Influence of riders in bicycle accidents in Madrid

Capstone Project
IBM Data Science Professional Certificate

May 2020

Author: Isidro Brevers Gómez

E-mail: ibrevers@gmail.com

GitHub: https://github.com/ibrevers/Influence-of-riders-in-bicycle-accidents-in-Madrid

Blogpost: https://www.linkedin.com/pulse/influence-riders-bicycle-accidents-madrid-isidro-brevers-g%C3%B3mez/

Table of Contents

1. Introduction and purpose

In recent years, there has been an increasing demand of food delivery services in Madrid, Spain which has led to a significant number of bicycle riders in the city.

The purpose of this project is to analyze whether the number of bicycle riders has any influence in regard to the number of bicycle accidents in Madrid.

2. Methodology

Data described in *3. Data description and source* has been obtained from multiple sources, stored in Jupyter Notebooks hosted by IBM Watson Studio and processed using Python language.

Data science, data analysis, scientific computing, chart and map visualization, geocoding and geolocation, machine learning and statistical methods and techniques have been used to transform and analyze source data for the purposes of measuring whether riders have any influence in the number of bicycle accidents in Madrid per district.

Influence has been measured as the statistical correlation between the number of bicycles involved in traffic accidents in Madrid in 2019, and the number of food venues per district.

A detailed description of Python libraries used is included in *Annex I: Python libraries*, and the code of the project, as well as the data and visualization outputs (charts and maps) are available in GitHub (https://github.com/ibrevers/Influence-of-riders-in-bicycle-accidents-in-Madrid).

3. Data description and source

The following data has been used for the purposes of the project:

Traffic accidents in Madrid in 2019

Information has been obtained from the data published in the open data website (https://datos.madrid.es/portal/site/egob/) of the Council of Madrid, comprehensive of detailed information about traffic accidents in Madrid in 2019.

A more detailed description of the data set and how the CSV file is structured can be found in https://datos.madrid.es/FWProjects/egob/Catalogo/Seguridad/Ficheros/Estructura_DS_Accidentes_trafico_desde_2019.pdf.

Coordinates for Madrid districts

Information has been obtained from the GeoDatos website (https://www.geodatos.net/), which aggregates information available through the Google Maps API.

Food venues in Madrid

Information has been obtained through the Foursquare's Developers API (https://developer.foursquare.com/).

4. Data analysis

[Jupyter Notebook]

4.1 Bicycle accidents in Madrid in 2019 by district

A CSV file (accidents2019.csv) containing information regarding traffic accidents in Madrid in 2019 is downloaded from the Madrid's Council open data website:

```
# Download the information
!wget -q -O 'accidents2019.csv' https://datos.madrid.es/egob/catalogo/300228-19-accidentes-trafico-detalle.csv
```

Check the encoding of the CSV file:

```
# Check the encoding of the csv
!pip install chardet
import chardet

with open("accidents2019.csv", 'rb') as file:
    print(chardet.detect(file.read()))
```

```
Requirement already satisfied: chardet in /opt/conda/envs/Python36/lib/python3.6/site-packages (3.0.4)
{'encoding': 'ISO-8859-1', 'confidence': 0.73, 'language': ''}
```

Read the information in the CSV file into a DataFrame (*df_accidents*):

```
# Read the csv downloaded into a df
df_accidents = pd.read_csv('accidents2019.csv', sep=';', encoding='ISO-8859-1')
df_accidents.head(5)
```

	Nº EXPEDIENTE	FECHA	HORA	CALLE	NÚMERO	DISTRITO	TIPO ACCIDENTE	ESTADO METEREOLÓGICO	TIPO VEHÍCULO	TIPO PERSONA	RANGO EDAD
0	2019S000020	01/01/2019	23:30	CALL. FUENCARRAL	149	CHAMBERÍ	Caída	Despejado	Ciclomotor	Conductor	DE 25 A 29 AÑOS
1	2019S000017	01/01/2019	22:15	CALL. OCA / CALL. PINZON	-	CARABANCHEL	Colisión fronto-lateral	Despejado	Turismo	Conductor	DE 40 A 44 AÑOS
2	2019S000017	01/01/2019	22:15	CALL. OCA / CALL. PINZON	-	CARABANCHEL	Colisión fronto-lateral	Despejado	Ciclomotor	Conductor	DE 35 A 39 AÑOS
3	2019S001812	01/01/2019	21:40	CALL. BAILEN / CUSTA. SAN VICENTE	-	CENTRO	Colisión fronto-lateral	Despejado	Turismo	Conductor	DE 40 A 44 AÑOS
4	2019S001812	01/01/2019	21:40	CALL. BAILEN / CUSTA. SAN VICENTE	-	CENTRO	Colisión fronto-lateral	Despejado	Turismo	Conductor	DE 30 A 34 AÑOS

Check the size of the DataFrame:

```
# Check the size of the df
df_accidents.shape
```

```
(51806, 14)
```

The DataFrame has 14 columns and 51,806 rows.

Data cleansing: drop useless columns and rename columns:

```
# Delete columns
df_accidents = df_accidents.drop(df_accidents.columns[[0, 2, 3, 4, 6, 7, 9, 10, 11, 12, 13]], axis=1)

# Rename columns
df_accidents.rename(columns={'Nº  EXPEDIENTE': 'EXP', 'FECHA':'DATE', 'DISTRITO':'DISTRICT', 'TIPO VEHÍCULO':'VEHICLE'}, inplace
=True)

df_accidents.head(5)
```

	DATE	DISTRICT	VEHICLE
0	01/01/2019	CHAMBERÍ	Ciclomotor
1	01/01/2019	CARABANCHEL	Turismo
2	01/01/2019	CARABANCHEL	Ciclomotor
3	01/01/2019	CENTRO	Turismo
4	01/01/2019	CENTRO	Turismo

Explore the number of accidents by type of vehicle:

```
df_accidents['VEHICLE'].value_counts()
```

```
Turismo                             36499
Motocicleta > 125cc                  3527
Furgoneta                            3125
Motocicleta hasta 125cc              2529
Autobús                              1408
Camión rígido                        1167
Bicicleta                             884
Ciclomotor                            809
Todo terreno                          689
Otros vehículos con motor             330
Tractocamión                          195
Maquinaria de obras                   117
Vehículo articulado                   106
Autobús articulado                     83
Sin especificar                        42
Ciclo                                  20
VMU eléctrico                          18
Cuadriciclo ligero                     15
Cuadriciclo no ligero                  14
Autocaravana                           13
Patinete                                9
Bicicleta EPAC (pedaleo asistido)       7
Otros vehículos sin motor               7
Semiremolque                            5
Autobus EMT                             3
Remolque                                3
Microbús <= 17 plazas                   2
Caravana                                1
Tranvía                                 1
Name: VEHICLE, dtype: int64
```

Reduce data in the DataFrame to accidents where bicycles where involved:

```
# We only need information for accidents where the type of vehicle involved was a bicycle
df_accidents = df_accidents[df_accidents['VEHICLE']=='Bicicleta']

# We drop the column vehicle as it does not provide useful information now
df_accidents = df_accidents.drop(df_accidents.columns[[2]], axis=1)

# Reset index
df_accidents.reset_index(inplace=True, drop=True)
df_accidents.head(5)
```

	DATE	DISTRICT
0	2019-01-01	SALAMANCA
1	2019-02-01	HORTALEZA
2	2019-03-01	VILLA DE VALLECAS
3	2019-03-01	VILLA DE VALLECAS
4	2019-03-01	VILLA DE VALLECAS

Check the size of the new DataFrame and whether there is any missing value:

```
# Check the new size of the df
df_accidents.shape
```

```
(884, 2)
```

```
# Check if there is any missing value
df_accidents.isna().sum()
```

```
DATE        0
DISTRICT    0
dtype: int64
```

The new DataFrame has 2 columns and 884 rows, and no information is missing.

Create a new DataFrame (*df_accidents_summary*) grouping bicycle accidents by district:

```
df_accidents_summary = df_accidents.groupby('DISTRICT').count().reset_index()
df_accidents_summary.rename(columns={'DATE':'NO ACCIDENTS'}, inplace=True)
df_accidents_summary.sort_values(by=['NO ACCIDENTS'], ascending=False, inplace=True)
df_accidents_summary.reset_index(inplace=True, drop=True)
df_accidents_summary
```

Data cleansing (replace district names with accents):

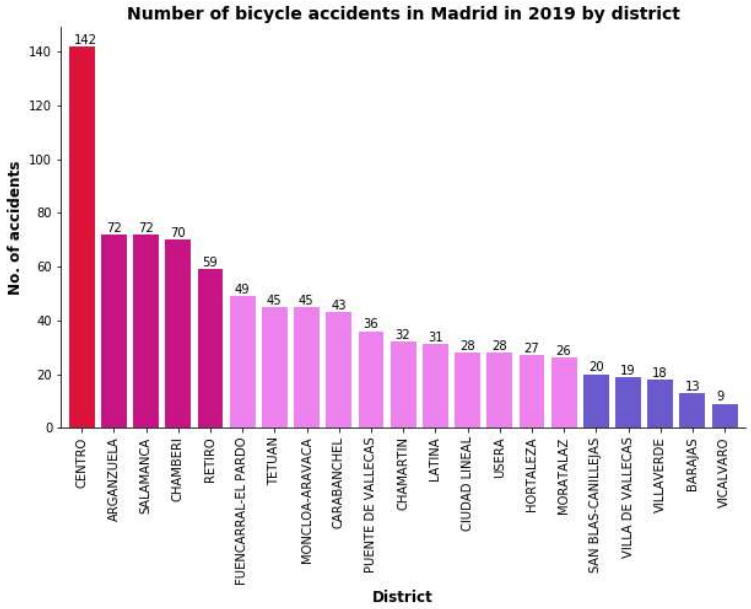
```
# Data cleansing: change the names with accent characters
df_accidents_summary.replace({'CHAMBERÍ':'CHAMBERI'}, inplace=True)
df_accidents_summary.replace({'TETUÁN':'TETUAN'}, inplace=True)
df_accidents_summary.replace({'CHAMARTÍN':'CHAMARTIN'}, inplace=True)
df_accidents_summary.replace({'VICÁLVARO':'VICALVARO'}, inplace=True)
df_accidents_summary
```

	DISTRICT	NO ACCIDENTS
0	CENTRO	142
1	ARGANZUELA	72
2	SALAMANCA	72
3	CHAMBERI	70
4	RETIRO	59
5	FUENCARRAL-EL PARDO	49
6	TETUAN	45
7	MONCLOA-ARAVACA	45
8	CARABANCHEL	43
9	PUENTE DE VALLECAS	36
10	CHAMARTIN	32
11	LATINA	31
12	CIUDAD LINEAL	28
13	USERA	28
14	HORTALEZA	27
15	MORATALAZ	26
16	SAN BLAS-CANILLEJAS	20
17	VILLA DE VALLECAS	19
18	VILLAVERDE	18
19	BARAJAS	13
20	VICALVARO	9

DataFrame is exported to a CSV file (*Bicycle_Accidents.csv*).

Chart 1: (Bar chart) Number of bicycle accidents in Madrid in 2019 by district.

```
df_barchart = df_accidents_summary.set_index(['DISTRICT'])

# Color array for the heat
c = [('crimson', 'mediumvioletred', 'mediumvioletred', 'mediumvioletred', 'mediumvioletred', 'violet', 'violet', 'violet', 'viol
et', 'violet', 'violet', 'violet', 'violet',
      'violet', 'violet', 'violet', 'slateblue', 'slateblue', 'slateblue', 'slateblue', 'slateblue')]

# Add markers to bar chart
ax = df_barchart.plot(kind='bar', figsize=(10, 6), width=0.8, color=c)

xs = df_accidents_summary['DISTRICT']
ys = df_accidents_summary['NO ACCIDENTS']

plt.xlabel('District', fontweight='bold', fontsize=12)
plt.ylabel('No. of accidents', fontweight='bold', fontsize=12)
plt.title('Number of bicycle accidents in Madrid in 2019 by district', fontweight='bold', fontsize=14)

ax.set_facecolor('none')
ax.legend().set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

for index,data in enumerate(ys):
    plt.text(x=index-0.25 , y =data+1 , s=f"{data}" , fontdict=dict(fontsize=10))

# Display chart
plt.show()
```



4.2 Geolocating Madrid districts

Information obtained from the GeoDatos website with the coordinates for Madrid districts has been included in a CSV file (Madrid_geo.csv) and read into a DataFrame (*df_madrid_geo*):

```
# @hidden_cell

def __iter__(self): return 0

body =                                                                                                    Key='Ma
drid_Geo.csv')['Body']

# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_madrid_geo = pd.read_csv(body, sep=',', encoding='ISO-8859-1')
```

```
df_madrid_geo.head()
```

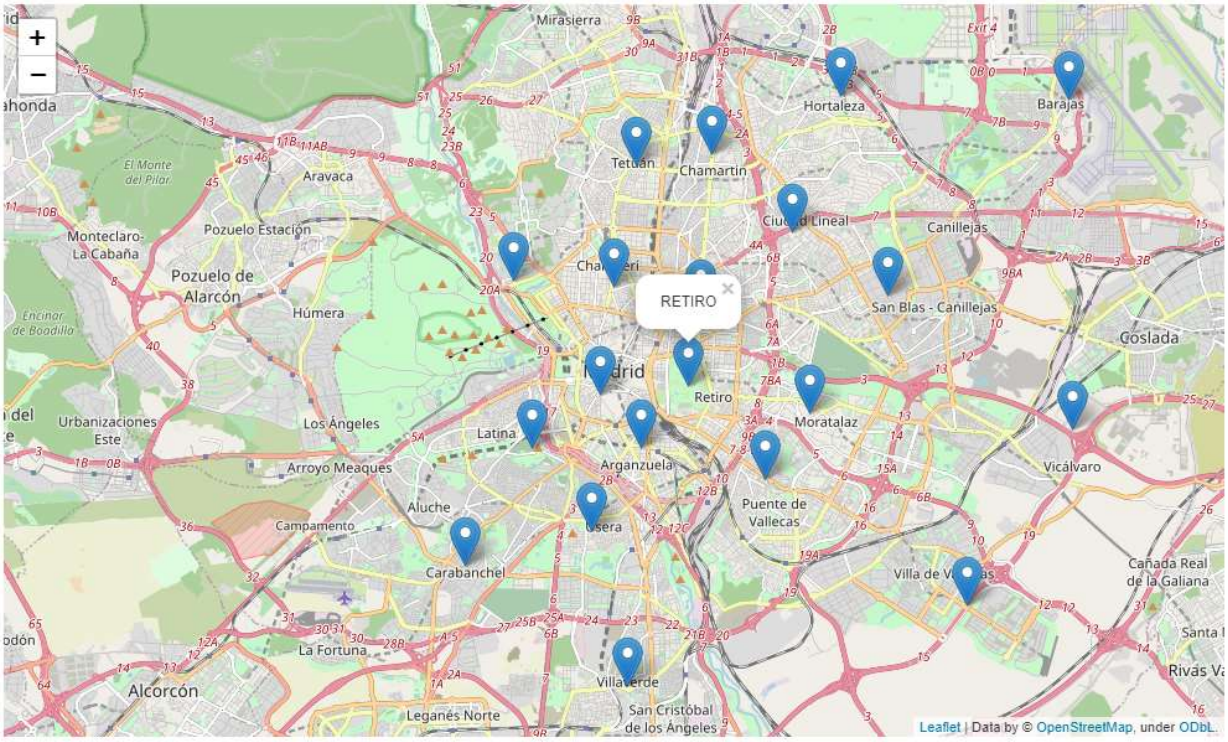
	DISTRICT	LATITUDE	LONGITUDE
0	ARGANZUELA	40.400211	-3.696180
1	BARAJAS	40.473659	-3.577770
2	CARABANCHEL	40.375215	-3.744876
3	CENTRO	40.411516	-3.707644
4	CHAMARTIN	40.462059	-3.676600

Coordinates for the city of Madrid, Spain are obtained from the Nominatim geocoding service:

```
# Obtain the coordinates for Madrid
address = 'Madrid'
geolocator = Nominatim(user_agent="madrid_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
```

Map 1: (Marker map) Madrid districts.

```
map_madrid1 = folium.Map(location=[latitude, longitude], zoom_start=12)

# Add markers to map
for lat, lng, district in zip(df_madrid_geo['LATITUDE'], df_madrid_geo['LONGITUDE'], df_madrid_geo['DISTRICT']):
    label = '{}'.format(district)
    label = folium.Popup(label, parse_html=True)
    folium.Marker(
        location=[lat, lng],
        icon=None,
        popup=label,
    ).add_to(map_madrid1)

# Display map
map_madrid1
```



4.3 Geolocating bicycle accidents in Madrid in 2019 by district

Create a new DataFrame (*df_accidents_geo*) with the number of bicycle accidents and coordinates per district as a result of merging *df_accidents_summary* and *df_madrid_geo*:

```
# Merge both tables
df_accidents_geo = pd.merge(df_accidents_summary, df_madrid_geo, on='DISTRICT')
```

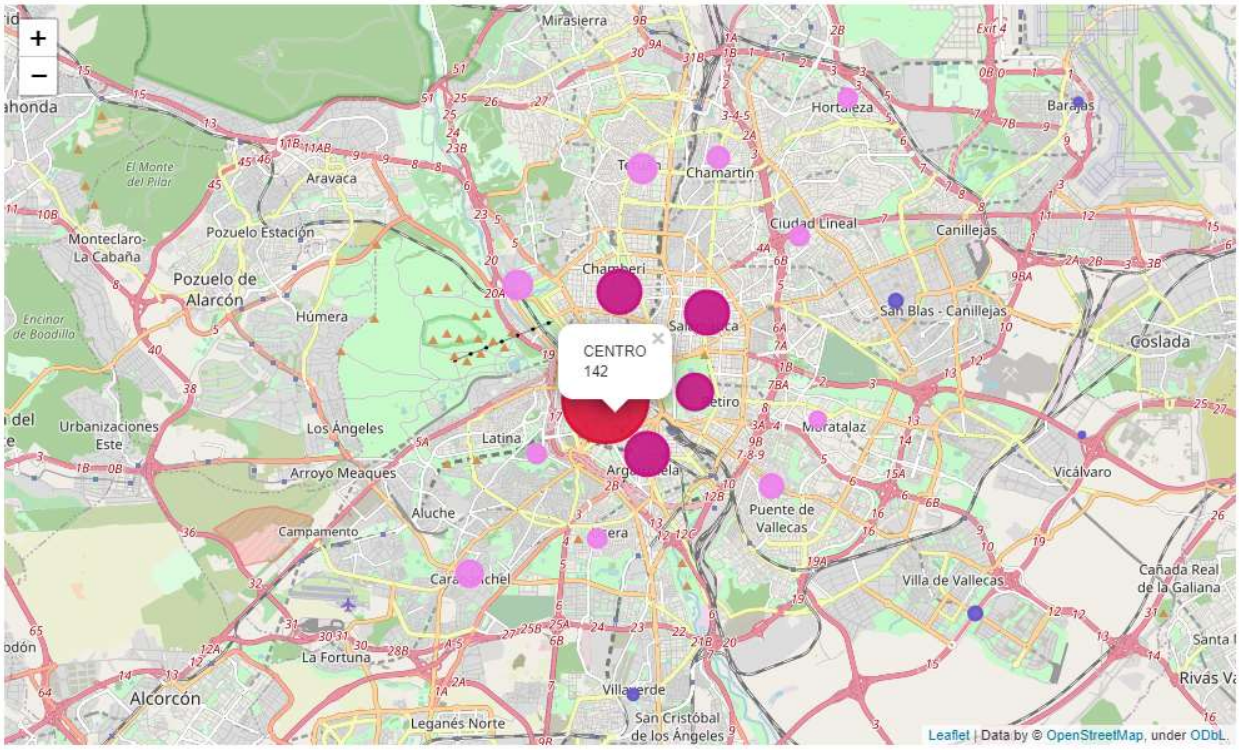
```
df_accidents_geo
```

	DISTRICT	NO ACCIDENTS	LATITUDE	LONGITUDE
0	CENTRO	142	40.411516	-3.707644
1	ARGANZUELA	72	40.400211	-3.696180
2	SALAMANCA	72	40.429722	-3.679750
3	CHAMBERI	70	40.434040	-3.703790
4	RETIRO	59	40.413170	-3.683070
5	FUENCARRAL-EL PARDO	49	40.498402	-3.731400
6	TETUAN	45	40.459751	-3.697500
7	MONCLOA-ARAVACA	45	40.435471	-3.731700
8	CARABANCHEL	43	40.375215	-3.744876
9	PUENTE DE VALLECAS	36	40.393539	-3.662000
10	CHAMARTIN	32	40.462059	-3.676600
11	LATINA	31	40.400211	-3.726519
12	CIUDAD LINEAL	28	40.445668	-3.654384
13	USERA	28	40.382593	-3.709875
14	HORTALEZA	27	40.474441	-3.641100
15	MORATALAZ	26	40.407421	-3.649350
16	SAN BLAS-CANILLEJAS	20	40.432184	-3.627880
17	VILLA DE VALLECAS	19	40.366955	-3.606064
18	VILLAVERDE	18	40.349998	-3.700000
19	BARAJAS	13	40.473659	-3.577770
20	VICALVARO	9	40.404200	-3.576780

Map 2: (Bubble heat map) Bicycle accidents in Madrid in 2019 by district.

```
# Bubble heat map

# New df clustering the districts into bins on the no of accidents
bins = [0, 25, 50, 125]
cluster = np.searchsorted(bins, df_accidents_geo['NO ACCIDENTS'].values)
df_accidents_geo['CLUSTER'] = cluster

# New df for each bin
df_cluster1 = df_accidents_geo[df_accidents_geo['CLUSTER']==1].reset_index()
df_cluster2 = df_accidents_geo[df_accidents_geo['CLUSTER']==2].reset_index()
df_cluster3 = df_accidents_geo[df_accidents_geo['CLUSTER']==3].reset_index()
df_cluster4 = df_accidents_geo[df_accidents_geo['CLUSTER']==4].reset_index()

# New empty map
map_madrid2 = folium.Map(location=[latitude, longitude], zoom_start=12)

# Add markers
for i in range(0,len(df_cluster1)):
   folium.Circle(
      location=[df_cluster1.iloc[i]['LATITUDE'], df_cluster1.iloc[i]['LONGITUDE']],
      popup='{} \n {}'.format(df_cluster1.iloc[i]['DISTRICT'], df_cluster1.iloc[i]['NO ACCIDENTS']),
      radius=float(df_cluster1.iloc[i]['NO ACCIDENTS'])*7,
      color='slateblue',
      fill=True,
      fill_color='slateblue',
      fill_opacity=0.9,
   ).add_to(map_madrid2)
for i in range(0,len(df_cluster2)):
   folium.Circle(
      location=[df_cluster2.iloc[i]['LATITUDE'], df_cluster2.iloc[i]['LONGITUDE']],
      popup='{} \n {}'.format(df_cluster2.iloc[i]['DISTRICT'], df_cluster2.iloc[i]['NO ACCIDENTS']),
      radius=float(df_cluster2.iloc[i]['NO ACCIDENTS'])*7,
      color='violet',
      fill=True,
      fill_color='violet',
      fill_opacity=0.9,
   ).add_to(map_madrid2)
```

```
for i in range(0,len(df_cluster2)):
   folium.Circle(
      location=[df_cluster2.iloc[i]['LATITUDE'], df_cluster2.iloc[i]['LONGITUDE']],
      popup='{} \n {}'.format(df_cluster2.iloc[i]['DISTRICT'], df_cluster2.iloc[i]['NO ACCIDENTS']),
      radius=float(df_cluster2.iloc[i]['NO ACCIDENTS'])*7,
      color='violet',
      fill=True,
      fill_color='violet',
      fill_opacity=0.9,
   ).add_to(map_madrid2)
for i in range(0,len(df_cluster3)):
   folium.Circle(
      location=[df_cluster3.iloc[i]['LATITUDE'], df_cluster3.iloc[i]['LONGITUDE']],
      popup='{} \n {}'.format(df_cluster3.iloc[i]['DISTRICT'], df_cluster3.iloc[i]['NO ACCIDENTS']),
      radius=float(df_cluster3.iloc[i]['NO ACCIDENTS'])*7,
      color='mediumvioletred',
      fill=True,
      fill_color='mediumvioletred',
      fill_opacity=0.9,
   ).add_to(map_madrid2)
for i in range(0,len(df_cluster4)):
   folium.Circle(
      location=[df_cluster4.iloc[i]['LATITUDE'], df_cluster4.iloc[i]['LONGITUDE']],
      popup='{} \n {}'.format(df_cluster4.iloc[i]['DISTRICT'], df_cluster4.iloc[i]['NO ACCIDENTS']),
      radius=float(df_cluster4.iloc[i]['NO ACCIDENTS'])*7,
      color='crimson',
      fill=True,
      fill_color='crimson',
      fill_opacity=0.9,
   ).add_to(map_madrid2)

map_madrid2
```



4.4 Geolocating Madrid Venues (Foursquare API)

Define Foursquare credentials (hidden) and version (April 30, 2020):

```
# @hidden_cell
CLIENT_ID =                                                    # your Foursquare ID
CLIENT_SECRET =                                                    # your Foursquare Secret
VERSION = '20200430' # Foursquare API version
```

Define a function to obtain the top 100 venues for each district (max value for the API) in a 400 meters radius for the 'food' category:

```
def getNearbyVenues(names, latitudes, longitudes, radius=400, limit=500, categoryId="4d4b7105d754a06374d81259"):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?categoryId={}&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&
limit={}'.format(
            categoryId,
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            limit)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']
```

```
        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['DISTRICT',
                  'LATITUDE',
                  'LONGITUDE',
                  'VENUE',
                  'VENUE LATITUDE',
                  'VENUE LONGITUDE',
                  'VENUE CATEGORY',]
    return(nearby_venues)
```

Create a new DataFrame (*df_venues*) including venues' data per district:

```
df_venues = getNearbyVenues(names=df_madrid_geo['DISTRICT'],
                                latitudes=df_madrid_geo['LATITUDE'],
                                longitudes=df_madrid_geo['LONGITUDE']
                               )
```

```
df_venues.head()
```

	DISTRICT	LATITUDE	LONGITUDE	VENUE	VENUE LATITUDE	VENUE LONGITUDE	VENUE CATEGORY
0	ARGANZUELA	40.400211	-3.69618	La Pequeña Graná	40.399574	-3.698550	Tapas Restaurant
1	ARGANZUELA	40.400211	-3.69618	Havana Blues	40.402050	-3.698488	Cuban Restaurant
2	ARGANZUELA	40.400211	-3.69618	El Quinto Pecado	40.400028	-3.694446	Gastropub
3	ARGANZUELA	40.400211	-3.69618	Tres Cerditos	40.397316	-3.694184	Chinese Restaurant
4	ARGANZUELA	40.400211	-3.69618	Restaurante Buen Gusto	40.401766	-3.698961	Chinese Restaurant

Check the size of the DataFrame:

```
# Check the size of the df
df_venues.shape
```

```
(410, 7)
```

The DataFrame has 7 columns and 410 rows.

Explore the number of venues per district:

```
df_venues['DISTRICT'].value_counts()
```

```
CENTRO                 100
CHAMBERI                85
SALAMANCA               34
ARGANZUELA              31
TETUAN                  24
CIUDAD LINEAL           24
BARAJAS                 20
PUENTE DE VALLECAS      18
SAN BLAS-CANILLEJAS     16
HORTALEZA                9
USERA                    8
CHAMARTIN                8
MORATALAZ                7
VILLA DE VALLECAS        5
LATINA                   5
RETIRO                   5
VILLAVERDE               5
CARABANCHEL              4
MONCLOA-ARAVACA          2
Name: DISTRICT, dtype: int64
```

Check if there is any missing value in the DataFrame:

```
# Check if there is any missing value
df_venues.isna().sum()
```

```
DISTRICT           0
LATITUDE           0
LONGITUDE          0
VENUE              0
VENUE LATITUDE     0
VENUE LONGITUDE    0
VENUE CATEGORY     0
dtype: int64
```

DataFrame is exported to a CSV file (*Venues.csv*).

Create a new DataFrame (*df_venues_summary*) grouping venues by district and adding new rows for districts with 0 venues:

```
# Create a new df with the no of venues grouped by district
df_venues_summary = df_venues.groupby('DISTRICT').count().reset_index()
df_venues_summary.rename(columns={'LATITUDE':'NO VENUES'}, inplace=True)
df_venues_summary = df_venues_summary.drop(df_venues_summary.columns[[2, 3, 4, 5, 6]], axis=1)
df_venues_summary.sort_values(by=['NO VENUES'], ascending=False, inplace=True)
df_venues_summary.reset_index(inplace=True, drop=True)
df_venues_summary
```

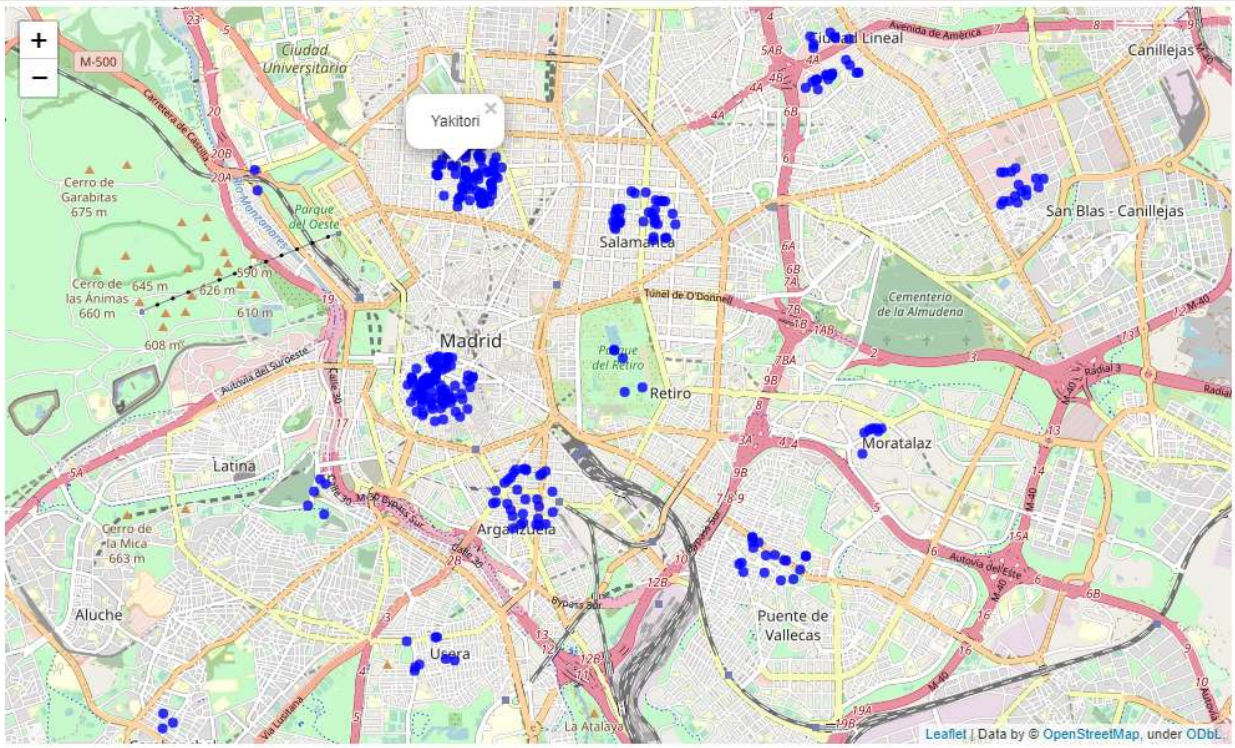
```
# Add new rows for districts with 0 venues
new_row1 = {'DISTRICT':'FUENCARRAL-EL PARDO', 'NO VENUES':0}
new_row2 = {'DISTRICT':'VICALVARO', 'NO VENUES':0}
df_venues_summary = df_venues_summary.append(new_row1, ignore_index=True)
df_venues_summary = df_venues_summary.append(new_row2, ignore_index=True)
```

	DISTRICT	NO VENUES
0	CENTRO	100
1	CHAMBERI	85
2	SALAMANCA	34
3	ARGANZUELA	31
4	CIUDAD LINEAL	24
5	TETUAN	24
6	BARAJAS	20
7	PUENTE DE VALLECAS	18
8	SAN BLAS-CANILLEJAS	16
9	HORTALEZA	9
10	CHAMARTIN	8
11	USERA	8
12	MORATALAZ	7
13	VILLA DE VALLECAS	5
14	VILLAVERDE	5
15	RETIRO	5
16	LATINA	5
17	CARABANCHEL	4
18	MONCLOA-ARAVACA	2
19	FUENCARRAL-EL PARDO	0
20	VICALVARO	0

DataFrame is exported to a CSV file (*Venues_Summary.csv*).

Map 3: (Circle map) Madrid venues.

```
map_madrid3 = folium.Map(location=[latitude, longitude], zoom_start=12)

# Add markers to map
for lat, lng, venue in zip(df_venues['VENUE LATITUDE'], df_venues['VENUE LONGITUDE'], df_venues['VENUE']):
    label = '{}'.format(venue)
    label = folium.Popup(label, parse_html=True)
    folium.Circle(
        location=[lat, lng],
        radius='60',
        color='slaterblue',
        fill=True,
        fill_color='blue',
        fill_opacity=0.8,
        popup=label,
    ).add_to(map_madrid3)

# Display map
map_madrid3
```



4.5 Correlation between number of bicycle accidents and number of venues per district

Create a new DataFrame (*df_influence*) with the number of bicycle accidents and the number of venues per district as a result of merging *df_accidents_summary* and *df_venues*:

```
# Merge both tables
df_influence = pd.merge(df_accidents_summary, df_venues_summary, on='DISTRICT')
```

```
df_influence
```

	DISTRICT	NO ACCIDENTS	NO VENUES
0	CENTRO	142	100
1	ARGANZUELA	72	31
2	SALAMANCA	72	34
3	CHAMBERI	70	85
4	RETIRO	59	5
5	FUENCARRAL-EL PARDO	49	0
6	TETUAN	45	24
7	MONCLOA-ARAVACA	45	2
8	CARABANCHEL	43	4
9	PUENTE DE VALLECAS	36	18
10	CHAMARTIN	32	8
11	LATINA	31	5
12	CIUDAD LINEAL	28	24
13	USERA	28	8
14	HORTALEZA	27	9
15	MORATALAZ	26	7
16	SAN BLAS-CANILLEJAS	20	16
17	VILLA DE VALLECAS	19	5
18	VILLAVERDE	18	5
19	BARAJAS	13	20
20	VICALVARO	9	0

Basic statistical information of the DataFrame:

```
df_influence.describe()
```

	NO ACCIDENTS	NO VENUES
count	21.000000	21.000000
mean	42.095238	19.523810
std	29.656205	26.312771
min	9.000000	0.000000
25%	26.000000	5.000000
50%	32.000000	8.000000
75%	49.000000	24.000000
max	142.000000	100.000000

Correlation between the number of accidents and the number of venues:

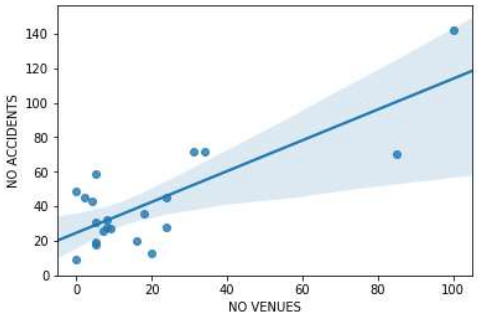
```
df_influence[['NO ACCIDENTS', 'NO VENUES']].corr()
```

	NO ACCIDENTS	NO VENUES
NO ACCIDENTS	1.000000	0.791514
NO VENUES	0.791514	1.000000

Chart 2: (Scatterplot) Regression between bicycle accidents and number of venues by district.

```
sns.regplot(x='NO VENUES', y='NO ACCIDENTS', data=df_influence)
plt.ylim(0,)
```

```
(0, 156.10100760542724)
```



Pearson Correlation Coefficient and P-Vale:

```
pearson_coef, p_value = stats.pearsonr(df_influence['NO VENUES'], df_influence['NO ACCIDENTS'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.7915140331955097  with a P-value of P = 1.9212793826658736e-05
```

There is a quite strong positive linear relationship between the number of bicycle accidents in Madrid in 2019 and the number of venues per district (as the number of venues increase, so it does the number of bicycle accidents).

Since P-value is lower than 0.0001, the correlation between the number of bicycle accidents in Madrid in 2019 and the number of venues per district is statistically significant.

4.6 Model development and evaluation

Model development

Create a linear regression object:

```
# Create the linear regression object
lm = LinearRegression()
lm
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

Predict the number of bicycle accidents based on the number of venues:

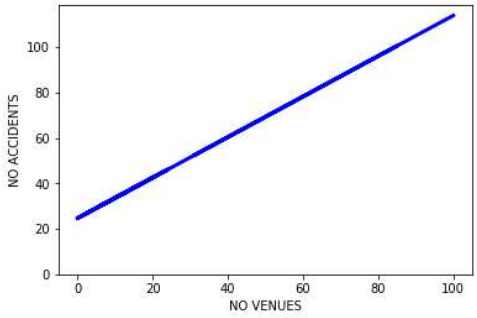
```
x = df_influence[['NO VENUES']] # Predictor variable
y = df_influence['NO ACCIDENTS'] # Response variable

lm.fit(x,y) # Fit the linear model

Yhat=lm.predict(x) # Output a prediction

print('Intercept (a): ', lm.intercept_)
print('Slope (b): ', lm.coef_)

print('No accidents = ', lm.intercept_, ' + ', lm.coef_, ' x No venues')
```

```
Intercept (a):  24.67828550991774
Slope (b):  [0.89208782]
No accidents =  24.67828550991774  +  [0.89208782]  x No venues
```

Chart 3: (Plot) Linear function.

```
plt.plot(x, Yhat, color='blue', linewidth=3)
plt.xlabel('NO VENUES', fontsize=10)
plt.ylabel('NO ACCIDENTS', fontsize=10)
plt.ylim(0,)
```

```
(0, 118.3475061212138)
```

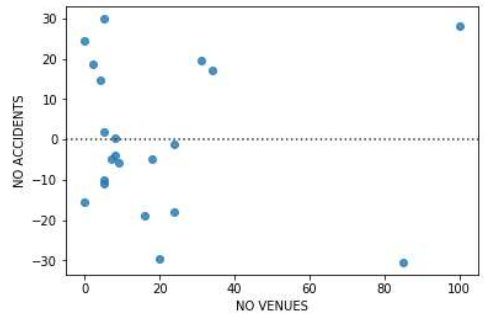


Model evaluation

Chart 4: (Scatterplot) Residual plot.

```
sns.residplot(x='NO VENUES', y='NO ACCIDENTS', data=df_influence)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb921055320>
```



Although most of the residuals are concentrated in the left half of the x-axis, in general residuals are randomly spread out around the x-axis, which leads us to believe that a linear model is appropriate considering the size of the data.

R-squared:

```
print('The R-square is: ', lm.score(x, y))
```

```
The R-square is:  0.6264944647454227
```

5. Statistical analysis

Pearson Correlation Coefficient

The Pearson product-momentum correlation coefficient is a measure of the strength of a linear association between two variables and attempts to draw a line of the best fit through the data of two variables.

Strength: Pearson correlation coefficient can range in value from -1 to 1, where an absolute value of 1 indicates a perfect linear relationship, and a value of 0 indicates no linear relationship between the variables.

Direction: The sign indicates the direction of the relationship, where a positive coefficient indicates that one variable tends to increase as the other increases, and a negative coefficient indicates that one variable tends to decrease as the other increases.

Pearson Correlation Coefficient between the number of bicycle accidents in Madrid in 2019 by district and the number of venues per district shows a value of **0.7915**, this meaning that there is a **quite strong positive relationship** between the variables (as the number of venues increase, so it does the number of bicycle accidents).

P-value

The P-value is the probability value that the correlation between two variables is statistically significant.

By convention, when the P-value is:

<< 0.001: there is strong evidence that the correlation is significant.
<< 0.05: there is moderate evidence that the correlation is significant.
<< 0.1: there is weak evidence that the correlation is significant.
>> 0.1: there is no evidence that the correlation is significant.

Pearson Correlation Coefficient between the number of bicycle accidents in Madrid in 2019 by district and the number of venues per district shows a P-value of **0.00001921**, this meaning that there is strong evidence that the correlation between the variables is **statistically significant**.

Simple Linear Regression

Simple Linear Regression is a statistical method that allows to analyze the relationship between two continuous variables, predicting the response or dependent variable as a function of the predictor or independent variable, by means of a Linear Function:

$$Yhat = a + b * X$$

'a' refers to the intercept of the regression line 0 (value of 'y' when 'x' is 0).

'b' refers to the slope of the regression line (value with which 'y' changes when 'x' increases in 1 unit.

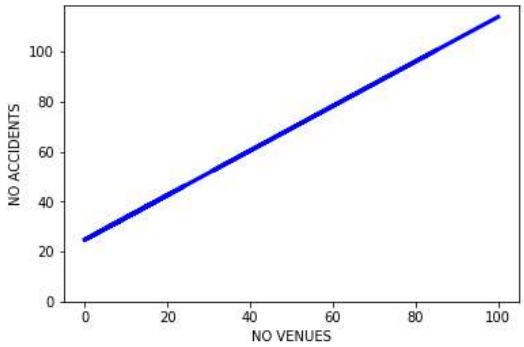
Simple Linear relationship between the number of bicycle accidents (Yhat - independent variable) and the number of venues per district (X - dependent variable) can be represented as:

No. accidents = 24.6783 + 0.8921 * No. venues

This meaning that:

- 24.6783 bicycle accidents may occur when there are no venues in a district.
- An increase in 1 venue in a district may give rise to new 0.8921 bicycle accidents.

The Simple Linear function can be represented as follows:

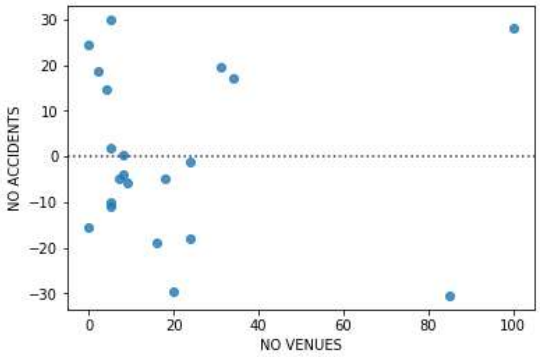


Model evaluation

- Residual plot

Residuals show the difference between the observed value and the predicted value. When looking at a regression plot, the residual is the distance from the data point to the fitted regression line.

A residual plot shows the residuals on the vertical y-axis and the independent variable on the horizontal x-axis. Randomly spread out residuals mean that the variance is constant, and therefore the linear model is a good fit or appropriate for the data.



The Residual plot shows that, although most of the residuals are concentrated in the left half of the x-axis, in general residuals are randomly spread out around the x-axis, which lead us to believe that **a linear model is appropriate**, also considering the size of the data.

- R-squared

R-squared or coefficient of determination is a measure to indicate how close the data is to the fitted regression line, by means of the percentage of variation of the response variable that is explained by a linear model.

R-squared value for the linear model predicted is 0.6265, this meaning that the **62.65% of the variation** of the number of bicycle accidents in Madrid in 2019 per district **is explained** by the number of venues per district.

6. Conclusions

1. There is a **quite strong positive correlation** (Pearson Correlation Coefficient: 0.7915) between the number of bicycle accidents in Madrid in 2019 by district and the number of food venues per district. This is, as the number of food venues increase, so it does the number of bicycle accidents.

2. There is strong evidence that the correlation between the variables is **statistically significant** (P-value < 0.001).

3. Simple Linear relationship between the number of bicycle accidents and the number of venues per district can be represented as:

   No. accidents = 24.6783 + 0.8921 * No. venues

4. An **increase in 1 food venue** in a district may predict **new 0.8921 bicycle accidents**.

5. A **linear model may be appropriate** considering the size of the data.

6. **62.65% of the variation** (R-squared) of the number of bicycle accidents in Madrid in 2019 by district **is explained** by the number of food venues per district.

Limitations to the conclusions

1. Although there is a quite strong positive correlation between the variables analyzed, it should be noted that correlation does not imply causation. Therefore, the number of bicycle riders, measured as the number of food venues per district, may have influence in the number of bicycle accidents, but does not cause such accidents *per se*.

2. Being a 62.65% of the variation explained by the number of food venues per district shall be interpreted as other variables having a significant relationship with the number of bicycle accidents by district, e.g. work commutes, sport, other.

3. Taking into consideration the size of the data and the assumptions introduced (e.g. influence of riders measured as the number of food venues per district, 1 year-period information), the analysis carried out could be non-conclusive for the purposes of measuring the influence between the variables subject of study.

4. Foursquare API (required for the project) has certain limitations, such as a 100 venues limit per request, which has forced to introduce the assumption that all districts have a 400 meters radius, not gathering this way the whole number of food venues for each district.

5. Finally, it has been considered that all venues included within Foursquare's 'food' category have a food delivery service which is attended by bicycle riders.

Annex I: Python libraries

```
# IBM cloud
import types
from botocore.client import Config
import ibm_boto3

import pandas as pd # Data analysis
import numpy as np # Scientific computing

# Visualization
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors

# Map visualization
!pip install folium
import folium
from folium import plugins
from folium.plugins import MarkerCluster

from geopy.geocoders import Nominatim # Geocoding

import seaborn as sns # Statistical data visualization

from scipy import stats # Statistics

# Machine learning
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

IBM cloud: create and import CSV files and assets within Jupyter Notebooks.

Pandas: BSD-licensed library providing data structures and data analysis tools (https://pandas.pydata.org/).

Numpy: BSD-licensed library for scientific computing (https://numpy.org/).

Matplotlib: PSF-licensed library for creating visualizations (https://matplotlib.org/).

Folium: MIT-licensed library for creating visualizations in a Leaflet map (https://pypi.org/project/folium/).

Geopy: MIT-licensed library for geocoding web services (https://geopy.readthedocs.io/).

Seaborn: BSD-licensed data visualization library based on Matplotlib (https://seaborn.pydata.org/).

Scipy: BSD-licensed scientific and numerical tools library (https://scipy.org/).

Scikit learn: BSD-licensed machine learning library (https://scikit-learn.org/)

Annex II: Description of DataFrames and CSV files

df_accidents: {date, district} Date and district name for each accident where a bicycle was involved in Madrid in 2019 (shape: 884, 2).

df_accidents_summary: {district, no accidents} Number of accidents where a bicycle was involved in Madrid in 2019 by district (shape: 21, 2).

df_madrid_geo: {district, latitude, longitude} Coordinates for each district in Madrid (shape: 21, 3).

df_accidents_geo: {district, no accidents, latitude, longitude} Number of accidents where a bicycle was involved in Madrid in 2019 and coordinates by district (shape: 21, 4).

df_venues: {district, latitude, longitude, venue, venue latitude, venue longitude, venue category} Coordinates, category, district and district's coordinates by food venue (shape: 410, 7).

df_venues_summary: {district, no venues} Number of venues grouped by district (shape: 21, 2).

df_influence: {district, no accidents, no venues} Number of accidents where a bicycle was involved in Madrid in 2019 and number of food venues by district (shape (21, 3).

Accidents2019.csv: information regarding traffic accidents in Madrid in 2019 available in the Madrid's Council open data website.

Bicycle_Accidents.csv: [df_accidents_summary]

Madrid_Geo.csv: [df_madrid_geo]

Venues.csv: [df_venues]

Venues_Summary.csv [df_venues_summary]

Annex III: Charts and maps

Chart 1: (Bar chart) Number of bicycle accidents in Madrid in 2019 by district.

Chart 2: (Scatterplot) Regression between bicycle accidents and number of venues by district.

Chart 3: (Plot) Linear function.

Chart 4: (Scatterplot) Residual plot.

Map 1: (Marker map) Madrid districts.

Map 2: (Bubble heat map) Bicycle accidents in Madrid in 2019 by district.

Map 3: (Circle map) Madrid venues.