

자바 플스텍&AI를 활용한 개발자 양성 취업과정(APS 플랫폼 개발)

Himedia IT 인재 개발원

2025.05.07 ~ 2025.10.31

# Team Project

환율 변동에 따른 외국인 여행객 증감 상관 관계

*Team.* 시앤파

황다니엘, 박시온, 이은지, 김민호

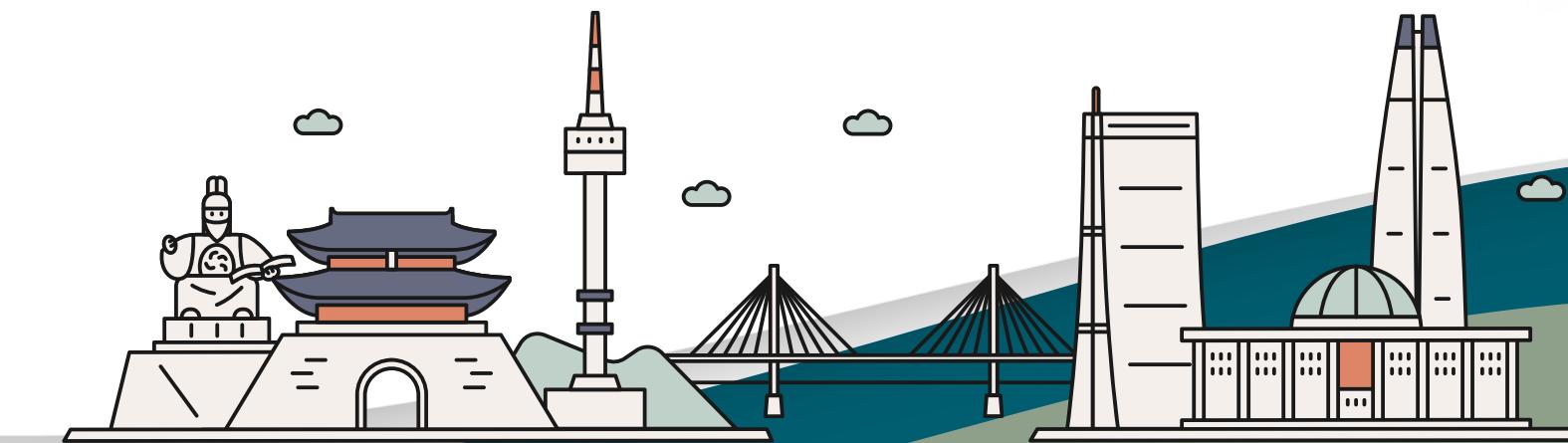
<https://github.com/ekdpf3636-jpg/fx-tourism-forecast>



고용노동부



KOREA TECH  
직업능력심사평가원





Himedia IT 인재 개발원

Since 2025

# 목차

- 01 프로젝트 개요**
- 02 프로젝트 팀 구성 및 역할**
- 03 프로젝트 수행 절차 및 방법**
- 04 프로젝트 수행 결과**
- 05 자체 평가 의견**



# 프로젝트 개요



01

## 프로젝트 주제 및 선정 배경, 기획의도

환율 변동으로 인한 방한 외국인 관광객 증감의 상관관계와 그에 따른 분석 및 예측

02

## 프로젝트 내용

과거 환율과 방한 외국인 여행객 입국 기록 공공 데이터 수집 후 환율을 고려하여 미래의 여행객 수를 예측하는 모델 구축 데이터 정제, 시각화, ARIMA 예측

03

## 활용 TOOL

- Python
- Colab
- Numpy
- Pandas
- Matplotlib
- ARIMA

04

## 프로젝트 구조

2025. 8/29 ~ 9/04

환율의 변동으로 인한 미래 방한 외국인 증감 예측 목적 데이터 수집 및 정제, 정규화 및 시각화, 예측 등

05

## 활용방안 및 기대효과

관광 산업이 환율 변동에 능동적으로 대응하고 효율적인 관광 정책 및 마케팅 전략을 수립하고 대비하여 방한 여행객 증가를 기대할 수 있습니다.



# 프로젝트 팀 구성 및 역할



황다니엘  
팀장

데이터 정제 및 정규화

ARIMA 예측 시각화

텍스트 마이닝



박시온  
팀원

외부 데이터 수집

데이터 시각화



이은지  
팀원

외부 데이터 수집

ppt 작성

데이터 시각화

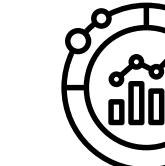
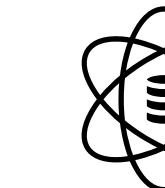
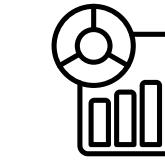


김민호  
팀원

데이터 정제 및 정규화

ARIMA 예측 시각화

# 프로젝트 수행 절차 및 방법

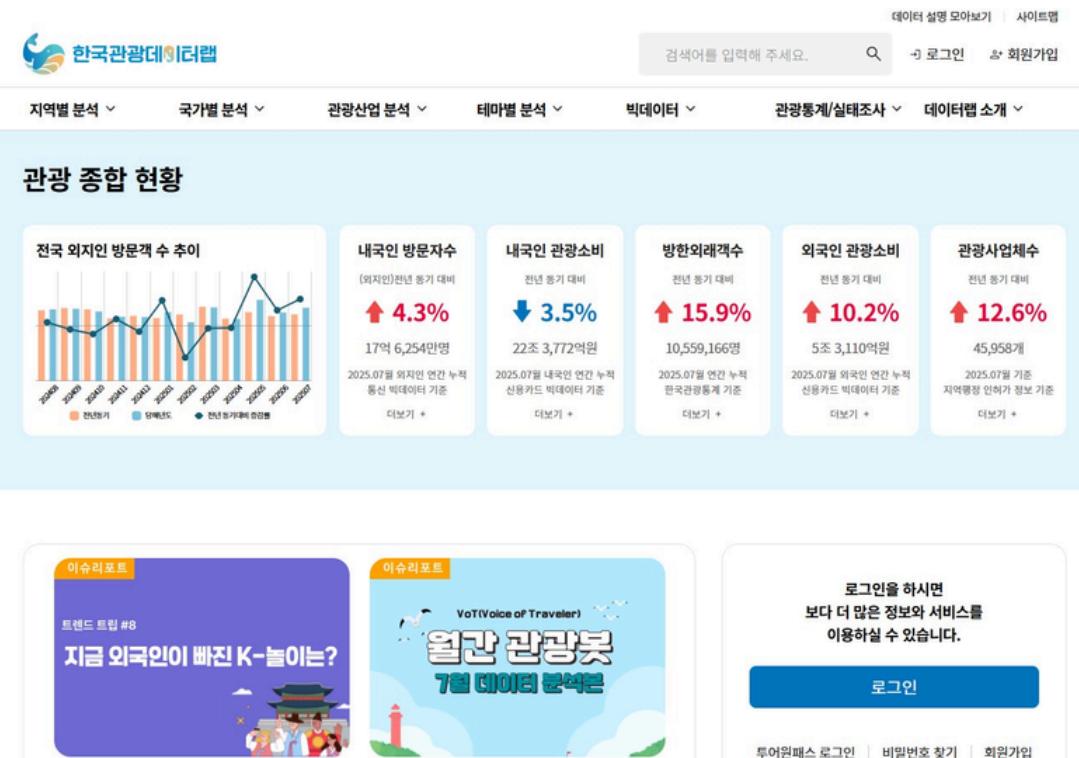
구분	기간	활동	비고
사전 기획	8월 29일 (금)	 프로젝트 기획 및 주제 선정 기획안 작성	아이디어 선정
데이터 수집	8월 29일 (금)~ 9월 1일 (월)	 필요 데이터 및 수집 절차 정의 외부 데이터 수집	공공데이터 수집
데이터 전처리	9월 2일 (화) ~ 9월 3일 (수)	 데이터 정제 및 정규화	최적화, 오류 수정
최종 데이터 분석	9월 4일 (목)	 전체 데이터 분석	
총 작업기간	8월 29일 (금) ~ 9월 4일 (목)		

# 프로젝트 수행 경과

- 01** 공공 데이터 수집
- 02** 여행객 입국 데이터 전처리
- 03** 여행객 데이터 시각화
- 04** 환율 데이터 전처리
- 05** 환율 데이터 시각화
- 06** 환율 & 여행객 데이터 병합
- 07** 환율 & 여행객 상관관계 그래프
- 08** ARIMA 예측 그래프



# 공공 데이터 수집



	관광객 입국 데이터 수집(박시온)	환율 데이터 수집(이은지)
URL	<b>한국 관광 데이터랩</b> <a href="https://datalab.visitkorea.or.kr/datalab/portal/main/getMainForm.do">https://datalab.visitkorea.or.kr/datalab/portal/main/getMainForm.do</a>	<b>서울외국환중개</b> <a href="http://www.smbs.biz/index.jsp">http://www.smbs.biz/index.jsp</a>
다운로드 경로	관광통계 - 방한 외래관광객 - 통계게시판	환율조회 - 월평균 매매기준율
파일 이름	전체국가통계_202507.xlsx 파일 다운	달러, 엔화, 유로 데이터 파일 다운

# 관광객 입국 데이터 전처리

(이은지)

중복된 한글 컬럼 및 여행객 타입 변환

01 관광객 입국 데이터 전처리 전

	A	B	C	D	E	F	G	H
1	[ 입 국 ]							
2								
3	Country	국가명	1984년	1985년	1986년	1987년	1988년	1989년
4	G.TOTAL		1,297,318	1,426,045	1,659,972	1,874,501	2,340,462	2,728,054
5								
6	Foreign Visitors		1,124,076	1,253,228	1,457,711	1,631,136	2,051,814	2,407,296
7	Overseas	교포	173,242	172,817	202,261	243,365	288,648	320,758
8								
9	ASIA		802,601	883,308	1,032,985	1,149,856	1,473,850	1,847,793
10								
11	Japan	일본	576,448	638,941	791,011	893,596	1,124,149	1,379,523
12	Taiwan	대만	93,543	99,622	94,799	110,373	124,185	156,530
13	Hong Kor	홍콩	46,587	47,110	55,315	54,030	62,298	68,504
14	Macao	마카오	15	20	23	35	28	22
15	Thailand	태국	10,107	10,527	11,313	10,225	13,835	19,535
<span>&lt; &gt;</span> <span>전체국가통계_202506</span> <span>+</span> <span>:</span> <span>◀ ▶</span>								



02 관광객 입국 데이터 전처리 후

	A	B	C	D	E
181	(China)	Jan-15	378816		
182	(China)	Feb-15	497369		
183	(China)	Mar-15	479713		
184	(China)	Apr-15	620406		
185	(China)	May-15	597030		
186	(China)	Jun-15	301083		
187	(China)	Jul-15	233997		
188	(China)	Aug-15	489506		
189	(China)	Sep-15	568322		
190	(China)	Oct-15	627276		
191	(China)	Nov-15	491509		
192	(China)	Dec-15	451459		
193	(China)	Jan-16	506201		
194	(China)	Feb-16	512777		
195	(China)	Mar-16	569848		

# 관광객 입국 데이터 전처리 코딩 과정

```
#문객 avg 전처리 (국가명 한 컬럼 제외)
import pandas as pd
import numpy as np

load_visitors_monthly_long(path):
    raw = pd.read_csv(path, encoding="cp949", header=0)
    header = raw.iloc[1].tolist()
    df = raw.iloc[2:1].copy()
    df.columns = header

    # 1) 'Country' 컬럼 표준화: 없으면 '국가명'을 'Country'로, 그것도 없으면 첫번째 컬럼을 'Country'
    if "Country" not in df.columns:
        if "국가명" in df.columns:
            df.rename(columns={"국가명": "Country"}, inplace=True)
        else:
            df.rename(columns={df.columns[0]: "Country"}, inplace=True)

    # 2) '국가명'이 여전히 존재하면 드롭
    if "국가명" in df.columns:
        df.drop(columns=["국가명"], inplace=True)

    # 3) 월 컬럼만 선별
    month_cols = [c for c in df.columns if isinstance(c, str) and ":" in c and "월" in c]

    # 반환시 "국가명" 컬럼 제외
    return long_df[["Country", "date", "visitors"]].sort_values(["Country", "date"]).reset_index(drop=True)

# csv 파일로 저장하기
vis_long = load_visitors_monthly_long("전체국가통계_202506.csv")
vis_long.to_csv("국가별_방문객통계.csv", index=False, encoding="cp949")
```



원본 데이터에 한글 컬럼과 G.TOTAL 등 필요 없는 데이터들이 포함되어있습니다 앞에 있을 환율 데이터와 병합을 위해서 Country 컬럼을 Long 형태로 변환 하고, 국가 마다 각 년도 여행객 수를 보기 쉽게 컬럼을 Country, date, visitors 로 정리 했습니다.



```
# 4) Country + 월 컬럼만 사용
dfm = df[["Country"] + month_cols].copy()
dfm["Country"] = dfm["Country"].ffill()

# 5) Long 형태로 변환
long_df = dfm.melt(id_vars=["Country"], var_name="ym_str", value_name="visitors")

# 6) 방문객 숫자 정리
long_df["visitors"] = (long_df["visitors"].astype(str)
                        .str.replace(".", "", regex=False)
                        .str.strip())
long_df["visitors"] = pd.to_numeric(long_df["visitors"], errors="coerce")

# 7) "YYYY.월" → Timestamp(YYYY-MM-01)
def parse_year_month(s):
    try:
        y, m = s.split(".")
        y = int(y); m = int(m.replace("월", "").strip())
        return pd.Timestamp(y, m, 1)
    except:
        return pd.NaT

long_df["date"] = long_df["ym_str"].apply(parse_year_month)
long_df = long_df.dropna(subset=["date"]).drop(columns=["ym_str"])
long_df["date"] = pd.to_datetime(long_df["date"]).dt.strftime("%Y-%m")
```



# 국가별 여행객 데이터 시각화 & 코딩 (박시온)

## 외국인 여행객 TOP 20 국가 막대 그래프

```
# 필수 라이브러리 설치 및 임포트
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf

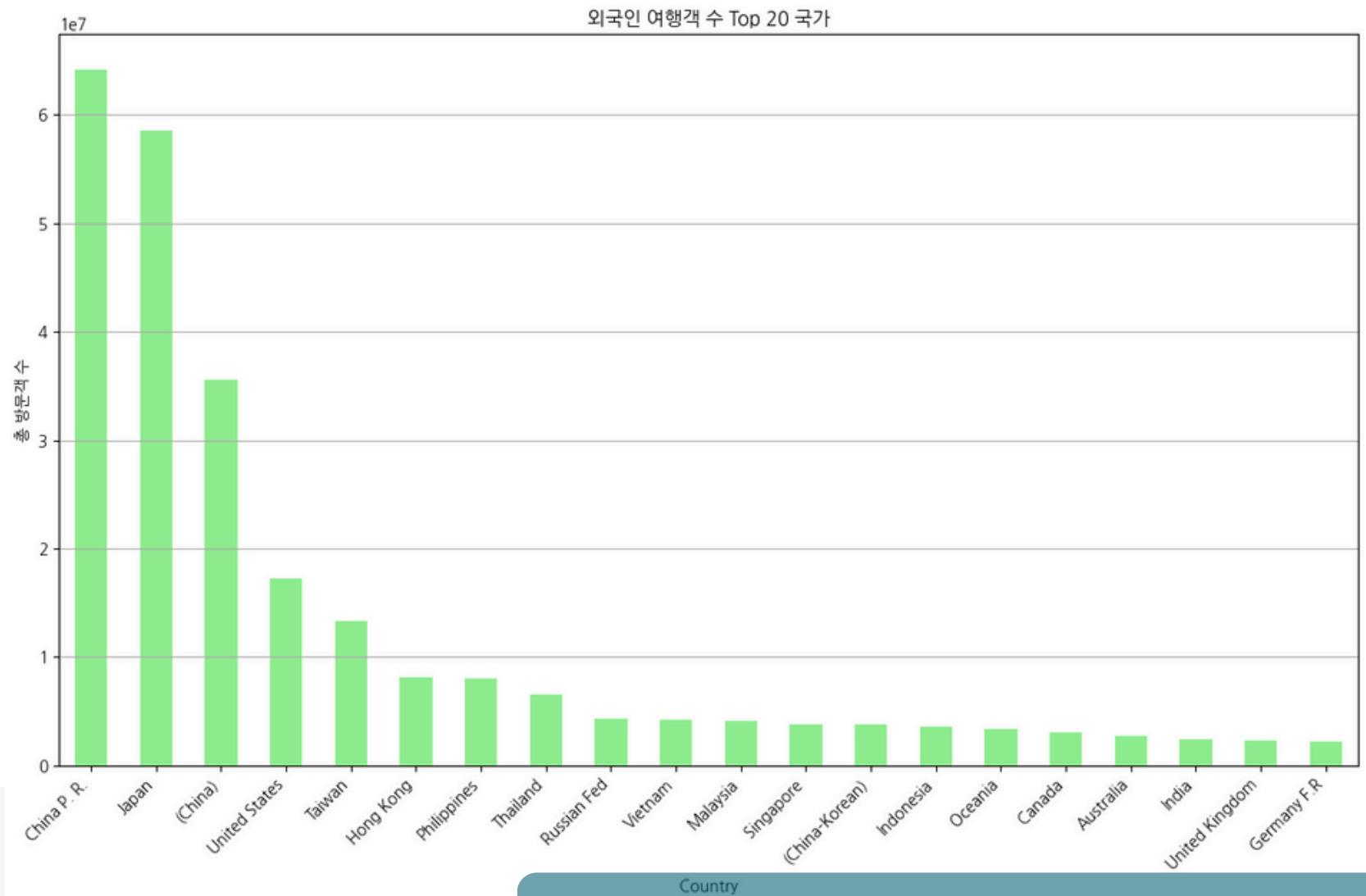
# 폰트 설정
plt.rc('font', family='NanumGothic')
# 마이너스 부호 깨짐 방지
plt.rcParams['axes.unicode_minus'] = False

# 파일 읽기
df = pd.read_csv("환율대비_여행객수.csv", encoding="utf-8")

# 전처리: 여행객 수 결측치 제거
df_clean = df.dropna(subset=['visitors'])
# 집계용 이름 제거 (실제 국가만 필터링)
exclude_keywords = ['TOTAL', 'Foreign', 'ASIA', 'Europe', 'Americas', 'Overseas']
real_countries_df = df_clean[~df_clean['Country'].str.contains('|'.join(exclude_keywords), case=False)]
# 'Hong Kong, China(including Hong Kong ID.)' 국가명을 'Hong Kong'으로 변경
real_countries_df['Country'] = real_countries_df['Country'].replace(
    'Hong Kong, China(including Hong Kong ID.)', 'Hong Kong'
)

# 국가별 여행객 합계
top20_real_countries = real_countries_df.groupby('Country')[['visitors']].sum().sort_values(ascending=False).head(20)

# 시각화
plt.figure(figsize=(12, 8))
top20_real_countries.plot(kind='bar', color='lightgreen')
plt.title('외국인 여행객 수 Top 20 국가')
plt.ylabel('총 여행객 수')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



앞서 병합한 데이터를 기반으로 여행객 수 TOP 20을 선정해 시각화한 결과. 'Asia', 'Total', 'Foreign'처럼 국가가 아닌 항목은 제외하고, 실제 국가명 기준으로 필터링한 것이 특징입니다. 결과를 통해 어떤 국가가 한국을 많이 방문한지 알 수 있고, 특정 국가의 비중이 높다면 그 국가와 문화 교류, 마케팅 전략에도 참고할 수 있습니다.



# 국가별 여행객 데이터 시각화 & 코딩 과정 (김민호)

## 외국인 여행객 TOP 20 국가 산점도 그래프

```

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# CSV 불러오기
df = pd.read_csv("여행객수_환율추이_그래프.csv", encoding="cp949")

# 날짜 처리
df['date'] = pd.to_datetime(df['date'])
df['year'] = df['date'].dt.year # 연도 추출
df = df[df['Country'] != 'G.TOTAL']

# 연도별 국가 평균 방문객 수 계산
df_grouped = df.groupby(['Country', 'year'])['visitors'].mean().reset_index()

# 상위 20개 국가만 선택 (총 방문객 수 기준)
top_countries = df_grouped.groupby('Country')['visitors'].sum().sort_values(ascending=False).head(20).index
df_top = df_grouped[df_grouped['Country'].isin(top_countries)].copy()

# 산점도
plt.figure(figsize=(15,8))

# y축 숫자로 변환 + jitter 적용
y_pos = df_top['Country'].astype('category').cat.codes
y_jitter = y_pos + np.random.uniform(-0.2, 0.2, size=len(y_pos))

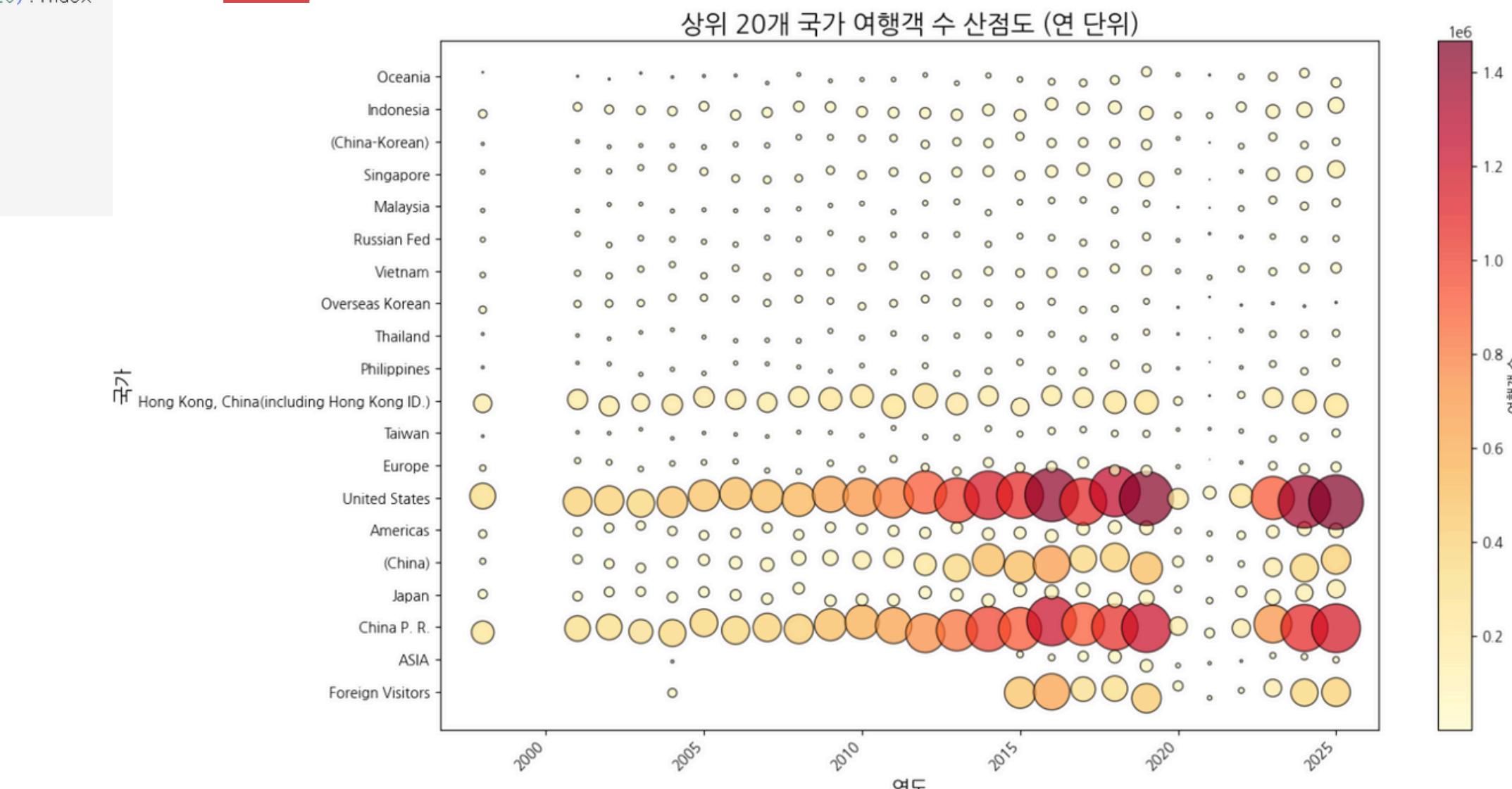
scatter = plt.scatter(
    x=df_top['year'],
    y=y_jitter,
    s=df_top['visitors']/1000, # 점 크기
    c=df_top['visitors'], # 점 색상
    cmap='YlOrRd',
    alpha=0.7,
    edgecolors='k'
)

plt.yticks(ticks=np.arange(len(top_countries)), labels=top_countries)
plt.colorbar(scatter, label='여행객 수')
plt.xlabel("연도", fontsize=14)
plt.ylabel("국가", fontsize=14)
plt.title("상위 20개 국가 여행객 수 산점도 (연 단위)", fontsize=18)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(False)
plt.show()

```



여행객과 환율의 “date” 컬럼으로 병합한 파일로 제작됐고,  
다차원 데이터 (국가, 연도, 여행객 수)를 산점도 그래프로 시각화 해,  
복잡한 데이터를 쉽게 분석할 수 있습니다.  
원색상이 짙을수록 여행객 수 증가를 나타내고.  
해당 국가에 대한 마케팅, 문화적 교류 계획 수립을 할 수 있게 돋는  
기초 분석 산점도 그래프입니다.



# 환율 데이터 전처리

(황다니엘)

매매 컬럼 제거, 날짜 형식 변환 및 환율 별 원화 기준으로 전처리 후 병합

## 01 환율 데이터 전처리 전

	A	B	C		A	B	C		A	B	C	
1	월평균 매매기준율				1	월평균 매매기준율			1	월평균 매매기준율		
2	기간 : 1999년 01월 ~ 2025년 08월				2	기간 : 1999년 01월 ~ 2025년 08월			2	기간 : 1999년 01월 ~ 2025년 08월		
3	날짜	통화명	원		3	날짜	통화명	원	3	날짜	통화명	원
4	1999.01	미국 달러	1,176.94		4	1999.01	일본 엔 (J)	1,039.18	4	1999.01	유로 (EUR)	1,365.12
5	1999.02	미국 달러	1,186.81		5	1999.02	일본 엔 (J)	1,018.47	5	1999.02	유로 (EUR)	1,330.04
6	1999.03	미국 달러	1,229.16		6	1999.03	일본 엔 (J)	1,027.55	6	1999.03	유로 (EUR)	1,337.48
7	1999.04	미국 달러	1,208.94		7	1999.04	일본 엔 (J)	1,009.87	7	1999.04	유로 (EUR)	1,295.03
8	1999.05	미국 달러	1,197.00		8	1999.05	일본 엔 (J)	982.07	8	1999.05	유로 (EUR)	1,272.64
9	1999.06	미국 달러	1,169.63		9	1999.06	일본 엔 (J)	968.42	9	1999.06	유로 (EUR)	1,215.38
10	1999.07	미국 달러	1,186.04		10	1999.07	일본 엔 (J)	991.98	10	1999.07	유로 (EUR)	1,229.09
11	1999.08	미국 달러	1,199.79		11	1999.08	일본 엔 (J)	1,057.77	11	1999.08	유로 (EUR)	1,272.34
12	1999.09	미국 달러	1,196.97		12	1999.09	일본 엔 (J)	1,111.46	12	1999.09	유로 (EUR)	1,256.01
13	1999.1	미국 달러	1,206.38		13	1999.1	일본 엔 (J)	1,138.81	13	1999.1	유로 (EUR)	1,292.47



## 02 환율 데이터 전처리 후

	A	B	C	D
1	date	usd_krw	JPY100_krw	eur_krw
2	Jan-99	1176.94	1039.18	1365.12
3	Feb-99	1186.81	1018.47	1330.04
4	Mar-99	1229.16	1027.55	1337.48
5	Apr-99	1208.94	1009.87	1295.03
6	May-99	1197	982.07	1272.64
7	Jun-99	1169.63	968.42	1215.38
8	Jul-99	1186.04	991.98	1229.09
9	Aug-99	1199.79	1057.77	1272.34
10	Sep-99	1196.97	1111.46	1256.01
11	Oct-99	1206.38	1138.81	1292.47
12	Nov-99	1177.22	1123.31	1217.82
13	Dec-99	1138.39	1109.08	1151.7

# 환율 데이터 전처리 코딩 과정

환율 원본 데이터 USD, JPY, EUR로 세 파일을 받았습니다.  
각 파일 데이터의 필요 없는 컬럼을 정리하고,  
날짜 데이터 형식을 YYYY-MM로 변환해서  
년도 별 USD, JPY, EUR를 원화 기준 환율로  
보기 쉽게 정리한 후 세 개의 환율 데이터를 병합했습니다.



```
# 환율 데이터 전처리 (date 컬럼을 맨 앞으로)
import pandas as pd

def load_fx_one(path, currency_tag):
    raw = pd.read_csv(path, encoding="cp949", header=0)
    header = raw.iloc[1].tolist()
    df = raw.iloc[2:].copy()
    df.columns = header

    col_date = [c for c in df.columns if "날짜" in str(c)][0]
    col_won = [c for c in df.columns if str(c).strip().endswith("원")][-1]

    out = df[[col_date, col_won]].copy()
    out.rename(columns={col_date: "ym", col_won: currency_tag}, inplace=True)

    out[currency_tag] = (out[currency_tag].astype(str)
                           .str.replace(".", "", regex=False)
                           .str.strip())
    out[currency_tag] = pd.to_numeric(out[currency_tag], errors="coerce")

    def parse_ym(s):
        try:
            y, m = s.split(".")
            return pd.Timestamp(int(y), int(m), 1)
        except:
            return pd.NaT

```

```
# 날짜 형식
out["date"] = out["ym"].apply(parse_ym)
out = out.dropna(subset=["date"]).drop(columns=["ym"]).sort_values("date")
# ✓ 날짜 형식을 YYYY-MM으로 통일 (← 여기만 바뀜)
out["date"] = out["date"].dt.strftime("%Y-%m")
return out

# 통화별 로드
fx_usd = load_fx_one("달러.csv", "usd_krw")
fx_jpy = load_fx_one("엔화.csv", "jpy100_krw") # 100엔 기준
fx_eur = load_fx_one("유로.csv", "eur_krw")

# 병합
fx_all = (
    fx_usd
    .merge(fx_jpy, on="date", how="outer")
    .merge(fx_eur, on="date", how="outer")
    .sort_values("date"))

# 안전하게 date를 맨 앞으로 정렬
fx_all = fx_all[[["date"] + [c for c in fx_all.columns if c != "date"]]]
# 문자열 YYYY-MM로 확정 -> csv 파일에 day 출력 X
fx_all["date"] = fx_all["date"].astype(str)

# 저장 & 다운로드
fx_all.to_csv("달러_엔_유로_원화환율.csv", index=False, encoding="cp949")
```

# 환율 데이터 시각화 코딩 과정 & 코딩 (김민호)

USD, JPY, EUR 대비 원화 환율 변동 추이를 보여주고,  
시계열 데이터로 환율이 어떻게 변했는지 한눈에 확인 가능하게 했으며,  
외국인 여행객 수 데이터와 상관 관계 분석을 위한 기초가 됩니다.

```
import matplotlib.pyplot as plt
import pandas as pd

dt = pd.read_csv('환율대비_여행객수.csv')

# 날짜는 datetime 그대로 유지 (문자열로 바꾸지 않음!)
dt['date'] = pd.to_datetime(dt['date'])

plt.figure(figsize=(20, 10))
plt.plot(dt['date'], dt['usd_krw'], label='USD/KRW 환율', color='green')
plt.plot(dt['date'], dt['jpy100_krw'], label='JPY/KRW 환율', color='red')
plt.plot(dt['date'], dt['eur_krw'], label='EUR/KRW 환율', color='darkblue')

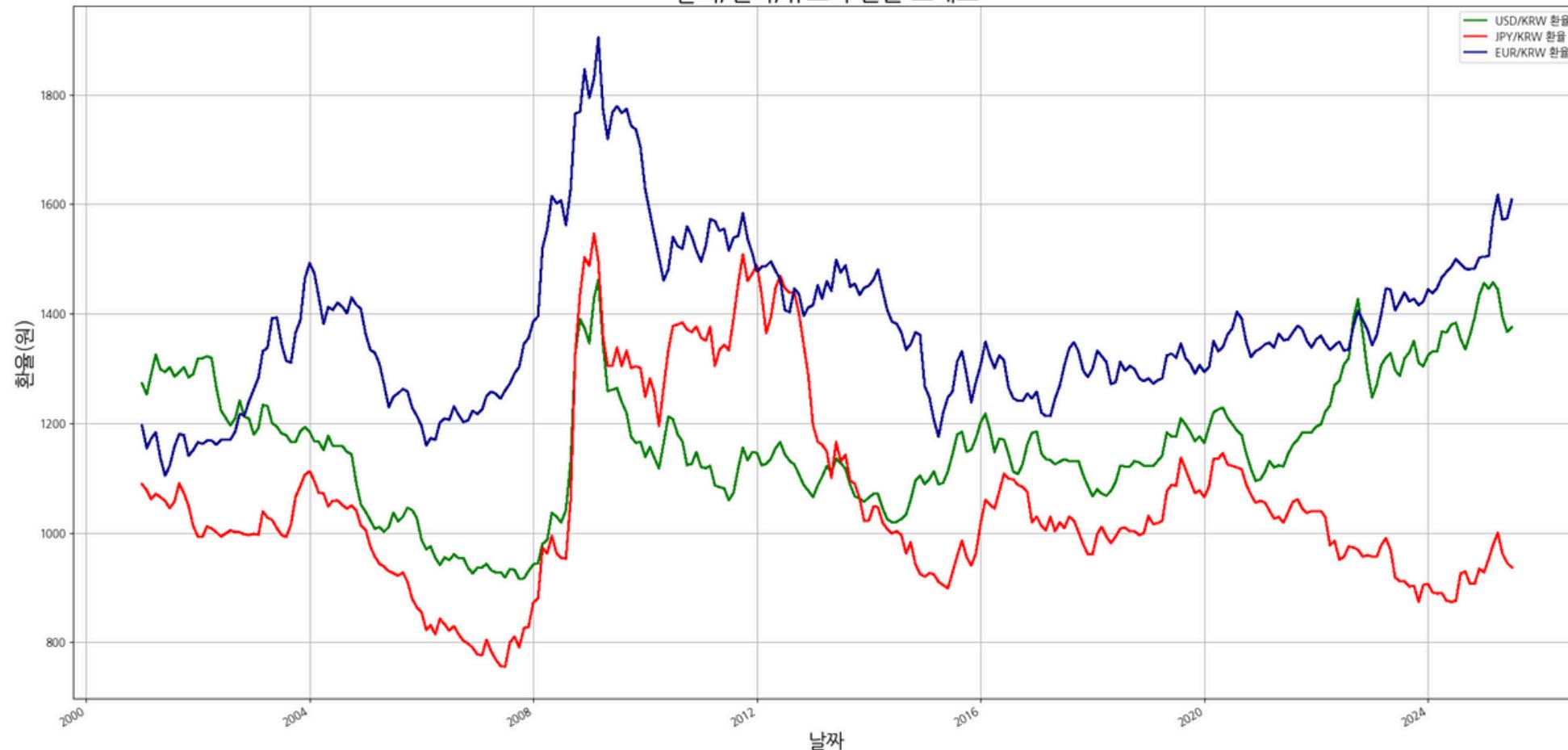
# x축 날짜 포맷팅 자동 처리
plt.gcf().autofmt_xdate() # x축 날짜 회전 + 간격 자동 조정

plt.xlabel('날짜', fontsize=18)
plt.ylabel('환율(원)', fontsize=18)
plt.title('달러/엔화/유로화 환율 그래프', fontsize=22)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

dt = dt.to_csv('여행객수_환율추이_그래프.csv', encoding='cp949')
```



달러/엔화/유로화 환율 그래프



# 환율 & 여행객 데이터 병합 (황다니엘)

여행객 수 & 환율을 date 컬럼에 맞춰 정렬하여 데이터 분석과 시각화에 용이하게 변환 후 병합.

A	B	C	D	E
hina)	Jan-15	378816		
hina)	Feb-15	497369		
hina)	Mar-15	479713		
hina)	Apr-15	620406		
hina)	May-15	597030		
hina)	Jun-15	301083		
hina)	Jul-15	233997		
hina)	Aug-15	489506		
hina)	Sep-15	568322		
hina)	Oct-15	627276		
hina)	Nov-15	491509		
hina)	Dec-15	451459		
hina)	Jan-16	506201		
hina)	Feb-16	512777		
hina)	Mar-16	569848		
hina)	Apr-16	661453		
hina)	May-16	684047		
hina)	Jun-16	738835		
hina)	Jul-16	894643		
hina)	Aug-16	850468		
hina)	Sep-16	699301		
hina)	Oct-16	658252		



A	B	C	D	E
1	date	usd_krw	jpy100_kn	eur_krw
2	Jan-99	1176.94	1039.18	1365.12
3	Feb-99	1186.81	1018.47	1330.04
4	Mar-99	1229.16	1027.55	1337.48
5	Apr-99	1208.94	1009.87	1295.03
6	May-99	1197	982.07	1272.64
7	Jun-99	1169.63	968.42	1215.38
8	Jul-99	1186.04	991.98	1229.09
9	Aug-99	1199.79	1057.77	1272.34
10	Sep-99	1196.97	1111.46	1256.01
11	Oct-99	1206.38	1138.81	1292.47
12	Nov-99	1177.22	1123.31	1217.82
13	Dec-99	1138.39	1109.08	1151.7
14	Jan-00	1131.07	1074.52	1146.91
15	Feb-00	1128.8	1030.78	1111.41
16	Mar-00	1117.19	1048.11	1078.11
17	Apr-00	1109.76	1052.99	1049.39
18	May-00	1120.01	1036.5	1017.59
19	Jun-00	1118.73	1053.17	1061.78



A	B	C	D	E	F
1	Country	date	visitors	usd_krw	jpy100_kn eur_krw
2	(China)	Jan-01		1272.82	1088.93 1196.42
3	(China)	Feb-01		1252.44	1078.01 1154.53
4	(China)	Mar-01		1288.43	1061.62 1171.67
5	(China)	Apr-01		1325.55	1071 1183.73
6	(China)	May-01		1298.46	1065.38 1137.12
7	(China)	Jun-01		1293.83	1058.2 1104.1
8	(China)	Jul-01		1302.6	1044.96 1122.27
9	(China)	Aug-01		1285.39	1057.22 1157.41
10	(China)	Sep-01		1293.7	1090.14 1180.07
11	(China)	Oct-01		1302.6	1073.11 1178.9
12	(China)	Nov-01		1284	1049.55 1140.69
13	(China)	Dec-01		1289.66	1013.12 1150.62
14	(China)	Jan-02		1317.6	993.41 1165.5
15	(China)	Feb-02		1318.72	992.71 1162.21
16	(China)	Mar-02		1322.51	1011.85 1168.21
17	(China)	Apr-02		1318.93	1007.97 1168.21

병합 코딩 →

# 여행객/환율 병합

```
vis_fx = vis_long.merge(fx_all, on="date", how="left")
vis_fx.to_csv("환율대비_여행객수.csv", index=False, encoding="utf-8")
```

# 환율 & 여행객 상관 관계 그래프 & 코딩

(김민호)

```

import matplotlib.pyplot as plt
import pandas as pd

# 폰트 설정
plt.rc('font', family='NanumGothic')
plt.rcParams['axes.unicode_minus'] = False
# CSV 불러오기
dt = pd.read_csv('환율대비_여행객수.csv')

dt['date'] = pd.to_datetime(dt['date'])

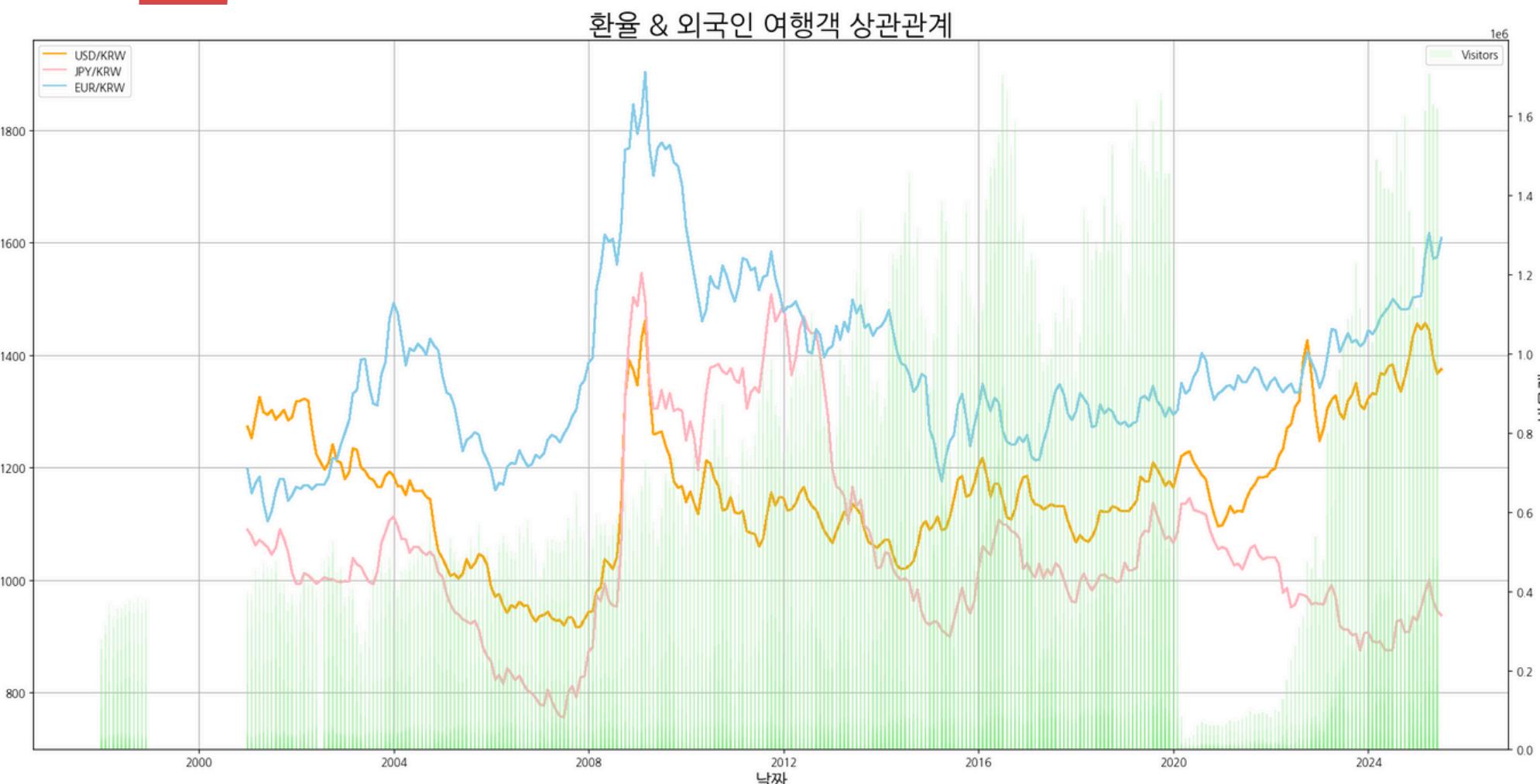
fig, ax1 = plt.subplots(figsize=(20, 10))

# 환율 선 그래프 (왼쪽 y축)
ax1.plot(dt['date'], dt['usd_krw'], label='USD/KRW', color='orange')
ax1.plot(dt['date'], dt['jpy100_krw'], label='JPY/KRW', color='lightpink')
ax1.plot(dt['date'], dt['eur_krw'], label='EUR/KRW', color='skyblue')
ax1.set_xlabel('날짜', fontsize=16)
ax1.set_ylabel('환율(원)', fontsize=16)
ax1.tick_params(axis='y')
ax1.legend(loc='upper left')
ax1.grid(True)

# 방문객 수 막대그래프 (오른쪽 y축)
ax2 = ax1.twinx()
ax2.bar(dt['date'], dt['visitors'], color='lightgreen', alpha=0.15, label='Visitors', width=15)
ax2.set_ylabel('방문객', fontsize=16)
ax2.tick_params(axis='y')
ax2.legend(loc='upper right')
plt.title('환율 & 외국인 여행객 상관관계', fontsize=25)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

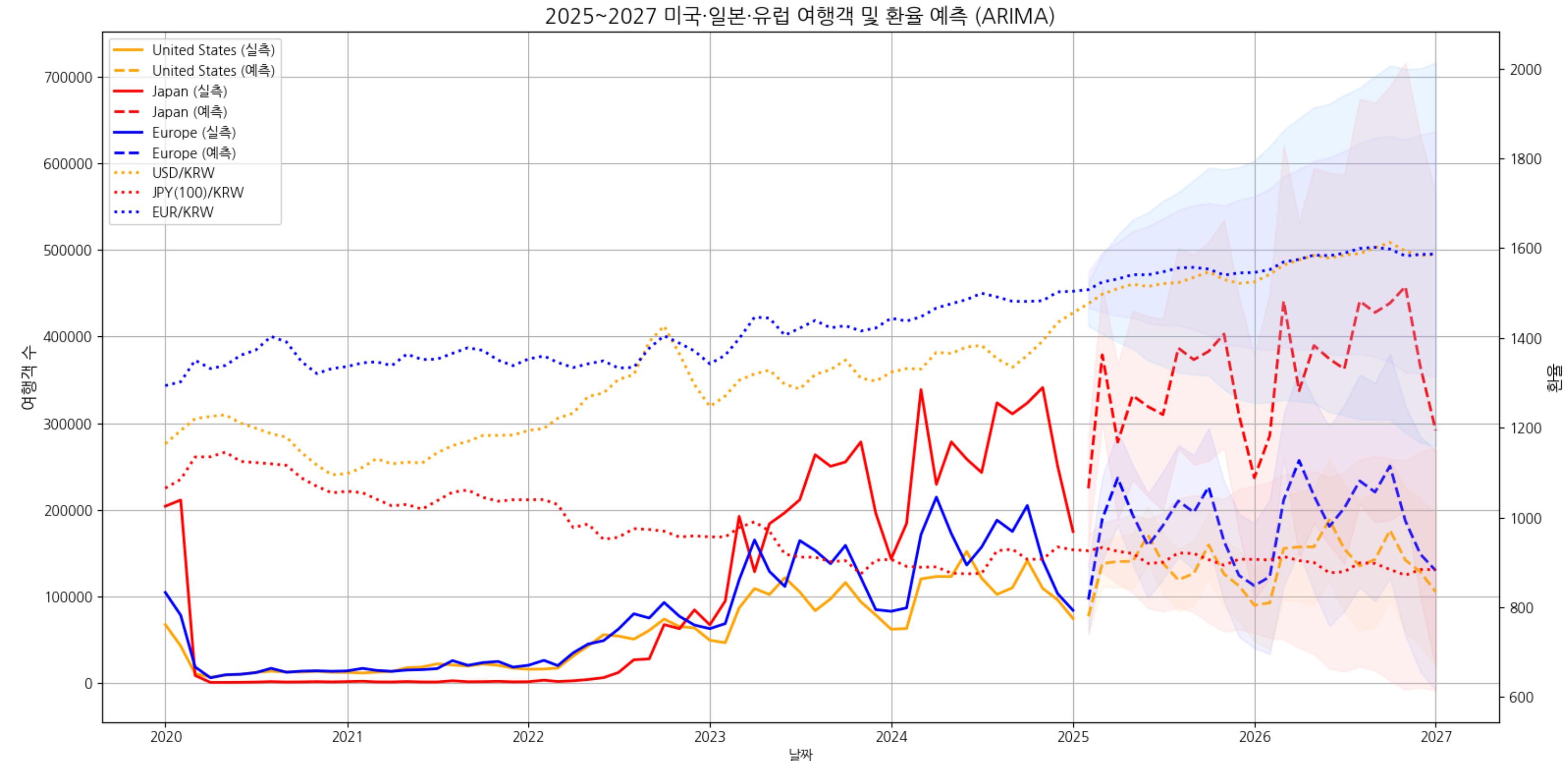
```

환율 변동에 따른 여행객 증감 상관 관계를 확인하기 위한 그래프입니다.  
 환율이 낮아지면 (원화 강세) 여행객은 경비가 높아져 방문이 감소하고,  
 반대로 환율이 높아지면(원화 약세) 여행 경비가 상대적으로 낮아져  
 증가 효과를 기대할 수 있습니다.



# ARIMA 시각화 환율 & 여행객 예측 그래프

(황다니엘)



# ARIMA 예측 분석

```
# ----- 결과 -----
# 왼쪽 y축(방문객) + 오른쪽 y축(환율) 모두 수집
lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines1 + lines2, labels1 + labels2, loc="upper left",
            fontsize=10)
# 전체 레이아웃 정리 및 출력
plt.tight_layout()
plt.show()
```



2025.02 ~ 2027.01 대상으로, 경제 규모가 큰 미국·일본·유럽 국가에서  
오는 외국인 여행객 수와 해당 국가 환율 변동을  
\*ARIMA\*를 활용해 예측 결과를 시각화 한 것입니다

- 실선 : 실제 데이터(2020.01~2025.01) 의미,
- 점선 : ARIMA 예측 값(2025.02~2027.01) 시각화.
- 색상 : 국가별 구분, 각 국가 예측 구간은 밝은 음영으로 신뢰 구간 표시.  
불확실성 범위도 함께 제시했습니다.
- 오른쪽 축 : 환율 구분. USD/KRW, JPY(100)/KRW, EUR/KRW  
환율도 함께 예측, 점선 그래프 음영으로 미래 환율 변동성  
과 추세를 시각화 했습니다.

예측 그래프는 단순히 여행객 수만 보는 게 아닌, 환율과의 관계성 까지  
고려하여 여행 수요에 어떤 영향을 줄 수 있는지 알아보는  
통합적 분석 시도입니다.

-- 예측 그래프를 통해 알 수 있는 정보 --  
원화 대비 달러나 유로가 강세를 보이면  
해외에서 한국 여행 경비가 상대적으로 감소해서  
여행객 수가 늘어날 가능성이 높고. 또한 시계열 분석은 관광 정책 수립,  
마케팅 전략, 환율 리스크 관리 측면에서  
실질적인 참고 자료로 활용될 수 있습니다.

# ARIMA 시각화 환율 & 여행객 예측 그래프 코딩

```

# ----- 여행객수 시각화 -----
fig, ax1 = plt.subplots(figsize=(16, 8))
colors = {"United States": "orange", "Japan": "red", "Europe": "blue"}
# 여행객수 시각화
for country in target_countries:

    data = results[country] # 해당 국가 예측 결과 호출
    actual = data[data["type"] == "actual"] # 실측 데이터
    forecast = data[data["type"] == "forecast"] # 예측 데이터
    c = colors[country]
    # 실측 값 그래프 (실선)
    ax1.plot(actual["date"], actual["value"], label=f"{country} (실측)",
             color=c, linewidth=2)
    # 예측 값 그래프 (점선)
    ax1.plot(forecast["date"], forecast["value"], label=f"{country} (예측)",
             linestyle="--", color=c, linewidth=2)
    # 예측 신뢰구간 시각화
    ax1.fill_between(forecast["date"], forecast["mean_ci_lower"],
                     forecast["mean_ci_upper"], color=c, alpha=0.05)

# y축(왼쪽): 여행객 수
ax1.set_ylabel("여행객 수", fontsize=12)
ax1.set_xlabel("날짜")
ax1.tick_params(axis='y')
ax1.set_title("2025~2027 미국 · 일본 · 유럽 여행객 및 환율 예측 (ARIMA)",
              fontsize=15)
ax1.grid(True)

```

```

# 유럽 국가 뮤기
europe_countries = [
    "United Kingdom", "Germany", "France", "Italy", "Spain",
    "Netherlands", "Sweden", "Norway", "Denmark", "Finland",
    "Switzerland", "Austria", "Belgium", "Ireland", "Greece",
    "Portugal", "Poland", "Czech Republic", "Hungary"
]
# df에서 유럽 국가에 해당하는 데이터만 추출
df_europe = df[df["Country"].isin(europe_countries)].copy()

# 유럽 국가들의 월별 방문객 수를 모두 합쳐서 하나의 "Europe"으로 그룹핑
df_europe_grouped = df_europe.groupby("date")["visitors"].sum().reset_index()

# 새롭게 만든 그룹에 국가명을 "Europe"으로 지정
df_europe_grouped["Country"] = "Europe"

# 유럽 외의 다른 국가 데이터는 그대로 사용
df_non_europe = df[~df["Country"].isin(europe_countries)]
# 유럽 그룹 데이터와 나머지 국가 데이터를 합침
df_final = pd.concat([df_non_europe, df_europe_grouped], ignore_index=True)
# Country, date 기준으로 정렬 후 인덱스 리셋
df_final = df_final.sort_values(["Country", "date"]).reset_index(drop=True)

# 환율 데이터 준비
fx_cols = ["usd_krw", "jpy100_krw", "eur_krw"]
fx_df = df.groupby("date")[fx_cols].mean().reset_index()
fx_df["date"] = pd.to_datetime(fx_df["date"])
fx_df.set_index("date", inplace=True)

# ----- 예측 설정 -----
# 예측 대상 국가 리스트 설정
target_countries = ["United States", "Japan", "Europe"]

# 학습 데이터의 종료일 (실제 데이터는 2025년 1월까지 사용)
cutoff_date = pd.to_datetime("2025-01-01")

# 예측 기간: 이후 24개월 (2025년 2월 ~ 2027년 1월)
forecast_periods = 24
future_dates = pd.date_range(start=cutoff_date + pd.DateOffset(months=1),
                             periods=forecast_periods, freq="MS") # 월별 시작일

# 결과 저장용 딕셔너리
results = {}

```

# ARIMA 시각화 환율 & 여행객 예측 그래프 코딩

```

# 각 국가별로 예측 수행
for country in target_countries:
    # 해당 국가 데이터만 필터링
    country_df = df_final[df_final["Country"] == country]
    # 날짜별 여행객 수 집계
    country_df = country_df.groupby("date")["visitors"].sum().reset_index()
    # 날짜를 인덱스로 설정 (시계열로 만들기 위해)
    country_df.set_index("date", inplace=True)
    # 학습용 데이터: 2020년 1월부터 cutoff_date까지
    train = country_df.loc["2020-01-01":cutoff_date]

    # ARIMA 모델 정의 및 학습
    model = SARIMAX(train, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
    result = model.fit(disp=False)

    # 24개월간 예측 수행
    forecast = result.get_forecast(steps=forecast_periods)
    forecast_df = forecast.summary_frame()

    # 예측 결과에서 필요한 컬럼만 숫자로 변환
    forecast_df = forecast_df[["mean", "mean_ci_lower", "mean_ci_upper"]].apply(
        pd.to_numeric, errors="coerce")
    # 국가명 및 날짜 추가
    forecast_df["country"] = country
    forecast_df["date"] = future_dates

```

```

# 실측 데이터 전처리: 열 이름 변경 및 태그 추가
train = train.reset_index()
train["type"] = "actual"
train["country"] = country
train.rename(columns={"visitors": "value"}, inplace=True)

# 예측 데이터 전처리: 열 이름 및 태그
forecast_df.rename(columns={"mean": "value"}, inplace=True)
forecast_df["type"] = "forecast"

# 실측과 예측 데이터를 하나로 병합
combined = pd.concat([
    train[["date", "value", "country", "type"]],
    forecast_df[["date", "value", "country", "type", "mean_ci_lower",
                 "mean_ci_upper"]]
], ignore_index=True)

# 국가별로 결과 저장
results[country] = combined

```

# ARIMA 시각화 환율 & 여행객 예측 그래프 코딩

```

# ----- 환율 예측 -----
# 각 통화별 예측 구간 색상 설정
color_map = {"usd_krw": "#84c1ff", "jpy100_krw": "#ffb3b3", "eur_krw": "#d2b4ff"}

# 예측 결과를 저장할 덱셔너리 (값과 신뢰구간 분리 저장)
fx_forecasts = {} # 평균 예측값 저장
fx_ci_bounds = {} # 신뢰구간(상/하단) 저장

# 환율 컬럼별로 반복 예측
for col in fx_cols:
    # 2020~2025.01 사이의 환율 학습 데이터 선택
    fx_train = fx_df[col].loc["2020-01-01":cutoff_date].dropna()
    # ARIMA 계절모델 구성 및 적합
    model = SARIMAX(fx_train, order=(1,1,1), seasonal_order=(1,1,1,12))
    result = model.fit(disp=False)
    # 24개월 예측
    forecast = result.get_forecast(steps=forecast_periods)
    forecast_df = forecast.summary_frame()
    forecast_df.index = future_dates # 예측 인덱스 future_dates로 설정
    # 예측 평균값 저장
    fx_forecast = forecast_df["mean"]
    fx_forecasts[col] = pd.concat([fx_train, fx_forecast]) # 과거 + 미래
    # 신뢰구간 저장
    fx_ci_bounds[col] = {
        "lower": forecast_df["mean_ci_lower"],
        "upper": forecast_df["mean_ci_upper"]}

```

```

# ----- 여행객수 시각화 -----
fig, ax1 = plt.subplots(figsize=(16, 8))
colors = {"United States": "orange", "Japan": "red", "Europe": "blue"}
# 여행객수 시각화
for country in target_countries:

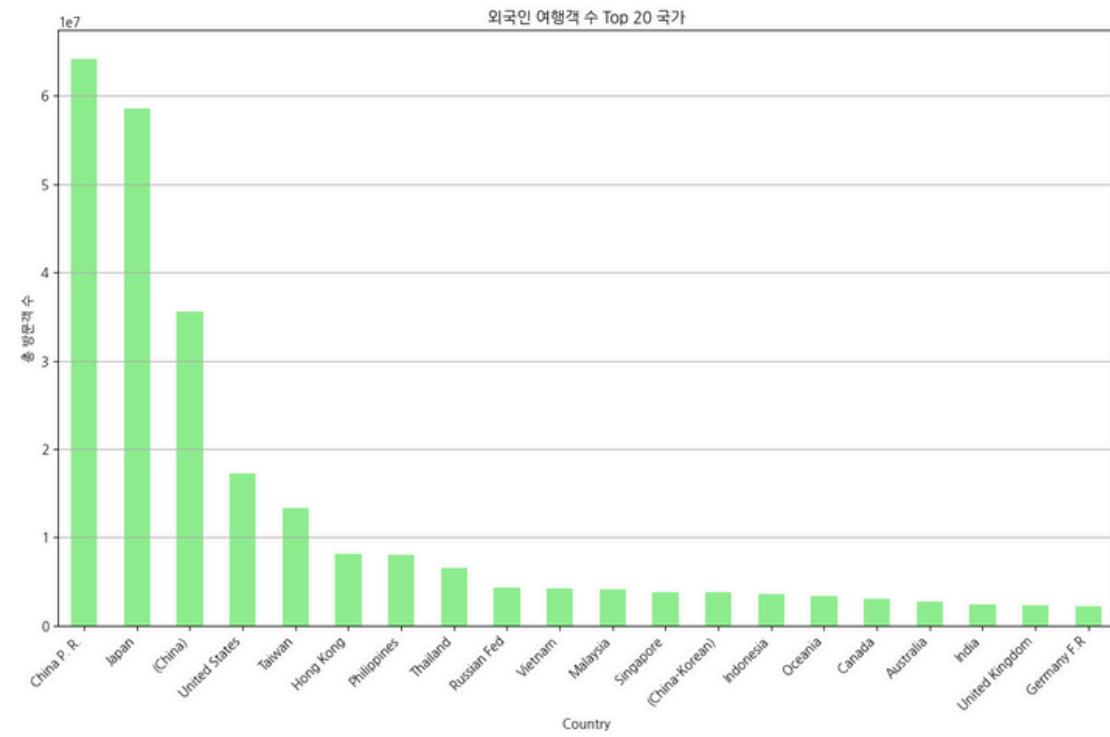
    data = results[country] # 해당 국가 예측 결과 호출
    actual = data[data["type"] == "actual"] # 실측 데이터
    forecast = data[data["type"] == "forecast"] # 예측 데이터
    c = colors[country]
    # 실측 값 그래프 (실선)
    ax1.plot(actual["date"], actual["value"], label=f"{country} (실측)",
             color=c, linewidth=2)
    # 예측 값 그래프 (점선)
    ax1.plot(forecast["date"], forecast["value"], label=f"{country} (예측)",
             linestyle="--", color=c, linewidth=2)
    # 예측 신뢰구간 시각화
    ax1.fill_between(forecast["date"], forecast["mean_ci_lower"],
                     forecast["mean_ci_upper"], color=c, alpha=0.05)

# y축(왼쪽): 여행객 수
ax1.set_ylabel("여행객 수", fontsize=12)
ax1.set_xlabel("날짜")
ax1.tick_params(axis='y')
ax1.set_title("2025~2027 미국·일본·유럽 여행객 및 환율 예측 (ARIMA)",
              fontsize=15)
ax1.grid(True)
# ----- 환율 시각화 -----
ax2 = ax1.twinx() # y축을 오른쪽에 하나 더 추가 (환율용)
# 각 환율에 대해 시각화
for col, label, line_color in zip(fx_cols,
                                   ["USD/KRW", "JPY(100)/KRW", "EUR/KRW"], ["orange", "red", "blue"]):
    # 환율 추이 (실측 + 예측 포함)
    ax2.plot(fx_forecasts[col], label=label,
             color=line_color, linestyle=":", linewidth=2)
    # 환율 예측 신뢰구간 시각화
    ax2.fill_between(future_dates,
                     fx_ci_bounds[col]["lower"],
                     fx_ci_bounds[col]["upper"],
                     color=color_map[col], # 위에서 지정한 색상
                     alpha=0.15)

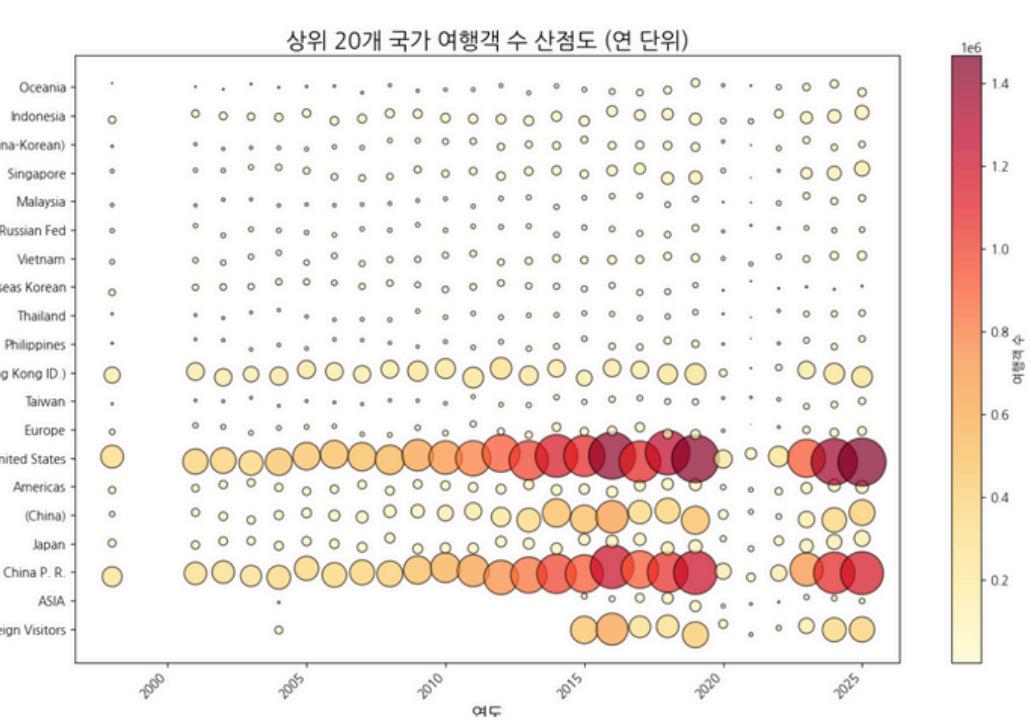
# y축(오른쪽): 환율
ax2.set_ylabel("환율", fontsize=12)
ax2.tick_params(axis='y')

```

여행객 수 TOP 20 국가 막대 그래프



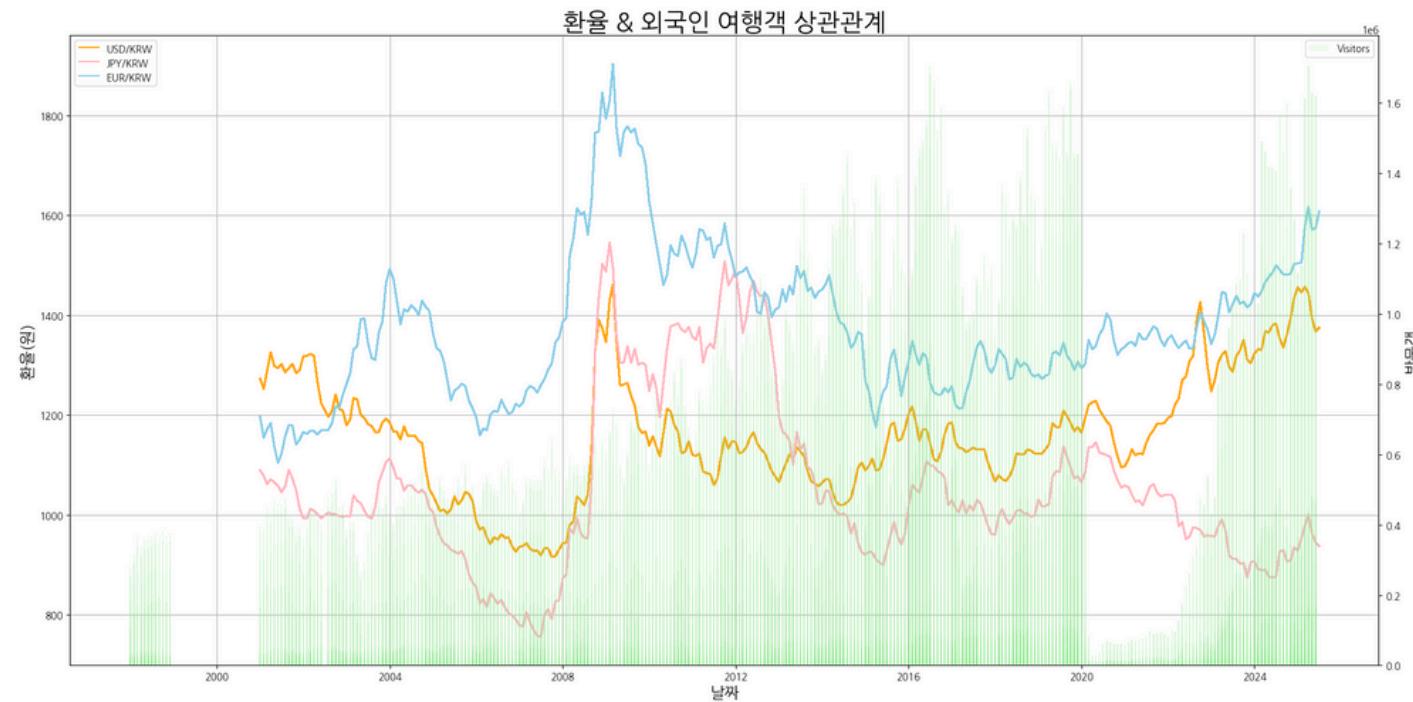
여행객 수 TOP 20 국가 산점도 그래프



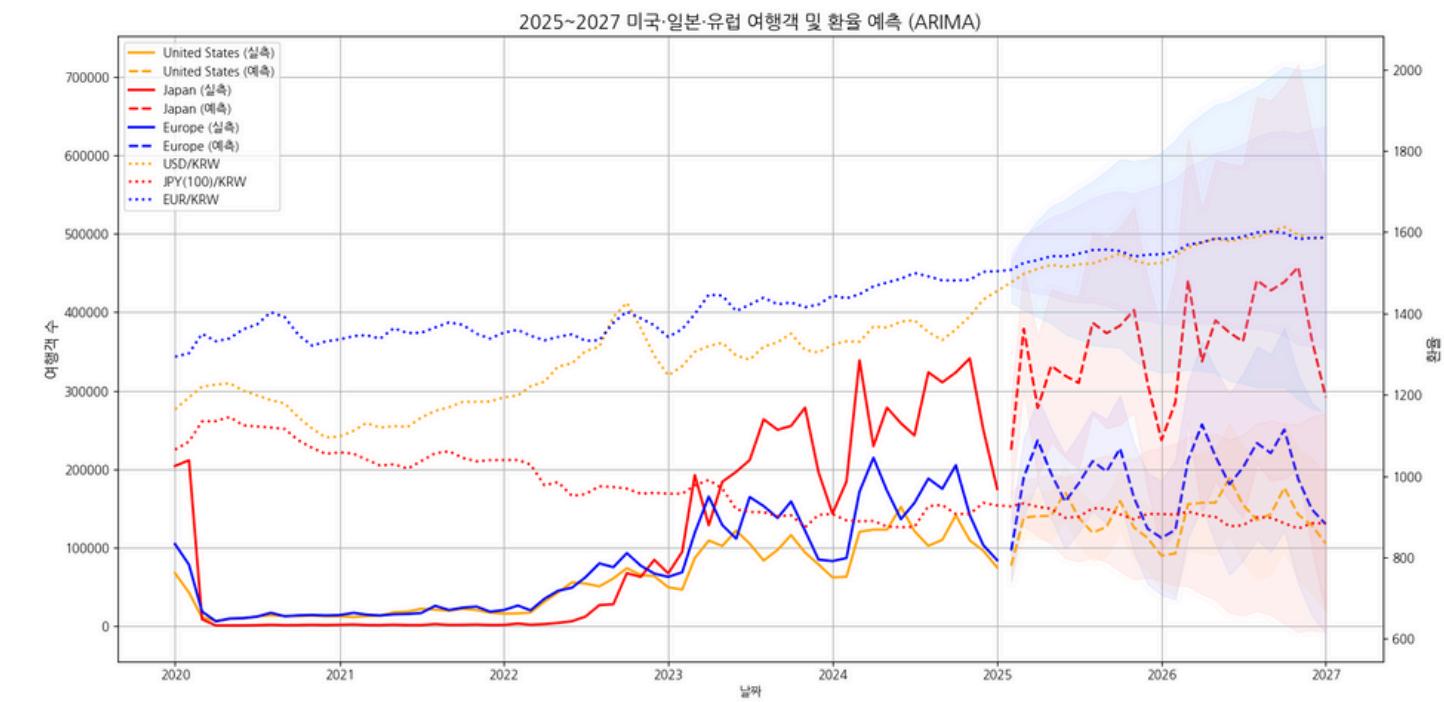
환율 변동 겹은선 그래프



환율 & 여행객 상관 관계 그래프



25.02~27.01 환율 & 여행객 ARIMA 예측 그래프



01

### 프로젝트를 통해 느낀점

사전에 기획한 의도대로 프로젝트와 관련된 데이터 자료 수집은 원활하게 진행 되었지만 수집한 데이터를 활용하여 ARIMA 예측 시각화 코딩과정에서 즉각적인 문제해결에 어려움을 겪었고 문제를 해결해 나가는 과정에서 개개인의 역량이 성장할 수 있는 발판이 되었다

완성도 평가 : 9점



02

### 추후 개선 & 보완할 점

환율변동에 따른 외국인 여행객 증감 상관관계를 시각화하는 결과물을 토대로 관광 수입의 대한 데이터 까지 추가 하여 더 다양한 관광 산업에 유용한 정보 제공

### 황다니엘

프로젝트를 팀장으로 이끌면서 협업과 소통의 중요성을 다시 한 번 깊이 느낄 수 있었습니다. 상황에 따라 빠르게 대처하고 업무를 효과적으로 지시하는 경험을 통해 실무 감각도 한층 향상된 것 같습니다. 또한 프로젝트를 진행하며 배운 내용을 바탕으로 관심 있는 주제를 데이터로 정리하고 결과물을 만들어보니, 코딩에 대한 이해도가 높아졌을 뿐만 아니라 앞으로 더 흥미로운 프로젝트에 도전하고 싶다는 동기부여도 얻을 수 있었습니다.

### 박시온

팀원들과 협업하여 팀프로젝트를 진행하면서 본인 스스로의 역량과 미흡한 부분들을 제대로 파악할 수 있는 시간이었고, 개개인의 능력도 중요하지만 팀원과의 원활한 소통과 협업 또한 간과할 수 없는 부분이라는 걸 다시 한 번 느끼게 해주는 계기가 되었습니다

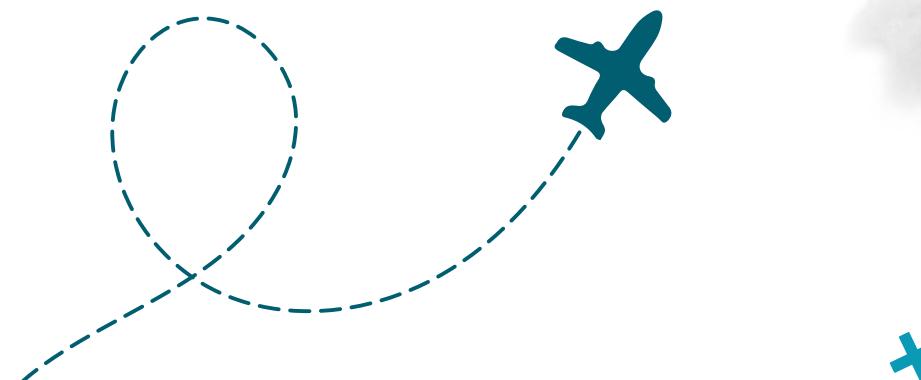
팀장이 각자에게 맞는 업무 지시를 잘 해주었습니다.  
프로젝트를 진행하면서 데이터 자료의 중요성을 체감하였고, 팀원 간의 커뮤니케이션 또한 중요한 업무라는 것을 배우는 시간이었습니다.  
스스로의 부족함을 배우는 경험을 통하여 한층 더 성장 할 수 있는 계기가 되었습니다.

데이터를 전처리하는 과정을 수행하면서, 이전보다 한 층 더 이해하는 시간이 이었으며, 협업을 진행하고 체감한 팀원 간 협업의 중요성을 몸소 느낀 경험이 되었습니다.  
부족한 점을 메우고 또 다른 것을 배우려 노력하는 계기가 되었다 생각합니다.  
이에 따라 실무에서는 더 원활한 팀원간의 소통이 필요할것이라 느꼈습니다

### 이은지

### 김민호

# Thank You For Your Time



*Explore The World  
With Ease!*