에 따른 전국 평균 실업률 변화 (연령대별)



png

■ #04. 두 그래프를 서브플롯으로 구현

```

# 1) 그래프 초기화
width_px = 2400 # 그래프 가로 크기
height_px = 800 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 2 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)

# ax 객체가 행, 열 수에 따라서 리스트가 된다.
# [fig, [ax1, ax2]]]
fig, (ax1, ax2) = plt.subplots(rows, cols, figsize=figsize,
                               dpi=my_dpi)

```

```

# 2) 그래프 그리기
sb.lineplot(data=tdf2, x=tdf2.index, y="실업률", ax=ax1)
sb.lineplot(data=gdf, x='년도', y='실업률', hue='연령계층별', ax=ax2)

# 3) 그래프 꾸미기
# 전체 제목
fig.suptitle('년도에 따른 전국 평균 실업률 변화', fontsize=14,
             color="#006600")

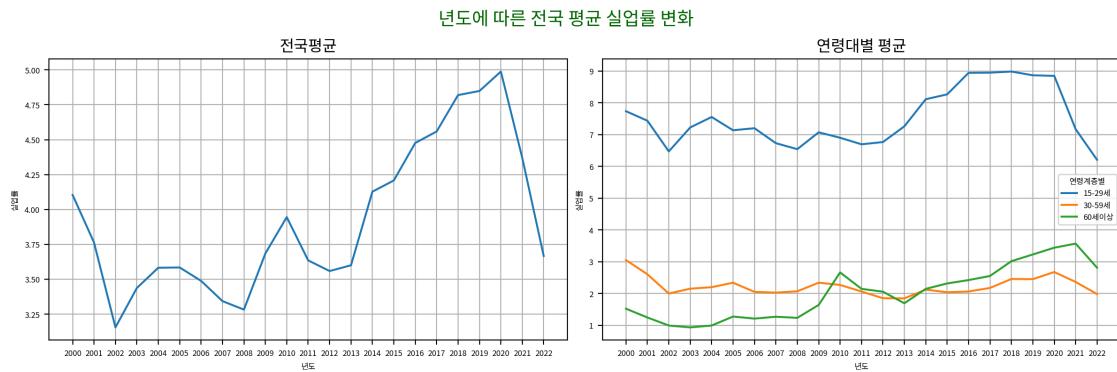
# 각 그래프 간의 가로(wspace), 세로(hspace) 간격 지정
fig.subplots_adjust(wspace=0.2, hspace=0.2)

ax1.title.set_text("전국평균")
ax1.title.set_fontsize(12)
ax1.grid()

ax2.title.set_text("연령대별 평균")
ax2.title.set_fontsize(12)
ax2.grid()

# 4) 출력
plt.tight_layout()      # 여백 제거
plt.show()               # 그래프 화면 출력
plt.close()              # 그래프 작업 종료

```



png



[LAB-06] 2. 데이터 분포 시각화 (1) - Boxplot, KDE



데이터 분포 시각화 종류

그래프 유형	seaborn 함수	설명 / 목적
박스플롯	boxplot()	사분위수 기반 분포 요약
KDE 곡선	kdeplot()	연속형 분포의 밀도
히스토그램	histplot()	연속형 변수의 분포 확인
히스토그램 + KDE	displot(kind="hist")/ histplot(kde=True)	분포 + 패턴
바이올린 플롯	violinplot()	분포 + 중앙값 + 퍼짐
스트립플롯	stripplot()	분포의 개별 관측값
스웜플롯	swarmplot()	겹치지 않는 스트립(마커) 분포
히트맵	heatmap()	복수 변수의 빈도/상관 분석



#01. 준비작업



1. 라이브러리 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```



2. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
font_path = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(font_path) # 폰트의 글꼴을 시스템에 등록
font_prop = fm.FontProperties(fname=font_path) # 폰트의 속성을 읽어옴
```

```

font_name = font_prop.get_name()          # 읽어온 속성에서 폰트의 이름
    만 추출
plt.rcParams['font.family'] = font_name   # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 10             # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지

```



3. 데이터 가져오기

```

origin = load_data("employee_data_40")
origin.head()

```

[94m[**data**] [0m https://data.hossam.kr/data/lab06/
employee_data_40.xlsx
[94m[**desc**] [0m 어느 기업의 직원 40명을 대상으로 성별과 결혼상태, 나이, 최종학력,
월수입을 조사한 가상의 데이터(인덱스, 메타데이터 없음)
[91m[!] Cannot read metadata [0m

	성별	결혼상태	나이	최종학력	월수입
0	남자	기혼	21	대학교	60
1	남자	기혼	22	대학원	100
2	남자	기혼	33	대학교	200
3	여자	미혼	33	대학교	120
4	남자	미혼	28	대학교	70



#02. Boxplot



1) 연속형 데이터의 분포를 사분위수 기반으로 확인

list, ndarray, Series 등 모든 연속형 객체를 data 파라미터에 지정한다.

orient로 방향을 설정할 수 있다.

- v : 세로 (기본값)
- h : 가로

```

# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px = 800                      # 그래프 가로 크기
height_px = 350                       # 그래프 세로 크기

```

```

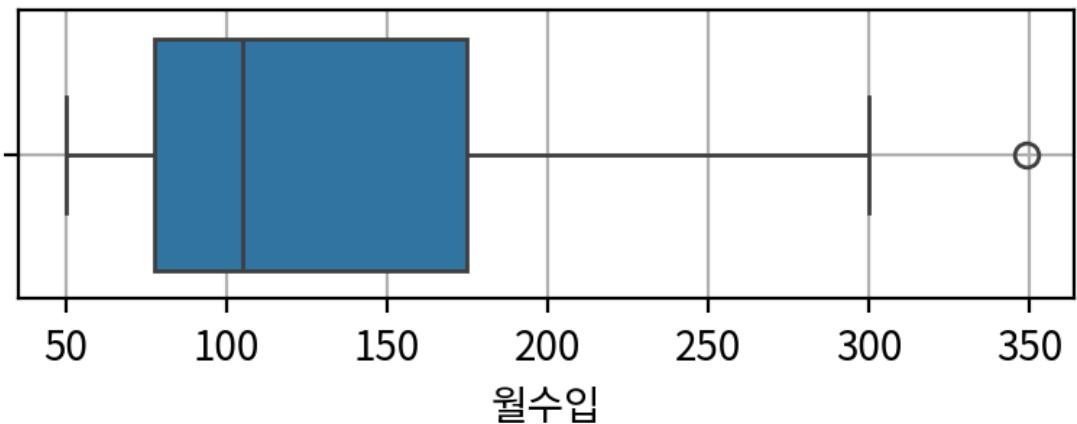
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.boxplot(data=origin['월수입'], orient="h")

# 3) 그래프 꾸미기
ax.grid(True) # 격자 표시

# 4) 출력
plt.tight_layout() # 여백 제거
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png



2. 데이터프레임을 통한 상자그림

data 파라미터에 데이터 프레임을 설정하고 y 파라미터에 표시하고자 하는 변수 이름을 문자열로 설정한다.

x 파라미터에 설정할 경우 가로 상자그림으로 표시된다.

```

# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px = 800 # 그래프 가로 크기
height_px = 350 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

```

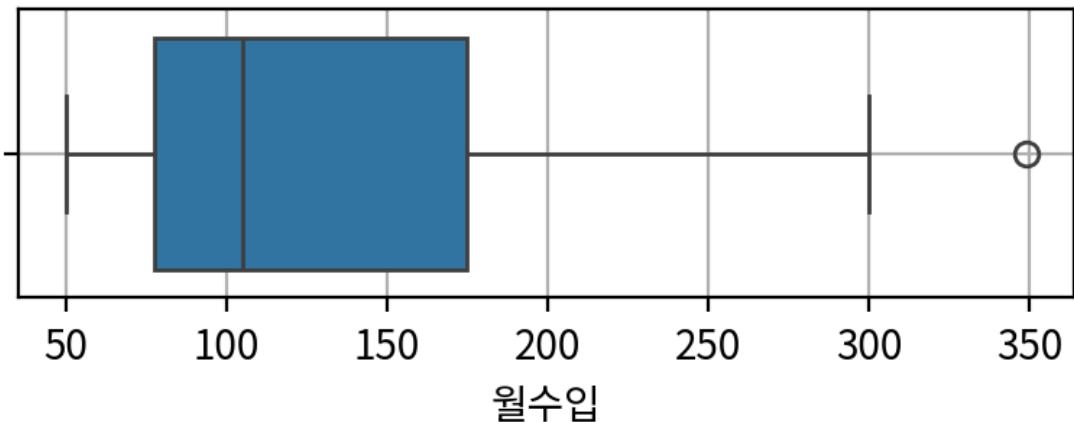
```

# 2) 그래프 그리기
sb.boxplot(data=origin, x='월수입')

# 3) 그래프 꾸미기
ax.grid(True) # 격자 표시

# 4) 출력
plt.tight_layout() # 여백 제거
#plt.savefig("myplot.png", dpi=my_dpi) # 생략 가능
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png



3) 복수 변수에 대한 처리

표시하고자 하는 변수를 필터링하여 data 파라미터에 설정한다.

```

# 1) 그래프 초기화 (캔버스(fig)와 도화지(ax) 준비하기)
width_px = 800 # 그래프 가로 크기
height_px = 500 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

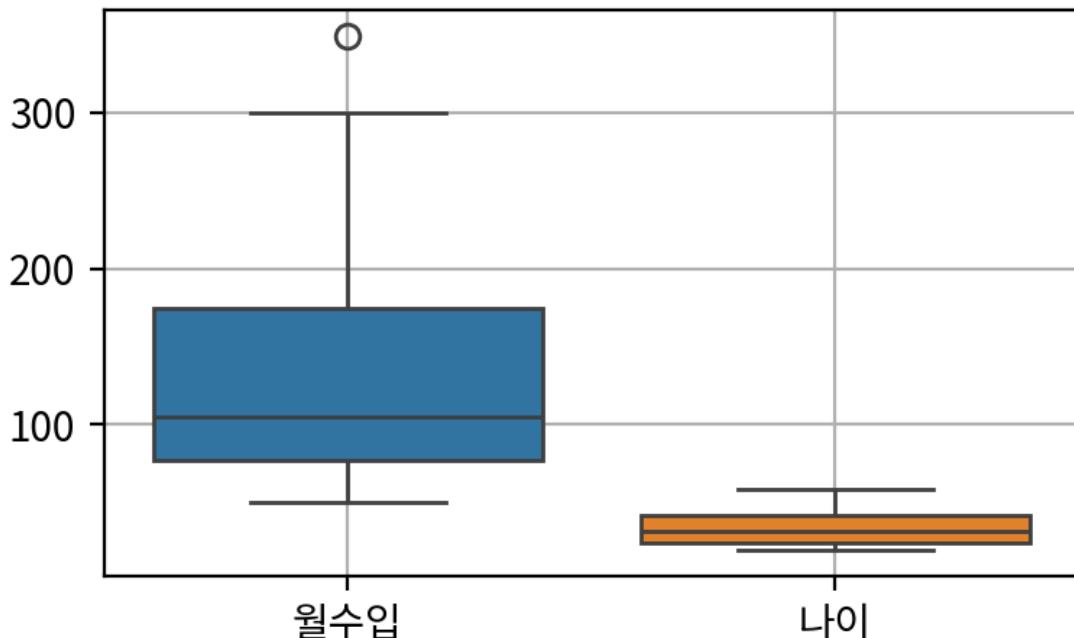
# 2) 그래프 그리기
sb.boxplot(data=origin[['월수입', '나이']])

# 3) 그래프 꾸미기
ax.grid(True) # 격자 표시

# 4) 출력

```

```
plt.tight_layout()          # 여백 제거  
plt.show()                # 그래프 화면 출력  
plt.close()               # 그래프 작업 종료
```



png

📘 #03. KDE(커널 밀도 추정) Plot

- 연속형 데이터의 분포를 부드러운 곡선 형태로 추정하는 비모수적 방법.
- 데이터의 전체적인 분포 모양, 봉우리(peak), 꼬리(tail) 등을 확인하는 데 유용함.



비모수적 방법이란?

- “데이터가 정규분포일 것이다” 같은 가정이 없음
- 대신 데이터 자체를 기반으로 분포의 모양을 추정함
- 필요한 것은 파라미터(평균·분산 등)보다 데이터의 구조와 패턴



1. 연속성 데이터 설정

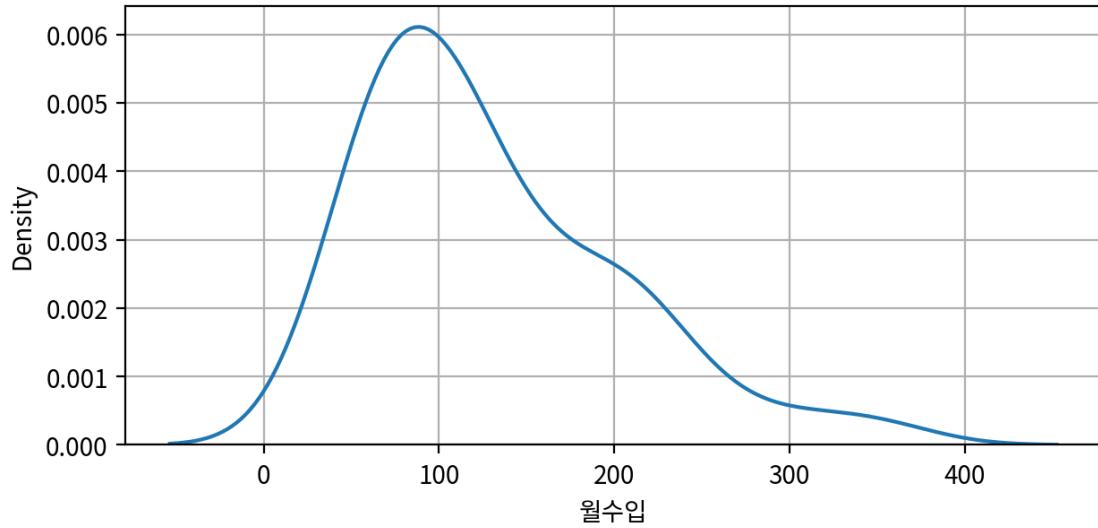
```
# 1) 그래프 초기화  
width_px  = 1280  
height_px = 640  
figsize = (width_px / my_dpi, height_px / my_dpi)  
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)
```

```

# 2) 그래프 그리기
sb.kdeplot(data=origin['월수입'])
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()

```



png

2. 데이터 프레임 자체를 적용하기

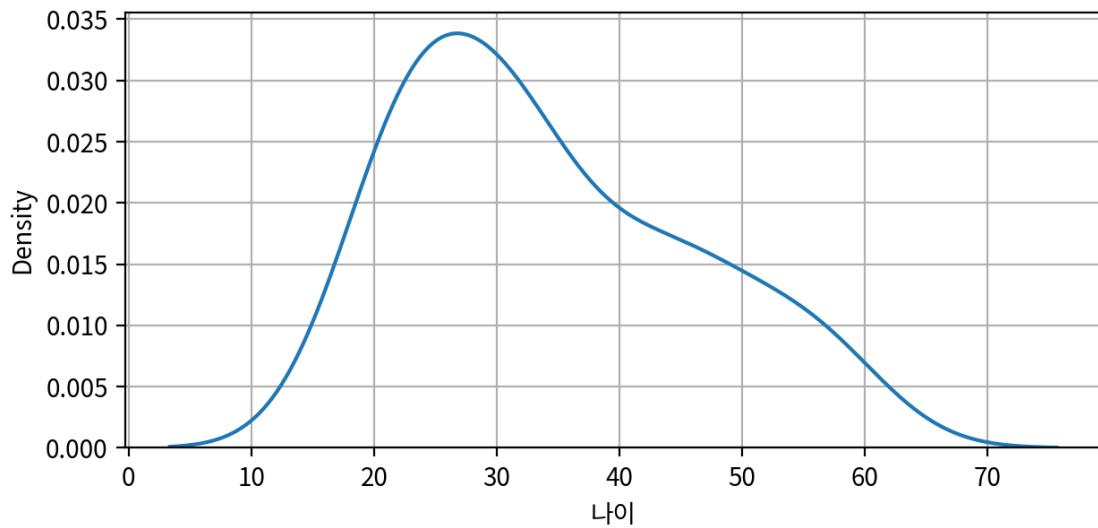
```

# 1) 그래프 초기화
width_px    = 1280
height_px   = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, x='나이')
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()

```



png



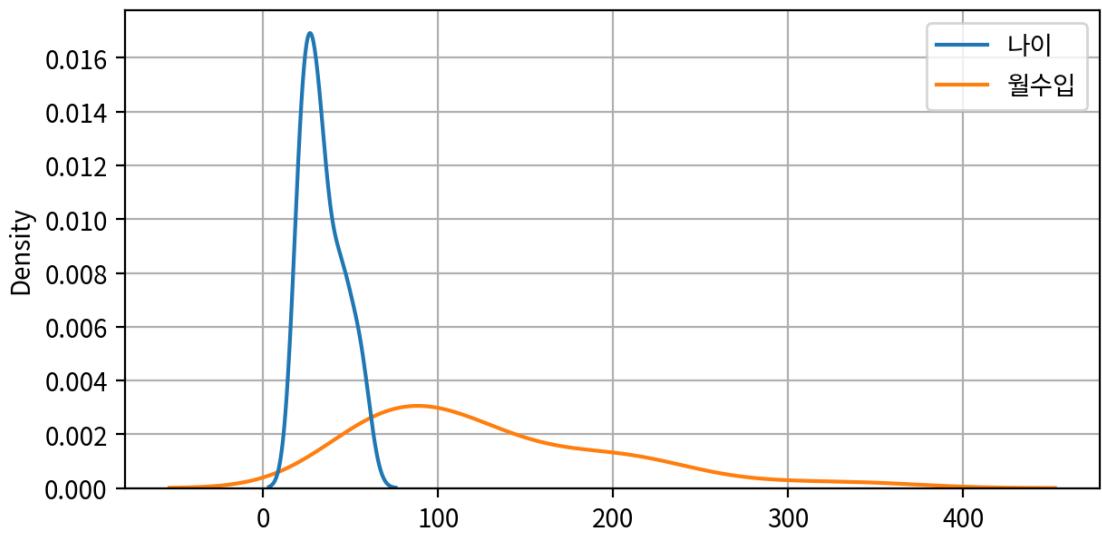
3. 다중 분포

데이터프레임을 data 파라미터에 적용하면서 x나 y파라미터를 지정하지 않으면 모든 연속형 변수에 대한 커널 밀도 곡선이 표현된다.

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin)
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



png



4. 색상 채우기

`fill=True` 파라미터를 설정하면 곡선 내부에 색상이 표시된다.

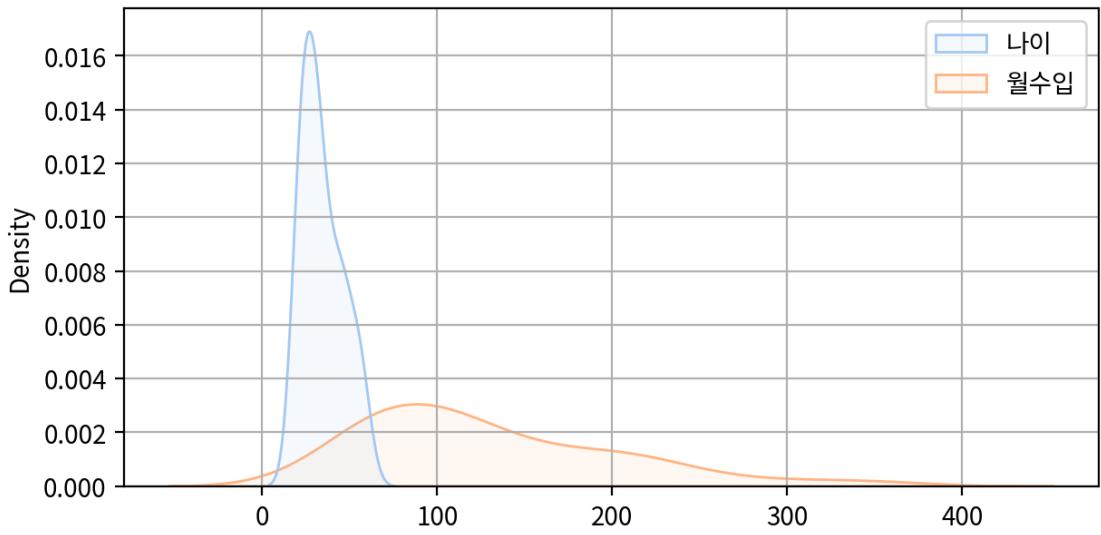
이 때, `alpha` 파라미터를 0~1사이의 값으로 설정하여 면의 투명도를 조절할 수 있다.

0=투명, 1=불투명

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, fill=True, alpha=0.1, palette="pastel")
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



png



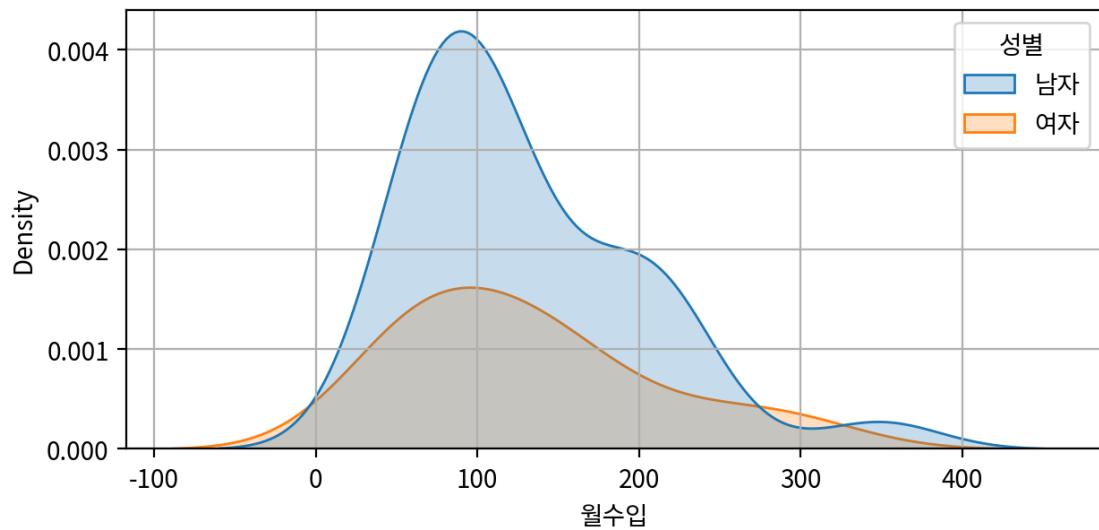
5. 범주에 따른 구분

`hue` 파라미터에 명목형 변수의 이름을 지정하면 범주에 따라 그래프를 분기한다.

```
# 1) 그래프 초기화
width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, x='월수입', hue='성별', fill=True)
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



png

📚 #04. Histogram

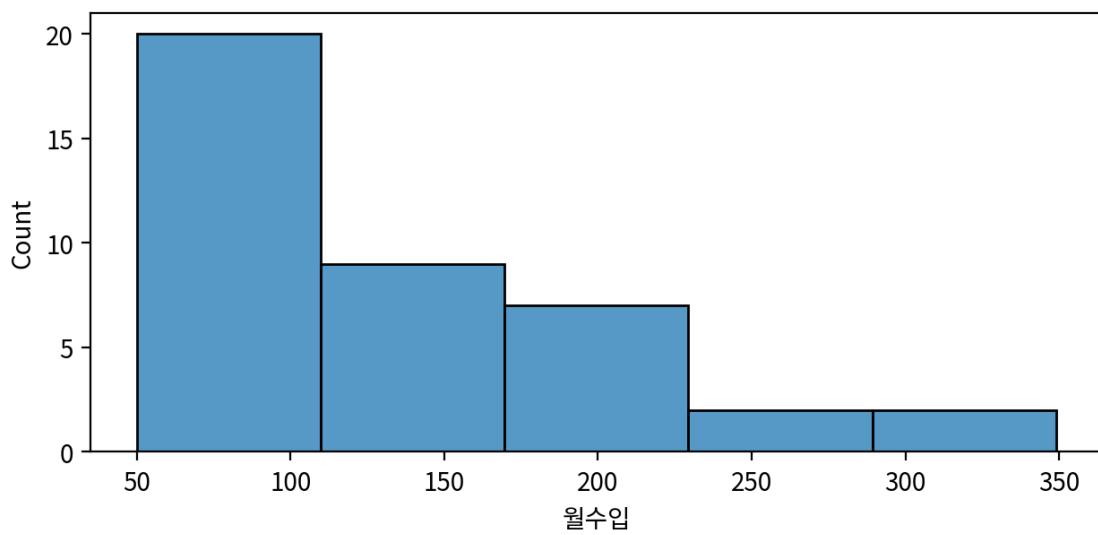
도수 분포표를 시각화 한 그래프

1. 구간 수 지정하기

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin['월수입'], bins=5)
# ax.grid(True) ← 비추

plt.tight_layout()
plt.show()
plt.close()
```



png



2. 구간을 표시하기

```
hist, bins = np.histogram(origin['월수입'], bins=5)
print(hist)
print(bins)
```

```
[20  9  7  2  2]
[ 50.  109.8 169.6 229.4 289.2 349.]
```

```
bins = bins.round().astype("int")
bins
```

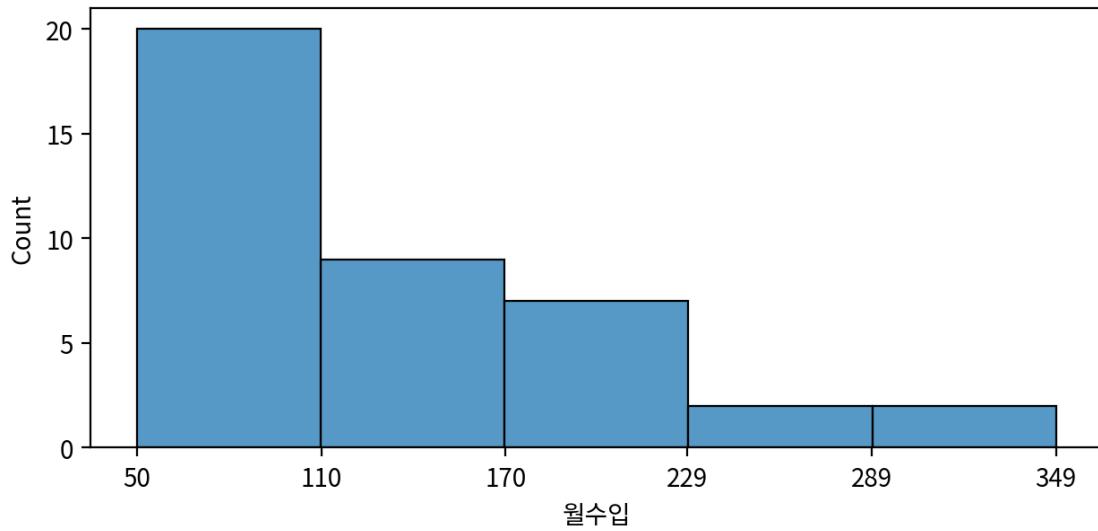
```
array([ 50, 110, 170, 229, 289, 349])
```

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin['월수입'], bins=5, edgecolor="#000000",
            linewidth=0.8)
ax.set_xticks(bins, bins)
# ax.grid(True) ← 비추

plt.tight_layout()
```

```
plt.show()  
plt.close()
```

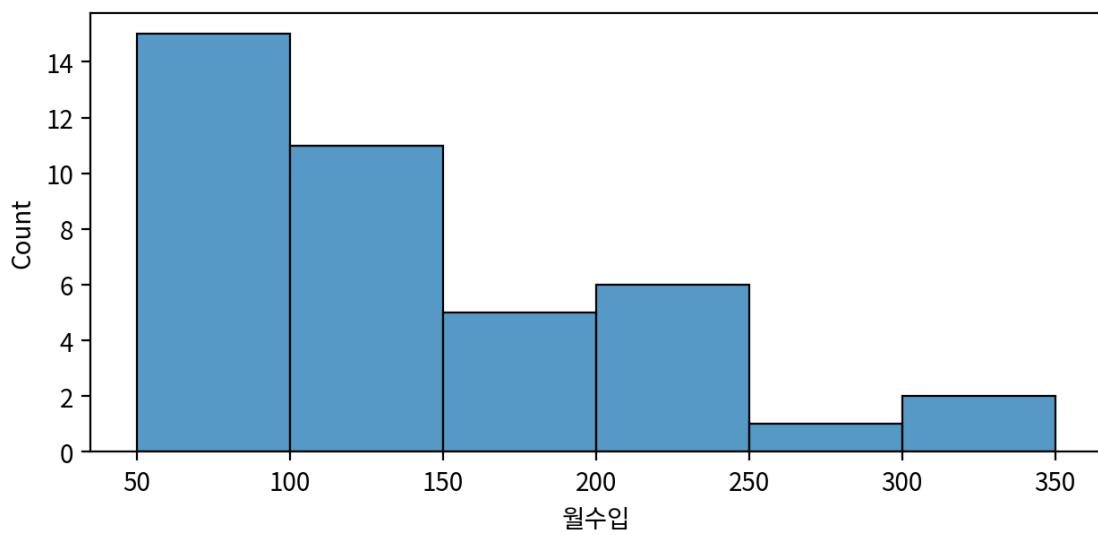


png

3. 구간을 직접 지정하기

bins 파라미터에 구간을 의미하는 리스트를 지정한다.

```
width_px  = 1280  
height_px = 640  
figsize = (width_px / my_dpi, height_px / my_dpi)  
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)  
  
sb.histplot(data=origin['월수입'], bins=[50, 100, 150, 200, 250, 300,  
    350],  
            edgecolor="#000000", linewidth=0.8)  
  
plt.tight_layout()  
plt.show()  
plt.close()
```



png

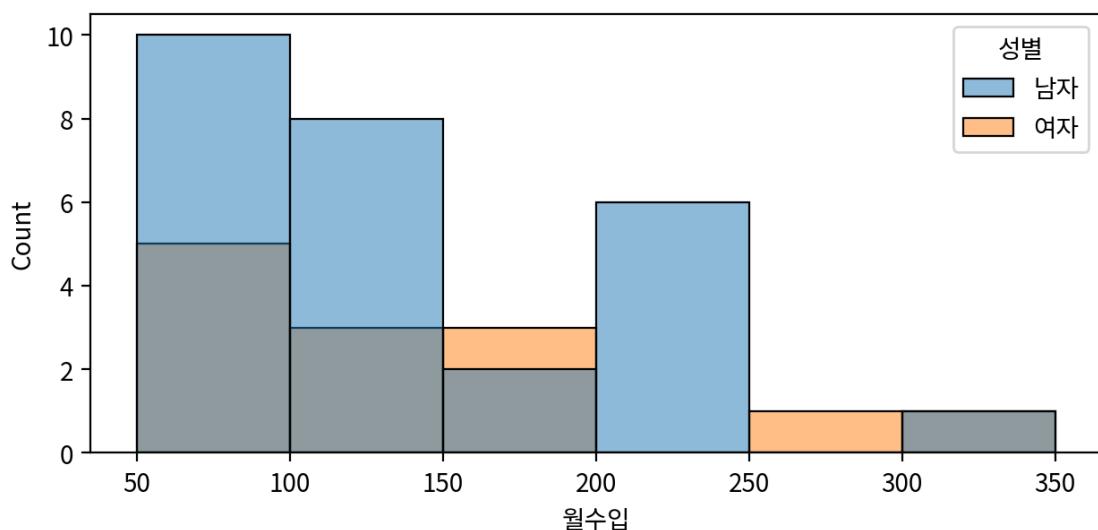


4. 범주에 따른 구분

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin, x='월수입', hue='성별',
            bins=[50, 100, 150, 200, 250, 300, 350],
            edgecolor="#000000", linewidth=0.8)

plt.tight_layout()
plt.show()
plt.close()
```



png

5. 히스토그램의 계급의 수와 간격의 크기

우리에게 주어지는 데이터는 종류도 다양하며 관측치의 개수도 다양하다.

또한 그 데이터의 최솟값과 최댓값 또한 다양하다. 따라서 데이터가 주어졌을 때 계급의 개수와 계급의 간격을 설정하는 것에 어려움이 있을 수 있다.

(1) 계급의 수를 구하는 방법

스터지스의 공식(Sturges' formula)

히스토그램의 계급의 수를 결정하는 공식

$$\text{계급구간수} = 1 + 3.3\log n$$

n = 관측치의 수

관측치의 수에 따른 계급구간 수 표 활용

통계학에서 일반적으로 활용하는 표

관측치의 수	계급구간의 수
50 미만	5 ~ 7
50 ~ 200	7 ~ 9
200 ~ 500	9 ~ 10
500 ~ 1,000	10 ~ 11
1,000 ~ 5,000	11 ~ 13
5,000 ~ 50,000	13 ~ 17
50,000 초과	17 ~ 20

(2) 간격의 크기를 구하는 방법

$$\text{계급구간의간격} = \frac{\text{관측치최대값} - \text{관측치최소값}}{\text{계급구간의수}}$$

여기서 중요한 지점은 계급구간의 첫 시작지점을 어떻게 잡아야 하는지이다.

이 때 필수적으로 알아야 하는 점은 첫 계급구간은 반드시 최소 관측치를 포함해야 한다는 점이다.

(3) 히스토그램의 모습

대칭성

정중앙 부분에 선을 수직으로 긋고 절반으로 접었을 때 일치하는 그래프

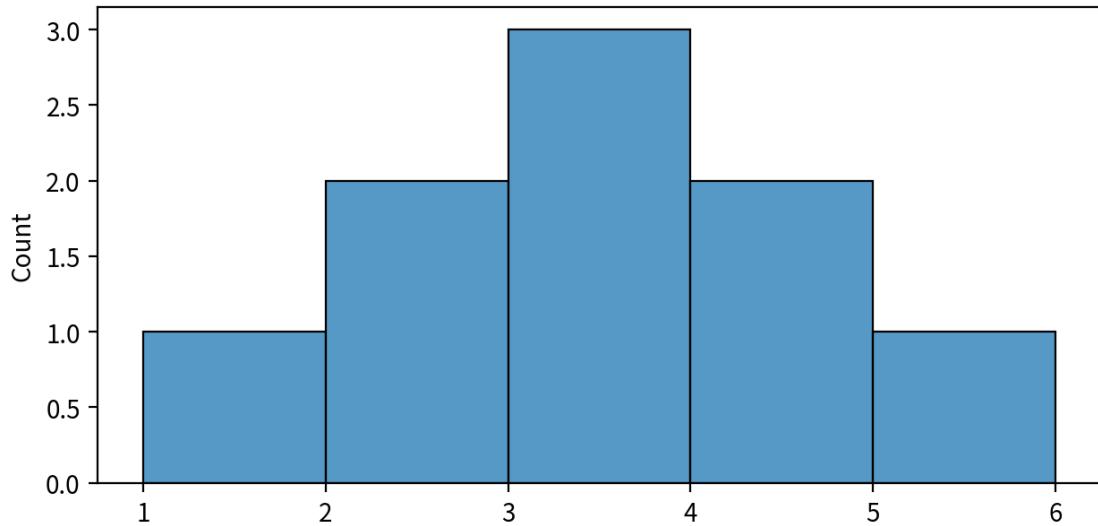
특히 정삼각형에 가까운 형태를 종모양이라고 하는데 이 그래프는 정규분포와 밀접한 관련이 있기 때문에 굉장히 중요한 히스토그램의 모습이라고 할 수 있다.

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 2, 2, 3, 3, 3, 4, 4, 5]

width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

plt.tight_layout()
plt.show()
plt.close()
```



png

비대칭성

그래프의 모양을 보면 점점 작아지는 히스토그램의 모양을 볼 수 있는데, 점점 작아지는 방향을 꼬리라고 칭하며 꼬리가 양의 방향으로 향해 있다면 양의 비대칭이며 꼬리가 음의 방향을 향해 있다면 음의 비대칭이라고 할 수 있다. 밑의 그림은 양과 음의 비대칭 히스토그램의 모습이다.

양의 비대칭은 평균이 중앙값보다 작으며 음의 비대칭은 평균이 중앙값보다 크다.

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 5]

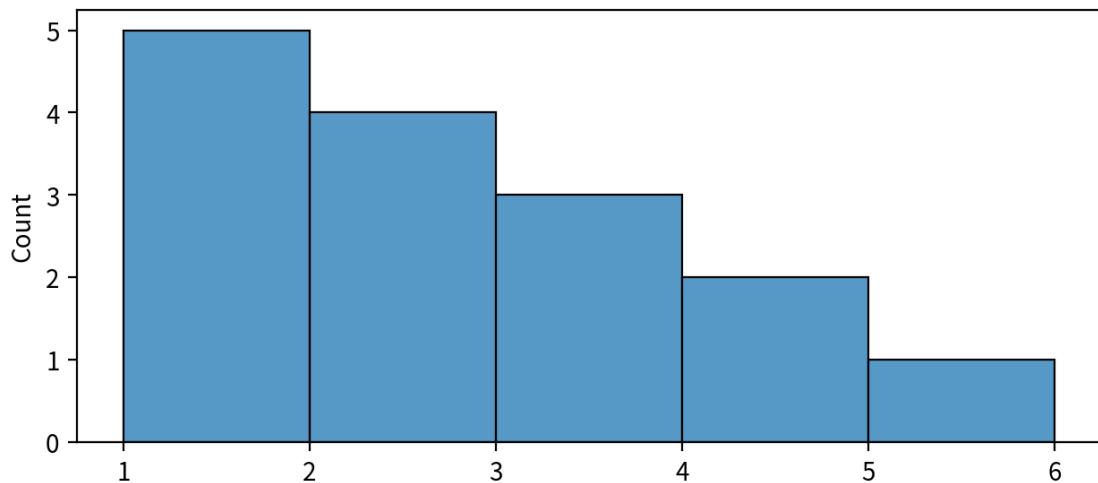
width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

ax.set_title("양의 비대칭", fontsize=15)

plt.tight_layout()
plt.show()
plt.close()
```

양의 비대칭



png

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

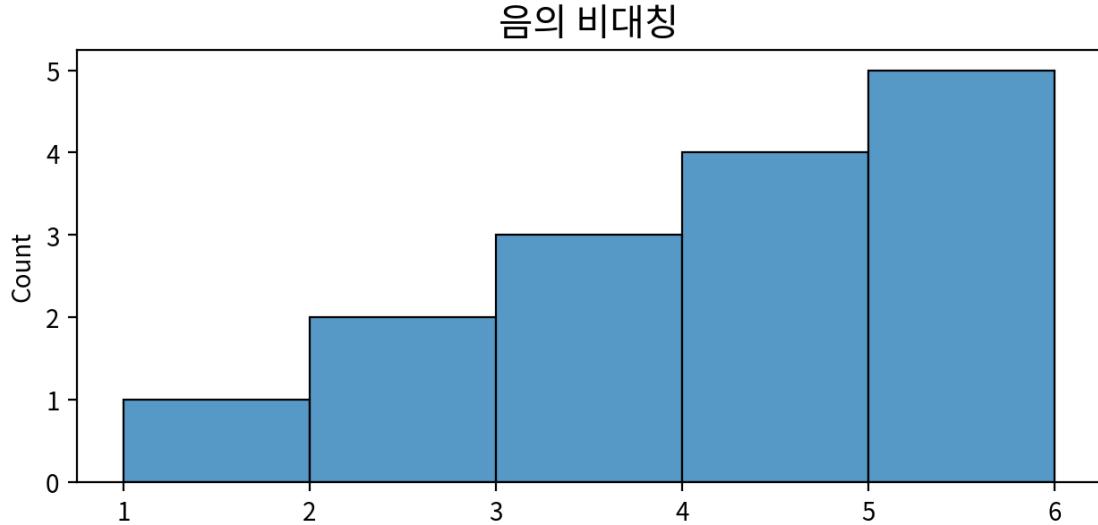
sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)
```

```

ax.set_title("음의 비대칭", fontsize=15)

plt.tight_layout()
plt.show()
plt.close()

```



png

봉우리 계급 구간의 수

히스토그램에서 가장 높은 도수를 나타내고 있는 수치를 최빈값(mode)이라고 부른다.

최빈계급이란 최대의 관측치 수를 가진 계급이다.

만일 최빈계급이 하나일 경우에는 단봉을 가진 히스토그램 이라고 불리며 최빈계급이 두 개일 경우에는 양봉을 가진 히스토그램이라고 불린다.

```

mybins = [1, 2, 3, 4, 5, 6]
data = [1, 1, 1, 2, 2, 3, 4, 4, 5, 5]

width_px = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

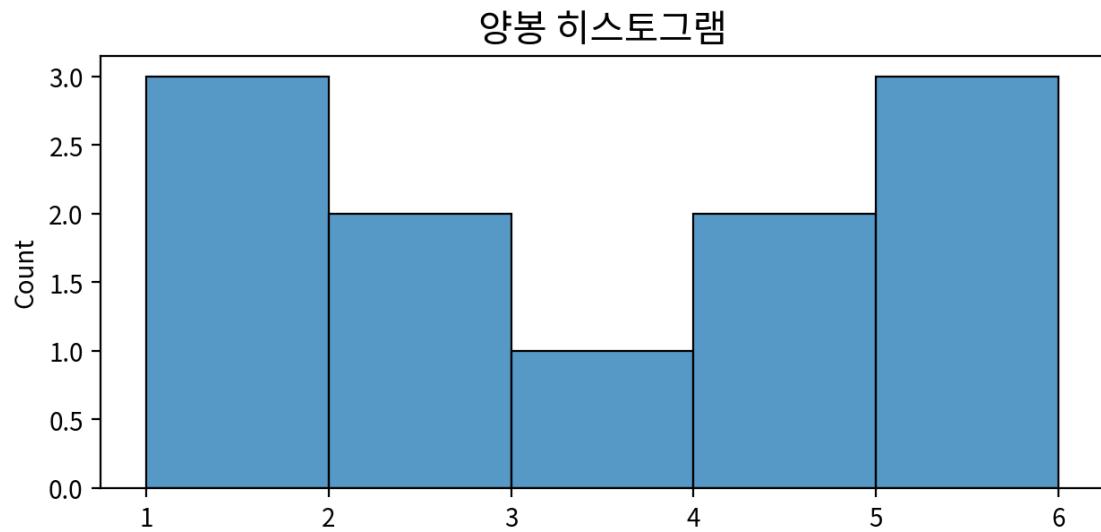
sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

ax.set_title("양봉 히스토그램", fontsize=15)

plt.tight_layout()

```

```
plt.show()  
plt.close()
```



png



[LAB-06] 3. 데이터 시각화 핵심 포인트



#01. 준비작업



1. 라이브러리 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
```



2. 데이터 불러오기

```
origin = load_data('traffic_acc')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab04/traffic_acc.xlsx
[94m[desc] [0m 2005년 1월부터 2018년 12월까지 월별 교통사고의 발생건수, 부상자
수, 사망자수 데이터(인덱스/메타데이터 없음, 출처: 공공데이터포털)
[91m[!] Cannot read metadata [0m
```

	년도	월	발생건수	사망자수	부상자수
0	2005	1	15494	504	25413
1	2005	2	13244	431	21635
2	2005	3	16580	477	25550
3	2005	4	17817	507	28131
4	2005	5	19085	571	29808
...
163	2018	8	18335	357	27749
164	2018	9	18371	348	27751
165	2018	10	19738	373	28836
166	2018	11	19029	298	28000
167	2018	12	18010	323	26463

168 rows × 5 columns

```
df = origin.drop('월', axis=1).groupby('년도').sum()
df
```

	발생건수	사망자수	부상자수
년도			
2005	214171	6376	342233
2006	213745	6327	340229
2007	211662	6166	335906
2008	215822	5870	338962
2009	231990	5838	361875
2010	226878	5505	352458
2011	221711	5229	341391
2012	223656	5392	344565
2013	215354	5092	328711
2014	223552	4762	337497
2015	232035	4621	350400
2016	220917	4292	331720
2017	216335	4185	322829
2018	217148	3781	323037



3. 그래프 초기화

```
my_dpi = 200
fpath = "./NotoSansKR-Regular.ttf"
fm.fontManager.addfont(fpath)
fprop = fm.FontProperties(fname=fpath)
fname = fprop.get_name()
    출
plt.rcParams['font.family'] = fname
plt.rcParams['font.size'] = 6
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
# 이미지 선명도(100~300)
# 한글을 지원하는 폰트 파일의 경로
# 폰트의 글꼴을 시스템에 등록함
# 폰트의 속성을 읽어옴
# 읽어온 속성에서 폰트의 이름만 추출
# 그래프에 한글 폰트 적용
# 기본 폰트 크기
# 그래프에 마이너스 깨짐 방지
```



#02. 시각화 기본 코드

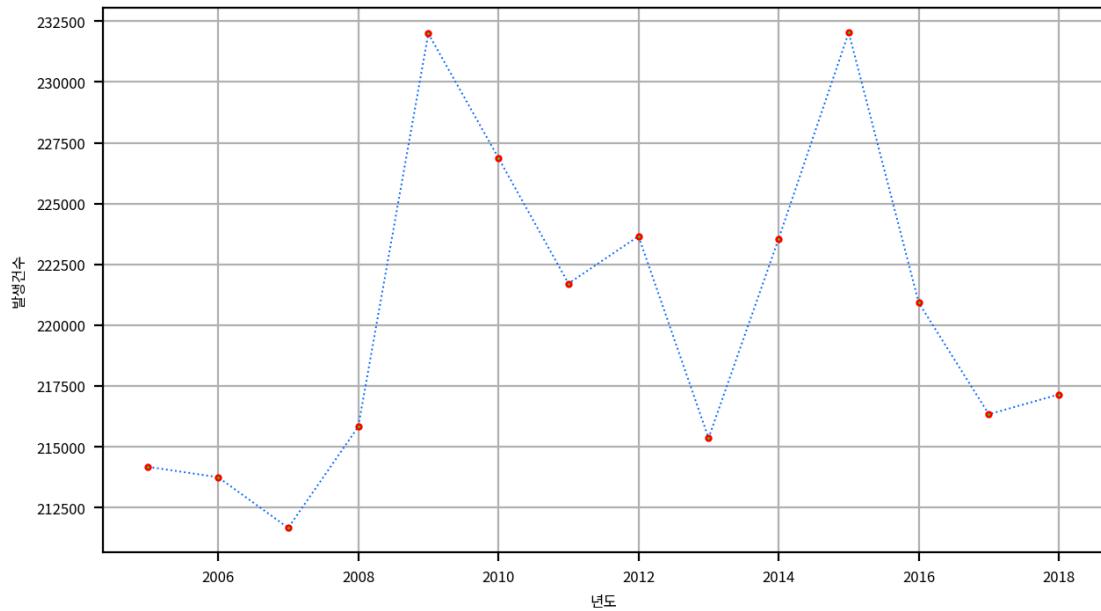
```
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax  = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2-1) BoxPlot 그리기
# sb.boxplot(data=df['발생건수'], orient="v")
# sb.boxplot(data=df['부상자수'], orient="v")
# sb.boxplot(data=df[['발생건수', '부상자수']], orient="v")

# 2-2) LinePlot
sb.lineplot(
    #df['발생건수'],
    #x=df.index, y=df['발생건수'],
    data=df, x=df.index, y='발생건수',
    color="#0066ff", linewidth=0.7, linestyle=':',
    marker="o", markersize=2,
    markerfacecolor="#00ff00",
    markeredgecolor="#ff0000", markeredgewidth=1)

# 3) 그래프 꾸미기
ax.grid(True)                         # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                    # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료
```



png

#03. 다중 그래프

```

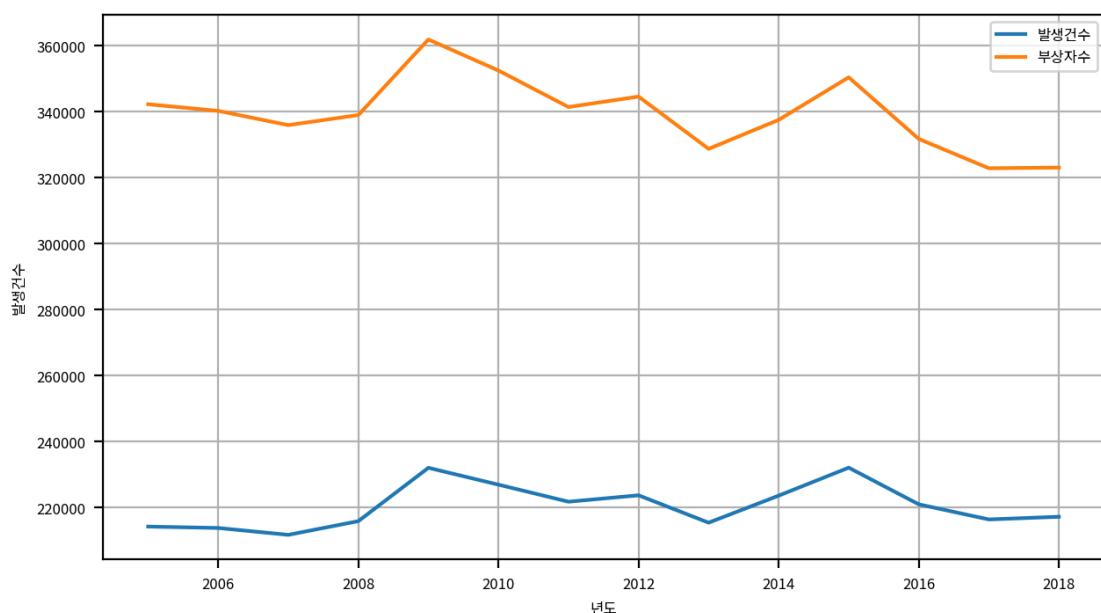
# 1) 그래프 초기화
width_px    = 1280          # 그래프 가로 크기
height_px   = 720           # 그래프 세로 크기
rows         = 1              # 그래프 행 수
cols         = 1              # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 선 그래프
sb.lineplot(data=df, x=df.index, y='발생건수', label='발생건수')
sb.lineplot(data=df, x=df.index, y='부상자수', label='부상자수')

# 3) 그래프 꾸미기
ax.grid(True)                # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()            # 여백 제거
plt.show()                    # 그래프 화면 출력
plt.close()                   # 그래프 작업 종료

```



png

#04. hue 파라미터 이해하기

1. 데이터 전처리 (melt)

melt를 적용할 경우 index 해제

```
df1 = df.reset_index()
df1
```

	년도	발생건수	사망자수	부상자수
0	2005	214171	6376	342233
1	2006	213745	6327	340229
2	2007	211662	6166	335906
3	2008	215822	5870	338962
4	2009	231990	5838	361875
5	2010	226878	5505	352458
6	2011	221711	5229	341391
7	2012	223656	5392	344565
8	2013	215354	5092	328711
9	2014	223552	4762	337497

10	2015	232035	4621	350400
11	2016	220917	4292	331720
12	2017	216335	4185	322829
13	2018	217148	3781	323037

```
df2 = df1.melt(id_vars='년도', value_vars=['발생건수', '사망자수', '부상자수'],
                 var_name='구분')
df2
```

	년도	구분	value
0	2005	발생건수	214171
1	2006	발생건수	213745
2	2007	발생건수	211662
3	2008	발생건수	215822
4	2009	발생건수	231990
5	2010	발생건수	226878
6	2011	발생건수	221711
7	2012	발생건수	223656
8	2013	발생건수	215354
9	2014	발생건수	223552
10	2015	발생건수	232035
11	2016	발생건수	220917
12	2017	발생건수	216335
13	2018	발생건수	217148
14	2005	사망자수	6376
15	2006	사망자수	6327
16	2007	사망자수	6166
17	2008	사망자수	5870
18	2009	사망자수	5838
19	2010	사망자수	5505
20	2011	사망자수	5229
21	2012	사망자수	5392
22	2013	사망자수	5092

23	2014	사망자수	4762
24	2015	사망자수	4621
25	2016	사망자수	4292
26	2017	사망자수	4185
27	2018	사망자수	3781
28	2005	부상자수	342233
29	2006	부상자수	340229
30	2007	부상자수	335906
31	2008	부상자수	338962
32	2009	부상자수	361875
33	2010	부상자수	352458
34	2011	부상자수	341391
35	2012	부상자수	344565
36	2013	부상자수	328711
37	2014	부상자수	337497
38	2015	부상자수	350400
39	2016	부상자수	331720
40	2017	부상자수	322829
41	2018	부상자수	323037



2. hue 파라미터를 사용한 시각화

```

# 1) 그래프 초기화
width_px    = 1280                         # 그래프 가로 크기
height_px   = 720                           # 그래프 세로 크기
rows         = 1                             # 그래프 행 수
cols         = 1                             # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)
fig, ax      = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

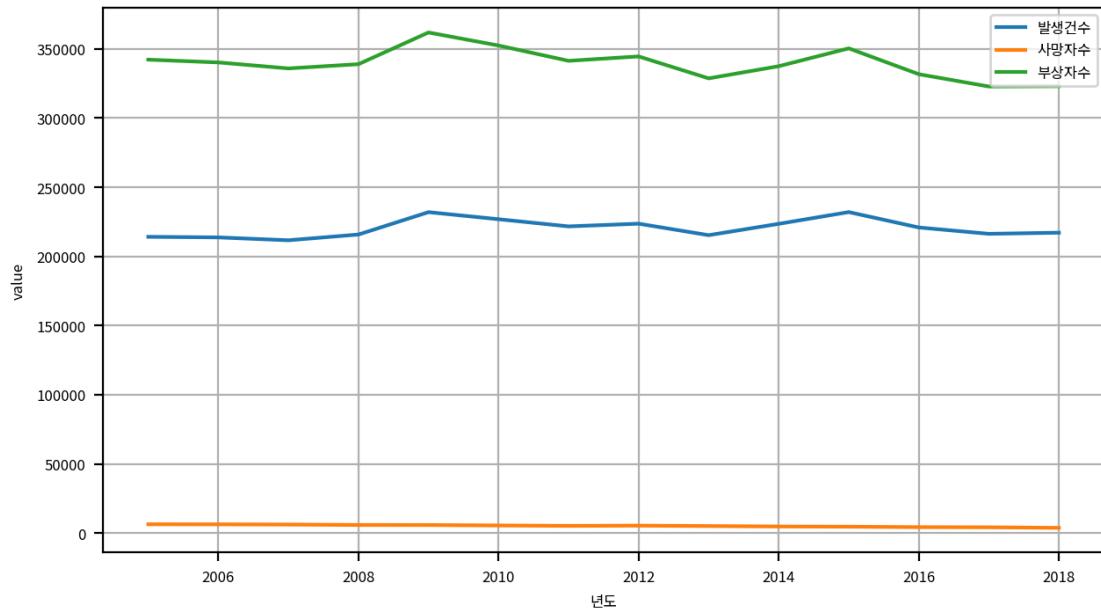
# 2) LinePlot 그리기
sb.lineplot(data=df2, x="년도", y="value", hue="구분")

# 3) 그래프 꾸미기
ax.grid(True)                                # 배경 격자 표시/숨김
ax.legend(loc="upper right")

# 4) 출력

```

```
plt.tight_layout()      # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show()              # 그래프 화면 출력
plt.close()             # 그래프 작업 종료
```



png



[LAB 06] 4. 데이터 분포 시각화 (2)



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
```



2. 데이터 불러오기

```
origin = load_data('employee_data_40')
origin.head()
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/
employee_data_40.xlsx
[94m[desc] [0m 어느 기업의 직원 40명을 대상으로 성별과 결혼상태, 나이, 최종학력,
월수입을 조사한 가상의 데이터(인덱스, 메타데이터 없음)
[91m[!] Cannot read metadata [0m
```

	성별	결혼상태	나이	최종학력	월수입
0	남자	기혼	21	대학교	60
1	남자	기혼	22	대학원	100
2	남자	기혼	33	대학교	200
3	여자	미혼	33	대학교	120
4	남자	미혼	28	대학교	70



3. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
```

```
fname = fprop.get_name()                      # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname            # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6                  # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False     # 그래프에 마이너스 깨짐 방지
```

#02. Histplot + KDE

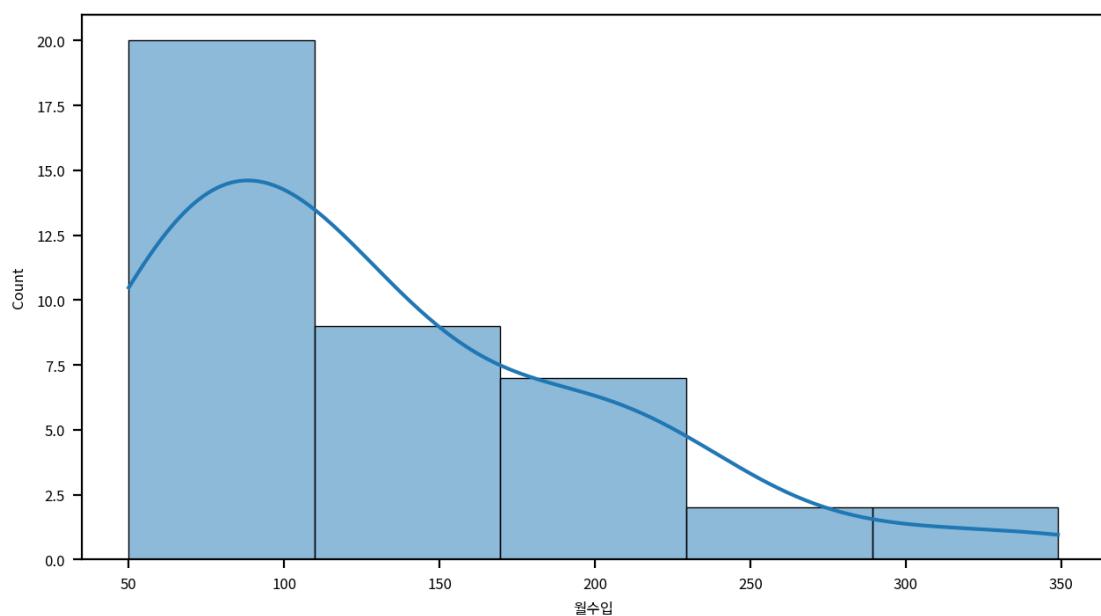
히스토그램에 커널밀도 그래프를 추가한 형태

histplot() 함수에 kde=True 파라미터를 추가한다.

```
# 1) 그래프 초기화
width_px = 1280                                # 그래프 가로 크기
height_px = 720                                 # 그래프 세로 크기
rows = 1                                         # 그래프 행 수
cols = 1                                         # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) Histogram 그리기
sb.histplot(data=origin, x="월수입", bins=5, edgecolor="#000000",
            linewidth=0.5, kde=True)

# 4) 출력
plt.tight_layout()                             # 여백 제거
plt.show()                                     # 그래프 화면 출력
plt.close()                                     # 그래프 작업 종료
```



png

#03. 그 밖의 데이터 분포 시각화 방법



1. 바이올린 플롯

- 데이터의 분포 형태(치우침, 뾰족함, 꼬리 등)를 한눈에 보여주는 그래프
- 박스플롯(boxplot)에 커널 밀도추정(KDE) 그래프를 결합한 형태
- 좌우 대칭의 “바이올린 모양” 폭이 해당 구간의 데이터 밀도를 의미
- 중앙에는 보통 중앙값, 사분위수(IQR) 등을 표시해 박스플롯 정보도 함께 제공
- 여러 그룹·카테고리의 분포 비교에 효과적
- 데이터의 세부적 분포 구조(멀티 모달, 꼬리 부분 등)를 박스플롯보다 더 잘 표현

1) 그래프 초기화

```
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows     = 1                            # 그래프 행 수
cols     = 1                            # 그래프 열 수
figsize  = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

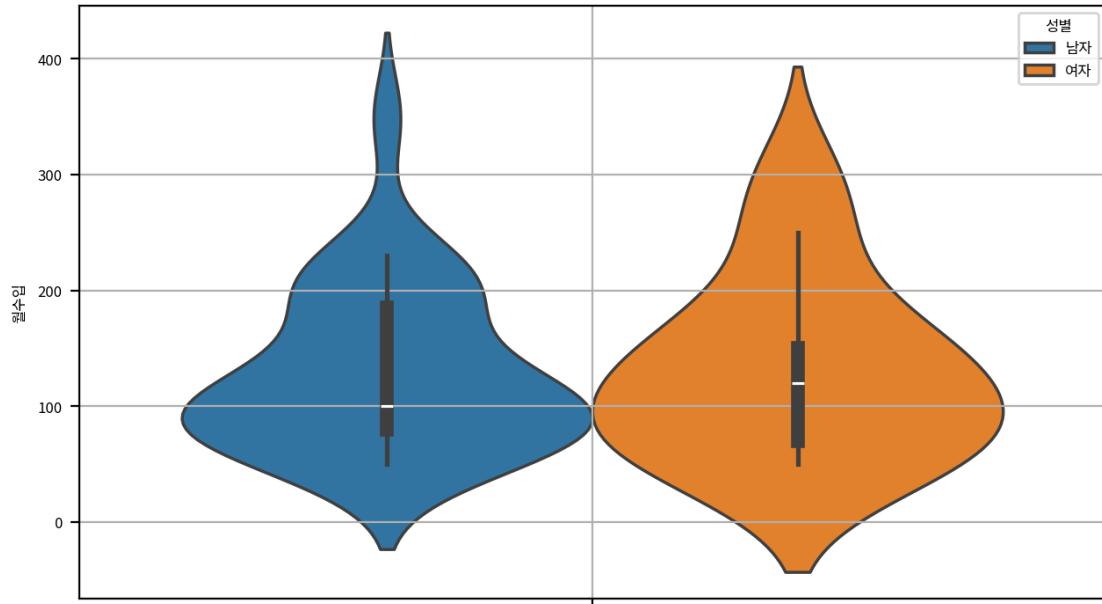
2) Histogram 그리기

```
sb.violinplot(data=origin, y='월수입', hue='성별')
```

3) 그래프 꾸미기

```
ax.grid(True)                           # 배경 격자 표시/숨김
```

```
# 4) 출력
plt.tight_layout()      # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show()                # 그래프 화면 출력
plt.close()                # 그래프 작업 종료
```



png



2. Strip Plot

- 개별 데이터를 점 하나씩 그대로 표시하는 분포 시각화
- x축(또는 y축) 기준으로 값들이 가로·세로로 흩뿌려진 형태
- 작은 표본이나 정확한 관측값 하나하나를 보여줄 때 효과적
- 범주형 그룹 간 원시 데이터 비교가 가능해 boxplot·violinplot보다 세밀한 확인 가능
- 여러 그룹 비교 시, swarmplot과 함께 많이 사용됨 (swarmplot은 겹침을 더 지능적으로 피함)

```
# 1) 그래프 초기화
width_px    = 1280                      # 그래프 가로 크기
height_px   = 720                        # 그래프 세로 크기
rows        = 1                           # 그래프 행 수
cols        = 1                           # 그래프 열 수
figsize     = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

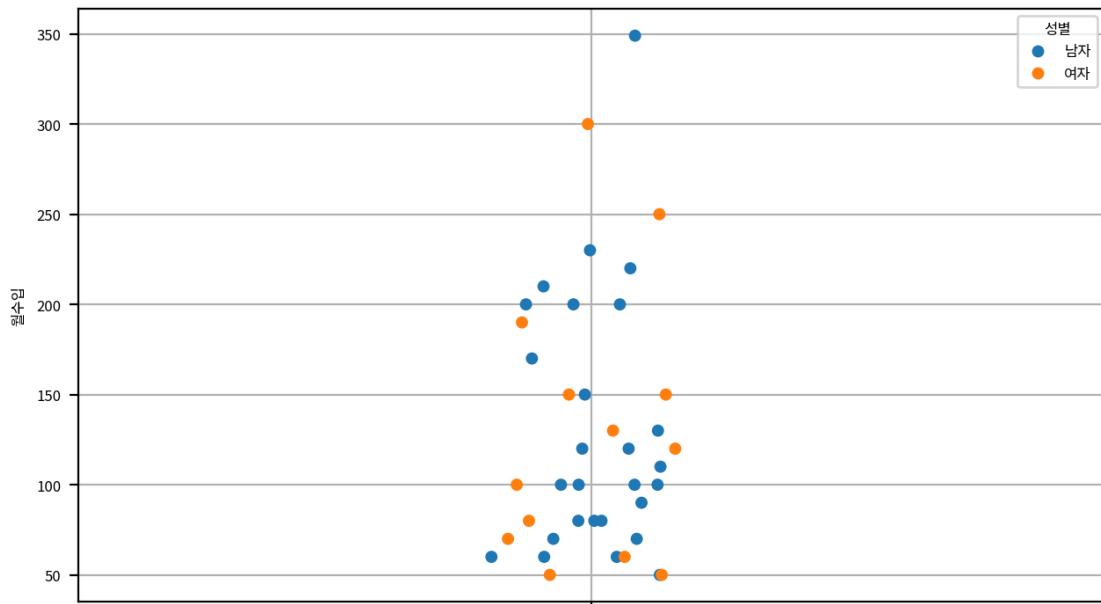
# 2) Histogram 그리기
sb.stripplot(data=origin, y='월수입', hue='성별')
```

```

# 3) 그래프 꾸미기
ax.grid(True) # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout() # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png



3. Swarm Plot

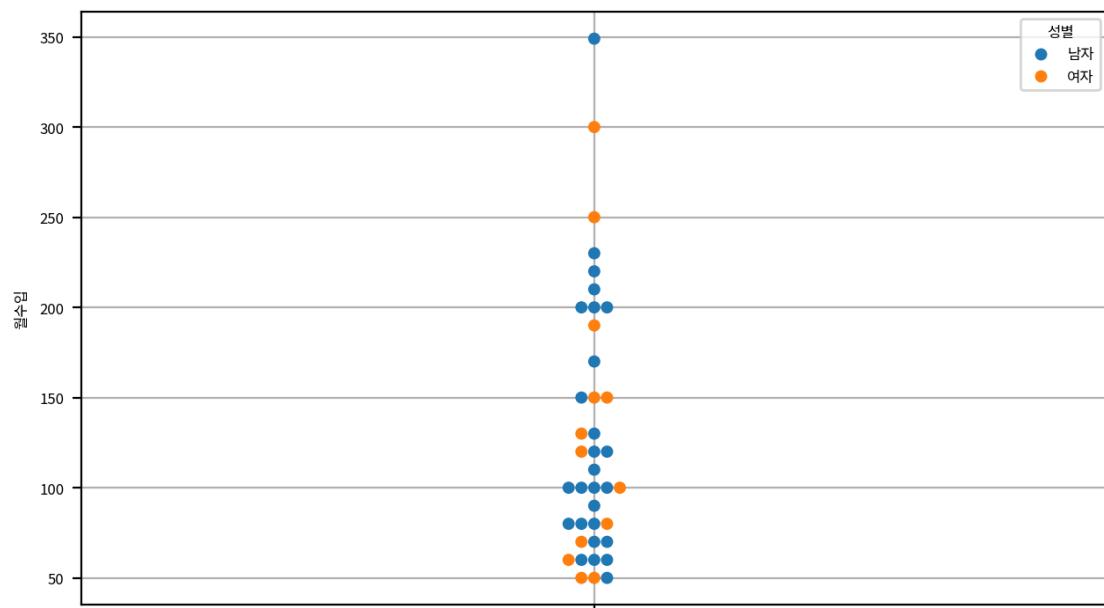
- stripplot처럼 개별 데이터 점을 그대로 표시하는 분포 시각화
- 단, 데이터 점들이 서로 겹치지 않도록 자동으로 배치해주는 알고리즘 사용
 - 이 점을 제외하면 stripplot과 차이가 없다.
- 관측값 하나하나를 보면서도 밀도와 구조를 직관적으로 파악 가능
- 표본 수가 많아도 stripplot보다 가독성이 훨씬 좋음
- 범주형 그룹별 데이터의 실제 분포 형태를 비교할 때 유용

```

# 1) 그래프 초기화
width_px = 1280 # 그래프 가로 크기
height_px = 720 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

```

```
# 2) Histogram 그리기  
sb.swarmplot(data=origin, y='월수입', hue='성별')  
  
# 3) 그래프 꾸미기  
ax.grid(True) # 배경 격자 표시/숨김  
  
# 4) 출력  
plt.tight_layout() # 여백 제거  
plt.savefig("plot.png", dpi=my_dpi * 2)  
plt.show() # 그래프 화면 출력  
plt.close() # 그래프 작업 종료
```



png



[LAB-06] 5. 데이터 분포 시각화 (3)



#01. 준비작업



[1] 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
from pandas import pivot_table
import seaborn as sb
```



[2] 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



[3] 데이터 가져오기

```
origin = load_data('flights')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/flights.xlsx
[94m[desc] [0m 어느 항공사의 년/월별 국제선 탑승객 수(출처: seaborn 내장 데이터)
```

field	description
-----	-----
year	항공 승객 수가 집계된 연도
month	항공 승객 수가 집계된 월
passengers	해당 연도/월의 국제선 항공 승객 수

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
...
139	1960	August	606
140	1960	September	508
141	1960	October	461
142	1960	November	390
143	1960	December	432

144 rows × 3 columns



[4] 데이터 전처리

```
df = origin.copy()
df['month'] = df['month'].map({
    "January": 1, "February": 2, "March": 3, "April": 4,
    "May": 5, "June": 6, "July": 7, "August": 8,
    "September": 9, "October": 10, "November": 11, "December": 12
})

df.head()
```

	year	month	passengers
0	1949	1	112
1	1949	2	118
2	1949	3	132
3	1949	4	129
4	1949	5	121

```
df2 = pivot_table(df, index="year", columns="month",
                  values="passengers")
df2
```

month	1	2	3	4	5	6	7	8	9	10	11	12
year												
1949	112.0	118.0	132.0	129.0	121.0	135.0	148.0	148.0	136.0	119.0	104.0	118.0
1950	115.0	126.0	141.0	135.0	125.0	149.0	170.0	170.0	158.0	133.0	114.0	140.0
1951	145.0	150.0	178.0	163.0	172.0	178.0	199.0	199.0	184.0	162.0	146.0	166.0
1952	171.0	180.0	193.0	181.0	183.0	218.0	230.0	242.0	209.0	191.0	172.0	194.0
1953	196.0	196.0	236.0	235.0	229.0	243.0	264.0	272.0	237.0	211.0	180.0	201.0
1954	204.0	188.0	235.0	227.0	234.0	264.0	302.0	293.0	259.0	229.0	203.0	229.0
1955	242.0	233.0	267.0	269.0	270.0	315.0	364.0	347.0	312.0	274.0	237.0	278.0
1956	284.0	277.0	317.0	313.0	318.0	374.0	413.0	405.0	355.0	306.0	271.0	306.0
1957	315.0	301.0	356.0	348.0	355.0	422.0	465.0	467.0	404.0	347.0	305.0	336.0
1958	340.0	318.0	362.0	348.0	363.0	435.0	491.0	505.0	404.0	359.0	310.0	337.0
1959	360.0	342.0	406.0	396.0	420.0	472.0	548.0	559.0	463.0	407.0	362.0	405.0
1960	417.0	391.0	419.0	461.0	472.0	535.0	622.0	606.0	508.0	461.0	390.0	432.0

#02. Heatmap 시각화

데이터의 패턴, 특히 밀도와 분포를 빠르게 파악할 수 있게 해주는 시각적 도구.

행과 열을 가진 행렬 형태의 데이터를 색상으로 나타내어 각 셀의 값에 따라 색상이 변한다.



[1] 기본 사용 방법

성적표 데이터에 대한 점수 분포 시각화

```
# 1) 그래프 초기화
width_px    = 1000                         # 그래프 가로 크기
height_px   = 1000                         # 그래프 세로 크기
rows        = 1                             # 그래프 행 수
cols        = 1                             # 그래프 열 수
figsize     = (width_px / my_dpi, height_px / my_dpi)
fig, ax     = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

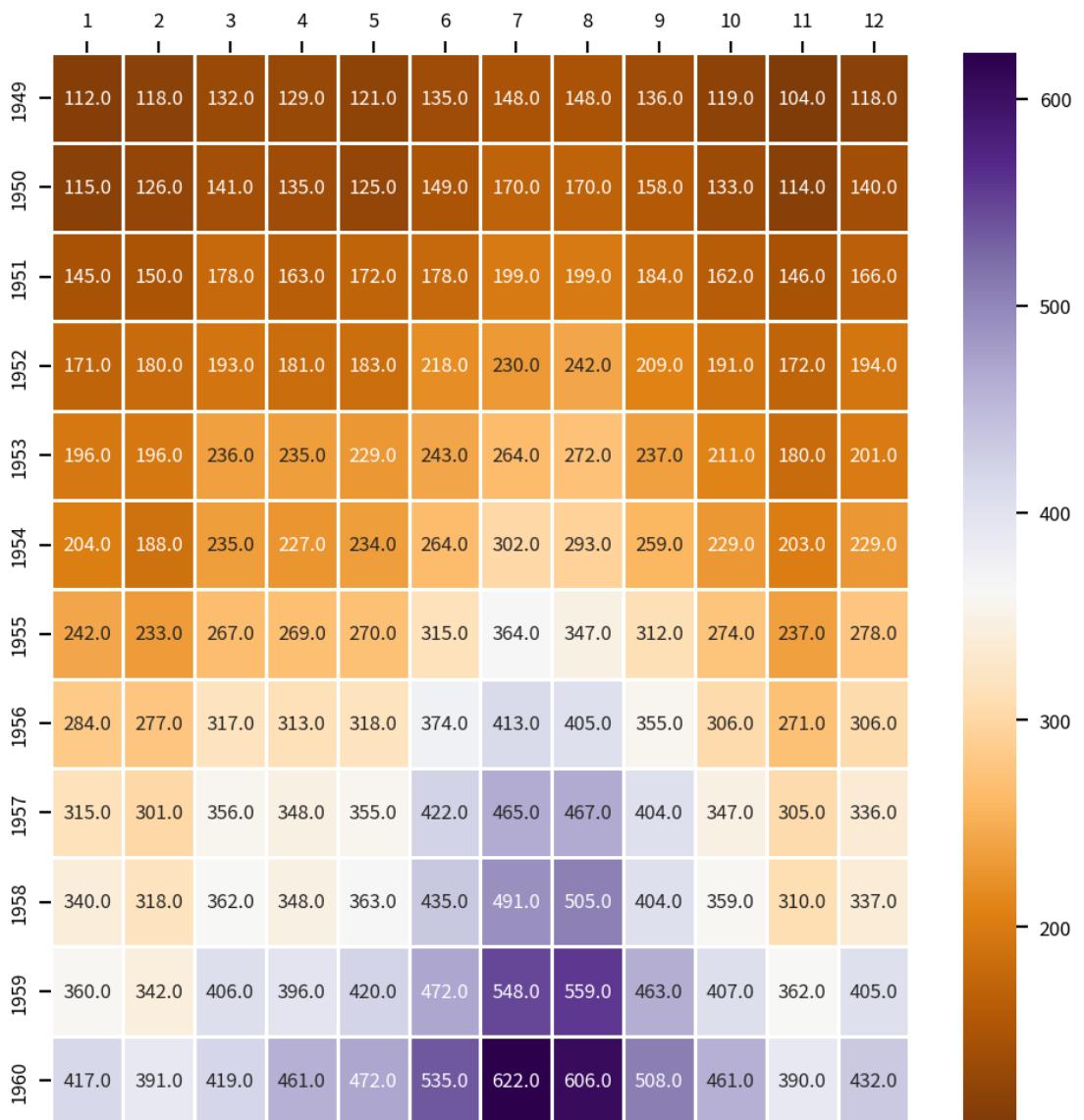
2) heatmap 그리기

```
# → annot=True : 수치값을 함께 표시함
# → fmt: annot=True가 설정된 경우 표시되는 수치값의 형식 지정
# → linewidth: 각 셀 사이의 선 굵기
```

```
# → cmap: 칼라맵 ('Greys', 'Purples', 'Blues', 'Greens', 'Oranges',
#                   'Reds',
#                   'YlOrBr', 'YlOrRd', 'OrRd', 'PuRd', 'RdPu', 'BuPu',
#                   'GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn', 'YlGn',
#                   'PiYG', 'PRGn', 'BrBG', 'PuOr', 'RdGy', 'RdBu',
#                   'RdYlBu',
#                   'RdYlGn', 'Spectral', 'coolwarm', 'bwr', 'seismic',
#                   'berlin', 'managua', 'viamo')
sb.heatmap(data=df2, annot=True, fmt="0.1f", linewidth=0.5,
            cmap="PuOr")

# 3) 그래프 꾸미기
ax.set_xlabel("")
ax.set_ylabel("")
ax.xaxis.tick_top()           # x축의 변수 이름을 상단으로 이동

# 4) 출력
plt.tight_layout()          # 여백 제거
plt.savefig("plot.png", dpi=my_dpi * 2)
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료
```



png

- 여름(6-8월)에 진한 색 → 승객 수가 가장 많음
- 연도가 증가할수록 전체 색 농도가 짙어짐 → 항공 운송 수요 증가 추세
- 연도별 패턴은 거의 동일하지만 크기만 증가 → 계절성 + 추세 구조가 혼합된 전형적 시계열 패턴



[LAB 06] 6. 집단별 요약 시각화



#01. 준비작업



1. 패키지 가져오기

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```



2. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



3. 데이터 가져오기

```
origin = load_data('penguins')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/penguins.xlsx
[94m[desc] [0m 남극 팔мер 군도의 펭귄 3종에 대해 신체 치수와 서식지 정보(출처:
seaborn 내장 데이터)
```

field	description
species	펭귄 종
island	서식지

bill_length_mm	부리 길이
bill_depth_mm	부리 두께
flipper_length_mm	날개 길이
body_mass_g	몸무게
sex	성별

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181	3750	MALE
1	Adelie	Torgersen	39.5	17.4	186	3800	FEMALE
2	Adelie	Torgersen	40.3	18.0	195	3250	FEMALE
3	Adelie	Torgersen	36.7	19.3	193	3450	FEMALE
4	Adelie	Torgersen	39.3	20.6	190	3650	MALE
...
329	Gentoo	Biscoe	47.2	13.7	214	4925	FEMALE
330	Gentoo	Biscoe	46.8	14.3	215	4850	FEMALE
331	Gentoo	Biscoe	50.4	15.7	222	5750	MALE
332	Gentoo	Biscoe	45.2	14.8	212	5200	FEMALE
333	Gentoo	Biscoe	49.9	16.1	213	5400	MALE

334 rows × 7 columns



#02. barplot

- 집단별 평균/합계/비율 등 요약 통계 시각화의 기본형
- estimator 파라미터로 평균(mean) 외에도 median, sum 등 자유롭게 설정 가능
- 범주형 분석에서 가장 표준적으로 쓰이는 그래프

```
# 1) 그래프 초기화
width_px = 1280 # 그래프 가로 크기
height_px = 720 # 그래프 세로 크기
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) barplot 그리기
sb.barplot(
    data=origin, # 사용할 데이터프레임
```

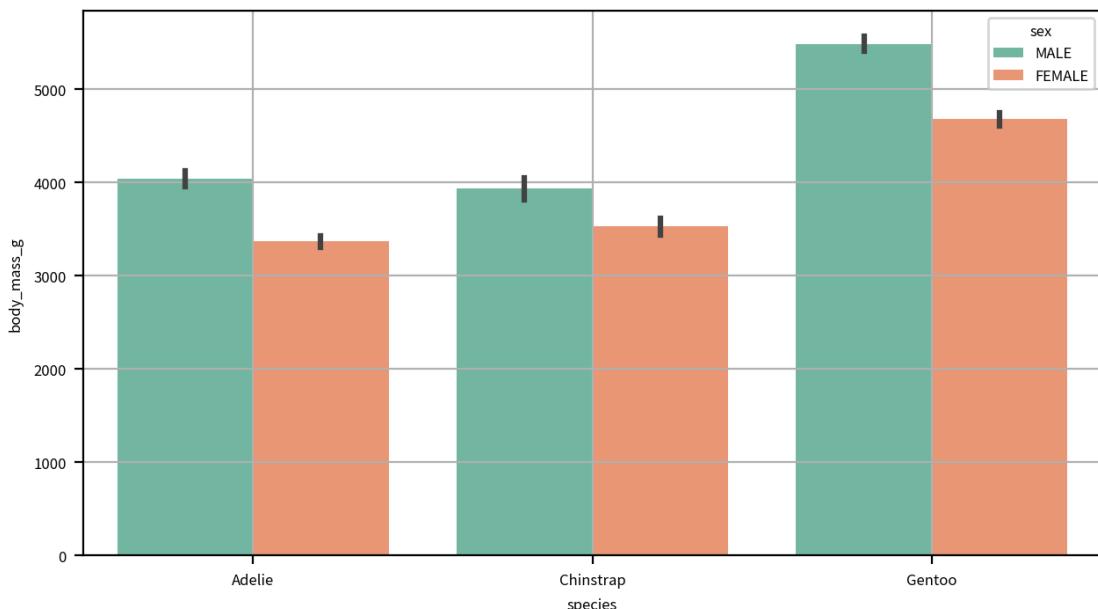
```

x="species",
y="body_mass_g",
hue="sex",
estimator=np.mean,
errorbar=("ci", 95),
준편차, None=없음
palette="Set2"
"bright", ...
)

# 3) 그래프 꾸미기
ax.grid(True)          # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()    # 여백 제거
plt.show()             # 그래프 화면 출력
plt.close()            # 그래프 작업 종료

```



png

#03. countplot (빈도 그래프)

- 범주형 빈도(Count)를 바로 보여주는 가장 단순하고 직관적인 요약 그래프
- 기술통계 보고서·EDA에서 거의 항상 등장

```

# 1) 그래프 초기화
width_px  = 1280           # 그래프 가로 크기
height_px = 720            # 그래프 세로 크기

```

```

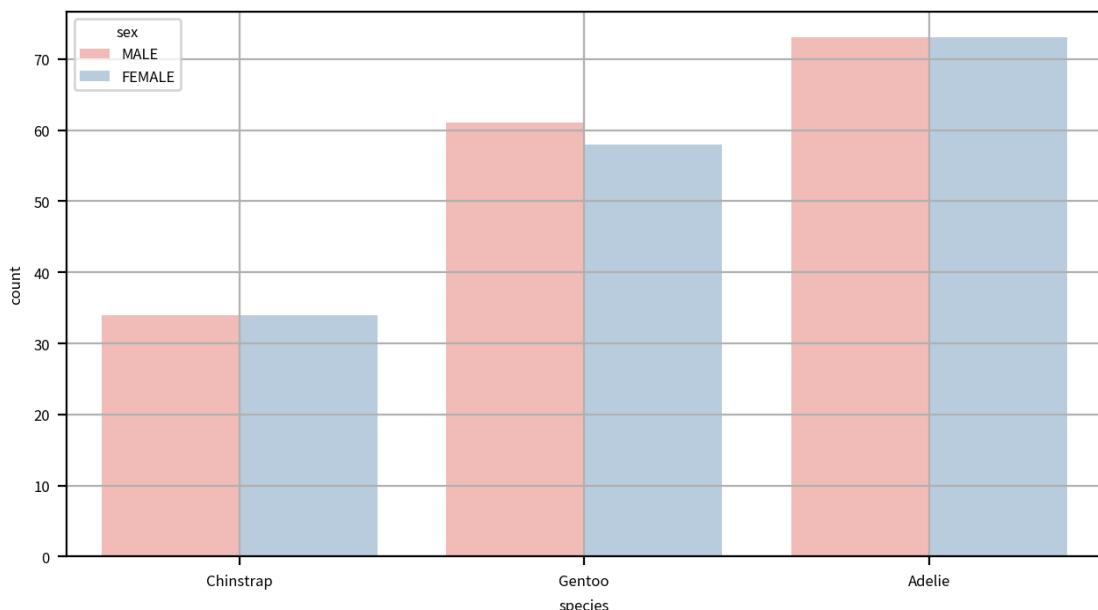
rows = 1 # 그래프 행 수
cols = 1 # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) barplot 그리기
sb.countplot(
    data=origin, # 사용할 데이터프레임
    x="species", # 집계할 범주
    hue="sex", # 그룹 구분: None 가능
    order=['Chinstrap', 'Gentoo', 'Adelie'], # x축 범주 순서
    palette="Pastel1" # 색상 팔레트
)

# 3) 그래프 꾸미기
ax.grid(True) # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout() # 여백 제거
plt.show() # 그래프 화면 출력
plt.close() # 그래프 작업 종료

```



png



#04. pointplot

- 점+오차막대 형태의 집단별 평균 + 신뢰구간 요약

- 여러 그룹(hue) 비교 시 가장 깔끔하고 해석성이 높음
- 회귀분석 전 EDA에서 추세 파악용으로 인기

```

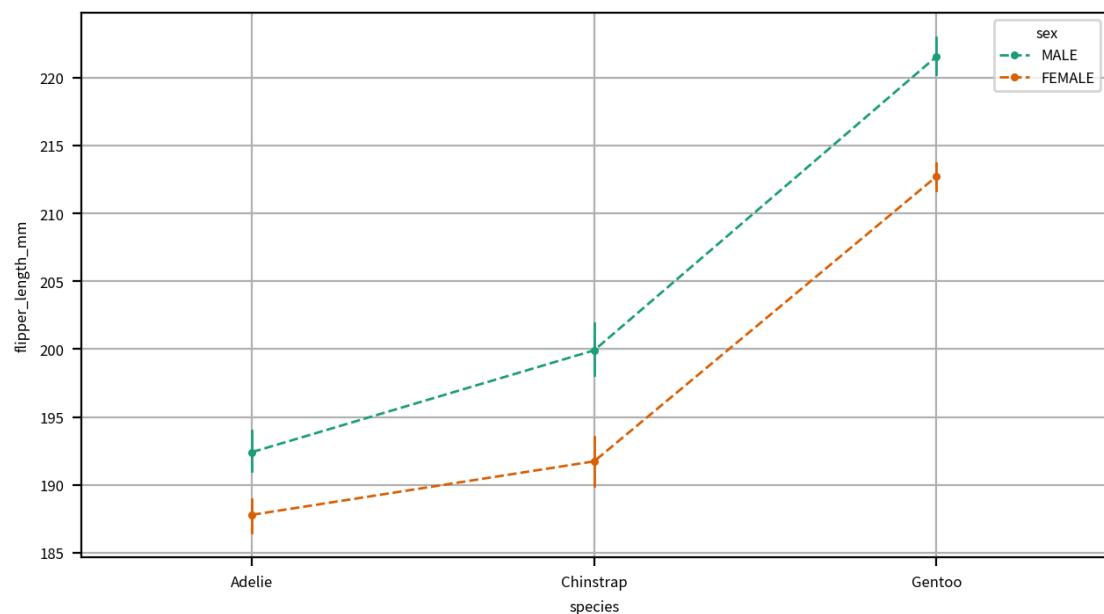
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) barplot 그리기
sb.pointplot(
    data=origin,                      # 사용할 데이터프레임
    x="species",
    y="flipper_length_mm",
    hue="sex",
    estimator=np.mean,                # 요약 방식
    errorbar=("ci", 95),              # 신뢰구간(기본 95%)
    linestyles="--",                  # "-", "--", ":" , "-."
    linewidth=1,
    markers="o",                      # 마커 종류: "o", "s", "D", "^", "v",
    "*",
    palette="Dark2"
)

# 3) 그래프 꾸미기
ax.grid(True)                         # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                    # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료

```



.png



[LAB-06] 7. 데이터간의 관계 시각화 (1) - Scatter Plot (산점도 그래프)



데이터 관계 시각화 그래프 4종

그래프	주요 목적	회귀선	그룹 비교	사용 상황
ScatterPlot	기본 관계 파악	X	X	기초 탐색
RegPlot	선형 경향 + 신뢰구간	O	X	관계 해석 강화
LmPlot	그룹별 선형관계 비교	O	O	비교 분석(성별/지역 등)
PairPlot	전체 변수 관계 한눈에 보기	변수쌍마다 단순 산점도	제한적(hue 지원)	EDA 초기 전체 스캔



Scatter Plot의 이해

- 두 연속형 변수의 관계를 시각화 하는 가장 기본적인 그래프
- 두 연속형 변수간의 영향력(분포와 상관 경향)을 점(point)으로 표시
- 패턴(직선적·곡선적), 군집 형태, 이상치를 쉽게 파악
- 통계적 모델은 포함하지 않으며 관계의 형태를 “그려만 준다”

언제 쓰나? → 변수 간 기본적인 관계가 있는지 빠르게 확인할 때



1. Scatter Plot 유형



2. Scatter Plot의 해석

마커들이 오밀조밀 뭉쳐 있으면 두 변수는 서로 관련성 정도가 높고 흩어져 있으면 관련성이 낮다.

이러한 관계를 상관관계라고 한다.

| 추론통계의 상관분석에서 좀 더 자세히 다룹니다.

예시

아래의 그래프에서 SAT에 응시한 학생 비율이 높을수록 수학 평균 점수는 낮아지는 경향이 있다고 볼 수 있다.



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```



2. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



3. 데이터 가져오기

```
origin = load_data("icecream")
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/icecream.xlsx
[94m[desc] [0m 기온과 아이스크림 판매량을 기록한 가상의 데이터 (메타데이터, 인덱스 없음)
[91m[!] Cannot read metadata [0m
```

	기온	판매량
0	23	431
1	36	593
2	30	512
3	25	474

4	26	476
5	31	523
6	29	491
7	32	526
8	33	550
9	24	456
10	34	566
11	35	581
12	27	480
13	28	487



#02. Scatter Plot 시작화

| hue 파라미터로 데이터 범주 구별이 가능함

```

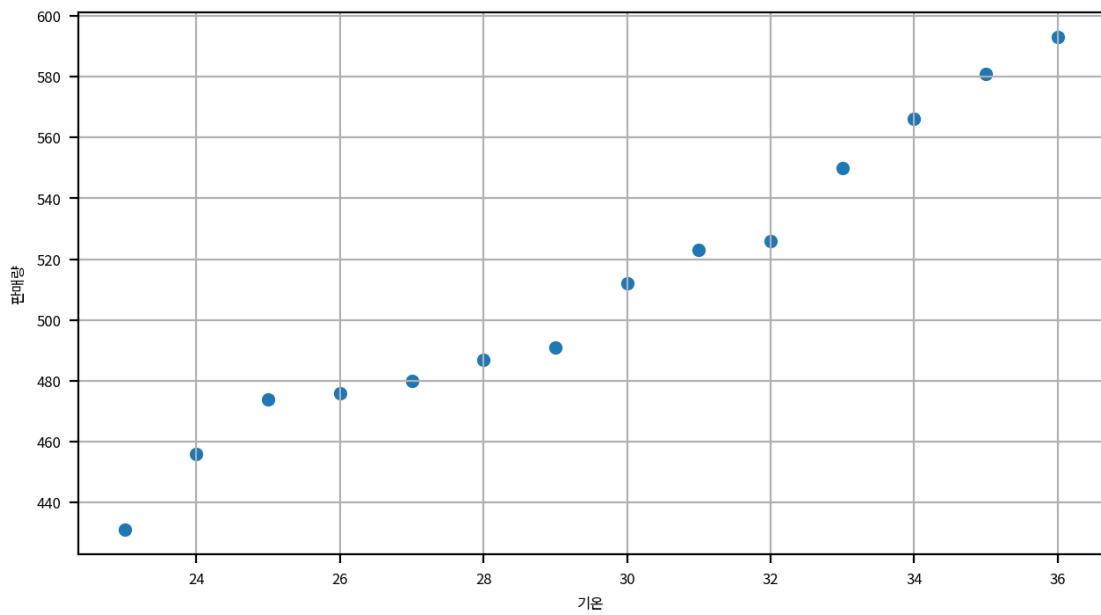
# 1) 그래프 초기화
width_px    = 1280                      # 그래프 가로 크기
height_px   = 720                        # 그래프 세로 크기
rows         = 1                          # 그래프 행 수
cols         = 1                          # 그래프 열 수
figsize      = (width_px / my_dpi, height_px / my_dpi)
fig, ax      = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) BoxPlot 그리기
sb.scatterplot(data=origin, x='기온', y='판매량')

# 3) 그래프 꾸미기
ax.grid(True)                           # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                      # 여백 제거
plt.show()                             # 그래프 화면 출력
plt.close()                            # 그래프 작업 종료

```



png

알 수 있는 사실

기온에 따른 아이스크림 판매량을 산점도 그래프로 확인한 결과 기온이 상승할 수록 아이스크림 판매량도 증가하는 추세를 보이는 것으로 확인되었다.



#03. 추세선(회귀선) 그리기

주어진 데이터의 일반적인 경향이나 패턴을 나타내는 선

추세선은 주로 선형 회귀 분석을 통해 계산되며 이를 통해 데이터 간의 관계를 파악하고 예측 모델을 개발하는데 도움이 된다.

일반적으로 추세선을 그리기 위해서는 `scipy`나 `sklearn` 패키지를 통해 선형회귀 모델을 구현하여 회귀식을 도출해야 하지만 간단한 선형회귀의 경우 `numpy`를 통해서도 분석 모델을 도출할 수 있다.



1. 기울기(계수, 가중치)와 절편(상수항, 편향) 구하기

계수, 상수항 = `np.polyfit(x, y, 차수)`

통계학에서는 계수(기울기)를 가중치, 상수항(절편)를 편향이라고 하지만 `numpy`는 수학적 기능을 구현하고 있는 패키지이므로 `numpy`에서는 상수항과 계수라고 표현한다.

```
z = np.polyfit(origin['기온'], origin['판매량'], 1)
print("상수항:", z[0])
print("계수:", z[1])
```

상수항: 11.39780219780219
계수: 174.19340659340702



2. 회귀 분석 모형

상수항과 계수를 활용하여 $y = ax + b$ 에 해당하는 방정식을 확인한다.

```
expr = "y = %0.1f * x + %0.1f" % (z[0], z[1])
expr
```

'y = 11.4 * x + 174.2'



3. 분석 모형 객체 생성

$y = ax + b$ 에 해당하는 방정식 객체를 생성한다.

```
f = np.poly1d(z)
f
```

poly1d([11.3978022 , 174.19340659])



4. 분석모형을 활용한 판매량 예

방정식 $f(x)$ 에 x 값을 전달하면 그에 대한 결과를 확인할 수 있다.

```
x = 40
print("기온이 %d일 경우 아이스크림 판매량은 %f로 예상됩니다." % (x, f(x)))
```

기온이 40일 경우 아이스크림 판매량은 630.105495로 예상됩니다.



5. 전체 기온에 대한 예측 판매량 확

```
x = origin['기온']
y = f(x)
y
```

```
array([436.34285714, 584.51428571, 516.12747253, 459.13846154,
       470.53626374, 527.52527473, 504.72967033, 538.92307692,
       550.32087912, 447.74065934, 561.71868132, 573.11648352,
       481.93406593, 493.33186813])
```

6. 추세선을 포함하는 산점도 그래프

```
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

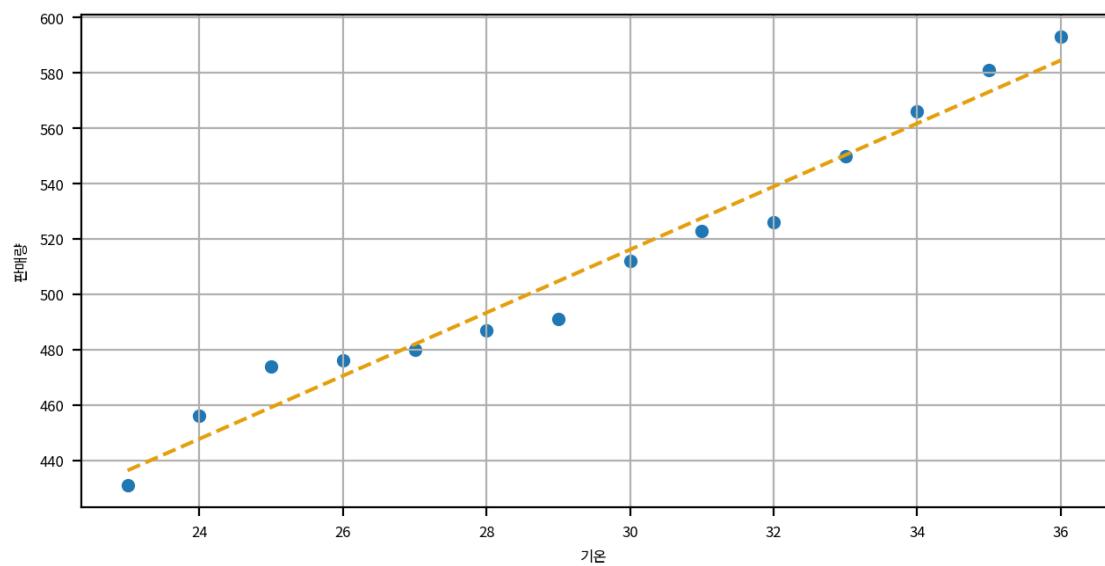
# 2-1) Scatter Plot 그리기
sb.scatterplot(data=origin, x='기온', y='판매량')

# 2-2) LinePlot 그리기
sb.lineplot(x=x, y=y, color="#e4a00c", linestyle="--")

# 3) 그래프 꾸미기
ax.set_title(expr, fontsize=14, pad=10)
ax.grid(True)                          # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                    # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료
```

$$y = 11.4 * x + 174.2$$



png



[LAB 06] 8. 데이터간의 관계 시각화 (2)



#01. 준비작업



1. 패키지 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
```



2. 그래프 초기화

```
my_dpi = 200 # 이미지 선명도(100~300)
fpath = "./NotoSansKR-Regular.ttf" # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(fpath) # 폰트의 글꼴을 시스템에 등록함
fprop = fm.FontProperties(fname=fpath) # 폰트의 속성을 읽어옴
fname = fprop.get_name() # 읽어온 속성에서 폰트의 이름만 추출
plt.rcParams['font.family'] = fname # 그래프에 한글 폰트 적용
plt.rcParams['font.size'] = 6 # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False # 그래프에 마이너스 깨짐 방지
```



3. 데이터 가져오기

```
origin = load_data('penguins')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/penguins.xlsx
[94m[desc] [0m 남극 팔мер 군도의 펭귄 3종에 대해 신체 치수와 서식지 정보(출처:
seaborn 내장 데이터)
```

field	description
species	펭귄 종
island	서식지
bill_length_mm	부리 길이

bill_depth_mm	부리 두께
flipper_length_mm	날개 길이
body_mass_g	몸무게
sex	성별

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181	3750	MALE
1	Adelie	Torgersen	39.5	17.4	186	3800	FEMALE
2	Adelie	Torgersen	40.3	18.0	195	3250	FEMALE
3	Adelie	Torgersen	36.7	19.3	193	3450	FEMALE
4	Adelie	Torgersen	39.3	20.6	190	3650	MALE
...
329	Gentoo	Biscoe	47.2	13.7	214	4925	FEMALE
330	Gentoo	Biscoe	46.8	14.3	215	4850	FEMALE
331	Gentoo	Biscoe	50.4	15.7	222	5750	MALE
332	Gentoo	Biscoe	45.2	14.8	212	5200	FEMALE
333	Gentoo	Biscoe	49.9	16.1	213	5400	MALE

334 rows × 7 columns

4. 명목형 변수에 대한 전처리

```
df = origin.astype({"species": "category", "island": "category",
                    "sex": "category"})
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 334 entries, 0 to 333
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   species          334 non-null    category
 1   island            334 non-null    category
 2   bill_length_mm   334 non-null    float64
 3   bill_depth_mm   334 non-null    float64
 4   flipper_length_mm 334 non-null    int64  
 5   body_mass_g      334 non-null    int64  
 6   sex               333 non-null    category
```

```
dtypes: category(3), float64(2), int64(2)
memory usage: 11.9 KB
```

#02. RegPlot

- ScatterPlot에 단일 회귀선(Linear Regression Line)을 자동으로 추가
- 회귀선 주변의 신뢰구간(confidence interval)도 함께 표시
- 두 변수 사이의 선형적 관계가 어느 정도인지 직관적으로 해석 가능
- 단일 그래프에서만 사용하며(hue파라미터를 지원하지 않음), 세부 커스터마이징이 쉬움

언제 쓰나?

→ “상관이 있나?”뿐 아니라 “어느 정도 기울기인가?”도 보고 싶을 때

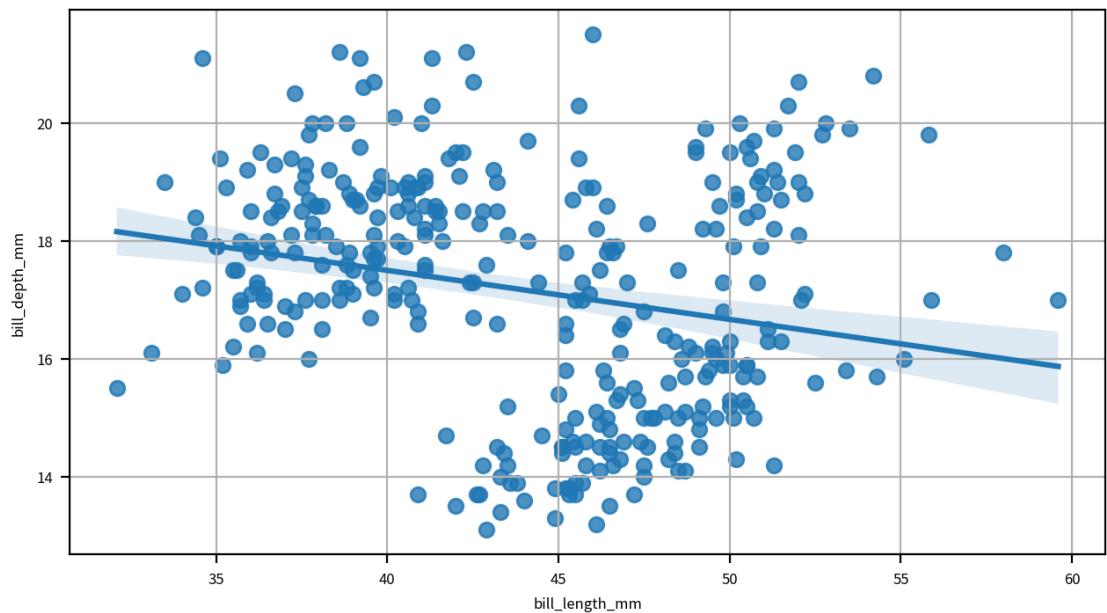
1. 부리 길이과 부리 두께의 관계

```
# 1) 그래프 초기화
width_px  = 1280                      # 그래프 가로 크기
height_px = 720                        # 그래프 세로 크기
rows      = 1                           # 그래프 행 수
cols      = 1                           # 그래프 열 수
figsize   = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) regplot 그리기
# `fit_reg=False` 파라미터를 추가할 경우 추세선이 표시되지 않는다. (scatterplot과
# 동일해짐)
# `scatter=False` 파라미터를 추가하면 산점도가 표시되지 않는다.(lineplot과 동일해
# 짐)
# `scatterplot()` 함수가 hue 파라미터를 적용하여 범주에 따라 구별할 수 있는 반면,
# RegPlot은 hue 파라미터를 적용할 수 없다.
sb.regplot(data=origin, x='bill_length_mm', y='bill_depth_mm')

# 3) 그래프 꾸미기
ax.grid(True)                          # 배경 격자 표시/숨김

# 4) 출력
plt.tight_layout()                     # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료
```



png



#03. LmPlot

- RegPlot의 기능을 확장한 형태 (scatterplot과 regplot의 결합)
- 여러 범주를 기준으로 행·열로 나눠(facet) 비교 가능
- 색(hue), 행(row), 열(col) 옵션을 통해 그룹 간 관계 차이를 시각화
- 관계 패턴을 다양한 하위집단으로 세분해 비교할 때 유용

언제 쓰나?

→ 성별/지역/연도 등 그룹별로 회귀선을 비교하고 싶을 때



1. 기본 사용 방법

산점도 그래프에 추세선을 추가함

기본 파라미터는 `regplot()` 메서드와 동일

`hue` 파라미터를 지원한다는 점에서 `regplot`과 차이를 보임

그래프 작성 코드가 기존 그래프와 다소 다르다

1) 그래프 초기화

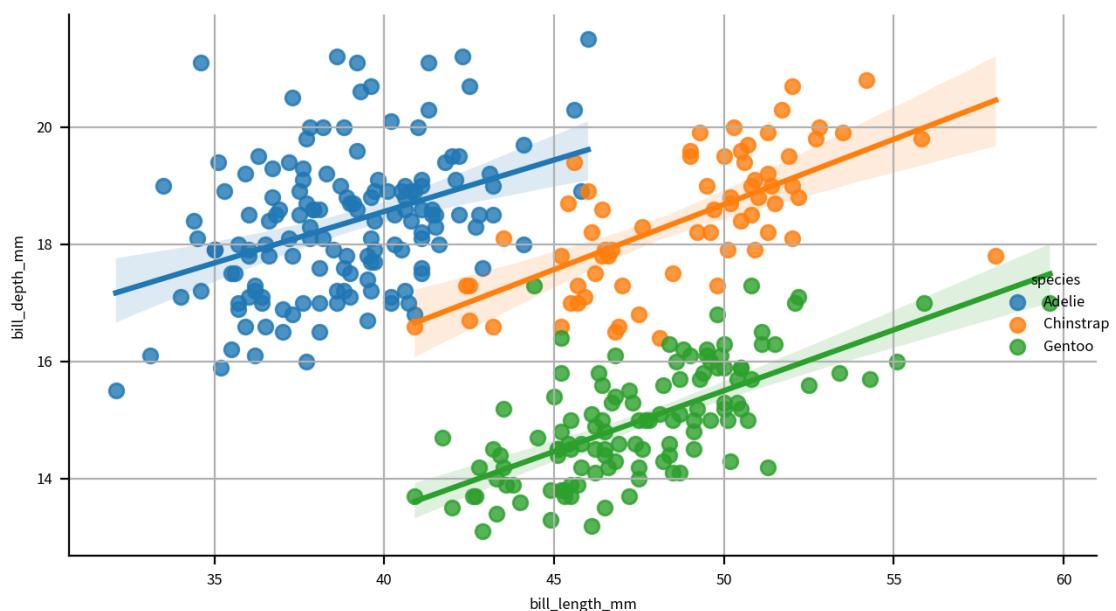
```
width_px = 1280 # 그래프 가로 크기
height_px = 720 # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)
```

```

# 2) LM Plot 그리기
# 그래프를 꾸미기 위해서 lmplot() 메서드로부터 리턴되는 객체(`g`)를 활용해야 함
g = sb.lmplot(data=df, x="bill_length_mm", y="bill_depth_mm",
               hue="species")
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
g.fig.set_figheight(figsize[1])
plt.grid()

# 3) 출력
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                 # 그래프 작업 종료

```



png



2. 조건별 병렬 시각화

범주에 따라 구분한 후 하위 변수를 사용하여 병렬 분할

```

# 1) 그래프 초기화
width_px   = 1280                      # 그래프 가로 크기
height_px  = 720                        # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) LM Plot 그리기
g = sb.lmplot(data=df, x="bill_length_mm", y="bill_depth_mm",
               hue="species", col='sex')
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])

```

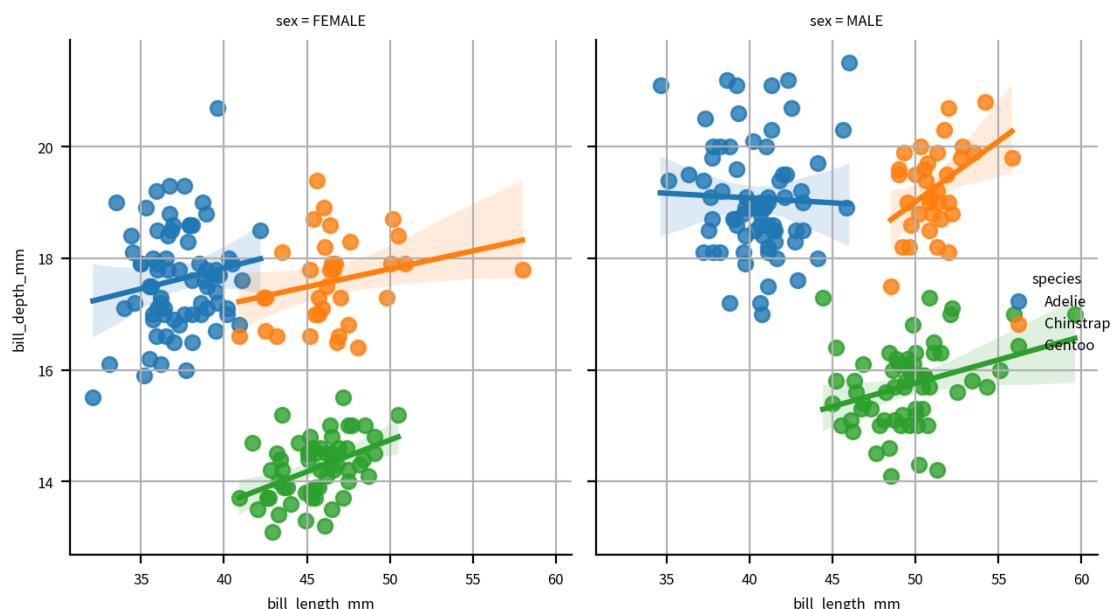
```

g.fig.set_figheight(figsize[1])

# 각 subplot에 grid 표시
for ax in g.axes.flatten():
    ax.grid(True)

# 3) 출력
plt.tight_layout()          # 여백 제거
plt.show()                  # 그래프 화면 출력
plt.close()                  # 그래프 작업 종료

```



png



3. 모든 조건에 따라 행, 열로 분할

row, col 파라미터를 사용한다.

```

# 1) 그래프 초기화
width_px   = 1800                      # 그래프 가로 크기
height_px  = 900                        # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) LM Plot 그리기
g = sb.lmplot(data=df, x="bill_length_mm", y="bill_depth_mm",
               hue="species", col="species", row='sex')
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
g.fig.set_figheight(figsize[1])

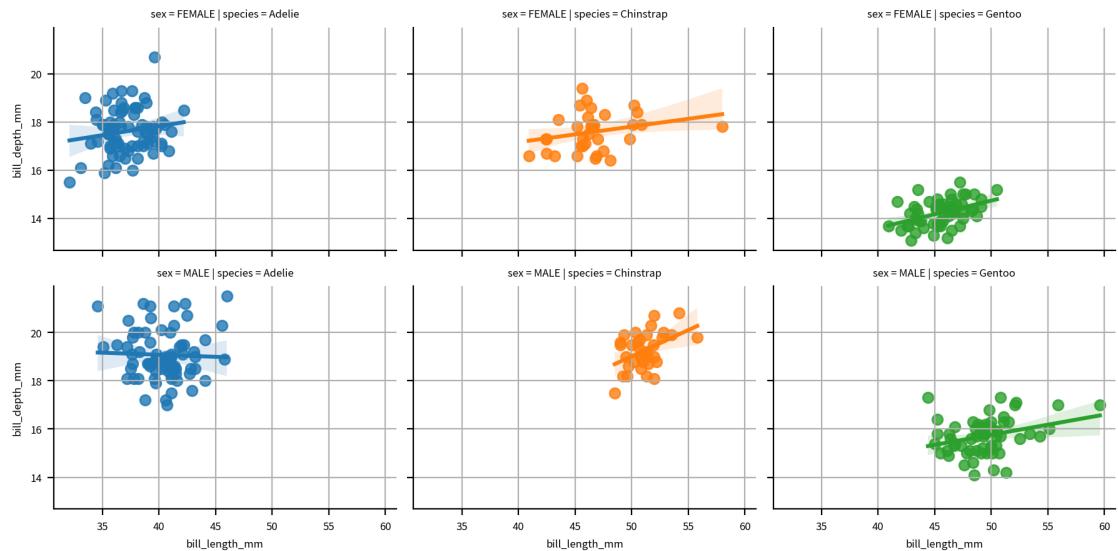
```

```

# 각 subplot에 grid 표시
for ax in g.axes.flatten():
    ax.grid(True)

# 3) 출력
plt.tight_layout()      # 여백 제거
plt.show()               # 그래프 화면 출력
plt.close()              # 그래프 작업 종료

```



png

#04. PairPlot (산점도 행)

- 모든 변수에 대한 교차 분석
- 전체 데이터의 구조와 변수 간 상관 관계를 한눈에 파악 가능
- 대각선 도표는 데이터의 주변 분포를 표시하기 위한 일변량 분포 도표(커널 밀도 곡선)이나 히스토그램이 그려진다.
- 탐색적 데이터 분석(EDA)에서 초기 전반 스캔용으로 매우 유용
- 다소 처리가 느리다.

언제 쓰나?

→ “전체 변수들이 서로 어떤 관계를 갖는지” 한 번에 파악할 때

1. 기본형

```

# 1) 그래프 초기화
width_px = 1200                      # 그래프 가로 크기

```

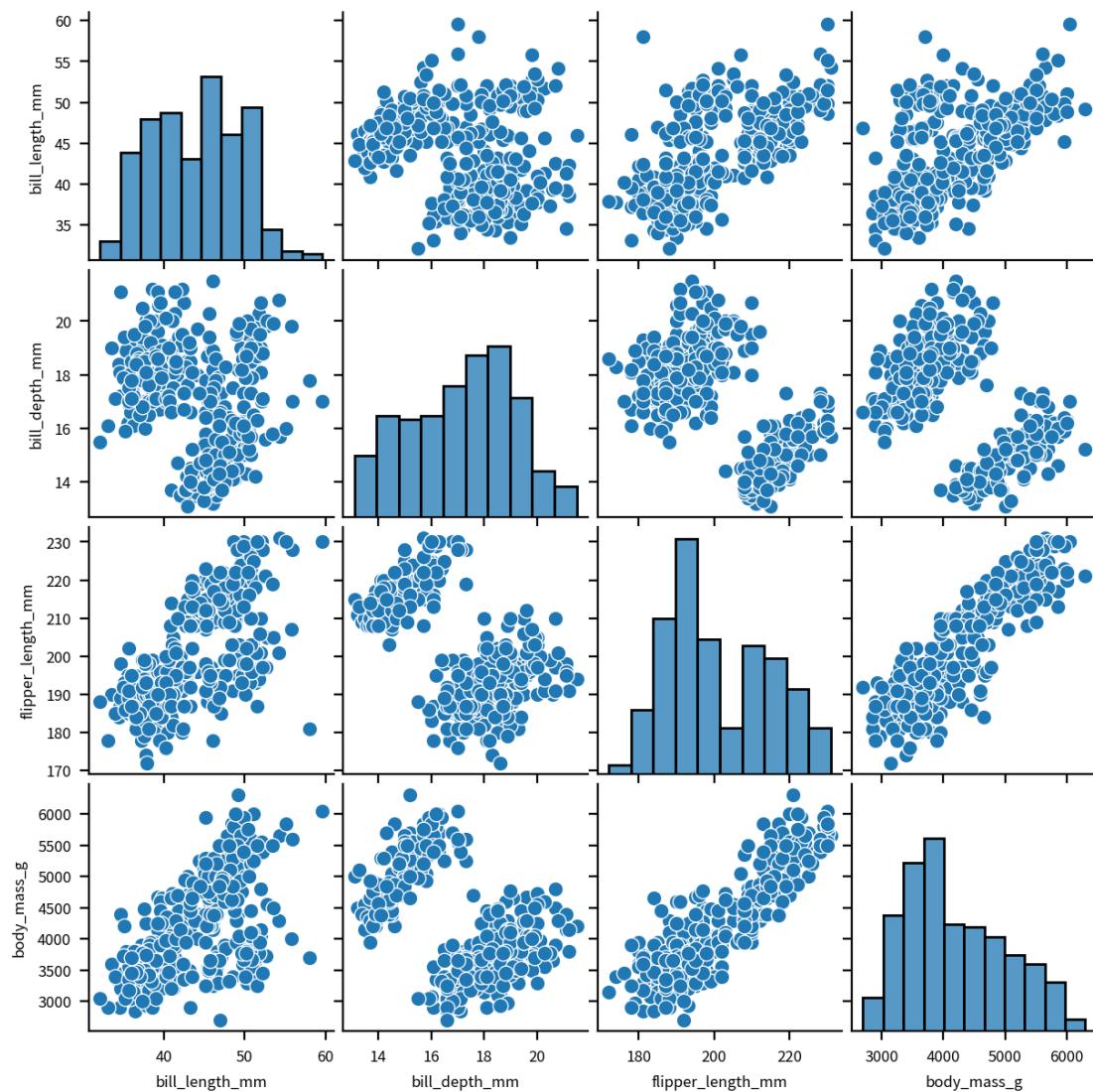
```

height_px = 1200 # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) Pair Plot 그리기
# `corner=True` 파라미터를 추가할 경우 아래쪽 삼각형만 플롯된다.
g = sb.pairplot(df)
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
g.fig.set_figheight(figsize[1])

# 3) 출력
plt.grid()
plt.show()
plt.close()

```



png



2. 범주별 구분

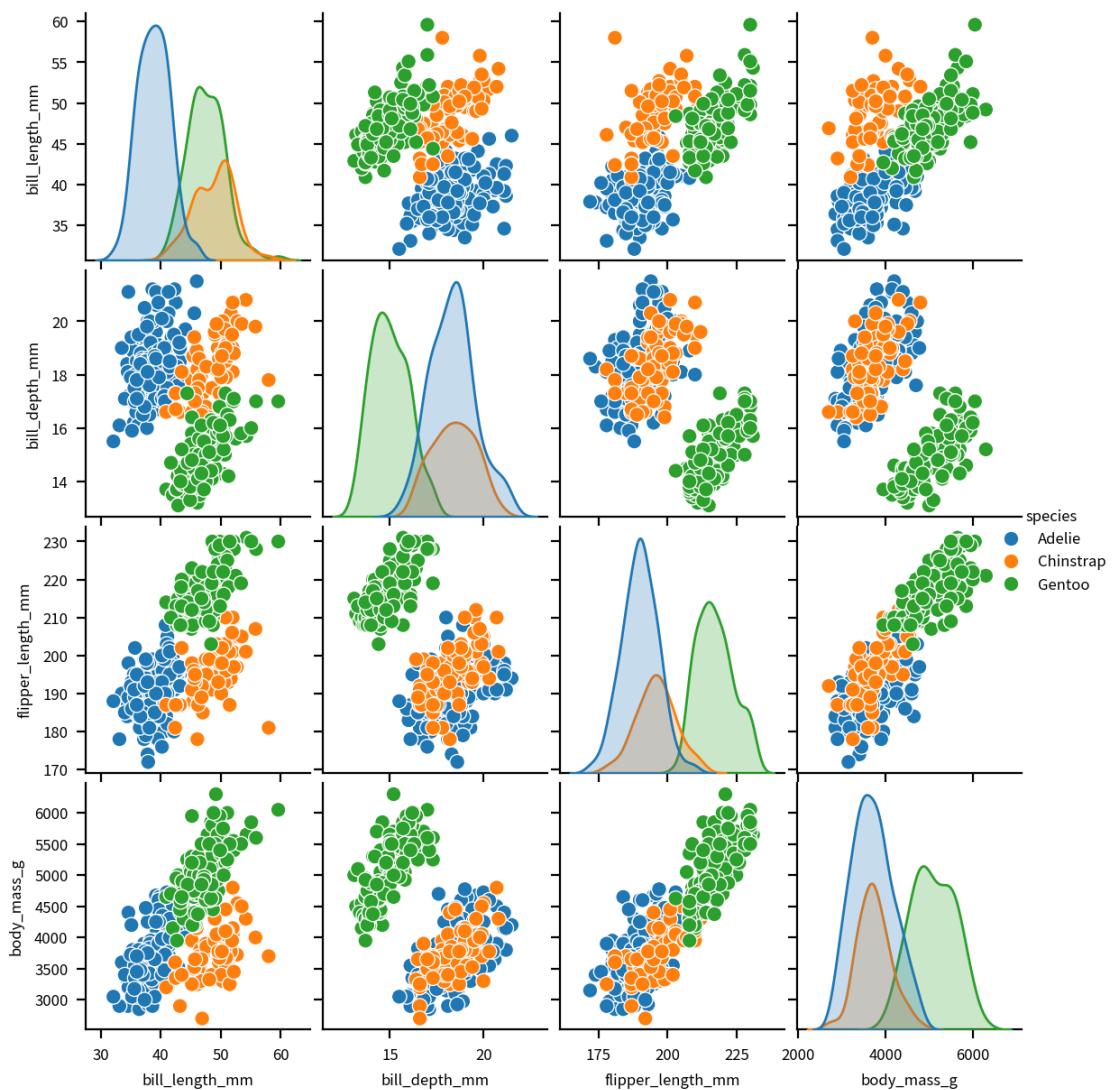
(1) hue 파라미터 적용

hue 파라미터에 변수를 할당하면 hue의 미론적 매핑이 추가되고 기본 주변 플롯이 계층화된 커널 밀도 추정(KDE)으로 변경된다.

```
# 1) 그래프 초기화
width_px  = 1200                                # 그래프 가로 크기
height_px = 1200                                 # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) Pair Plot 그리기
# `diag_kind` 파라미터에 `hist` 값을 적용한다.
# -> 적용 가능한 값: `auto`, `hist`, `kde`(기본값)
g = sb.pairplot(df, hue='species', diag_kind='kde')
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
g.fig.set_figheight(figsize[1])

# 3) 출력
plt.grid()
plt.show()
plt.close()
```



png



3. 선택적 변수 적용

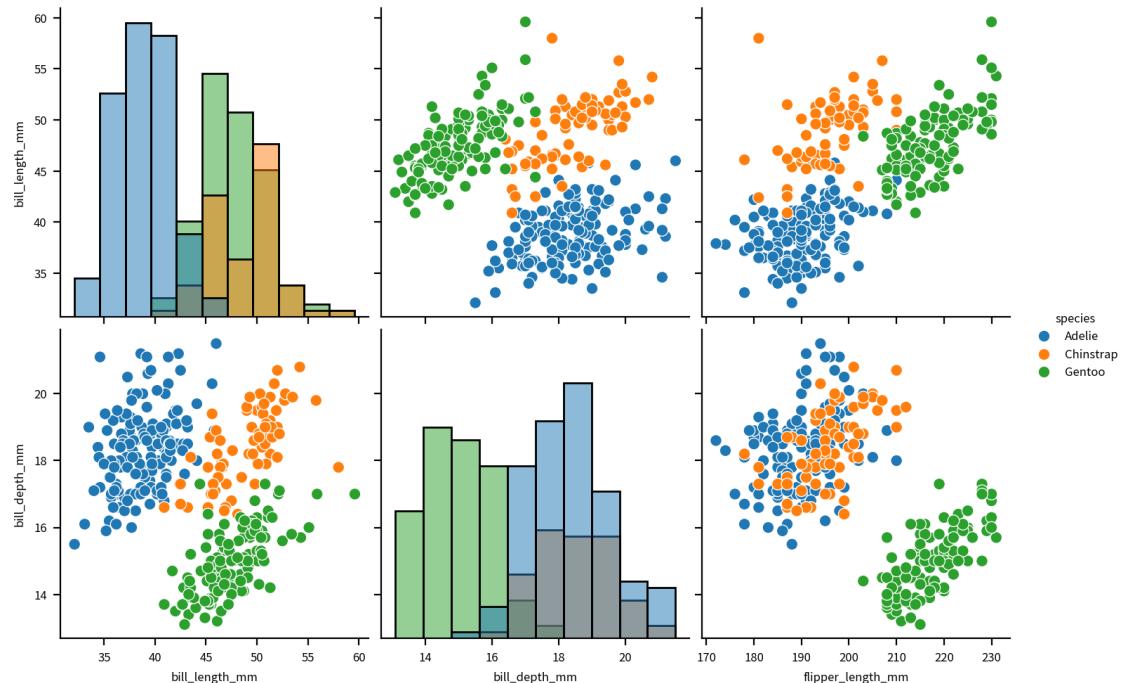
x_vars 파라미터와 y_vars 파라미터에 원하는 변수를 리스트 타입으로 지정한다.

```
# 1) 그래프 초기화
width_px  = 1600 # 그래프 가로 크기
height_px = 1000 # 그래프 세로 크기
figsize = (width_px / my_dpi, height_px / my_dpi)

# 2) Pait Plot 그리기
g = sb.pairplot(df, hue='species', diag_kind='hist',
                 x_vars=["bill_length_mm", "bill_depth_mm",
                         "flipper_length_mm"],
                 y_vars=["bill_length_mm", "bill_depth_mm"])
g.fig.set_dpi(my_dpi)
g.fig.set_figwidth(figsize[0])
```

```
g.fig.set_figheight(figsize[1])
```

```
# 3) 출력  
plt.grid()  
plt.show()  
plt.close()
```



png



4. 데이터를 그룹별로 묶어서 표시하기

pairplot() 메서드가 리턴하는 객체를 받아서 map_lower() 메서드를 호출한다.

map_lower() 메서드에 다른 종류의 함수 이름을 적용하면 대각선 기준으로 서로 다른 종류의 시각화 결과물을 표시할 수 있다.

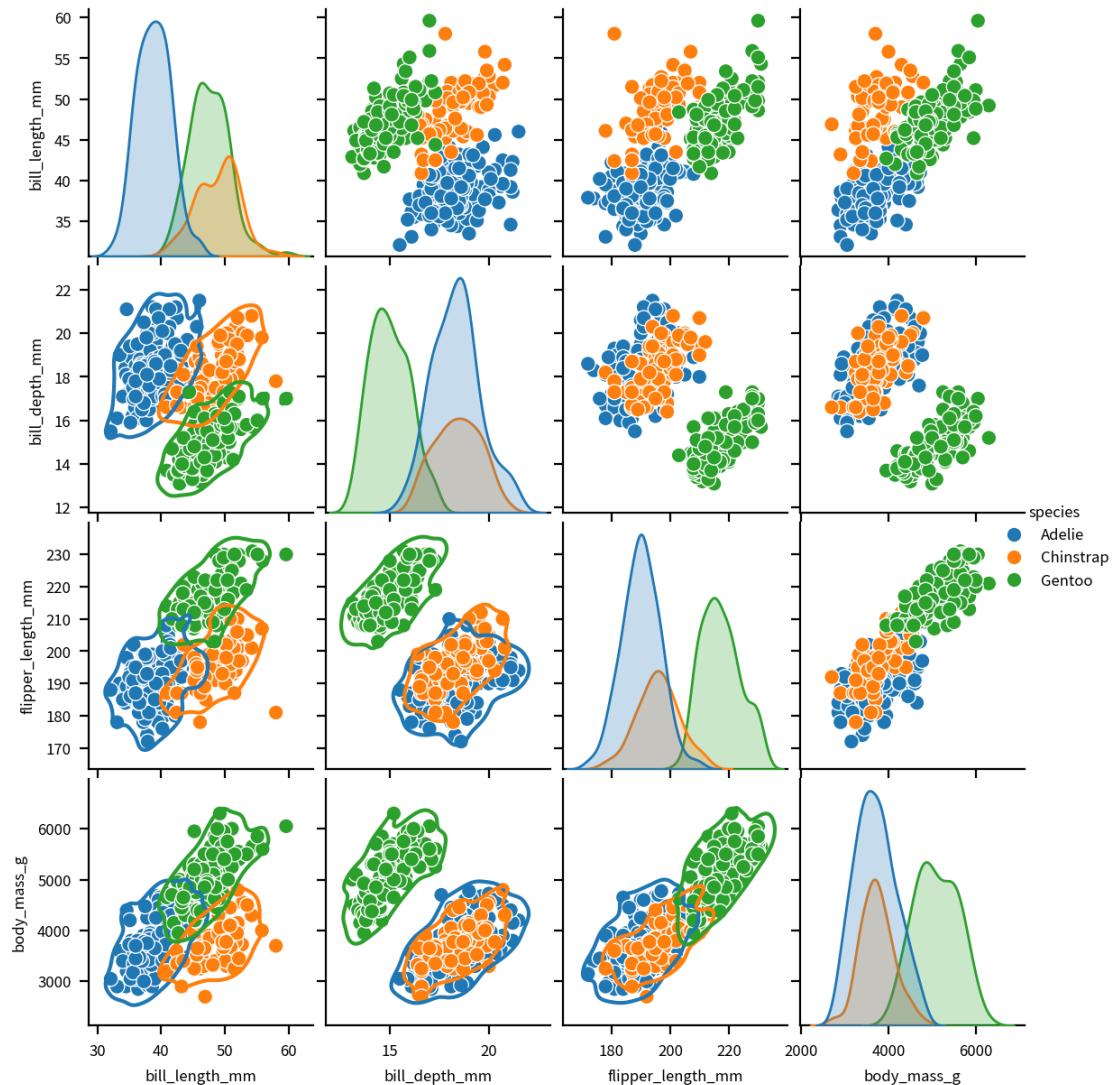
```
# 1) 그래프 초기화
```

```
width_px = 1200 # 그래프 가로 크기  
height_px = 1200 # 그래프 세로 크기  
figsize = (width_px / my_dpi, height_px / my_dpi)
```

```
# 2) Pair Plot 그리기
```

```
g = sb.pairplot(df, hue='species')  
g.map_lower(sb.kdeplot, levels=1, color=0.2)  
g.fig.set_dpi(my_dpi)  
g.fig.set_figwidth(figsize[0])  
g.fig.set_figheight(figsize[1])
```

```
# 3) 출력  
plt.grid()  
plt.show()  
plt.close()
```



png



[LAB-07] 1. 지도 시작화



#01. 준비작업



1. 라이브러리 참조

folium 패키지가 설치되어 있어야 한다.

```
from hossam import load_data
from os import path, mkdir
import folium
```



#02. 지도 표현하기



1. 지도 객체 생성

지도의 중심이 되는 위도와 경도를 설정

```
MY_PLACE = [37.4935982, 127.0327129]
```



2. 지도 불러오기

```
# zoom_start: 배율 1~22
map_osm1 = folium.Map(location=MY_PLACE, zoom_start=17)

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

# 지도가 표시된 웹 페이지 파일이 저장됨
# --> 결과 확인은 윈도우 폴더 창에서 직접 웹 페이지 파일 더블 클릭
map_osm1.save('output/map_osm1.html')
```



3. 지도 객체에 마커 추가

1) 일반 마커

아이콘 색상값 종류

'lightgreen', 'darkgreen', 'darkblue', 'cadetblue', 'orange', 'lightred', 'darkred', 'green', 'blue',
'black', 'lightblue', 'white', 'lightgray', 'red', 'pink', 'beige', 'gray', 'purple', 'darkpurple'

```
# 새로운 지도 객체 생성
map_osm2 = folium.Map(location=MY_PLACE, zoom_start=17)

# 마커 객체 생성
# -> 마커가 표시될 위치 [위도, 경도], 클리시 표시될 메시지, 아이콘 색상
my_marker = folium.Marker(MY_PLACE,
                           popup='아이티윌 교육센터',
                           icon=folium.Icon(color='darkred'))

my_marker.add_to(map_osm2) # 마커 객체를 지도에 추가함

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

# 지도가 표시된 웹 페이지 파일이 저장됨
# --> 결과 확인은 윈도우 폴더 창에서 직접 웹 페이지 파일 더블 클릭
map_osm2.save('output/map_osm2.html')
```

2) 사용자 지정 아이콘, HTML 팝업

```
# 새로운 지도 객체 생성
map_osm3 = folium.Map(location=MY_PLACE, zoom_start=17)

# HTML을 사용한 팝업
popup_html = folium.Popup("<div style='white-space: nowrap'><h3>아이티
    윌 교육센터</h3><img src='https://www.itwill.co.kr/css/wvtex/
    img/wvUser/logo.png' width='100%'><br/><p>tel: <a
    href='tel:02-6255-8002'>02-6255-8002</a></p><p>서울특별시 강남구
    역삼로 120 2층 (역삼동, 성보역삼빌딩)</p></div>",
    parse_html=False)

# 사용자 지정 아이콘 이미지 사용
# --> 온라인 상의 URL, 내 컴퓨터 상의 상대, 절대경로 모두 가능함
icon_img = folium.features.CustomIcon('https://data.hossam.kr/
    favicon.png', icon_size=(50, 50))
```

```

# 마커 객체 생성
custom_marker = folium.Marker(MY_PLACE,
                             popup=popup_html,
                             icon=icon_img)

custom_marker.add_to(map_osm3) # 마커 객체를 지도에 추가함

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

map_osm3.save('output/map_osm3.html') #파일이 저장될 위치

```

3) 원형 마커(범위지정)

```

# 새로운 지도 객체 생성
map_osm4 = folium.Map(location=MY_PLACE, zoom_start=17)

# 마커 객체 생성
my_marker = folium.Marker(MY_PLACE, icon=folium.Icon(color='orange'))

# 원형마커
circle_marker = folium.CircleMarker(MY_PLACE,
                                      radius=100,                      # 범위(m단위)
                                      color='#3186cc',                  # 선 색상
                                      fill_color='#3186cc')             # 면 색상

# 원형 마커 위에 아이콘 마커 올리기
circle_marker.add_to(map_osm4)
my_marker.add_to(map_osm4)

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

map_osm4.save('output/map_osm4.html') #파일이 저장될 위치

```



#03. (예제) 서울의 고등학교 분포 확인하기



1. 데이터 준비하기

전국 초,중,고 학교 위치 데이터 (데이터 로드에 다소 시간이 소요됨)

```
origin = load_data('schools')
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab07/schools.xlsx
[94m[desc] [0m 전국 초,중,고 학교 위치 데이터 (출처: 공공데이터 포털)
[91m[!] Cannot read metadata [0m
```

	학교ID	학 교 명	학 교 급 구 분	설립일자	설 립 형 태	본 교 분 교 구 분	운 영 상 태	소재 지지 번주 소	소재 지도 로명 주소	시도교육청 코드	시 도 교 育 청 명	교 育 지 원 청 원 청	생성	
0	B000004204	한 울 초 등 학 교	초 등 학 교	2008-09-01	공 립	본 교	운 영	경 기 도 화 성 시 향 남 읍	경 기 도 화 성 시 향 남 읍 행 정 중 앙 1 로 25. 한 울 초 등 학 교 (향 남 읍)	7530000	경 기 도 교 育 청	7679000	경 기 도 화 성 오 산 교 育 지 원 청	2013-1
1	B000011476	수 원 농 생 명 과 학 고 등 학 교	고 등 학 교	1936-07-01	공 립	본 교	운 영	경 기 도 수 원 시 장 안 구 영 화 동 109	경 기 도 수 원 시 장 안 구 광 교 산 로 13 (영 화	7530000	경 기 도 교 育 청	7541000	경 기 도 수 원 교 育 지 원 청	2013-1

2	B000009647	녹양중학교	중학교	2008-03-01	공립	본교	운영	경기도의정부시녹양동191-5	경기도의정부체육로187.녹양중학교(녹양동)	7530000	경기도교육청	경기도의정부교육지원청	2013-1
3	B000005955	초락초등학교	초등학교	1959-04-02	공립	본교	운영	충청남도당진시석문면샛터말길35(석문면)	충청남도당진시석문면샛터말길35(석문면)	8140000	충청남도교육청	충청남도당진교육지원청	2013-1
4	B000005385	상봉초등학교	초등학교	1946-09-01	공립	본교	운영	충청북도청주시흥덕구오송읍상봉길9.상봉초등학교(오송읍)	충청북도청주시흥덕구오송읍상봉길9.상봉초등학교(오송읍)	8000000	충청북도교육청	충청북도청주교육지원청	2013-1

...
11983	B000003371	대전 옥계초등학교	초등학교	1982-12-09	공립	본교	운영	대전 광역시 중구 옥계동 65	대전 광역시 중구 모암로 35 (옥계동·대전 옥계초등학교)	7430000	대전 광역시 교육청	7441000	대전 광역시 동부교육지원청	2013-1	
11984	B000003345	성덕초등학교	초등학교	2011-03-11	공립	본교	운영	광주 광역시 광산구 풍영로 313 . 성덕초등학교 (장덕동) 1042	광주 광역시 광산구 풍영로 313 . 성덕초등학교 (장덕동)	7380000	광주 광역시 교육청	7401000	광주 광역시 서부교육지원청	2013-1	
11985	B000005441	이월초등학교	초등학교	1920-04-01	공립	본교	운영	충청북도 진천군 이월면 송림6길 26 . 이월초등학교 (이월면) 667	충청북도 진천군 이월면 송림6길 26 . 이월초등학교 (이월면)	8000000	충청북도 교육청	8081000	충청북도 진천교육지원청	2013-1	
11986	B000009875	해안중학교	중학교	1979-03-08	공립	본교	운영	강원도 양구군 해안면 현	강원도 양구군 해안면	7800000	강원도 교	7951000	강원도 양구교	2013-1	

								리 143	편치 볼로 1279 (해 안 면)		육 청		육 지 원 청
11987	B000011237	초 계 중 학 교	중 학 교	1952-06-01	공 립	본 교	운 영	경상 남도 합천 군 초계 면 초계 중앙 로 83. 초계 중학 교 (초 계 면. 초계 중학 교)	9010000	경 상 남 도 교 育 청	9221000	경 상 남 도 합 천 교 育 지 원 청	2013-1

11988 rows × 20 columns



2. 데이터 전처리

1) 사용할 필드만 추출

```
df = origin.filter(['학교명', '학교급구분', '소재지도로명주소', '위도', '경  
도'])  
df
```

	학교명	학교급구 분	소재지도로명주소	위도	경도
0	한울초등학교	초등학교	경기도 화성시 향남읍 행정중앙1로 25. 한울초등학교 (향 남읍)	37.126961	126.917854
1	수원농생명과학고 등학교	고등학교	경기도 수원시 장안구 광교산로 13 (영화동.농생명과학고 등학교)	37.295154	127.019450
2	녹양중학교	중학교	경기도 의정부시 체육로 187. 녹양중학교 (녹양동)	37.761864	127.028084
3	초락초등학교	초등학교	충청남도 당진시 석문면 샛터말길 35 (석문면)	36.993080	126.510472

4	상봉초등학교	초등학교	충청북도 청주시 흥덕구 오송읍 상봉길 9 . 상봉초등학교 (오송읍)	36.638251	127.286142
...
11983	대전옥계초등학교	초등학교	대전광역시 중구 모암로 35 (옥계동. 대전옥계초등학교)	36.301199	127.449039
11984	성덕초등학교	초등학교	광주광역시 광산구 풍영로 313 . 성덕초등학교 (장덕동)	35.199148	126.814301
11985	이월초등학교	초등학교	충청북도 진천군 이월면 송림6길 26 . 이월초등학교 (이월면)	36.931076	127.431922
11986	해안중학교	중학교	강원도 양구군 해안면 편치불로 1279 (해안면)	38.283771	128.135686
11987	초계중학교	중학교	경상남도 합천군 초계면 초계중앙로 83 . 초계중학교 (초계면. 초계중학교)	35.560066	128.270502

11988 rows × 5 columns

```
df_test = df.copy()
df_test
```

	학교명	학교급구 분	소재지도로명주소	위도	경도
0	한울초등학교	초등학교	경기도 화성시 향남읍 행정중앙1로 25 . 한울초등학교 (향남읍)	37.126961	126.917854
1	수원농생명과학고등학교	고등학교	경기도 수원시 장안구 광교산로 13 (영화동. 농생명과학고등학교)	37.295154	127.019450
2	녹양중학교	중학교	경기도 의정부시 체육로 187 . 녹양중학교 (녹양동)	37.761864	127.028084
3	초락초등학교	초등학교	충청남도 당진시 석문면 샛터말길 35 (석문면)	36.993080	126.510472
4	상봉초등학교	초등학교	충청북도 청주시 흥덕구 오송읍 상봉길 9 . 상봉초등학교 (오송읍)	36.638251	127.286142
...
11983	대전옥계초등학교	초등학교	대전광역시 중구 모암로 35 (옥계동. 대전옥계초등학교)	36.301199	127.449039
11984	성덕초등학교	초등학교	광주광역시 광산구 풍영로 313 . 성덕초등학교 (장덕동)	35.199148	126.814301
11985	이월초등학교	초등학교	충청북도 진천군 이월면 송림6길 26 . 이월초등학교 (이월면)	36.931076	127.431922
11986	해안중학교	중학교	강원도 양구군 해안면 편치불로 1279 (해안면)	38.283771	128.135686
11987	초계중학교	중학교	경상남도 합천군 초계면 초계중앙로 83 . 초계중학교 (초계면. 초계중학교)	35.560066	128.270502

11988 rows × 5 columns

2) 서울시의 고등학교만 추출

학교급구분 필드 값이 고등학교이고, 소재지도로명주소에 서울 이라는 단어가 포함된 경우

LIKE 연산

컬럼이름.str.contains('검색어')

```
df2 = df.query("학교급구분 == '고등학교' and 소재지도로명주  
소.str.contains('서울')")
```

```
df2
```

	학교명	학교급 구분	소재지도로명주소	위도	경도
6	경기고등학교	고등학 교	서울특별시 강남구 영동대로 643 . 경기고등학교 (삼성동)	37.517565	127.056076
89	대일관광고등학 교	고등학 교	서울특별시 양천구 신정이펜1로 11 . 대일관광고등학교 (신정 동. 대일관광고등학교)	37.511414	126.834907
97	한광고등학교	고등학 교	서울특별시 강서구 등촌로13길 110 (화곡동)	37.538844	126.857922
98	상일미디어고등 학교	고등학 교	서울특별시 강동구 천호대로219길 61 . 상일미디어고등학교 (상일동)	37.549470	127.170767
118	선정국제관광고 등학교	고등학 교	서울특별시 은평구 서오릉로20길 19 . 선정국제관광고등학교 (갈현동)	37.618705	126.909032
...
11429	성수고등학교	고등학 교	서울특별시 성동구 서울숲길 18 . 성수고등학교 (성수동1가)	37.547342	127.038253
11451	대진여자고등학 교	고등학 교	서울특별시 노원구 공릉로 438 . 대진여자고등학교 (중계동)	37.646174	127.067197
11564	도선고등학교	고등학 교	서울특별시 성동구 마장로 156 (하왕십리동)	37.566844	127.026996
11600	금호고등학교	고등학 교	서울특별시 성동구 금호로 118 (금호동1가. 금호고등학교)	37.553621	127.023434
11929	서울인공지능고 등학교	고등학 교	서울특별시 송파구 양산로 21 (거여동)	37.491753	127.142331

320 rows × 5 columns



3. 데이터 시각화

```
# zoom_start: 배율 1~22 (여기서는 출력 안함)
map_osm5 = folium.Map(location=MY_PLACE, zoom_start=12)

html = "<font color='green' style='white-space: nowrap'><b>%s</b></font>"

# 데이터프레임의 행 수만큼 반복하면서 마커 생성
for i in df2.index:
    # 행 우선 접근 방식으로 값 추출하기
    name = df2.loc[i, '학교명']
    lat = df2.loc[i, '위도']
    lng = df2.loc[i, '경도']

    # 추출한 정보를 지도에 표시
    popup_html = folium.Popup(html % name, parse_html=False)
    marker = folium.Marker([lat,lng], popup=popup_html)
    marker.add_to(map_osm5)

# 웹 페이지 파일이 저장될 폴더 생성
if not path.exists('output'):
    mkdir('output')

map_osm5.save('output/map_osm5.html') #파일이 저장될 위치
```



[LAB-07] 2. SVG 지도 시각화 (서울)



SVG(Scalable Vector Graphics)

JPEG, PNG와 같은 그래픽 포맷(Graphic format)의 하나

벡터 기반이기 때문에 리사이징이 되어도 전혀 깨지지 않고 모든 해상도에서 자유자재로 활용할 수 있다.

SVG파일 포맷은 XML로 구성되어 있기 때문에 BeautifulSoup 패키지를 활용하여 HTML 파싱과 같은 구현과정을 통해 원하는 부분을 취득, 변형 할 수 있다.



#01. 준비작업



1. 패키지 참조

bs4 패키지가 필요하다.

| 웹 페이지 데이터 수집 단원을 통해 이미 설치되어 있음

jenksipy 패키지 설치가 필요하다 (연구과제용)

```
from hossam import load_data

# jupyter 상에서 SVG 이미지를 표시하기 위한 패키지(jupyter 기본 내장 패키지)
from IPython.display import SVG

# TAG로부터 원하는 내용을 추출하는 클래스 -> SVG 이미지의 핸들링을 위함
from bs4 import BeautifulSoup

# 연구과제용
import jenksipy
```



2. 데이터 가져오기

```
origin = load_data("senior_lsf")
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/senior_lsf.xlsx  
[94m[desc] [0m 서울시의 행정구역별 노인복지시설의 수를 조사한 가상의 데이터  
[91m[!] Cannot read metadata [0m
```

	지역명	복지시설
0	Jongno-gu	61
1	Jung-gu	53
2	Yongsan-gu	110
3	Seongdong-gu	155
4	Gwangjin-gu	103
5	Dongdaemun-gu	146
6	Jungnang-gu	128
7	Seongbuk-gu	158
8	Gangbuk-gu	111
9	Dobong-gu	139
10	Nowon-gu	252
11	Eunpyeong-gu	154
12	Seodaemun-gu	103
13	Mapo-gu	160
14	Yangcheon-gu	192
15	Gangseo-gu	215
16	Guro-gu	192
17	Geumcheon-gu	75
18	Yeongdeungpo-gu	208
19	Dongjak-gu	143
20	Gwanak-gu	127
21	Seocho-gu	129
22	Gangnam-gu	184
23	Songpa-gu	173
24	Gangdong-gu	140

#02. 지도 이미지 가져오기

위키미디어에서 Seoul districts.svg 키워드로 검색하여 서울 지도 이미지를 내려받아 map_seoul.svg라는 이름으로 작업 폴더에 추가하고 open() 함수로 파일을 읽어올 수 있다.

https://commons.wikimedia.org/wiki/File:Seoul_districts.svg?uselang=ko

혹은 아래의 URL에 접속한 후 Ctrl+S를 눌러서 파일을 저장한다.

https://data.hossam.kr/data/lab07/map_seoul.svg

저장된 파일은 /작업폴더/svg/map_seoul.svg 경로에 저장한다.



1. 지도 이미지 읽어오기

```
map_file_path = "svg/map_seoul.svg"

try:
    with open(map_file_path, 'r', encoding="utf-8") as f:
        map_svg = f.read()
    print("파일 읽어오기 완료")

    # 출력되는 내용이 많으므로 내용 확인 후 주석으로 막아두자.
    #print(map_svg)
except Exception as e:
    print("파일 읽어오기 에러:", e)
```

파일 읽어오기 완료



2. 이미지 확인

```
SVG(map_svg)
```



svg

#03. 데이터 시각화

1. 단계별 색상 팔레트 만들기

단계는 분석가가 임의로 정한다.

색상값을 1단계 ~ 높은단계 순으로 점점 진한 색상이 되도록 구성

사용할 색상값 (단계별로 6개 색상 준비)

```
colors = ['#F1EEF6', '#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77',  
         '#980043']
```



2. BeautifulSoup 객체 생성

svg파일의 내용을 BeautifulSoup객체로 변환

```
soup = BeautifulSoup(map_svg, features="xml")
# 출력되는 내용이 많으므로 내용 확인 후 주석으로 막아두자.
#soup
```



[3] 구 단위로 추출

id속성을 갖는 path 태그 가져오기

```
# soup.select() 메서드의 리턴값은 항상 리스트이다.
# → path 태그이면서 id속성을 갖는 요소를 리스트로 반환
#   <path id="???" ... >
path_list = soup.select('path[id]')
print("가져온 도형의 수: ", len(path_list))
```

가져온 도형의 수: 25



[3] 지도에서 확인한 지역명 수 만큼 반복

```
for p in path_list:
    #print(p)

    지역명 = p['id']
    #print(지역명)

    복지시설수 = origin.query('지역명 == @지역명')['복지시설'].values[0]
    print(지역명, " → ", 복지시설수)

    # 복지시설 수에 따라 단계값 설정 (단계는 색상값의 수에 따른)
    if 복지시설수 > 250:    color_index = 5
    elif 복지시설수 > 200:  color_index = 4
    elif 복지시설수 > 150:  color_index = 3
    elif 복지시설수 > 100:  color_index = 2
    elif 복지시설수 > 50:   color_index = 1
    else:                  color_index = 0

    # svg 이미지의 면 색상 변경
    p['fill'] = colors[color_index]
```

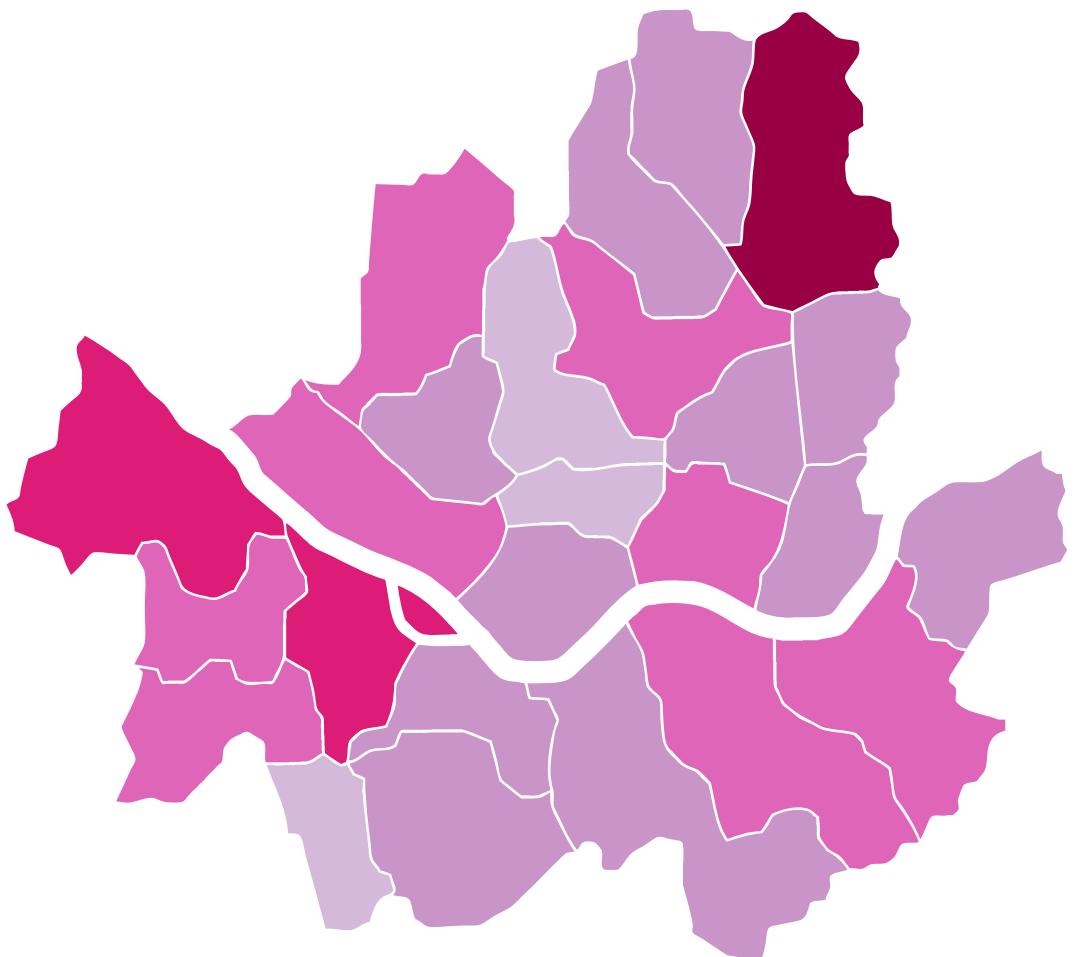
Dobong-gu → 139
Dongdaemun-gu → 146
Dongjak-gu → 143
Eunpyeong-gu → 154
Gangbuk-gu → 111
Gangdong-gu → 140
Gangseo-gu → 215
Geumcheon-gu → 75
Guro-gu → 192
Gwanak-gu → 127
Gwangjin-gu → 103
Gangnam-gu → 184
Jongno-gu → 61
Jung-gu → 53
Jungnang-gu → 128
Mapo-gu → 160
Nowon-gu → 252
Seocho-gu → 129
Seodaemun-gu → 103
Seongbuk-gu → 158
Seongdong-gu → 155
Songpa-gu → 173
Yangcheon-gu → 192
Yeongdeungpo-gu → 208
Yongsan-gu → 110



[4] 재구성된 내용을 토대로 새로운 svg 소스코드 얻기

```
# bs4 객체의 내용을 문자열로 리턴
new_seoul_svg = soup.prettify()

# jupyter에서 svg 이미지 표시하기
# 사용방법 -> SVG(소스문자열) 혹은 SVG(파일경로)
SVG(new_seoul_svg)
```



svg



[5] 생성된 이미지를 파일로 저장해야 하는 경우

```
# 저장된 파일은 윈도우 폴더창에서 직접 더블클릭 해서 웹 브라우저를 통해 확인해야 한다.  
with open('svg/new_seoul_svg.svg', 'w', encoding="utf-8") as f:  
    f.write(new_seoul_svg)
```



지도 시각화 연구과제

covid19_clinic 데이터는 2021년 05월 27일 기준 전국의 코로나 검사가 가능한 병원의 목록이다.

서울의 구 단위로 코로나 검사가 가능한 병원의 분포를 5단계로 구분하여 시각화 하시오.

색상 단계값은 아래의 리스트를 활용하세요.

```
colors = ['#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77', '#980043']
```



결과물 예시

img

img



#01. 데이터 불러오기

```
origin = load_data("covid19_clinic")
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab07/covid19_clinic.xlsx
[94m[desc] [0m 서울시의 Covid19 검진 가능 진료소 데이터 (출처: 공공데이터 포털)
[91m[!] Cannot read metadata [0m
```

	시도	시군구	의료기관명	주소
0	서울	강남구	강남구보건소	서울 강남구 삼성동(삼성2동) 8 강남구보건소
1	서울	강남구	삼성서울병원	서울 강남구 일원로81 삼성서울병원
2	서울	강남구	강남세브란스병원	서울 강남구 언주로211 강남세브란스병원
3	서울	강동구	강동구보건소	서울 강동구 성내동 541-2
4	서울	강동구	중앙보훈병원	서울 강동구 진황도로 61길 53
...
621	제주	제주시	한마음병원	연신로 52
622	제주	제주시	한국병원	서광로 193
623	제주	제주시	중앙병원	월랑로 91

624	제주	제주시	제주시(동부보건소)	제주 제주시 구좌읍 김녕리 1697-1 동부보건소 동부보건소
625	제주	제주시	제주시(서부보건소)	제주특별자치도 제주시 한림읍 한림리 966-1번지 (한림리)

626 rows × 4 columns

#02. 데이터 전처리



1. 필요한 변수만 추출

```
#seoul_df = origin.query('시도 == "서울"')
seoul_df = origin[origin["시도"] == '서울']
seoul_df
```

	시도	시군구	의료기관명	주소
0	서울	강남구	강남구보건소	서울 강남구 삼성동(삼성2동) 8 강남구보건소
1	서울	강남구	삼성서울병원	서울 강남구 일원로81 삼성서울병원
2	서울	강남구	강남세브란스병원	서울 강남구 언주로211 강남세브란스병원
3	서울	강동구	강동구보건소	서울 강동구 성내동 541-2
4	서울	강동구	중앙보훈병원	서울 강동구 진황도로 61길 53
...
66	서울	종로구	서울직십자병원	서울시 종로구 평동 164
67	서울	종로구	서울대학교병원	서울시 종로구 대학로 101(연건동)
68	서울	중랑구	중랑구보건소	서울 중랑구 신내2동 662 중랑구청
69	서울	중랑구	서울의료원	중랑구 신내로 156
70	서울	중랑구	녹색병원	중랑구 사가정로 49길 53

71 rows × 4 columns



2. 그룹별 집계

```
group_df = seoul_df.filter(["시군구", "의료기관명"]).groupby('시군구').count()
group_df.head()
```

	의료기관명
--	-------

시군구	
강남구	3
강동구	4
강북구	1
강서구	1
관악구	2



3. 구 이름에 대한 영문명 처리

앞 예제의 출력결과를 데이터셋의 구 이름과 직접 연결함

```
df = group_df.rename(
    columns={"의료기관명": "의료기관수"},
    index={
        "은평구": "Eunpyeong-gu",
        "영등포구": "Yeongdeungpo-gu",
        "동대문구": "Dongdaemun-gu",
        "강동구": "Gangdong-gu",
        "종로구": "Jongno-gu",
        "양천구": "Yangcheon-gu",
        "강남구": "Gangnam-gu",
        "중구": "Jung-gu",
        "송파구": "Songpa-gu",
        "성동구": "Seongdong-gu",
        "서초구": "Seocho-gu",
        "중랑구": "Jungnang-gu",
        "동작구": "Dongjak-gu",
        "노원구": "Nowon-gu",
        "구로구": "Guro-gu",
        "서대문구": "Seodaemun-gu",
        "도봉구": "Dobong-gu",
        "성북구": "Seongbuk-gu",
        "금천구": "Geumcheon-gu",
        "광진구": "Gwangjin-gu",
        "용산구": "Yongsan-gu",
        "관악구": "Gwanak-gu",
        "강서구": "Gangseo-gu",
        "강북구": "Gangbuk-gu",
        "마포구": "Mapo-gu"
    }
)
```

df

	의료기관수
시군구	
Gangnam-gu	3
Gangdong-gu	4
Gangbuk-gu	1
Gangseo-gu	1
Gwanak-gu	2
Gwangjin-gu	2
Guro-gu	3
Geumcheon-gu	2
Nowon-gu	3
Dobong-gu	2
Dongdaemun-gu	5
Dongjak-gu	3
Mapo-gu	1
Seodaemun-gu	2
Seocho-gu	3
Seongdong-gu	3
Seongbuk-gu	2
Songpa-gu	3
Yangcheon-gu	4
Yeongdeungpo-gu	5
Yongsan-gu	2
Eunpyeong-gu	5
Jongno-gu	4
Jung-gu	3
Jungnang-gu	3



Jenks Natural Breaks

GIS 분야에서 많이 사용하는 데이터의 구간을 나누는 방법.

데이터의 구간 수를 지정하면 값이 비슷한 항목끼리 그룹을 만드는 알고리즘으로 “시각화”라는 기준에서 봤을 때 사람의 눈이 가장 “자연스럽다”라고 느끼는 것에 초점을 두는 분류 방법이다.

1차원 데이터에 대한 k-means clustering 알고리즘과 같은 방식으로 작동한다.

1. 그룹의 갯수와 각 그룹의 중심값을 임의로 정함
2. 각 데이터 별로, 가장 가까운 중심값을 찾아 그룹을 할당
3. 할당된 결과를 가지고 그룹의 중심점을 재계산
4. 2~3을 반복

img

img

```
bins = jenks spy.jenks_breaks(df['의료기관수'], n_classes=5)
bins
```

```
[np.int64(1), np.int64(1), np.int64(2), np.int64(3), np.int64(4),
np.int64(5)]
```

#03. 지도 이미지 처리



1. 지도 불러오기

```
map_file_path = "svg/map_seoul.svg"

try:
    with open(map_file_path, 'r', encoding="utf-8") as f:
        map_svg = f.read()
    print("파일 읽어오기 완료")
except Exception as e:
    print("파일 읽어오기 에러:", e)

SVG(map_svg)
```

```
파일 읽어오기 완료
```



svg



2. 단계별 색상 팔레트 만들기

```
colors = ['#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77', '#980043']  
colors
```

```
[ '#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77', '#980043' ]
```



3. 지도에서 구 단위 추

```
soup = BeautifulSoup(map_svg)  
path_list = soup.select('path[id]')  
path_list[0]
```

```
/var/folders/wk/8tx_v8l94cqwt2b6dzgdc2h0000gn/T/
ipykernel_10929/342595318.py:1: XMLParsedAsHTMLWarning: It looks like
you're using an HTML parser to parse an XML document.
```

Assuming this really is an XML document, what you're doing might work, but you should know that using an XML parser will be more reliable. To parse this document as XML, make sure you have the Python package 'lxml' installed, and pass the keyword argument `features="xml"` into the BeautifulSoup constructor.

If you want or need to use an HTML parser on this document, you can make this warning go away by filtering it. To do that, run this code before calling the BeautifulSoup constructor:

```
from bs4 import XMLParsedAsHTMLWarning
import warnings

warnings.filterwarnings("ignore",
category=XMLParsedAsHTMLWarning)

soup = BeautifulSoup(map_svg)

<path clip-rule="evenodd" d="M964.064,164.667
c-1.447,9.018-0.285,18.105-2.002,27.506c-2.068,11.332-9.018,22.101-11.50
c-0.508,4.656-1.969,10.129-1.5,14.003c0.779,6.456,5.756,14.04,8.502,21.0
c0.539,4.856-0.953,11.628-1.502,17.504c-0.547,5.879-1.484,11.904-2,17.50
c-1.582,7.641-5.57,14.402-7.002,21.505c-1.725,8.558-1.271,18.438-3,27.50
c-14.793-19.111-31.705-39.509-48.51-58.013c-4.902-5.398-11.217-16.078-17
c-4.459-4.876-9.127-9.544-14.002-14.003c-0.148-1.02-1.354-0.98-1.502-2c-
c-2.484-9.723,2.434-16.186,3.5-24.005c1.156-1.678,0.176-5.493,0.5-8.001c
c-1.914-13.504,2.932-25.383,2.502-37.009c-0.459-12.384-5.236-23.798-6.00
c0.838-8.333,5.449-13.907,6.502-19.504c9.998-4.506,22.598-6.408,38.008-5
```

```
c3.451,0.612,7.951-0.803,10.502,0c9.178,2.887,3.551,20.857,10.002,25.005
```

```
c7.441,0.328,14.299,0.634,21.004,1C944.035,158.024,948.826,166.568,964.0  
fill="#C8C8C8" fill-rule="evenodd" id="Dobong-gu">></path>
```



4. 지도에서 확인된 지역명 만큼 반

```
for p in path_list:  
    지역명 = p['id']  
    #print(지역명)  
  
    의료기관수 = df.loc[지역명, '의료기관수']  
    # Dobong-gu → 2  
    #print(지역명, "-->", 의료기관수)  
  
    # 미리 생성한 구간만큼 반복  
    # [np.int64(1), np.int64(1), np.int64(2), np.int64(3),  
    # np.int64(4), np.int64(5)]  
    for i, v in enumerate(bins):  
        if i == 0:  
            continue  
        elif i + 1 < len(bins):  
            if 의료기관수 < v:  
                color_index = i - 1;  
                break  
        else:  
            if 의료기관수 ≤ v:  
                color_index = i - 1;  
                break  
                break  
  
    print(지역명, 의료기관수, color_index)  
    p['fill'] = colors[color_index]
```

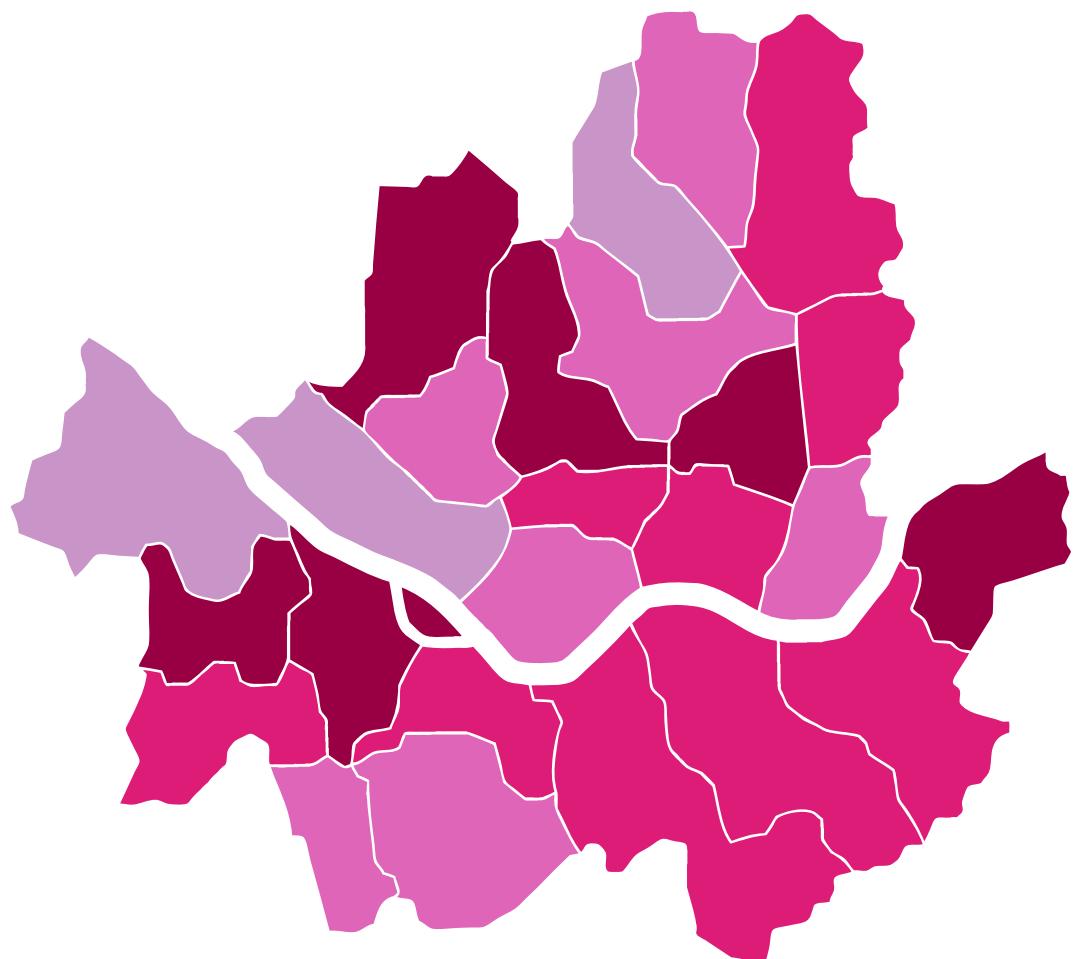
```
Dobong-gu 2 2  
Dongdaemun-gu 5 4  
Dongjak-gu 3 3  
Eunpyeong-gu 5 4  
Gangbuk-gu 1 1  
Gangdong-gu 4 4  
Gangseo-gu 1 1  
Geumcheon-gu 2 2  
Guro-gu 3 3
```

```
Gwanak-gu 2 2
Gwangjin-gu 2 2
Gangnam-gu 3 3
Jongno-gu 4 4
Jung-gu 3 3
Jungnang-gu 3 3
Mapo-gu 1 1
Nowon-gu 3 3
Seocho-gu 3 3
Seodaemun-gu 2 2
Seongbuk-gu 2 2
Seongdong-gu 3 3
Songpa-gu 3 3
Yangcheon-gu 4 4
Yeongdeungpo-gu 5 4
Yongsan-gu 2 2
```

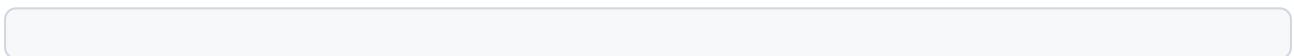
4. 재구성된 내용을 토대로 새로운 svg 생성

```
# bs4 객체의 내용을 문자열로 리턴
new_seoul_svg = soup.prettify()

# jupyter에서 svg 이미지 표시하기
# 사용방법 -> SVG(소스문자열) 혹은 SVG(파일경로)
SVG(new_seoul_svg)
```



svg





[LAB-07] 2. SVG 지도 시각화 (서울)



SVG(Scalable Vector Graphics)

JPEG, PNG와 같은 그래픽 포맷(Graphic format)의 하나

벡터 기반이기 때문에 리사이징이 되어도 전혀 깨지지 않고 모든 해상도에서 자유자재로 활용할 수 있다.

SVG파일 포맷은 XML로 구성되어 있기 때문에 BeautifulSoup 패키지를 활용하여 HTML 파싱과 같은 구현과정을 통해 원하는 부분을 취득, 변형 할 수 있다.



#01. 준비작업



1. 패키지 참조

bs4 패키지가 필요하다.

| 웹 페이지 데이터 수집 단원을 통해 이미 설치되어 있음

```
from hossam import load_data

# jupyter 상에서 SVG 이미지를 표시하기 위한 패키지(jupyter 기본 내장 패키지)
from IPython.display import SVG

# TAG로부터 원하는 내용을 추출하는 클래스 -> SVG 이미지의 핸들링을 위함
from bs4 import BeautifulSoup

from pandas import DataFrame
import numpy as np
import os
```



2. 데이터 가져오기

```
origin = load_data("senior_lsf")
origin
```

```
[94m[data] [0m https://data.hossam.kr/data/lab06/senior_lsf.xlsx  
[94m[desc] [0m 서울시의 행정구역별 노인복지시설의 수를 조사한 가상의 데이터  
[91m[!] Cannot read metadata [0m
```

	지역명	복지시설
0	Jongno-gu	61
1	Jung-gu	53
2	Yongsan-gu	110
3	Seongdong-gu	155
4	Gwangjin-gu	103
5	Dongdaemun-gu	146
6	Jungnang-gu	128
7	Seongbuk-gu	158
8	Gangbuk-gu	111
9	Dobong-gu	139
10	Nowon-gu	252
11	Eunpyeong-gu	154
12	Seodaemun-gu	103
13	Mapo-gu	160
14	Yangcheon-gu	192
15	Gangseo-gu	215
16	Guro-gu	192
17	Geumcheon-gu	75
18	Yeongdeungpo-gu	208
19	Dongjak-gu	143
20	Gwanak-gu	127
21	Seocho-gu	129
22	Gangnam-gu	184
23	Songpa-gu	173
24	Gangdong-gu	140

#02. 지도 이미지 가져오기

위키미디어에서 Seoul districts.svg 키워드로 검색하여 서울 지도 이미지를 내려받아 map_seoul.svg라는 이름으로 작업 폴더에 추가하고 open() 함수로 파일을 읽어올 수 있다.

https://commons.wikimedia.org/wiki/File:Seoul_districts.svg?uselang=ko

혹은 아래의 URL에 접속한 후 Ctrl+S를 눌러서 파일을 저장한다.

https://data.hossam.kr/data/lab07/map_seoul.svg

저장된 파일은 /작업폴더/svg/map_seoul.svg 경로에 저장한다.



1. 지도 이미지 읽어오기

```
map_file_path = "svg/map_seoul.svg"

try:
    with open(map_file_path, 'r', encoding="utf-8") as f:
        map_svg = f.read()
    print("파일 읽어오기 완료")

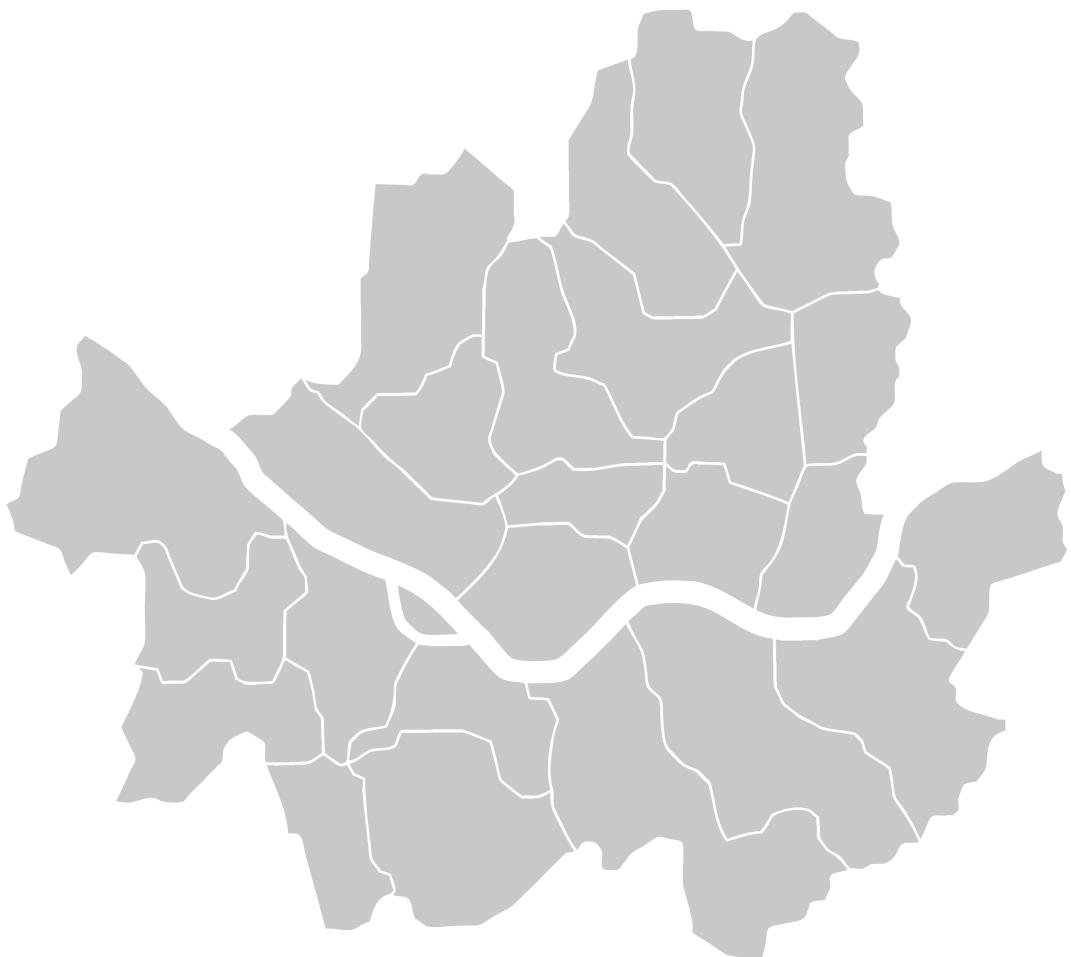
    # 출력되는 내용이 많으므로 내용 확인 후 주석으로 막아두자.
    #print(map_svg)
except Exception as e:
    print("파일 읽어오기 에러:", e)
```

파일 읽어오기 완료



2. 이미지 확인

```
SVG(map_svg)
```



svg

#03. 데이터 시각화

1. 단계별 색상 팔레트 만들기

단계는 분석가가 임의로 정한다.

색상값을 1단계 ~ 높은단계 순으로 점점 진한 색상이 되도록 구성

사용할 색상값 (단계별로 6개 색상 준비)

```
colors = ['#F1EEF6', '#D4B9DA', '#C994C7', '#DF65B9', '#DD1C77',  
         '#980043']
```



2. BeautifulSoup 객체 생성

svg파일의 내용을 BeautifulSoup객체로 변환

```
soup = BeautifulSoup(map_svg, features="xml")
# 출력되는 내용이 많으므로 내용 확인 후 주석으로 막아두자.
#soup
```



[3] 구 단위로 추출

id속성을 갖는 path 태그 가져오기

```
# soup.select() 메서드의 리턴값은 항상 리스트이다.
# → path 태그이면서 id속성을 갖는 요소를 리스트로 반환
#   <path id="???" ... >
path_list = soup.select('path[id]')
print("가져온 도형의 수: ", len(path_list))
```

가져온 도형의 수: 25



[3] 지도에서 확인한 지역명 수 만큼 반복

```
for p in path_list:
    #print(p)

    지역명 = p['id']
    #print(지역명)

    복지시설수 = origin.query('지역명 == @지역명')['복지시설'].values[0]
    print(지역명, " → ", 복지시설수)

    # 복지시설 수에 따라 단계값 설정 (단계는 색상값의 수에 따른)
    if 복지시설수 > 250:    color_index = 5
    elif 복지시설수 > 200:  color_index = 4
    elif 복지시설수 > 150:  color_index = 3
    elif 복지시설수 > 100:  color_index = 2
    elif 복지시설수 > 50:   color_index = 1
    else:                  color_index = 0

    # svg 이미지의 면 색상 변경
    p['fill'] = colors[color_index]
```

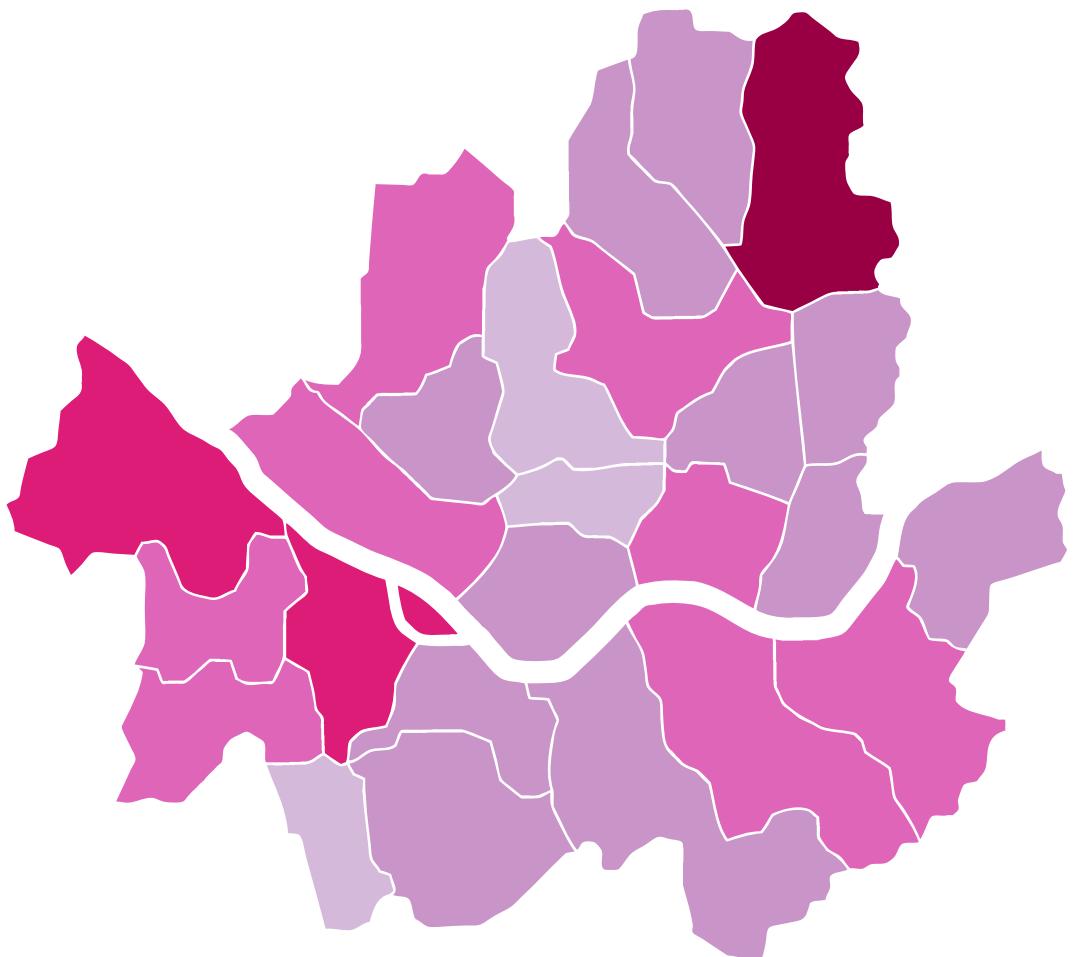
Dobong-gu → 139
Dongdaemun-gu → 146
Dongjak-gu → 143
Eunpyeong-gu → 154
Gangbuk-gu → 111
Gangdong-gu → 140
Gangseo-gu → 215
Geumcheon-gu → 75
Guro-gu → 192
Gwanak-gu → 127
Gwangjin-gu → 103
Gangnam-gu → 184
Jongno-gu → 61
Jung-gu → 53
Jungnang-gu → 128
Mapo-gu → 160
Nowon-gu → 252
Seocho-gu → 129
Seodaemun-gu → 103
Seongbuk-gu → 158
Seongdong-gu → 155
Songpa-gu → 173
Yangcheon-gu → 192
Yeongdeungpo-gu → 208
Yongsan-gu → 110



[4] 재구성된 내용을 토대로 새로운 svg 소스코드 얻기

```
# bs4 객체의 내용을 문자열로 리턴
new_seoul_svg = soup.prettify()

# jupyter에서 svg 이미지 표시하기
# 사용방법 -> SVG(소스문자열) 혹은 SVG(파일경로)
SVG(new_seoul_svg)
```



svg



[5] 생성된 이미지를 파일로 저장해야 하는 경우

```
# 저장된 파일은 윈도우 폴더창에서 직접 더블클릭 해서 웹 브라우저를 통해 확인해야 한다.  
with open('svg/new_seoul_svg.svg', 'w', encoding="utf-8") as f:  
    f.write(new_seoul_svg)
```

QGIS 활용 | 공간 데이터 분석

[LAB-10] GeoCoding

(강남구 지진 해일 대피소 위치 / 서울시 응급실 분포)





학습안내

이번 시간에 학습할 내용과 목표입니다.

학습 내용

1. 구글 스프레드시트 활용
2. 브이월드 Open API 활용

학습 목표

1. 지오코딩에 대해 이해하고 설명할 수 있다.
2. 구글 스프레드시트를 활용하여 지오코딩을 수행할 수 있다.
3. 브이월드 Open API를 활용하여 지오코딩을 수행할 수 있다.

1. 구글 스프레드시트 활용



Geo Coding (지오코딩) 개요

1. 구글스프레드시트 활용

2. 브이월드 API 활용

□ 고유명칭(주소나 산, 호수의 이름 등)을 가지고 위도와 경도의 좌표값을 얻는 것

- ◎ 쉐이프파일 등의 GIS 자료를 제공해 주는 기관의 경우 자료를 바로 QGIS 등의 GIS 플랫폼에서 열람하고 분석할 수 있지만, 대부분의 기관이나 업체는 데이터를 GIS 형태가 아닌 주소나 좌표 형태로 제공한다.
- ◎ 주소만 알고 있는 경우 자연어로 된 주소를 좌표로 바꾸는 작업인 지오코딩(geocoding)을 거쳐 쉐이프 파일을 만든 후, 이를 QGIS에서 불러오는 형식으로 분석을 진행할 수 있다.

□ 수업에서 제공되는 “강남구 지진해일대피소.xlsx” 파일의 예시

A	B	C	D	E	F	G	
1	지진해일대피소명	지진해일대피소구분	지진해일대피소유형	지진해일대피소유형구분	소재지도로명주소	수용가능면적	최대수용인원수
2	개원초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 29	12032	3646
3	구룡중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 103	7038	2133
4	개일초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 401	6600	2000
5	양전초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 513	7515	2277
6	개원중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 영동대로 101	10260	3109
7	개포고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 402	8882	2692
8	경기여자고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 삼성로 29	8292	2513
9	수도전기공업고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 410	20800	6303



구글 스프레드시트를 활용한 지오코딩 (1)

□ 구글 스프레드시트에 접속하여 빈 스프레드시트를 생성한다.

◎ <https://docs.google.com/spreadsheets/>

The screenshot shows the Google Sheets interface. At the top, there's a toolbar with various icons. Below it is a navigation bar with a back arrow, forward arrow, refresh icon, and a URL field containing "https://docs.google.com/spreadsheets/u/0/". To the right of the URL are search, star, and other account-related icons. The main area is titled "스프레드시트" (Spreadsheet) and features a search bar. On the left, there's a sidebar with the title "새 스프레드시트 시작하기" (Start a new spreadsheet). It displays several template cards. One card, "빈 스프레드시트" (Blank Spreadsheet), is highlighted with a red dashed border. Other cards include "캔트 차트 템플릿" (Canva Chart Template) by Smartsheet, "할 일 목록" (To-do list), "연간 가계부" (Annual household budget), and "월간 가계부" (Monthly household budget). At the bottom, there are buttons for "이전 7일" (Last 7 days), "모든 항목" (All items), "내가 마지막으로 열어본 항목" (Items I last opened), and sorting/filtering options.



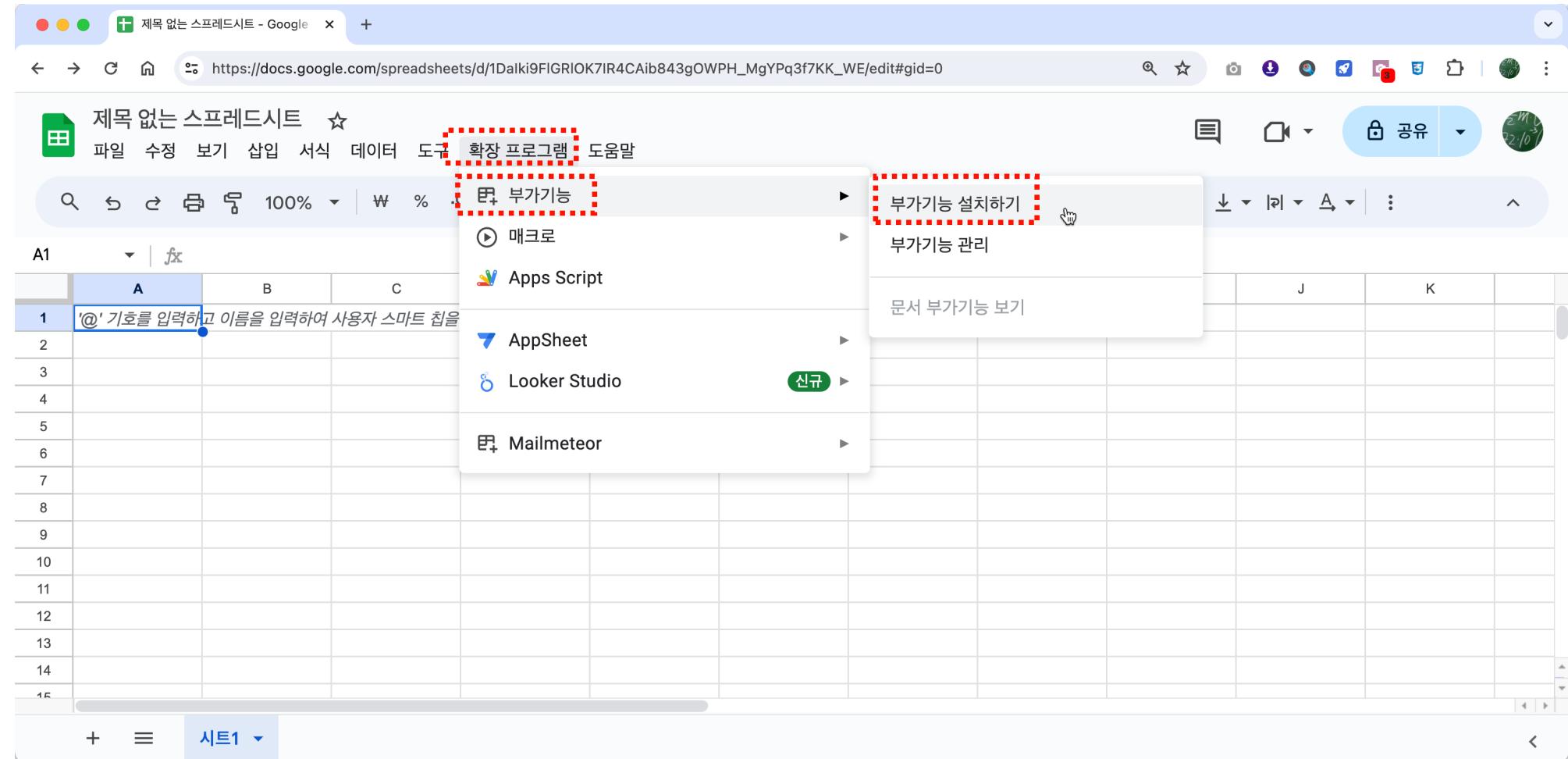
구글 스프레드시트를 활용한 지오코딩 (2)

□ 부가기능 설치하기

◎ “확장 프로그램 → 부가기능 → 부가기능 설치하기” 메뉴를 선택한다.

1. 구글스프레드시트활용

2. 브이월드 API 활용





구글 스프레드시트를 활용한 지오코딩 (3)

□ “Geocode by Awesome Table” 검색

◎ 검색 결과 중에서 설치하고자 하는 추가기능을 클릭한다.

1. 구글스프레드시트활용

2. 브이월드 API 활용

The screenshot shows the Google Sheets Marketplace interface. A search bar at the top right contains the text "Geocode by Awesome Table". A red box with the number "1" highlights this search bar. Below the search bar, a list of add-ons is displayed. The first item in the list, "GeoCode by Awesome Table" by Talarian, is highlighted with a red box and the number "2". This item has a blue icon featuring a map and a location pin. The description below the icon reads: "Geocode is a tool that helps you get latitudes & longitudes from addresses in a Google Sheet t...". The rating is 3.9 stars with 121 reviews. To the right of this are three other add-ons: "Awesome Table" (rating 4.5), "Geocode & Mapping Sheets" (rating 4.8), and "Sheets Mapper" (rating 4.9). The background shows a portion of a Google Sheets document with columns A and K visible.



구글 스프레드시트를 활용한 지오코딩 (4)

□ 프로그램 설치하기

- ◎ “설치” 버튼을 클릭하여 부가기능을 설치한다.
- ◎ 설치 과정 중에서 구글 로그인이 요구될 수 있다.

제목 없는 스프레드시트 - Google

https://docs.google.com/spreadsheets/d/1Dalki9FIGRIOK7IR4CAib843gOWPH_MgYPq3f7KK_WE/edit#gid=0

파일 수첩 보기 산인 서식 데이터 도구 화자 프로그램 도움말

Google Workspace Marketplace

Geocode by Aweso...

Geocode is a tool that helps you get latitudes & longitudes from addresses in a Google Sheet to display them on a map you can share.

개발자: Talarian

정보 업데이트: 2024년 4월 1일

호환 기기: 422 121만+

설치



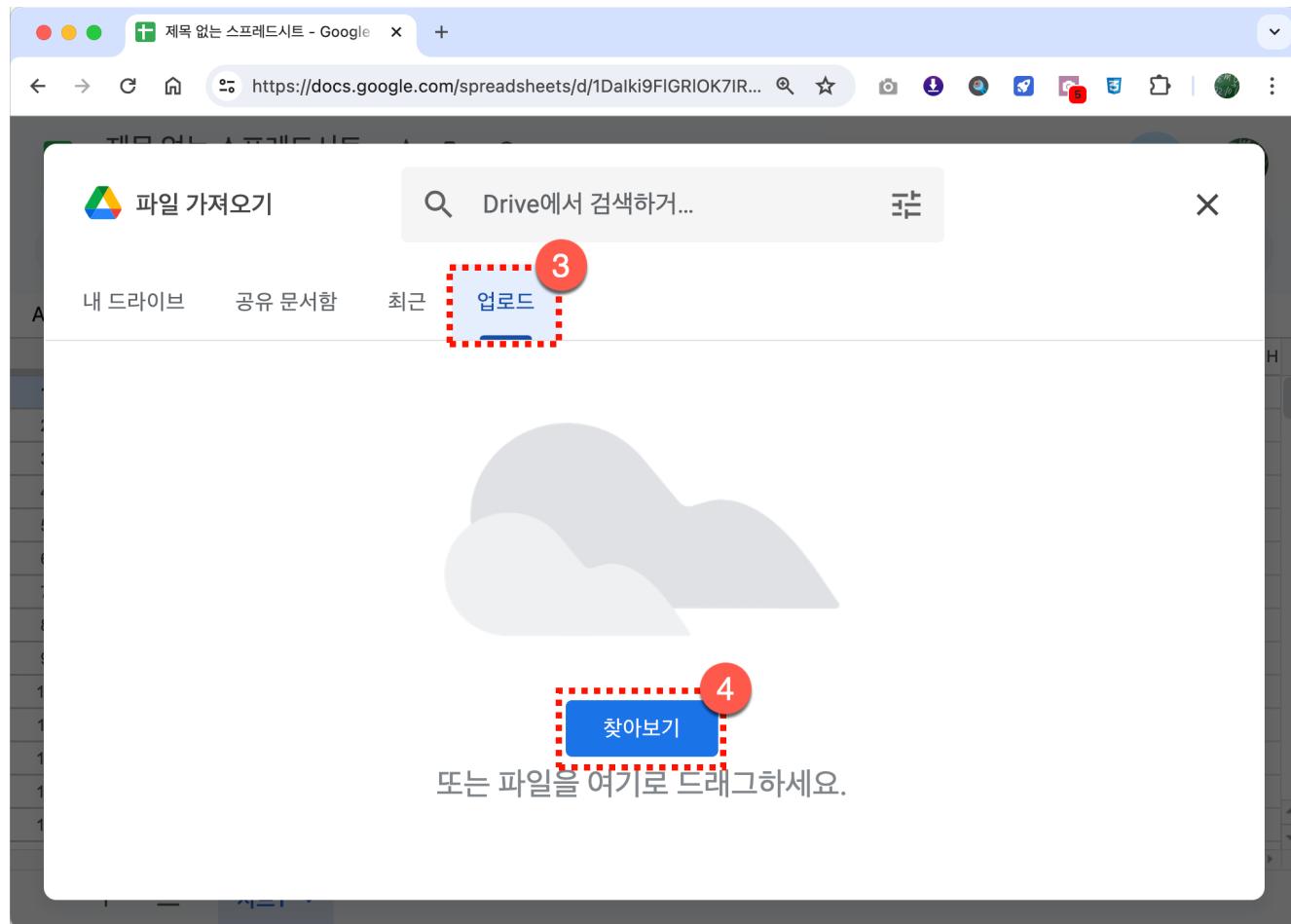
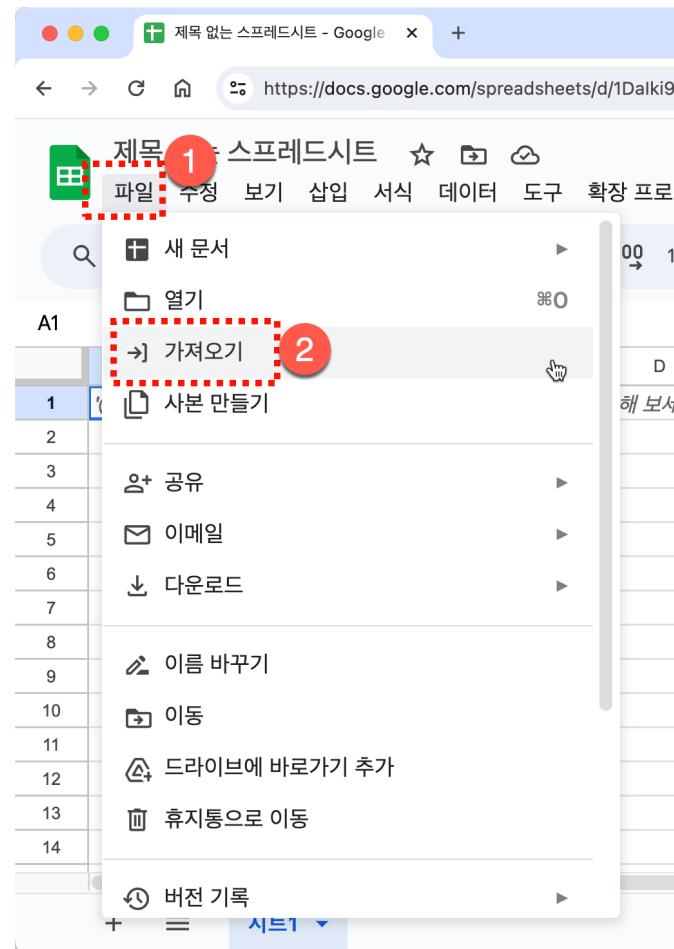
구글 스프레드시트를 활용한 지오코딩 (5)

□ 데이터 파일(엑셀) 업로드하기

◎ “강남구 지진해일대피소.xlsx” 파일을 업로드 한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용



또는 파일을 여기로 드래그하세요.



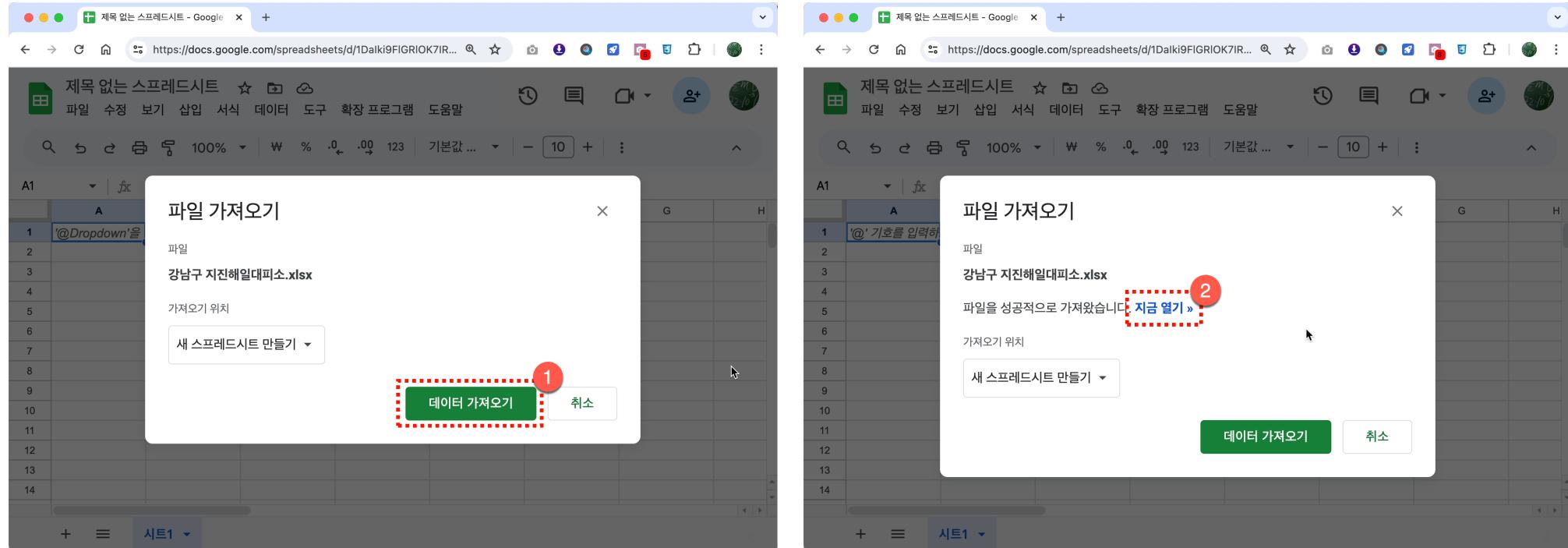
구글 스프레드시트를 활용한 지오코딩 (6)

□ 데이터 가져오기

◎ 업로드가 완료되면 “데이터 가져오기” 버튼을 클릭하여 파일을 연다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용





구글 스프레드시트를 활용한 지오코딩 (7)

□ 지오코딩 실행하기

- ◎ “확장 프로그램 → Geocode by Awesome Table → Start Geocoding” 메뉴를 실행하여 설치한 확장 프로그램을 시작한다.

The screenshot shows a Google Sheets document titled "강남구 지진해일대피소". A context menu is open over a table cell containing "지진대피소명". The menu items include "부가기능", "매크로", "Apps Script", "AppSheet", "Looker Studio", and two entries under "신규": "Geocode by Awesome Table" and "Mailmeteor". The "Geocode by Awesome Table" item is highlighted with a red dashed box. A second red dashed box highlights the "Start Geocoding" option within the same submenu. The main sheet contains a table with columns for location names and addresses, and the right pane shows another table with columns for address, usage count, and maximum users.

	E	F	G	H
1	소재지도로명주소	수용가능면적	최대수용인원수	
2	서울특별시 강남구 선릉로 29	12032	3646	
3	서울특별시 강남구 선릉로 103	7038	2133	
4	서울특별시 강남구 개포로 401	6600	2000	
5		7515	2277	
6		10260	3109	
7		8882	2692	
8		8292	2513	
9		20800	6303	
10		12874	3901	
11		12279	3721	
12		1392	422	
13		4640	1406	
14		6136	1859	
15		6444	1953	
16		13435	4071	



구글 스프레드시트를 활용한 지오코딩 (8)

□ 스크롤을 아래로 내려 파라미터 설정 구간으로 이동한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

The screenshot shows a Google Sheets document titled "강남구 지진해일대피소". The spreadsheet contains a table with 16 rows of data, each listing a location name, its category, type, specific category, and address. The first row is selected. A sidebar on the right is titled "Geocode" and displays promotional text for "Awesome Table Connectors". A red dashed arrow points downwards from the top of the sidebar towards the bottom of the page, indicating where to scroll. A yellow box highlights the text "구글 계정 하나당 하루 10000건의 처리 제한이 있음".

	A	B	C	D	E
1	지진해일대피소명	지진해일대피소구분	지진해일대피소유형	지진해일대피소유형구분	소재지도로명주소
2	개원초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 29
3	구룡중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 103
4	개일초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 401
5	양전초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 513
6	개원중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 역동대로 101
7	개포고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 103
8	경기여자고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 삼성로 29
9	수도전기공업고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 410
10	구룡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 263
11	포이초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 22길 87
12	국립국악고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 22길 65
13	논현초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 강남대로 120길
14	학동초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 115길 42
15	언복중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 도산대로 38길 2
16	대치초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 양재천로 363

Geocode gets latitudes and longitudes from

Try now for FREE

스크롤을 아래로 내린다.

구글 계정 하나당 하루 10000건의 처리 제한이 있음

- 10k addresses per day limit for Google Workspace accounts
- Schedule automatic refreshes
- Import data from many sources, such as Salesforce, Quickbooks, Hubspot etc



구글 스프레드시트를 활용한 지오코딩 (9)

□ 파라미터 설정하기

◎ 시트 이름과 주소가 기입된 필드를 지정한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

	A	B	C	D	E
1	지진해일대피소명	지진해일대피소구분	지진해일대피소유형	지진해일대피소유형구분	소재지도로명주소
2	개원초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 29
3	구룡중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 103
4	개일초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 401
5	양전초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 513
6	개원중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 영동대로 101
7	개포고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 402
8	경기여자고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 삼성로 29
9	수도전기공업고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 410
10	구룡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 263
11	포이초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로22길 87
12	국립국악고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로22길 65
13	논현초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 강남대로120길
14	학동초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로115길 42
15	연북중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 도산대로38길
16	대치초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 양재천로 363



구글 스프레드시트를 활용한 지오코딩 (10)

□ 지오코딩 실행 확인

◎ 경도(Latitude), 위도(Longitude) 필드가 자동으로 추가되면서 지오코딩이 수행된다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

The screenshot shows a Google Sheets document titled "강남구 지진해일대피소". A sidebar titled "Geocode" is open, displaying progress: "45 out of 74 addresses". The main table contains 16 rows of address data, with columns for address, latitude, longitude, usage status, and maximum users. The "Latitude" and "Longitude" columns are highlighted with red dashed boxes, indicating they were automatically added by the geocoding process.

유형구분	소재지도로명주소	Latitude	Longitude	수용가능면적	최대수용인원수
1	서울특별시 강남구 선릉로 29	37.4813655	127.0590553	12032	3646
2	서울특별시 강남구 선릉로 103	37.4858999	127.0564272	7038	2133
3	서울특별시 강남구 개포로 401	37.4859589	127.0580194	6600	2000
4	서울특별시 강남구 개포로 513	37.4904255	127.07003	7515	2277
5	서울특별시 강남구 영동대로 101	37.4913769	127.0714248	10260	3109
6	서울특별시 강남구 개포로 402	37.484898	127.0591661	8882	2692
7	서울특별시 강남구 삼성로 29	37.4867111	127.0659079	8292	2513
8	서울특별시 강남구 개포로 410	37.4858037	127.0634758	20800	6303
9	서울특별시 강남구 개포로 263	37.4807968	127.0519292	12874	3901
10	서울특별시 강남구 개포로22길 87	37.4755995	127.0523238	12279	3721
11	서울특별시 강남구 개포로22길 65	37.4761552	127.0513959	1392	422
12	서울특별시 강남구 강남대로120길 33	37.5082167	127.0263166	4640	1406
13	서울특별시 강남구 선릉로115길 42	37.5120662	127.0398114	6136	1859
14	서울특별시 강남구 도산대로38길 27	37.5191189	127.0332439	6444	1953
15	서울특별시 강남구 양재천로 363	37.4909106	127.0622675	13435	4071
16					



구글 스프레드시트를 활용한 지오코딩 (11)

□ 처리 결과 다운로드

◎ “파일 → 다운로드 → Microsoft Excel(.xlsx)” 메뉴를 클릭하여 파일을 내려받는다.

1. 구글스프레드시트활용

2. 브이월드 API 활용

The screenshot shows a Google Sheets document titled "강남구 지진해일대피소". A sidebar on the right is titled "Geocode" and contains a section for "Awesome Table Connectors" with a "Try now for FREE" button. The main sheet displays a table of data with columns F, G, H, I, J, and K. The first row is a header: F (Latitude), G (Longitude), H (수용가능면적), I (최대수용인원수). Below the table, a dropdown menu is open under the "File" tab, showing options like "새 문서", "열기", "가져오기", "사본 만들기", "공유", "이메일", "다운로드", "이름 바꾸기", "이동", "드라이브에 바로가기 추가", "휴지통으로 이동", "버전 기록", and "오프라인 사용 설정". The "다운로드" option is highlighted with a red box. A sub-menu for "Microsoft Excel (.xlsx)" is also highlighted with a red box.

F	G	H	I	J	K
Latitude	Longitude	수용가능면적	최대수용인원수		
37.4813655	127.0590553	12032	3646		
37.4858999	127.0564272	7038	2133		
37.4859589	127.0580194	6600	2000		
37.4904255	127.07003	7515	2277		
		10260	3109		
		8882	2692		
		8292	2513		
		20800	6303		
		12874	3901		
		12279	3721		
		1392	422		
		4640	1406		
		6136	1859		
		37.5191189	127.0332439	6444	1953
		37.4909106	127.0622675	13435	4071



구글 스프레드시트를 활용한 지오코딩 - 결과 확인

□ 내려 받은 엑셀 파일 확인

◎ 경, 위도 값이 추가되어 있음을 확인할 수 있다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

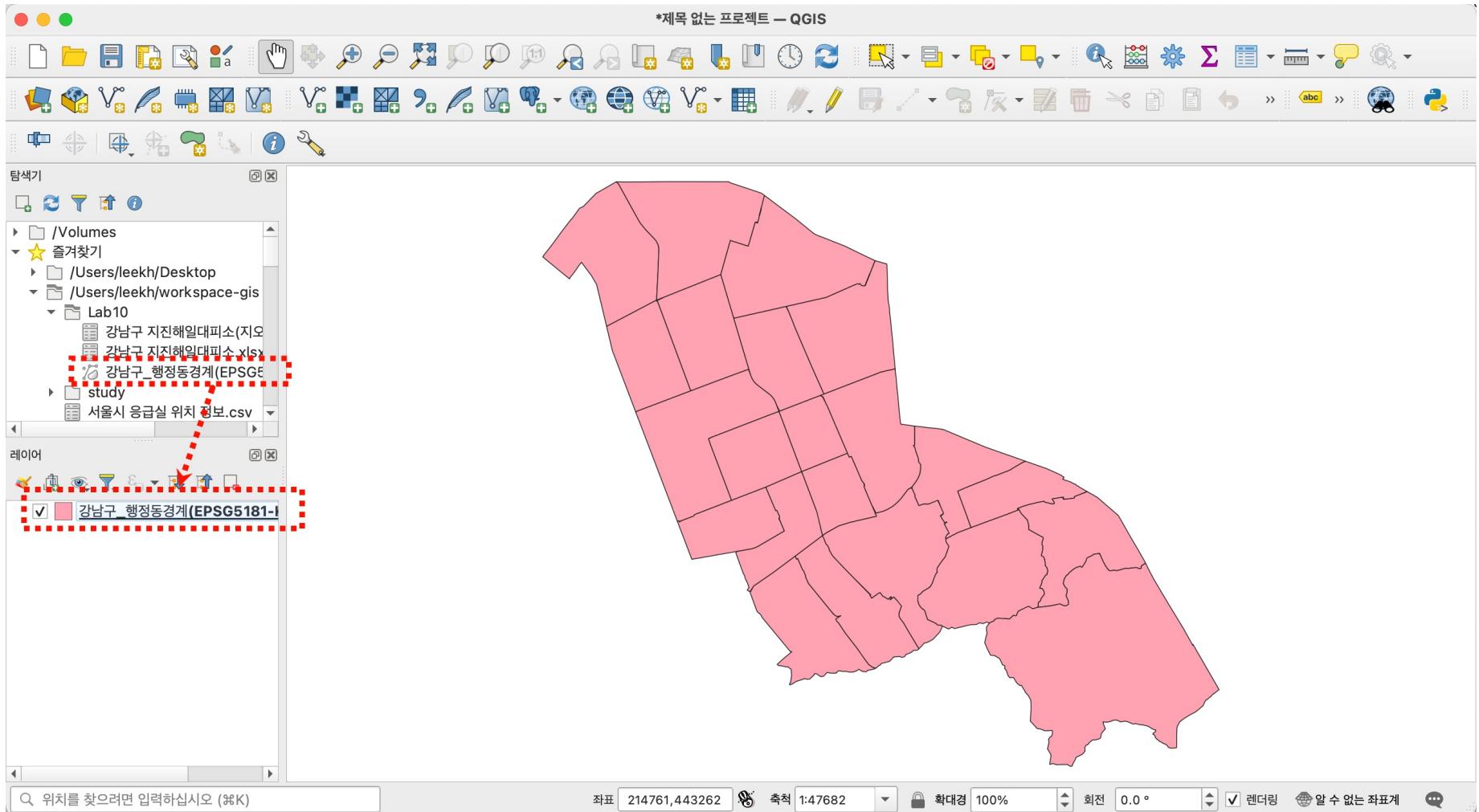
Google Sheets screenshot showing a table of data with columns A through I. Columns F (Latitude) and G (Longitude) are highlighted with red dashed boxes, indicating they were added to the original data. The table lists various locations, primarily sports fields at schools, with their addresses, types, and coordinates.

	A	B	C	D	E	F	G	H	I
1	지진해일대피소명	지진해일대피소구분	지진해일대피소유형	지진해일대피소유형구분	소재지도로명주소	Latitude	Longitude	수용가능면적	최대수용인원수
2	개원초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 29	37.4813655	127.0590553	12032	3646
3	구룡중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로 103	37.4858999	127.0564272	7038	2133
4	개일초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 401	37.4859589	127.0580194	6600	2000
5	양천초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 513	37.4904255	127.07003	7515	2277
6	개원중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 영동대로 101	37.4913769	127.0714248	10260	3109
7	개포고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 402	37.484898	127.0591661	8882	2692
8	경기여자고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 삼성로 29	37.4867111	127.0659079	8292	2513
9	수도전기공업고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 410	37.4858037	127.0634758	20800	6303
10	구룡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로 263	37.4807968	127.0519292	12874	3901
11	포이초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로22길 87	37.4755995	127.0523238	12279	3721
12	국립국악고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 개포로22길 65	37.4761552	127.0513959	1392	422
13	논현초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 강남대로120길 33	37.5082167	127.0263166	4640	1406
14	학동초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로115길 42	37.5120662	127.0398114	6136	1859
15	언북중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 도산대로38길 27	37.5191189	127.0332439	6444	1953
16	대치초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 양재천로 363	37.4909106	127.0622675	13435	4071
17	단대부속고등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 도곡로64길 21	37.496529	127.0564676	7800	2364
18	대청중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 양재천로 321	37.4901314	127.0585502	8336	2526
19	대곡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 남부순환로 3022	37.4944062	127.0651765	7805	2365
20	대현초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 역삼로98길 16	37.5035728	127.0638597	8585	2602
21	대명중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 역삼로87길 26	37.5060559	127.061066	5400	1636
22	휘문중학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 역삼로 541	37.505133	127.0621203	32455	9835
23	도곡초등학교 운동장	지진대피소	옥외대피소	운동장	서울특별시 강남구 선릉로64길 33	37.4990608	127.0546706	8080	2448



경,위도 좌표를 Point Layer로 추가하기 (1)

- 강남구 행정동 경계도를 작업 레이어에 추가
 - ◎ “강남구_행정동경계(EPSG5181-KGD2002,UTF8).shp” 파일을 추가한다.





경,위도 좌표를 Point Layer로 추가하기 (2)

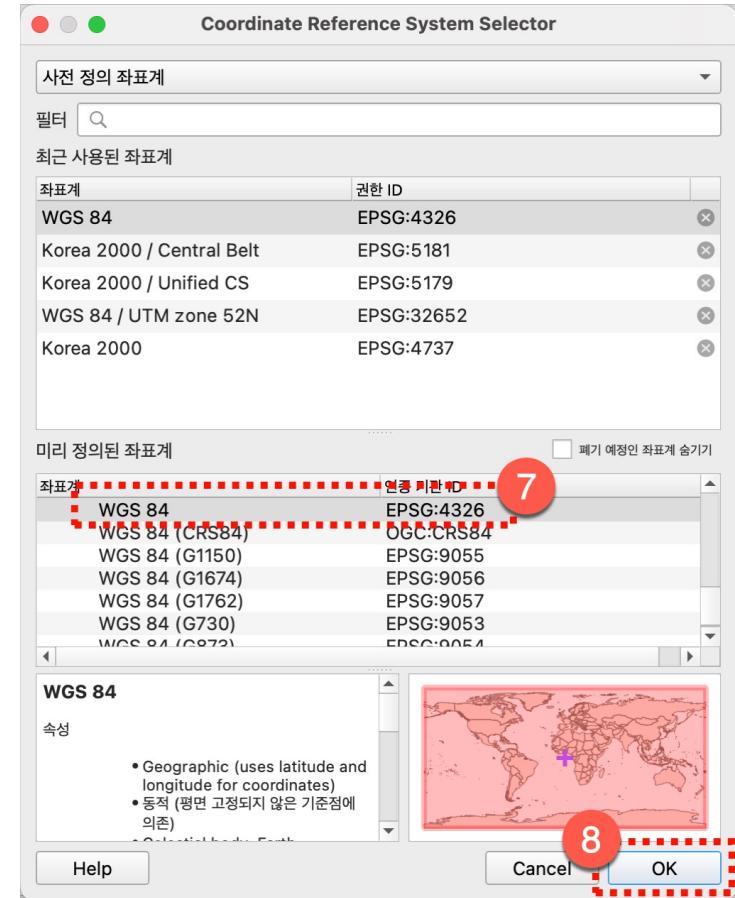
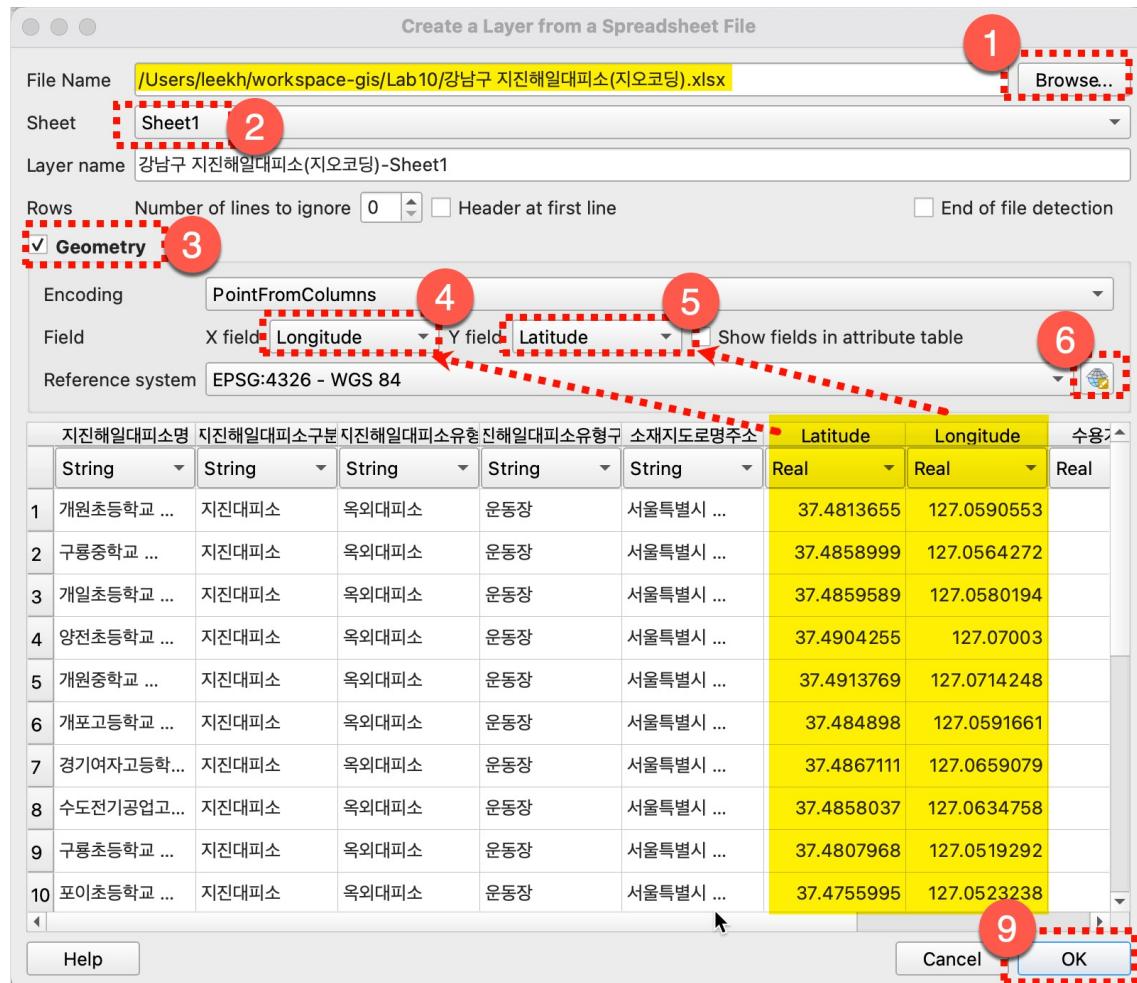
□ 지오코딩 결과가 포함된 엑셀 파일을 Point 레이어로 추가

◎ 툴바에서 “Add Spreadsheet Layer” 버튼을 클릭한다.



Add Spreadsheet Layer...

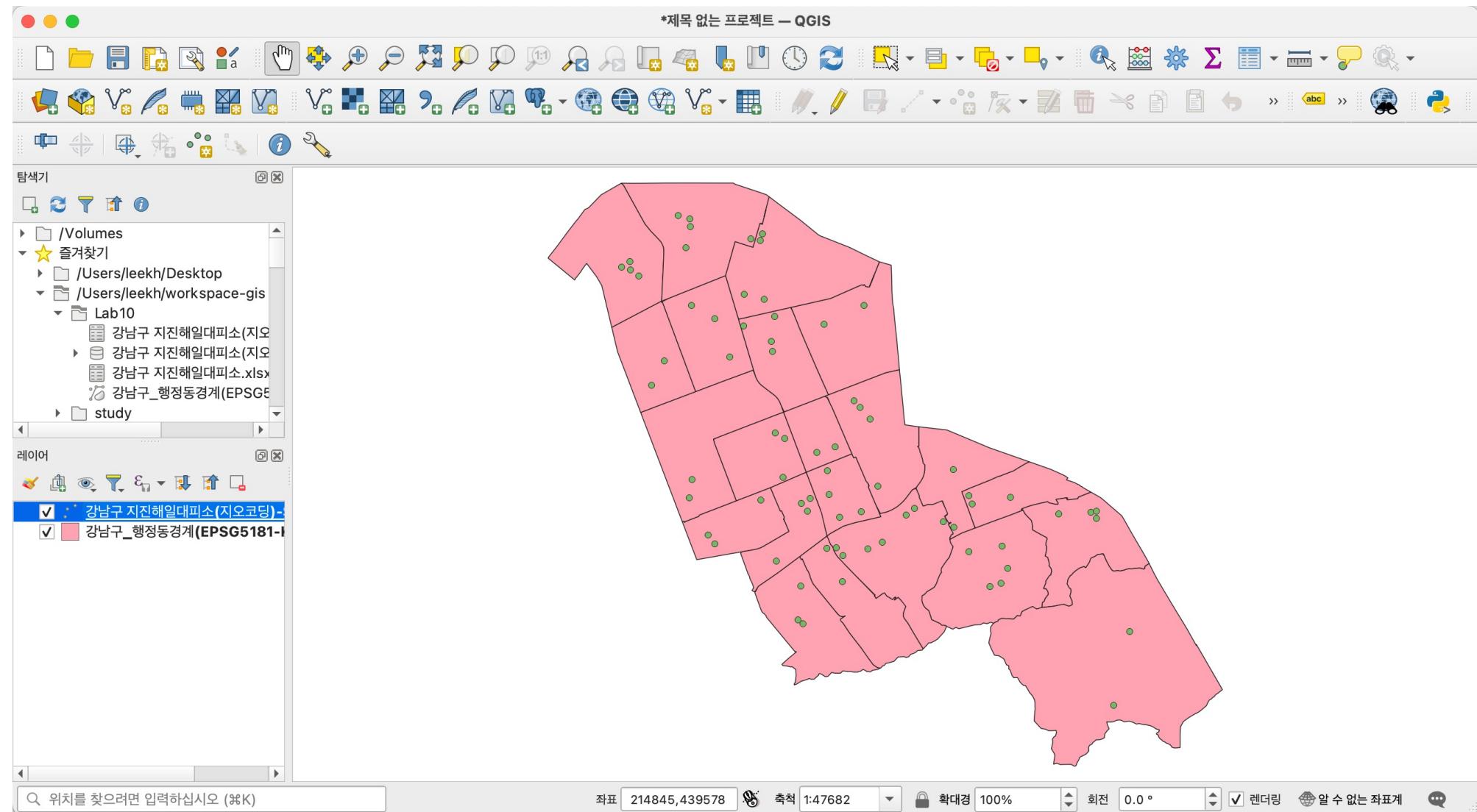
◎ 경위도 기반이므로 좌표계는 “EPSG 4326 - WGS84”를 지정해야 한다.





경,위도 좌표를 Point Layer로 추가하기 (3)

□ 지오코딩 처리된 좌표에 대한 포인트 확인

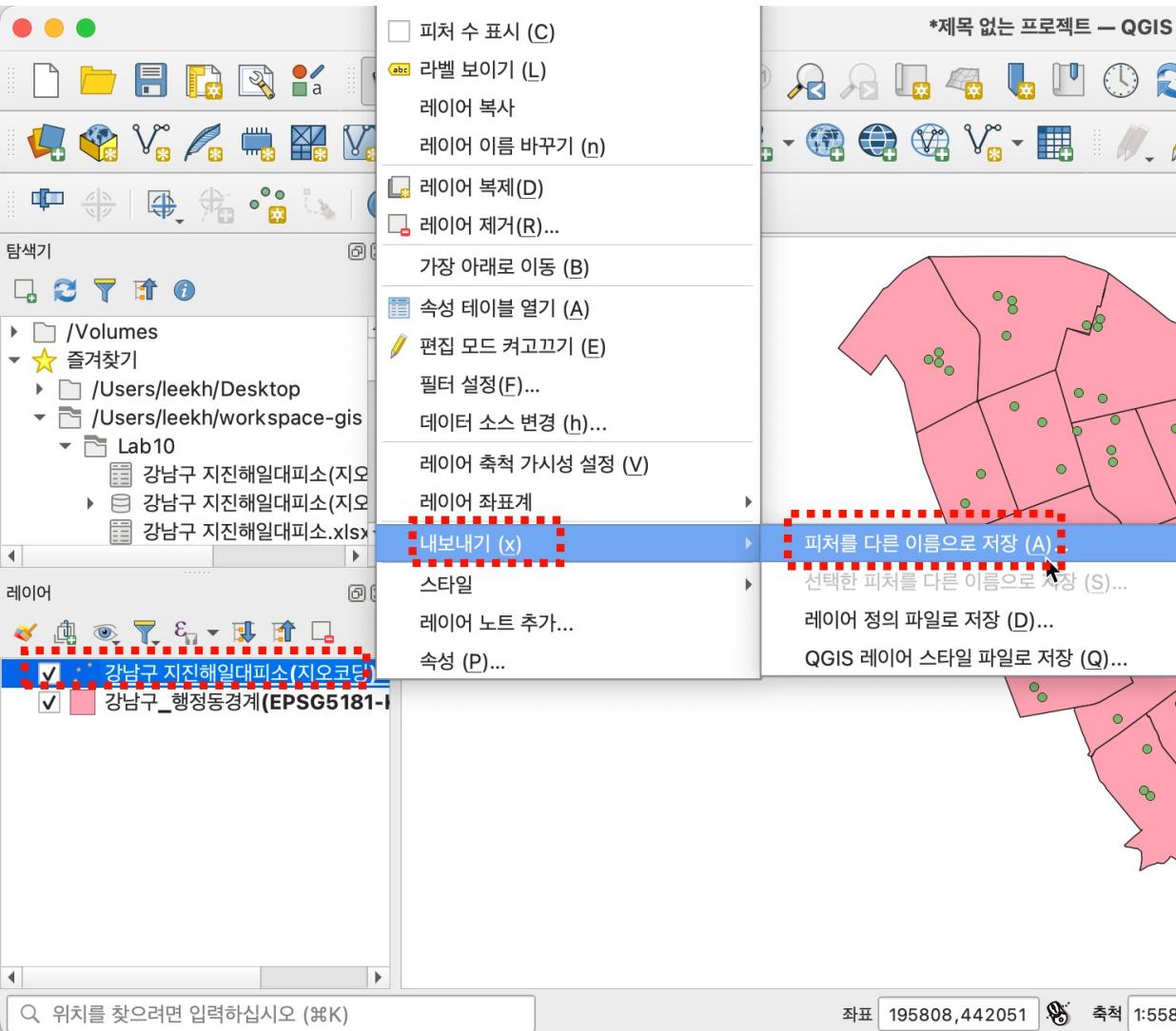




경,위도 좌표를 Point Layer로 추가하기 (4)

□ 포인트 레이어를 shape 파일로 저장하기

- ◎ 엑셀 파일을 Point Layer로 변경한 상태에서 shape 파일로 저장하면 작업시마다 변환해야 하는 번거로움을 피할 수 있다.



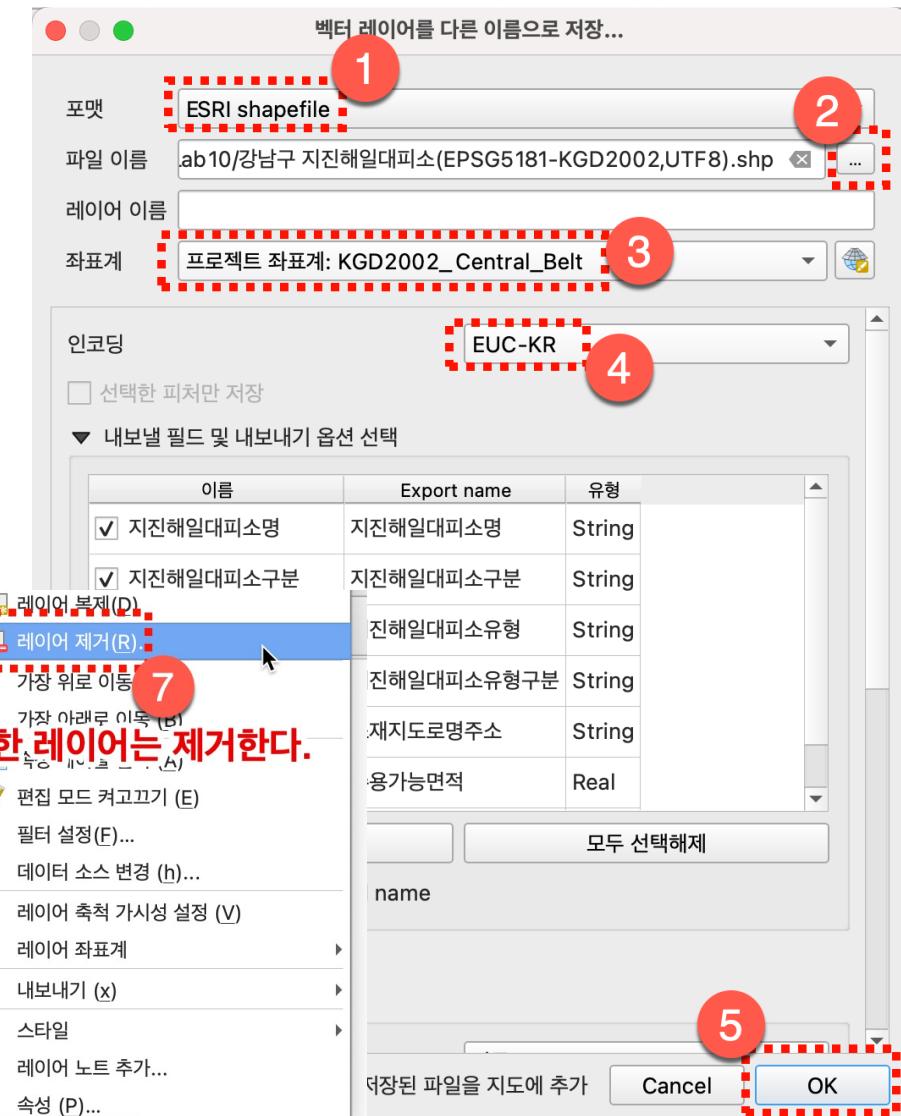
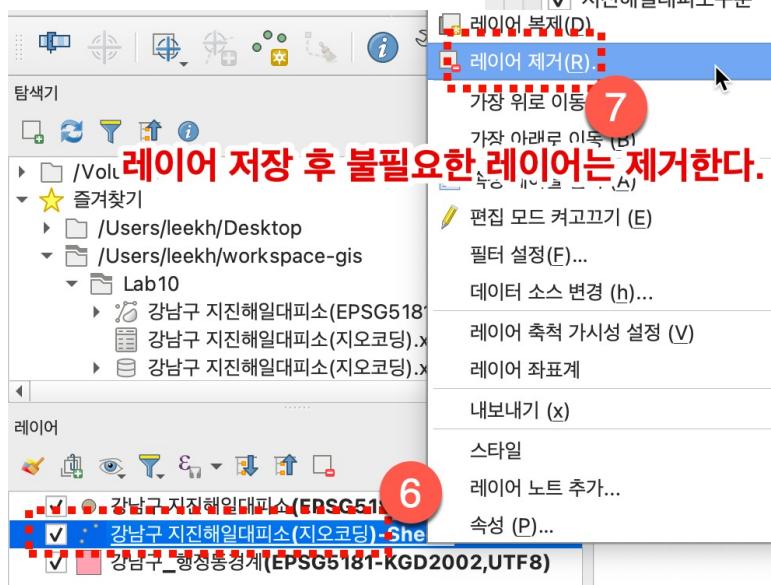


경,위도 좌표를 Point Layer로 추가하기 (5)

□ 저장 파라미터 설정

1. 구글스프레드시트 활용
2. 브이월드 API 활용

- ① 저장 포맷을 shape 파일로 설정
- ② 저장 경로 지정
- ③ 좌표계를 변경하고자 할 경우 설정
 - 여기서는 프로젝트 좌표계와 일치시킴
- ④ 한글 필드명이 있으므로 인코딩을 EUC-KR로 지정
 - UTF-8은 한글 필드명이 깨진다(QGIS 버그)
- ⑤ 설정 내용 실행





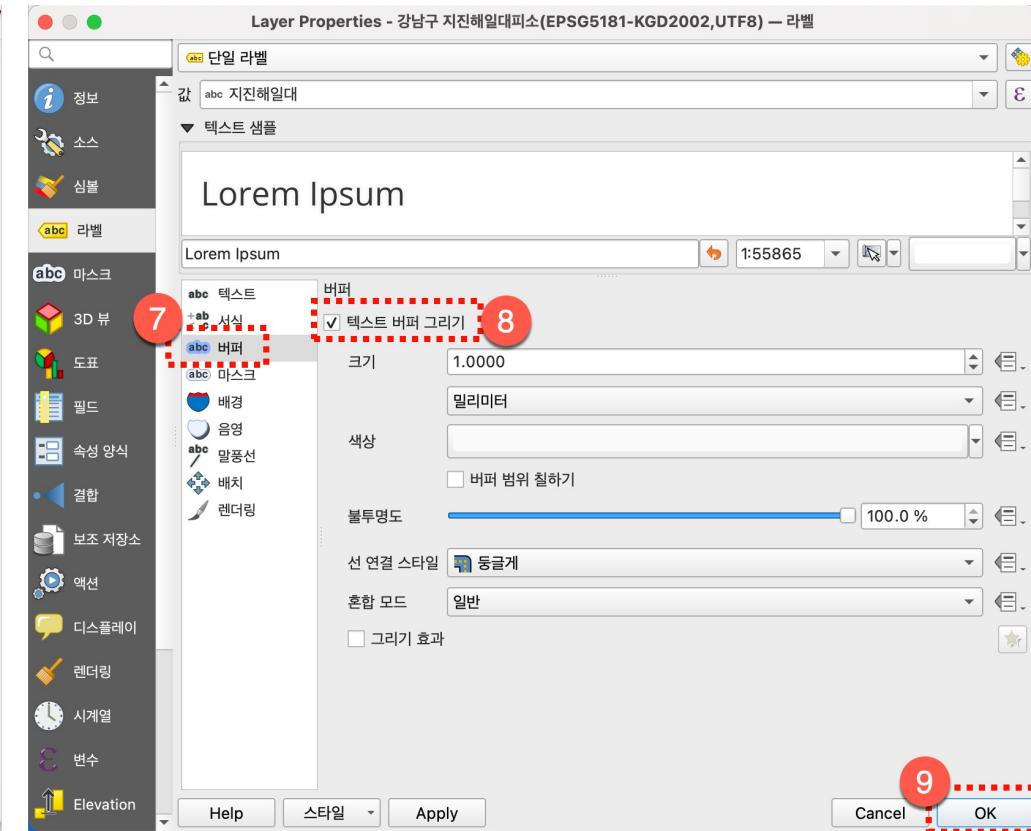
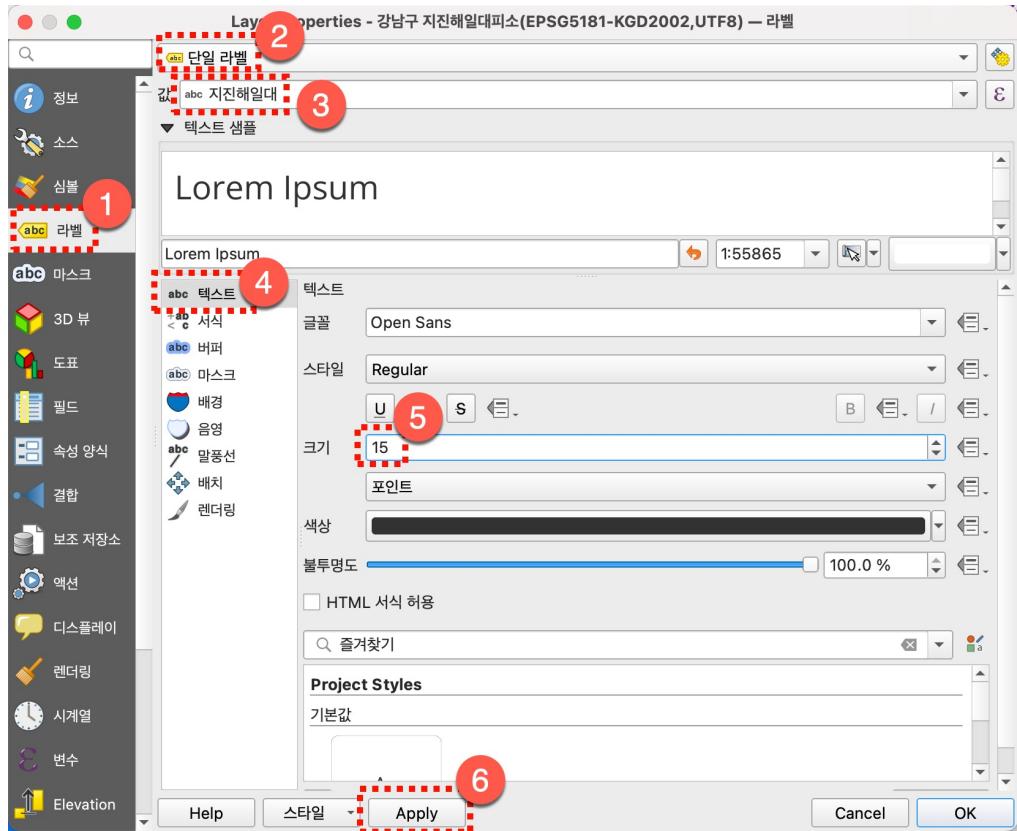
경,위도 좌표를 Point Layer로 추가하기 (6)

□ Point Layer의 라벨 속성 설정

◎ 대피소 명칭을 표시할 수 있도록 속성을 설정한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용



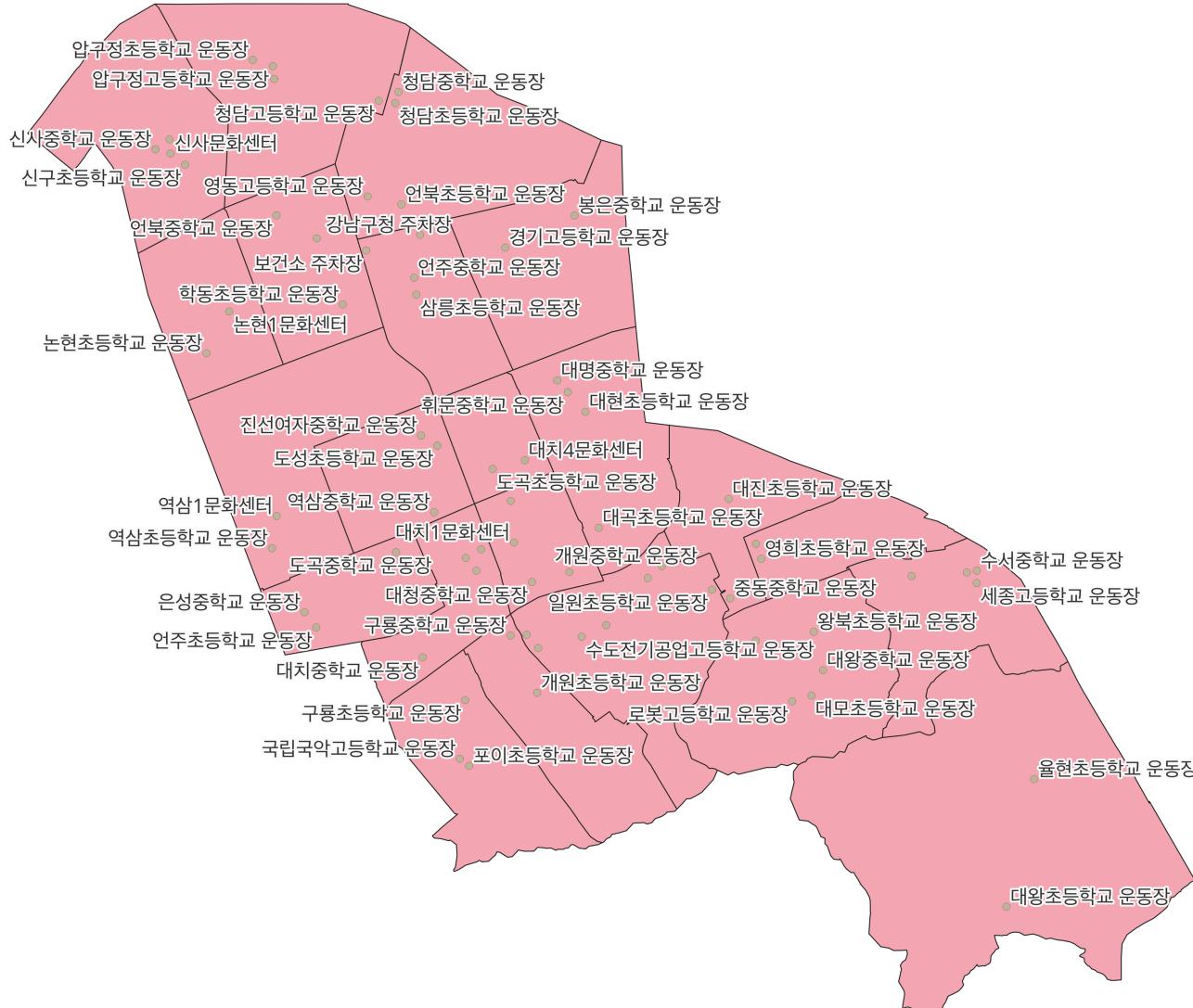


경,위도 좌표를 Point Layer로 추가하기 – 결과 확인

□ 강남구 지진해일 대피소에 대한 지오코딩 결과

1. 구글스프레드시트 활용

2. 브이월드 API 활용



1. 구글스프레드시트 활용

2. 브이월드 API 활용



실습하기

구글 스프레드시트 활용



2. 브이월드 Open API 활용



Open API의 이해

□ API와 Open API

API

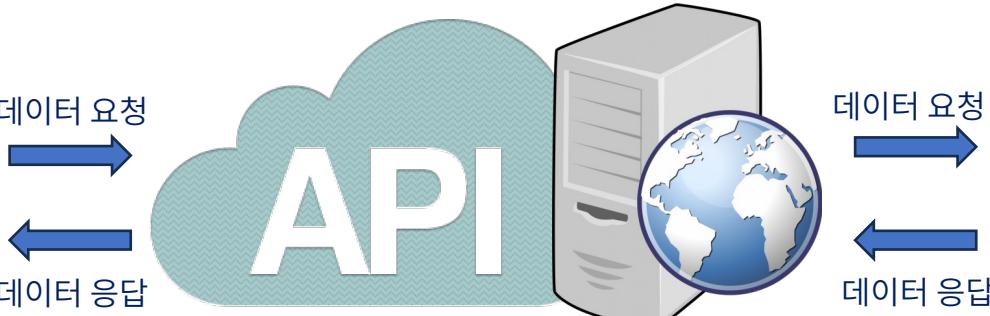
- Application Programming Interface
- 응용 프로그램에서 데이터를 주고 받기 위한 방법

Open API

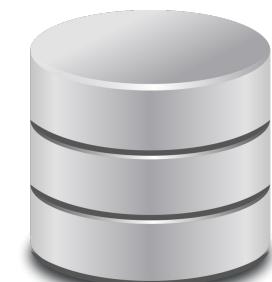
- 누구나 사용할 수 있도록 공개된 API
- 주로 공공 데이터에 대한 제공을 목적으로 관공서, 포털 등이 구축한다.
- 다양한 종류의 프로그램과 연동하기 유리하도록 웹 기반으로 구축한다.



CLIENT
여러가지 유형의 프로그램
Python, Java, C, JS 등으로 제작 가능



API
주로 웹 형태의 시스템으로 이루어짐



DATABASE
입력, 수정, 삭제, 조회
SQL 활용

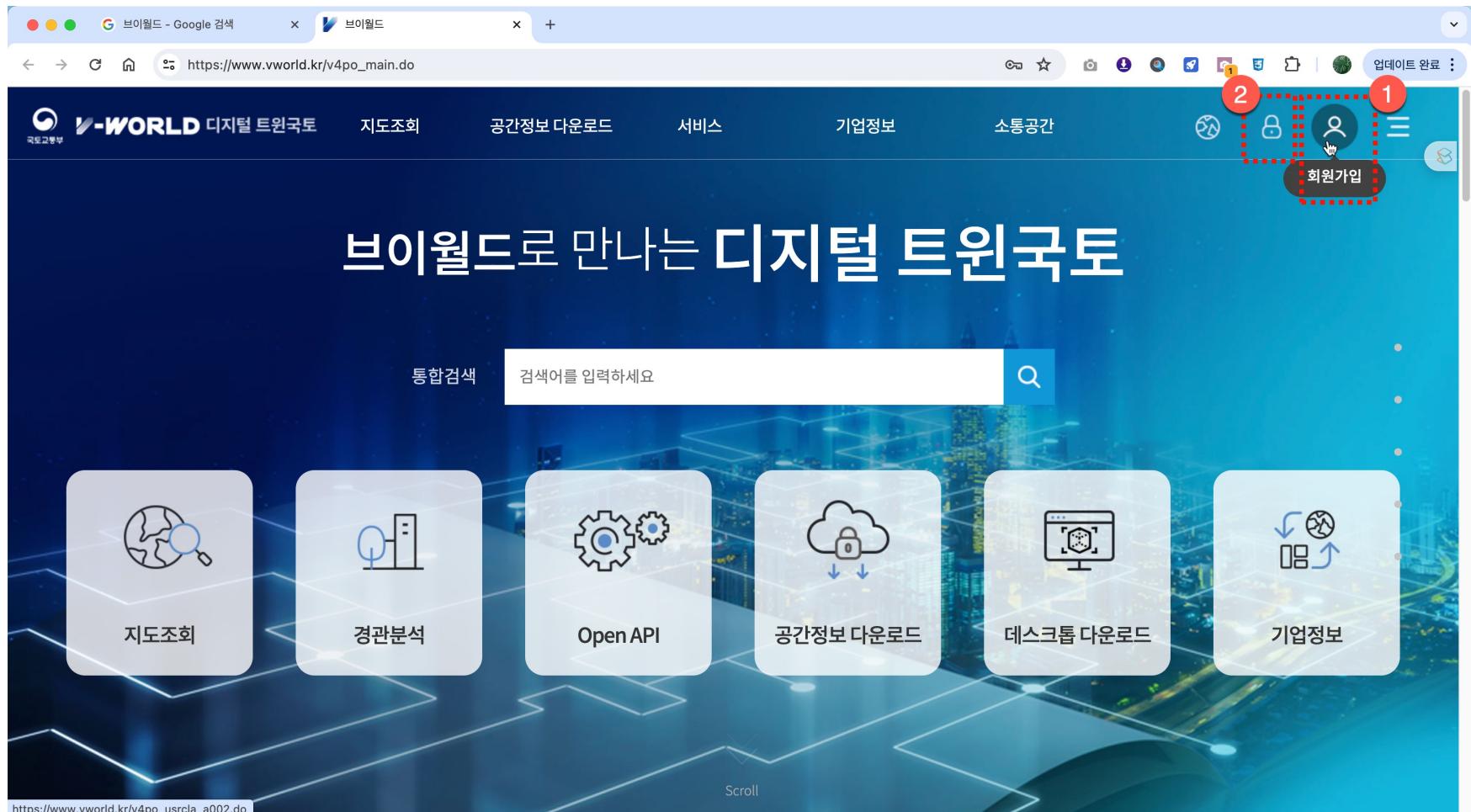
1. 구글스프레드시트 활용

2. 브이월드 API 활용



브이월드 Open API 인증키 발급받기 (1)

- 브이월드(https://www.vworld.kr/v4po_main.do) 회원가입 및 로그인
 - ◎ OpenAPI 연동을 위해서는 서비스를 제공하는 웹 사이트의 회원임을 인증하기 위한 인증키를 발급받아야 한다.



1. 구글스프레드시트 활용

2. 브이월드 API 활용

브이월드 Open API 인증키 발급받기 (2)

□ 인증키 발급을 위한 페이지 이동

The screenshot shows the V-World website at https://www.vworld.kr/v4po_main.do. A red box labeled '1' highlights the '서비스' (Services) button in the top navigation bar. A red box labeled '2' highlights the '인증키' (Authentication Key) link under the 'Open API' section of the dropdown menu. The page features a dark blue background with various service icons and Korean text.

브이월드 Open API 인증키 발급받기 (2)

인증키

Open API

- 지도API
- 국가주체
- 인증키
- 활용사례
- 오픈API 활용모델
- API인큐베이터

3D데스크톱

- 다운로드
- 사용방법안내

부동산 서비스

- 부동산개발업
- 부동산중개업
- 내토지찾기
- 조상땅찾기

지도조회

경관분석

Open API

공간정보 다운로드

데스크톱 다운로드

기업정보

브이월드로 만난다

검색어를 입력하세요

통합검색

Scoll

javascript:showLoginForm('/dev/v4dv_apikey_s001.do');

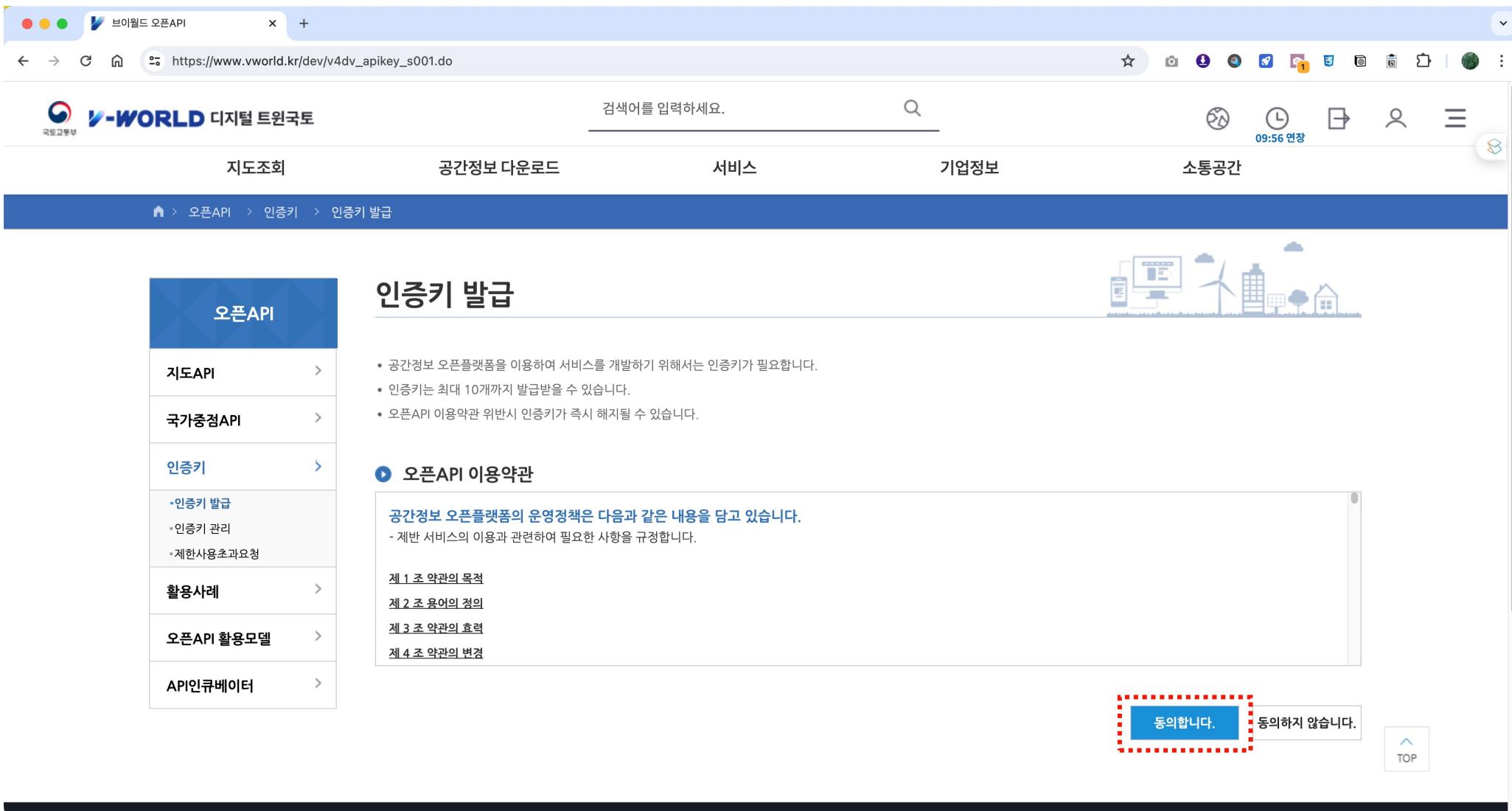
ITWILL 이광호강사

1. 구글스프레드시트 활용

2. 브이월드 API 활용

브이월드 Open API 인증키 발급받기 (3)

□ 인증키 발급 약관 동의하기



The screenshot shows the V-World Open API website's '인증키 발급' (API Key Issuance) page. On the left, a sidebar menu lists various API categories. The main content area is titled '인증키 발급' and contains a summary of the requirements for using spatial information services. Below this, a section titled '● 오픈API 이용약관' (Open API Usage Agreement) provides the legal terms. At the bottom right, there are two buttons: '동의합니다.' (Agree) and '동의하지 않습니다.' (Do not agree), with the 'Agree' button being highlighted by a red dashed box.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

브이월드 Open API 인증키 발급받기 (4)

□ 인증키 발급 신청 양식 작성 – 필요 목적에 맞게 양식을 작성한다.

The screenshot shows the '인증키 발급' (API Key Application) page on the V-World website. A sidebar on the left lists various API categories. The main form has several fields and checkboxes highlighted with red boxes and numbers:

- 서비스명 ***: '개인학습용' (highlighted by a red box with number 1)
- 서비스분류 ***: '교육' (highlighted by a red box with number 2)
- 서비스유형 ***: '기타(QGIS, ArcGIS 등)' (highlighted by a red box with number 3)
- 서비스URL***: A text input field containing 'QGIS 학습용 키 발급' (highlighted by a red box with number 4)
- 서비스 설명 ***: A text area with placeholder text and character count (13 / 3000자) (highlighted by a red box with number 5)
- 브이월드 활용API ***: A group of checkboxes including '2D 지도 API' (unchecked), '2D 데이터 API' (unchecked), '2D 모바일 API' (unchecked), '지오코더 API' (checked), '배경지도 API' (unchecked), '검색 API' (unchecked), '이미지 API' (unchecked), '3D 모바일 API' (unchecked), '3D데스크톱 API' (unchecked), 'WMS/WFS API' (unchecked), 'WMTS/TMS API' (unchecked), and '국가중점 API' (unchecked). (highlighted by a red box with number 6)
- 사용기관 ***: Radio buttons for '민간' (checked), '공공' (unchecked), and '개인' (unchecked). (highlighted by a red box with number 7)
- Submit Button**: A button labeled '지도 인증키 발급' (highlighted by a red box with number 8)

1. 구글스프레드시트 활용

2. 브이월드 API 활용



브이월드 Open API 인증키 발급받기 (5)

□ 발급받은 키 보관하기

◎ 이 값을 OpenAPI 연동시 설정해야 하므로 별도로 복사해서 관리한다.

The screenshot shows a web browser window for the V-World API management system. The URL is https://www.vworld.kr/dev/v4dv_apikey_s002.do. The page title is "브이월드 오픈API". The main content area is titled "인증키 관리" (API Key Management). On the left, there is a sidebar menu under "오픈API" with options like "지도API", "국가중점API", "인증키" (which is expanded to show "인증키 발급" and "인증키 관리"), "활용사례", "오픈API 활용모델", and "API인큐베이터". The main content area contains a list of issued API keys, with one entry highlighted by a red dashed box. The highlighted entry shows the key value "개인학습용" (Personal Learning), its status "개발" (Development), and its creation date "2024-12-02". A red text overlay at the bottom right of the highlighted row says "이 값을 복사해서 보관한다." (Copy this value and save it).

서비스명	인증키	상태	만료일
개인학습용	[Redacted]	개발	2024-12-02



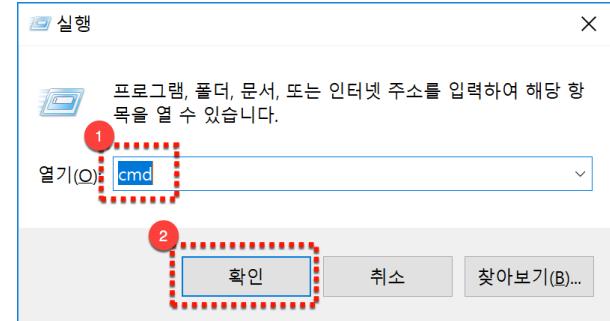
CLI 환경 실행하기

□ CLI (Command Line Interface) - 명령어 라인 인터페이스

- ◎ 사용자와 시스템의 상호작용을 위한 방식이 명령어 입력과 결과 메시지 출력으로 이루어지는 컴퓨팅 환경
- ◎ Windows는 명령프롬프트, Mac은 터미널이라는 프로그램으로 CLI 환경을 사용할 수 있다.

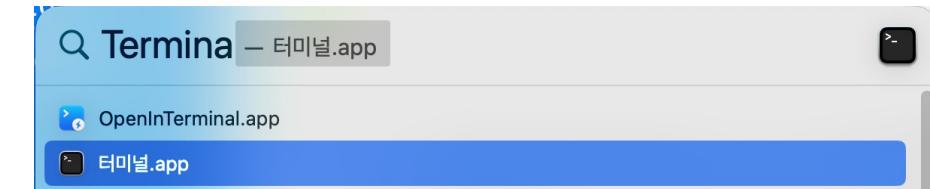
□ Window 환경 – 명령프롬프트 실행하기

WinKey + R > cmd (엔터)



□ Mac 환경 – 터미널 실행하기

⌘ + Space > terminal (엔터)





파이썬 설치 여부 확인하기 (1) - Window의 경우

□ 명령프롬프트에서 파이썬 버전 확인 명령어 수행하기

```
python --version
```

이 수업에서 파이썬을 직접적으로 코딩하는 것은 아니지만 사용하려는 지오코더 프로그램이 파이썬을 기반으로 하기 때문에 파이썬 환경 구성이 필요하다.

파이썬이 설치되어 있는 경우 (3.x 버전이면 가능, 3.11 이상 권장)

```
C:\Windows\system32\cmd + 
Microsoft Windows [Version 10.0.22000.2960]
(c) Microsoft Corporation. All rights reserved.

C:\Users\leekh>python --version
Python 3.11.2

C:\Users\leekh>
```

파이썬이 설치되어 있지 않은 경우 → 설치가 필요함

```
C:\Windows\system32\cmd + 
Microsoft Windows [Version 10.0.22000.2960]
(c) Microsoft Corporation. All rights reserved.

C:\Users\leekh>python --version
'python'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

C:\Users\leekh>
```

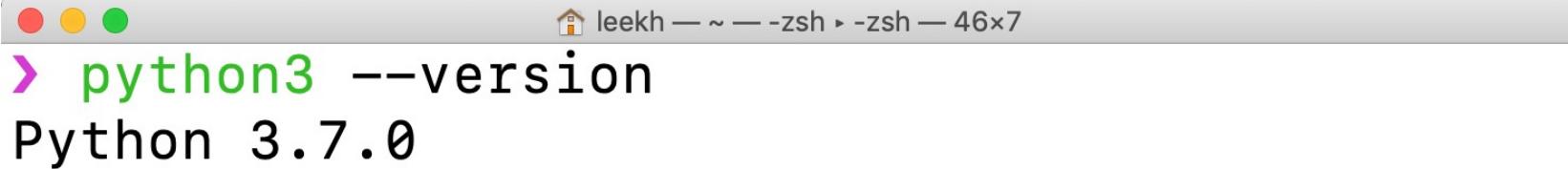


파이썬 설치 여부 확인하기 (2) - Mac의 경우

- 터미널에서 파이썬 버전 명령어 수행하기

```
python3 --version
```

파이썬이 설치되어 있는 경우



```
leekh ~ -- zsh -zsh -46x7
> python3 --version
Python 3.7.0
```

파이썬이 설치되어 있지 않은 경우 → 설치가 필요함



```
leekh ~ -- zsh -zsh -46x9
> python3 --version
zsh: command not found: python3
```



파이썬 설치 (1)

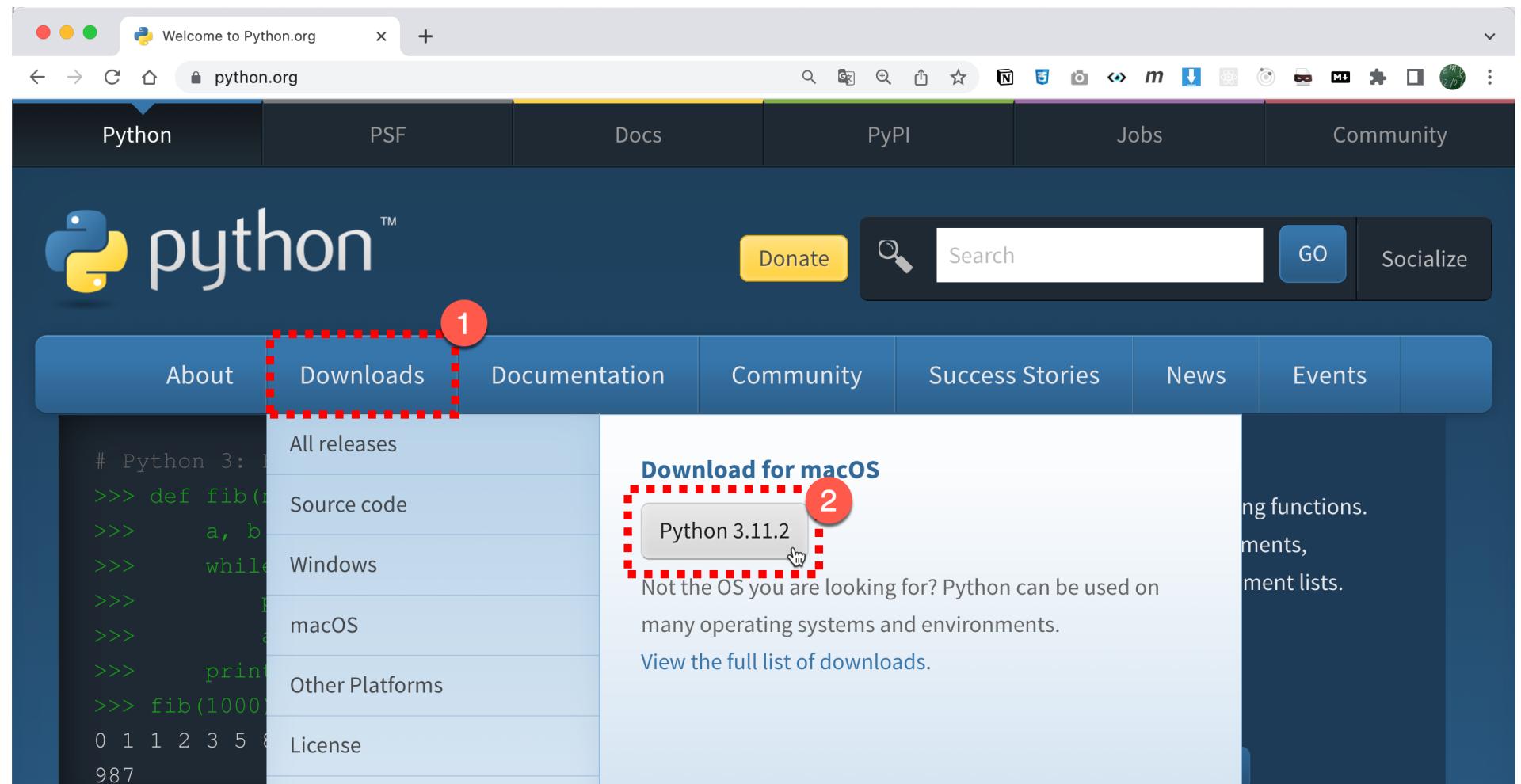
□ 파이썬 설치가 필요한 경우 설치 프로그램 다운로드

◎ 3.11 버전 혹은 그 이후 버전을 내려 받는다.

“<https://www.python.org>

1. 구글스프레드시트 활용

2. 브이월드 API 활용





1. 구글스프레드시트 활용
2. 브이월드 API 활용

□ Windows에서 파이썬 설치 시 주의사항

- ◎ 내려 받은 설치 프로그램을 실행하여 설치를 진행한다.
- ◎ 첫 화면의 맨 아래에 있는 “Add python.exe to PATH” 항목을 반드시 체크하고 다음으로 넘어간다. (Window에만 해당함)
- ◎ Mac의 경우는 특별한 이슈 없이 설치가 가능하다.





pip 업그레이드

1. 구글스프레드시트 활용

2. 브이월드 API 활용

□ PIP

- ◎ 파이썬의 패키지 관리 도구 → 패키지의 설치, 업그레이드, 삭제 기능을 수행한다.
 - 패키지는 파이썬에 추가할 수 있는 확장팩으로 이해하자.

□ PIP 업그레이드 명령어를 CLI 환경에서 수행

- ◎ window의 경우

```
python -m pip install --upgrade pip
```

- ◎ Mac의 경우

```
python3 -m pip install --upgrade pip
```

- ◎ Mac에서 위의 명령이 에러나는 경우

- 관리자 권한을 부여하기 위해 sudo 명령을 추가한다.

```
sudo python3 -m pip install --upgrade pip
```

1. 구글스프레드시트 활용

2. 브이월드 API 활용



py-geocoder 설치하기 (1)

- py-geocoder 소개 (<https://github.com/leekh4232/py-geocoder>)
 - ◎ Python 기반으로 작성된 오픈소스 지오코더 프로그램
 - ◎ 브이월드 OpenAPI와 연동된다.

The screenshot shows the GitHub repository page for 'py-geocoder' by user 'leekh4232'. The page includes the README file, which describes the project as a geocoding tool developed by Lee KwangHo and Ju YoungA, version 0.1.0, using Python and Pandas. It mentions that the tool is intended for a research project at Yonsei University's Department of Urban Planning and Design. The repository has no published packages and is primarily written in Python. Suggested workflows include SLSA Generic generator and Pylint.

Packages
No packages published
[Publish your first package](#)

Languages
Python 100.0%

Suggested workflows
Based on your tech stack

- SLSA Generic generator** [Configure](#)
Generate SLSA3 provenance for your existing release workflows
- Pylint** [Configure](#)
Lint a Python application with pylint.

py-geocoder

Author Lee KwangHo Author Ju YoungA version 0.1.0 license MIT Python Pandas

이 자료는 연세대학교 도시공학과 주영아(j.purplerose@gmail.com)의 논문 프로젝트에 활용되기 위해 작성된 Geocoder 툴입니다.

브이월드 OpenAPI키를 발급받아 사용할 수 있으며, 브이월드 정책에 따라 개발키 적용시 1일 40,000건의 데이터를 처리할 수 있습니다.

비동기를 지원하기 때문에 빠른 처리가 가능합니다.

Installation

pip 명령으로 *.whl 파일을 설치합니다.



py-geocoder 설치하기 (2)

□ py-geocoder 설치하기

◎ window의 경우

```
pip install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl
```

```
C:\Windows\system32\cmd + - Microsoft Windows [Version 10.0.22000.2960]
(c) Microsoft Corporation. All rights reserved.

C:\Users\leekh>pip install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl
Collecting py-geocoder==0.1.0
  Downloading https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl (3.4 kB)
Collecting pandas<3.0.0,>=2.2.2
  Downloading pandas-2.2.2-cp311-cp311-win_amd64.whl (11.6 MB)
    11.6/11.6 MB 34.5 MB/s eta 0:00:00
Collecting tqdm<5.0.0,>=4.66.4
  Downloading tqdm-4.66.4-py3-none-any.whl (78 kB)
    78.3/78.3 kB ? eta 0:00:00
Collecting typer[all]<0.13.0,>=0.12.3
  Downloading typer-0.12.3-py3-none-any.whl (47 kB)
    47.2/47.2 kB ? eta 0:00:00
Collecting numpy>=1.23.2
  Downloading numpy-1.26.4-cp311-cp311-win_amd64.whl (15.8 MB)
    15.8/15.8 MB 54.4 MB/s eta 0:00:00
```

설치 명령을 <https://github.com/leekh4232/py-geocoder> 페이지에 접속하여 복사 후 명령프롬프트(터미널)에 붙여넣기 하여 설치하세요.



py-geocoder 설치하기 (3)

□ py-geocoder 설치하기

◎ Mac의 경우

```
pip3 install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl
```

◎ Mac에서 위의 명령이 에러나는 경우

- 관리자 권한을 부여하기 위해 sudo 명령을 추가한다.

```
sudu pip3 install --upgrade https://raw.githubusercontent.com/leekh4232/py-geocoder/main/dist/py_geocoder-0.1.0-py3-none-any.whl
```



서울시 응급실 위치 정보 데이터 확인

1. 구글스프레드시트 활용
2. 브이월드 API 활용

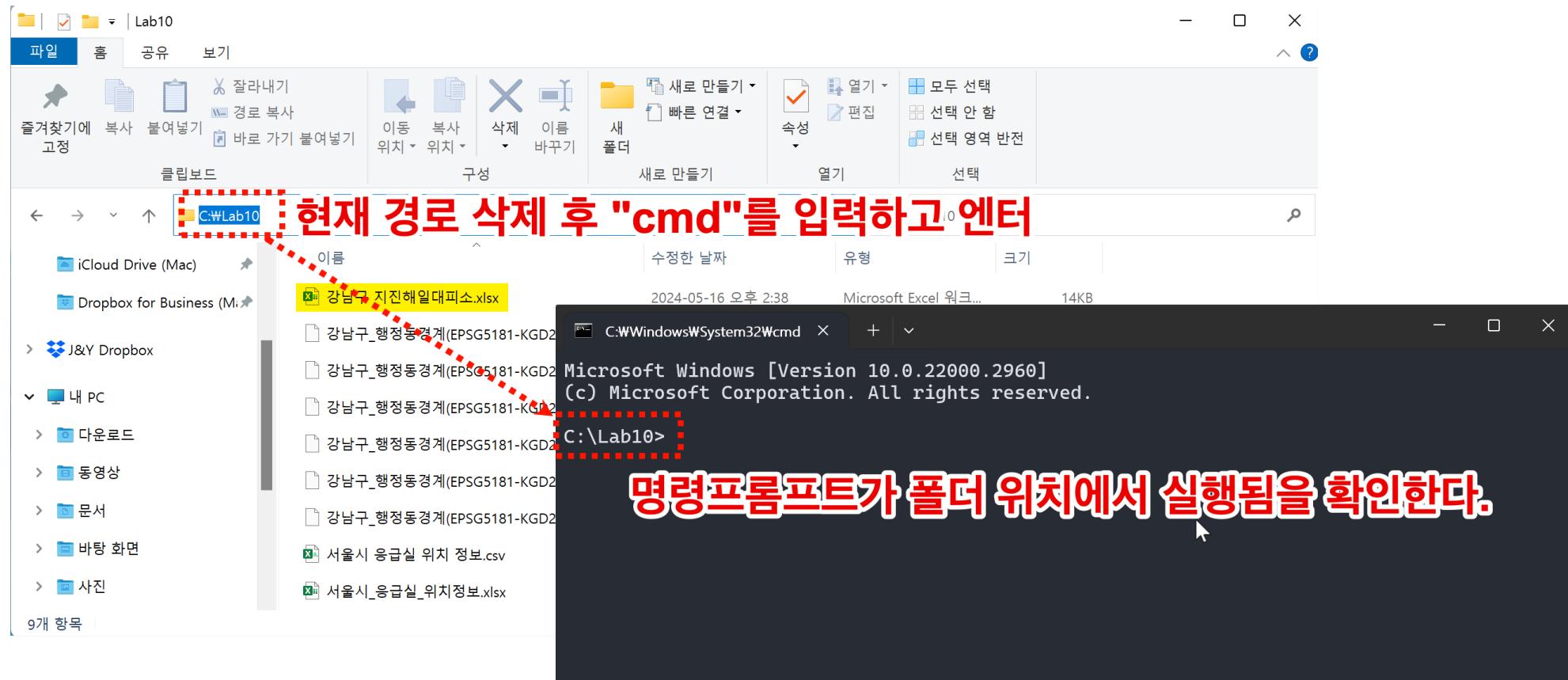
- “서울시_응급실_위치정보.xlsx” (엑셀파일)
 - ◎ 서울시에 위치한 응급실 이름, 주소, 구분, 연락처 등이 정리되어 있는 데이터 자료
 - ◎ 출처: 서울시 공공데이터 포털

기관ID	기관명	주소	병원분류	응급의료기관코드	응급의료기관코드명	대표전화	응급실전화
A1100011	가톨릭대학교 여의도성모병원	서울특별시 영등포구 63로 10, 여의도성모병원 (여의도동)	종합병원	G006	지역응급의료센터	1661-7575	02-3779-1188
A1121013	가톨릭대학교 은평성모병원	서울특별시 은평구 통일로 1021 (진관동)	종합병원	G006	지역응급의료센터	02-1811-7755	02-1811-7755
A1100047	강남고려병원	서울특별시 관악구 관악로 242 (봉천동)	병원	G009	응급의료기관 외의 의료기관(응급의료시설)	1661-8267	02-874-7227
A1100141	강남베드로병원	서울특별시 강남구 남부순환로 2649, 베드로병원 (도곡동)	종합병원	G009	응급의료기관 외의 의료기관(응급의료시설)	1544-7522	1544-7522
A1100076	강남힐병원	서울특별시 관악구 남부순환로 1449, 강남힐병원 (신림동)	병원	G009	응급의료기관 외의 의료기관(응급의료시설)	02-853-4600	02-853-9100
A1100043	강동경희대학교의대병원	서울특별시 강동구 도남로 892 (상일동)	종합병원	G001	권역응급의료센터	02-440-7114	02-440-7000
A1100006	강북삼성병원	서울특별시 종로구 새문안로 29 (평동)	종합병원	G006	지역응급의료센터	02-2001-2001	02-2001-1000
A1121842	강북으뜸병원	서울특별시 강북구 도봉로 187, 지하1층, 2층~5층 (미아동)	병원	G009	응급의료기관 외의 의료기관(응급의료시설)	02-986-0334	02-986-0334
A1100002	건국대학교병원	서울특별시 광진구 능동로 120-1 (화양동)	종합병원	G006	지역응급의료센터	1588-1533	02-2030-5555
A1100039	경찰병원	서울특별시 송파구 송이로 123, 국립경찰병원 (가락동)	종합병원	G007	지역응급의료기관	02-3400-1114	02-3400-1300
A1100001	경희대학교병원	서울특별시 동대문구 경희대로 23 (경희동)	종합병원	G006	지역응급의료센터	02-958-8114	02-958-8114
A1100014	고려대학교의과대학부속구로병원	서울특별시 구로구 구로동로 148, 고려대부속구로병원 (구로동)	종합병원	G001	권역응급의료센터	02-2626-1114	02-2626-1550
A1100026	구로성심병원	서울특별시 구로구 경인로 427, 구로성심병원 (고척동)	종합병원	G007	지역응급의료기관	02-2067-1500	02-2067-1515
A1100052	국립중앙의료원	서울특별시 종구 을지로 245, 국립중앙의료원 (을지로6가)	종합병원	G006	지역응급의료센터	02-2260-7114	02-2260-7414
A1100048	노원을지대학교병원	서울특별시 노원구 한글비석로 68, 을지병원 (하계동)	종합병원	G006	지역응급의료센터	02-970-8000	02-970-8282
A1100044	녹색병원	서울특별시 종량구 사가정로49길 53 (면목동)	종합병원	G007	지역응급의료기관	02-490-2000	02-490-2113
A1100037	대림성모병원	서울특별시 영등포구 시흥대로 657 (대림동, 대림성모병원)	종합병원	G007	지역응급의료기관	02-829-9000	02-829-9129
A1100024	명지성모병원	서울특별시 영등포구 도림로 156, 명지성모병원 (대림동)	종합병원	G007	지역응급의료기관	02-1899-1475	02-829-7800
A1100046	미즈메디병원	서울특별시 강서구 강서로 295 (내발산동)	종합병원	G009	응급의료기관 외의 의료기관(응급의료시설)	02-2007-1252	02-2007-1118
A1100036	부민병원	서울특별시 강서구 공항대로 389, 부민병원 (등촌동)	종합병원	G007	지역응급의료기관	1577-7582	02-2620-0119
A1100148	사랑의병원	서울특별시 관악구 남부순환로 1860, -1, 1, 3, 4, 5층 (봉천동,)	병원	G009	응급의료기관 외의 의료기관(응급의료시설)	02-880-0114	02-880-0119
A1100010	삼성서울병원	서울특별시 강남구 일원로 81 (일원동, 삼성의료원)	종합병원	G006	지역응급의료센터	02-3410-2114	02-3410-2060
A1100021	삼육서울병원	서울특별시 동대문구 망우로 82 (휘경동)	종합병원	G006	지역응급의료센터	02-2244-0191	02-2210-3566
A1100017	서울대학교병원	서울특별시 종로구 대학로 101 (연건동)	종합병원	G001	권역응급의료센터	1588-5700	02-2072-2475
A1100050	서울성심병원	서울특별시 동대문구 왕산로 259, 서울성심병원 (청량리동)	종합병원	G007	지역응급의료기관	02-966-1616	02-966-1616
A1100029	서울적십자병원	서울특별시 종로구 새문안로 9, 적십자병원 (평동)	종합병원	G007	지역응급의료기관	02-2002-8000	02-2002-8888
A1100022	서울특별시동부병원	서울특별시 동대문구 무학로 124 (용두동)	종합병원	G007	지역응급의료기관	02-920-9114	02-920-9119
A1100040	서울특별시보라매병원	서울특별시 동작구 보라매로5길 20 (신대방동)	종합병원	G006	지역응급의료센터	02-870-2114	02-870-2119

1. 구글스프레드시트 활용
2. 브이월드 API 활용

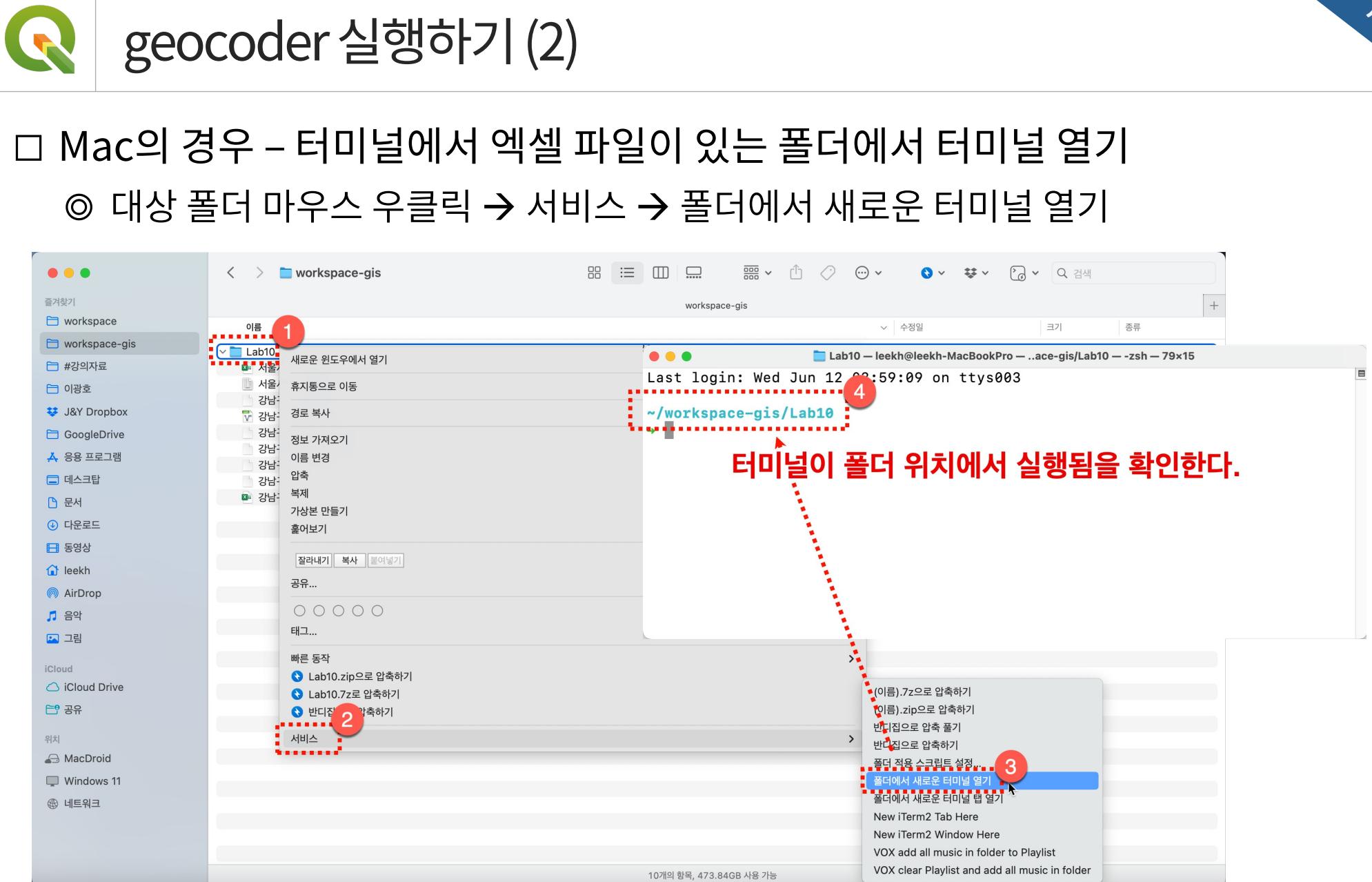
geocoder 실행하기 (1)

- Windows의 경우 - 엑셀 파일이 있는 위치에서 명령프롬프트 실행하기
 - ◎ 윈도우 탐색기를 엑셀 파일이 포함되어 있는 폴더의 위치로 이동하고, 경로 표시줄에 “cmd”를 입력후 엔터를 누른다.



1. 구글스프레드시트 활용

2. 브이월드 API 활용



The QGIS logo consists of a green stylized letter 'Q' with a 3D orange and yellow block extending from the top-left corner of its vertical stroke.

geocoder 실행하기 (3)

□ py-geocoder 실행하기 (Windows, Mac 공통)

◎ 아래의 명령어 형식에 맞춰 실행한다.

```
py-geocoder --key={브이월드OpenAPI키} --input={주소가작성된엑셀파일명} --addr={주소필드이름}
```



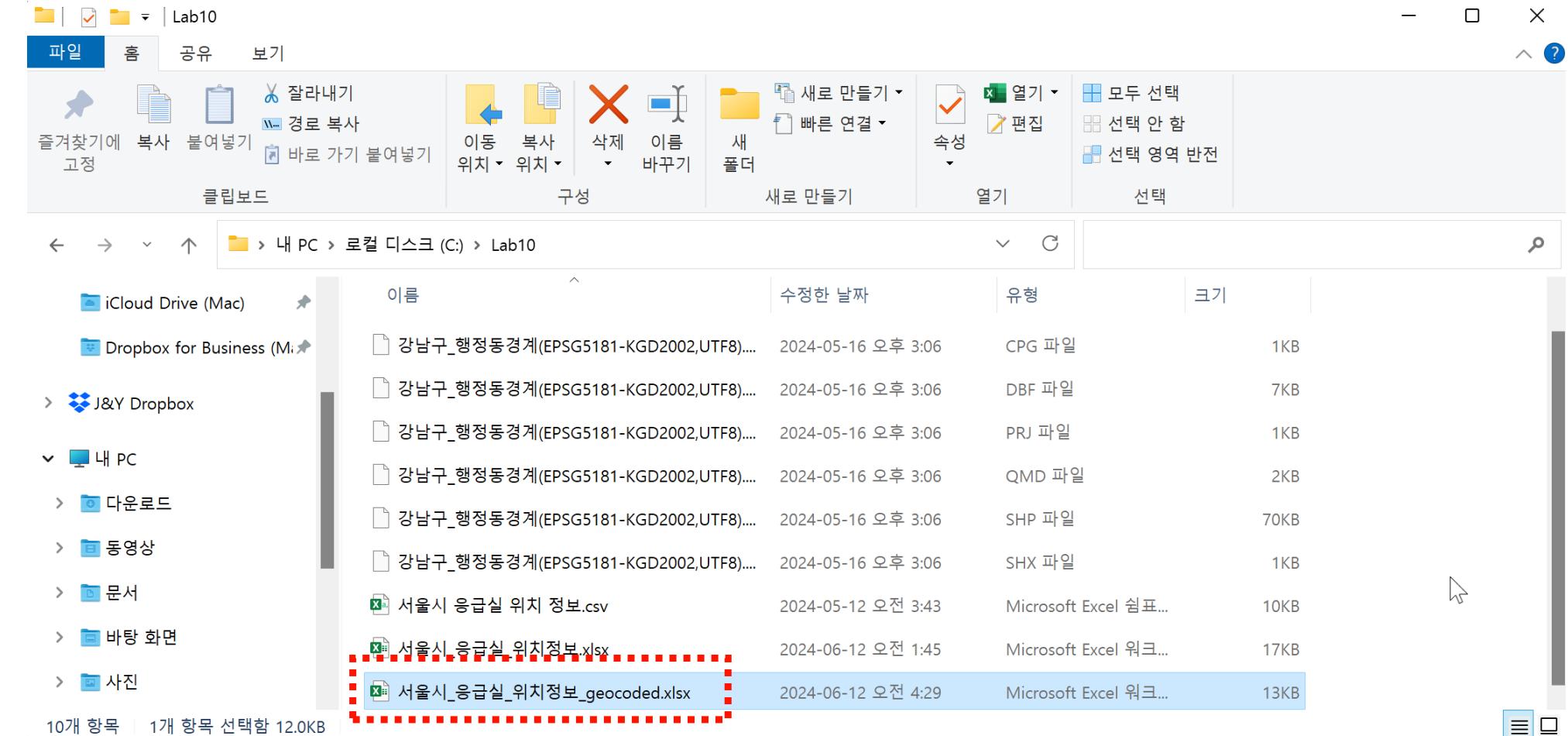
geocoder 실행하기 (4)

□ geocoder 결과 확인

◎ “[기존파일이름]_geocoded.xlsx” 파일이 생성됨을 확인할 수 있다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용



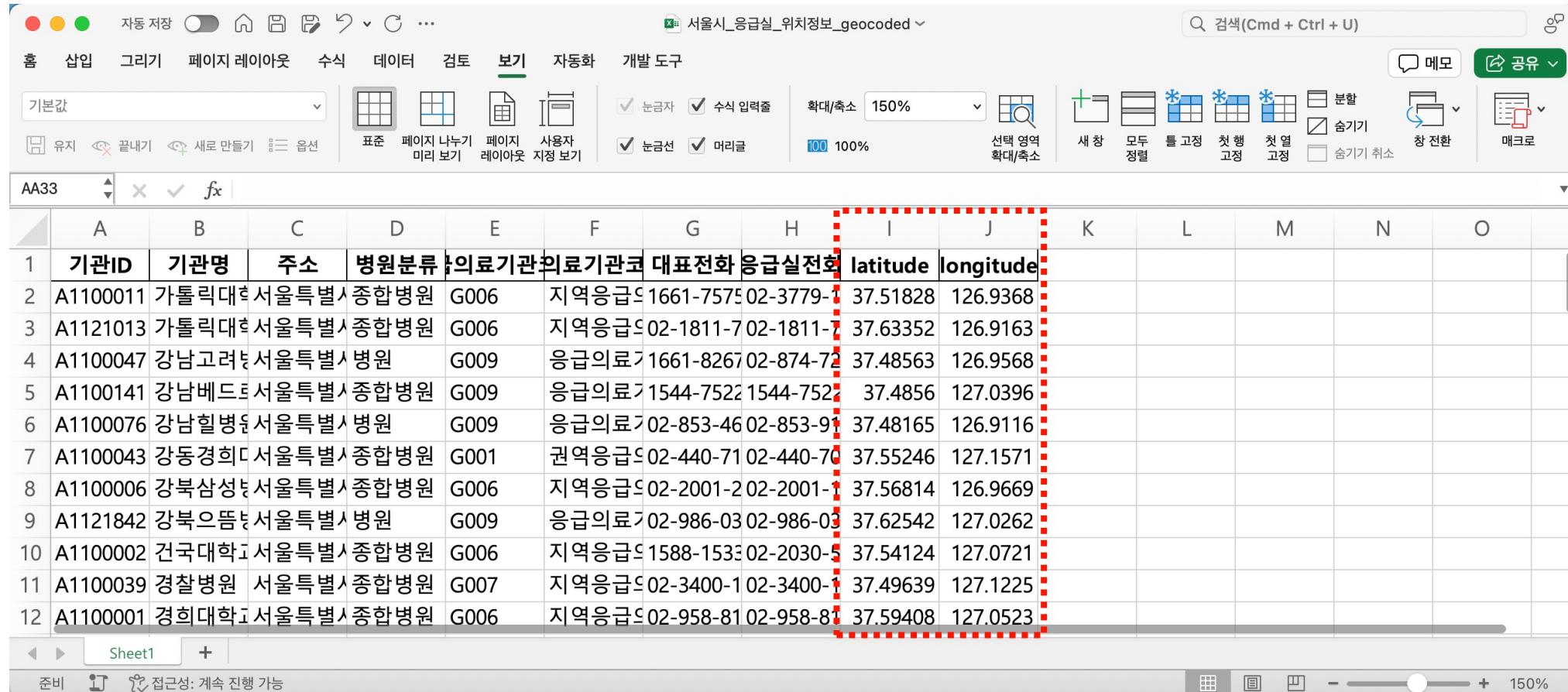
1. 구글스프레드시트 활용

2. 브이월드 API 활용

geocoder 실행하기 (5)

□ 주소 데이터에 대한 경,위도 확인

◎ 새로 생성된 엑셀 파일상에 경,위도 필드가 추가되어 있음을 확인한다.

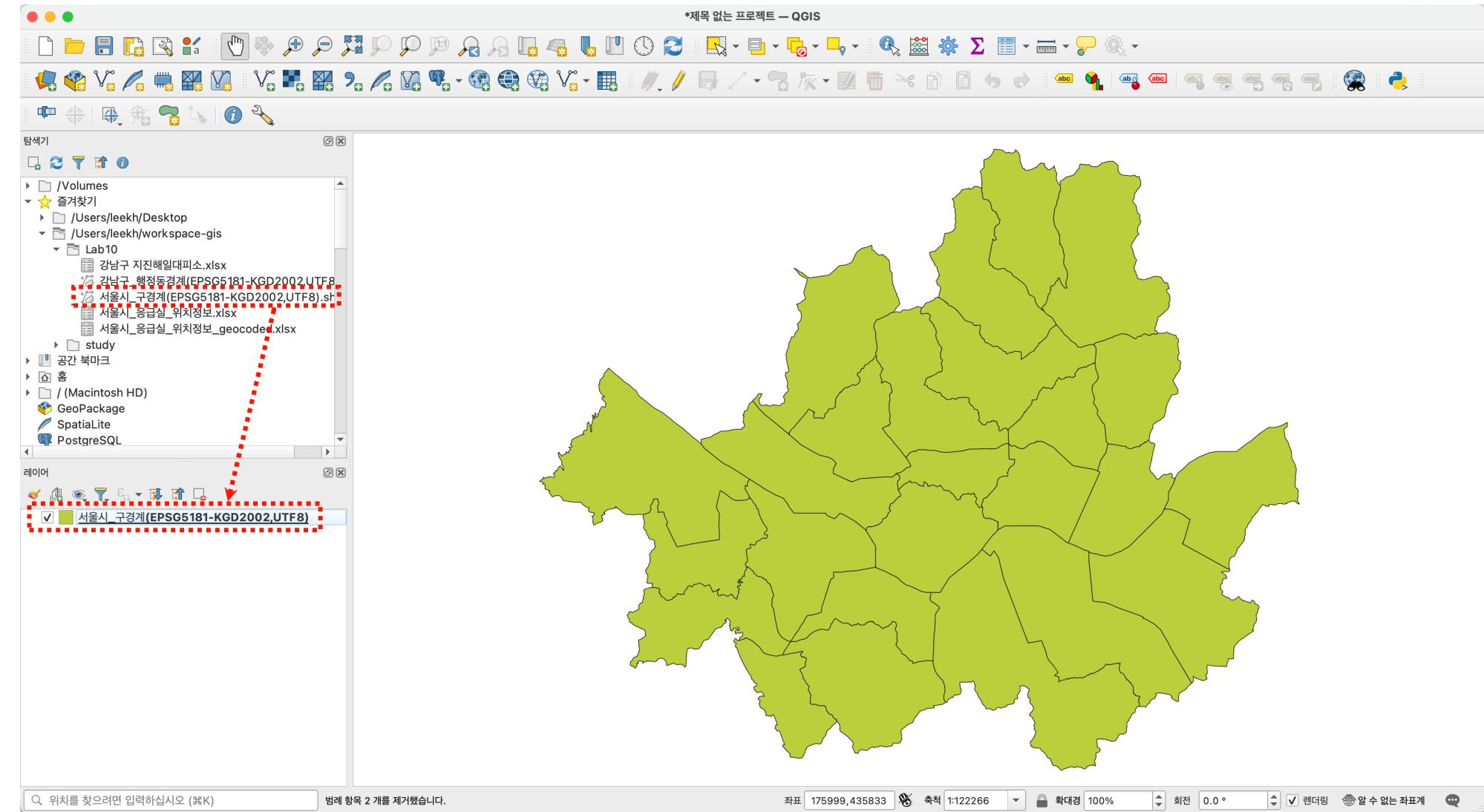


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	I
1	기관ID	기관명	주소	병원분류	의료기관	의료기관코드	대표전화	응급실전화	latitude	longitude						
2	A1100011	가톨릭대서울특별시 종합병원	G006	지역응급의료기관	1661-7575	02-3779-1700	37.51828	126.9368								
3	A1121013	가톨릭대학서울특별시 종합병원	G006	지역응급의료기관	02-1811-702-1811-7000	1811-702-1811-7000	37.63352	126.9163								
4	A1100047	강남고려한서울특별시 병원	G009	응급의료기관	1661-8267	02-874-7200	37.48563	126.9568								
5	A1100141	강남베드로서울특별시 종합병원	G009	응급의료기관	1544-7522	1544-7522	37.4856	127.0396								
6	A1100076	강남힐병원서울특별시 병원	G009	응급의료기관	02-853-4602	853-9100	37.48165	126.9116								
7	A1100043	강동경희대학교서울특별시 종합병원	G001	권역응급의료기관	02-440-7102	440-7000	37.55246	127.1571								
8	A1100006	강북삼성한서울특별시 종합병원	G006	지역응급의료기관	02-2001-2002	2001-1000	37.56814	126.9669								
9	A1121842	강북으뜸한서울특별시 병원	G009	응급의료기관	02-986-0302	986-0302	37.62542	127.0262								
10	A1100002	건국대학교서울특별시 종합병원	G006	지역응급의료기관	1588-1533	02-2030-5000	37.54124	127.0721								
11	A1100039	경찰병원서울특별시 종합병원	G007	지역응급의료기관	02-3400-1002	3400-1000	37.49639	127.1225								
12	A1100001	경희대학교서울특별시 종합병원	G006	지역응급의료기관	02-958-8102	958-8100	37.59408	127.0523								



서울시 구 단위 응급실 위치 지도 만들기 (1)

- “서울시_구경계(EPSG5181-KGD2002,UTF8).shp” 파일을 작업 레이어에 추가





서울시 구 단위 응급실 위치 지도 만들기 (2)

1. 구글스프레드시트 활용
2. 브이월드 API 활용

□ 지오코딩 결과가 포함된 엑셀 파일을 Point 레이어로 추가

- ◎ 툴바에서 “Add Spreadsheet Layer” 버튼을 클릭한다.
- ◎ 경위도 기반이므로 좌표계는 “EPSG 4326 - WGS84”를 지정해야 한다.



Add Spreadsheet Layer...

The screenshot shows two QGIS dialog boxes:

Create a Layer from a Spreadsheet File Dialog (Left):

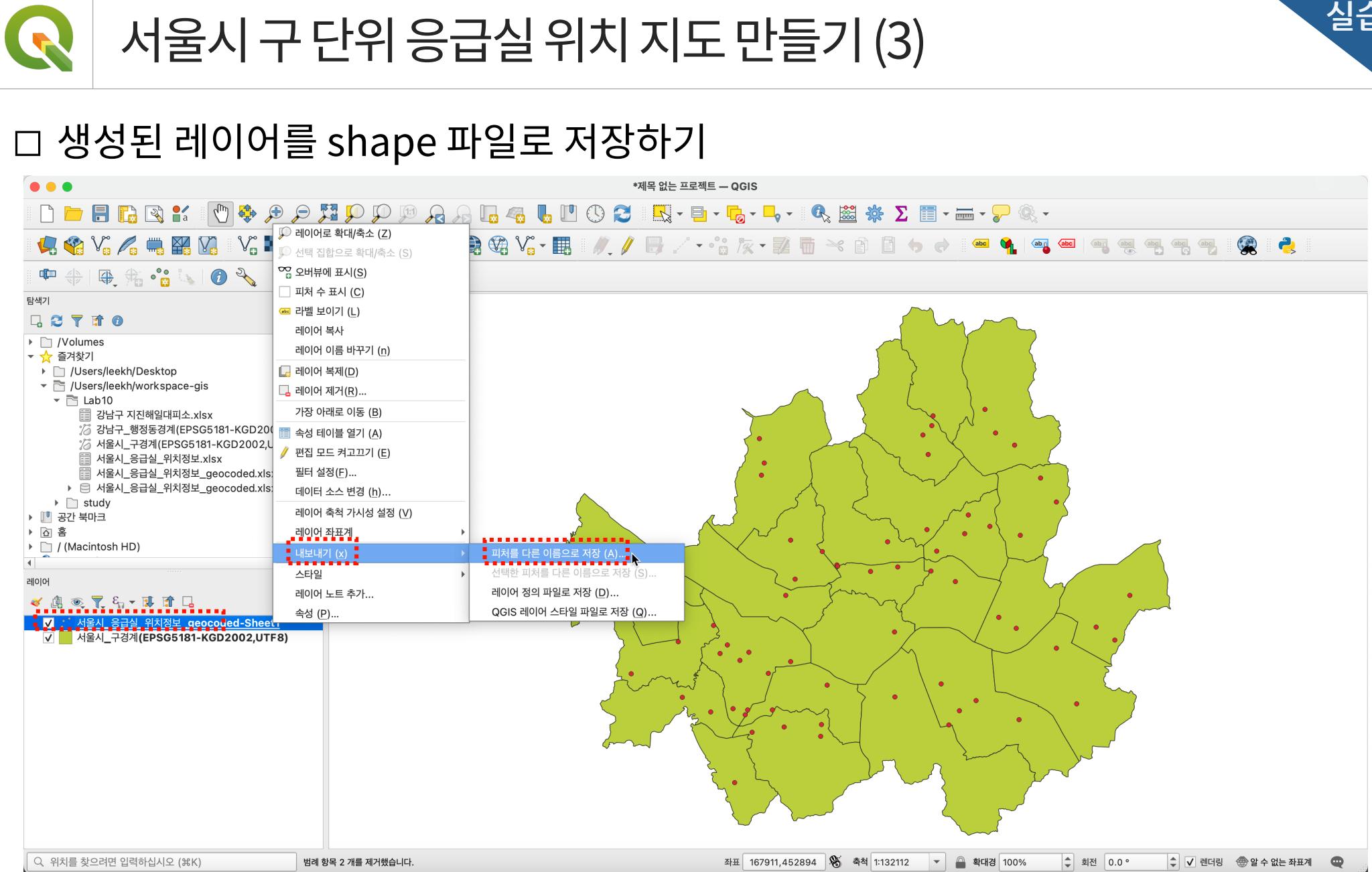
- File Name: /Users/leekh/workspace-gis/Lab10/서울시_응급실_위치정보_geocoded.xlsx (highlighted by red box 1)
- Sheet: Sheet1 (highlighted by red box 2)
- Layer name: 서울시_응급실_위치정보_geocoded-Sheet1
- Row 0: Number of lines to ignore: 0 (highlighted by red box 3)
- Geometry: Encoding: PointFromColumns (highlighted by red box 4)
- Field: X field: longitude, Y field: latitude (highlighted by red box 5)
- Reference system: 무결하지 않은 투영체 (highlighted by red box 6)

Coordinate Reference System Selector Dialog (Right):

- Search bar: 사전 정의 좌표계 (highlighted by red box 6)
- Filter: 4326 (highlighted by red box 7)
- Search button: 검색 (highlighted by red box 8)
- Result list: WGS 84 (highlighted by red box 9)
- Details panel: WGS 84 (Geographic (uses latitude and longitude for coordinates), EPSG:4326) (highlighted by red box 10)

1. 구글스프레드시트 활용

2. 브이월드 API 활용



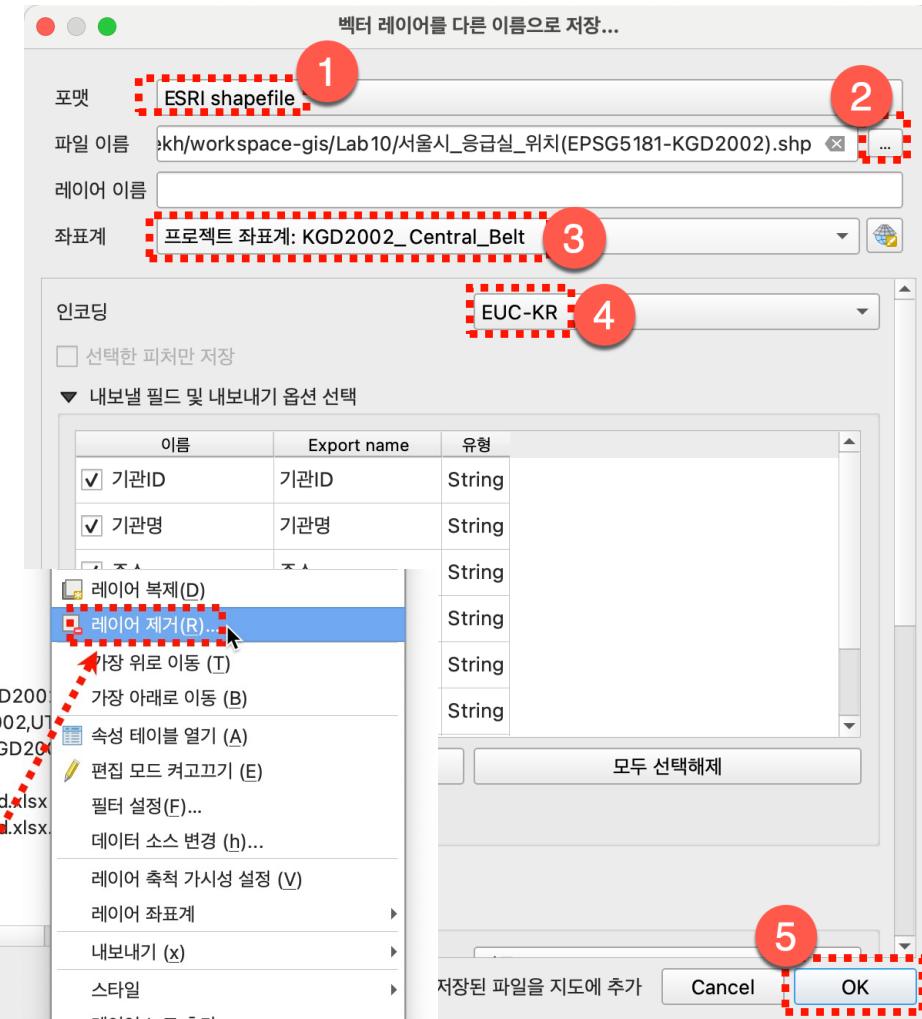
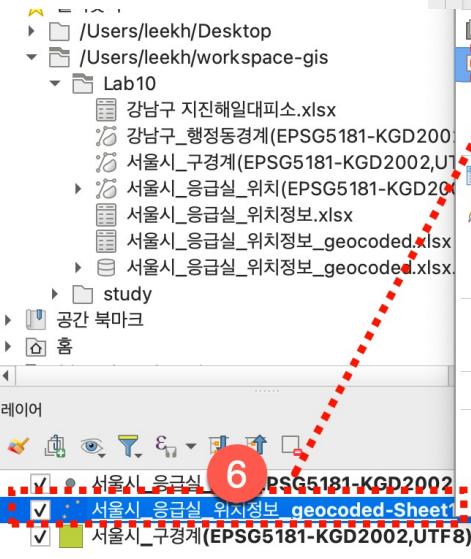


서울시 구 단위 응급실 위치 지도 만들기 (4)

1. 구글스프레드시트 활용
2. 브이월드 API 활용

□ 저장 파라미터 설정

- ① 저장 포맷을 shape 파일로 설정
- ② 저장 경로 지정
- ③ 좌표계를 변경하고자 할 경우 설정
 - 여기서는 프로젝트 좌표계와 일치시킴
- ④ 한글 필드명이 있으므로 인코딩을 EUC-KR로 지정
 - UTF-8은 한글 필드명이 깨진다(QGIS 버그)
- ⑤ 설정 내용 실행
- ⑥ 레이어 저장 후 불필요한 기존 레이어 "삭제"



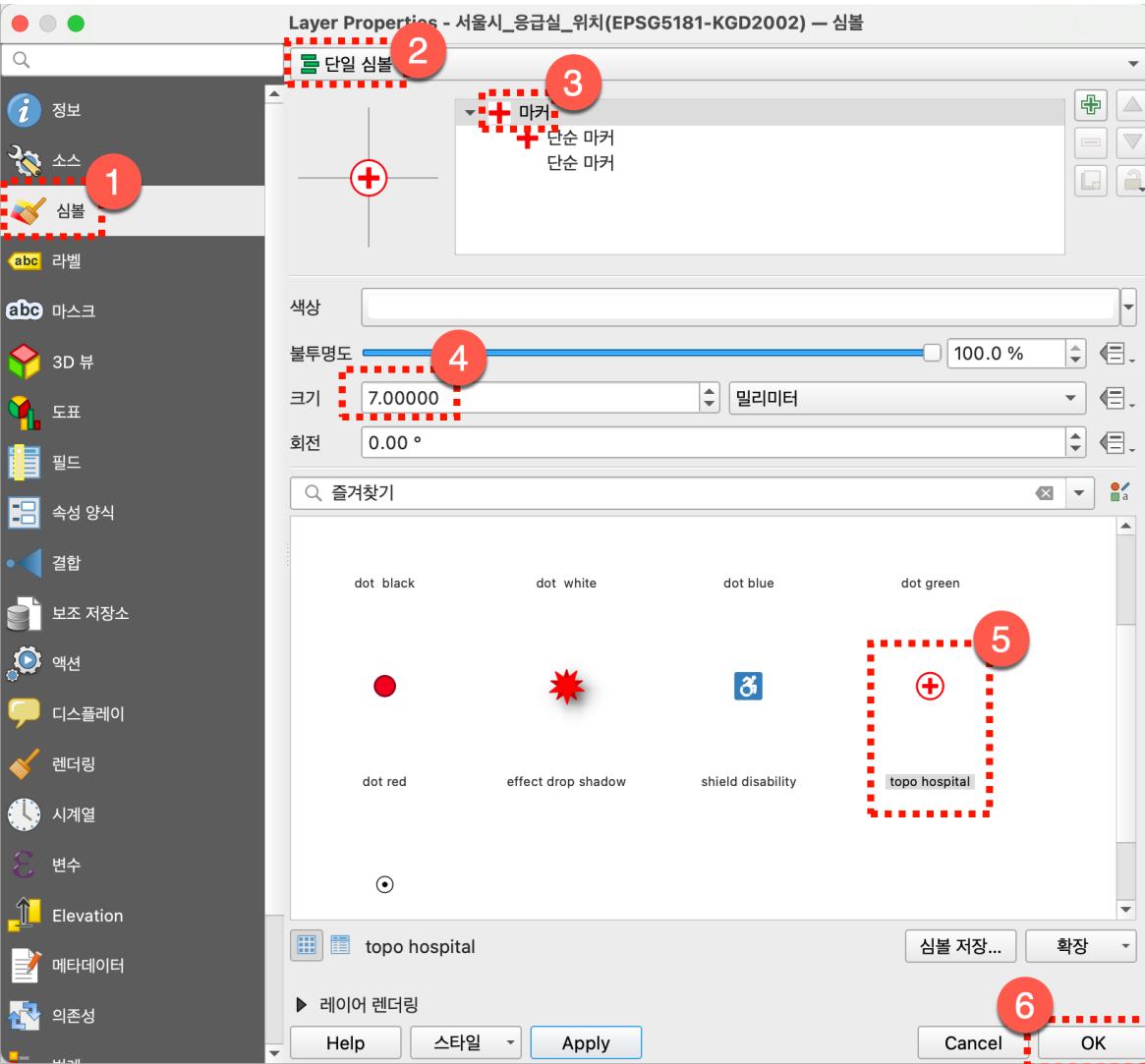
1. 구글스프레드시트 활용

2. 브이월드 API 활용

서울시 구 단위 응급실 위치 지도 만들기 (5)

□ 응급실 shape 레이어의 속성창에서 심볼을 설정한다.

- ① 심볼 선택
- ② 단일 심볼 선택
- ③ 마커 선택
- ④ 마커의 적정 크기 지정
- ⑤ 병원 아이콘 선택
- ⑥ 설정 내용 실행



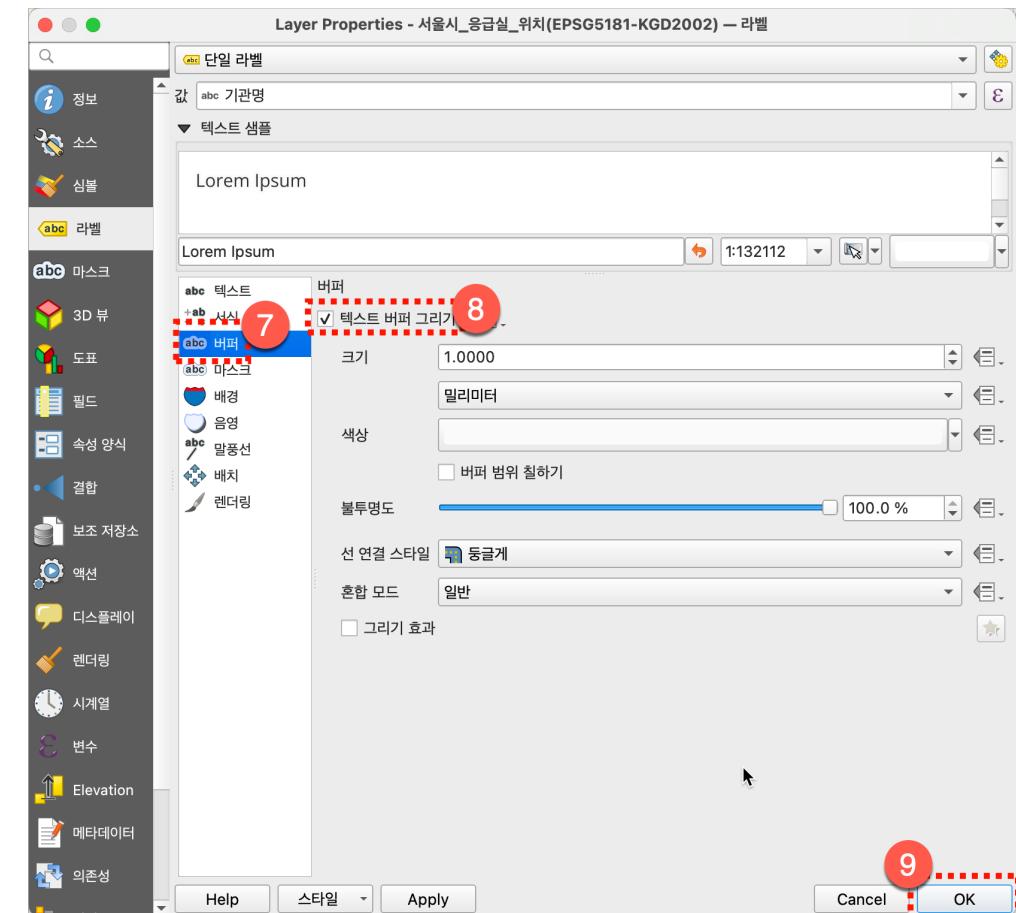
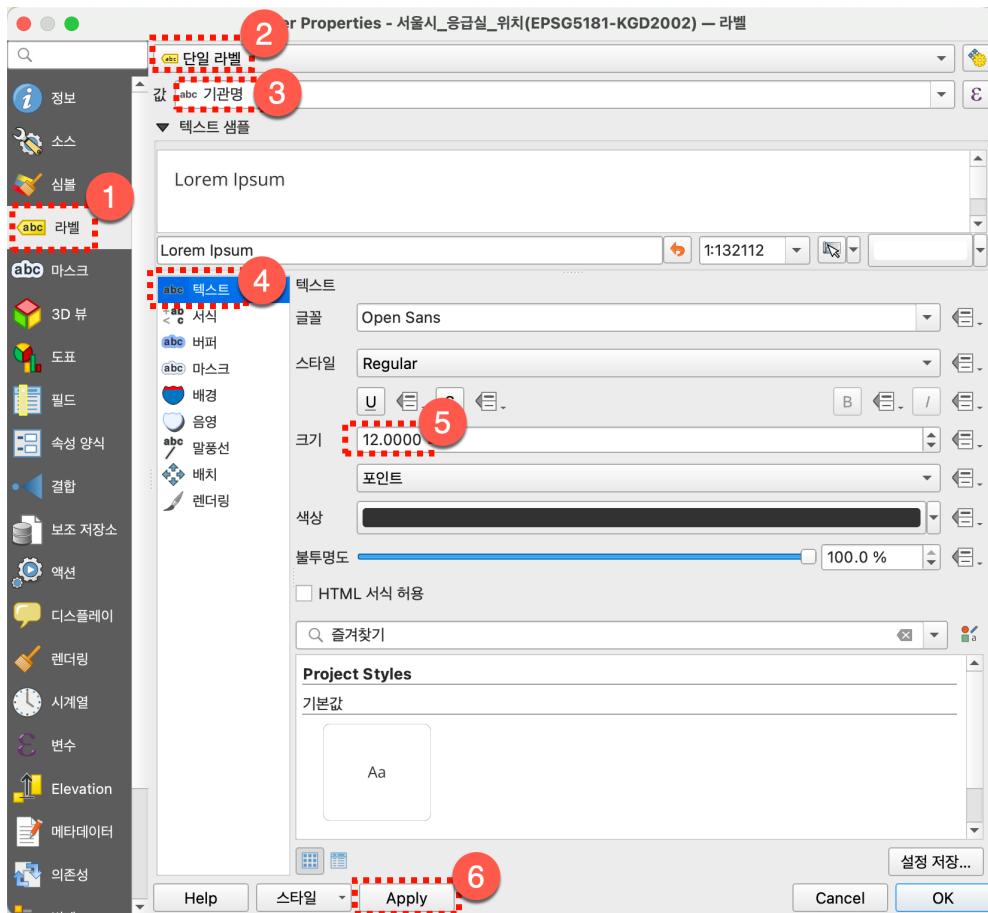


서울시 구 단위 응급실 위치 지도 만들기 (6)

□ 병원 이름에 대한 라벨 설정

1. 구글스프레드시트 활용

2. 브이월드 API 활용

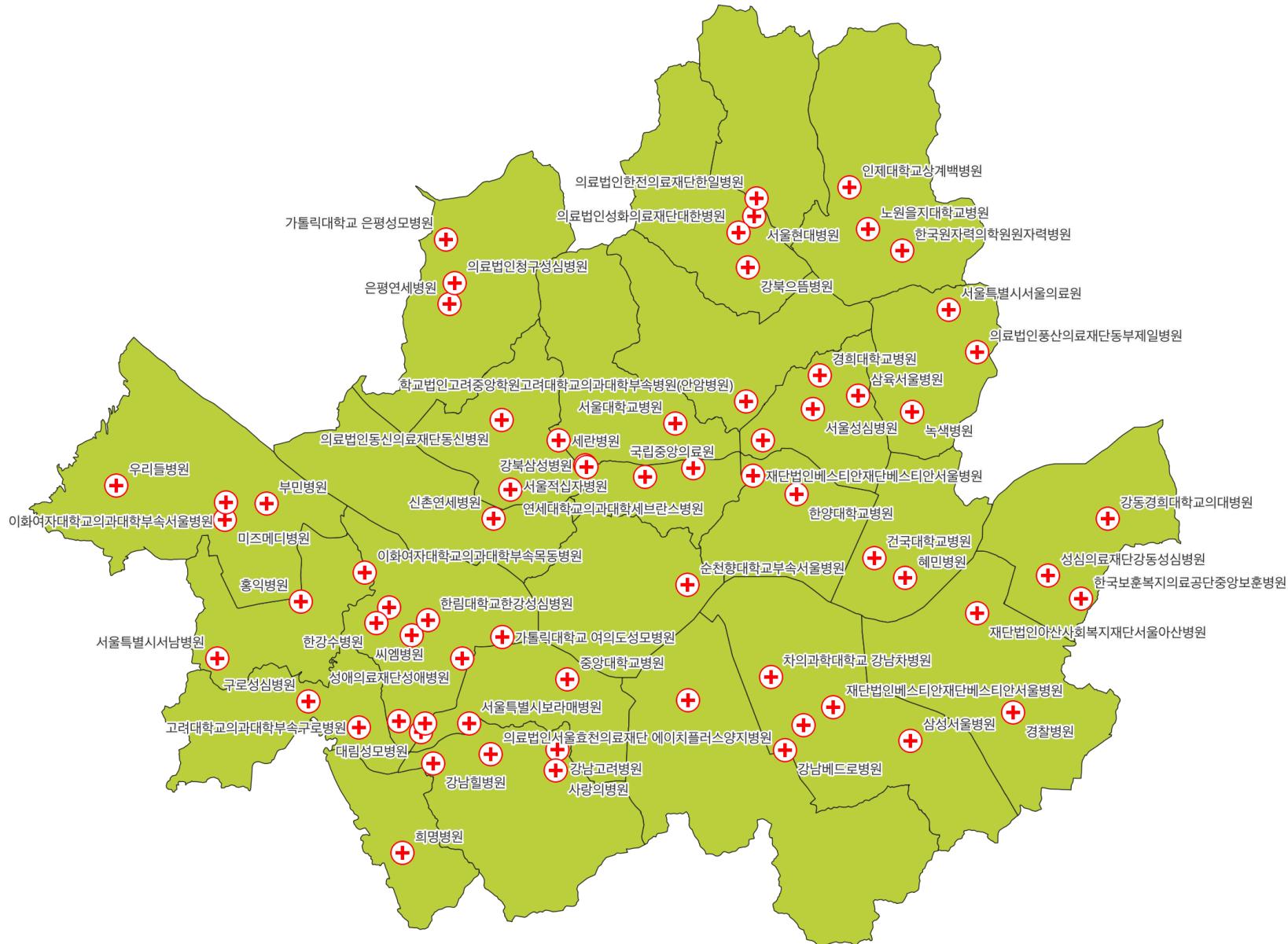




서울시 구 단위 응급실 위치 지도 만들기 (7) – 결과 확인

1. 구글스프레드시트 활용

2. 브이월드 API 활용





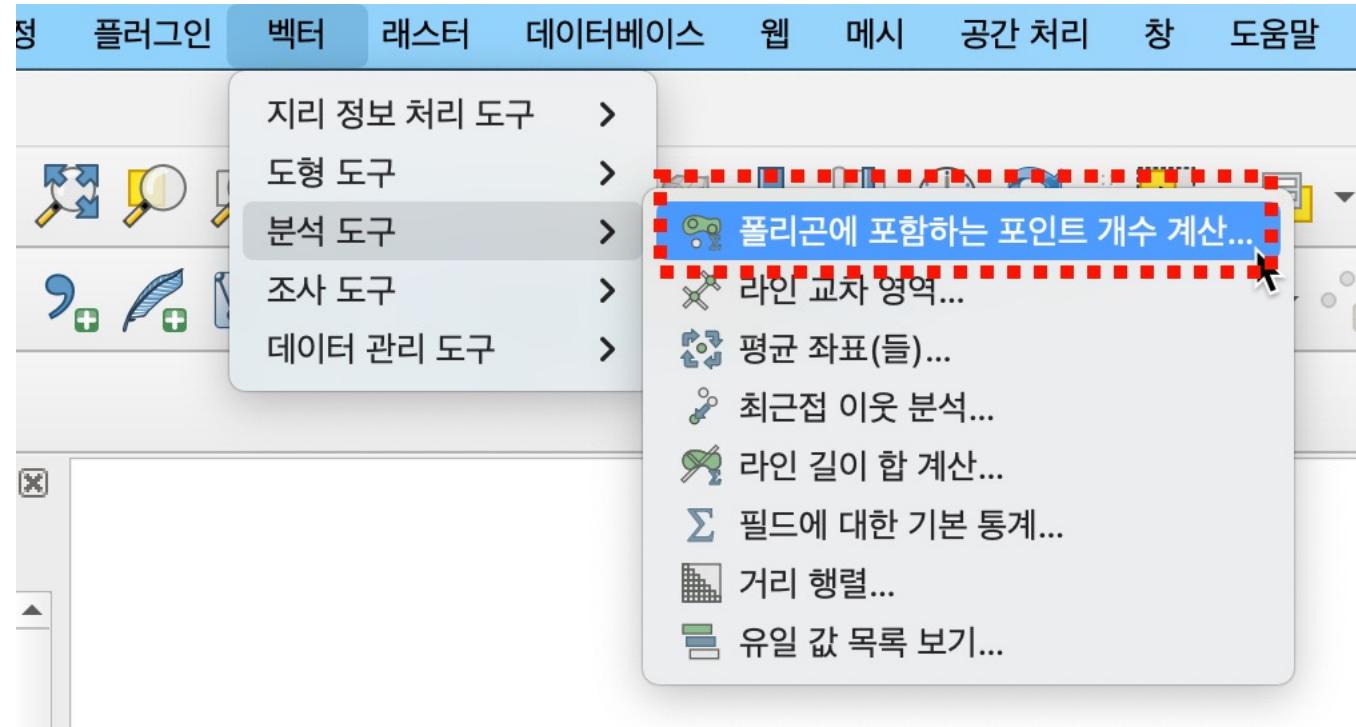
구 단위 응급실 분포 단계 구분도 만들기 (1)

□ 폴리곤에 포함되는 포인트(응급실)의 수 계산하기 (1)

- ◎ “벡터 → 분석 도구 → 폴리곤에 포함하는 포인트 개수 계산” 메뉴를 실행한다.

1. 구글스프레드시트 활용

2. 브이월드 API 활용

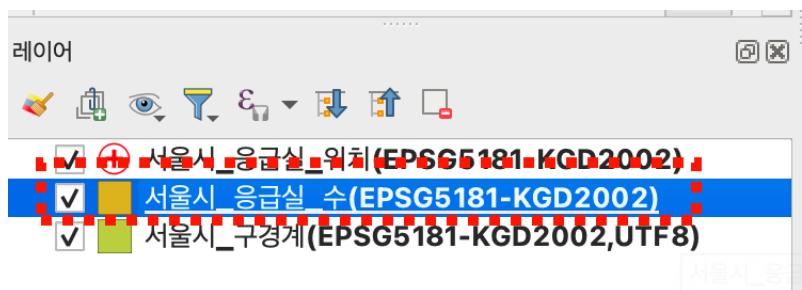




구 단위 응급실 분포 단계 구분도 만들기 (2)

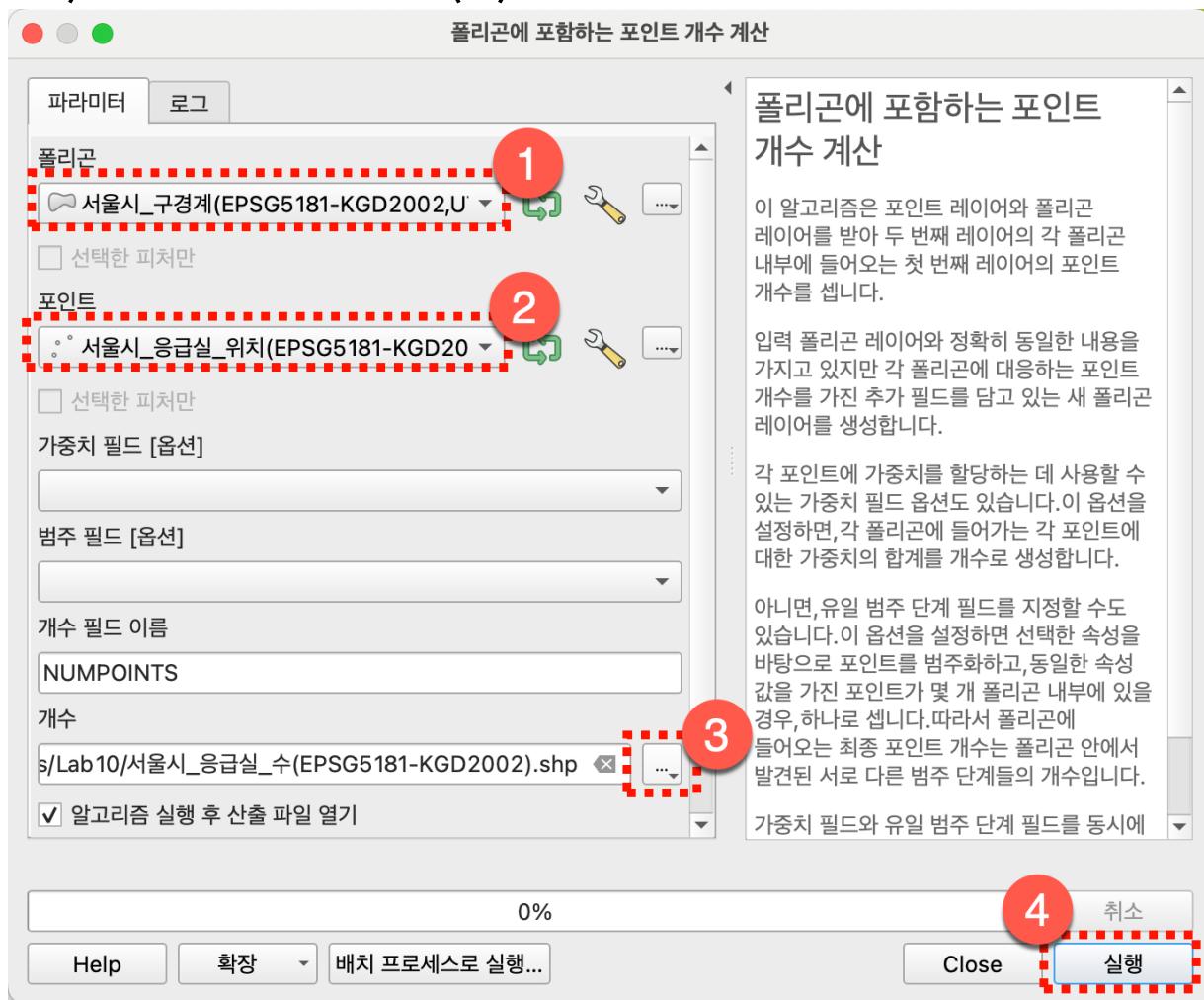
□ 폴리곤에 포함되는 포인트(응급실)의 수 계산하기 (2)

- ① 폴리곤 레이어 선택
→ 서울시 구 경계 지도
- ② 포인트 레이어 선택
→ 서울시 응급실 위치
- ③ 저장될 레이어 경로 지정
- ④ 설정 내용 실행



“서울시 응급실 수” 필드가 추가된 폴리곤 레이어가 생성된다.

이 레이어를 응급실 위치 포인트 레이어의 아래로 배치한다.



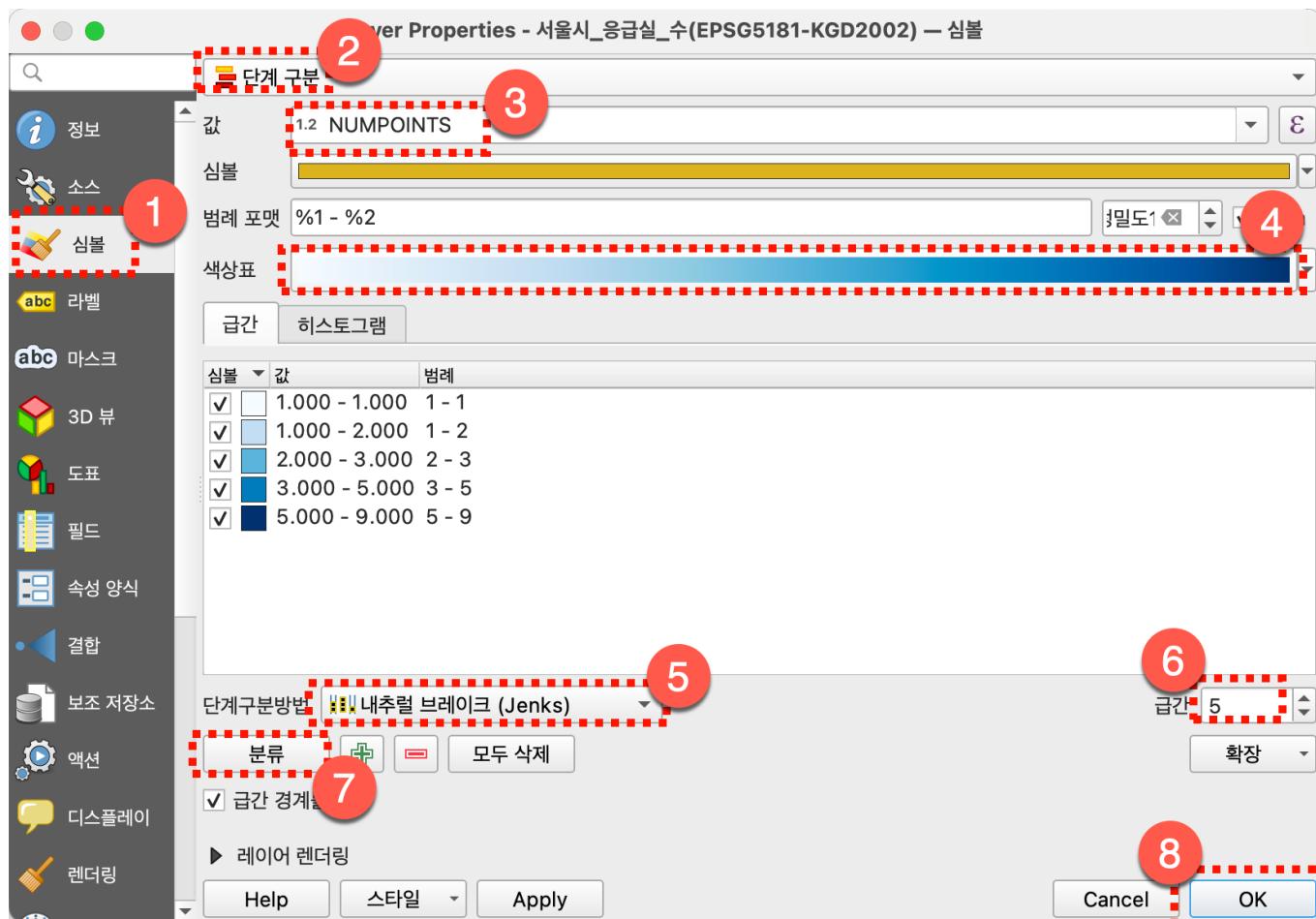


구 단위 응급실 분포 단계 구분도 만들기 (3)

1. 구글스프레드시트 활용
2. 브이월드 API 활용

□ 단계 구분 설정

- ◎ 새로 생성된 레이어에는 “NUMPOINTS”라는 필드가 생성된다.
- ◎ 이 필드는 폴리곤(서울시 구 경계)안에 포함되는 포인트(응급실)의 수를 저장하고 있다.
- ◎ 이 값을 사용하여 단계 구분도를 작성한다.





구 단위 응급실 분포 단계 구분도 만들기 - 결과 확인

□ 서울시 응급실 분포 및 위치 지도

1. 구글 스프레드시트 활용

2. 브이월드 API 활용



1. 구글스프레드시트 활용

2. 브이월드 API 활용

실습하기

브이월드 API 활용



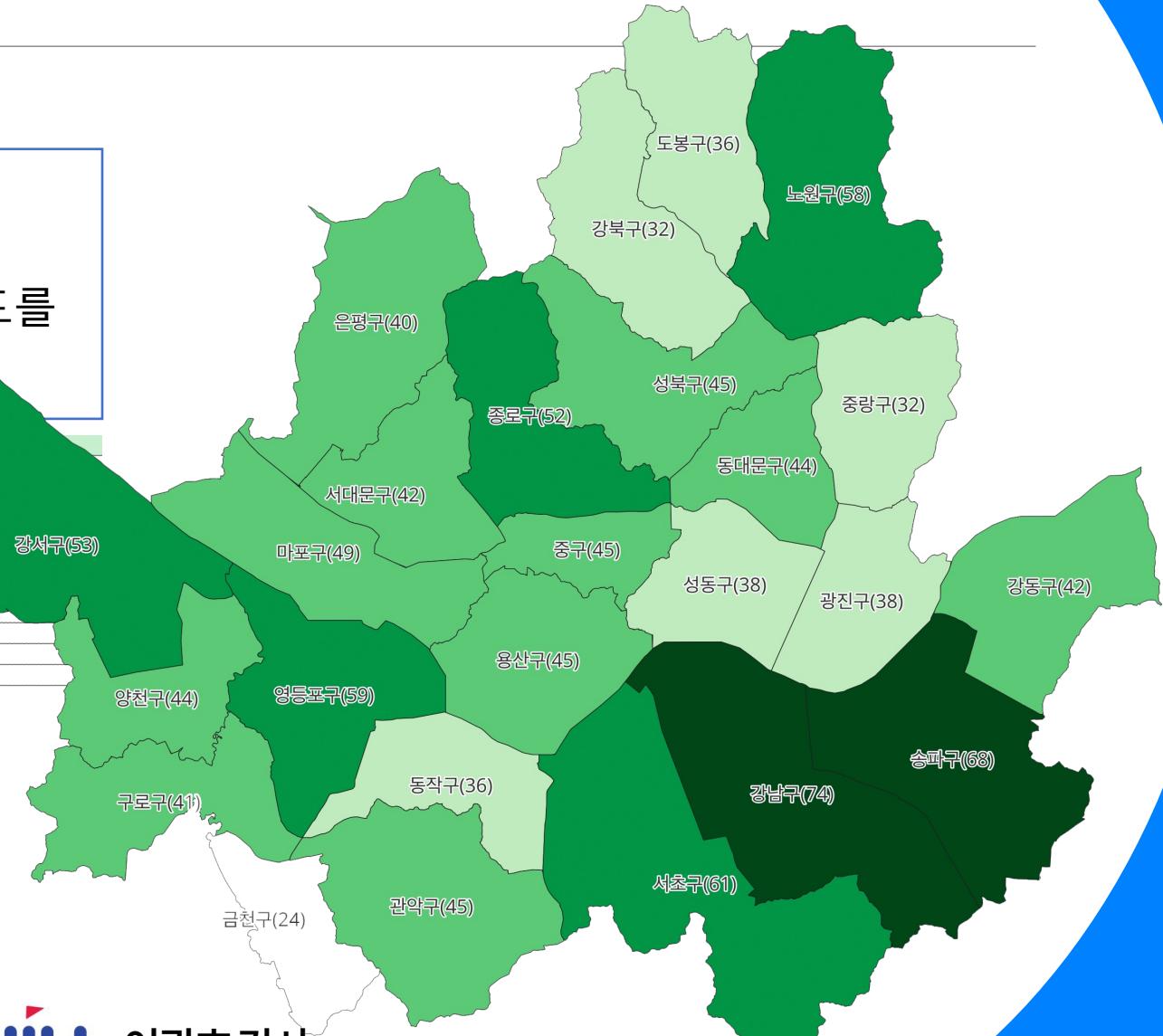


연구과제

□ 서울시 행정 기관 분포도

“서울_행정기관.xlsx”파일은 서울시에 위치한 행정기관의 주소를 담고 있다. 이 데이터를 사용하여 서울시의 구단위 행정기관의 분포도를 작성하시오.

기관유형	기관유형별분류	대표기관명	전체기관명
우정	우정_우체국	과학기술정보통신부	과학기술정보통신부 서울지방우정청 서울도봉우체국 서울우이동우편취급국
보훈	보훈부_국립묘지	국가보훈부	국가보훈부 국립4.19민주묘지관리소
지자체	8읍면동_동	서울특별시	서울특별시 강북구 우이동주민센터
지자체	8읍면동_동	서울특별시	서울특별시 강북구 인수동주민센터
지자체	소방_119_안전센터	서울특별시	서울특별시 강북소방서 우이119안전센터
우정	우정_우체국	과학기술정보통신부	과학기술정보통신부 서울지방우정청 서울도봉우체국 서울수유동우체국
지자체	8읍면동_동	서울특별시	서울특별시 강북구 수유2동주민센터
우정	우정_우체국	과학기술정보통신부	과학기술정보통신부 서울지방우정청 서울도봉우체국 서울수유3동우체국
지자체	8읍면동_동	서울특별시	서울특별시 강북구 번1동주민센터
지자체	5시군구_구	서울특별시	서울특별시 강북구청
지자체	8읍면동_동	서울특별시	서울특별시 강북구 수유3동주민센터
우정	우정_우체국	과학기술정보통신부	과학기술정보통신부 서울지방우정청 서울도봉우체국 서울인수동우편취급국





THANK YOU

수고하셨습니다.