

영국 구성국 별 생활 패턴 분석

Pub, Coffee shop, Café, Bar 메뉴 중심으로

김윤희, 양동주

목차

- | PART.1 프로젝트 의도 및 전처리
- | PART.2 구성국 별 패턴과 상호보완적 메뉴 관계
- | PART.3 카페 메뉴와 커피숍 메뉴의 패턴
- | PART.4 예측 알고리즘(빅데이터언어관련)
- | PART.5 후기

| 프로젝트 의도

- 영국은 개성이 강한 4개의 구성국으로 이뤄져 있음.
- 한 나라 안에서도 구성국마다 라이프 스타일에 차이가 있는 지 확인
- 각 구성국의 차이에 따라 시장 진출 현지 전략 등에 활용 가능.

| 전처리

안 쓰는 데이터 삭제

```
1 import pandas as pd
2
3 data = pd.read_csv("C:/Users/yuno/Desktop/데이터시각화/GB.csv", sep=",")
4
5 del data["Offset"]
6 del data["방문국가 수"]
7 del data["메뉴 수"]
8 del data["메뉴 카테고리 수"]
9 del data["메뉴당 체크인 수"]
10 del data["사용자 수"]
11 del data["인당 체크인 수"]
12 del data["체크인 수"]
13
```

User_ID로 정렬한 후 깨진 문자 대체하기

```
48 data.sort_values(by=["User_ID", ascending=False])
49
50 data = data.replace(['Caf💎💎', "Cafe"])
51 data = data.replace(['Caf💎', "Cafe"])
52 df = data.iloc[0:500000]
53 df2 = data.iloc[500000:]
```

| 전처리


‘Date’를 태블로에서 활용할 수 있게 나눔

```
14 df = data["Date"].str.split(" ", expand=True)
15
16 data1 = pd.concat([data, df], axis=1)
17 data1.rename(columns={0:"Week", 1:"Month", 2:"Day", 3:"Time", 5:"Year"})
18 df = data["Month"]
19 Month_arr = df.tolist()
20 Month_num = []
21 for i in range(0, len(Month_arr)):
22     if Month_arr[i] == 'Jan':
23         Month_num.append("1")
24     elif Month_arr[i] == 'Feb':
25         Month_num.append("2")
26     elif Month_arr[i] == 'Mar':
27         Month_num.append("3")
28     elif Month_arr[i] == 'Apr':
29         Month_num.append("4")
30     elif Month_arr[i] == 'May':
31         Month_num.append("5")
32     elif Month_arr[i] == 'Jun':
33         Month_num.append("6")
34     elif Month_arr[i] == 'Jul':
35         Month_num.append("7")
36     elif Month_arr[i] == 'Aug':
37         Month_num.append("8")
38     elif Month_arr[i] == 'Sep':
39         Month_num.append("9")
40     elif Month_arr[i] == 'Oct':
41         Month_num.append("10")
42     elif Month_arr[i] == 'Nov':
43         Month_num.append("11")
44     elif Month_arr[i] == 'Dec':
45         Month_num.append("12")
46 data["Month_Num"] = Month_num
```

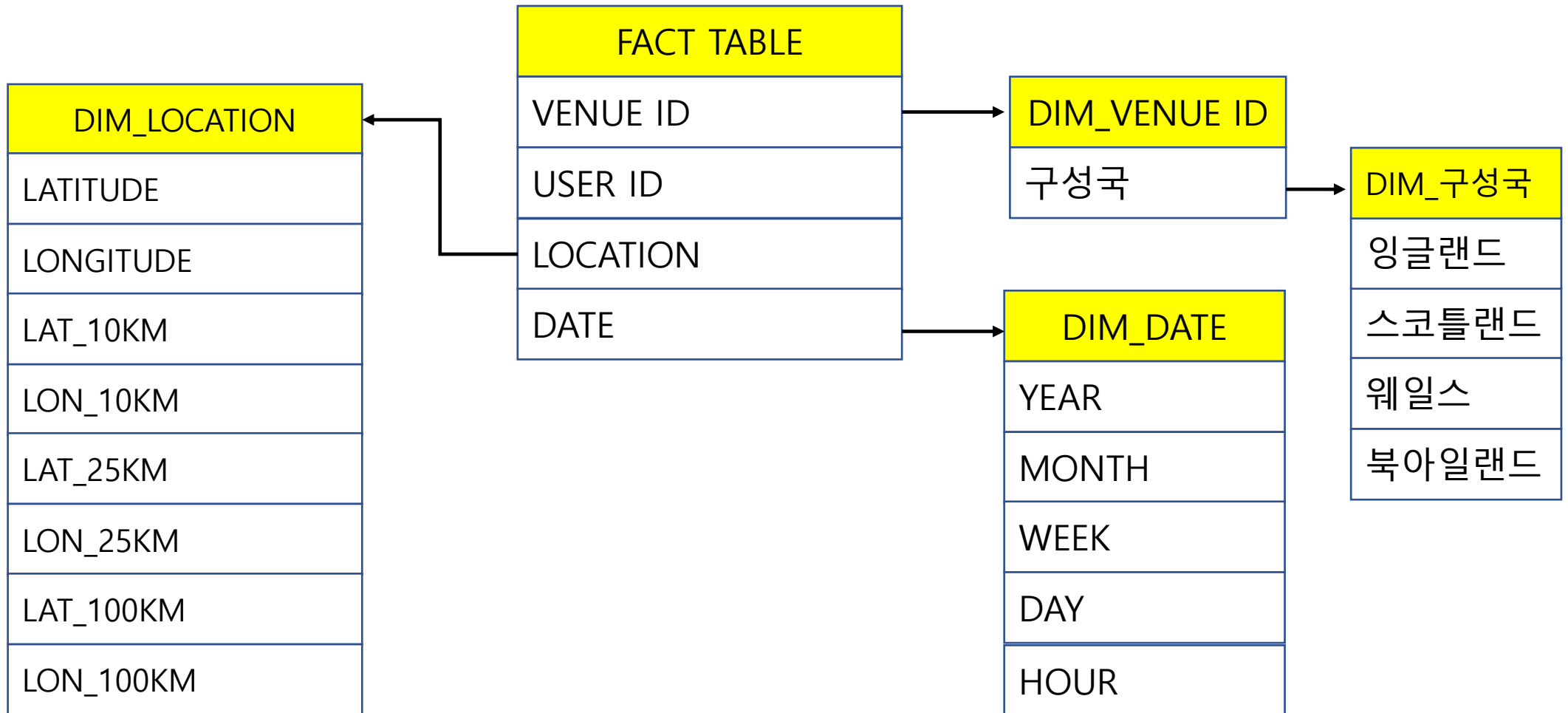
| 전처리

공간 분할 차원을 엑셀(ROUND, MROUND)로 추가하기엔 영국 데이터가 너무 큼.

이를 해결하기 위해 데이터를 반으로 나눠서 엑셀로 공간 분할 차원을 추가.

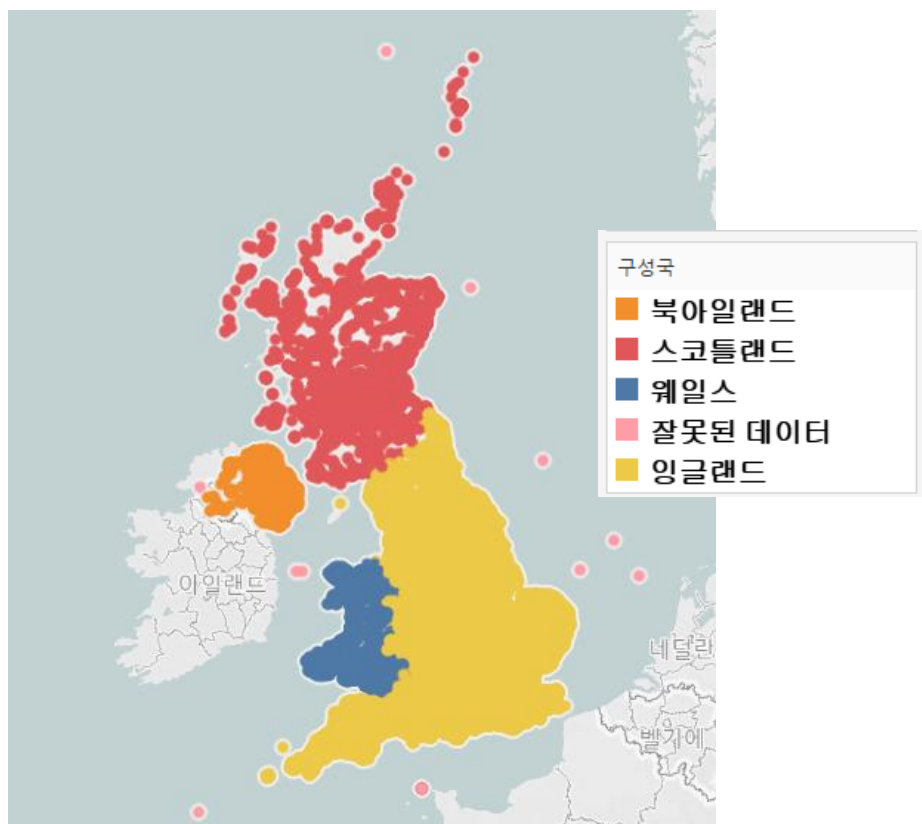
```
55 df.to_csv("C:/Users/yuno/Desktop/데이터시각화/GB_0~499999.csv")
56 df2.to_csv("C:/Users/yuno/Desktop/데이터시각화/GB_500000~.csv")
57 
58 data.to_csv("C:/Users/yuno/Desktop/데이터시각화/GB.csv")
```

| 전처리- 데이터 구조



전처리

태블로의 자유 선택 도구와 그룹 기능을 활용해
영국 국토 외부에 발생한 데이터는 제외하고, 구성국 국토에 따라 VENUE ID를 그룹화



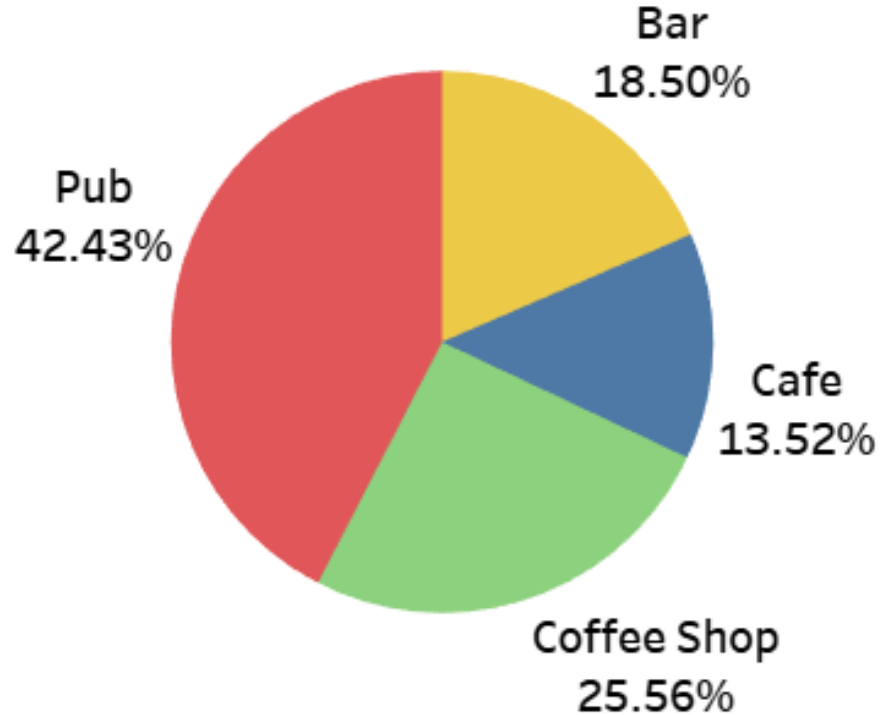
필드명: 구성국

그룹:

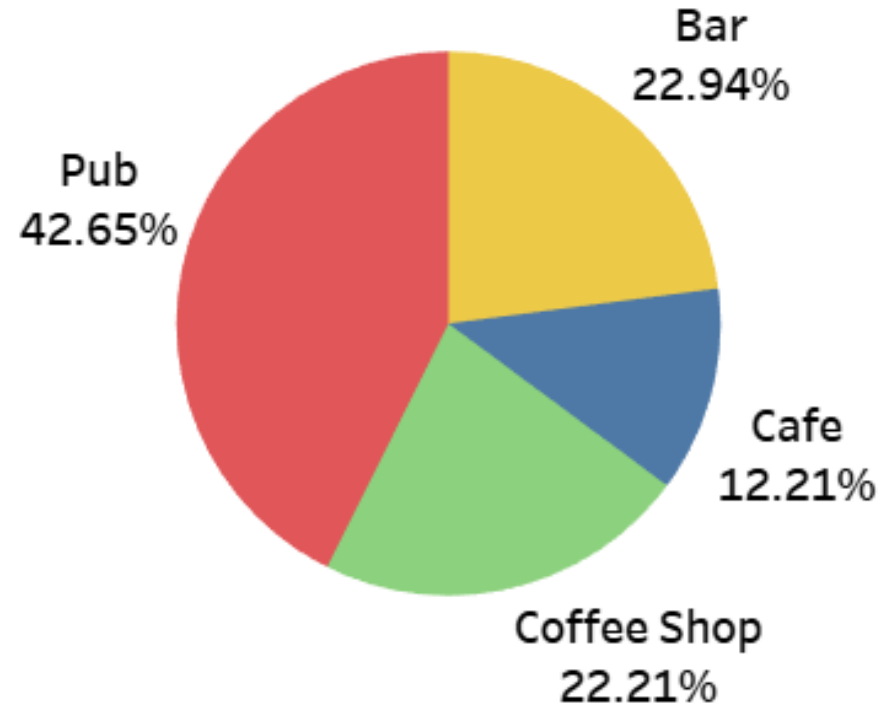
- > @ 스코틀랜드
- > @ 북아일랜드
- > @ 웨일스
- > @ 잘못된 데이터
- > @ 잉글랜드

| 구성국 별 패턴

구성국 별 메뉴 이용률



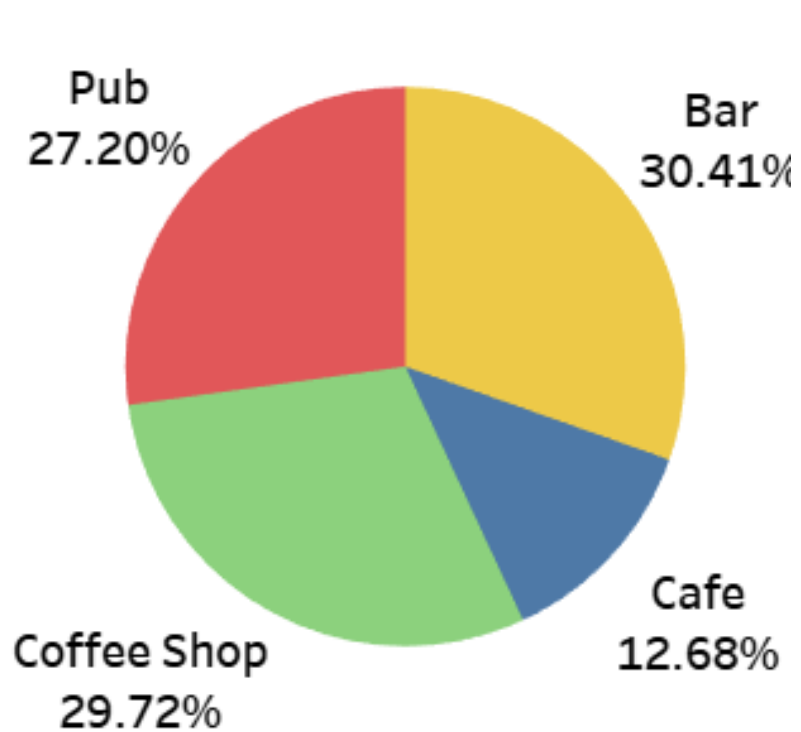
잉글랜드



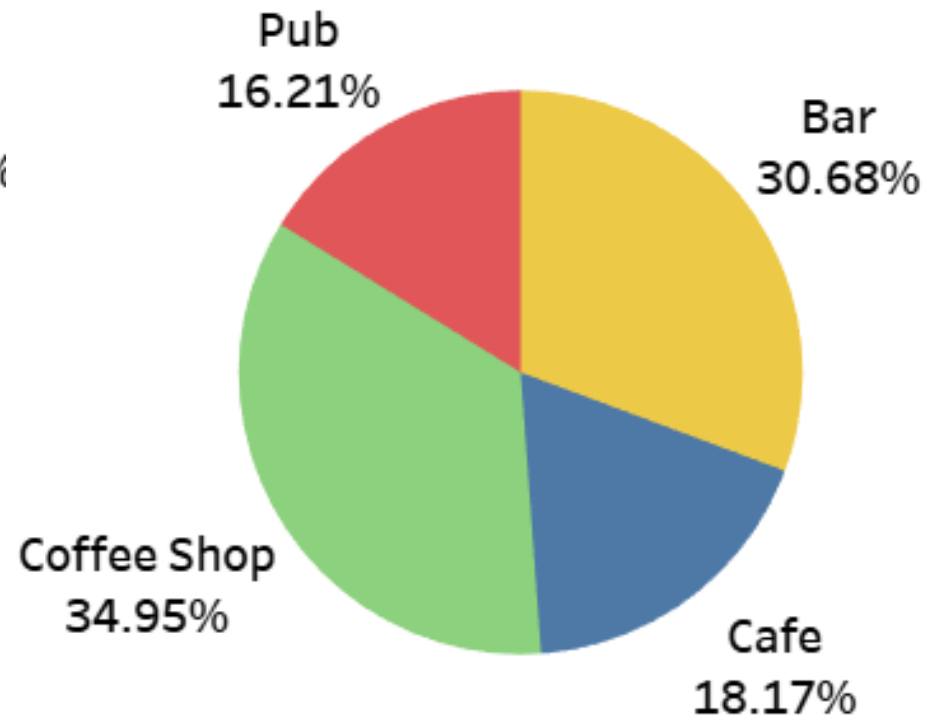
웨일스

| 구성국 별 패턴

구성국 별 메뉴 이용률



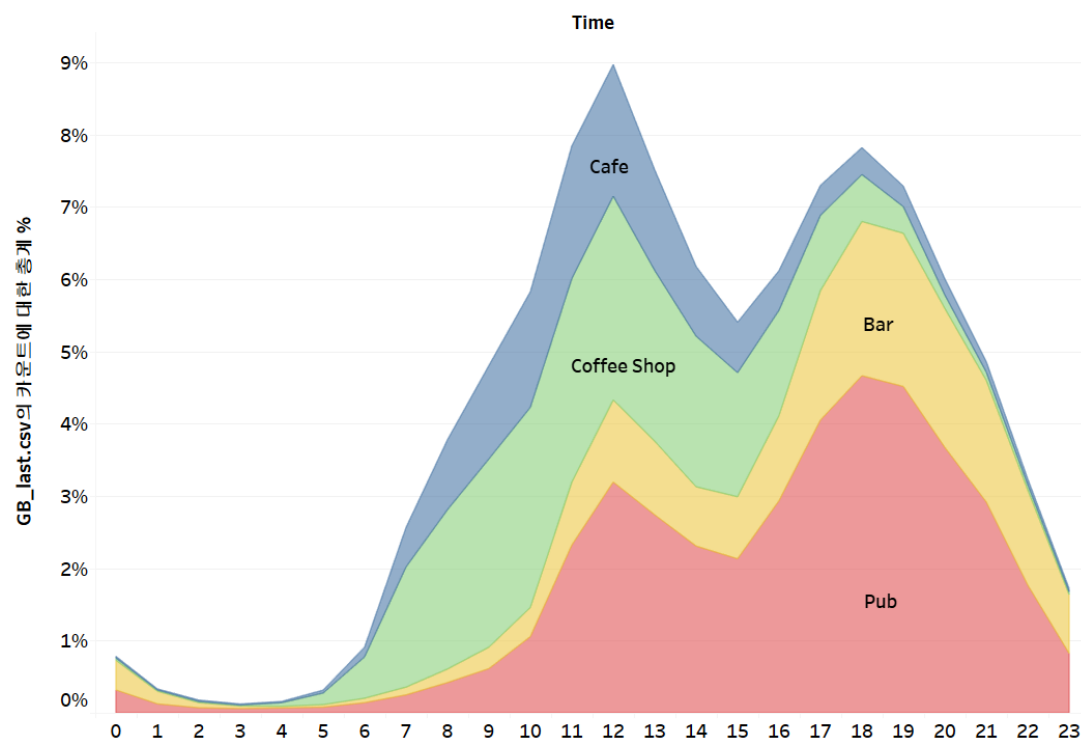
스코틀랜드



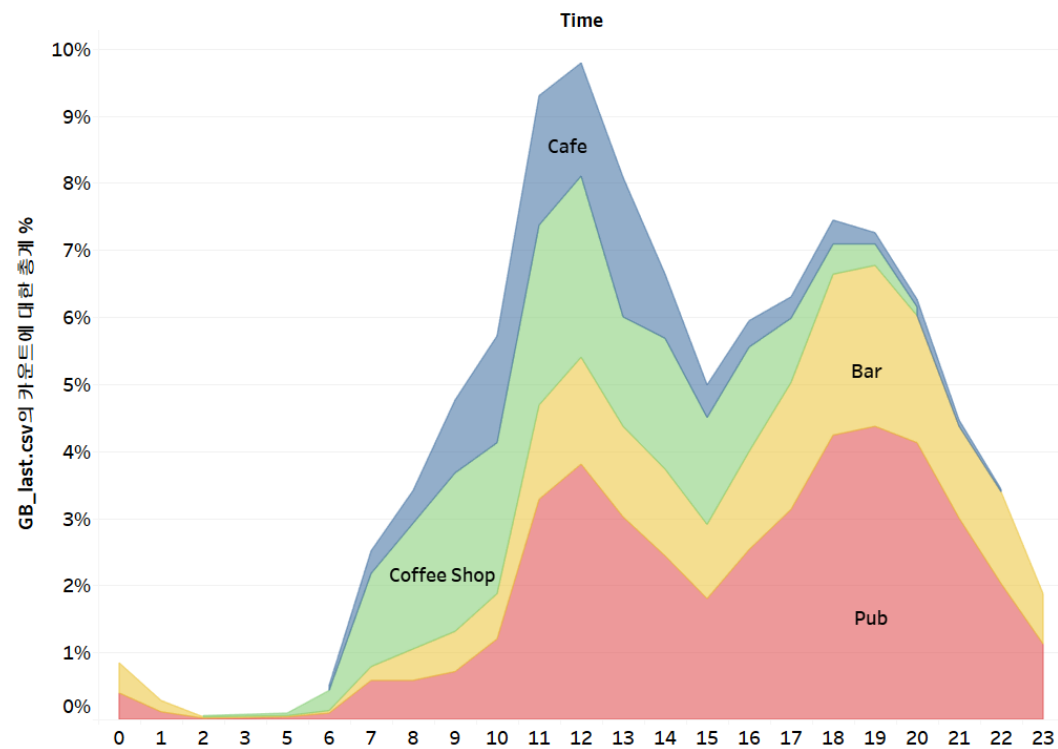
북아일랜드

시간대 패턴

구성국 별 시간대에 따른 메뉴 이용률



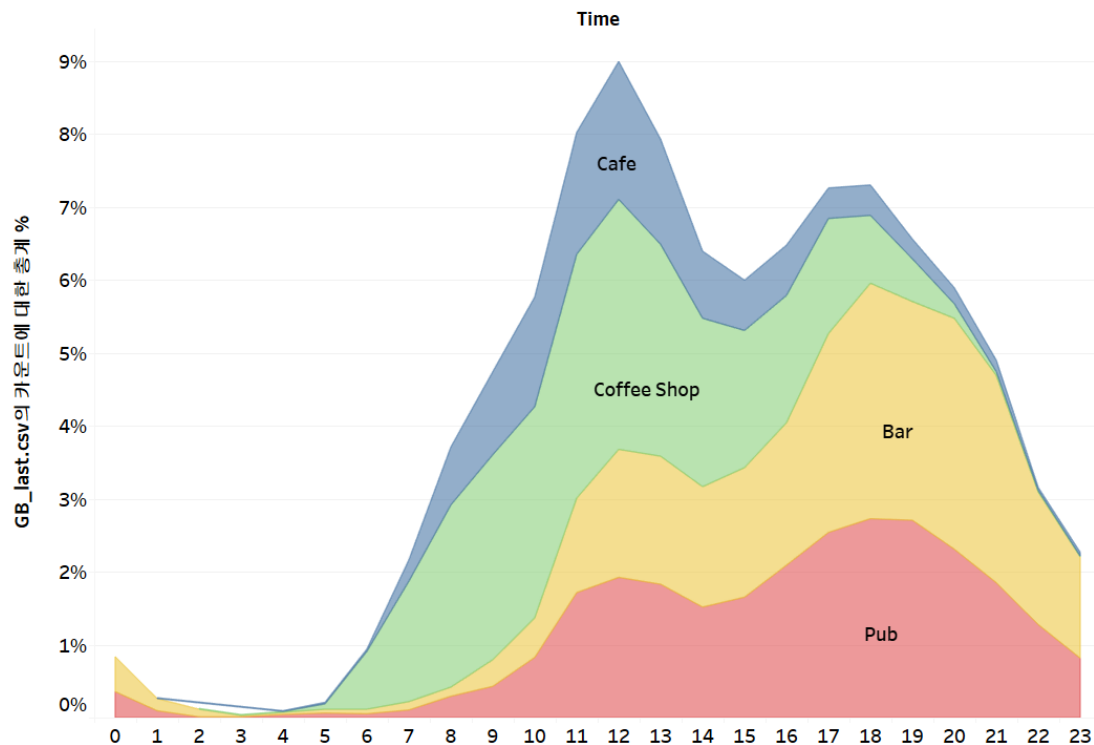
잉글랜드



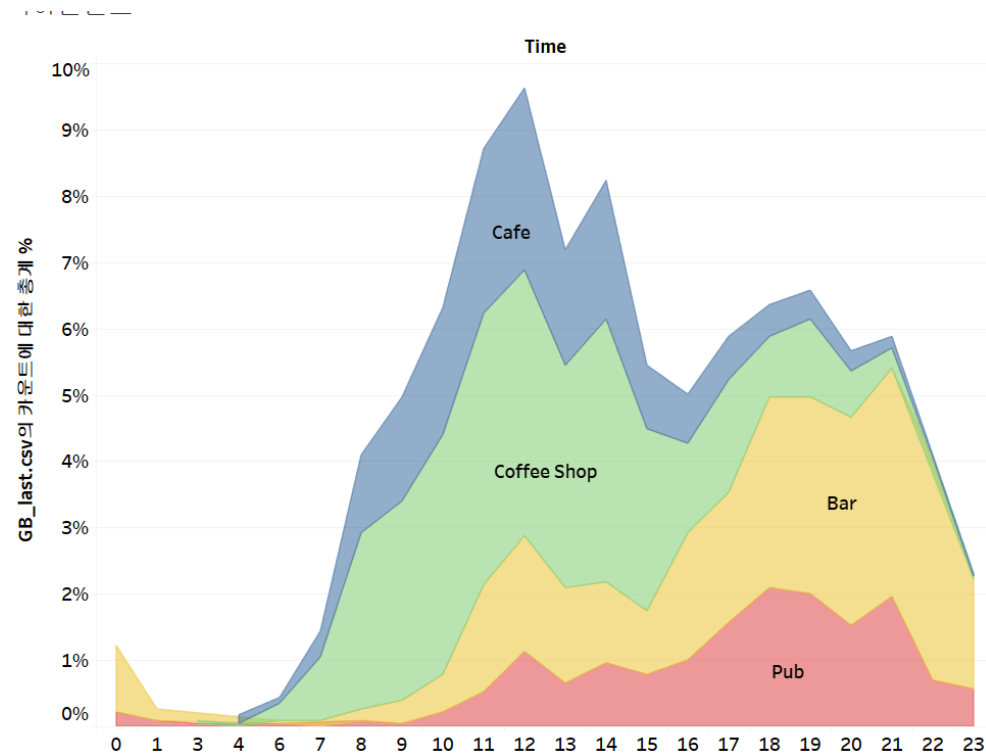
웨일스

시간대 패턴

구성국 별 시간대에 따른 메뉴 이용률



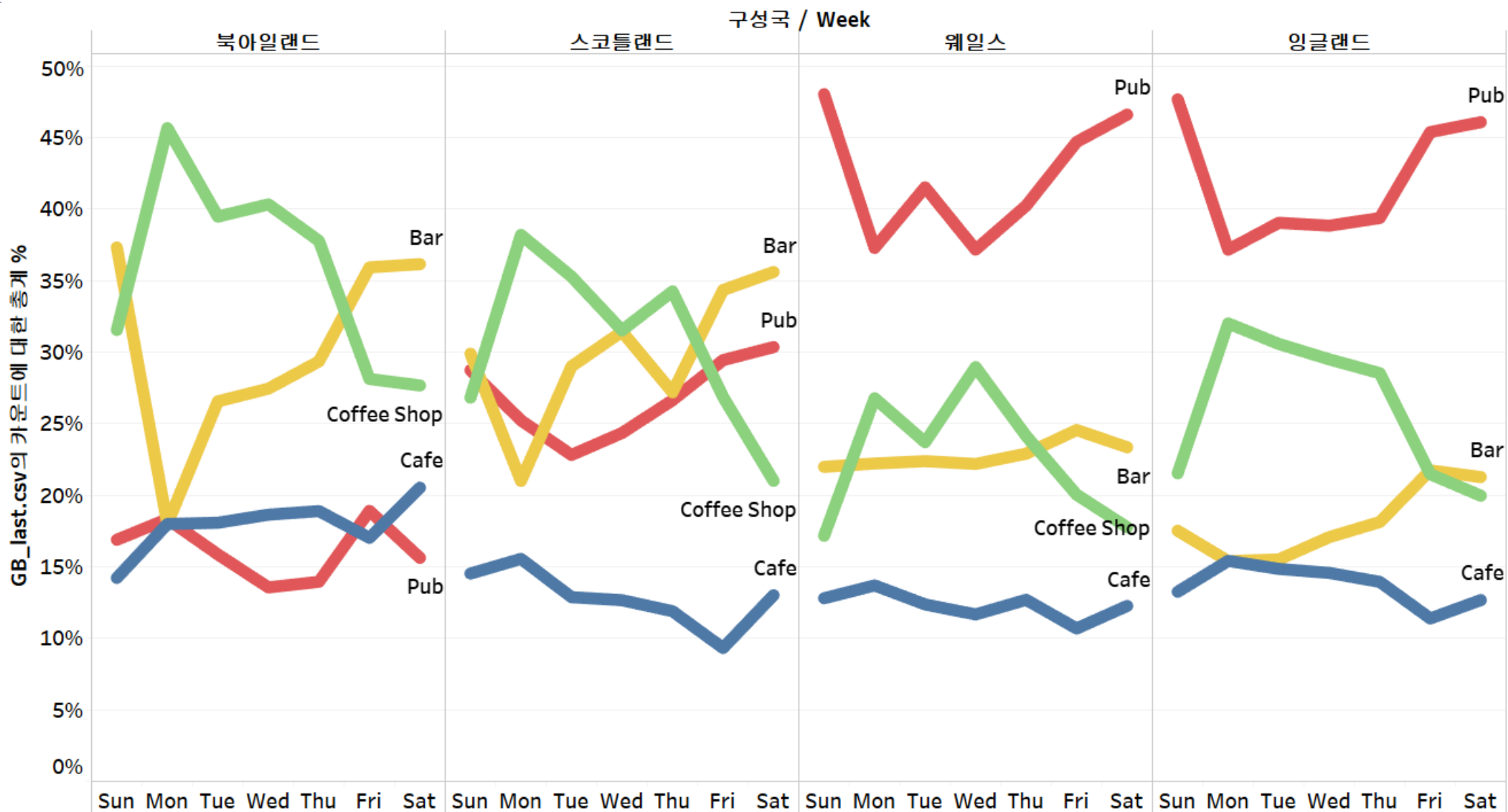
스코틀랜드



북아일랜드

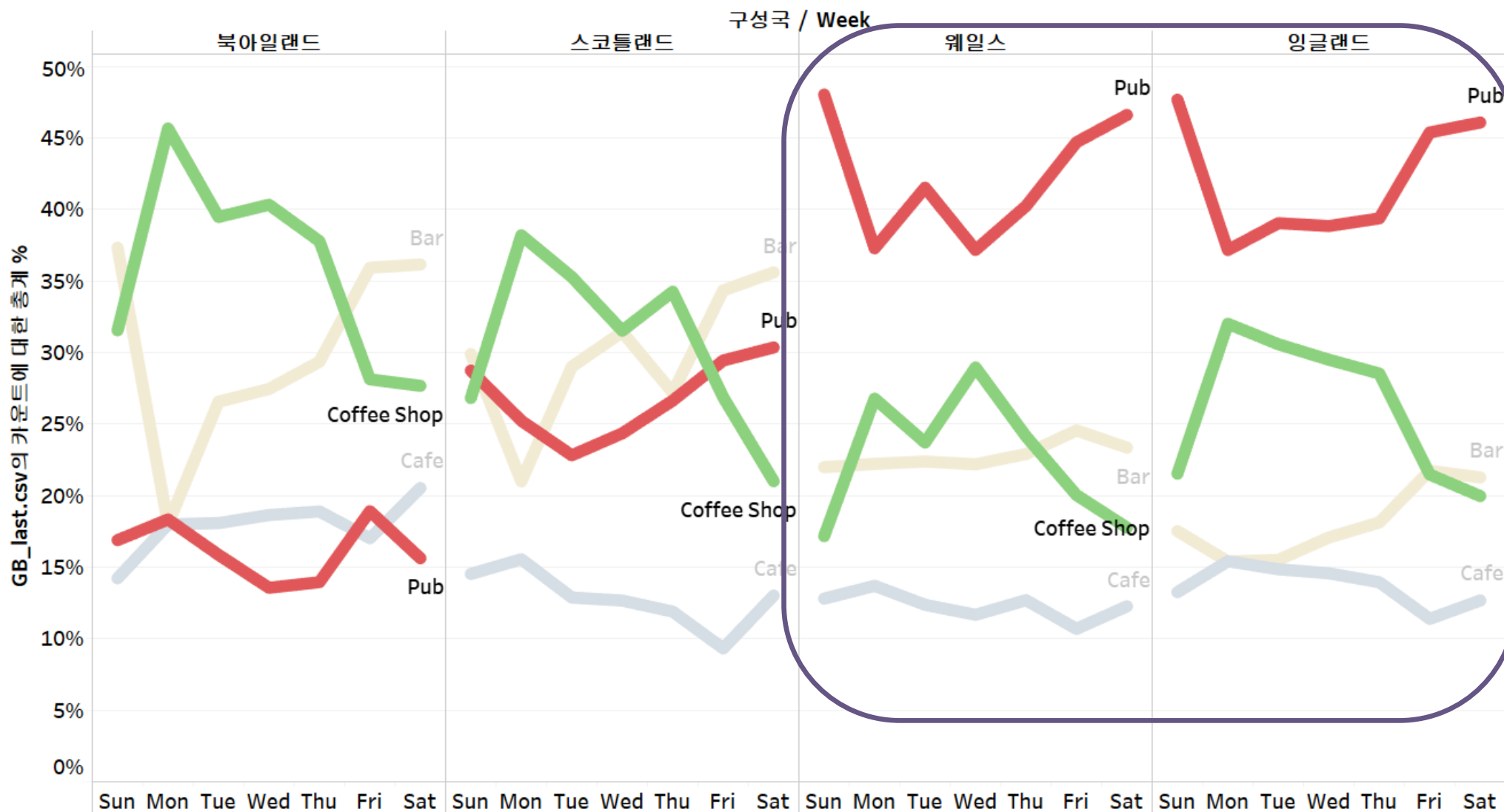
상호보완적 패턴

구성국 별 요일에 따른 상호보완적 관계



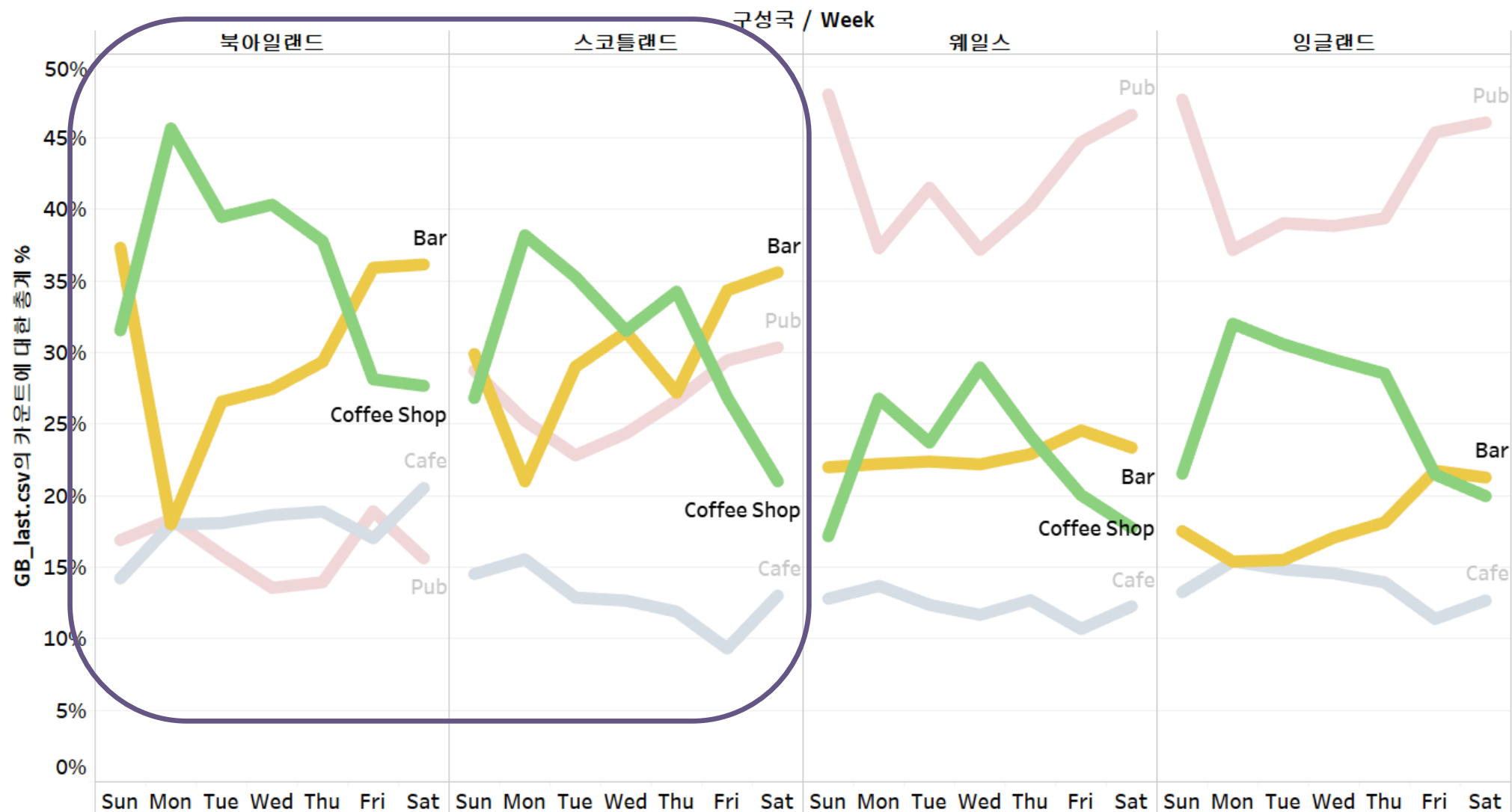
상호보완적 패턴

구성국 별 요일에 따른 상호보완적 관계



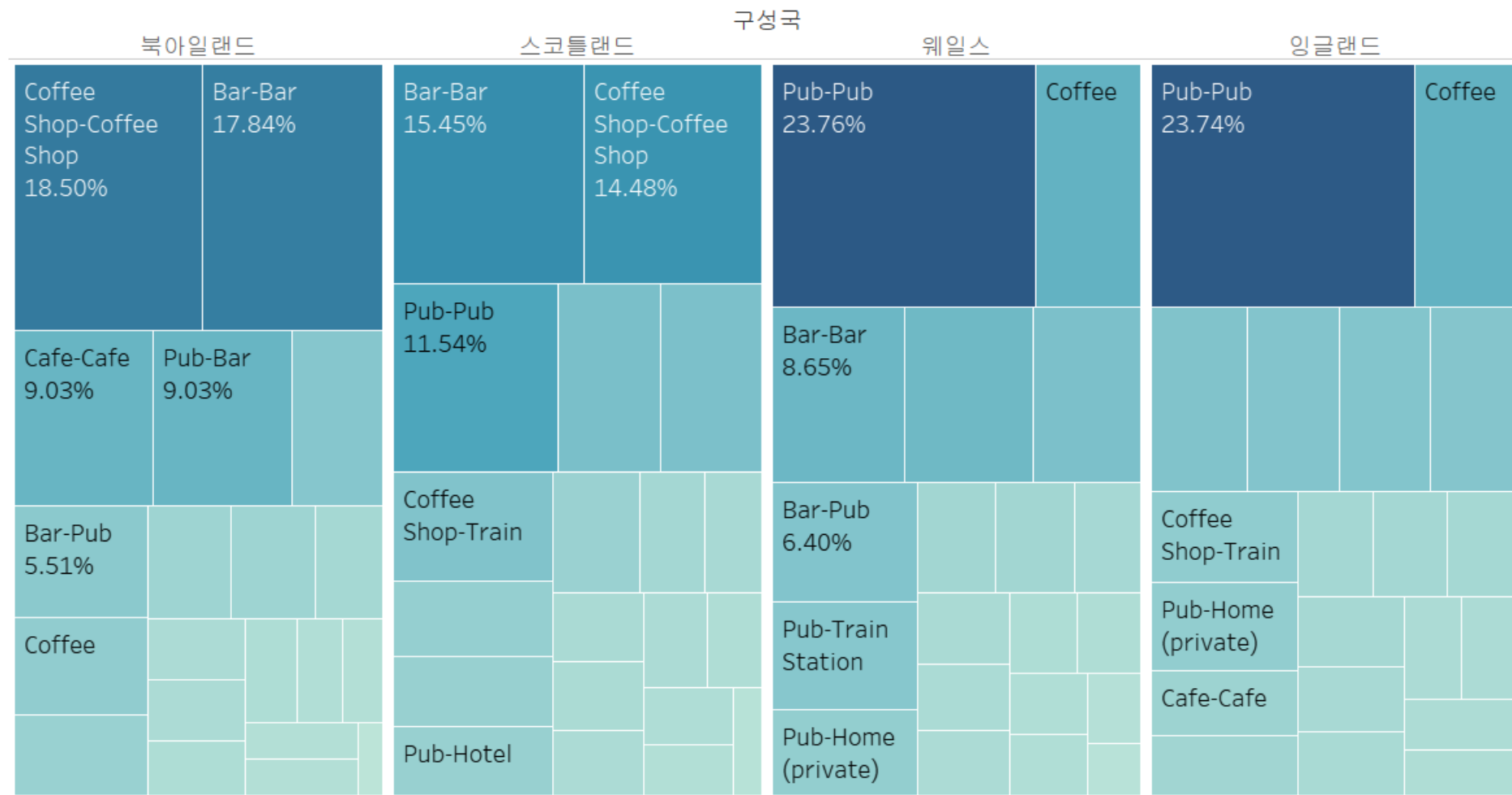
상호보완적 패턴

구성국 별 요일에 따른 상호보완적 관계



| 메뉴 이동 패턴

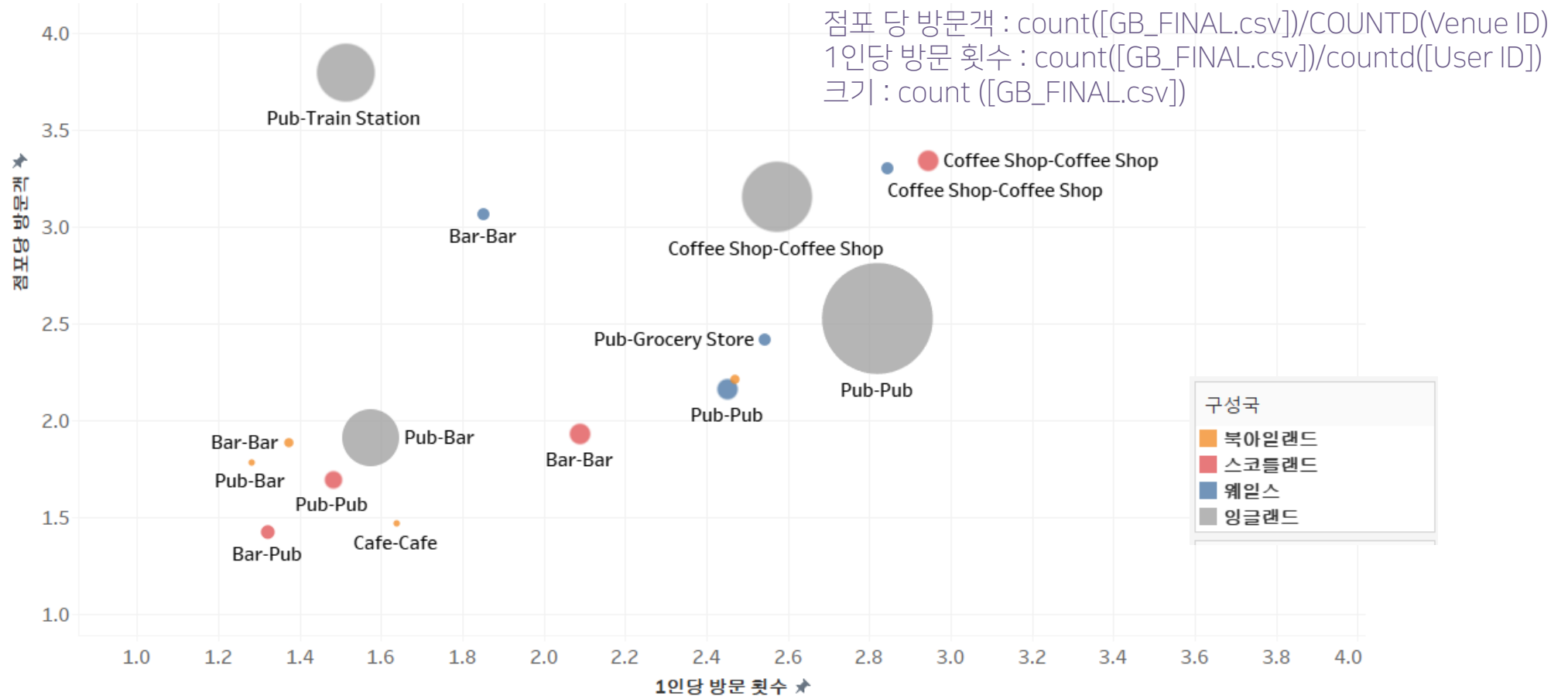
구성국 별 "이전 메뉴-현재 메뉴" 분포 시각화



"이전메뉴-현재메뉴"의 구성국 별 분포 시각화를 할 경우 동일 메뉴 카테고리가 연결된 패턴이 1위를 차지한다.

| 메뉴 이동 패턴

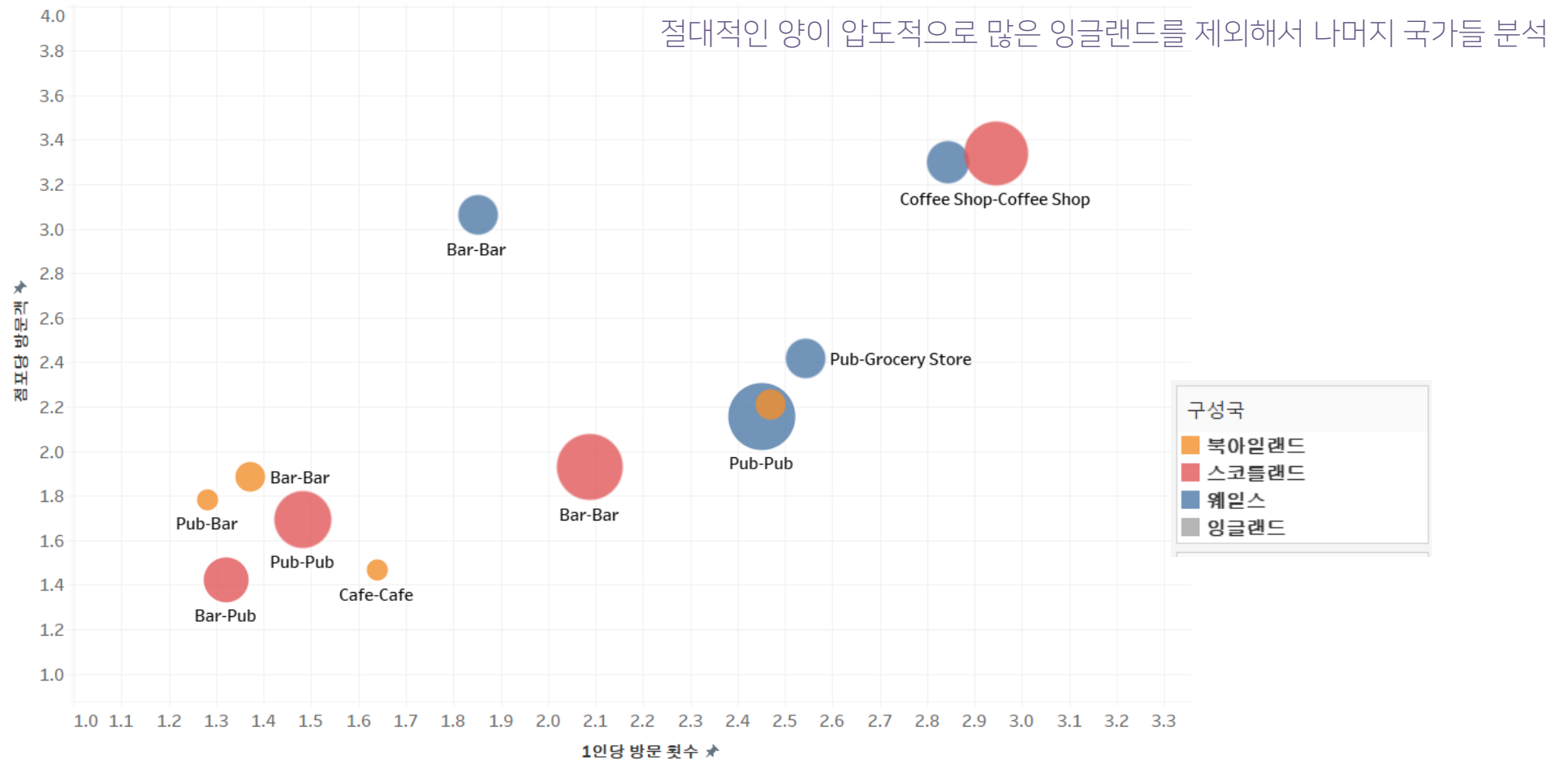
구성국 별 "이전 메뉴-현재 메뉴" 상위 16개(4패턴x4구성국) 패턴 관계 시각화



커피숍 연속 패턴이 개인 별 횟수와 점포 성과 지표 모두 우수하게 나오는 편이다.

| 메뉴 이동 패턴

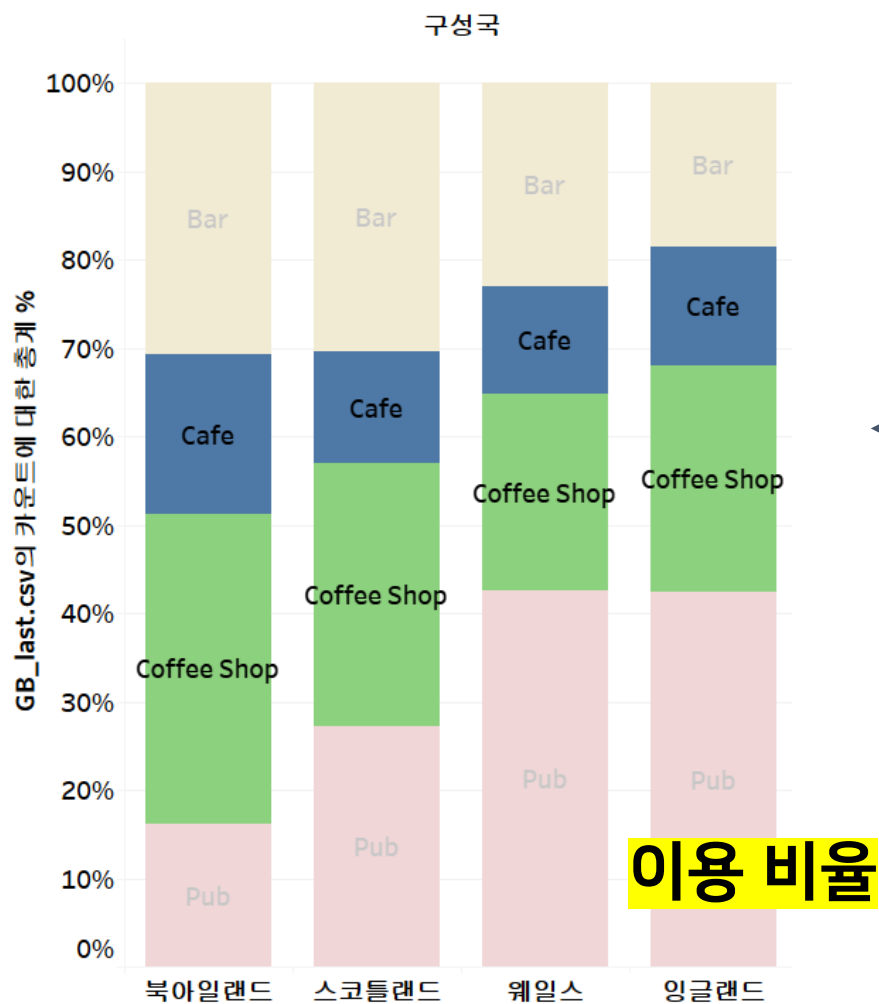
구성국 별 "이전 메뉴-현재 메뉴" 상위 16개(4패턴x4구성국) 패턴 관계 시각화



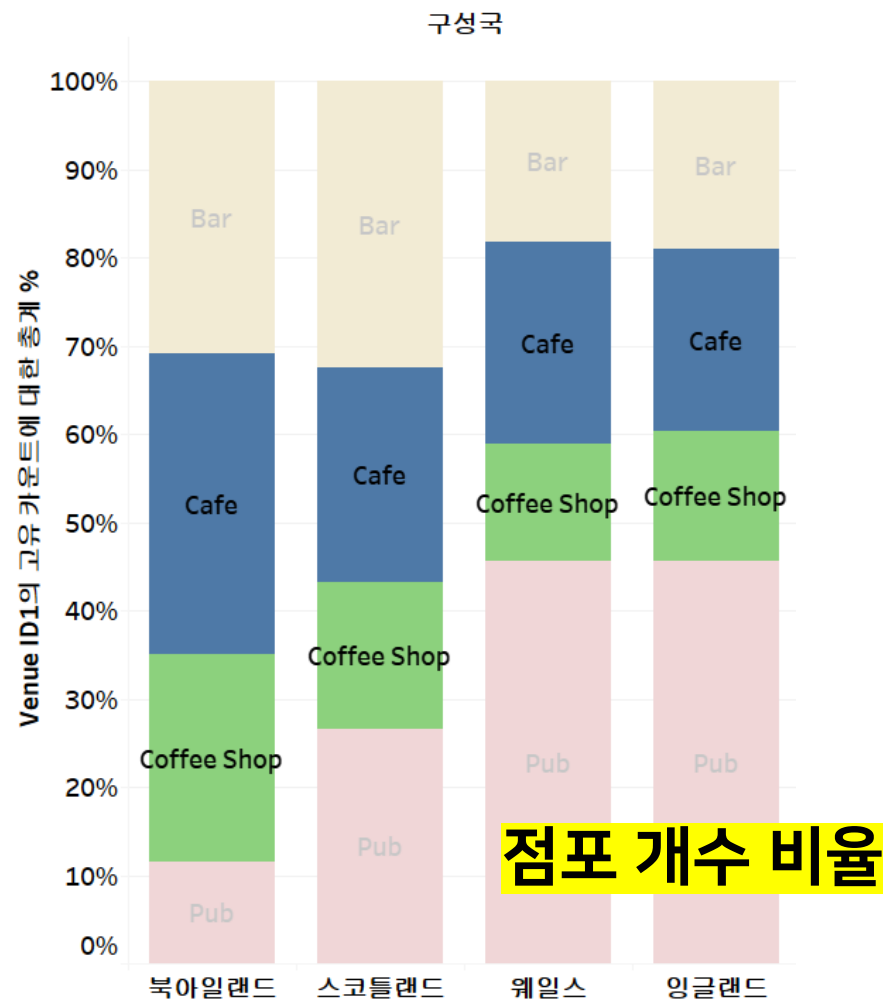
패턴 카운트 횟수가 비슷한 경우에도 커피숍 연속 패턴이 펍 연속 패턴이나 바 연속 패턴보다 개인 지표와 가게 성과 지표 모두 좋은 편임을 알 수 있다.

커피숍과 카페 사이의 특이 패턴

이용률과 점포 개수 사이의 모순



모순



| 커피숍과 카페 사이의 특이 패턴 이용률과 점포 개수 사이의 모순

발견한 특이 패턴의 이유를 알아보기 위한 공간 시각화

- 구성국의 국토 규모에 따라 10km, 25km, 100km로 공간 분할
- 푸른색은 카페, 녹색은 커피숍
- 크기는 "총 방문 횟수/점포 개수" 필드 사용

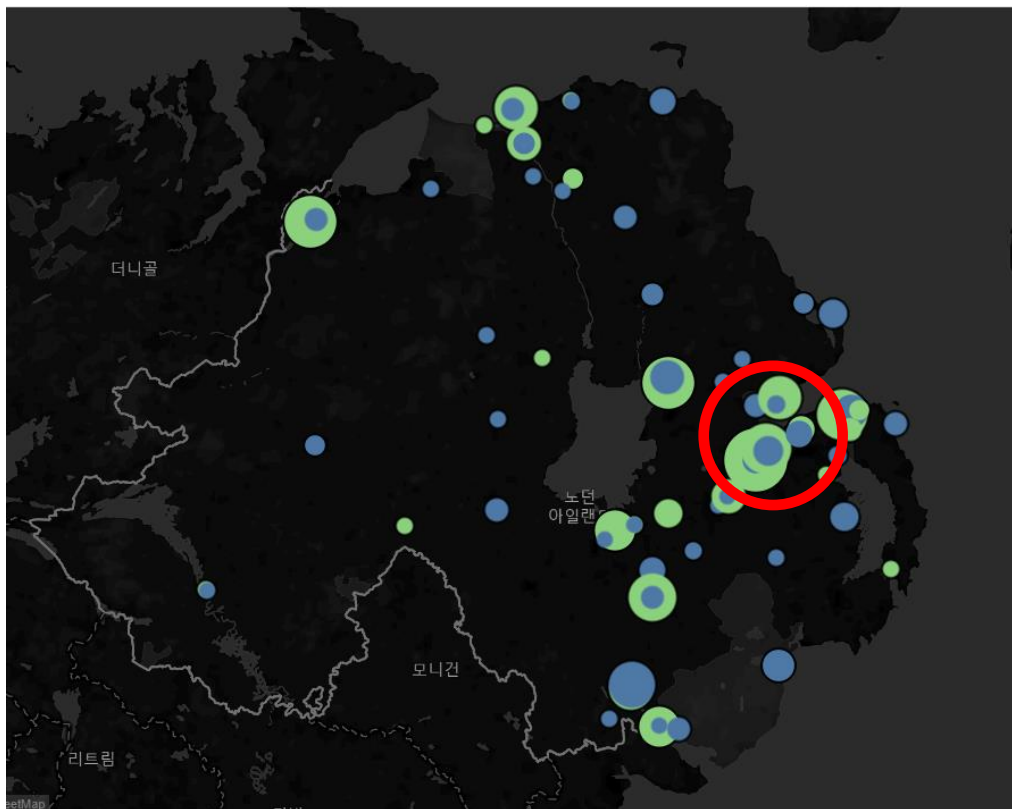
커피숍과 카페 사이의 특이 패턴

이용률과 점포 개수 사이의 모순

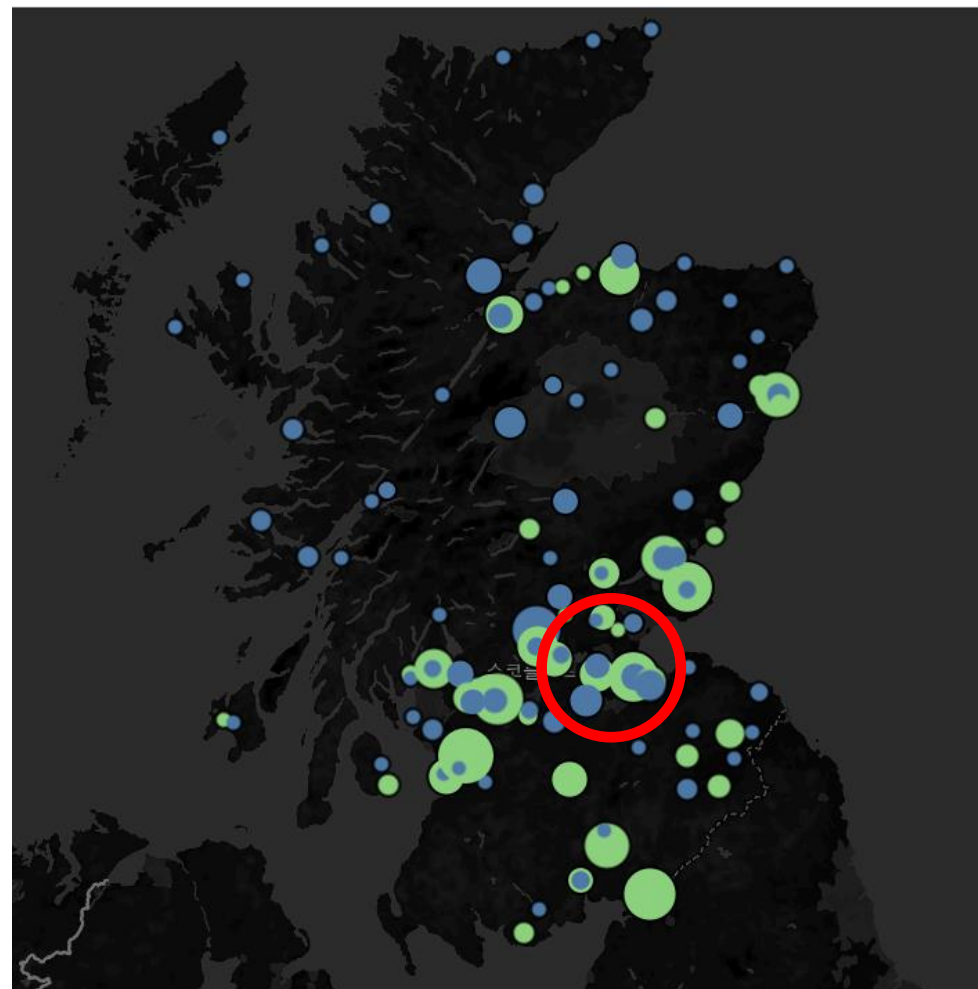
붉은 원 : 수도 지역

녹색 : 커피숍

파란색 : 카페

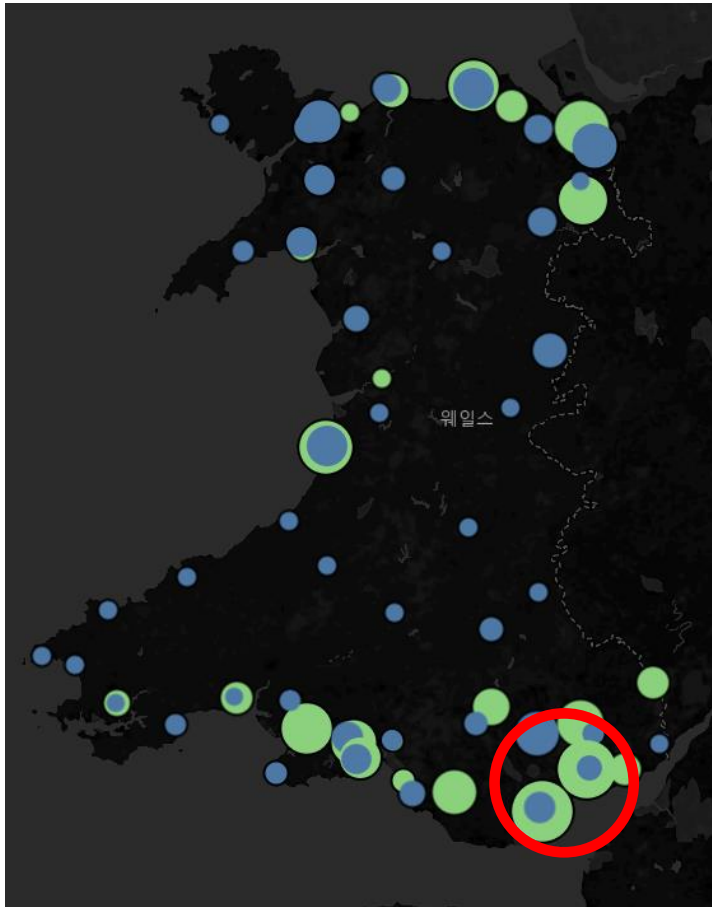


북아일랜드 (10KM 공간 분할)

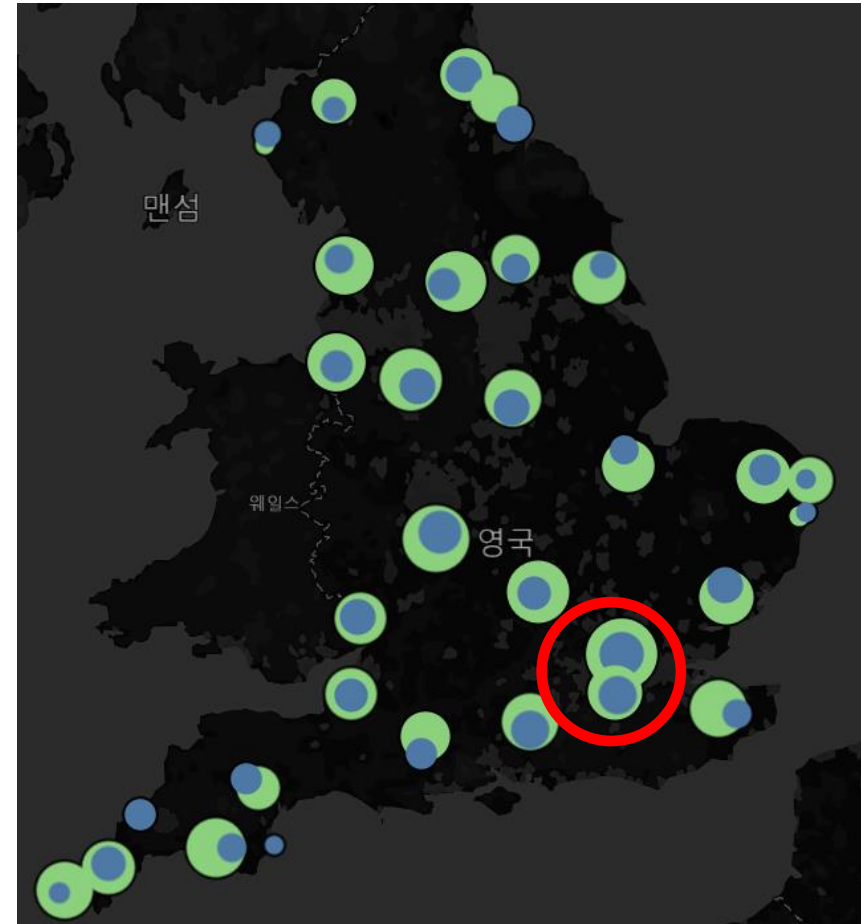


스코틀랜드 (25KM 공간 분할)

| 커피숍과 카페 사이의 특이 패턴 이용률과 점포 개수 사이의 모순



웨일스 (10KM 공간 분할)



잉글랜드 (100KM 공간 분할)

| 커피숍과 카페 사이의 특이 패턴

이용률과 점포 개수 사이의 모순

커피숍 : 수도 근처 지역 등 일부 지역에만 분포. 점포 별 방문 횟수 높음

카페 : 전국에 고르게 분포, 점포 별 방문 횟수 낮음

잉글랜드 : 커피숍, 카페 모두 고르게 분포, 다만 점포 별 방문 횟수는 커피숍이 높음.

왜 커피숍과 카페의 공간 분포에 차이가 있을까?

| 커피숍과 카페 사이의 특이 패턴

이용률과 점포 개수 사이의 모순



이를 확인하기 위해 각 구성국의 카페, 커피숍 상위 10개의 VENUE ID를 모으고
모은 VENUE ID를 ko.foursquare.com에 검색해 상호명을 크롤링

커피숍과 카페 사이의 특이 패턴

이용률과 점포 개수 사이의 모순

```
file = open("C:/Users/yuno/Desktop/데이터시각화/그거/스코트랜드.txt", 'r')
lines = file.readlines()
con_name = []
value_name = []
```

```
for line in lines:
    con_name.append(line[:-1])
```

```
del con_name[0]
```

```
def get_html(url):
    _html = ""
    resp = requests.get(url)
    if resp.status_code == 200:
        _html = resp.text
    return _html
```

```
for i in range(0, len(north_island)):
    URL = "https://ko.foursquare.com/v/" + con_name[i]
    html = get_html(URL)
    soup = BeautifulSoup(html, 'html.parser')
    table = soup.find(id = "h1")
    venues = soup.find_all(class_ = "venueName")
    name = venues[0].text.replace('\n', '').replace('\t', '').replace(' ', '')
    value_name.append(name)
```

크롤링 할 Venue_ID가 정리된
텍스트 파일 열기

List에 읽은 테스트 파일
내용을 저장

Html에 접근하기 위해
서 requests 작성.

텍스트 파일의 Venue_ID마다 사이
트 초기화. 이후 정보를 받아 원하
는 부분(VenueName)만 추출.

커피숍과 카페 사이의 특이 패턴

이용률과 점포 개수 사이의 모순

```
In [121]: north_island_name
```

```
Out[121]: ['Maud's',  
          'CafeCarberry',  
          'CostaCoffee',  
          'TheDarkHorse',  
          'CafeLePetitOrmeau',  
          'CommonGroundsCafé',  
          'CaffèNero',  
          'CaffèNero',  
          'Grounded',  
          'Relish',  
          'SinnamonCoffee',  
          'ClementsCafé',  
          'ClementsCafé',  
          'CaffèNero',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks']
```

```
In [137]: scott
```

```
Out[137]: ['TheAllanWaterCafe',  
          'Spoon',  
          'Mono',  
          'Henry'sCoffeeHouse',  
          'DobbiesGardenWorldCafe',  
          'PatisserieValerie',  
          'HulaJuiceBar',  
          'Clive's',  
          'TapaCoffeehouse',  
          'Starbucks',  
          'BrewLabCoffee',  
          'TheElephantHouse',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks']
```

```
In [145]: 웨일스
```

```
Out[145]: ['DobbiesGardenWorldCafe',  
          'SquareCafe',  
          'IBERbach',  
          'CaffèNero',  
          'Starbucks',  
          'PatisserieValerie',  
          'HulaJuiceBar',  
          'Clive's',  
          'PearTreeCafeBar',  
          'TapaCoffeehouse',  
          'TheElephantHouse',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks']
```

```
In [147]: 잉글랜드
```

```
Out[147]: ['TheAlbion',  
          'TheRidingHouseCafé',  
          'CafeGodiva',  
          'GranCaffèLondra',  
          'Cafe1001',  
          'lookmumnohands!',  
          'WorkshopCoffeeCo.',  
          'BalansSohoSociety',  
          'CafeMonde',  
          'TheTeapot',  
          'Starbucks',  
          '스타벅스',  
          'UrbanCoffeeCompany',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'Starbucks',  
          'KahveDünyası',  
          'OzoneCoffeeRoasters',  
          'ShoreditchGrind',  
          'DepartmentofCoffeeandSocialAffairs']
```

크롤링 결과,

카페 = 음식점을 겸비한 카페 혹은 개인 카페

커피숍 = 대다수가 스타벅스 혹은 커피만을 판매하는 커피 프랜차이즈

즉 커피숍이 수도권 같은 일부 지역에 분포된 이유는 커피숍 메뉴는 유명 프랜차이즈였기 때문이고,
카페가 전국에 분포된 이유는 커피를 겸비한 음식점이 포함되었기 때문

| 예측 알고리즘

120만개의 데이터를 받았고 이를 유용하게 사용하기 위해 Post_Venue 예측 알고리즘을 작성 하였습니다.

모든 FourSquare 데이터에서 사용 할 수 있을 것입니다.

| 예측 알고리즘 전처리

2. Step 생성

```
df = data.sort_values(by = ["User_ID", "Date"], ascending=[False, True])
sorted_user = df[["User_ID"]]
user_arr = sorted_user.values

step = []
j = 1
for i in range(0, len(user_arr)):
    if i == 0:
        step.append(j)
    else:
        user = user_arr[i]
        if user == user_arr[i-1]:
            j += 1
            step.append(j)
        else:
            j = 1
            step.append(j)
df["step"] = step
```

Step

동일한 User_ID가 발생시킨 데이터 수를 알 수 있습니다.

User_ID, Date로 소팅한 후 i번째 User_ID와 i+1번째 User_ID가 같을때 step이 1씩 증가 합니다.

| 예측 알고리즘 전처리

2. Formal_Venue 생성

```
#위도, 경도, 베뉴카테고리명, 스텝 으로 데이터프레임 생성
df2 = data[["Latitude (raw_POIs.txt)", "Longitude (raw_POIs.txt)",
            "Venue category name (raw_POIs.txt)", "step"]]

#Formal_Venue 구하기. 배열 생성해서 각각 Step, Lat, Lon, venue를 배열로 저장
step_arr = df2["step"].tolist()
Lat_arr = df2["Latitude (raw_POIs.txt)"].tolist()
Lon_arr = df2["Longitude (raw_POIs.txt)"].tolist()
venue_arr = df2["Venue category name (raw_POIs.txt)"].tolist()

Formal_Lat = []
Formal_Lon = []
Formal_Venue = []

for i in range(0, len(step_arr)):
    if(step_arr[i] == 1):
        Formal_Lat.append("NULL")
        Formal_Lon.append("NULL")
        Formal_Venue.append("NULL")
    else:
        Formal_Lat.append(Lat_arr[i - 1])
        Formal_Lon.append(Lon_arr[i - 1])
        Formal_Venue.append(venue_arr[i-1])

data["Formal_Lat"] = Formal_Lat
data["Formal_Lon"] = Formal_Lon
data["Formal_Venue"] = Formal_Venue
```

Formal_Venue

지금 위치에 오기전 어디서 Venue를 발생시켰는지 알 수 있습니다.

Step에 따라 $step[i] > 1$ 이라면 $step[i-1]$ 의 좌표를 가져 옵니다.

| 예측 알고리즘 전처리

3. Post_Venue 생성

```
#Post Venue 구하기, 위와 방법 비슷
step_arr = df2["step"].tolist()
venue_arr = df2["Venue category name (raw_POIs.txt)"].tolist()
lat_arr = df2["Latitude (raw_POIs.txt)"].tolist()
lon_arr = df2["Longitude (raw_POIs.txt)"].tolist()

post_Venue = []
post_Lat = []
post_Lon = []

for i in range(0, len(step_arr)-1):
    if(step_arr[i+1] == step_arr[i]):
        post_Venue.append(venue_arr[i+1])
        post_Lat.append(lat_arr[i+1])
        post_Lon.append(lon_arr[i+1])
    else:
        post_Venue.append("NULL")
        post_Lat.append("NULL")
        post_Lon.append("NULL")
post_Venue.append("NULL")
post_Lat.append("NULL")
post_Lon.append("NULL")

data["Post_Lat"] = post_Lat
data["Post_Lon"] = post_Lon
data["Post_Venue"] = post_Venue
```

Post_Venue

지금 위치에서 다음 위치를 확인 할 수 있습니다.

Step에 따라 $step[i]+1$ 이 $step[i+1]$ 과 같다면 User_ID가 발생시킨 미래 데이터가 있다는 뜻 이므로 좌표를 가져옵니다.

예측 알고리즘 전처리

3. Shape생성

```
#####Post_Venue 예측 알고리즘 작성#####  
# Formal_Venue - Now_Venue 형태로 데이터 추가 (중복처리로 유형 정리)  
df = data[["Venue category name (raw_POIs.txt)", "Formal_Venue"]]  
loc_arr = []  
for i in range(0, len(df)):  
    loc_arr.append(str(df.iloc[i][0])+"-"+str(df.iloc[i][1]))  
data["Loc_Plus"] = loc_arr  
  
# 유형처리를 위해 방금만든 데이터로 소팅 (이제 이게 원본 데이터가 될것.)  
sorted_df = data.sort_values(by=["Loc_Plus"], ascending=True)  
  
shape_arr = sorted_df[["Loc_Plus"]].values.tolist() #리스트로 반환  
  
# 유형만들기. i번째와 i-1번째의 값이 다를경우 유형 넘버를 하나 올리는 형식.  
input_arr = ["1shape"]  
a = 1  
for i in range(1, len(shape_arr)):  
    if (shape_arr[i] == shape_arr[i-1]):  
        input_arr.append(str(a)+"shape")  
    else:  
        a+=1  
        input_arr.append(str(a)+"shape")  
sorted_df["shape"] = input_arr
```

shape

Formal Venue와 지금 위치간의 관계를 유형으로 정리해 놓은 것입니다.

유형 정리를 위해 '이전-현재' Venue 형태로 문자열을 생성해 Loc_Plus로 데이터 저장

위 생성한 문자열로 Sort한 이후 step생성 방식과 유사한 방식으로 유형을 생성합니다. 반드시 Loc_Plus로 소팅한 이후 해야 합니다.

| 예측 알고리즘 전처리

4. Next_Venue 생성

```
# 예측되는 Post_Venue값을 구하기 위해 새로운 데이터 추가 shape - Post_Venue형식.
data = sorted_df #편의를 위해 다시 data로 정의한겁니다.
df2 = data[["shape", "Post_Venue"]]
next_loc = []
for i in range(0, len(df2)):
    next_loc.append(str(df2.iloc[i][0])+"-"+str(df2.iloc[i][1]))
data["next_venue(1try)"] = next_loc
```

Next_venue

각 유형별 Post_Venue를 정리해 놓은 것입니다.

| 예측 알고리즘 전처리

5. Shape과 next_venue

| | shape | next_venue(1try) |
|--------|------------|-------------------------|
| 75126 | 1shape | 1shape-Shoe Store |
| 321783 | 1shape | 1shape-Train Station |
| 76617 | 1shape | 1shape-Grocery Store |
| 483892 | 1shape | 1shape-nan |
| 449541 | 1shape | 1shape-Office |
| ... | ... | ... |
| 86613 | 67452shape | 67452shape-nan |
| 930288 | 67452shape | 67452shape-Hotel |
| 595784 | 67452shape | 67452shape-Airport |
| 396234 | 67452shape | 67452shape-Airport Gate |
| 0 | 67453shape | 67453shape-Building |

1271622 rows × 2 columns

전처리 결과

Shape과 next_venue 데이터의 결과입니다. 유형은 총 67453개로 정리됨을 볼 수 있습니다.

예측 알고리즘 결과

예측 코드

```
#예측 알고리즘
a = input("이전 Venue입력: ")
b = input("이후 Venue입력: ")
c = a+"-"+b
d = ''
df = data[["Loc_Plus", "shape"]]
for i in range(0, len(df)):
    if (c==df.iloc[i,0]):
        d = df.iloc[i,1]
        break
my_frame = pd.DataFrame({"shape_count":data["shape"].value_counts()})
df3 = data[["shape", "Post_Venue", "next_venue(1try)"]]
df4 = df3[df3['shape'] ==d]
print(d)
shape_count = my_frame.loc[d]
print(shape_count)
print(df4["next_venue(1try)"].value_counts(normalize=True)*100)
```

Formal과 지금 Venue를 입력 받아서 shape과 동일한 형태로 저장합니다.

입력받은 shape과 동일한 shape을 반복을 통해 찾습니다.

사용자가 입력한 Venue의 shape과 동일한 shape의 next_venue의 비율을 리턴하기 위해 새로운 데이터 프레임을 만들어 저장합니다.

보기 편하도록 비율로 리턴 합니다.

| 예측 알고리즘 결과

```
이전 Venue입력: Bar
이후 Venue입력: Pub
5168shape
shape_count      3385
Name: 5168shape, dtype: int64
5168shape-Pub      21.802068
5168shape-Bar      11.669129
5168shape-NULL      4.135894
5168shape-Train Station  3.929099
5168shape-Home (private) 2.570162
...
5168shape-Law School  0.029542
5168shape-Pool Hall  0.029542
5168shape-Shop & Service 0.029542
5168shape-Skating Rink 0.029542
5168shape-Nightlife Spot 0.029542
Name: next_venue, Length: 258, dtype: float64
```

결과 예시

한 사람이 Bar에서 Pub으로 갔을때
다음 행선지로 Pub을 갈 확률은 21
퍼, Bar로 갈 확률은 11퍼, 아무 데이
터도 발생시키지 않을 확률은 4퍼 입
니다.

| 후기

[융합] '데이터시각화 & 빅데이터 언어' 과목을 수강신청 할 때에 왜 이 둘이 융합 교과목인지 의문을 가졌습니다. 프로젝트 진행 후 '데이터 시각화' 과목은 '빅데이터 언어'과목 수강을 하지 않았을 때에는 제대로 활용하지 못함을 알았습니다. 전과생과 타과생 2인 팀으로 이루어진 저희 팀은 빅데이터 언어를 통해 파이썬을 처음 접했는데 데이터 시각화에 필요한 전처리 과정은 무리없이 할 수 있게 되었습니다. 따라서 이 둘이 [융합] 교과목인 것에 감사함을 먼저 표합니다.

이번 프로젝트를 진행 하면서 데이터를 다루는 것에 익숙해 졌습니다. 교수님이 정제해준 데이터가 아닌, 엑셀도 읽지 못하는 데이터를 Pandas를 통해 파이썬 언어로 처리 하면서 '빅데이터'란 무엇인지 조금은 알게 되었습니다.

<데이터 시각화 프로세스>

1. 주제에 맞게 데이터를 수집한다.
2. 데이터를 다듬는다.
3. 발견한 것에 대한 설명을 효과적으로 하기 위해 시각화를 한다.

수업을 통해 배웠던 데이터 시각화 과정 중 두번째 '데이터를 다듬는다' 부분을 대부분 파이썬 으로 진행 했습니다. 이는 간단하게 데이터 삭제, 추가 를 넘어 '새로운 데이터 수집', 목적에 맞게 데이터 수정, 예측 알고리즘 작성 등이 있었습니다. 공간 분할은 csv로, 이렇게 정제된 데이터를 Tableau로 읽어 시각화 했습니다. 모든 과정을 '데이터 시각화' 라는 과목 하나에서 배운 것이 아니라는 것을 프로젝트 진행하며 피부로 느끼게 되었습니다.

하지만 주제를 정하고 FourSquare데이터를 받았을 때 이를 전처리 하는 과정에서 주제를 폐기해야만 했습니다. (저희 주제에 맞는 데이터 수가 부족했습니다..) 다행히 받은 데이터 수가 많아 다시 주제를 잡을 수 있었습니다. 역시 세상은 계획대로 되는게 아니구나..라는걸 느꼈습니다.

2인이라는 팀 구성은 다른 조에 비해 인원이 적기 때문에 의견공유에 많은 시간을 할애 했습니다. 결과로 영국의 특성을 가장 잘 살릴 수 있는 '영국의 4 구성국 비교' 라는 것에 초점을 맞추어 새롭게 기말 프로젝트를 진행 하게 되었습니다. 영국의 여러 Venue Category 중 Pub, Cafe, Coffeeshop, Bar를 비교하며 영국 내에서의 차이점, 한국과 다른 카테고리화 등에 강한 흥미를 느꼈습니다. 나중에 영국에 가면 Pub 간판을 달고 있어도 식사가 가능한지 가보고 싶어 졌습니다.

| 프로젝트를 마치며.

감사합니다!