Ch13. 파리바게뜨와 감탄떡볶이의 거리 구하기

1. 프로젝트 소개 및 공공데이터 가져오기

이디야의 입점 전략은 스타벅스 옆일까?

논문) 위치 정보를 활용한 커피 전문점의 입점 분석 : https://koreascience.kr/article/JAKO201625058596417.pdf

감탄 떡볶이의 입점전략은 파리바게트옆인가?

감탄떡볶이 (구 아딸떡볶이)의 이경수 대표의 인터뷰

◆"파리바게뜨 옆에 떡볶이집을 차려라"

—점포 위치 선정하는 원칙이 독특하던데요.

"저는 창업주들께 '아파트 입구, 횡단보도 앞, 파리바게뜨 옆'을 추천합니다."

—'파리바게뜨 옆'은 뭡니까.

"파리바게뜨는 전액 포장매출입니다. '아딸'의 목표와 같습니다. 그리고 사는 사람은 어른이지만 누구나 좋아하는 음식이란 공통점도 있습니다."

—파리바게뜨는 되고 뚜레쥬르는 안됩니까.

"(웃으며) 됩니다. 그런데 파리바게뜨 숫자가 훨씬 많습니다. 장사 잘되는 저희 점포를 보면 대개 파리바게뜨 옆입니다. 횡단보도는 동네와 동네를 잇는 다리입니다. 그곳에 사람들이 모이죠. 신호를 기다리면서 건너편 가게들을 보게 되는 이점도 있습니다."

인터뷰 링크 : https://www.chosun.com/site/data/html_dir/2011/09/02/2011090201323.html

공공데이터

소상공인시장진흥공단_상가(상권)정보 : https://www.data.go.kr/data/15083033/fileData.do

링크에서 csv파일을 다운받고 서울의 데이터를 이용해서 실습을 해도 된다

```
import pandas as pd
url1 = '/content/소상공인시장진흥공단_상가(상권)정보_서울_202212.csv'
pd.read_csv(url1, engine='python', encoding='utf-8')
```

실습에 쓰일 데이터

다운받아 실습하는 것이 불편할 수강생을 위해 다운받은 공공데이터의 일부 열만 추린 csv파일을 올려두었다

https://raw.githubusercontent.com/panda-kim/csv_files/main/food_business_seoul_20221231.csv

과연 감탄떡볶이의 입점전략이 파리바게트 옆인지 알아보기위해 감탄떡볶이외 국대떡볶이, 죠스떡볶이, 엽기떡볶이, 신전떡볶이와 파리바게트의 거리를 구해서 비교해보자

```
# 프로젝트 코드
import pandas as pd
pd.options.display.max_rows = 6 # 판다스 1.4+ 에서 6행까지만 출력
pd.options.display.float_format = '{:.4f}'.format # 소수점 네자리 까지만 출력
url = 'https://raw.githubusercontent.com/panda-kim/csv_files/main/food_business_seoul_20221231.csv'
tt = ['감탄떡볶이', '국대떡볶이', '죠스떡볶이', '엽기떡볶이', '신전떡볶이']

df = pd.read_csv(url)
df
```

	상가업소번호	상호명	지점명	상권업종중분류명	상권업종소분류명	시군구명	행정동명	도로명주소	경도	위도
0	17174175	비지트	NaN	한식	한식/백반/한정식	서초구	방배4동	서울특별시 서초구 동광로18길 82	126.9914	37.4884
1	17174119	쓰리에프	NaN	한식	한식/백반/한정식	동작구	사당2동	서울특별시 동작구 동작대로27 가길 12	126.9810	37.4871
2	17174040	다향	NaN	한식	한식/백반/한정식	서초구	서초3동	서울특별시 서초구 효령로 230	127.0094	37.4834
...	...	...	...	...	...	...	...	...	...	...
90829	17264432	북창동순두부	서초반포점	한식	한식/백반/한정식	서초구	반포1동	서울특별시 서초구 주흥길 10	127.0208	37.5043

2. 떡볶이 프랜차이즈와 파리바게뜨의 데이터 추출하기

```
# 프로젝트 코드
import pandas as pd
pd.options.display.max_rows = 6 # 판다스 1.4+ 에서 6행까지만 출력
pd.options.display.float_format = '{:.4f}'.format # 소수점 네자리 까지만 출력
url = 'https://raw.githubusercontent.com/panda-kim/csv_files/main/food_business_seoul_20221231.csv'
tt = ['감탄떡볶이', '국대떡볶이', '죠스떡볶이', '엽기떡볶이', '신전떡볶이']
df = pd.read_csv(url)
```

이번 강의에 쓰이는 `str.contains` 함수와 `str.extract` 함수가 생소하다면

다음 링크에서 **6. 문자열 다루기**부분의 수업자료를 복습하거나 **05-06 문자열 다루기 강의**를 다시 복습하는 것이 좋다

Ch05 수업자료 링크

먼저 파리바게뜨의 데이터 추출하기

str.contains 는 특정 문자열 포함여부를 True, False로 반환하는 함수

```
# 파리바게뜨 데이터 필터링하고 정제하기
cond1 = df['상호명'].str.contains(r'파리바게뜨|파리바게트')
df_paris = df.loc[cond1, ['상가업소번호', '경도', '위도']].copy()
df_paris = (df_paris
            .set_axis(['p_업소번호', 'p_경도', 'p_위도'], axis=1)
            .reset_index(drop=True))
df_paris
```

	p_업소번호	p_경도	p_위도
0	26669769	126.8839	37.4903
1	9171904	127.0330	37.6078
2	23396020	126.9886	37.5742
...	...	...	...
356	20589842	126.9019	37.4522
357	20595157	127.1310	37.5101
358	17182192	127.1644	37.5569

359 rows × 3 columns

떡볶이 프랜차이즈의 데이터 추출하기

이번에는 떡볶이 프랜차이즈의 상호를 추출하기 위해 str.contains 대신 str.extract 함수를 쓴다

str.extract 함수는 특정 문자열을 포함하고 있다면 그 문자열을 반환하고 아니면 NaN을 반환하는 함수

```
# 떡볶이 프랜차이즈 데이터 필터링하고 정제하기
df_tt = df.copy()
df_tt['상호명'] = df_tt['상호명'].str.extract(r'({})'.format('|'.join(tt)))[0]
df_tt = (df_tt
         .dropna(subset=['상호명'])
         .iloc[:, [0, 1, 5, 8, 9]]
         .reset_index(drop=True))
df_tt
```

	상가업소번호	상호명	시군구명	경도	위도
0	20844443	신전떡볶이	중랑구	127.1014	37.6003
1	25541815	신전떡볶이	중랑구	127.0794	37.6119
2	20738752	엽기떡볶이	양천구	126.8535	37.5210
...	...	...	...	...	...
257	23444901	죠스떡볶이	영등포구	126.9058	37.5140
258	16990013	신전떡볶이	용산구	126.9992	37.5278
259	20081411	국대떡볶이	서초구	126.9966	37.4814

260 rows × 5 columns

3. 곱집합(cartesian product) & 위도와 경도로 거리구하기(haversine)

```
# 프로젝트 코드
import pandas as pd
pd.options.display.max_rows = 6 # 판다스 1.4+ 에서 6행까지만 출력
```

```
pd.options.display.float_format = '{:.4f}'.format # 소수점 네자리 까지만 출력
url = 'https://raw.githubusercontent.com/panda-kim/csv_files/main/food_business_seoul_20221231.csv'
tt = ['감탄떡볶이', '국대떡볶이', '죠스떡볶이', '엽기떡볶이', '신전떡볶이']
df = pd.read_csv(url)

# 파리바게뜨 데이터 필터링하고 정제하기
cond1 = df['상호명'].str.contains(r'파리바게뜨|파리바게트')
df_paris = df.loc[cond1, ['상가업소번호', '경도', '위도']].copy()
df_paris = (df_paris
            .set_axis(['p_업소번호', 'p_경도', 'p_위도'], axis=1)
            .reset_index(drop=True))

# 떡볶이 프랜차이즈 데이터 필터링하고 정제하기
df_tt = df.copy()
df_tt['상호명'] = df_tt['상호명'].str.extract(r'({})'.format('|'.join(tt)))[0]
df_tt = (df_tt
         .dropna(subset=['상호명'])
         .iloc[:, [0, 1, 5, 8, 9]]
         .reset_index(drop=True))
```

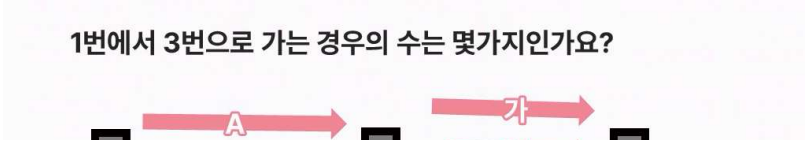
특정 떡볶이 가게와 파리바게트와의 거리는 어떻게 구해야 할까?

특정 가게와 모든 파리바게뜨와의 거리를 구한 뒤 최소값이 그 가게에서 파리바게뜨와의 거리이다

곱집합이란?

양측의 모든 순서쌍을 만드는 조합

과거 경우의 수에서도 쓰였다




```
# 곱집합(cartesian product)을 만드는 cross join
df1 = pd.DataFrame(['A', 'B'])
df2 = pd.DataFrame(['가', '나', '다'])
df1.merge(df2, how='cross')
```

	0_x	0_y
0	A	가
1	A	나
2	A	다
3	B	가
4	B	나
5	B	다

```
# df_tt와 df_paris의 곱집합 만들기
df_cross = df_tt.merge(df_paris, how='cross')
df_cross
```

	상가업소번호	상호명	시군구명	경도	위도	p_업소번호	p_경도	p_위도
0	20844443	신전떡볶이	중랑구	127.1014	37.6003	26669769	126.8839	37.4903
1	20844443	신전떡볶이	중랑구	127.1014	37.6003	9171904	127.0330	37.6078
2	20844443	신전떡볶이	중랑구	127.1014	37.6003	23396020	126.9886	37.5742
...	...	...	...	...	...	...	...	...
93337	20081411	국대떡볶이	서초구	126.9966	37.4814	20589842	126.9019	37.4522
93338	20081411	국대떡볶이	서초구	126.9966	37.4814	20595157	127.1310	37.5101
93339	20081411	국대떡볶이	서초구	126.9966	37.4814	17182192	127.1644	37.5569

93340 rows × 8 columns

** 두 가게 사이의 거리구하기 **

두 지점의 위도와 경도를 입력하면 거리를 구해주는 라이브러리 haversine 을 이용한다

```
# haversine 라이브러리 설치하기
!pip install haversine
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting haversine
  Downloading haversine-2.8.0-py2.py3-none-any.whl (7.7 kB)
Installing collected packages: haversine
Successfully installed haversine-2.8.0
```

```
# haversine 사용법

from haversine import haversine
#  서울과 파리의 위도와 경도(위도 경도순)
seoul = [37.541, 126.986]
paris = [48.8567, 2.3508]

# 서울과 파리사이의 거리 계산
haversine(seoul, paris, unit='m') # 변수로 넣을 경우
haversine([37.541, 126.986], [48.8567, 2.3508], unit='m') # 리스트로
```

```
8968562.580161477
```

단 haversine 함수를 하나의 행의 여러열로 적용하려면 데이터프레임에 바로 적용되는 함수가 아니므로 apply의 도움이 필요하다

```
# df_cross에 거리열 만들기
from haversine import haversine
df_cross['거리'] = df_cross.apply(
    lambda x: haversine([x[4],x[3]], [x[7], x[6]], unit='m'),
    axis=1)
df_cross
```

	상가업소번호	상호명	시군구명	경도	위도	p_업소번호	p_경도	p_위도	거리
0	20844443	신전떡볶이	중랑구	127.1014	37.6003	26669769	126.8839	37.4903	22745.8986
1	20844443	신전떡볶이	중랑구	127.1014	37.6003	9171904	127.0330	37.6078	6080.4618
2	20844443	신전떡볶이	중랑구	127.1014	37.6003	23396020	126.9886	37.5742	10356.9671
...	...	...	...	...	...	...	...	...	...
93337	20081411	국대떡볶이	서초구	126.9966	37.4814	20589842	126.9019	37.4522	8969.1388
93338	20081411	국대떡볶이	서초구	126.9966	37.4814	20595157	127.1310	37.5101	12272.9853
93339	20081411	국대떡볶이	서초구	126.9966	37.4814	17182192	127.1644	37.5569	17013.7789

93340 rows × 9 columns

4. 떡볶이 프랜차이즈와 파리바게뜨의 거리 다양하게 집계하기

```
# 프로젝트 코드1 (도합 3개의 코드를 모두 copy해서 실행해야 한다)
import pandas as pd
pd.options.display.max_rows = 6 # 판다스 1.4+ 에서 6행까지만 출력
pd.options.display.float_format = '{:.4f}'.format # 소수점 네자리 까지만 출력
url = 'https://raw.githubusercontent.com/panda-kim/csv_files/main/food_business_seoul_20221231.csv'
tt = ['감탄떡볶이', '국대떡볶이', '죠스떡볶이', '엽기떡볶이', '신전떡볶이']
df = pd.read_csv(url)

# 파리바게뜨 데이터 필터링하고 정제하기
cond1 = df['상호명'].str.contains(r'파리바게뜨|파리바게트')
df_paris = df.loc[cond1, ['상가업소번호', '경도', '위도']].copy()
df_paris = (df_paris
            .set_axis(['p_업소번호', 'p_경도', 'p_위도'], axis=1)
            .reset_index(drop=True))

# 떡볶이 프랜차이즈 데이터 필터링하고 정제하기
df_tt = df.copy()
df_tt['상호명'] = df_tt['상호명'].str.extract(r'({})'.format('|'.join(tt)))[0]
df_tt = (df_tt
         .dropna(subset=['상호명'])
         .iloc[:, [0, 1, 5, 8, 9]]
         .reset_index(drop=True))

# df_tt와 df_paris의 곱집합 만들기
df_cross = df_tt.merge(df_paris, how='cross')

# 프로젝트 코드2
!pip install haversine
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: haversine in /usr/local/lib/python3.9/dist-packages (2.8.0)
```

```
# 프로젝트 코드3
from haversine import haversine
df_cross['거리'] = df_cross.apply(
    lambda x: haversine([x[4],x[3]], [x[7], x[6]], unit='m'),
    axis=1)

# 개별 떡볶이 매장과 파리바게뜨와의 거리
df_dis = df_cross.groupby(['상가업소번호', '상호명'])['거리'].min().reset_index()
df_dis
```

	상가업소번호	상호명	거리
0	8616690	신전떡볶이	392.8313
1	8625310	신전떡볶이	606.1897
2	10520263	죠스떡볶이	327.7421
...	...	...	...
257	28513476	엽기떡볶이	446.9590
258	28514842	엽기떡볶이	753.3187
259	28519528	신전떡볶이	725.6968

260 rows × 3 columns

```
# 프랜차이즈 별 파리바게뜨와의 평균 거리
df_dis.groupby('상호명')['거리'].mean()
```

```
상호명
감탄떡볶이    474.1331
국대떡볶이    377.8993
신전떡볶이    420.6615
엽기떡볶이    476.6015
죠스떡볶이    467.2652
Name: 거리, dtype: float64
```

```
# 프랜차이즈 별 파리바게뜨와의 평균 거리와 매장개수
df_dis.groupby('상호명')['거리'].agg(['mean', 'count'])
```

상호명	mean	count
감탄떡볶이	474.1331	55
국대떡볶이	377.8993	14
신전떡볶이	420.6615	58
엽기떡볶이	476.6015	92
죠스떡볶이	467.2652	41

```
# 프랜차이즈 별 파리바게뜨와 50m미만인 매장의 평균거리와 매장개수
cond = df_dis['거리'] < 50
df_dis[cond].groupby('상호명')['거리'].agg(['mean', 'count'])
```

상호명	mean	count
감탄떡볶이	15.2256	7
국대떡볶이	19.3064	2
신전떡볶이	49.3257	1
엽기떡볶이	27.7970	5
죠스떡볶이	13.6234	5

```
# 프랜차이즈 별 파리바게뜨와 30m미만인 매장의 평균거리와 매장개수
cond = df_dis['거리'] < 30
df_dis[cond].groupby('상호명')['거리'].agg(['mean', 'count'])
```

상호명	mean	count
감탄떡볶이	0.0000	4
국대떡볶이	0.0000	1
엽기떡볶이	17.9833	3
죠스떡볶이	4.7981	4

```
# 거리에 따라 집계를 해주는 함수를 만들자
def distance(x):
  cond = df_dis['거리'] < x
  return df_dis[cond].groupby('상호명')['거리'].agg(['mean', 'count'])
distance(100)
```

상호명	mean	count
감탄떡볶이	33.6436	10
국대떡볶이	41.0192	3
신전떡볶이	80.7178	5
엽기떡볶이	59.2283	11
죠스떡볶이	51.1618	11

5. 감탄떡볶이의 입점전략 시각화하기

pyecharts

echarts를 python에서 사용할 수 있게 해주는 라이브러리

- 기본값이 예쁘다
- 반응형이다

pyecharts 사이트 링크 : https://pyecharts.org/#/en-us/

갤러리 링크 : https://gallery.pyecharts.org/#/README_EN

pyecharts 설치 (1.9.1버전 설치)

```
!pip install --upgrade pyecharts==1.9.1
```

pandasecharts

데이터 프레임에 pyecharts를 쉽게 사용하게 해주는 라이브러리

pandasecharts 설치

```
!pip install pandasecharts
```

```
# pyecharts 1.9.1 설치
!pip install --upgrade pyecharts==1.9.1
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyecharts==1.9.1
  Downloading pyecharts-1.9.1-py3-none-any.whl (135 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.6/135.6 KB 6.9 MB/s eta 0:00:00
Collecting simplejson
  Downloading simplejson-3.18.4-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (136 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 136.8/136.8 KB 15.5 MB/s eta 0:00:00
```

```
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from pyecharts==1.9.1) (3.1.2)
Requirement already satisfied: prettytable in /usr/local/lib/python3.9/dist-packages (from pyecharts==1.9.1) (0.7.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->pyecharts==1.9.1) (2.1.2)
Installing collected packages: simplejson, pyecharts
Successfully installed pyecharts-1.9.1 simplejson-3.18.4
```

```
# pandasecharts 설치
!pip install pandasecharts
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pandasecharts
  Downloading pandasecharts-0.5-py3-none-any.whl (15 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from pandasecharts) (1.4.4)
Requirement already satisfied: pyecharts>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from pandasecharts) (1.9.1)
Requirement already satisfied: simplejson in /usr/local/lib/python3.9/dist-packages (from pyecharts>=1.0.0->pandasecharts) (3.18.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from pyecharts>=1.0.0->pandasecharts) (3.1.2)
Requirement already satisfied: prettytable in /usr/local/lib/python3.9/dist-packages (from pyecharts>=1.0.0->pandasecharts) (0.7.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->pandasecharts) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas->pandasecharts) (2.8.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.9/dist-packages (from pandas->pandasecharts) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas->pandasecharts) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->pyecharts>=1.0.0->pandasecharts) (2.1.2)
Installing collected packages: pandasecharts
Successfully installed pandasecharts-0.5
```

```
# haversine 설치
!pip install haversine
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: haversine in /usr/local/lib/python3.9/dist-packages (2.8.0)
```

```
# 프로젝트 코드
import pandas as pd
from haversine import haversine
from pyecharts.charts import Timeline, Grid
from pandasecharts import echart
import IPython

pd.options.display.max_rows = 6 # 판다스 1.4+ 에서 6행까지만 출력
pd.options.display.float_format = '{:.4f}'.format # 소수점 네자리 까지만 출력
url = 'https://raw.githubusercontent.com/panda-kim/csv_files/main/food_business_seoul_20221231.csv'
tt = ['감탄떡볶이', '국대떡볶이', '죠스떡볶이', '엽기떡볶이', '신전떡볶이']
df = pd.read_csv(url)

# 파리바게뜨 데이터 필터링하고 정제하기
cond1 = df['상호명'].str.contains(r'파리바게뜨|파리바게트')
df_paris = df.loc[cond1, ['상가업소번호', '경도', '위도']].copy()
df_paris = (df_paris
            .set_axis(['p_업소번호', 'p_경도', 'p_위도'], axis=1)
            .reset_index(drop=True))

# 떡볶이 프랜차이즈 데이터 필터링하고 정제하기
df_tt = df.copy()
df_tt['상호명'] = df_tt['상호명'].str.extract(r'({})'.format('|'.join(tt)))[0]
df_tt = (df_tt
         .dropna(subset=['상호명'])
         .iloc[:, [0, 1, 5, 8, 9]]
         .reset_index(drop=True))

# df_tt와 df_paris의 곱집합 만들기
df_cross = df_tt.merge(df_paris, how='cross')

# df_cross에 거리열 만들기
df_cross['거리'] = df_cross.apply(
    lambda x: haversine([x[4],x[3]], [x[7], x[6]], unit='m'),
```

```
    axis=1)

# 개별 떡볶이 매장과 파리바게뜨와의 거리
df_dis = df_cross.groupby(['상가업소번호', '상호명'])['거리'].min().reset_index()


# 거리에 따라 프랜차이즈별 평균 거리와 매장수를 집계를 해주는 함수
def distance(x):
  cond = df_dis['거리'] < x
  return df_dis[cond].groupby('상호명')['거리'].agg(['mean', 'count'])
```

pyecharts로 만든 그래프를 구글 코랩에 출력하는 코드

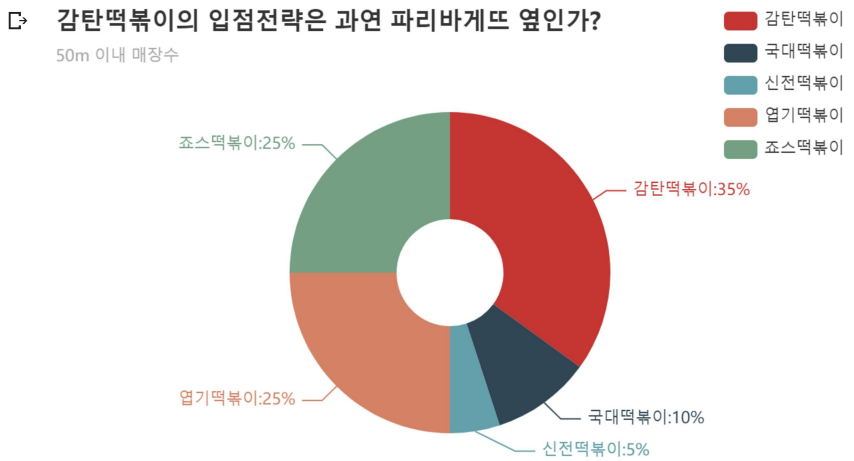
```
IPython.display.HTML(filename='/content/render.html')
```

거리마다 데이터를 집계해주는 사용자 정의 함수 distance 를 활용한다

```
# distance 함수로 50m의 집계하기(reset_index로 상호명도 열로 만든다)
distance(50).reset_index()
```

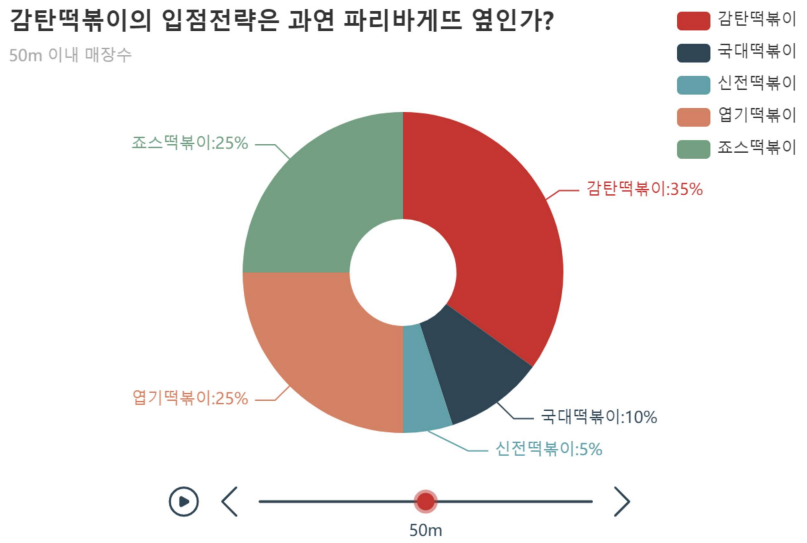
	상호명	mean	count
0	감탄떡볶이	15.2256	7
1	국대떡볶이	19.3064	2
2	신전떡볶이	49.3257	1
3	엽기떡볶이	27.7970	5
4	죠스떡볶이	13.6234	5

```
# 50m의 데이터로 원형 그래프 만들기
df_g = distance(50).reset_index()
df_g.echart.pie(x='상호명', y='count', figsize=(600, 400),
                radius=['20%', '60%'], label_opts={'position':'outer'},
                title='감탄떡볶이의 입점전략은 과연 파리바게뜨 옆인가?',
                legend_opts={'pos_right':'0%', 'orient':'vertical'},
                subtitle='50m 이내 매장수').render()
IPython.display.HTML(filename='/content/render.html')
```

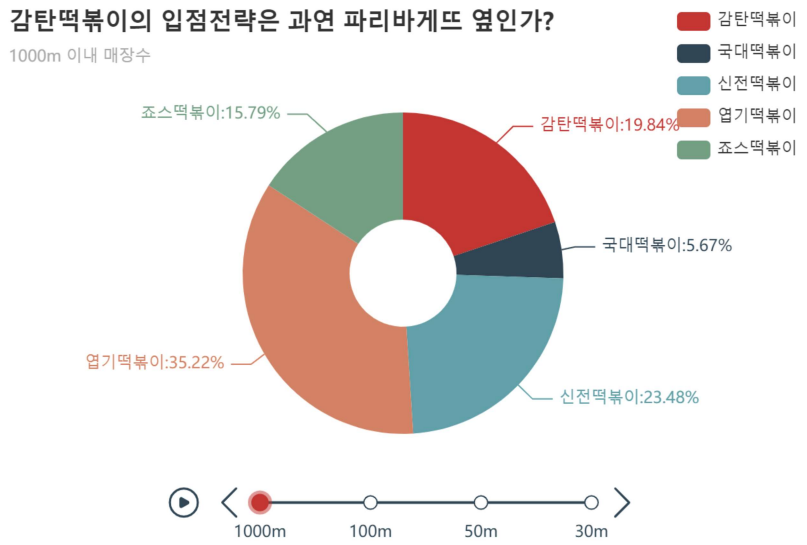



```
# 앞서 만든 그래프를 timeline으로 만들기
tl = Timeline({'width':'600px', 'height': '400px'})
```

```
df_g = distance(50).reset_index()
pie1 = df_g.echart.pie(x='상호명', y='count', figsize=(600, 400),
                       radius=['20%', '60%'], label_opts={'position':'outer'},
                       title='감탄떡볶이의 입점전략은 과연 파리바게뜨 옆인가?',
                       legend_opts={'pos_right':'0%', 'orient':'vertical'},
                       subtitle='50m 이내 매장수')
tl.add(pie1, '50m').render()
IPython.display.HTML(filename='/content/render.html')
```




```
# 반복문으로 1000m, 100m, 50m, 30m를 타임라인으로 만들기
tl = Timeline({'width':'600px', 'height': '400px'})
for i in [1000, 100, 50, 30]:
  df_g = distance(i).reset_index()
  pie1 = df_g.echart.pie(x='상호명',
                         y='count',
                         figsize=(600, 400),
                         radius=['20%', '60%'],
                         label_opts={'position':'outer'},
                         title='감탄떡볶이의 입점전략은 과연 파리바게뜨 옆인가?',
                         legend_opts={'pos_right':'0%', 'orient':'vertical'},
                         subtitle='{}m 이내 매장수'.format(i))
  tl.add(pie1, '{}m'.format(i)).render()
IPython.display.HTML(filename='/content/render.html')
```



✓ 0초 오후 5:15에 완료됨