ICT for smart mobility

Report of Lab 3

Group 10
Setareh Pourgholamali S309064
Hossein Zahedi Nezhad S309247
SeyedMohammadhossein Sajjadikaboudi S313242

January 2024

Introduction:

The aim of our third lab is to further examine the data from the first two labs to pinpoint likely user categories in the car sharing system. We did this by comparing Origin-Destination (OD) matrices from the Rental database with OD matrices reached from user interviews data (IMQ dataset). An OD matrix maps movement in an area, crucial for understanding transportation needs. In these matrices, rows are trip starting points and columns are destinations. Each matrix cell represents a trip's start and end point. This gives insight into the demand for various transport links. Our analysis of OD matrices from both sources will help us identify user patterns and trends, enhancing our understanding of user preferences, travel habits, and the specific requirements of different user groups in the car sharing system.

Data Extraction from Rental Dataset

To derive the Origin-Destination (OD) matrices from the Rental dataset, we employed the ictts-PermanentBookings dataset housed in MongoDB which is for Car2go Company. This dataset includes details on the starting point, endpoint, and starting time of car rentals. We established zone boundaries using GeoJSON data customized for Turin to construct the OD matrices. By applying filters and extracting the corresponding data, we derived four separate OD matrices, each offers crucial information on the patterns of spatial distribution and movement dynamics of car-sharing users, categorized according to particular time intervals, depicted in Table 1.

Name	Days	Time
Week Days - Morning	Monday - Friday	6 - 12
Week Days - Afternoon	Monday - Friday	13 - 23
Weekends - Morning	Saturday and Sunday	6 - 12
Weekends - Afternoon	Saturday and Sunday	13 - 23

Table 1. Filtered time periods

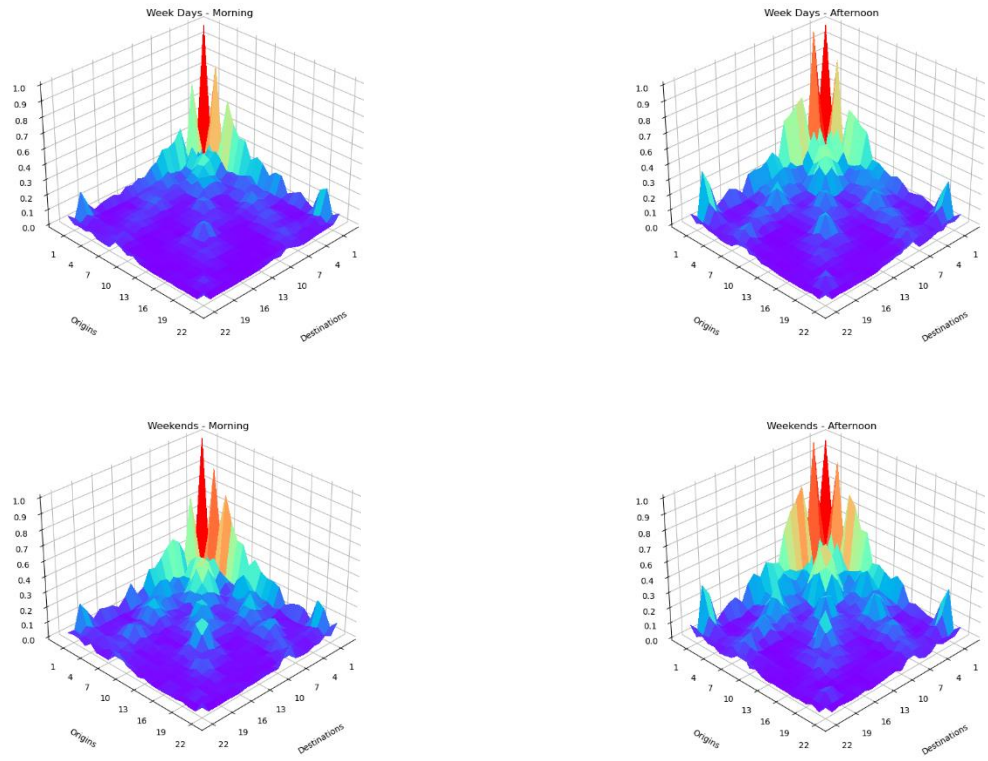


Figure 1. Extracted OD matrices from Dataset for different time periods

These Three Dimensional OD matrices help us to observe any variations in the travel patterns and destinations between weekends and weekdays. The depicted plots indicate a concentration of trips within the central urban zones in all days of the week. Additionally, following the central district, significant trip volumes are noted in the areas of S. Salvario, Crocetta, San Paolo, and Cenisia, where the Polytechnic University is situated.

Moreover, the image presented beneath illustrates the three-dimensional matrices constructed from the comprehensive dataset for Car2go and Enjoy. From these visualizations, it becomes evident that the majority of travel occurs between the city's primary zones which are district 1 to 5. Conversely, in the areas farther from the city center such as Mirafiori SUD and Cavoretto, there is a notable decrease in the number of trips. Also, this matrix is depicted without normalization to show the total number of trips made with each service, which is significantly higher than Enjoy for Car2go.

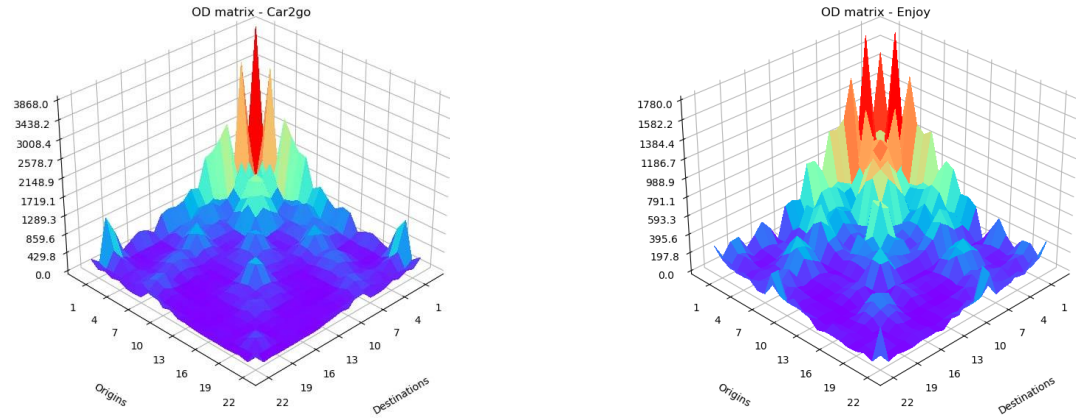


Figure 2. Extracted OD matrices from Dataset for Car2go and Enjoy

Data Extraction from IMQ

The IMQ dataset, gathered via telephone surveys, encompasses essential details from 105,099 journeys that took place between Monday and Friday. Encompassing 185 zones throughout the Piedmont area and highlighting 23 specific zones in Turin makes this dataset comprehensive. These interviews were designed to gather a variety of demographic details, such as gender, age brackets, and the reasons for travel.

To derive the Origin-Destination (OD) matrices from the IMQ dataset, we utilized the pivot table feature in Excel. This technique simplified the extraction process, providing an efficient means to generate the OD matrices categorized by zones of departure and arrival. Initially, a comprehensive matrix detailing the origins and destinations of the interviewees' journeys was created without the application of any filters, as illustrated in Figure 3.

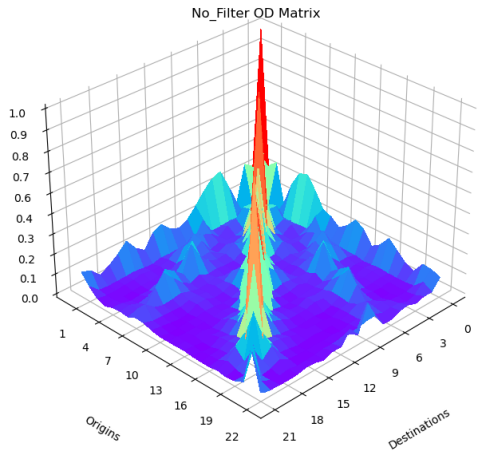
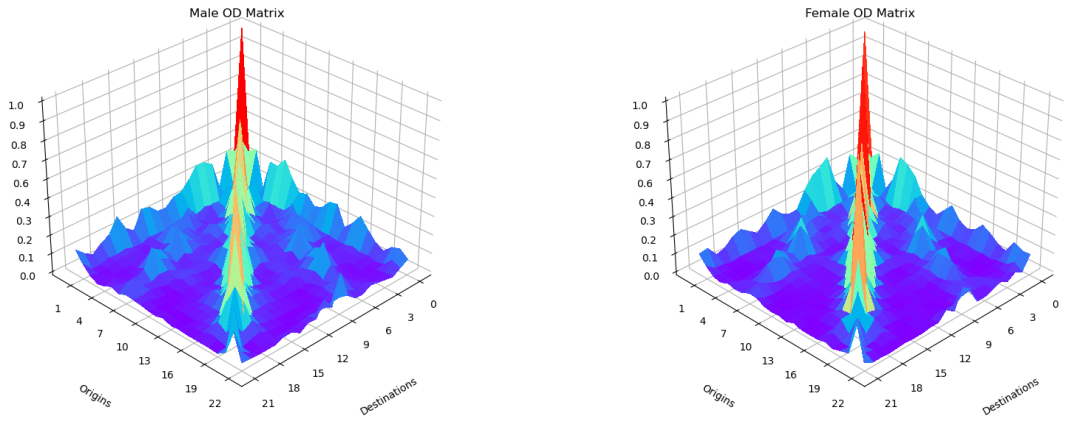


Figure 3. Extracted OD matrix from IMQ dataset without using Filters

The IMQ - OD matrix reveals that a majority of the trips are concentrated along the matrix's diagonal, indicating that, in most instances, the trip's origin and destination were the same area. Consequently, trips extending to other regions were fewer in comparison. Now, various filters were implemented on the data for further analysis.

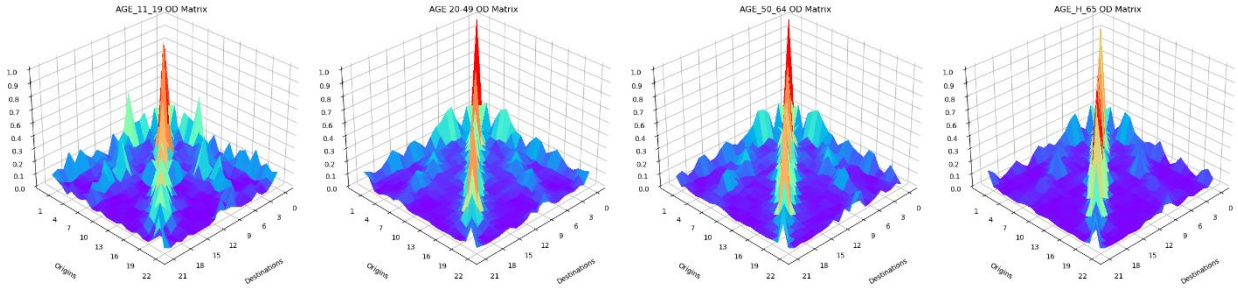
1) Filters on Gender



(a) Male (b) Female

Figure 4. Extracted OD matrix from IMQ dataset for Male and Female

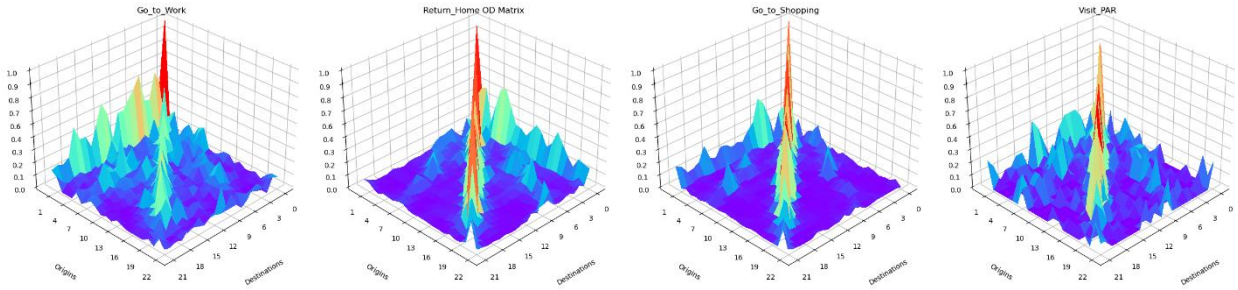
2) Filters on Age



(a) Age 11 – 19 (b) Age 20 – 49 (c) Age 50 – 64 (d) Age more than 65

Figure 5. Extracted OD matrix from IMQ dataset for Different Bracket Age

3) Filters on Motivation



(a) Go to Work (b) Returning Home (c) Go to Shopping (d) Visiting Parents or friends

Figure 6. Extracted OD matrix from IMQ dataset for various travel purpose

The 3D visualizations, generated from the IMQ dataset's OD matrices, illustrate the relationship between origin zones, destination zones, and the normalized number of trips. This normalization is achieved by dividing the number of trips by the maximum number observed within the dataset. The application of various filters has been strategically chosen to ensure that substantial data remains post-filtering, enabling accurate matrix plotting. This approach addresses challenges posed by null data and the prevalence of very small numbers within the matrix, which could otherwise hinder the visualization process.

The diagrams prominently feature a higher number of trips along the matrix diagonal, where the trip's origin and destination are the same zone. This pattern underscores a tendency for shorter, localized trips within the same area. Notably, the OD matrix filtered for 'Go to work' purposes exhibits a more evenly spread distribution of trips across the entire matrix, not just the diagonal. This distribution suggests that work-related travel often involves longer distances.

Conversely, the OD matrix for individuals aged 65 and over shows that trips predominantly occur within the same zone, indicating limited mobility or a preference for staying local. In contrast, the data for the 11 to 19 age group reveals a more extensive spread of trips across different zones. This wider distribution implies that younger individuals likely partake in a variety of activities or have diverse transportation needs, leading to more varied trip destinations.

Comparison and Similarity

The main objective of this Lab is to extract and compare different OD matrices gained from the two mentioned dataset (Car sharing and IMQ). To accomplish this, we have undertaken a sequence of steps.

1) In the first step, we normalized the OD matrices to facilitate meaningful comparisons between distances. This normalization process guarantees that the sum of all elements equals 1 ($\sum_{ij} a_{ij} = 1$), achieved by dividing each element by the total sum ($\sum_{ij} a_{ij}$).

2) in order to determine whether two matrices are similar, it was essential to first establish a metric for similarity. To achieve this, we calculate the Euclidean distance between two matrices derived from the IMQ dataset for Male and Female.

$$d_2(A,B) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij} - b_{ij})^2}$$

If you consider the elements of male matrix as a_{ij}, and the female matrix as b_{ij}, calculating the Euclidean distance between each corresponding element of these two matrices and then summing all these individual distances gives you a cumulative difference of 0.9864.

3) After normalizing the data and then calculating the Euclidean distance between the various OD matrices derived from the filtered IMQ data, as well as those obtained for different time periods from the Car2go dataset, we generated a heat map that illustrates the similarity between matrices. The smaller the difference and the numerical value, the greater the similarity between the matrices. Figure 7 reveals a consistent pattern regarding the potential users of each service, albeit with some exceptions.

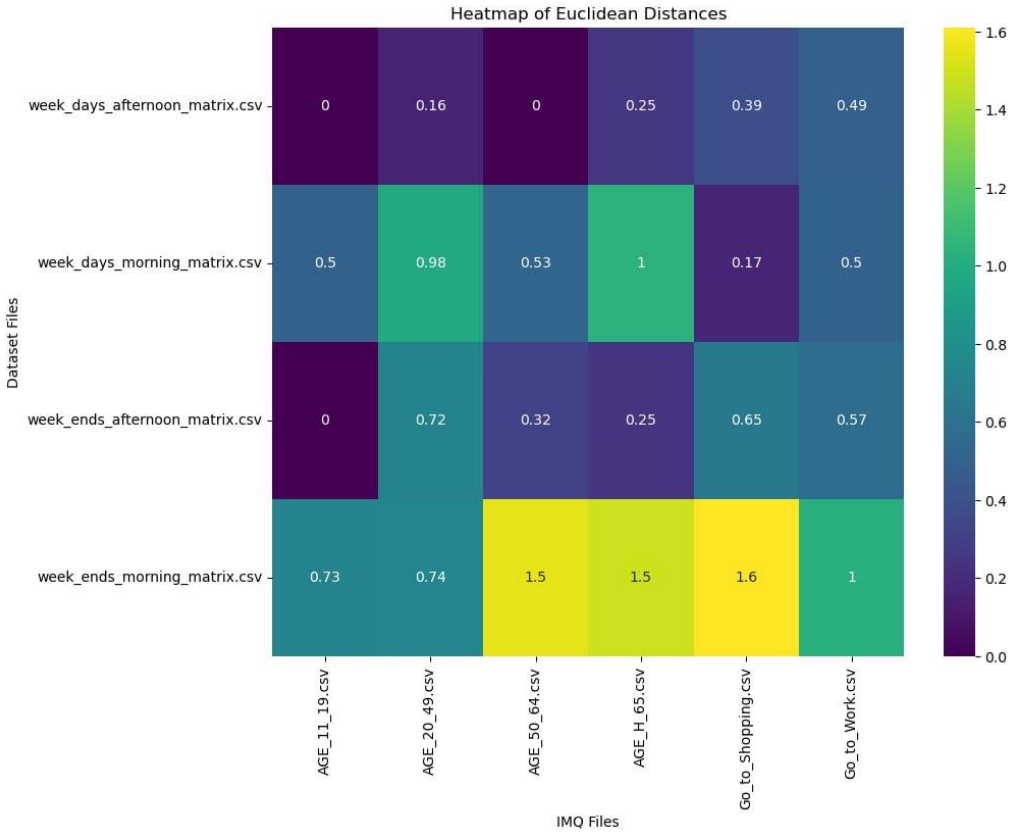


Figure 7. Calculated Euclidean distances for different filters and time periods (Car2go)

During the weekend, all demographic groups exhibit reduced distances in their activity patterns during the afternoon hours, indicating a higher utilization of the service from 13:00 to 23:00. Observations also reveal narrower distances for commutes labeled "go to work," suggesting an increased likelihood of users opting for car-sharing services for their work-related travel needs.

In most instances, the variations derived from the Euclidean distances among various matrices are less than the metric matrix's difference, noted at 0.9864. Exceptions are seen in the age brackets of 50 to 64 and those older than 65, suggesting a decrease in travel and a preference for resting during holiday mornings. This pattern is similarly observed for shopping activities on holiday mornings. Additionally, the limited participation of individuals aged 11 to 19 in the survey probably due to Less likely to have a driver's license led to insufficient data for this age group, contributing to a high incidence of null entries in the corresponding matrix. This lack of data, in turn, has resulted in minor anomalies in the calculation of Euclidean distances for this demographic segment.

4) In the last step, the final comparison has been done. For this, using the mentioned methods, the Euclidean distance between the imq matrix and the car2go, enjoy matrices has been compared, and finally, the summation of all the distances obtained in the resulting matrix with the value of the matrix The criterion has been measured. These values are shown in Table 2 and shown in Figure 8 for better understanding. Also, considering that the IMQ matrix is a 23 x 23 matrix, in order to perform the calculations, the obtained matrices for Kartogo and Enjoy had to be reshipped from the dataset.

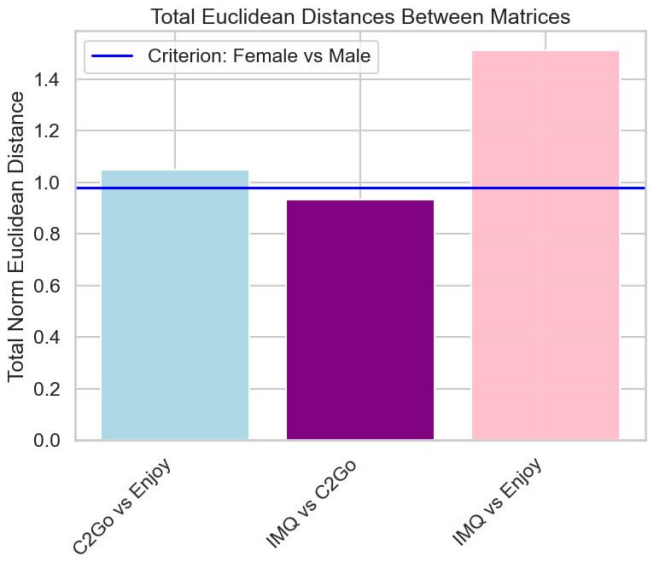


Figure 8. Total Euclidean Distances between Different matrices

Matrices	Caar2go	Enjoy	IMQ
Caar2go	-	1.0468	0.9317
Enjoy	1.0468	-	1.5101
IMQ	0.9317	1.5101	-

Table2. Exact number of total Euclidean Distances between Different matrices

Figure 8 demonstrates that the distance between the Car2go matrix and the IMQ matrix is smaller than the Euclidean distance observed between the IMQ and Enjoy matrices. Additionally, the Euclidean distance between Car2go and IMQ is lower compared to the distance for the defined metric matrix, whereas the scenario is reversed for the distance between Enjoy and IMQ. This pattern suggests a higher volume of trips, and consequently more data and smaller differences, for the Car2go system.

APPENDIX

1. Connecting Database

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import pymongo as pm
import pprint
from enum import Enum
from datetime import datetime, timedelta
import geojson
import seaborn as sb
#Connect to the DB
client = pm.MongoClient('bigdatadb.polito.it',
ssl=True,
authSource = 'carsharing',
username = 'ictts',
password ='Ict4SM22!',
tlsAllowInvalidCertificates=True)
db = client['carsharing']
#Choose the DB to use
Ictts_enj_p_booking = db['ictts_enjoy_PermanentBookings']
Ictts_c2g_p_booking = db['ictts_PermanentBookings'] #PermanentBookings
with open("C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\TorinoZonescol.geojson") as f:
    gj = geojson.load(f)
```

2. Dataset Extraction & Plots

Extracting Enjoy OD matrix

```
# Initialize a 23x23 matrix with zeros
od_matrix_enjoy = np.zeros((23, 23))

# Loop through each origin and destination
for orig in range(23):
    for dest in range(23):
        orig_zone = gj['features'][orig]["geometry"]["coordinates"]
        dest_zone = gj['features'][dest]["geometry"]["coordinates"]

        pipeline = [
            {"$project": {
                "init_loc": 1, "final_loc": 1, "init_time": 1
            }},
            {"$match": {
                "init_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": orig_zone}}},
                "final_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": dest_zone}}}
            }},
            {"$count": "tot"}
        ]
        res = list(Ictts_enj_p_booking.aggregate(pipeline))

        if len(res) > 0:
            od_matrix_enjoy[orig][dest] = res[0]['tot']

# Convert the matrix to a DataFrame
od_matrix_enjoy = pd.DataFrame(od_matrix_c2g, columns=[f"dest_{i}" for i in range(23)],
                               index=[f"orig_{i}" for i in range(23)])

output_directory = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset"
output_filename = "od_matrix_enjoy.csv"
output_path = os.path.join(output_directory, output_filename)
```

Extracting Car2go OD matrix

```
# Initialize a 23x23 matrix with zeros
od_matrix_c2go = np.zeros((23, 23))

# Loop through each origin and destination
for orig in range(23):
    for dest in range(23):
        orig_zone = gj['features'][orig]["geometry"]["coordinates"]
        dest_zone = gj['features'][dest]["geometry"]["coordinates"]

        pipeline = [
            {"$project": {
                "init_loc": 1, "final_loc": 1, "init_time": 1
            }},
            {"$match": {
                "init_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": orig_zone}}},
                "final_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": dest_zone}}}
            }},
            {"$count": "tot"}
        ]
        res = list(Ictts_c2g_p_booking.aggregate(pipeline))

        if len(res) > 0:
            od_matrix_c2go[orig][dest] = res[0]['tot']

# Convert the matrix to a DataFrame
od_matrix_c2go = pd.DataFrame(od_matrix_c2g, columns=[f"dest_{i}" for i in range(23)],
                              index=[f"orig_{i}" for i in range(23)])

output_directory = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset"
output_filename = "od_matrix_c2go.csv"
output_path = os.path.join(output_directory, output_filename)
```

Weekdays - Morning

```
# Initialize a 23x23 matrix with zeros
od_matrix_week_days_morning = np.zeros((23, 23))

# Loop through each origin and destination
for orig in range(23):
    for dest in range(23):
        orig_zone = gj['features'][orig]["geometry"]["coordinates"]
        dest_zone = gj['features'][dest]["geometry"]["coordinates"]

        pipeline = [
            {"$project": {
                "hour": {"$hour": "$init_date"},
                "day": {"$dayOfWeek": "$init_date"},
                "init_loc": 1, "final_loc": 1, "init_time": 1
            }},
            {"$match": {
                "day": {"$gte": 2, "$lte": 6},
                "hour": {"$gte": 6, "$lte": 12},
                "init_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": orig_zone}}},
                "final_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": dest_zone}}}
            }},
            {"$count": "tot"}
        ]
        res = list(Ictts_c2g_p_booking.aggregate(pipeline))

        if len(res) > 0:
            od_matrix_week_days_morning[orig][dest] = res[0]['tot']

# Convert the matrix to a DataFrame
od_matrix_week_days_morning = pd.DataFrame(od_matrix_c2g, columns=[f"dest_{i}" for i in range(23)],
                                index=[f"orig_{i}" for i in range(23)])

output_directory = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset"
output_filename = "week_days_morning.csv"
output_path = os.path.join(output_directory, output_filename)
```

Weekdays - Afternoon

```
# Initialize a 23x23 matrix with zeros
od_matrix_week_days_afternoon = np.zeros((23, 23))

# Loop through each origin and destination
for orig in range(23):
    for dest in range(23):
        orig_zone = gj['features'][orig]["geometry"]["coordinates"]
        dest_zone = gj['features'][dest]["geometry"]["coordinates"]

        pipeline = [
            {"$project": {
                "hour": {"$hour": "$init_date"},
                "day": {"$dayOfWeek": "$init_date"},
                "init_loc": 1, "final_loc": 1, "init_time": 1
            }},
            {"$match": {
                "day": {"$gte": 2, "$lte": 6},
                "hour": {"$gte": 13, "$lte": 23},
                "init_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": orig_zone}}},
                "final_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": dest_zone}}}
            }},
            {"$count": "tot"}
        ]
        res = list(Ictts_c2g_p_booking.aggregate(pipeline))

        if len(res) > 0:
            od_matrix_week_days_afternoon[orig][dest] = res[0]['tot']

# Convert the matrix to a DataFrame
od_matrix_week_days_afternoon = pd.DataFrame(od_matrix_c2g, columns=[f"dest_{i}" for i in range(23)],
                                index=[f"orig_{i}" for i in range(23)])

output_directory = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset"
output_filename = "od_matrix_week_days_afternoon.csv"
output_path = os.path.join(output_directory, output_filename)
```

Weekends - Morning

```
# Initialize a 23x23 matrix with zeros
od_matrix_week_ends_morning = np.zeros((23, 23))

# Loop through each origin and destination
for orig in range(23):
    for dest in range(23):
        orig_zone = gj['features'][orig]["geometry"]["coordinates"]
        dest_zone = gj['features'][dest]["geometry"]["coordinates"]

        pipeline = [
            {"$project": {
                "hour": {"$hour": "$init_date"},
                "day": {"$dayOfWeek": "$init_date"},
                "init_loc": 1, "final_loc": 1, "init_time": 1
            }},
            {"$match": {
                "day": {"$in": [7, 1]},
                "hour": {"$gte": 6, "$lte": 12},
                "init_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": orig_zone}}},
                "final_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": dest_zone}}}
            }},
            {"$count": "tot"}
        ]
        res = list(Ictts_c2g_p_booking.aggregate(pipeline))

        if len(res) > 0:
            od_matrix_week_ends_morning[orig][dest] = res[0]['tot']

# Convert the matrix to a DataFrame
od_matrix_week_ends_morning = pd.DataFrame(od_matrix_c2g, columns=[f"dest_{i}" for i in range(23)],
                                index=[f"orig_{i}" for i in range(23)])

output_directory = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset"
output_filename = "week_ends_morning.csv"
output_path = os.path.join(output_directory, output_filename)
```

Weekends - Afternoon

```
# Initialize a 23x23 matrix with zeros
od_matrix_week_ends_afternoon = np.zeros((23, 23))

# Loop through each origin and destination
for orig in range(23):
    for dest in range(23):
        orig_zone = gj['features'][orig]["geometry"]["coordinates"]
        dest_zone = gj['features'][dest]["geometry"]["coordinates"]

        pipeline = [
            {"$project": {
                "hour": {"$hour": "$init_date"},
                "day": {"$dayOfWeek": "$init_date"},
                "init_loc": 1, "final_loc": 1, "init_time": 1
            }},
            {"$match": {
                "day": {"$in": [7, 1]},
                "hour": {"$gte": 13, "$lte": 23},
                "init_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": orig_zone}}},
                "final_loc": {"$geoWithin": {"$geometry": {"type": "MultiPolygon", "coordinates": dest_zone}}}
            }},
            {"$count": "tot"}
        ]
        res = list(Ictts_c2g_p_booking.aggregate(pipeline))

        if len(res) > 0:
            od_matrix_week_ends_afternoon[orig][dest] = res[0]['tot']

# Convert the matrix to a DataFrame
od_matrix_week_ends_afternoon = pd.DataFrame(od_matrix_c2g, columns=[f"dest_{i}" for i in range(23)],
                                index=[f"orig_{i}" for i in range(23)])

output_directory = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset"
output_filename = "week_ends_afternoon.csv"
output_path = os.path.join(output_directory, output_filename)
```

Plotting OD matrices

```
#3D OD Matrixes

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import LinearLocator
from tqdm import tqdm

# Directory paths
dataset_dir = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset"

# Reading CSV files from the Dataset directory
WD_M = pd.read_csv(dataset_dir + "\\week_days_morning.csv")
WD_A = pd.read_csv(dataset_dir + "\\week_days_afternoon.csv")
WE_M = pd.read_csv(dataset_dir + "\\week_ends_morning.csv")
WE_A = pd.read_csv(dataset_dir + "\\week_ends_afternoon.csv")

ALLc2 = pd.read_csv(dataset_dir + "\\od_matrix_c2go.csv")
ALLenjoy = pd.read_csv(dataset_dir + "\\od_matrix_enjoy.csv")

# Plotting 3D matrices
%matplotlib inline
##%matplotlib widget

def plot_3D_adjusted(count, title, step=3):
    fig = plt.figure(figsize=(8, 6))
    ax = plt.subplot(111, projection='3d')
    ax.xaxis.pane.fill = False
    ax.xaxis.pane.set_edgecolor('white')
    ax.yaxis.pane.fill = False
    ax.yaxis.pane.set_edgecolor('white')
    ax.zaxis.pane.fill = False
    ax.zaxis.pane.set_edgecolor('white')
    ax.grid()
```

```
# Plotting Week Days - Morning
WD_M_array = np.array(WD_M["count"])
WD_M_array = WD_M_array.reshape(23, -1)
WD_M_array_norm = WD_M_array / np.max(WD_M_array)
plot_3D_adjusted(WD_M_array_norm, "Week Days - Morning")

# Plotting Week Days - Afternoon
WD_A_array = np.array(WD_A["count"])
WD_A_array = WD_A_array.reshape(23, -1)
WD_A_array_norm = WD_A_array / np.max(WDA_array)
plot_3D_adjusted(WD_A_array_norm, "Week Days - Afternoon")

# Plotting Weekends - Morning
WE_M_array = np.array(WE_M["count"])
WE_M_array = WE_M_array.reshape(23, -1)
WE_M_array_norm = WE_M_array / np.max(WE_M_array)
plot_3D_adjusted(WE_M_array_norm, "Weekends - Morning")

# Plotting Weekends - Afternoon
WE_A_array = np.array(WE_A["count"])
WE_A_array = WE_A_array.reshape(23, -1)
WE_A_array_norm = WE_A_array / np.max(WE_A_array)
plot_3D_adjusted(WE_A_array_norm, "Weekends - Afternoon")

#All car2go
ALLc2_array = np.array(ALLc2["count"])
ALLc2_array = ALLc2_array.reshape(23, -1)
#ALLc2_array_norm = WEM_array / np.max(WEM_array)
plot_3D_adjusted(ALLc2_array, "OD matrix - Car2go")

# All Enjoy
ALLenjoy_array = np.array(ALLenjoy["count"])
ALLenjoy_array = ALLenjoy_array.reshape(23, -1)
# ALLenjoy_array_norm = ALLenjoy_array / np.max(ALLenjoy_array)
plot_3D_adjusted(ALLenjoy_array, "OD matrix - Enjoy")
```

3. IMQ Extraction & Plots

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import LinearLocator

# Directory paths
imq_dir = "C:\\Users\\ASUS\\Desktop\\LAB3 Smart mobility\\Directory\\IMQ"


###_________________All files have been created using the Pivot Table option in Excel_____________________

# Reading CSV files from the IMQ directory
Male = pd.read_csv(imq_dir + "\\Male.csv")
Female = pd.read_csv(imq_dir + "\\Female.csv")

IMQ = pd.read_csv(imq_dir + "\\No_Filter.csv")

age_B_1 = pd.read_csv(imq_dir + "\\AGE_11_19.csv")
age_B_1 = pd.read_csv(imq_dir + "\\AGE_20_49.csv")
age_B_2 = pd.read_csv(imq_dir + "\\AGE_50_64.csv")
age_B_3 = pd.read_csv(imq_dir + "\\AGE_H_65.csv")

work = pd.read_csv(imq_dir + "\\Go_to_Work.csv")
Home = pd.read_csv(imq_dir + "\\Return_Home.csv")
Vis-PAR/AMC = pd.read_csv(imq_dir + "\\Visit_PAR_AMC")
Leisure = pd.read_csv(imq_dir + "\\Go_to_Leisure.csv")
shopping = pd.read_csv(imq_dir + "\\Go_to_Shopping.csv")
```

```
#Age 11 to 19
age_11_19_df = pd.read_csv(img_dir + "\\AGE_11_19.csv")

# Select the data excluding 'Row Labels' and 'Grand Total' columns
age_11_19_matrix = age_11_19_df.iloc[:, 1:-1].to_numpy()

# Handle NaN values by replacing them with zeros
age_11_19_matrix_filled = np.nan_to_num(age_11_19_matrix)

# Normalize the matrix
age_11_19_matrix_norm = age_11_19_matrix_filled / np.max(age_11_19_matrix_filled)

# Plotting the OD matrix
plot_3D_adjusted(age_11_19_matrix_norm, "AGE 11-19 OD Matrix", step=3)

#Age 20 to 49
age_20_49_df = pd.read_csv(img_dir + "\\AGE_20_49.csv")
age_20_49_matrix = age_20_49_df.iloc[:, 1:-1].to_numpy()
age_20_49_matrix_filled = np.nan_to_num(age_20_49_matrix)
age_20_49_matrix_norm = age_20_49_matrix_filled / np.max(age_20_49_matrix_filled)
plot_3D_adjusted(age_20_49_matrix_norm, "AGE 20-49 OD Matrix", step=3)

#Age 50 to 64
age_50_64_df = pd.read_csv(img_dir + "\\AGE_50_64.csv")
age_50_64_matrix = age_50_64_df.iloc[:, 1:-1].to_numpy()
age_50_64_matrix_filled = np.nan_to_num(age_50_64_matrix)
age_50_64_matrix_norm = age_50_64_matrix_filled / np.max(age_50_64_matrix_filled)
plot_3D_adjusted(age_50_64_matrix_norm, "AGE 50-64 OD Matrix", step=3)
```

```
#Age more than 65
age_H_65_df = pd.read_csv(img_dir + "\\AGE_H_65.csv")
age_H_65_matrix = age_H_65_df.iloc[:, 1:-1].to_numpy()
age_H_65_matrix_filled = np.nan_to_num(age_H_65_matrix)
age_H_65_matrix_norm = age_H_65_matrix_filled / np.max(age_H_65_matrix_filled)
plot_3D_adjusted(age_H_65_matrix_norm, "AGE More than 65 OD Matrix", step=3)

#Male
Male_df = pd.read_csv(img_dir + "\\Male.csv")
Male_matrix = Male_df.iloc[:, 1:-1].to_numpy()
Male_matrix_filled = np.nan_to_num(Male_matrix)
Male_matrix_norm = Male_matrix_filled / np.max(Male_matrix_filled)
plot_3D_adjusted(Male_matrix_norm, "Male.csv", step=3)

#FeMale
Female_df = pd.read_csv(img_dir + "\\Female.csv")
Female_matrix = Female_df.iloc[:, 1:-1].to_numpy()
Female_matrix_filled = np.nan_to_num(Female_matrix)
Female_matrix_norm = Female_matrix_filled / np.max(Female_matrix_filled)
plot_3D_adjusted(Female_matrix_norm, "Female.csv", step=3)

#Go to Work
work_df = pd.read_csv(img_dir + "\\Go_to_Work.csv")
work_matrix = Work_df.iloc[:, 1:-1].to_numpy()
work_matrix_filled = np.nan_to_num(work_matrix)
work_matrix_norm = Work_matrix_filled / np.max(work_matrix_filled)
plot_3D_adjusted(Work_matrix_norm, "Go_to_Work.csv", step=3)
```

```
#Return Home
Home_df = pd.read_csv(img_dir + "\\Return_Home.csv")
Home_matrix = Home_df.iloc[:, 1:-1].to_numpy()
Home_matrix_filled = np.nan_to_num(Home_matrix)
Home_matrix_norm = Home_matrix_filled / np.max(Home_matrix_filled)
plot_3D_adjusted(Home_matrix_norm, "Return_Home.csv", step=3)

#Visit Parents/Amici
PAR_df = pd.read_csv(img_dir + "\\Visit_PAR.csv")
PAR_matrix = PAR_df.iloc[:, 1:-1].to_numpy()
PAR_matrix_filled = np.nan_to_num(PAR_matrix)
PAR_matrix_norm = PAR_matrix_filled / np.max(PAR_matrix_filled)
plot_3D_adjusted(PAR_matrix_norm, "Visit_PAR.csv", step=3)

#Go for Shopping
Shop_df = pd.read_csv(img_dir + "\\Go_to_Shopping.csv")
Shop_matrix = Shop_df.iloc[:, 1:-1].to_numpy()
Shop_matrix_filled = np.nan_to_num(Shop_matrix)
Shop_matrix_norm = Shop_matrix_filled / np.max(Shop_matrix_filled)
plot_3D_adjusted(Shop_matrix_norm, "Go_to_Shopping.csv", step=3)
```

4. Comparison and Similarity

Euclidean Distance between Male and Female

```
#Euclidean Distane between Male and Female Matrix

# Convert all columns in Female and Male to numeric, setting non-numeric values to NaN
Female_numeric = Female.apply(pd.to_numeric, errors='coerce').fillna(0)
Male_numeric = Male.apply(pd.to_numeric, errors='coerce').fillna(0)

# Normalize the Female and Male
def normalize_to_sum_one(df):
    total_sum = df.sum().sum()  # total sum of all elements
    return df / total_sum

Female_normalized = normalize_to_sum_one(Female_numeric)
Male_normalized = normalize_to_sum_one(Male_numeric)

# Define the distance calculation function
def calculate_distance(df1, df2):
    """
    Calculate the Euclidean distance between two DataFrames element-wise.
    Assumes df1 and df2 have the same shape.
    """
    if df1.shape != df2.shape:
        raise ValueError("DataFrames do not have the same shape")

    # Element-wise Euclidean distance
    distance = np.sqrt((df1 - df2)**2)
    return distance

# Check if Female and Male have the same shape and calculate the distance
if Female_normalized.shape == Male_normalized.shape:
    try:
        distance_matrix = calculate_distance(Female_normalized, Male_normalized)
        total_distance = (distance_matrix.sum().sum())
        print("Total Distance Female vs Male:", total_distance)
```

Heat map of Euclidean Distances of IMQ matrices and Dataset Matrices

```
# read and normalize
def read_and_normalize(file_path):
    df = pd.read_csv(file_path)
    df_numeric = df.apply(pd.to_numeric, errors='coerce').fillna(0)
    total_sum =  df_numeric.sum().sum()  # total sum of all elements
    normalized_df =  df_numeric / total_sum
    return normalized_df

#  Euclidean distance
def calculate_distance(df1, df2):
    if df1.shape != df2.shape:
        raise ValueError("DataFrames do not have the same shape")
    distance = np.sqrt((df1 - df2)**2)
    return distance

# Initialize a DataFrame to store distances
distance_results = pd.DataFrame(columns=imq_files, index=dataset_files)

#  calculate distances and fill the DataFrame
def calculate_and_fill_distances(dataset_files, imq_files):
    for d_file in dataset_files:
        dataset_df = read_and_normalize(os.path.join(dataset_dir, d_file))
        for i_file in imq_files:
            imq_df = read_and_normalize(os.path.join(imq_dir, i_file))
            if dataset_df.shape == imq_df.shape:
                try:
                    distance_matrix = calculate_distance(dataset_df, imq_df)
                    total_distance = distance_matrix.sum().sum()
                    distance_results.at[d_file, i_file] = total_distance
                except Exception as e:
                    print(f"Error in {d_file} vs {i_file}: {e}")
            else:
                distance_results.at[d_file, i_file] = np.nan
```

```
# Convert the DataFrame entries to numeric, replacing non-numeric values with NaN
distance_results_numeric = distance_results.apply(pd.to_numeric, errors='coerce')

    return distance_results_numeric

# List of file names in each directory
dataset_files = os.listdir(dataset_dir)
imq_files = os.listdir(imq_dir)

# Calculate distances and get numeric DataFrame
distance_results_numeric = calculate_and_fill_distances(dataset_files, imq_files)

# Plotting the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(distance_results_numeric, annot=True, cmap='viridis')
plt.title('Heatmap of Euclidean Distances')
plt.ylabel('Dataset Files')
plt.xlabel('IMQ Files')
plt.show()
```

Final Comparison and plotting

```
path_c2go = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset\\od_matrix_c2go.csv"
path_enjoy = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\Dataset\\od_matrix_enjoy.csv"
path_no_filter = "C:\\Users\\hosse\\Desktop\\LAB3 Smart mobility\\Directory\\IMQ\\No_Filter.csv"

matrix_c2go = pd.read_csv(path_c2go, header=None)
matrix_enjoy = pd.read_csv(path_enjoy, header=None)
matrix_no_filter = pd.read_csv(path_no_filter, header=None)
def normalize_to_sum_one(df):
    total_sum = df.sum().sum() # total sum of all elements
     scaler = df / total_sum
    return df / total_sum
# Normalize and adjust the data
def normalize_and_adjust(matrix):
    scaled = scaler.fit_transform(matrix.values.reshape(-1, 1)).reshape(matrix.shape)
    adjusted = scaled
    return adjusted

# Applying normalization and adjustment
matrix_c2go_adjusted = normalize_and_adjust(matrix_c2go)
matrix_enjoy_adjusted = normalize_and_adjust(matrix_enjoy)
matrix_no_filter_adjusted = normalize_and_adjust(matrix_no_filter)

# Calculating Euclidean distances
def calculate_total_euclidean_distance(matrix1, matrix2):
    euclidean_distances = np.sqrt((matrix1 - matrix2) ** 2)
    return np.sum(euclidean_distances)

# Calculating distances for each pair
total_distance_c2go_enjoy = calculate_total_euclidean_distance(matrix_c2go_adjusted, matrix_enjoy_adjusted)
total_distance_no_filter_c2go = calculate_total_euclidean_distance(matrix_no_filter_adjusted, matrix_c2go_adjusted)
total_distance_no_filter_enjoy = calculate_total_euclidean_distance(matrix_no_filter_adjusted, matrix_enjoy_adjusted)
```

```
# Plotting the Euclidean distances
labels = ['C2Go vs Enjoy', 'No Filter vs C2Go', 'No Filter vs Enjoy']
distances = [total_distance_c2go_enjoy, total_distance_no_filter_c2go, total_distance_no_filter_enjoy]

# Data for plotting
labels = ['C2Go vs Enjoy', 'IMQ vs C2Go', 'IMQ vs Enjoy']
distances = [total_distance_c2go_enjoy, total_distance_no_filter_c2go, total_distance_no_filter_enjoy]
criterion_value = 0.98

# Plot configuration
plt.figure(figsize=(8, 7))
bars = plt.bar(labels, distances, color=['lightblue', 'purple', 'pink'], alpha=0.99)

# Adding a horizontal line for the criterion
plt.axhline(y=criterion_value, color='blue', linestyle='-', label='Criterion: Female vs Male')

# Adding text for labels, title, and x-axis tick labels
plt.ylabel('Total Norm Euclidean Distance')
plt.title('Total Euclidean Distances Between Matrices')
plt.xticks(labels, rotation=45, ha="right")

plt.legend()
plt.tight_layout()
plt.show()
```