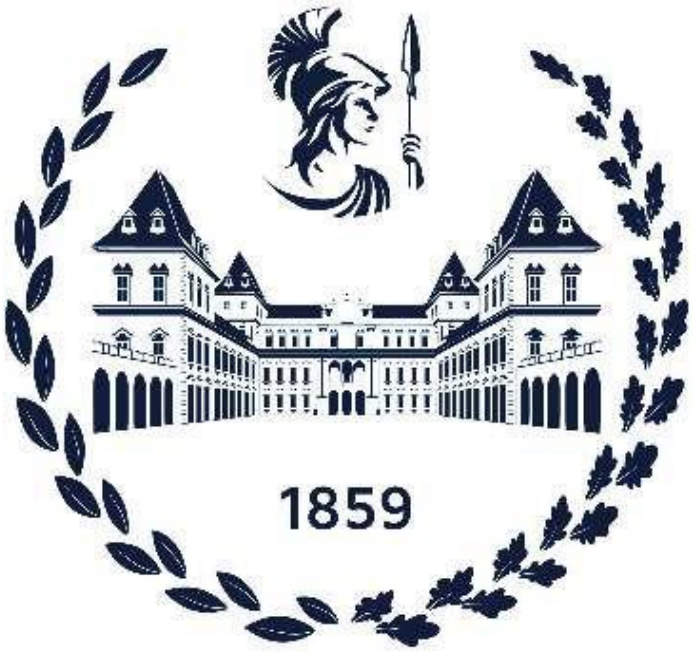


Politecnico di Torino

Laboratory 3(OD Matrix Similarity)

Group 07

Author

Ali Mohammad Alizadeh 308885

Mohammad Eftekhari Pour 307774

Supervisors:

Prof. Marco Mellia

Prof. Luca Vassio

Academic year 2023/24

Introduction

The goal of this laboratory experiment was to identify the probable user profiles of a Carsharing system. To achieve this objective, the task involved comparing the origin-destination matrices of carsharing users with a specified dataset. The aim was to determine the most closely matching OD matrix between the carsharing user data and the provided dataset.

Datasets

The IMQ dataset provides us with mobility patterns of individuals, various modes of transportation. Collected through telephone interviews in 2013 within the Piedmont region, our analysis specifically concentrated on 23 zones associated with the city of Turin. The dataset comprises journeys occurring from Monday to Friday and provides insights into individual profiles, including factors such as gender and age groups, as well as the motivations behind trips, such as work, study, shopping, and more.

Methodology

We created pivot tables using python (to avoid manual data handling in case of repetitive work) to extract the OD matrices. OD matrices were extracted Based on zone of arrival and departure by filtering data based on age, gender and trip motivations.

Records in Mongo dB retrieved by querying the database using pymongo and handling the IMQ dataset and zones dataset was done using Pandas library.

Data extraction

To create Origin-Destination (OD) matrices from the **Carsharing Rental dataset**, we utilized the "**ictts-PermanentBookings"** dataset in MongoDB. This dataset contains crucial information about the starting point, destination, and time of car rentals. Also, we separated the dataset to 4 different one with filters and ended up with these data, "Weekday-Morning", "Weekday-afternoon", "Weekend-Morning", "Weekend-Afternoon". For the morning we took the data during hours 6am to 12pm and for the afternoon we picked the data between 12pm and 20 pm. Weekday are from Monday to Friday and Weekend are the Saturday and Sunday.

We defined zones using **GeoJSON data specific for Turin** to categorize rental locations accurately which will eventually be used to map each trip from a specific zone to another inside the described regions.

The data separation allows us to compare in detail different pattern for different situations based on different days and hours and in what sense they are different from one another.

To obtain Origin-Destination (OD) matrices from the **IMQ dataset**, we utilized the pivot table functionality in Pandas. This method simplified the extraction process, providing a direct means to create OD matrices based on the departure and arrival zones. Through data filtering based on gender, age, and trip motivations, we customized the OD matrices, shedding light on travel patterns and preferences across various demographic segments.

Distance calculation (Comparison)

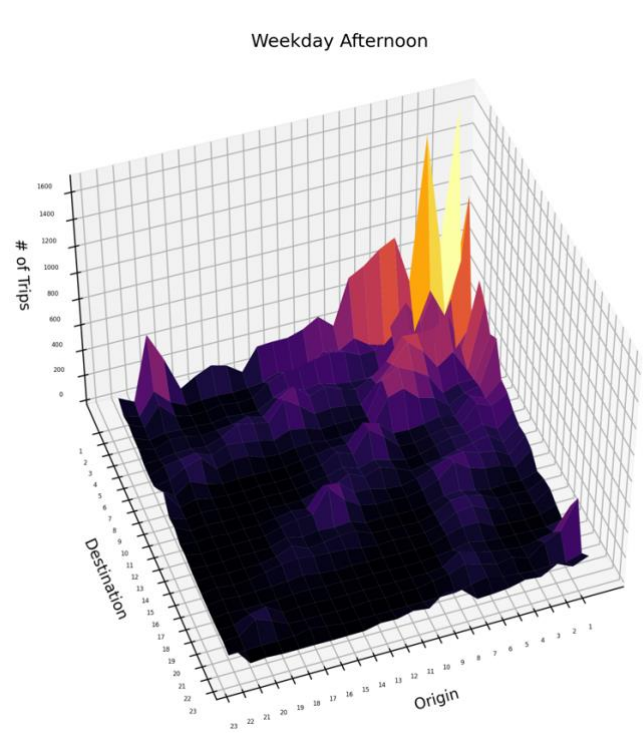
In our analysis, we employed the Euclidean distance metric to quantify the dissimilarity between two distinct matrices. **Euclidean distance** measures the straight-line distance between points in a multidimensional space, providing a quantitative measure of dissimilarity. By calculating the Euclidean distance between corresponding elements of the matrices, we obtained a numerical representation of their dissimilarity. It's important to note that a smaller Euclidean distance indicates a higher similarity, while a larger distance suggests greater dissimilarity. Therefore, the results from our distance calculations served as a basis for evaluating the similarity between matrices. In essence, a closer Euclidean distance implies a stronger resemblance between matrices, highlighting the relationship between distance and similarity in our analysis.

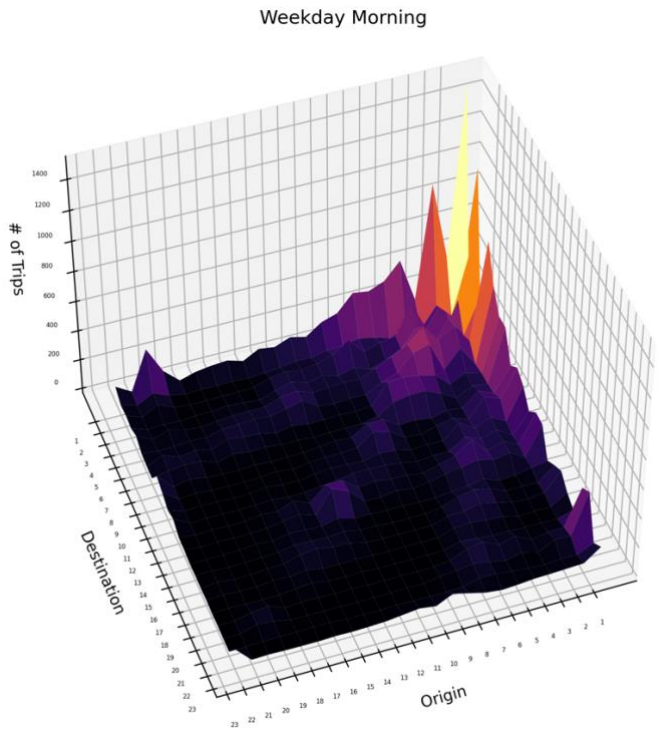
$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij} - B_{ij})^2}$$

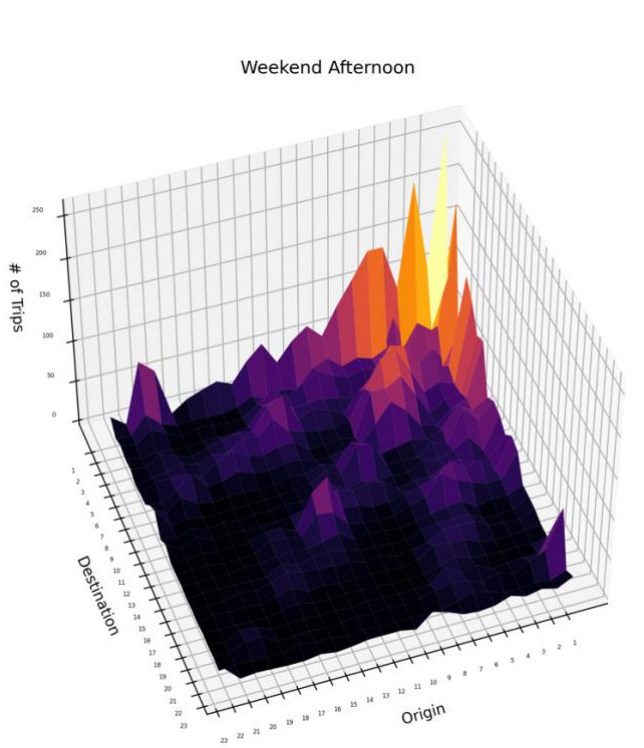
Data Visualization (Carsharing Datasets)

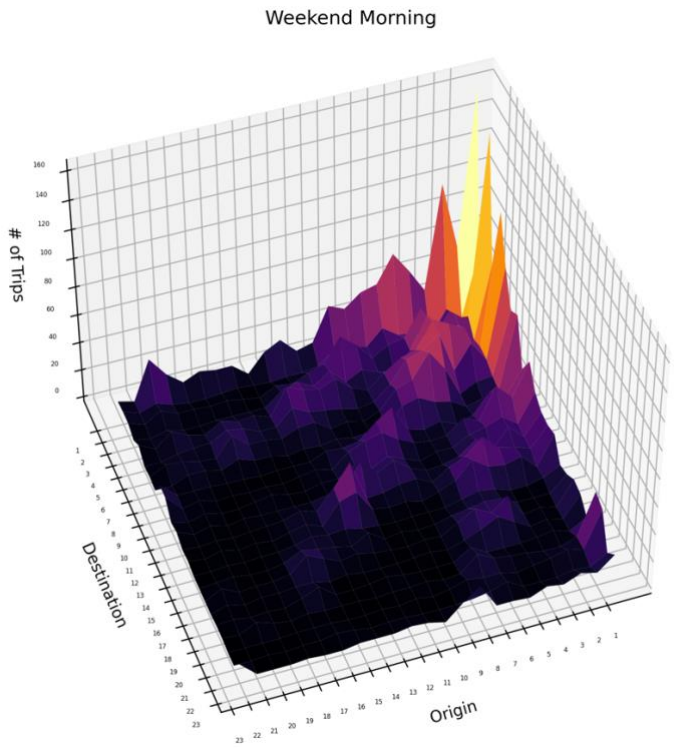
When dealing with matrices, the mesh-grid allows us to define the x and y coordinates for each element in the matrix, creating a mesh-like grid. This grid is then utilized to plot 3D surfaces or other visualizations. For example, in heatmaps, the mesh-grid helps to accurately position each data point, ensuring that the visualization reflects the underlying structure of the matrix. In summary, mesh-grid diagrams are beneficial for translating matrix data into a visual representation, especially in 3D plots, by providing a structured grid that aligns with the matrix elements.

The Data extracted from IMQ dataset includes some interesting data to be visualized and compared, these visualizations are created based on the origin zone, destination zone, and the ratio of the number of trips to the maximum number of trips. The data is extracted from the Origin-Destination (OD) matrices of the IMQ dataset, considering various filters to capture different scenarios.









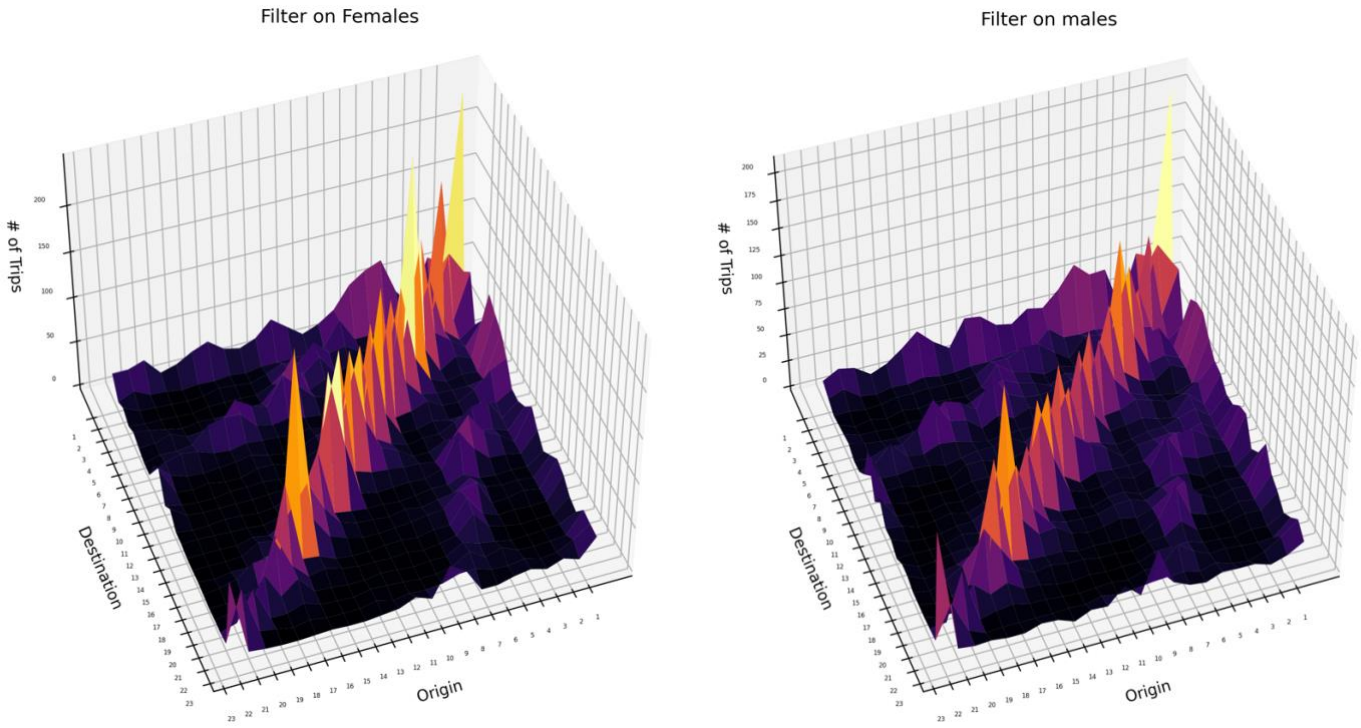
First, we plot the mesh-grid for different situations we derived from carsharing dataset and mapped with GeoJson dataset to get the number of corresponding trips among different regions. The lighter the color the more the number of trips and the darker the color the lower the number of trips. As it can be seen most of the concentration of trips is related to

city center which is Zone 1 of Torino, the only difference is the total number of trips but the heatmap helps us to capture the patterns. this pattern can be seen for different week days and hours since the city center provides citizens with different facilities such as daily routing works and shopping and leisure activities and tourist attractions and …. Also we can observe that the trips with the same origin and destination zone 1 has the highest number, the movement between zones close to city center is higher and city center is the most popular origin or destination.

Data Visualization (IMQ Datasets)

we set different filters on IMQ dataset as described in class and then mapped with GeoJson file, the result of related filters are as shown

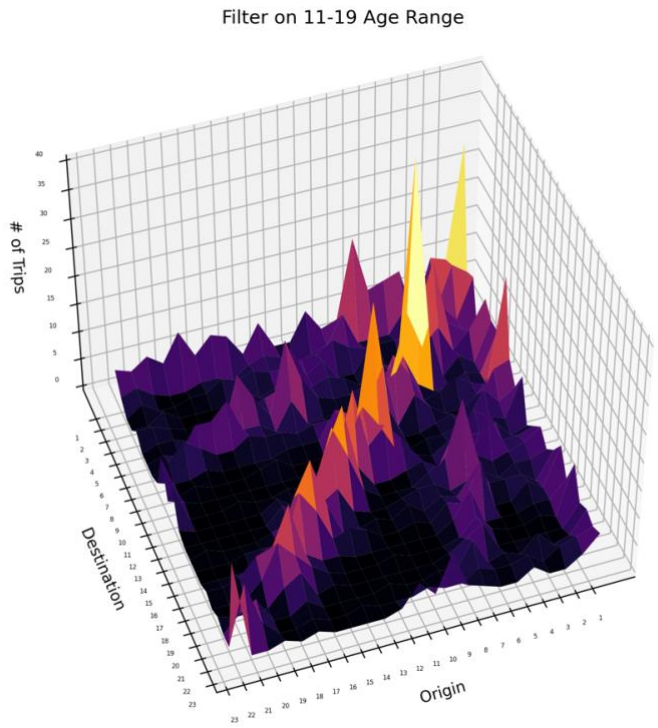
Filter on Gender

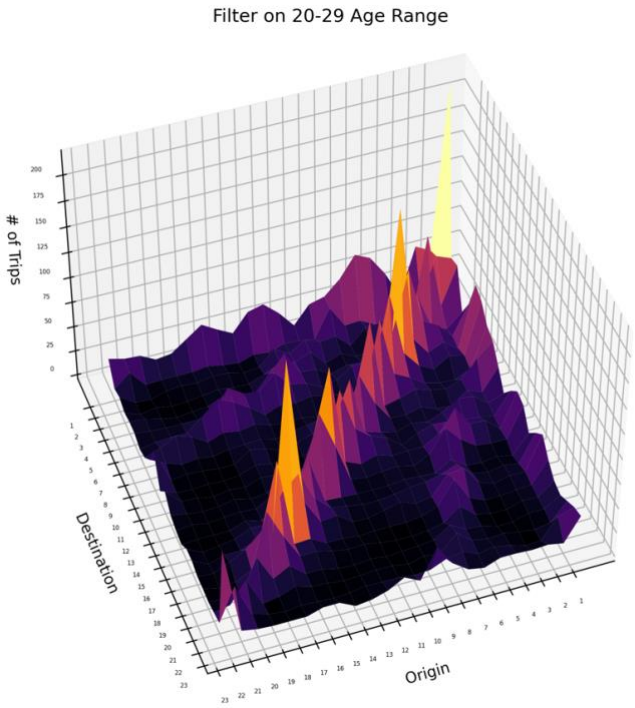


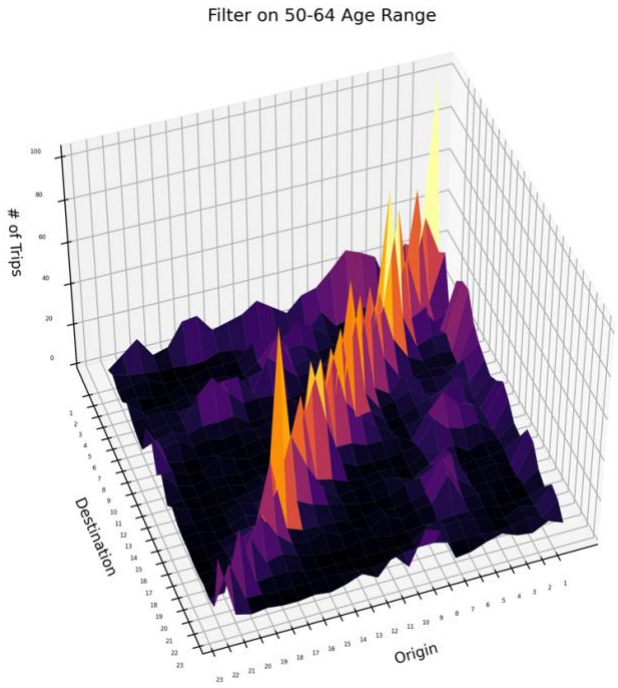
The observed data is showing almost the same pattern for both male and female with a touch of difference in total number of trips which is higher in figure related to females than males, this might be due to more participation of women in data collection phase. Also another interesting pattern can be seen which is the fact that the diagonal part has the higher density which shows that trips are mostly happening in the same zone.

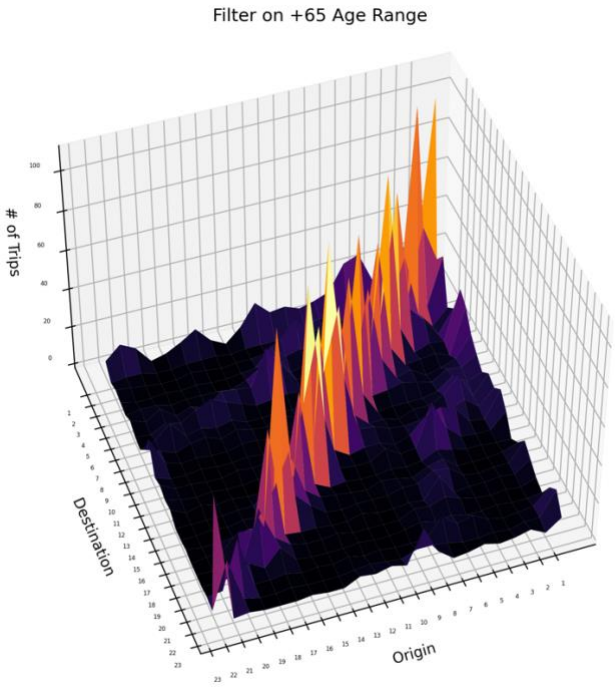
Filter on Age Groups

Observing the figures provided below, the most spread data is related to age group of 11-19 years old and the most concentrated data is related to age group +65. The reason might be different activities that each age group does on daily bases like study or sport and leisure activities in different zones for younger people and activities done in the same zone for older people. Also, it can be seen that the total number of trips done by older people is much higher than younger people according to the survey.

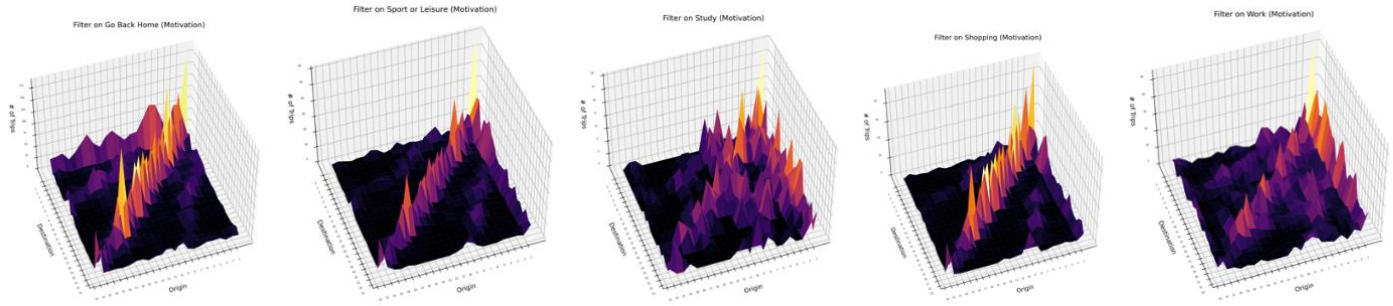








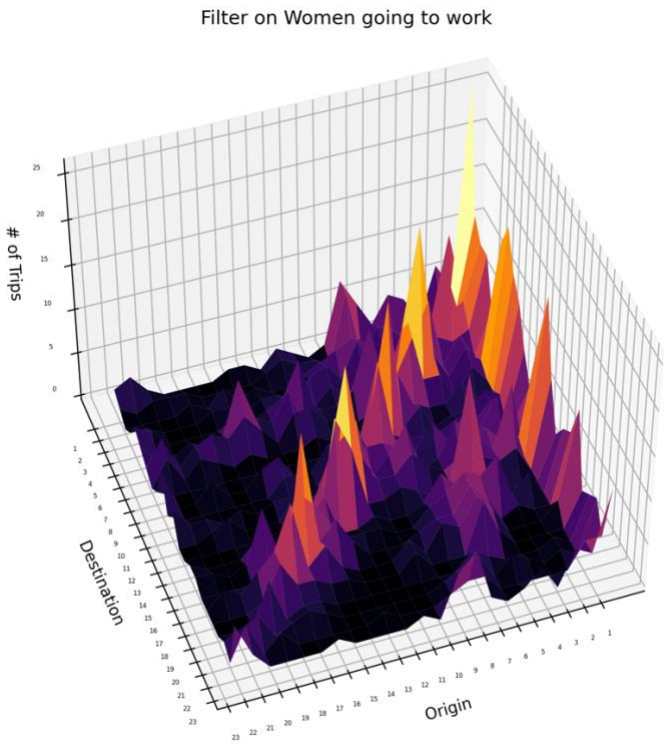
Filter on Trip Motivations

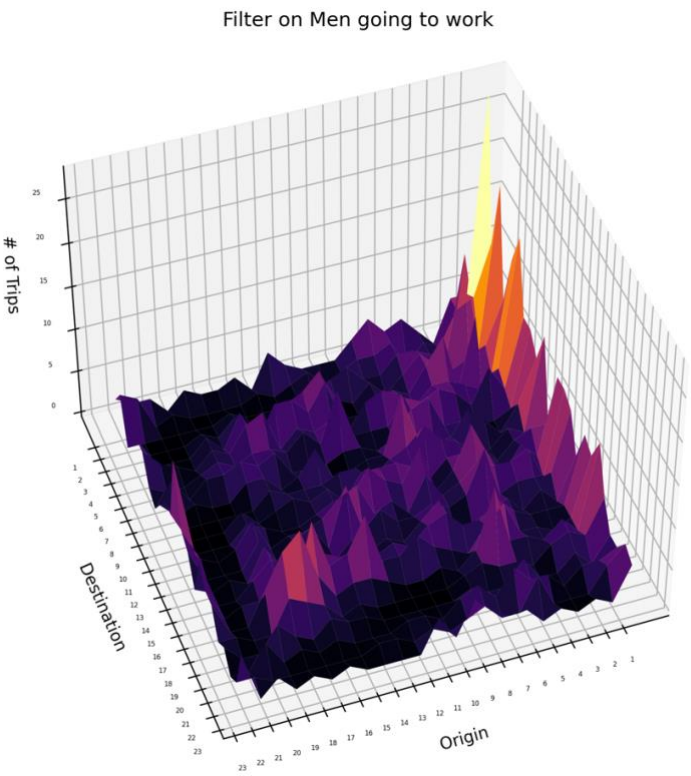


Selected filters for motivations were “Go to Work”, “Shopping”, “Study”, “Sport or Leisure”, “Going Back Home”. There are different spreading patterns for different Motivations, as shown the most spread trips in terms of origin and destination is related to Work and Study this might be due to studying and working in different zones and commuting between them, shopping and sport activities mostly done in the same zone the trips were started.

Filter on Gender and Motivation

We tried to study the data with the filters of gender (Male, Female) with the same trip motivation which in this case is “Going to Work” and the result is shown in the next figures:



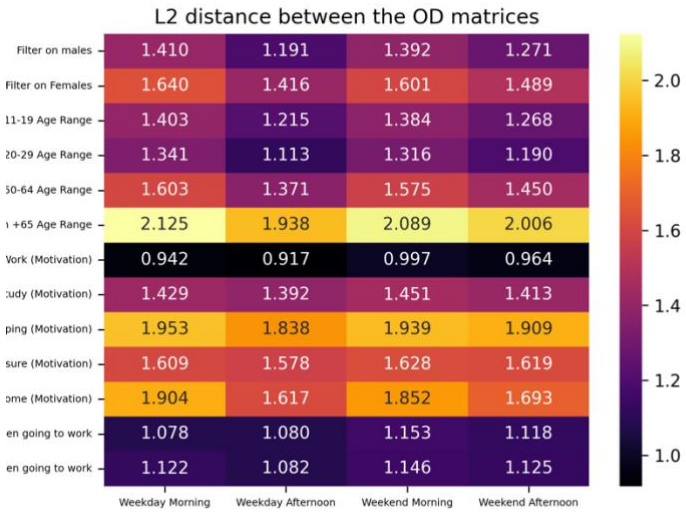


According to the data collected through phone calls it is observed that the women traveling with work purposes are more than males with the same purpose and the spread of the data is almost the same, but the difference in the numbers might be due to more collaboration from females than males. But over all to make sure of the similarity between the data collected using surveys and the carsharing data we need to calculate the distance between different matrices introduced before.

Understanding different travel patterns is vital for urban planning and transportation services. It helps customize strategies to meet the specific needs of various groups, leading to more efficient and accessible urban mobility.

Distance Comparison

Each Data set related to Carsharing and filtered IMQ Datasets were compared to each other using the Distance formula and utilizing Pandas library methods. The distance shows the similarity between different datasets. The lowest number in terms of distance which is the highest similarity is related to **“Weekday Afternoon”** and **“Going to Work”** with the values of **0.917**, and the lowest similarity is related to **“Weekday Morning”** and **“+65 age Group”** with the values of 2.125.



Based on first two rows of the table, it can be said that the similarity between data related to men is higher than the IMQ data related to women and males’ users have better match.

Conclusion

According to the Figures and the data provided in brief we can observe:

- The trips done by men have the most similarity
- In age groups, the best fit is related to 20-49.
- In terms of motivations, the best match is related to Going to Work, and after that in corresponding order are Study and Sport or Leisure activities.
-

In [ ]:

```
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_p_booking = db['ictts_PermanentBookings'] #PermanentBookings

with open("TorinoZonescol.geojson") as f: #GeoJson file with the zones of Turin
    gj = geojson.load(f)

#Function to get the origin and destination zones of a booking using geoWithin operator of MongoDB
#weekday - days 2-6 -morning 6-12 - afternoon 12-20
#weekend - days 1,7 -morning 6-12 - afternoon 12-20
def weekday_piper(start_hour,end_hour,origin_zone,destination_zone):
  return [
    { "$project":
     {
       "hour":{"$hour":"$init_date"},
       "day":{"$dayOfWeek":"$init_date"},
       "init_loc":1,
       "final_loc":1,
       "init_time":1
       }
    },
    { "$match": {
      "day":{"$gte":2,"$lte":6},
      "hour":{"$gte":start_hour,"$lte":end_hour},
      "init_loc":{"$geoWithin":{"$geometry":{"type":"MultiPolygon","coordinates":origin_zone}}},
      "final_loc":{"$geoWithin":{"$geometry":{"type":"MultiPolygon","coordinates":destination_zone}}}}
    },
    { "$count":"total"}
  ]

def weekend_piper(start_hour,end_hour,origin_zone,destination_zone):
  return [
    { "$project":
     {
       "hour":{"$hour":"$init_date"},
       "day":{"$dayOfWeek":"$init_date"},
       "init_loc":1,
       "final_loc":1,
       "init_time":1
       }
    },
    { "$match":{
      "day":1 and 7,
      "hour":{"$gte":start_hour,"$lte":end_hour},
      "init_loc":{"$geoWithin":{"$geometry":{"type":"MultiPolygon","coordinates":origin_zone}}},
      "final_loc":{"$geoWithin":{"$geometry":{"type":"MultiPolygon","coordinates":destination_zone}}}}},
    {"$count": "total"}
  ]

#Function to get the total number of bookings in a specific zone in a specific time interval
#orining zone and destination zone are retieved from the geojson file
#origin_zone and destination_zone are the coordinates of the zone as a list of lists which is passed to the geoWithin operato
def extract_od_matrix(start_hour =1 , end_hour =23, pipeline=[{}]):
  OD_matrix = [([0]*23) for i in range(23) ]
  for i in range(23) :
    orig_zone = gj["features"][i]["geometry"]["coordinates"]
    for j in range(23) :
      dest_zone = gj["features"][j]["geometry"]["coordinates"]
      result = list ( Ictts_p_booking.aggregate(pipeline(start_hour,end_hour,orig_zone,dest_zone)))
      if( len(result ) > 0):
        OD_matrix[i][j] = result[0]["total"]
      else :
        OD_matrix[i][j] = 0
  output_df = pd.DataFrame ( OD_matrix )
  output_df.columns =["Q"+f"{i:03d}" for i in range(1, 24) ]
  output_df['index'] =["Q"+f"{i:03d}" for i in range(1, 24) ]
  output_df = output_df.set_index('index', drop = True ).rename_axis( None )
  return output_df

#now we can use the function to get the OD matrix for the morning and afternoon of weekdays and weekends
weekday_morning = extract_od_matrix(6,12,weekday_piper)
weekday_afternoon = extract_od_matrix(12,20,weekday_piper)
weekend_morning = extract_od_matrix(6,12,weekend_piper)
weekend_afternoon = extract_od_matrix(12,20,weekend_piper)
```

```
booking_OD_matrix = []
booking_OD_matrix.append(weekday_morning)
booking_OD_matrix.append(weekday_afternoon)
booking_OD_matrix.append(weekend_morning)
booking_OD_matrix.append(weekend_afternoon)

weekday_morning.to_csv("weekday_morning.csv")
weekday_afternoon.to_csv("weekday_afternoon.csv")
weekend_morning.to_csv("weekend_morning.csv")
weekend_afternoon.to_csv("weekend_afternoon.csv")

bookingFigureTitles = ["Weekday Morning","Weekday Afternoon","Weekend Morning","Weekend Afternoon"]

#filter the IMQ matrix with different parameters
IMQ = pd.read_csv("spostamentiTorino.csv")
copyIMQ = IMQ.copy()
IMQ_OD_matrices = []

filtered_data = copyIMQ[copyIMQ['SESSO']==1]
pivot_table1 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table1.to_csv("IMQ_OD_1"+".csv")

filtered_data = copyIMQ[copyIMQ['SESSO']==2]
pivot_table2 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table2.to_csv("IMQ_OD_2"+".csv")

filtered_data = copyIMQ[copyIMQ['FASCIA_ETA']==1]
pivot_table3 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table3.to_csv("IMQ_OD_3"+".csv")

filtered_data = copyIMQ[copyIMQ['FASCIA_ETA']==2]
pivot_table4 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table4.to_csv("IMQ_OD_4"+".csv")

filtered_data = copyIMQ[copyIMQ['FASCIA_ETA']==3]
pivot_table5 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table5.to_csv("IMQ_OD_5"+".csv")

filtered_data = copyIMQ[copyIMQ['FASCIA_ETA']==4]
pivot_table6 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table6.to_csv("IMQ_OD_6"+".csv")

filtered_data = copyIMQ[copyIMQ['SCOPO']==1]
pivot_table7 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table7.to_csv("IMQ_OD_7"+".csv")

filtered_data = copyIMQ[copyIMQ['SCOPO']==3]
pivot_table8 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table8.to_csv("IMQ_OD_8"+".csv")

filtered_data = copyIMQ[copyIMQ['SCOPO']==4]
pivot_table9 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table9.to_csv("IMQ_OD_9"+".csv")

filtered_data = copyIMQ[copyIMQ['SCOPO']==7]
pivot_table10 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table10.to_csv("IMQ_OD_11"+".csv")

filtered_data = copyIMQ[copyIMQ['SCOPO']==8]
pivot_table11 = filtered_data.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
pivot_table11.to_csv("IMQ_OD_10"+".csv")


#-------------------------------------------------------------------------------------------------------------------
#male-work and female-work
filter1 = (copyIMQ['SESSO']==1)
filter2 = (copyIMQ['SCOPO']==1)
men_work =  copyIMQ[filter1 & filter2]
men_work_pivot = men_work.pivot_table( index ='COD_ZONA_PAR',
                                      columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
men_work_pivot.to_csv("IMQ_OD_11"+".csv")


filter1 = (copyIMQ['SESSO']==2)
filter2 = (copyIMQ['SCOPO']==1)
women_work =  copyIMQ[filter1 & filter2]
women_work_pivot = women_work.pivot_table( index ='COD_ZONA_PAR',
                                          columns='COD_ZONA_ARR',values='ID_INT',aggfunc =len ,fill_value =0)
women_work_pivot.to_csv("IMQ_OD_12"+".csv")
#-------------------------------------------------------------------------------------------------------------------

IMQ_OD_matrices.append(pivot_table1)
IMQ_OD_matrices.append(pivot_table2)
IMQ_OD_matrices.append(pivot_table3)
```

```
IMQ_OD_matrices.append(pivot_table4)
IMQ_OD_matrices.append(pivot_table5)
IMQ_OD_matrices.append(pivot_table6)
IMQ_OD_matrices.append(pivot_table7)
IMQ_OD_matrices.append(pivot_table8)
IMQ_OD_matrices.append(pivot_table9)
IMQ_OD_matrices.append(pivot_table10)
IMQ_OD_matrices.append(pivot_table11)
#------------------------------------------------------------
#male-work and female-work
IMQ_OD_matrices.append(men_work_pivot)
IMQ_OD_matrices.append(women_work_pivot)
#------------------------------------------------------------
# to be used in the heatmap as labels also in meshgrid titles
figureTitles = ["Filter on males", "Filter on Females",
                "Filter on 11-19 Age Range", "Filter on 20-29 Age Range",
                "Filter on 50-64 Age Range", "Filter on +65 Age Range",
                "Filter on Work (Motivation)", "Filter on Study (Motivation)",
                "Filter on Shopping (Motivation)", "Filter on Sport or Leisure (Motivation)",
                "Filter on Go Back Home (Motivation)",
                "Filter on Men going to work", "Filter on Women going to work"]


#------------------------------------------------------------
#calculating the distance between each two matrices
# • Comparison between the OD Matrices
# • L2 distance between the OD matrices formula is given by:
# L2_distance = sqrt( sum( ( normalized_matrix1 - normalized_matrix2 )^2 ) )
def L2_distance( matrix1 , matrix2 ):
 # Ensure matrices have the same dimensions
 # assert is used to check if a condition is true, if not, the program will raise an error
 assert matrix1.shape == matrix2.shape , "Matrices must have the same dimensions"
 normalized_matrix1 = matrix1 / matrix1.sum( axis =1, keepdims = True )
 normalized_matrix2 = matrix2 / matrix2.sum( axis =1, keepdims = True )

 # Calculate the squared differences between corresponding cells
 squared_diff = np.square( normalized_matrix1 - normalized_matrix2 )
 # Sum the squared differences
 sum_squared_diff = np.sum( squared_diff )
 l2_distance = np.sqrt( sum_squared_diff )
 return l2_distance

#creating a list of lists to store the distances between each two matrices
distances = [([0]* len( booking_OD_matrix )) for i in range(len( IMQ_OD_matrices ))]

#calculating the distance between each two matrices
for i, imq in enumerate( IMQ_OD_matrices ):
  for j, rental in enumerate( booking_OD_matrix ):
    distance = L2_distance(imq.values , rental.values )
    distances[i][j] = distance

#  plotting the heatmap of the distances
hm = sb.heatmap( data =np.array( distances ), annot = True, fmt=".3f",
                cmap="inferno" ,xticklabels=bookingFigureTitles, yticklabels=figureTitles)#rainbow
hm.set_xticklabels(hm.get_xticklabels(), rotation=0, fontsize=6)
hm.set_yticklabels(hm.get_yticklabels(), rotation=0, fontsize=6)
plt.title("L2 distance between the OD matrices")
plt.savefig("L2_distance.png", dpi=300)
plt.show()
plt.close()

#------------------------------------------------------------
# Plotting the OD Matrices using 3D surface plots and meshgrids
def plot_matrix( od_matrix , title ):
 x, y = np.meshgrid( range(od_matrix.shape[0]), range(od_matrix.shape[1]))

 fig = plt.figure(dpi =300 , figsize =(8 , 8))
 ax = fig.add_subplot(111 , projection ='3d')
 ax.plot_surface(x, y, od_matrix , cmap ='inferno')#rainbow
 ax.set_xlabel('Origin ')
 ax.set_ylabel('Destination ')
 ax.set_zlabel('# of Trips')
 ax.set_title( title )

# Customize tick positions for both x and y axes
 ax.set_xticks(np.arange(1,24))
 ax.set_yticks(np.arange(1,24))
 ax.tick_params(axis='both', which='major', labelsize=4)
 ax. view_init ( elev =45 , azim =70)

 plt.savefig(title+".png", dpi=300)
#  plt.show()

#plotting the OD matrices
# calling the function plot_matrix for each matrix of the booking_OD_matrix and IMQ_OD_matrices
for i, element in enumerate( booking_OD_matrix):
  plot_matrix( booking_OD_matrix[i].values , bookingFigureTitles[i])

for i, od_matrix in enumerate( IMQ_OD_matrices ):
  plot_matrix( IMQ_OD_matrices[i].values , figureTitles[i])
```

In [ ]: