



제주도의 재난지원금 사용 실태 분석

- 월별/시간대별/소상공인구분별/업종별/행정구역별 -

팀명: SNU GIS

팀원: 석사과정1, 석사과정2

CONTENTS

01
서론

02
방법론

03
분석 결과

4
결론

01

서론

1. 개요 - [분석 배경]

- 재난지원금의 지급 목적: 경제 침체가 심화되고 장기화까지 우려되는 상황을 고려해 국민 생활의 안정과 위축된 경제 회복을 위한 최소한의 안전장치이자 '국민 안전망' 역할
- 재난지원금 지급 횟수: 총 2회 (4월, 8월)
- 총 재난지원금 지급 액수: 두 차례에 걸쳐 약 1100여억원 투입

노컷뉴스 | 2020.05.24. | 네이버뉴스

제주형 재난긴급생활지원금 401억원 지급

제주도 관계자는 “1차 재난긴급생활지원금 결과를 분석하는 합동평가회를 6월 중 추진할 계획”이라며 “지원금 지급에 대한 도민인식 조사, 빅데이터 등을 활용한 ...

제주형 재난지원금, 오늘까지 12만2천 가구에 401억... BBS NEWS | 2020.05.24.
1차 제주형 재난긴급생활지원금 12만3000세대 수혜 제민일보 | 2020.05.24.
제주 재난긴급지원금, 접수 한겨레 | 2020.08.11. | 네이버뉴스

관련뉴스 4건 전체보기 >



총 2회에 걸쳐 지급된 '제주형 재난지원금'

2차 제주형 재난지원금 1인 10만원 지급...24일 접수

도는 2차 재난지원금 지급 대상을 제주도의회의 재난지원금 사업 예산 의결 다음 날인 지난달 29일 기준 주민등록에 등록된 세대로, 가구의 소득 및 가구원의 직...

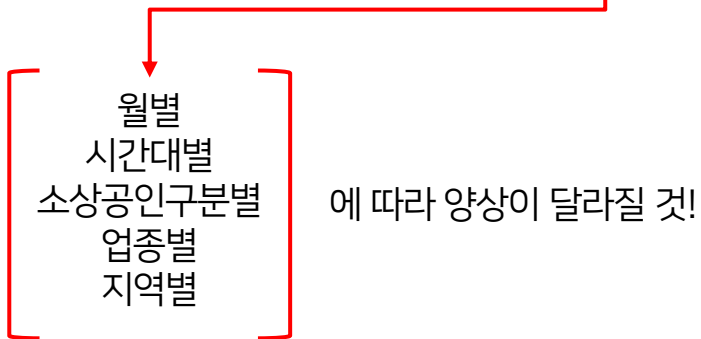
'1인당 10만원' 제주 2차 재난지원금 지급 조선일보 | 2020.08.11. | 네이버뉴스
제주도, 2차 제주형 재난긴급생활지원금 ... 프레시안 | 2020.08.11. | 네이버뉴스
제주도 '2차 제주형 재난지원금' 지급..... 아시아경제 | 2020.08.11. | 네이버뉴스
제주도, 추석 전까지 도민 1인당 10만원 ... 국민일보 | 2020.08.11. | 네이버뉴스

관련뉴스 7건 전체보기 >



1. 개요 - [분석 목적]

- Research questions:
 - 총 2회에 걸쳐 지급된 재난지원금은 **어떻게** 사용되었을까?



KDI, 보편 지급방식 1차 재난지원금 효과 분석자료 발표
긴급재난지원금 총 14조 중 소비진작 효과 4조원 그쳐
현금 93.7% 소비 지출 사용...5~6%는 저축·빚 상환
‘대면서비스업’ 효과 미미...피해업종 ‘선별지원’ 요구된다*



긴급재난지원금 CG 사진=연합뉴스

[출처: 뉴스위치]

- 카테고리 별 세분화 목적:
 - > 재난지원금의 지급 목적이 소상공인 경제 활성화, 소비 진작인만큼 사용 실태를 여러 카테고리로 세분화하여 다각도, 다방면에서 분석 해야 함. 그래야 재난지원금의 지원 효과를 본 업종과 그렇지 못한 업종을 명확히 확인할 수 있음. 재정적 지원이 정말 필요한 업종에 재난지원금이 사용됐는지 보다 세밀하게 분석하기 위함임.

02

방법론

2. 방법론 - [분석 목록]

[Macro Analysis]

- ① 소상공인 업종 대분류 별 총 사용금액 대비 재난지원금 사용금액 비교
- ② 월별 총 사용금액 대비 재난지원금 사용금액 비교
- ③ 한 시간별 재난지원금 사용금액 비교
- ④ 각 월의 소상공인구분별/업종명 별 총 사용금액 및 재난지원금 사용금액 비교



[Micro Analysis]

- ① 각 시간대에 대하여 소상공인구분별 업종 비율 (파이차트)
 - 시간대는 총 4구간으로 분할 (AM1, AM2, PM1, PM2)
- ② 시간대 별 상위 N개의 업종에 대한 재난지원금 사용금액/이용건수 정량 분석 (누적막대그래프)
- ③ 시간대 별 상위 N개의 업종에 대한 재난지원금 사용금액/이용건수 비율 분석 (파이차트)
- ④ 시간대 별 상위 N개의 업종에 대한 지리적 위치 분포 분석 (히트맵)



[Regional Analysis]

- ① 월별 제주도 읍면동리 별 재난지원금 사용금액/이용건수 지도 시각화
- ② 월별 제주도 읍면동리 별 재난지원금 사용금액/이용건수 시계열 분석



2. 방법론 - [사용 라이브러리 및 파이썬 버전]

- python==3.8.5
- bokeh==2.2.3
- folium==0.11.0
- geopandas==0.8.1
- geopy==2.0.0
- geojson==2.5.0
- geopandas==0.8.1
- geopy==2.0.0
- kaleido==0.1.0.post1
- matplotlib==3.3.3
- networkx==2.5
- numpy==1.19.4
- orca==1.5.4
- osmnx==0.16.2
- pandas==1.1.5
- plotly==4.14.1
- pyproj==3.0.0.post1
- scikit-learn==0.23.2
- seaborn==0.11.1
- selenium==3.141.0
- Shapely @ file:///C:/Users/user/Desktop/Shapely-1.7.1-cp38-cp38-win_amd64.whl
- sklearn==0.0
- tqdm==4.54.1

* 추가 업데이트는 깃헙의 README 참조: https://github.com/henewsuh/jeju_gis

2. 방법론 - [사용 데이터셋]

- ① DAICON: <https://dacon.io/competitions/official/235682/data/>
- ② 읍면동리 SHAPE 파일: <http://www.gisdeveloper.co.kr/?p=2332>
 - 읍면동 2020년 5월 파일, 리 2020 5월 파일 사용
- ③ 법정동명 및 법정동코드: <https://www.code.go.kr/stdcode/regCodeL.do>
 - 지역선택: 제주특별자치도 선택
 - [조회] 버튼 클릭
 - [사용자 검색자료] 버튼 클릭을 클릭하여 제주특별자치도의 법정동코드와 법정동명 데이터 다운로드



* 기본 데이콘 데이터셋 이외에 추가적으로 사용한 데이터셋은 깃헙의 'extra_data' 폴더 참조
: https://github.com/henewsuh/jeju_gis/tree/main/extra_data

2. 방법론 - [전처리 및 FYI]

- ① 결측값 확인: 5월 - 8월 모든 데이터에 대한 결측값 확인 결과, 결측값 존재 x
- ② 좌표 변환: epsg:5178 --> epsg:4326
- ③ 좌표 기반 도로명 주소 및 법정동 코드 검색
 - (x, y)좌표를 입력으로 주고 해당되는 도로명주소 및 법정동 코드를 출력으로 받음.
 - 읍면동리 SHAPE 파일의 지오메트리를 활용함.
- ④ FYI : 시간대 / 소상공인구분 / 업종대분류
 - 다각도의 세분화 분석을 위해 다음과 같이 임의로 카테고리를 구분, 코드를 부여함.

구분 카테고리	시간대	소상공인구분	업종대분류	
구분 내용	AM1: 00시 - 06시 AM2: 06시 - 12시 PM1: 12시 - 18시 PM2: 18시 - 24시	a: 영세 b: 일반 c: 중소 d: 중소1 e: 중소2	0: 음식 1: 소매 2: 생활서비스 3: 학문, 교육 4: 숙박 5: 관광, 여가, 오락 6: 문화, 예술, 종교	7: 도매, 유통, 무역 8: 부동산 9: 스포츠 10: 의료 11: 교통, 운송 12: 전자, 정보통신 13: 기타

2. 방법론 - 전체 분석 프로세스 개요

PART 0. Macro Analysis

- 전체 기간 업종 대분류 별 분석
- 월 별 사용액 & 재난지원금 사용률 분석
- 시간(hour) 별 지출 분석
- 업종명, 소상공인 구분 별 재난지원금 사용금액 분석

PART 1. Micro Analysis

- 시간대 별 업종 분석
 - 업종 별 점유율 분석
 - 업종 별 재난지원금 사용 비율 분석
 - 업종 별 재난지원금 관련 점유율 분석

PART 2. Regional Analysis

- 전체 구역 재난지원금 사용 관련 지도 시각화
- 지역 별 재난지원금 이용건수 시계열 분석
- 지역 별 재난지원금 사용 금액 시계열 분석

2. 방법론 - [분석 목록 별 코드 설명 - Macro Analysis]

(1) 소상공인 업종 대분류 별 총 사용금액 대비 재난지원 사용금액 비교

```
types, types_cnt = np.unique(df_all['Type'], return_counts = True)
cols = ['TotalSpent', 'DisSpent', 'NumofSpent', 'NumofDisSpent']
type_df = pd.DataFrame(columns = ['Type'] + cols + ['ratio (%)', 'perDisSpent'])

for idx, t in enumerate(types):
    cur_type = pd.Series({'Type' : t})
    cur_type = cur_type.append(df_all.loc[df_all['Type'] == t][cols].sum())
    cur_type['ratio (%)'] = round(cur_type['DisSpent']/cur_type['TotalSpent']*100, 2)
    if cur_type['NumofDisSpent'] != 0:
        cur_type['perDisSpent'] = round(cur_type['DisSpent']/cur_type['NumofDisSpent'])
    else:
        cur_type['perDisSpent'] = int(0)
    type_df.loc[idx] = cur_type

os.chdir('../')
code_table = pd.read_csv('code_table.csv', encoding = 'euc-kr', sep = ',')
type_df['code'] = code_table['code']
code_df = type_df.groupby(by = 'code').sum()
code_df['ratio (%)'] = round(code_df['DisSpent']/code_df['TotalSpent']*100, 2)
code_df['class'] = code_dict.values()
```

code	TotalSpent	DisSpent	NumofSpent	umofDisSpe	ratio (%)	class
0	2.59947e+11	1.84233e+10	8.30503e+06	607311	7.09	음식
1	1.97569e+11	8.55254e+09	9.17835e+06	408845	4.33	소매
2	2.66328e+10	1.14583e+09	573356	23811	4.3	생활 서비스
3	3.34848e+10	8.24448e+08	127898	3574	2.46	학문, 교육
4	2.66351e+10	1.48573e+08	247420	3205	0.56	숙박
5	2.62333e+10	4.0996e+08	565399	12760	1.56	관광, 여가, 오락
6	3.2487e+09	5.42909e+07	131327	2116	1.67	문화, 예술, 종교
7	1.38692e+10	9.06318e+08	130760	11897	6.53	도매, 유통, 무역
8	3.89886e+08	1.92857e+06	13023	88	0.49	부동산
9	1.71568e+10	1.70357e+07	159560	712	0.1	스포츠
10	4.56736e+10	2.80377e+09	1.31086e+06	112177	6.14	의료
11	4.6873e+10	2.30465e+09	1.11981e+06	52928	4.92	교통, 운송
12	6.17156e+08	4.06665e+07	10981	503	6.59	전자, 정보통신
13	2.09484e+09	1.00388e+08	44158	2207	4.79	기타

[출력된 code_df 데이터프레임]

- [1] 제공받은 5월 - 8월의 모든 데이터에 대하여 'Type'컬럼 별로 'TotalSpent,' 'DisSpent,' 'NumofSpent,' 'NumofDisSpent' 컬럼의 누적합계를 합산하여 type_df 생성
- [2] type_df의 'Type'컬럼에 대하여 소상공인 업종 대분류 별로 분류하여 code_df 생성 (대분류 출처: 소상공인진흥공단)
각각의 Type에 대한 업종 대분류 코드 파일 (.csv)는 깃헙 참조
- [3] (DisSpent의 합 / TotalSpent의 합)을 계산하여 소상공인 업종 대분류 별 총 사용금액 대비 재난지원금으로 지불한 사용금액의 비율 도출

2. 방법론 - [분석 목록 별 코드 설명 - Macro Analysis]

(2) 월별 총 사용금액 대비 재난지원 사용금액 비교

```
monthly_spent = pd.DataFrame(columns = ['month'] + cols + ['ratio (%)', 'perDisSpent'])
for m in [5, 6, 7, 8]:
    cur_df = dfs[m]
    cur_df = cur_df[cols].sum()
    monthly_spent = monthly_spent.append({'month': m, 'TotalSpent' : cur_df['TotalSpent'], 'DisSpent' : cur_df['DisSpent'], \
        'NumofSpent' : cur_df['NumofSpent'], 'NumofDisSpent' : cur_df['NumofDisSpent'], 'ratio (%)': \
        round(cur_df['DisSpent']/cur_df['TotalSpent']*100, 2), \
        'perDisSpent': round(cur_df['DisSpent']/cur_df['NumofDisSpent'])}, ignore_index = True)
```

[1] 제공받은 5월 - 8월의 모든 데이터에 대하여 'TotalSpent', 'DisSpent', 'NumofSpent', 'NumofDisSpent' 컬럼 별로 누적합계를 합산

[2] (DisSpent의 합 / TotalSpent의 합)을 계산하여 총 사용금액 대비 재난지원금으로 지불한 사용금액의 비율 도출

[3] (DisSpent의 합 / NumofDisSpent의 합)을 계산하여 재난지원금 1회 당 평균 재난지원금 사용금액 도출

Index	month	TotalSpent	DisSpent	NumofSpent	NumofDisSpent	ratio (%)	perDisSpent
0	5	1.68688e+11	2.41801e+10	5.15104e+06	782769	14.33	30890
1	6	1.66979e+11	9.66635e+09	5.18591e+06	384342	5.79	25150
2	7	1.79294e+11	1.34356e+09	5.74684e+06	56363	0.75	23838
3	8	1.85466e+11	5.43655e+08	5.83413e+06	18660	0.29	29135

출력된

monthly_spent
데이터프레임

2. 방법론 - [분석 목록 별 코드 설명 - Macro Analysis]

(3) 한 시간별 재난지원금 사용금액 비교

```
print("Macro 3. 시간별 지출 분석")
hours = df_all['Time'].unique()
hours_dict = dict()
han_dict = dict()
foreign_dict = dict()
pyun_dict = dict()
super_dict = dict()
princ_types = ['일반한식', '편의점', '슈퍼마켓', '서양음식']

for i in tqdm(range(len(hours))):
    hours_dict[hours[i]] = 0
    han_dict[hours[i]] = 0
    pyun_dict[hours[i]] = 0
    super_dict[hours[i]] = 0
    foreign_dict[hours[i]] = 0

for h in hours:
    cur_h = df_all.loc[df_all['Time'] == h]
    hours_dict[h] = cur_h['DisSpent'].sum()
    han_dict[h] = cur_h.loc[cur_h['Type'] == '일반한식']['DisSpent'].sum()
    pyun_dict[h] = cur_h.loc[cur_h['Type'] == '편의점']['DisSpent'].sum()
    super_dict[h] = cur_h.loc[cur_h['Type'] == '슈퍼마켓']['DisSpent'].sum()
    foreign_dict[h] = cur_h.loc[cur_h['Type'] == '서양음식']['DisSpent'].sum()

# x시 지출액
hours_dict.pop('x시', None)
han_dict.pop('x시', None)
pyun_dict.pop('x시', None)
super_dict.pop('x시', None)
foreign_dict.pop('x시', None)

def dict2df(dictionary, c1, c2):
    df = pd.DataFrame.from_dict(dictionary, orient='index')
    df = df.reset_index()
    df.columns = [c1, c2]

    return df

hours_df = dict2df(hours_dict, 'hours', 'DisSpent')
han_df = dict2df(han_dict, 'hours', 'DisSpent')
pyun_df = dict2df(pyun_dict, 'hours', 'DisSpent')
super_df = dict2df(super_dict, 'hours', 'DisSpent')
foreign_df = dict2df(foreign_dict, 'hours', 'DisSpent')

fig1 = px.pie(hours_df, values='DisSpent', names='hours', title='한시간별 재난지원금 사용금액비율')
fig1.write_image('한시간별_재난지원금_사용금액_비율.png')
```

[1] 입력데이터: 5월 - 8월까지의 모든 데이터

[2] 모든 데이터에 대하여 각 'Time'컬럼 내 유니크한 값을 추출하여 각 월의 딕셔너리의 키값으로 부여

[3] 5월 - 8월까지의 모든 데이터를 포함하는 df_all을 대상으로 각 시간에 대하여 탐색알고리즘 실행 및 'TotalSpent' 'DisSpent'컬럼의 값의 누적합계를 계산
예: df_all내에서 'Time'컬럼 값이 '00시'인 데이터들을 대상으로 'TotalSpent'와 'DisSpent'값의 누적합계를 계산 후 5월, 6월, 7월, 8월 딕셔너리에 할당

[4] 3번의 출력값을 각각 시각화

2. 방법론 - [분석 목록 별 코드 설명 - Macro Analysis]

(4) 각 월의 소상공인 구분별/업종명 별 총 사용금액 및 재난지원금 사용금액 비교

```
print('Macro 3. 월별 업종 종류별 재난지원금 사용실태 분석')
jongs = df_all['Type'].unique().tolist()

# DisSpent
may_dict = dict()
june_dict = dict()
july_dict = dict()
august_dict = dict()

for i in range(len(jongs)):
    may_dict[jongs[i]] = 0
    june_dict[jongs[i]] = 0
    july_dict[jongs[i]] = 0
    august_dict[jongs[i]] = 0

for j in jongs:
    cur_j = df_all.loc[df_all['Type'] == j]

    may_dict[j] = cur_j.loc[cur_j['YM'] == 202005]['DisSpent'].sum()
    june_dict[j] = cur_j.loc[cur_j['YM'] == 202006]['DisSpent'].sum()
    july_dict[j] = cur_j.loc[cur_j['YM'] == 202007]['DisSpent'].sum()
    august_dict[j] = cur_j.loc[cur_j['YM'] == 202008]['DisSpent'].sum()

may_df = dict2df(may_dict, 'Type', 'DisSpent')
may_df = may_df.sort_values(['DisSpent'], ascending=False)

june_df = dict2df(june_dict, 'Type', 'DisSpent')
june_df = june_df.sort_values(['DisSpent'], ascending=False)

july_df = dict2df(july_dict, 'Type', 'DisSpent')
july_df = july_df.sort_values(['DisSpent'], ascending=False)

august_df = dict2df(august_dict, 'Type', 'DisSpent')
august_df = august_df.sort_values(['DisSpent'], ascending=False)

def make_overtime bargraph(df, x, y, title, color, labels, height, color_scale, width):
    fig = px.bar(df, x=x, y=y, title=title, color=color, color_continuous_scale=color_scale,
                 labels=labels, height=height, width=width)
    fig.write_image(title + '.png')
```

[1] 입력데이터: 5월 - 8월까지의 모든 데이터

[2] 모든 데이터에 대하여 각 'Type'컬럼 내 유니크한 값을 추출하여 각 월의 딕셔너리의 키값으로 부여

[3] 5월 - 8월까지의 모든 데이터를 포함하는 df_all을 대상으로 각 Type에 대하여 탐색알고리즘 실행 및 'TotalSpent' 'DisSpent'컬럼의 값의 누적합계를 계산

예: df_all내에서 'Type'컬럼 값이 '일반한식'인 데이터들을 대상으로

'TotalSpent와 'DisSpent'값의 누적합계를 계산 후 5월, 6월, 7월, 8월 딕셔너리에 할당

[4] 3번의 출력값을 각각 시각화

2. 방법론 - [분석 목록 별 코드 설명 - Micro Analysis]

(1) 각 시간대에 대하여 소상공인구분별 업종 비율 (def pie_chart)

```
def pie_chart(df_orig, target_prop, value_prop, cut = 0.01, top = 10, title = None):  
    ...  
    df_orig: 입력 데이터 프레임  
    target_prop: 표현하고자 하는 대상 (범례로 들어감)  
    value_prop: 표현되는 값  
    cut: 얼마큼의 부분 이하는 자르기 (%)  
    top: 상위 n개 이외의 자료는 기타로 부여  
    titla: 차트 제목  
    ...  
  
    df = pd.DataFrame(columns = ['names', 'values'])  
    if target_prop == value_prop:  
        df['names'] = df_orig[target_prop].value_counts().keys()  
        df['values'] = df_orig[target_prop].value_counts().values  
    else:  
        df['names'] = df_orig[target_prop]  
        df['values'] = df_orig[value_prop]  
  
    df.sort_values(by=['values'])  
  
    if cut:  
        df.loc[df['values']/df['values'].cumsum() < cut, 'names'] = '기타'  
    if top:  
        critria = df.iloc[top - 1]['values']  
        df.loc[df['values'] < critria, 'names'] = '기타'  
  
    fig = px.pie(df, values='values', names='names', title = title)  
    # fig2 = go.Figure(data = [go.Pie(labels = df.names, values = df.values)])  
    return df, fig
```

[1] 입력데이터: 특정 시간대의 특정 소상공인 데이터프레임

예: df_AM1_a 데이터프레임이 입력으로 들어왔다면 00시 - 06시의 영세소상공인의 데이터프레임

[2] target_property: 범례로 입력되는 대상 (예: 업종)

value_property: 각 타겟의 값 (예: 타겟이 전체에서 차지하는 비율)

[3] target_property와 value_property가 동일하면 해당 타겟의 전체 대비 비율이 계산 됨

* cut: 특정 타겟의 비율이 n% 미만이면 누락 (현재 n = 1)

* top: 시각화에 사용될 상위 업종의 개수 (현재 top = 10)

* cut과 top에서 누락된 값은 기타로 할당

2. 방법론 - [분석 목록 별 코드 설명 - Micro Analysis]

(2) 시간대 별 상위 N개의 업종에 대한 재난지원금 사용금액/이용건수 정량 분석 (def stacked_bar)

```
def stacked_bar(df_orig, x, cols, sum_col = None, name_used = False, title = False):
    """
    df_orig: 입력 데이터 프레임
    x: x 축에 들어갈 카테고리 정보(요일 등)
    cols: 누적할 value 할 이름 (리스트)
    sum_col: cols에 이미 누적된 데이터가 있을 경우 (0: TotalSpent)
    name_used: x의 값에 사용금액을 합소가 아닌 합계가 있는 경우 (리스트)
    title: 차트 제목
    """

    if sum_col:
        y = sum_col
        columns = [x]
        for c in cols:
            if c == sum_col:
                columns.append(sum_col + ' - others')
            else:
                columns.append(c)
        columns.append(y)
    else:
        y = 'sum'
        columns = [x]
        for c in cols:
            columns.append(c)
        columns.append(y)

    df = pd.DataFrame(columns = columns)
    df[x] = df_orig[x]
    for c in cols:
        df[c] = df_orig[c]

    df = df.groupby(by = 'Type').sum()

    if name_used:
        for name in df.index:
            if name not in name_used:
                df = df.drop(name)

    if sum_col:
        df[sum_col + ' - others'] = 2*df[sum_col] - df.sum(axis = 1)
    else:
        df[y] = df.sum(axis = 1)
    df = df.reset_index()
    df = df.sort_values(by=[y], ascending=False)
    for col in df.columns:
        if df[col].dtype not in ['object', 'str']:
            df[col] = df[col]/1000

    if sum_col:
        y_s = list(df.columns)
        y_s.remove(x)
        y_s.remove(y)
        fig = px.bar(df, x=x, y=y_s, title = title)

        data = []
        for i in y_s:
            data.append(go.Bar(x = df[x], y = df[i], text = df[i], name = i))
        layout = go.Layout(barmode='stack', plot_bgcolor='rgba(0,0,0,0)', title=title)
        total_labels = [{"x": x, "y": total + 100, "text": str(total), "showarrow": False} for x, total in zip(df[x], df[sum_col])]
        fig = go.Figure(data=data, layout=layout)
        fig = fig.update_layout(annotations=total_labels)
    else:
        fig = px.bar(df, x=x, y=cols, title = title)

    return fig
```

[1] 입력데이터: 특정 시간대의 특정 소상공인 데이터프레임

예: df_AM1_a 데이터프레임이 입력으로 들어왔다면 00시 - 06시의 영세소상공인의 데이터프레임

[2] name_used: 이전 파이차트에서 도출된 전체 대비 비율이 상위 10개인 업종

[3] name_used에 속한 상위 10개 업종에 대해 'TotalSpent'와 'DisSpent' 컬럼 별 누적합산을 계산

[4] 3번의 결과를 수직누적막대그래프로 시각화

2. 방법론 - [분석 목록 별 코드 설명 - Micro Analysis]

(3) 시간대 별 상위 N개의 업종에 대한 재난지원금 사용금액/이용건수 비율 분석 (def horizontal_stacked_bar)

```
def horizontal_stacked_bar(df_orig, y_s, cut = 0.03, sum_col = None, name_used = False, title = False):  
    cols = ['Type']  
    for y in y_s:  
        cols.append(y)  
  
    df = pd.DataFrame(columns = cols)  
  
    for c in cols:  
        df[c] = df_orig[c]  
    df = df.groupby(by = 'Type').sum()  
    if name_used:  
        for name in df.index:  
            if name not in name_used:  
                df = df.drop(name)  
  
    for y in y_s:  
        df[y] = df[y] / df[y].sum()  
  
    x_data = []  
    y_data = ['총 사용 금액', '재난 지원금', '총 사용 건수', '재난 지원금<br>사용 건수']  
    for c in df.columns:  
        x_data.append(df[c])  
    np.random.seed(42)  
    rand = np.random.uniform(-100, 100, 100)  
    fig = go.Figure()  
    for i in range(0, len(x_data[0])):  
        cur_color = colors[-1 * i]  
        fig.add_trace(go.Bar(  
            y = y_data,  
            x = df.iloc[i].tolist(),  
            name = df.iloc[i].name,  
            orientation='h',  
            marker=dict(  
                color = cur_color,  
                line=dict(color='rgb(45, 45, 45)', width=1)  
            )  
        ))  
    fig.update_layout(title = title, barmode = 'stack', paper_bgcolor='rgb(255, 255, 255)',  
        plot_bgcolor='rgb(255, 255, 255)', margin=dict(l=30, r=40, t=30, b=30), showlegend=True)  
  
    annotations = []  
    for yd, xd in zip(y_data, x_data):  
        if xd[0] > cut:  
            annotations.append(dict(xref='x', yref='y',  
                x=xd[0] / 2, y=yd,  
                text=str(int(round(xd[0]*2 * 100)) + '%',  
                    font=dict(family='Arial', size=14,  
                        color='rgb(67, 67, 67)'),  
                showarrow=False))  
  
            space = xd[0]  
            for i in range(1, len(xd)):  
                # labeling the rest of percentages for each bar (x_axis)  
                if xd[i] > cut:  
                    annotations.append(dict(xref='x', yref='y',  
                        x=space + (xd[i]/2), y=yd,  
                        text=str(int(round(xd[i]*2 * 100)) + '%',  
                            font=dict(family='Arial', size=14,  
                                color='rgb(67, 67, 67)'),  
                        showarrow=False))  
  
                    space += xd[i]  
    fig.update_layout(annotations=annotations)  
    return fig
```

[1] 입력데이터: 특정 시간대의 특정 소상공인 데이터프레임

예: df_AM1_a 데이터프레임이 입력으로 들어왔다면 00시 - 06시의 영세 소상공인의 데이터프레임

[2] name_used: 이전 파이차트에서 도출된 전체 대비 비율이 상위 10개 인 업종

[3] name_used에 속한 상위 10개 업종에 대해 'TotalSpent',

'DisSpent', 'NumofSpent', 'NumofDisSpent' 컬럼 별 비율을 계산

예: 편의점에서 재난지원금의 이용건수 비율, 총 사용건수 비율, 재난지원금 사용금액 비율, 총 사용금액 비율

[4] 3번의 결과를 수평누적막대그래프로 시각화

2. 방법론 - [분석 목록 별 코드 설명 - Micro Analysis]

(4) 시간대 별 상위 N개의 업종에 대한 지리적 위치 분포 분석 (def gis_analysis)

```
def gis_analysis(df, label, map_path, fc_class, month, time_s):  
    filepath = map_path  
    center = [33.3989, 126.60]  
  
    df_5a = df  
  
    df_total_spent_top = df_5a.sort_values(by='TotalSpent', ascending=False).groupby('FrncClass').head(100)  
    labels = df_total_spent_top['Type'].unique().tolist()  
  
    df_disspent_top = df_5a.sort_values(by='DisSpent', ascending=False).groupby('FrncClass').head(100)  
    llabels = df_total_spent_top['Type'].unique().tolist()  
  
    df_num_total_spent_top = df_5a.sort_values(by='NumofSpent', ascending=False).groupby('FrncClass').head(100)  
    llabels = df_total_spent_top['Type'].unique().tolist()  
  
    df_num_disspent_top = df_5a.sort_values(by='NumofDisSpent', ascending=False).groupby('FrncClass').head(100)  
    llabels = df_total_spent_top['Type'].unique().tolist()  
  
    def df_spent_sum(df_5z, labels):  
        df_5a_finance = pd.DataFrame(data=[], columns=['Type', 'TotalSpent', 'DisSpent', 'NumofSpent', 'NumofDisSpent'])  
        def spent_sum(df_5z, typee):  
            total_spent_ls = []  
            dis_spent_ls = []  
            num_spent_ls = []  
            num_disspent_ls = []  
  
            for i in range(len(df_5z)):  
                tt = df_5z.iloc[i]['Type']  
  
                if tt in labels:  
                    if tt == typee:  
                        total_spent = df_5z.iloc[i]['TotalSpent']  
                        total_spent_ls.append(total_spent)  
  
                        dis_spent = df_5z.iloc[i]['DisSpent']  
                        dis_spent_ls.append(dis_spent)  
  
                        num_spent = df_5z.iloc[i]['NumofSpent']  
                        num_spent_ls.append(num_spent)  
  
                        num_disspent = df_5z.iloc[i]['NumofDisSpent']  
                        num_disspent_ls.append(num_disspent)  
  
            sum_total_spent = sum(total_spent_ls)  
            sum_disspent = sum(dis_spent_ls)  
            sum_num_spent = sum(num_spent_ls)  
            sum_num_disspent = sum(num_disspent_ls)  
  
            return sum_total_spent, sum_disspent, sum_num_spent, sum_num_disspent  
  
        for i in range(len(labels)):  
            sum_total_spent, sum_disspent, sum_num_spent, sum_num_disspent = spent_sum(df_5z, labels[i])  
            df_5a_finance.loc[i] = [labels[i], sum_total_spent, sum_disspent, sum_num_spent, sum_num_disspent]  
  
    return df_5a_finance
```

[1] 입력데이터: 특정 시간대의 특정 소상공인 데이터프레임

예: df_AM1_a 데이터프레임이 입력으로 들어왔다면 00시 - 06시의 영세소상공인의 데이터프레임

[2] name_used: 이전 파이차트에서 도출된 전체 대비 비율이 상위 10개업종

[3] name_used에 속한 상위 10개 업종에 대해 'TotalSpent', 'DisSpent', 'NumofSpent', 'NumofDisSpent' 컬럼 별 상위 100개의 좌표 추출

예: 재난지원금 사용금액이 53번째로 높은 일반 한식 업종의 좌표

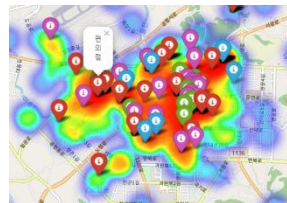
[4] 입력데이터를 기반으로 히트맵을 그리고, 3번의 좌표점들을 pin point 형태로 시각화

* 사용금액 상위 100개: 초록색

* 재난지원금 사용금액 상위 100개: 보라색

* 이용건수 상위 100개: 파란색

* 재난지원금 이용건수 상위 100개: 빨간색



2. 방법론 - [분석 목록 별 코드 설명 - Regional Analysis]

(1) 월별 제주도 읍면동리 별 재난지원금 사용금액/이용건수 지도 시각화 (def emdli_gis_analysis)

```
def emdli_gis_analysis(df_d, month, map_path, extra_data_path, preprocessed):
    """
    1. 읍면동리 .shp 파일 출처: http://www.gisdeveloper.co.kr/?p=2332
       - 읍면동 2020년 5월 파일, 리 2020 5월 파일 다운로드
    2. 법정동명 및 코드 데이터 출처: https://www.code.go.kr/stdcode/regCode.do
       - 지역선택, 제주특별자치도 선택
       - [조회] 버튼 클릭
       - [사용자 검색자료] 버튼 클릭을 클릭하여 제주특별자치도와 법정동명 데이터 다운로드
    """
    os.chdir(extra_data_path)

    # 읍면동 리 파일 불러오기
    # end = gpd.read_file('end.shp', encoding = 'euc-kr')
    # li = gpd.read_file('li.shp', encoding = 'euc-kr')

    # 제주도만 뽑아내고 한치기 (읍과 면은 제외)
    # emdli = gpd.GeoDataFrame(columns = ['CODE', 'ENG_NM', 'KOR_NM'], geometry = [])
    # for i, row in end.iterrows():
    #     if row['END_CD'][:2] == '50' and row['END_KOR_NM'][-1] != '읍' and row['END_KOR_NM'][-1] != '면':
    #         emdli = emdli.append({'CODE': row['END_CD'], 'ENG_NM': row['END_ENG_NM'], 'KOR_NM': row['END_KOR_NM'], 'geometry': row['geometry']})
    # for i, row in li.iterrows():
    #     if row['LI_CD'][:2] == '50':
    #         emdli = emdli.append({'CODE': row['LI_CD'], 'ENG_NM': row['LI_ENG_NM'], 'KOR_NM': row['LI_KOR_NM'], 'geometry': row['geometry']})

    # 좌표계 변환 (utm-k -> CRS80)
    emdli.crs = 'EPSG:5178'
    # jeju = emdli.to_crs('EPSG:4326')
    # jeju.to_file('jeju.shp', encoding = 'utf-8')

    # 지도 불러오기
    jeju = gpd.read_file('jeju.shp', encoding = 'utf-8')
    df_new_crs = df_d

    # 각 거래기록의 좌표를 기준으로 주소와 행정구역 코드 부여
    if preprocessed == False:
        coords = []
        for i in tqdm(range(len(df_d))):
            row = df_d.iloc[i]
            if (row['POINT_X'], row['POINT_Y']) not in coords:
                coords.append((row['POINT_X'], row['POINT_Y']))
        ADDs = dict()
        for k in tqdm(range(len(coords))):
            for i in range(len(jeju)):
                cur_point = Point(coords[k])
                if jeju.iloc[i]['geometry'].contains(cur_point):
                    ADDs[k] = (coords[k], jeju.iloc[i]['CODE'], jeju.iloc[i]['KOR_NM'])
                    break
            if i == len(jeju) - 1:
                ADDs[k] = (coords[k], 'NaN', 'NaN')
        ADD_dict = dict()
        CODE_dict = dict()
```

[1] 입력데이터

- 특정 한 달의 데이터프레임 (예: 5월 데이터)
- 읍면동, 리 SHAPE 파일

[2] 읍면동, 리 SHAPE 파일에서 동과 리의 상위개념인 읍, 면에 대한 지오메트리 삭제

[3] 데이터셋의 (x, y)좌표와 읍면동 리 SHAPE파일의 GEOMETRY사이의 공간 비교 연산을 시행해 각 좌표점의 행정구역과 지오코드 할당

[4] 각각의 동리 법정명 코드에 대해 'TotalSpent', 'DisSpent', 'NumofSpent', 'NumofDisSpent' 컬럼 별로 누적합계를 합산

[5] 각 컬럼에 결과를 지도에 표현, HTML 지도 출력

2. 방법론 - [분석 목록 별 코드 설명 - Regional Analysis]

(2) 월별 제주도 읍면동리 별 재난지원금 사용금액/이용건수 시계열 분석 (def part_2)

```
def part_2 (all_sum_dict, month, mainpath, data_path, extra_data_path, map_path):

    # 6. 법정동코드 및 법정동명 엑셀 파일 불러오기 및 전처리
    bjd_code = pd.read_excel(extra_data_path + '/BJD_CODE.xls')
    bjd_copy = bjd_code.copy()
    bjd_code_ = bjd_preprocessing(bjd_code)

    for i in range(len(all_sum_dict)):
        cur_month = str(i+5)
        cur_dict = all_sum_dict[cur_month]
        for k, v in cur_dict.items():
            sum_code = k
            jj_adrs = bjd_code['법정동명'][bjd_code['법정동코드'] == sum_code].values[0]
            cur_dict[k].append(jj_adrs)

    # 빈으로 저장
    os.chdir('..../')
    os.chdir('..../')
    location_stat_path = os.path.join(os.getcwd(), '지역별_통계')
    write_data(all_sum_dict, '{}_all_sum_dict'.format(month))

    all_sum_df = pd.DataFrame.from_dict(all_sum_dict, orient='index').T
    all_sum_df = all_sum_df.reset_index()
    all_sum_df = all_sum_df.rename(columns = {'index' : 'CODE'})
    months = [i for i in all_sum_dict.keys()]

    if not os.path.exists(location_stat_path):
        os.mkdir(location_stat_path)

    os.chdir(location_stat_path)

    fig1_ls = [] #금액
    fig2_ls = [] #건수

    for i in range(len(all_sum_dict['5'])):
        area = all_sum_df.iloc[i]['5'][4]

        total_spent_5 = all_sum_df.iloc[i]['5'][0]/1000
        dis_spent_5 = all_sum_df.iloc[i]['5'][1]/1000
        num_spent_5 = all_sum_df.iloc[i]['5'][2]
        num_disspent_5 = all_sum_df.iloc[i]['5'][3]

        total_spent_6 = all_sum_df.iloc[i]['6'][0]/1000
        dis_spent_6 = all_sum_df.iloc[i]['6'][1]/1000
        num_spent_6 = all_sum_df.iloc[i]['6'][2]
        num_disspent_6 = all_sum_df.iloc[i]['6'][3]
```

[1] 입력데이터

- 모든 월의 법정동 별 'TotalSpent', 'DisSpent', 'NumofSpent', 'NumofDisSpent' 누적합계 데이터프레임 (all_sum_dict)
- 법정동 코드 엑셀

[2] 법정동 코드 엑셀 파일에서 각 all_sum_dict의 법정동코드에 해당하는 읍면동리 주소 추출 및 부여

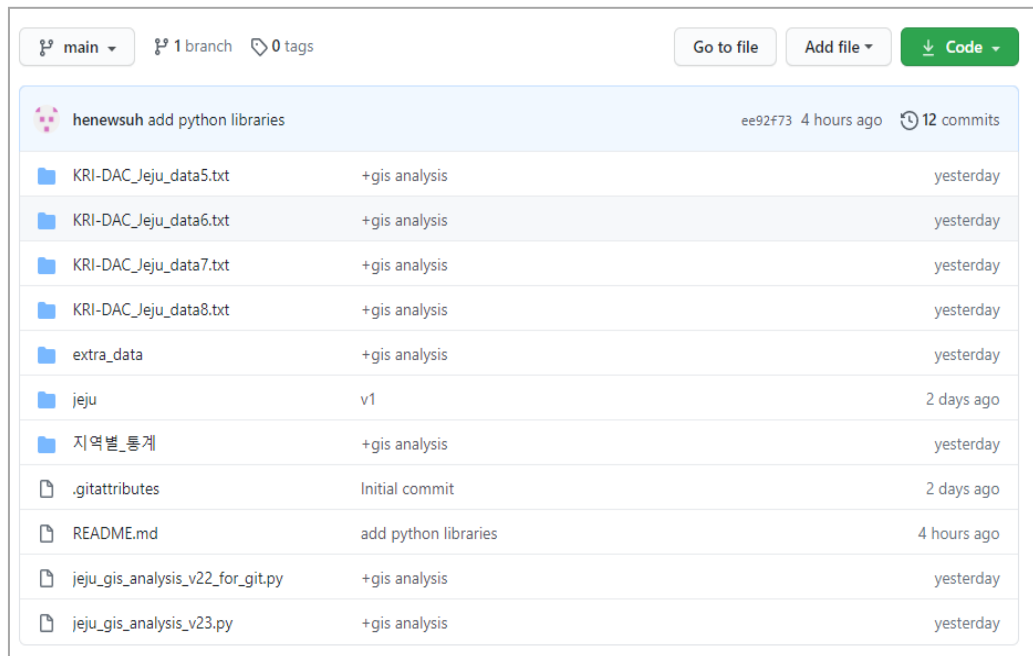
[3] 2번의 데이터 프레임을 금액과 건수 별로 막대그래프를 생성

예: 제주도 서귀포시 남원읍 남원리의 월별 결제건수 대비 재난지원금 이용건수 비교

03

분석 결과

3. 분석 결과 - [깃합]



The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with 'main' branch selected, '1 branch', and '0 tags'. Below this, the repository name 'henewsuh add python libraries' is displayed with commit hash 'ee92f73' and '4 hours ago' and '12 commits'. The main content is a list of files and folders. The first six items are folders named 'KRI-DAC_Jeju_data5.txt' through 'KRI-DAC_Jeju_data8.txt', 'extra_data', and 'jeju', all with '+gis analysis' and 'yesterday' or '2 days ago' timestamps. The next item is a folder named '지역별_통계' with '+gis analysis' and 'yesterday'. The last three items are files: '.gitattributes' (Initial commit, 2 days ago), 'README.md' (add python libraries, 4 hours ago), and 'jeju_gis_analysis_v22_for_git.py' (yesterday). The last file is 'jeju_gis_analysis_v23.py' (yesterday).

File/Folder	Commit Message	Time
KRI-DAC_Jeju_data5.txt	+gis analysis	yesterday
KRI-DAC_Jeju_data6.txt	+gis analysis	yesterday
KRI-DAC_Jeju_data7.txt	+gis analysis	yesterday
KRI-DAC_Jeju_data8.txt	+gis analysis	yesterday
extra_data	+gis analysis	yesterday
jeju	v1	2 days ago
지역별_통계	+gis analysis	yesterday
.gitattributes	Initial commit	2 days ago
README.md	add python libraries	4 hours ago
jeju_gis_analysis_v22_for_git.py	+gis analysis	yesterday
jeju_gis_analysis_v23.py	+gis analysis	yesterday

[Teslanaires의 깃합 페이지 캡처]

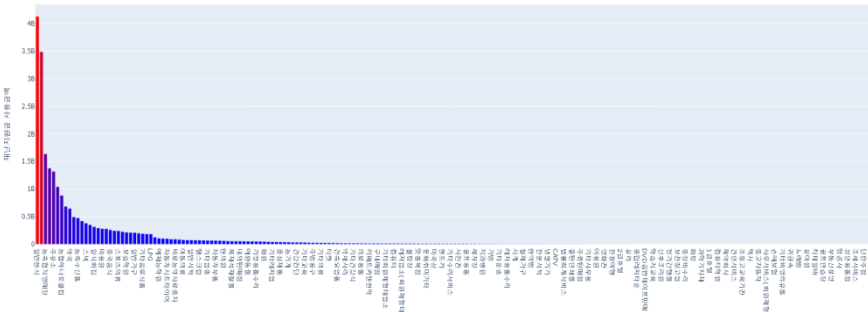


- ▶ 월별/시간대별/소상공인구분별/업종별/지역별로 자세한 결과 이미지는 깃합 페이지에 정리해두었습니다.
- ▶ 더불어, 전체 코드와 저희가 사용한 추가 데이터셋도 업로드 해 두었으니 참고 바랍니다.
- ▶ 본 PPT에선 재난지원금 사용 건수가 가장 높았던 5월에 대한 결과를 집중적으로 공유하도록 하겠습니다.

3. 분석 결과 - [각 월의 업종명 별 총 사용금액 및 재난지원금 사용금액 비교]

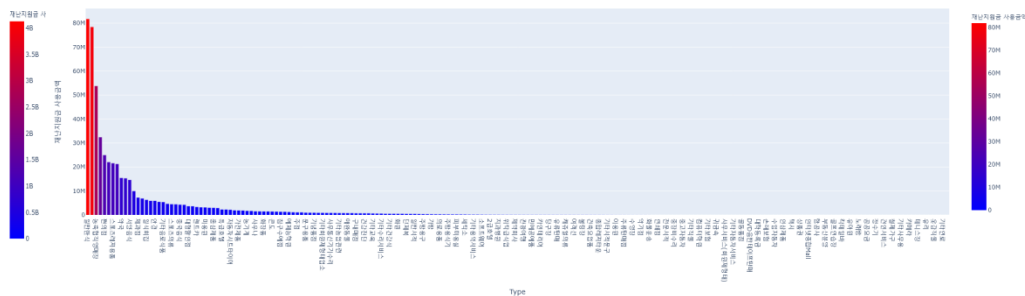
5월 업종명 별 재난지원금 사용금액

5월 각 업종명 별 재난지원금 사용금액



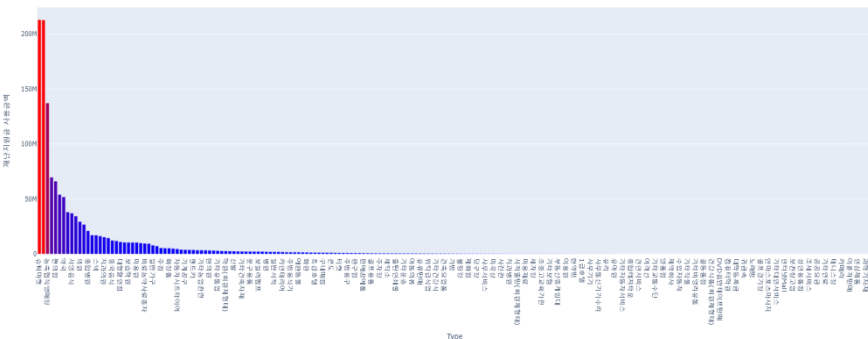
6월 업종명 별 재난지원금 사용금액

6월 각 업종명 별 재난지원금 사용금액



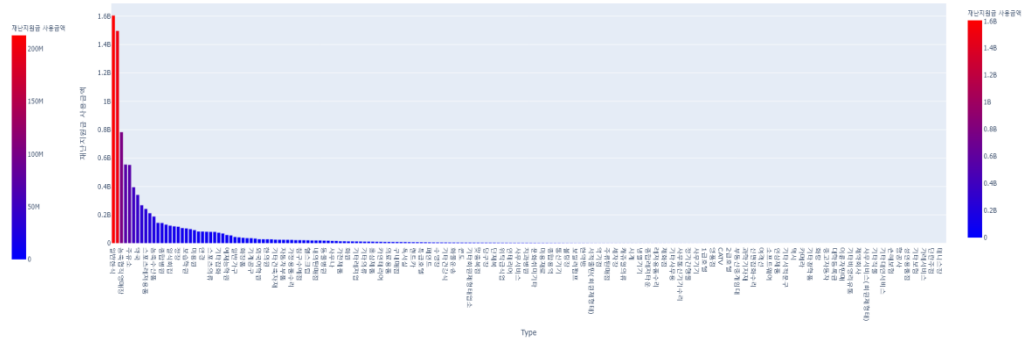
7월 업종명 별 재난지원금 사용금액

7월 각 업종명 별 재난지원금 사용금액



8월 업종명 별 재난지원금 사용금액

8월 각 업종명 별 재난지원금 사용금액

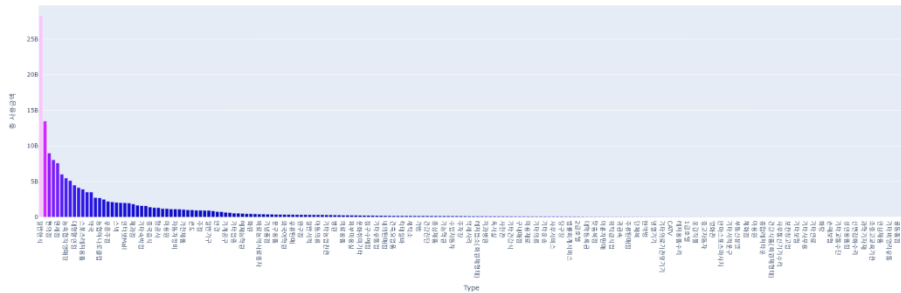


- 5월부터 8월까지 높은 재난지원금 사용금액을 기록한 업종들에는 일반한식, 편의점, 슈퍼마켓, 농협마트, 주유소등이 있었다. 특히 일반한식과 농축협직영매장은 모든 달에서 상위 1등과 2등을 다투는 재난지원금 사용처인 것으로 나타났다.

3. 분석 결과 - [각 월의 업종명 별 총 사용금액 및 재난지원금 사용금액 비교]

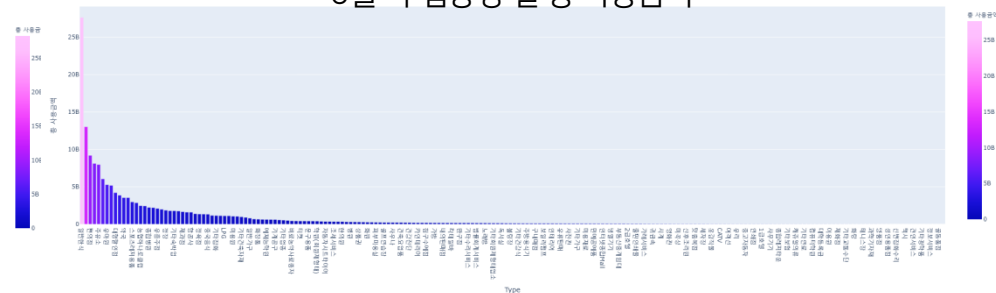
5월 업종명 별 총 사용금액

5월 각 업종명 별 총 사용금액



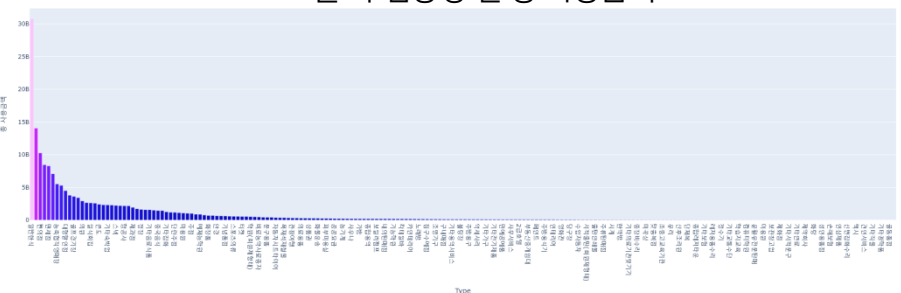
6월 업종명 별 총 사용금액

6월 각 업종명 별 총 사용금액



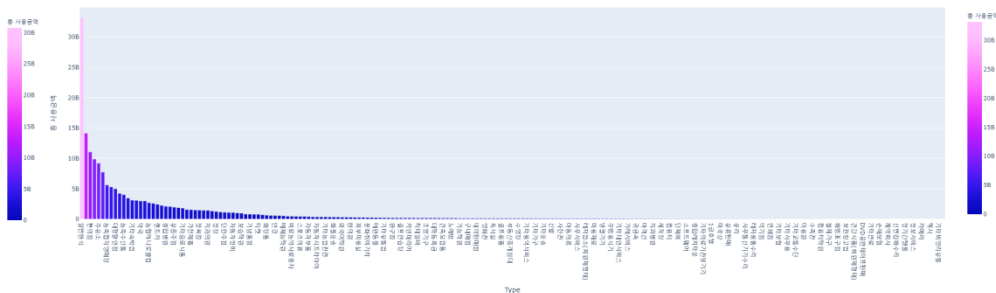
7월 업종명 별 총 사용금액

7월 각 업종명 별 총 사용금액



8월 업종명 별 총 사용금액

8월 각 업종명 별 총 사용금액



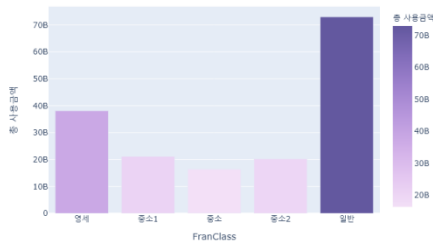
- 5월부터 8월까지 소비금액이 가장 큰 업종 중 앞 장의 재난지원금 카테고리과 다른 업종은 면세점이었다. 일반적인식, 편의점, 주유소, 농협관련마트는 재난지원금 사용금액이 높은 업종이면서 일반 사용금액이 높은 업종으로 나타났다.

3. 분석 결과 - [월 별 소상공인구분 별 사용금액 및 재난지원금 사용금액]

총 사용금액

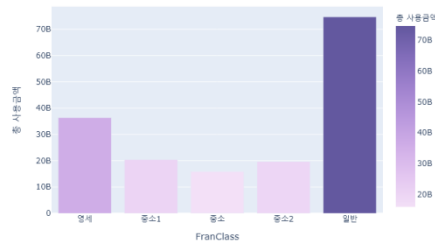
5월

5월 소상공인구분 별 총 사용금액



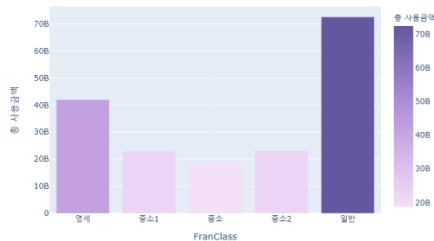
6월

6월 소상공인구분 별 사용금액



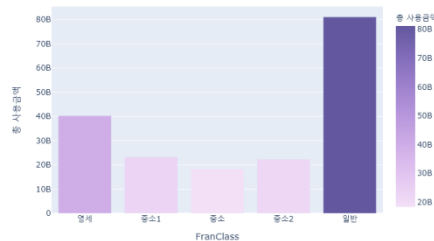
7월

7월 소상공인구분 별 총 사용금액



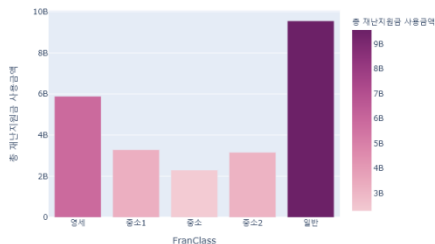
8월

8월 소상공인구분 별 총 사용금액

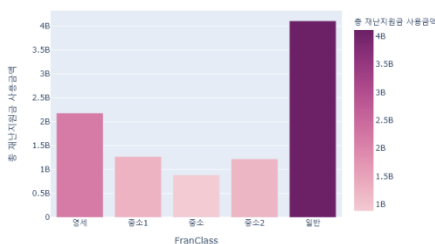


재난지원금 사용금액

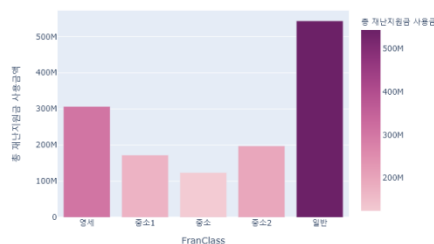
5월 소상공인구분 별 총 재난지원금 사용금액



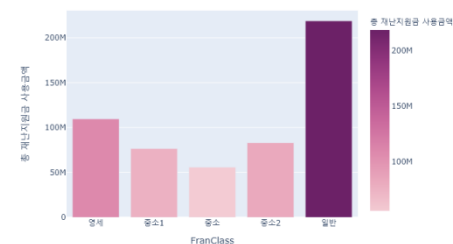
6월 소상공인구분 별 총 재난지원금 사용금액



7월 소상공인구분 별 총 재난지원금 사용금액



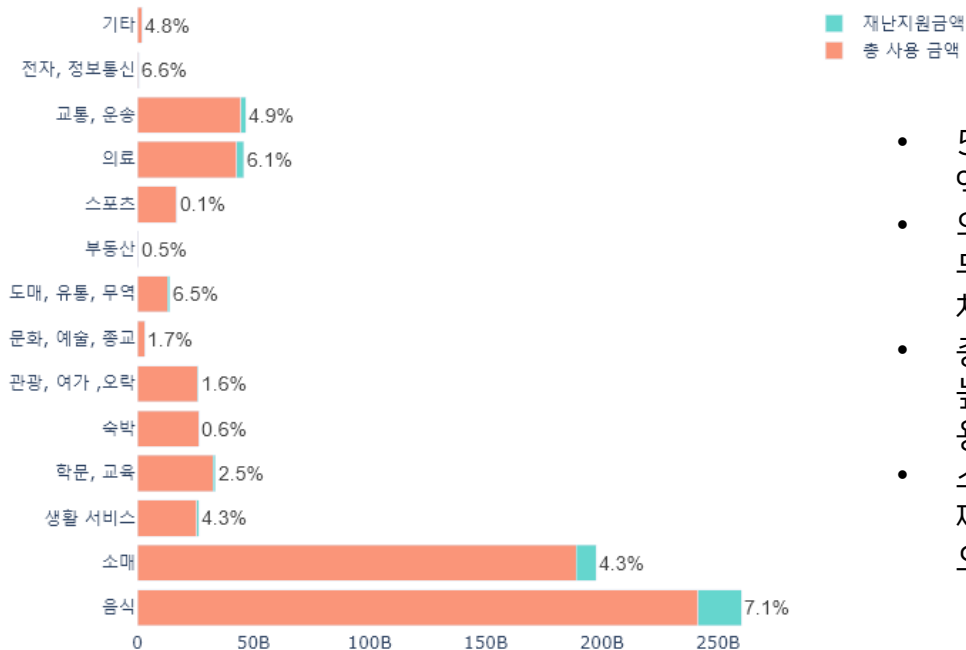
8월 소상공인구분 별 총 재난지원금 사용금액



- 5월 - 8월까지 일반 사용금액과 재난지원금 사용금액이 가장 높은 소상공인은 일반으로 나타났다. 영세소상공인은 두 번 째로 높은 재난지원금 사용처였으며, 소상공인구분 별 사용금액 및 재난지원금 사용금액 의 양상이 모든 월에 걸쳐 비슷한 양상을 띄었다.

3. 분석 결과 - [소상공인 업종 대분류별 총 사용금액 대비 재난지원금 사용 금액 비교]

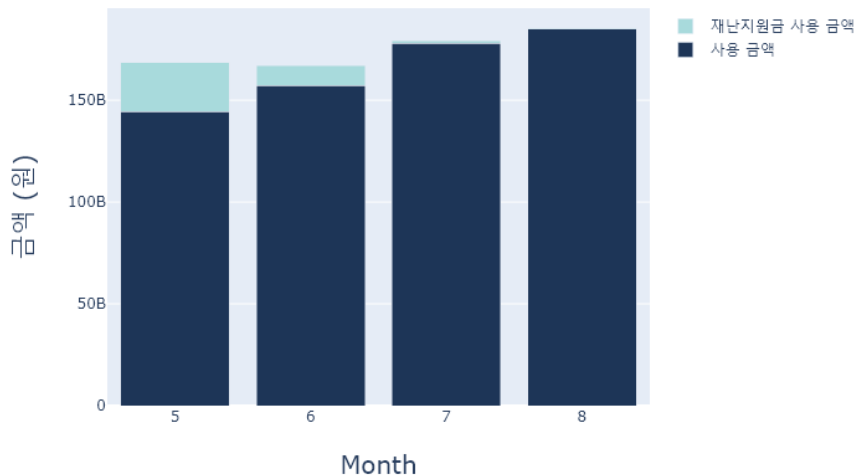
업종 대분류 별 사용 금액 (%: 재난지원금의 비율)



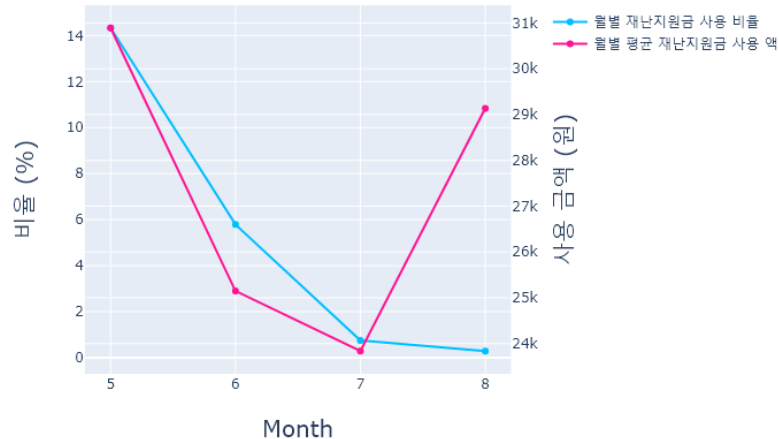
- 5월 ~ 8월을 통틀어 총 사용금액 대비 재난지원금 사용금액이 가장 높은 업종 대분류는 '음식'이다.
- 의료 부문은 사용금액 대비 재난지원금 사용금액 비율이 두 번째로 높지만 총 사용금액과 재난지원금 사용금액 자체는 작은 편이다.
- 총 사용금액 대비 재난지원금 사용금액 비율이 세 번째로 높은 교통, 운송 (주유) 또한 총 사용금액과 재난지원금 사용금액 자체는 작은 편이다.
- 소매업은 총 사용금액 대비 재난지원금 사용금액이 네 번째로 높지만 재난지원금 사용금액 자체는 음식 부문 다음으로 두 번째를 차지했다.

3. 분석 결과 - [월별 총 사용금액 대비 재난지원금 사용금액 비교]

월별 사용금액

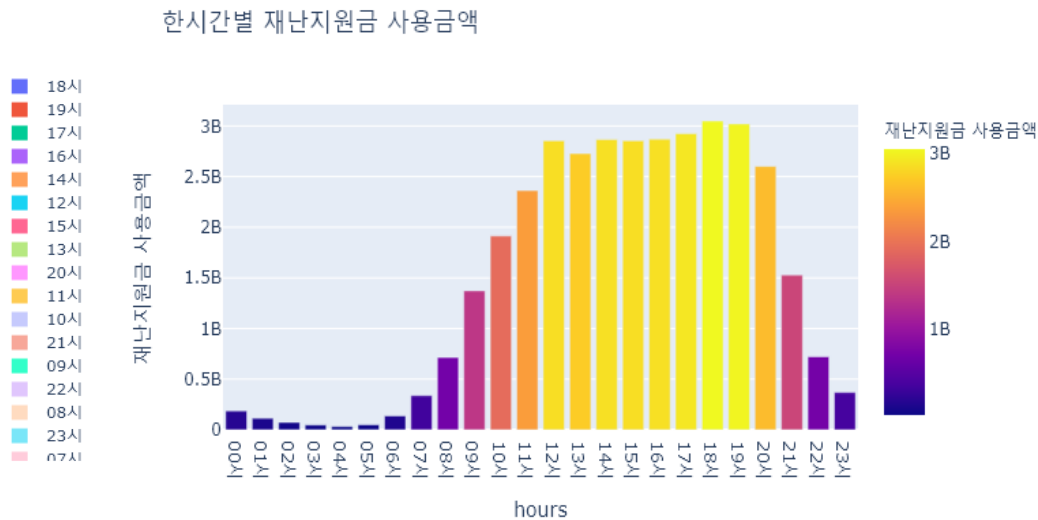
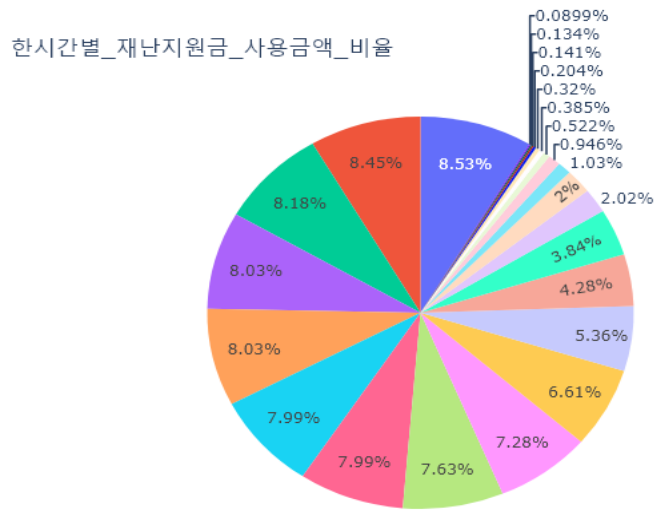


월별 재난지원금 사용금액 비율/평균 사용 액



- 좌측의 누적막대그래프를 보면 5월부터 8월까지의 소비금액은 꾸준히 증가했지만, 재난지원금 사용금액은 5월달부터 8월까지 점점 증가하는 추세를 보였다. 이에 따라 재난지원금 지급으로 인한 효과는 5월이 가장 높았을 것으로 추정된다.
- 우측의 선 그래프에서도 확인할 수 있다시피 5월부터 8월까지 총 사용금액 대비 재난지원금의 사용금액의 비율이 점점 감소하는 것을 확인할 수 있다. 5월에는 총 사용금액 대비 재난지원금의 사용금액의 비율이 14%이었던 것에 반해 8월에는 2%미만까지 감소하였다.
- 반면 월별 평균 재난지원금 결제 건당 사용금액 (한 번 재난지원금을 이용할 때마다 결제하는 재난지원금 금액)은 5월달이 약 31,000 원으로 가장 높았고 7월까지 다소 감소하다가 8월에는 약 29,000원으로 증가하는 것을 확인했다.

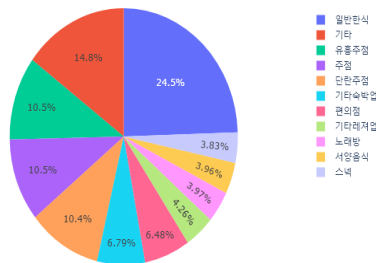
3. 분석 결과 - [시간 별 재난지원금 사용금액 비율 비교]



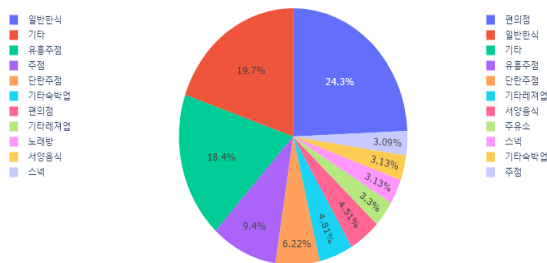
- 시간대 별 재난지원금 사용금액의 비율을 분석한 결과, 오전 10시부터 사용금액이 점차 증가하기 시작하여 오후 8시부터는 새벽까지 서서히 감소하는 양상을 보임.
- 재난지원금이 가장 활발히 사용되는 시간대는 점심시간이 시작되는 오후 12시부터 오후 7시 저녁시간대인 것으로 나타남.

3. 분석 결과 - [5월 새벽시간대의 소상공인구분 별 업종 비율 비교]

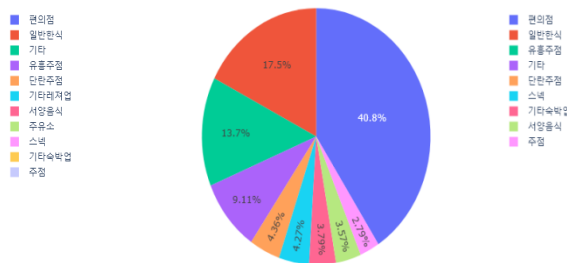
5월 0시 ~ 6시 제주도 영세 업종 별 파이



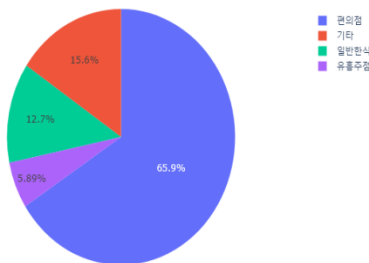
5월 0시 ~ 6시 제주도 일반 업종 별 파이



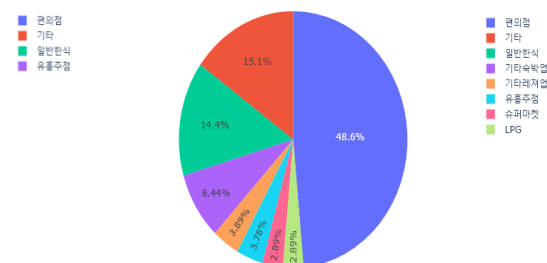
5월 0시 ~ 6시 제주도 중소 업종 별 파이



5월 0시 ~ 6시 제주도 중소1 업종 별 파이



5월 0시 ~ 6시 제주도 중소2 업종 별 파이

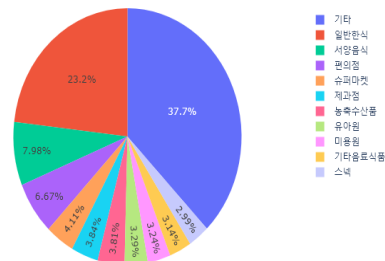


- 새벽시간대: 00시 - 06시
- 새벽시간대에는 연매출 3억 이하의 영세를 제외한 일반, 중소, 중소1, 중소2에서 편의점의 기록 건수가 가장 높음.
- 연매출 3억 이하의 영세에서는 일반한식이 가장 높은 비율을 차지함.
- 모든 소상공인 구분 별 일반한식과 편의점은 모두 상위권에 위치함.

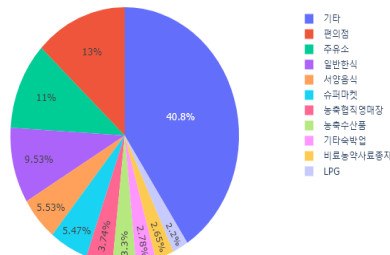
> 새벽시간대에는 편의점 이용이 가장 활발한 시간대임을 확인할 수 있음.

3. 분석 결과 - [5월 아침시간대의 소상공인구분 별 업종 비율 비교]

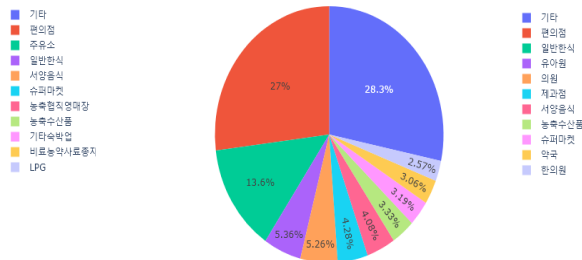
5월 6시 ~ 12시 제주도 영세 업종 별 파이



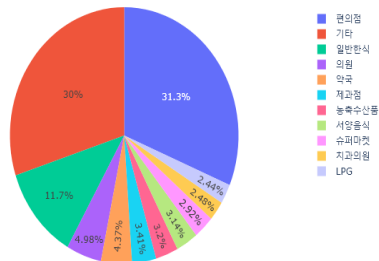
5월 6시 ~ 12시 제주도 일반 업종 별 파이



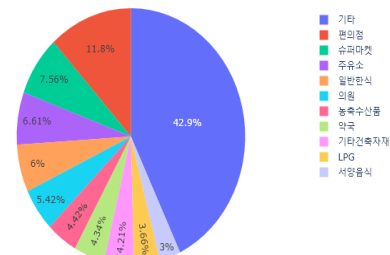
5월 6시 ~ 12시 제주도 중소 업종 별 파이



5월 6시 ~ 12시 제주도 중소1 업종 별 파이



5월 6시 ~ 12시 제주도 중소2 업종 별 파이

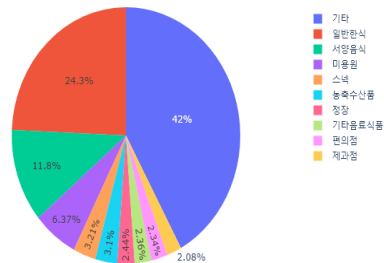


- 아침시간대: 06시 - 12시
- 새벽시간대에서 가장 높은 기록 건수를 나타낸 편의점의 기록이 확연하게 줄어듦.
- 기록 건수가 1% 미만인 업종들을 모두 포함하고 있는 기타의 기록이 모든 소상공인구분을 통틀어 가장 높았음.
- 기타 이외에 일반한식, 편의점, 슈퍼마켓, 서양음식 등이 상위권에 위치함.

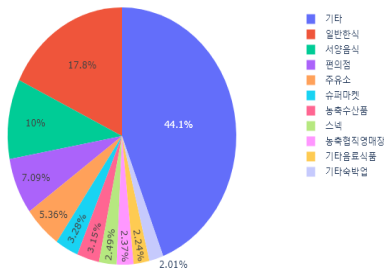
> 편의점, 음식점, 주점 등 주로 '식'에 해당하는 업종의 비율이 높았던 새벽시간대에 비해 제과점, 마켓, 식품점, 의원, 치과, 주유소 등 다양한 업종의 기록이 눈에 띄게 증가함.

3. 분석 결과 - [5월 점심-오후시간대의 소상공인구분 별 업종 비율 비교]

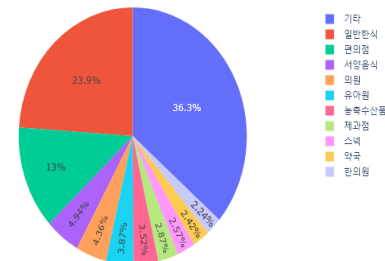
5월 12시 ~ 18시 제주도 영세 업종 별 파이



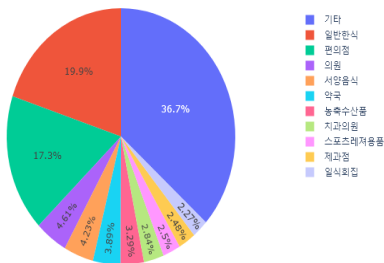
5월 12시 ~ 18시 제주도 일반 업종 별 파이



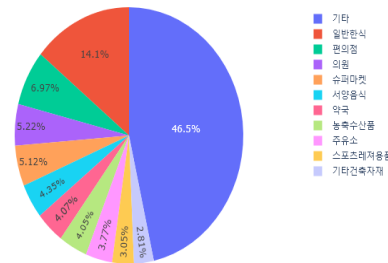
5월 12시 ~ 18시 제주도 중소 업종 별 파이



5월 12시 ~ 18시 제주도 중소1 업종 별 파이



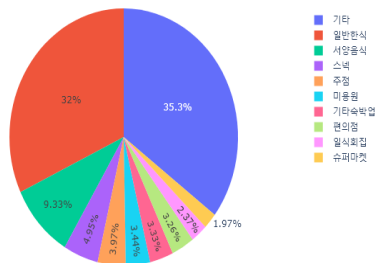
5월 12시 ~ 18시 제주도 중소2 업종 별 파이



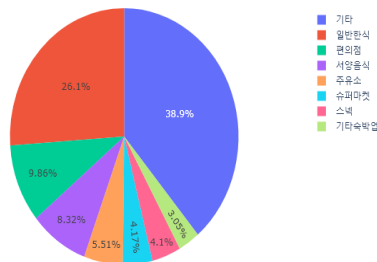
- 점심-오후시간대: 12시 - 18시
 - 기록 건수가 1% 미만인 업종들을 모두 포함하고 있는 기타의 기록이 모든 소상공인구분을 통틀어 여전히 높은 비율을 기록 - 다양한 소비가 이루어지고 있음을 증명함
- > 일반한식, 서양음식 등 음식점과 관련된 소비 비율이 증가

3. 분석 결과 - [5월 저녁-밤시간대의 소상공인구분 별 업종 비율 비교]

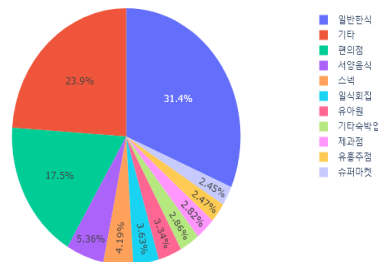
5월 18시 ~ 24시 제주도 영세 업종 별 파이



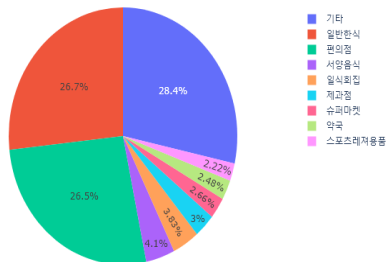
5월 18시 ~ 24시 제주도 일반 업종 별 파이



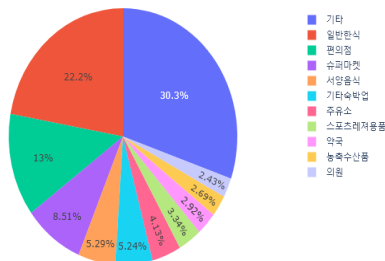
5월 18시 ~ 24시 제주도 중소 업종 별 파이



5월 18시 ~ 24시 제주도 중소1 업종 별 파이



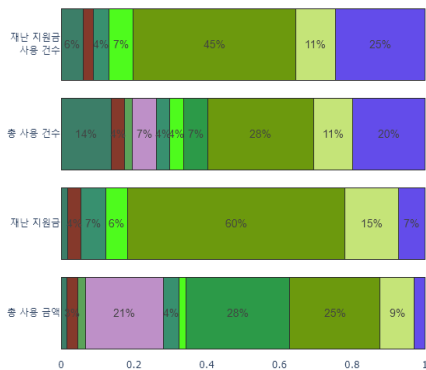
5월 18시 ~ 24시 제주도 중소2 업종 별 파이



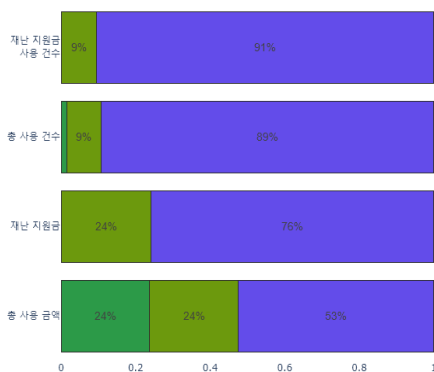
- 밤시간대: 18시 - 24시
- 중소1과 중소2의 기타 비율이 줄어들고 일반한식과 편의점 비율이 급증함
- 영세, 일반, 중소에선 일반한식이 기타 다음으로 가장 높은 비율을 차지함
- 다른 시간대에선 미미했던 숙박업과 관련된 소비 건수가 증가함

3. 분석 결과 - [5월 새벽시간대 소상공인구분 별 상위 10개 업종에 대한 재난지원금 사용실태]

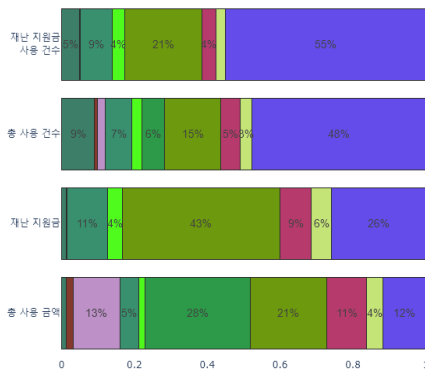
영세 업종 별 재난지원금 사용건수 및 금액: 5월 0시 ~ 6시



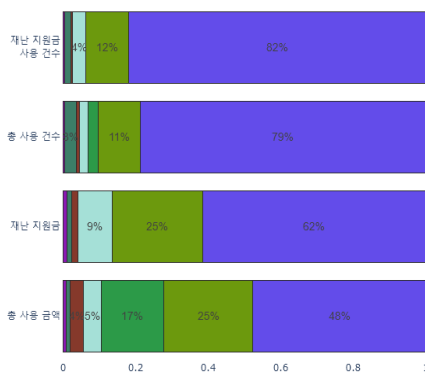
중소1 업종 별 재난지원금 사용건수 및 금액: 5월 0시 ~ 6시



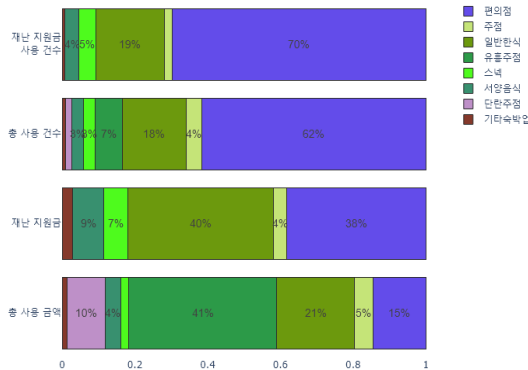
일반 업종 별 재난지원금 사용건수 및 금액: 5월 0시 ~ 6시



중소2 업종 별 재난지원금 사용건수 및 금액: 5월 0시 ~ 6시



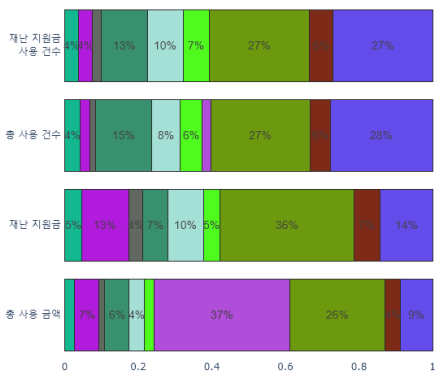
중소 업종 별 재난지원금 사용건수 및 금액: 5월 0시 ~ 6시



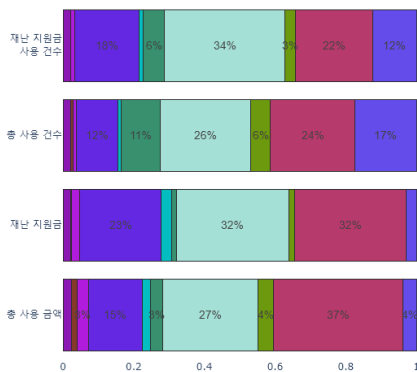
- 새벽시간대: 00시 - 06시
- 건수 비율이 가장 높았던 편의점이 일반, 중소, 중소1, 중소2에서 모든 항목에서 가장 높은 비율을 차지함. 특히 새벽시간대엔 소상공인의 매출 규모가 작아질수록 편의점의 재난지원금 이용건수 및 사용금액의 크기가 높아지는 것을 알 수 있음.
- 규모가 가장 작은 영세소상공인과 일반소상공인에선 일반업종의 사용금액 대비 재난지원금 사용금액이 높음.

3. 분석 결과 - [5월 아침시간대 소상공인구분 별 상위 10개 업종에 대한 재난지원금 사용실태]

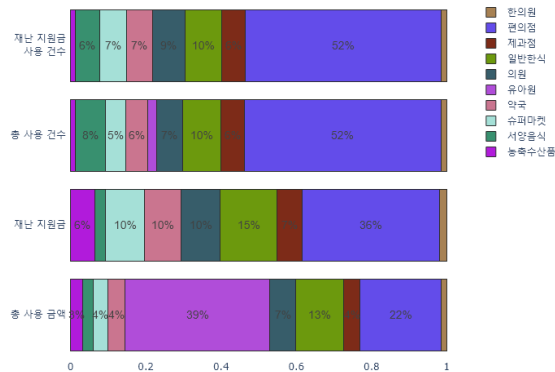
영세 업종 별 재난지원금 사용건수 및 금액: 5월 6시 ~ 12시



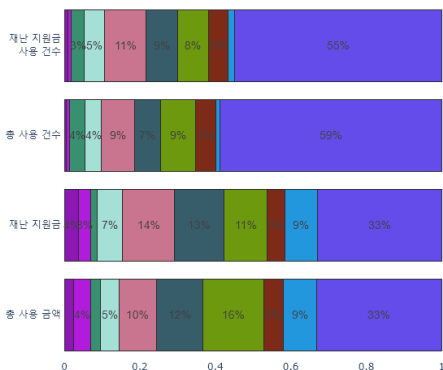
일반 업종 별 재난지원금 사용건수 및 금액: 5월 6시 ~ 12시



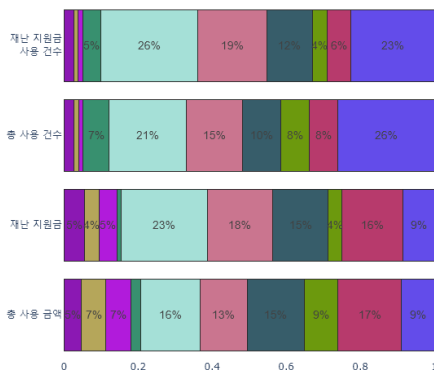
중소 업종 별 재난지원금 사용건수 및 금액: 5월 6시 ~ 12시



중소1 업종 별 재난지원금 사용건수 및 금액: 5월 6시 ~ 12시



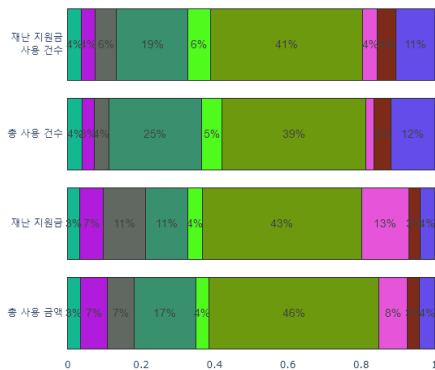
중소2 업종 별 재난지원금 사용건수 및 금액: 5월 6시 ~ 12시



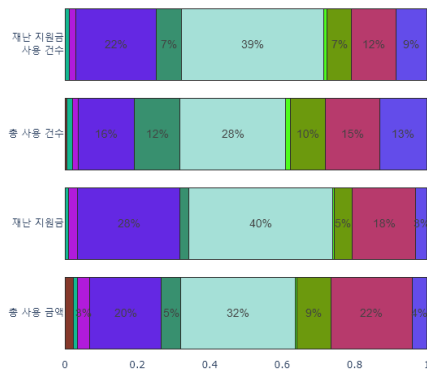
- 아침시간대: 06시 - 12시
- 모든 소상공인구분에 대해 편의점의 재난지원금 이용 건수 및 사용 금액이 여전히 매우 높은 것을 확인
- 새벽시간대에서 모든 소상공인구분에 대해 20-25%의 비율을 차지했던 유흥주점의 사용금액 및 이용건수가 줄어듦
- 영세소상공인에서는 일반한식업종에 대한 소비 및 재난지원금 소비가 활발히 이루어지고 있지만 소상공인의 규모가 커짐에 따라 점점 줄어듦
- 중소2 소상공인에서는 상위 업종들의 사용금액 비율이 거의 비슷하지만 슈퍼마켓에서 유독 재난지원금의 사용 비율이 높음

3. 분석 결과 - [5월 점심-오후시간대 소상공인구분 별 상위 10개 업종에 대한 재난지원금 사용실태]

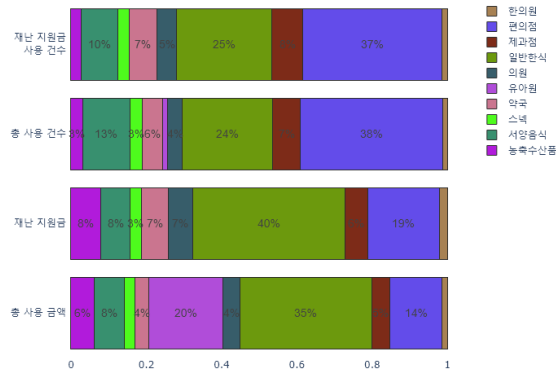
영세 업종 별 재난지원금 사용건수 및 금액: 5월 12시 ~ 18시



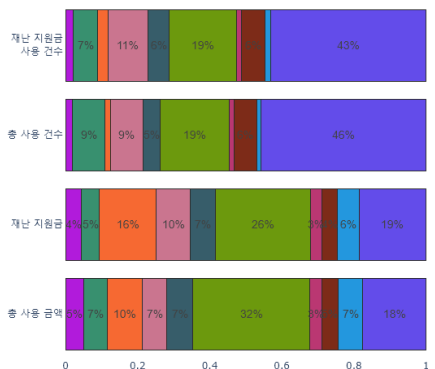
일반 업종 별 재난지원금 사용건수 및 금액: 5월 12시 ~ 18시



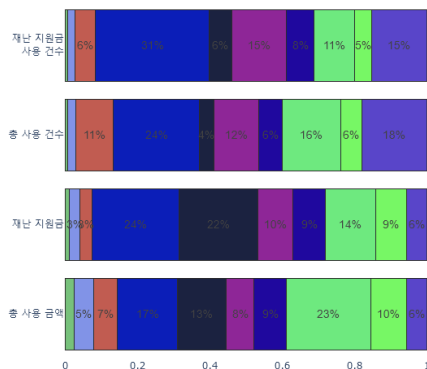
중소 업종 별 재난지원금 사용건수 및 금액: 5월 12시 ~ 18시



중소1 업종 별 재난지원금 사용건수 및 금액: 5월 12시 ~ 18시



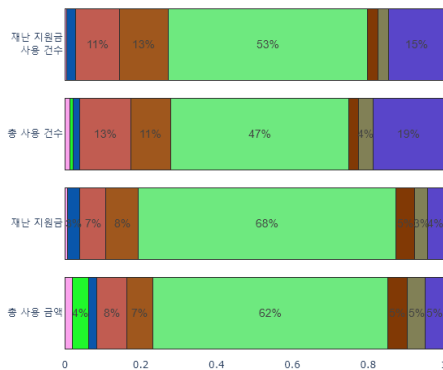
중소2 업종 별 재난지원금 사용건수 및 금액: 5월 12시 ~ 18시



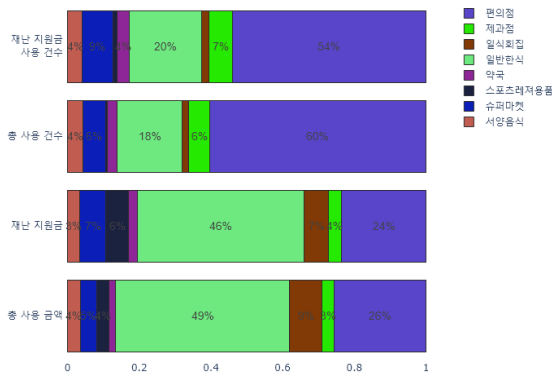
- 점심-오후시간대: 12시 - 18시
- 일반 소상공인에서는 슈퍼마켓의 재난지원금 이용건수 및 사용금액이 가장 높았고, 영세/중소1/중소2 소상공인 구분에서는 일반한식의 재난지원금 이용건수 및 사용금액 비율이 가장 높은 것을 확인함.
- 중소 소상공인에서 일반한식 업종이 재난지원금 사용 금액 및 총 사용금액 비율은 가장 높았지만, 이용건수 측면에서는 편의점이 가장 높음.

3. 분석 결과 - [5월 저녁-밤시간대의 소상공인구분 별 상위 10개 업종에 대한 재난지원금 사용실태]

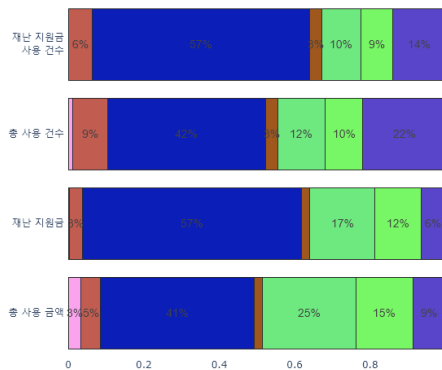
영세 업종 별 재난지원금 사용건수 및 금액: 5월 18시 ~ 24시



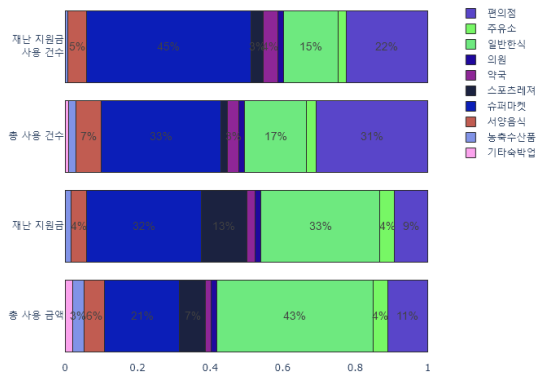
중소1 업종 별 재난지원금 사용건수 및 금액: 5월 18시 ~ 24시



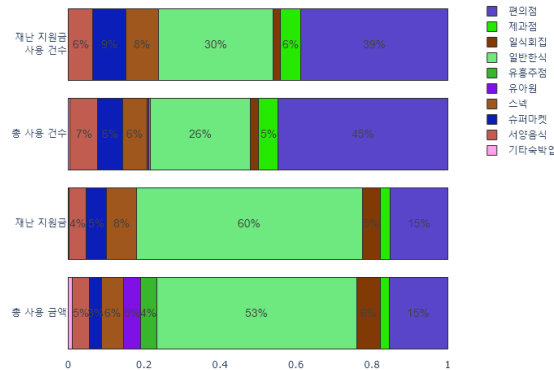
일반 업종 별 재난지원금 사용건수 및 금액: 5월 18시 ~ 24시



중소2 업종 별 재난지원금 사용건수 및 금액: 5월 18시 ~ 24시



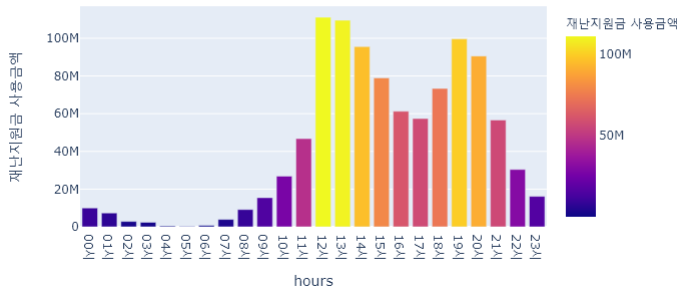
중소 업종 별 재난지원금 사용건수 및 금액: 5월 18시 ~ 24시



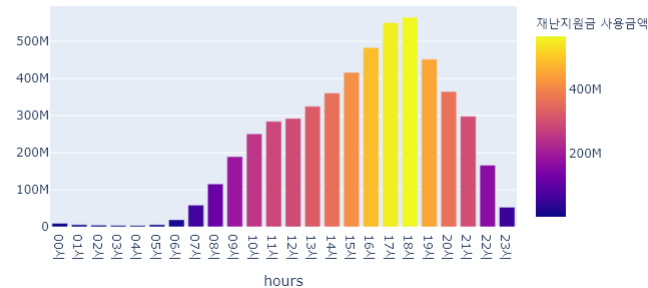
- 저녁-밤시간대: 18시 - 24시
- 다른 시간대에 비해 영세 소상공인을 제외한 모든 소상공인구분에서의 일반한식의 재난지원금 이용건수 및 사용금액이 큰 폭으로 증가해 매우 큰 비율을 차지함
- 영세 소상공인에서는 슈퍼마켓에서의 재난지원금 이용건수 및 사용금액이 크게 증가함

3. 분석 결과 - [재난지원금이 활발히 사용된 업종의 시간대 별 재난지원금 사용 실태]

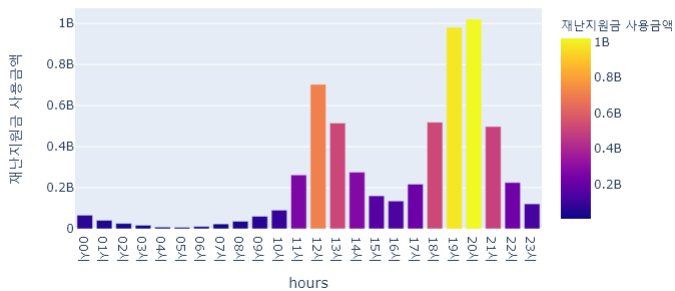
외국음식 한시간별 재난지원금 사용금액



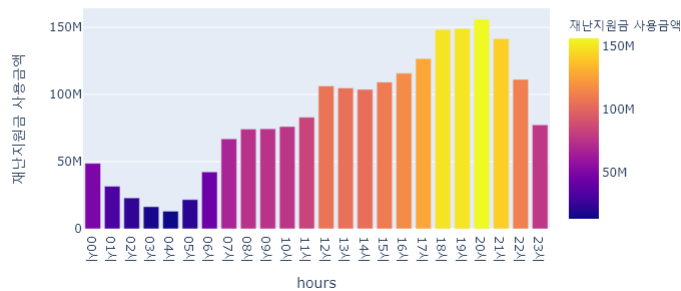
슈퍼마켓 한시간별 재난지원금 사용금액



일반한식 한시간별 재난지원금 사용금액



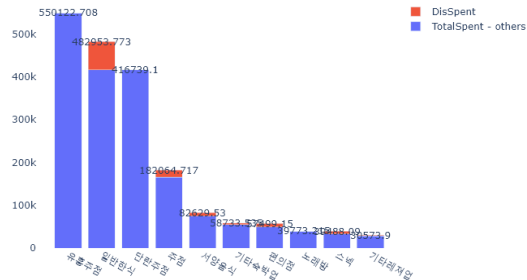
편의점 한시간별 재난지원금 사용금액



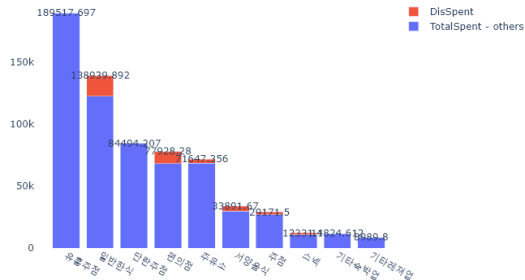
- 재난지원금이 활발히 사용된 서양음식, 슈퍼마켓, 일반한식, 편의점의 시간대 별 재난지원금 사용 양상을 시각화 하였음.
- 음식점의 경우 점심, 저녁 시간대에 재난지원금 사용금액이 높았으며 슈퍼마켓과 편의점은 오전 11시부터 사용금액이 점차 증가하여 오후 8시의 늦은 저녁시간대부터 서서히 감소하기 시작함.
- 특히 편의점의 경우에는 아주 이른 새벽시간대를 제외한 거의 모든 시간에 재난지원금 사용금액이 꾸준하였음.

3. 분석 결과 - [5월 새벽시간대 소상공인구분 별 상위 10개 업종에 대한 재난지원금 사용실태]

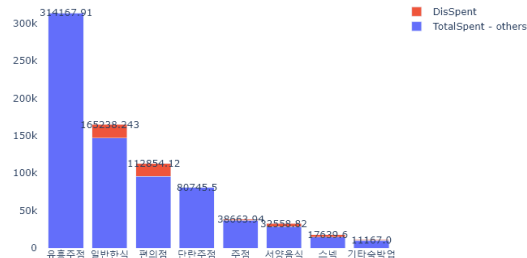
영세 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 0시 ~ 6시 [단위: 1000원]



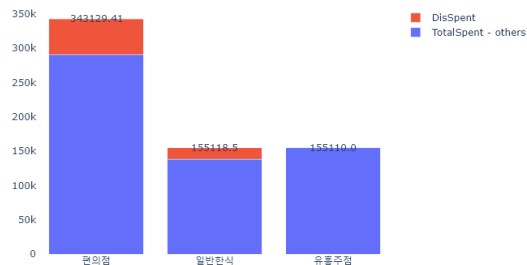
일반 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 0시 ~ 6시 [단위: 1000원]



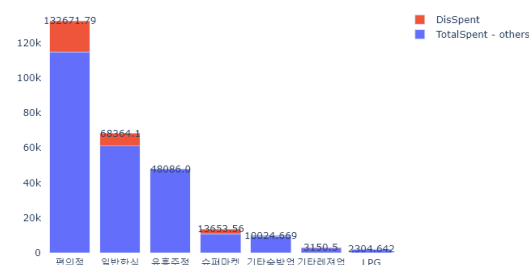
중소 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 0시 ~ 6시 [단위: 1000원]



중소1 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 0시 ~ 6시 [단위: 1000원]



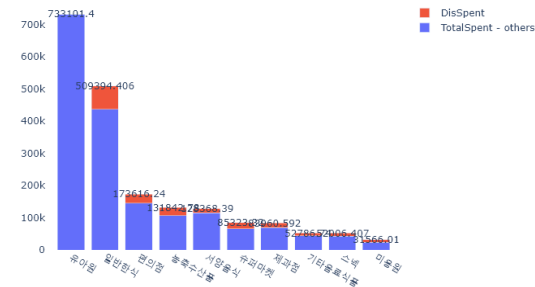
중소2 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 0시 ~ 6시 [단위: 1000원]



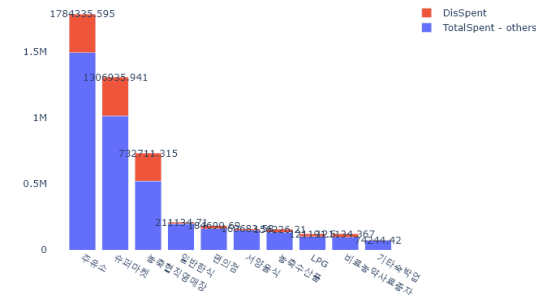
- 새벽시간대: 00시 - 06시
- 중소1과 중소2를 제외한 모든 소상공인구분에서 유흥주점에서의 사용금액이 가장 높았지만, 유흥주점에서 재난지원금의 사용금액은 미미한 수준임. 두 번째로 높은 사용금액을 기록한 일반한식에서 재난지원금의 사용금액은 상당히 높음.
- 중소1, 중소2에서 편의점이 가장 높은 사용금액을 기록했으며 사용금액 대비 재난지원금 사용금액도 매우 높음.

3. 분석 결과 - [5월 아침시간대 소상공인구분 별 상위 10개 업종에 대한 재난지원금 사용실태]

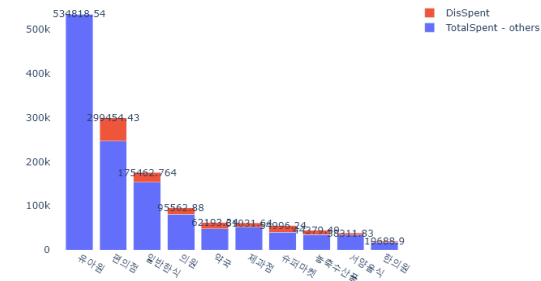
영세 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 6시 ~ 12시 [단위: 1000원]



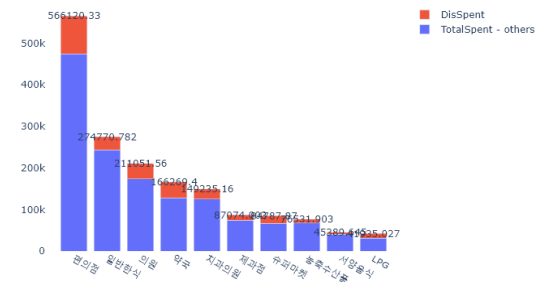
일반 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 6시 ~ 12시 [단위: 1000원]



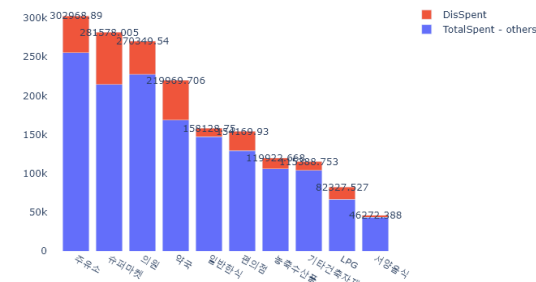
중소 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 6시 ~ 12시 [단위: 1000원]



중소1 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 6시 ~ 12시 [단위: 1000원]



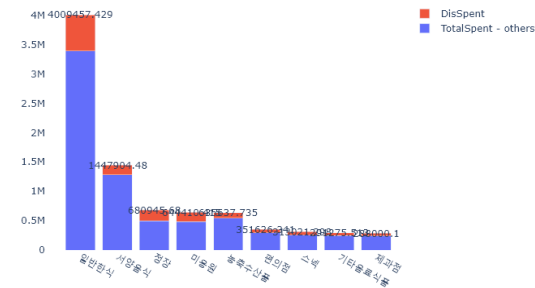
중소2 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 6시 ~ 12시 [단위: 1000원]



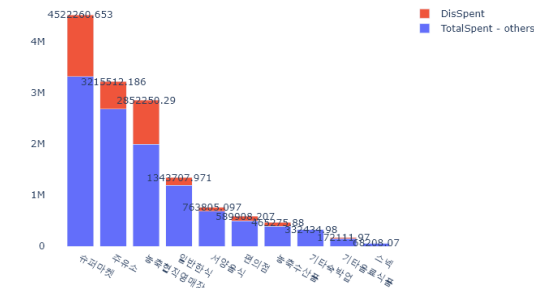
- 아침시간대: 06시 - 12시
- 영세, 중소 소상공인에서는 유아원에서의 사용금액이 가장 높았지만 사용금액 대비 재난지원금 사용금액은 매우 낮음.
- 일반, 중소2 소상공인에서 가장 높은 사용금액을 기록한 주유소에서는 재난지원금의 사용금액 비율이 높은 편.

3. 분석 결과 - [5월 점심-오후시간대 소상공인구분 별 상위 10개 업종에 대한 재난지원금 사용실태]

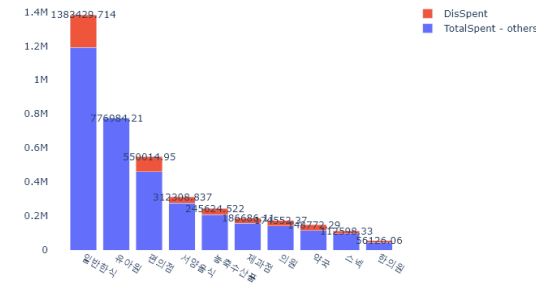
영세 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 12시 ~ 18시 [단위: 1000원]



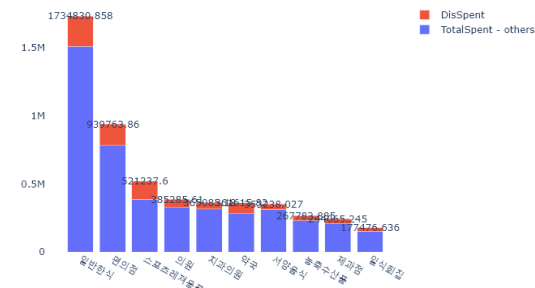
일반 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 12시 ~ 18시 [단위: 1000원]



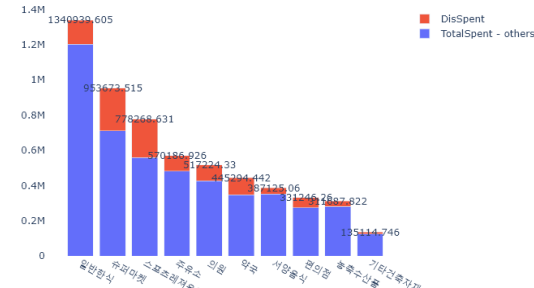
중소 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 12시 ~ 18시 [단위: 1000원]



중소1 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 12시 ~ 18시 [단위: 1000원]



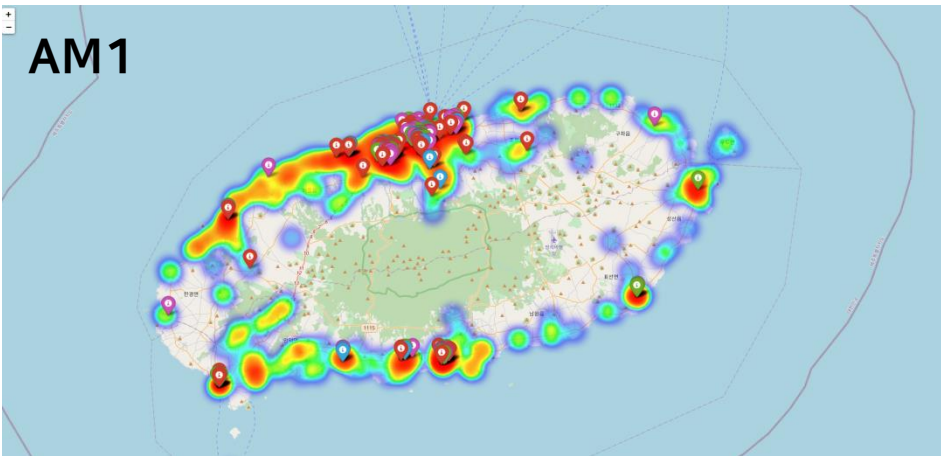
중소2 업종 별 총 사용금액 대비 재난지원금 사용금액: 5월 12시 ~ 18시 [단위: 1000원]



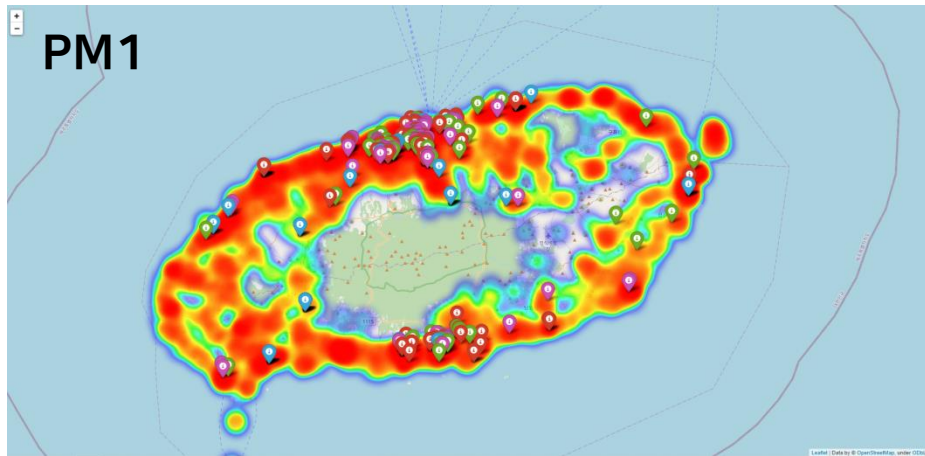
- 점심-오후시간대: 12시 - 18시
- 일반 소상공인을 제외한 모든 소상공인 구분에서 일반 한식 업종의 사용금액이 가장 높았고, 사용금액 대비 재난지원금 사용금액도 높음.

3. 분석 결과 - [6월 영세 소상공인구분의 상위 10개 업종에 대한 지리적 위치 분포 분석]

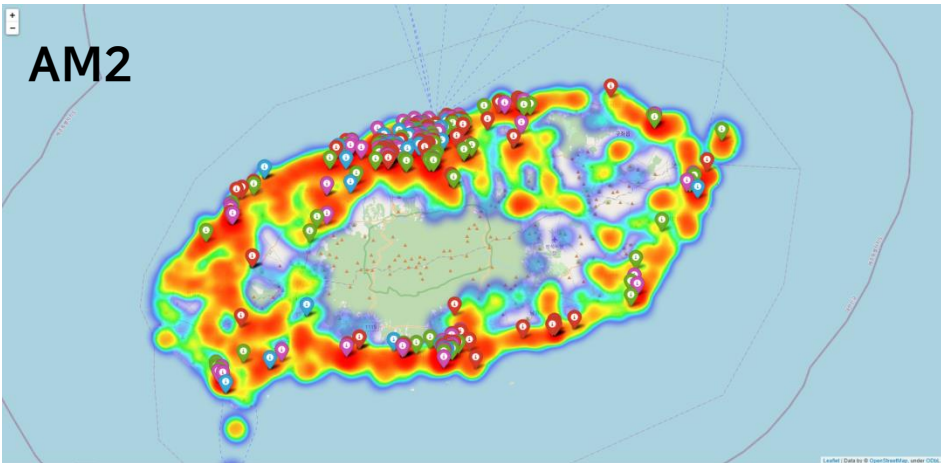
AM1



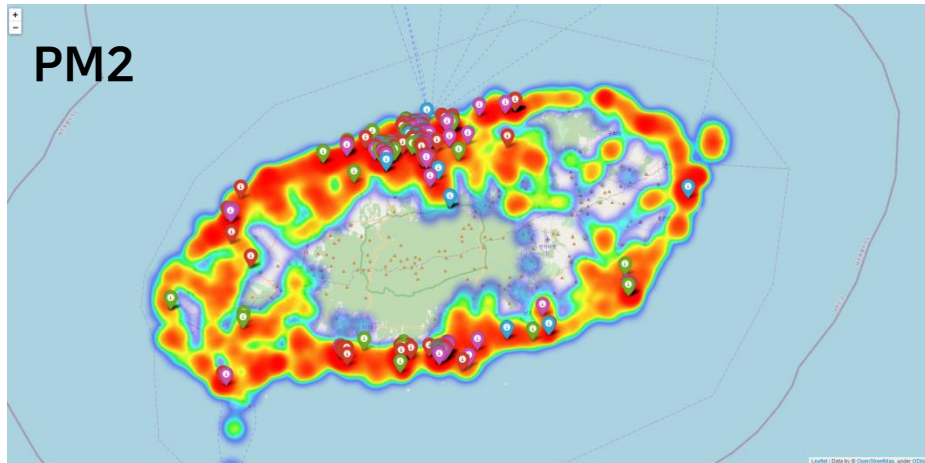
PM1



AM2

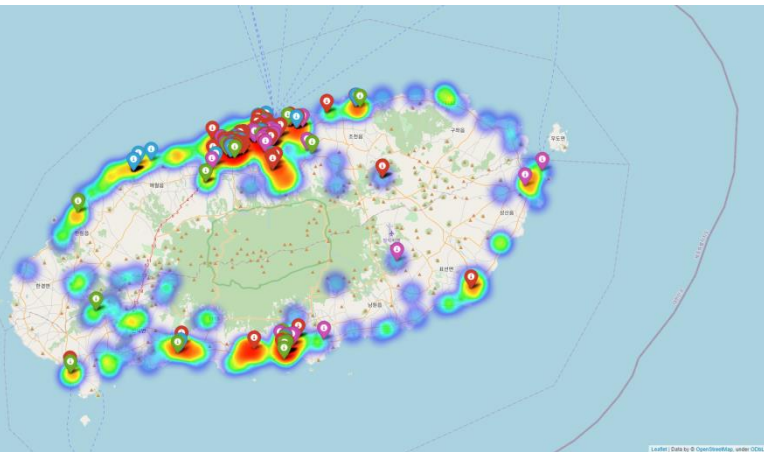


PM2

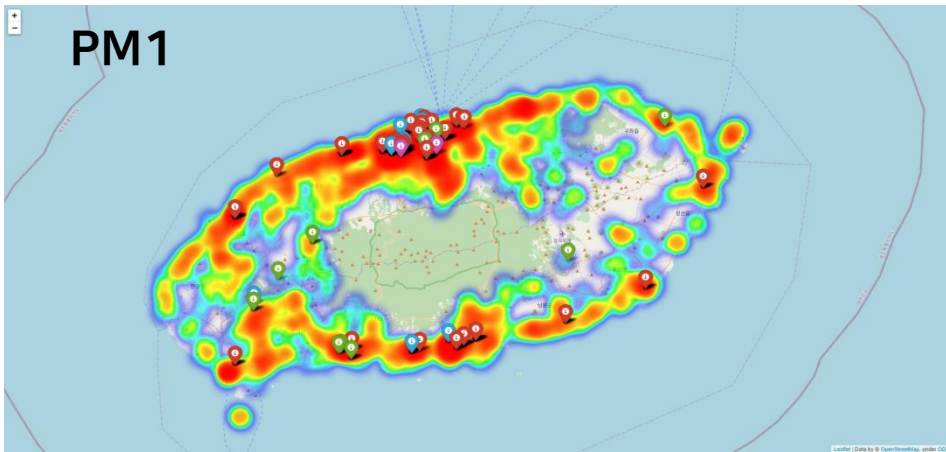


3. 분석 결과 - [6월 일반 소상공인구분의 상위 10개 업종에 대한 지리적 위치 분포 분석]

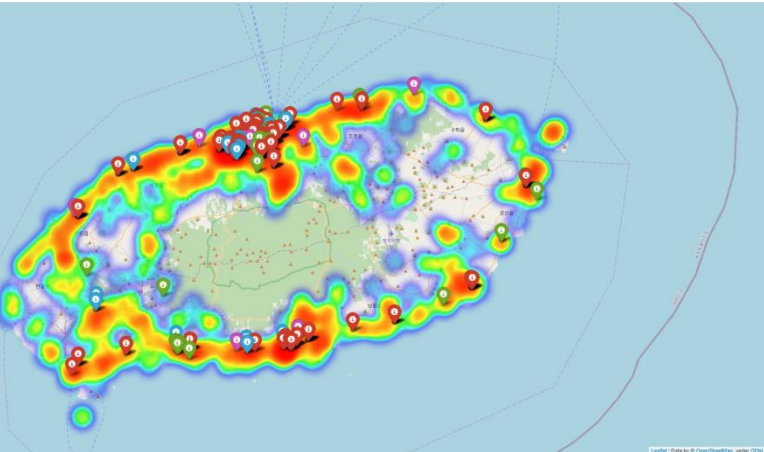
AM1



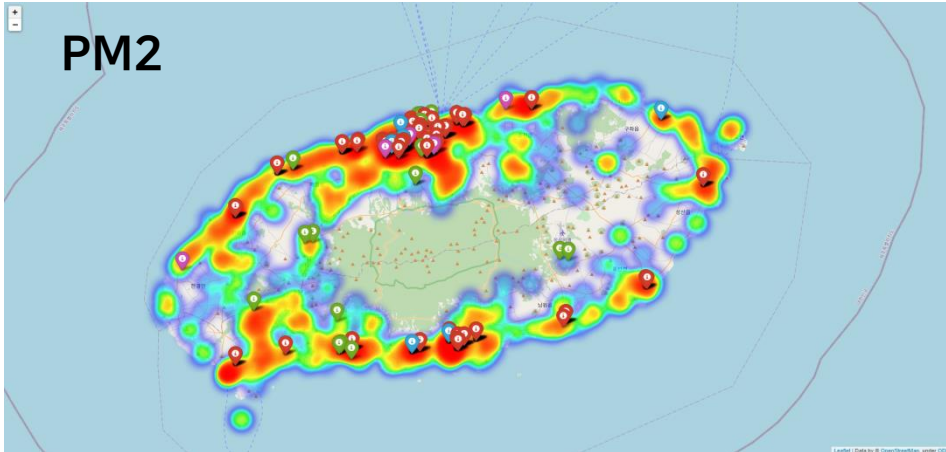
PM1



AM2

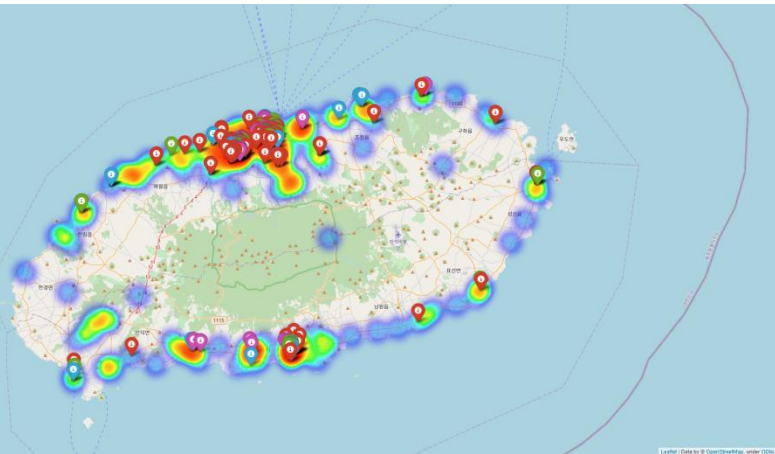


PM2

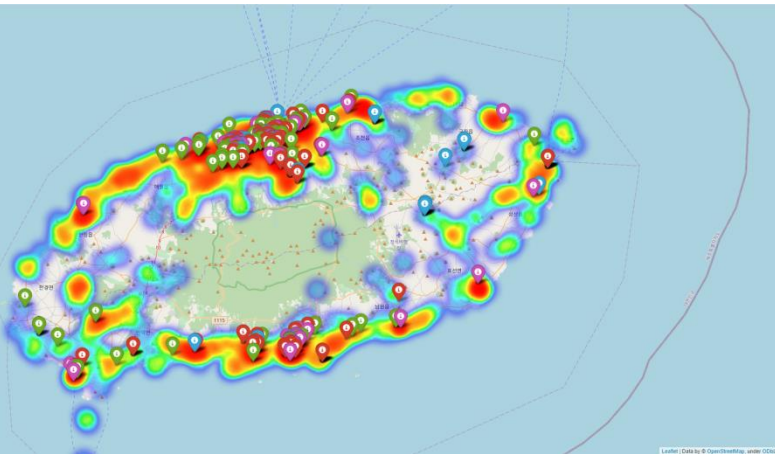


3. 분석 결과 - [6월 중소 소상공인구분의 상위 10개 업종에 대한 지리적 위치 분포 분석]

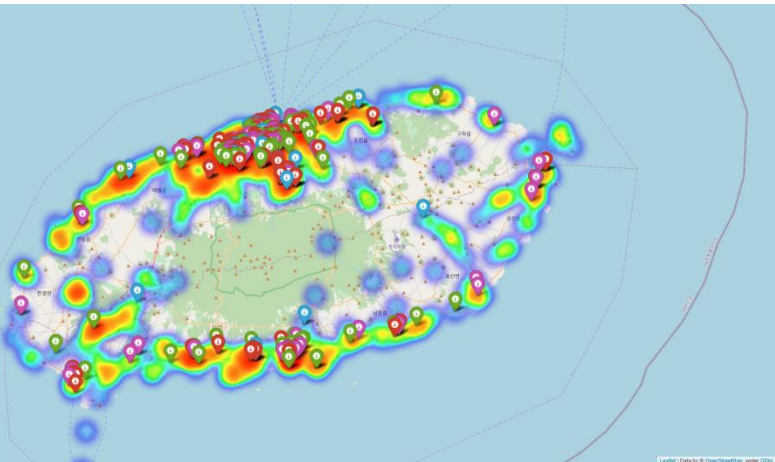
AM1



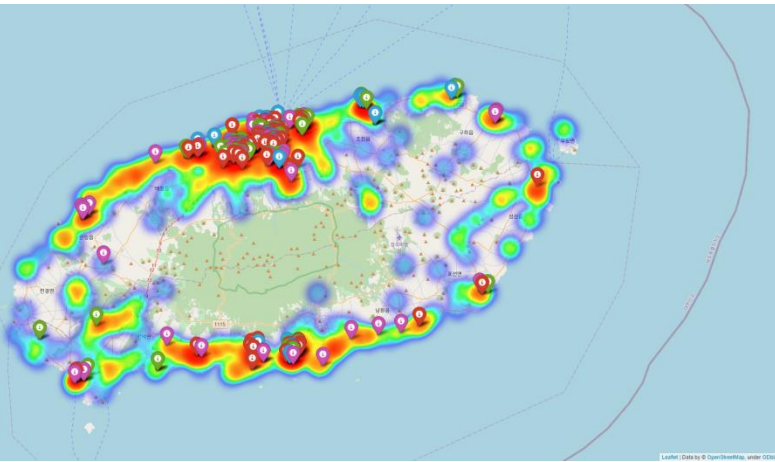
PM1



AM2

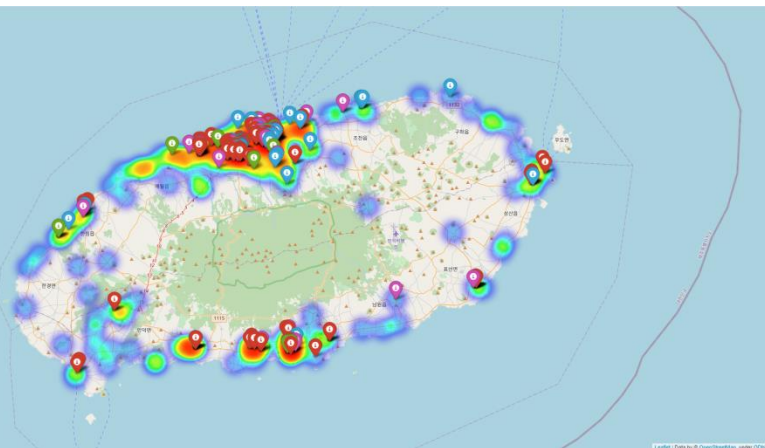


PM2

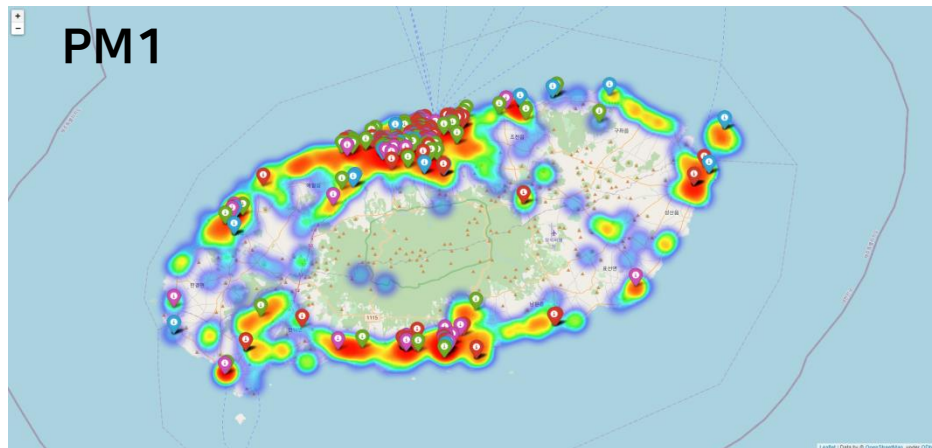


3. 분석 결과 - [6월 중소1 소상공인구분의 상위 10개 업종에 대한 지리적 위치 분포 분석]

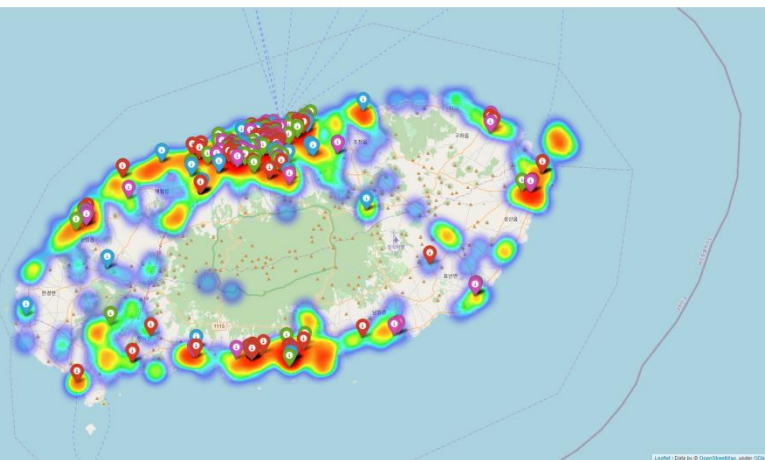
AM1



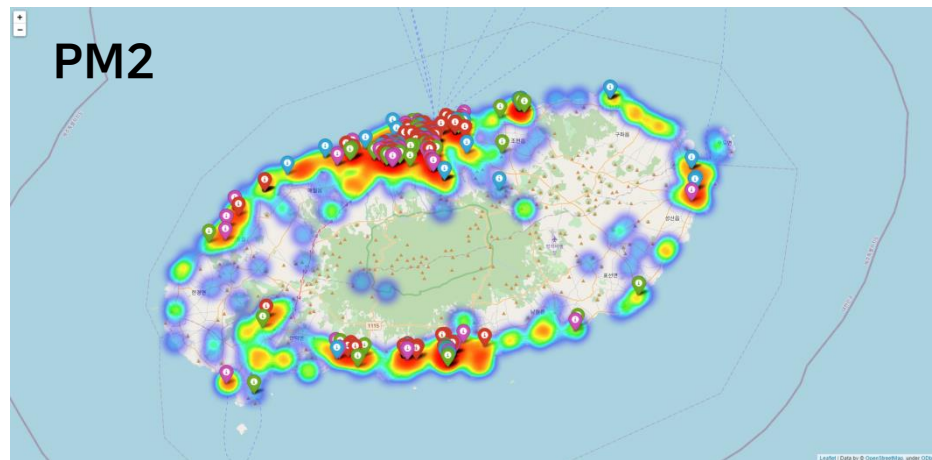
PM1



AM2

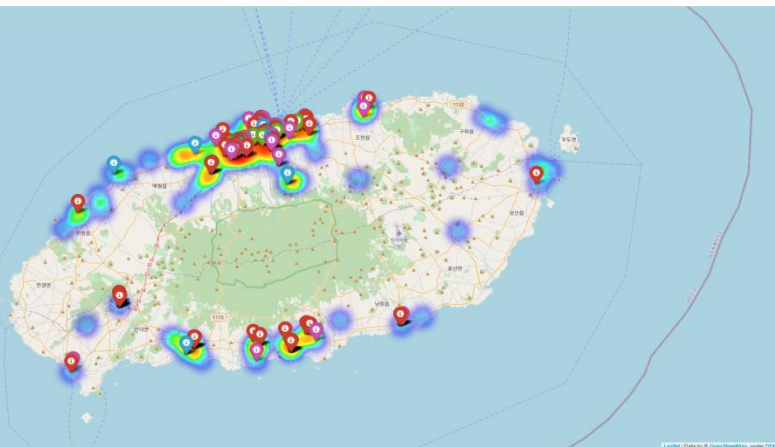


PM2

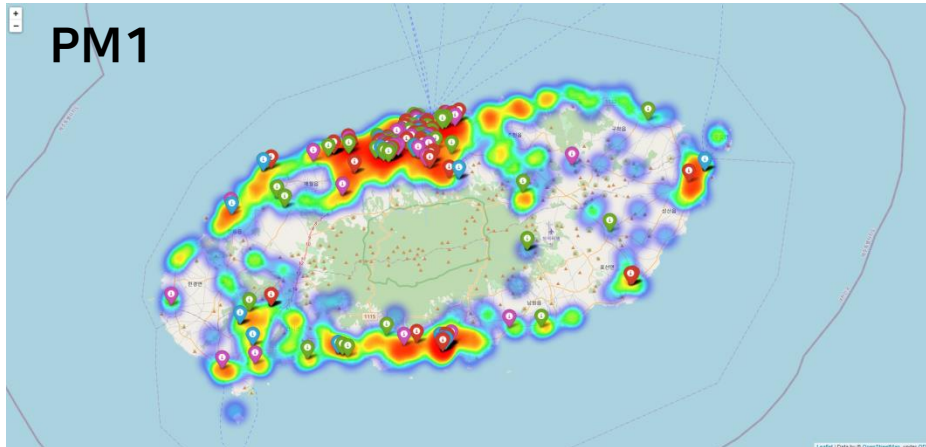


3. 분석 결과 - [6월 중소2 소상공인구분의 상위 10개 업종에 대한 지리적 위치 분포 분석]

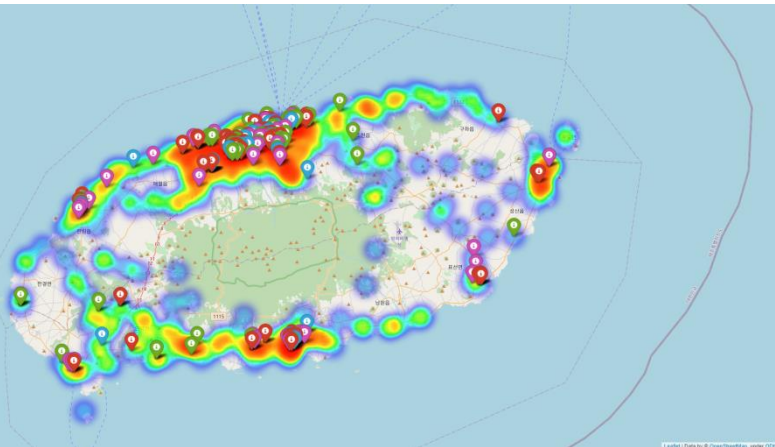
AM1



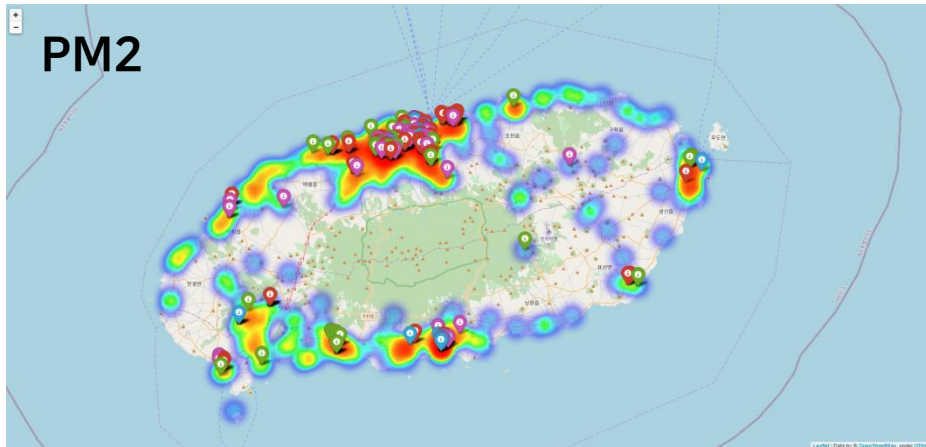
PM1



AM2

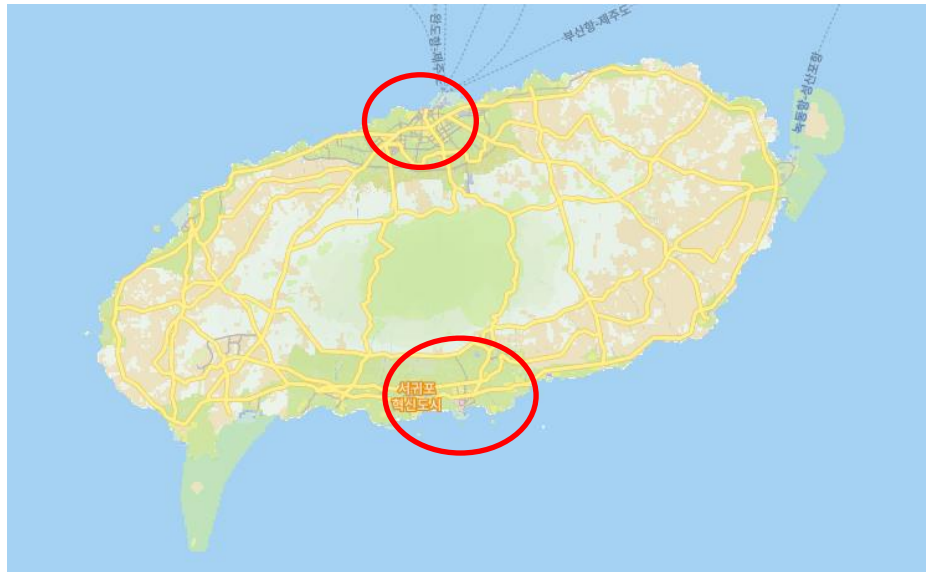


PM2



3. 분석 결과 - [6월 소상공인구분 별 지리적 위치 분포 시각화]

- 소상공인 구분 별로 시간대에 따른 소비 활성화 지역을 히트 맵으로 시각화 한 결과, 영세 소상공인이 시간대에 따라 소비 활성화 지역에 가장 큰 차이를 보였다. 영세, 일반, 중소, 중소1, 중소2 모두 새벽 시간대에 소비가 지역 전반적으로 위축되어 있었지만, 영세 소상공인은 아침, 점심, 저녁 시간대에 소비 활성화 지역이 크게 넓어짐을 알 수 있었다.
- 중소, 중소1, 중소2 소상공인은 새벽 시간대를 제외한 아침, 점심, 저녁 시간대에 소비 활성화 지역이 비슷한 양상을 나타내었다. 반면에 영세와 일반 소상공인은 점심 시간대에 소비 활성화 지역이 가장 광범위 했고 아침과 저녁은 비슷한 양상을 띄었다.
- 모든 소상공인을 통틀어 가장 활발한 소비 지역은 제주도 제주시, 서귀포시 모두 제 2종 일반주거지역 인근의 일반 상업 지역인 것으로 나타났다.



3. 분석 결과 - [5월 동, 리별 지역기반 재난지원금 사용실태 지도 시각화]

[총 이용건수]



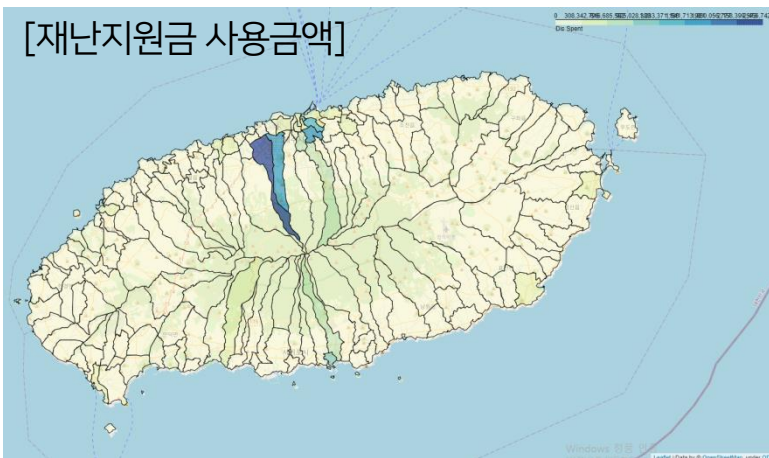
[재난지원금 이용건수]



[총 사용금액]

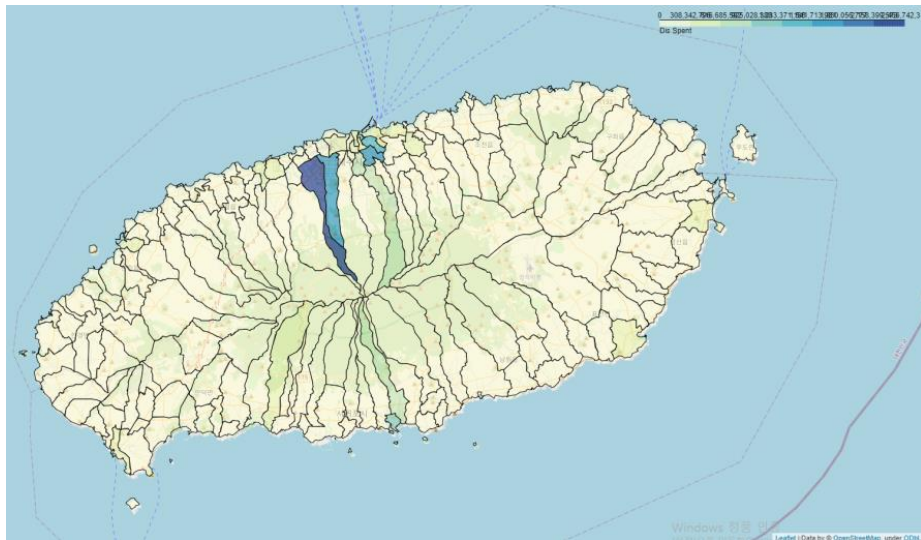


[재난지원금 사용금액]



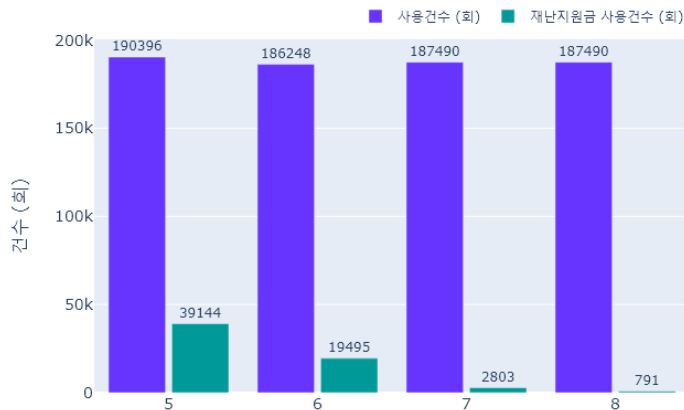
3. 분석 결과 - [5월 동, 리별 지역기반 재난지원금 사용실태 지도 시각화]

- 지도를 보면 모든 데이터 카테고리에서 제주시가 압도적인 비율을 차지한다. 특히 노형동과 연동에서 상당히 큰 재난지원금 사용량을 보인다.
- 노형동은 주변 다른 법정동과 달리 압도적으로 주거지역의 성향이 짙은 곳이다. 주거지역임에 따라 상업지역보다 편의점과 슈퍼마켓, 각종 의원의 비율이 높아 재난지원금 사용이 빈번하게 이루어진 것으로 보인다.
- 연동은 이와 다르게 상업지구가 발달한 곳이다. 신시가지로 격자형의 도로망에 각종 관공서가 위치해 있으며 특히 누웨 마루거리에서 큰 소비가 일어난다. 다만 상업지구인 만큼 총 소비액에 비한 재난지원금 사용액은 저조했다.
- 그외 이도동과 일도동은 오래된 주거지역과 업무지구로 노형동과 연동에 이어 큰 지출을 보인다.
- 제주시와 달리 서귀포 시에는 재난 지원금 사용이 크지 않았다.



3. 분석 결과 - [일도이동의 월별 재난지원금 이용건수 및 사용금액 비교]

제주특별자치도 제주시 일도이동의 월별 결제건수 대비 재난지원금 이용건수 비교



제주특별자치도 제주시 일도이동의 월별 결제금액 대비 재난지원금 결제금액 비교



- 제주도의 모든 법정동별 재난지원금 이용건수 및 사용금액을 분석/시각화하였는데 개수가 너무 많아서 일도이동을 예시로 공유합니다. 모든 법정동 별에 대한 결과는 깃헙에 결과를 업로드 해 두었으니 참조 바랍니다.
- 제주도 일도이동의 경우 각 월별로 사용건수는 모두 비슷하였으나 재난지원금의 사용건수는 5월이 가장 높았고 8월까지 점진적으로 감소하였다.
- 결제금액 및 재난지원금의 결제금액 또한 5월부터 점차 감소하는 것을 보였는데, 이는 일도이동의 경우 이용건수에 무관하게 결제금액과 재난지원금 결제금액이 위축되는 것을 알 수 있다.

04

결론

재난지원금은 언제 가장 많이 사용됐나요?

- ✓ 월별: 재난지원금은 5월에 사용금액이 가장 높았고, 월마다 점차 줄어들어 8월에 가장 낮은 사용율을 보였습니다. 특히 5월에는 전체 소비금액 대비 약 14퍼센트가 재난지원금 사용금액이었습니다.
- ✓ 시간대별: 점심시간대, 저녁시간대에 재난지원금 사용율이 높았습니다.
- ✓ 시간별: 저녁 6시, 7시에 재난지원금 사용금액이 높았습니다.

재난지원금은 어디에 가장 많이 사용됐나요?

- ✓ 소상공인구분별: 일반(대형) 소상공인에서 재난지원금이 가장 많이 사용되었고, 영세 소상공인이 두 번째를 이었습니다. 코로나19로 인해 큰 피해를 입은 자영업자들에게 도움을 주기 위해 지급된 재난지원금인만큼 일반(대형) 소상공인이 아닌, 규모가 작은 소상공인 업종들에게 확실한 경제적 도움을 줄 수 있도록 해야합니다.
- ✓ 업종별: 재난지원금은 일반한식, 편의점, 슈퍼마켓, 농협마트, 주유소등 식생활과 생활 편의에 관련된 업종에서 주로 사용되었습니다.
- ✓ 업종대분류별: 5월 ~ 8월을 통틀어 총 사용금액 대비 재난지원금 사용금액이 가장 높은 업종 대분류는 '음식' 이었습니다.

재난지원금은 제주도의 어느 지역에서 많이 사용되었나요?

- ✓ 재난지원금이 활발히 사용된 지역은 제주도 제주시, 서귀포시 모두 제 2종 일반주거지역 인근의 일반 상업 지역인 것으로 나타났습니다.
- ✓ 제주도의 두 시(제주시, 서귀포시)를 비교해 보았을 때 서귀포시보다 제주시에서 재난지원금의 사용율이 눈에 띄게 높았습니다. 특히 노형동과 연동에서 매우 큰 재난지원금 사용량을 보였습니다.
- ✓ 지구별 재난지원금 사용량이 높은 동:
 - ✓ 주거지역: 노형동
 - ✓ 상업지역: 연동
 - ✓ 주거지역 및 업무지구: 이도동, 일도동



감사합니다!