POLITECNICO DI TORINO

ICT in Transport Systems

Lab 1 report

GROUP 4

Professor:

Marco Mellia

Students Group 4:

Can Akgol 274948
Paolo De Santis 280398
Francesco Conforte 277683

Academic year 2020/2021

Contents

1 Preliminary data analysis **2**
1.1 Investigation of collections at disposal . . . 2

2 Analysis of the data **3**
2.1 Cumulative distribution function of booking/parking durations . . . 3
2.2 System utilization over time and filtering outliers criteria . . . 4
2.3 Rentals statistics . . . 4

3 Rentals location analysis **5**

4 Optional Task: Probability of a rental with respect to the availability of other transport means **7**

A Step 1 **8**
A.1 Connection to Database . . . 8
A.2 How many documents are present in each collection? . . . 8
A.3 For which cities the system is collecting data? . . . 8
A.4 When the collection started? When the collection ended? . . . 8
A.5 How many cars are available in each city? . . . 9
A.6 How many bookings have been recorded on the November 2017 in each city? . . . 9
A.7 How many bookings have also the alternative transportation modes recorded in each city? . . . 9

B Step 2 **9**
B.1 Cumulative distribution function . . . 9
B.2 System utilization . . . 12
B.3 Rentals statistics analysis . . . 14
B.4 Rentals location analysis . . . 17
B.4.1 Parking positions . . . 17
B.4.2 O-D Matrix . . . 18
B.5 Optional Task . . . 22

1 Preliminary data analysis

The aim of the first laboratory was to investigate and analyse free floating car sharing data collected from open systems in Internet (mainly from sites of FFCS like *Car2go* and *Enjoy*). Data were made available throughout a *MongoDB* database.

The goal was to extract data by making queries with filters, in order to obtain only useful information to make right analyses.

To have access to MongoDB stored data, Python's programming language with the package '*pymongo*' was used. Moreover, the analysis was focused on data stored in collections regarding Car2Go and only for three cities: Milano, Amsterdam and Calgary.

1.1 Investigation of collections at disposal

Firstly, in order to get used to available data and to MongoDB, some simple queries were done and the following information were gathered:

- **How many documents are present in each collection?**

  CAR2GO
  ActiveBookings: 8743
  ActiveParkings: 4790
  PermanentBookings: 28180508
  PermanentParkings: 28312676

  ENJOY
  enjoy_ActiveBookings: 0
  enjoy_ActiveParkings: 0
  enjoy_PermanentBookings: 6653472
  enjoy_PermanentParkings: 6689979

- **Why the number of documents in PermanentParkings and PermanentBookings is similar?**

  The reason is because when someone wants to use a car: firstly he/she books the car and, after using it, he/she parks the car. The system, in turn, does the same: it stores firstly a booking and subsequently it stores a parking. The number is not exactly the same because sometimes there may be errors or maintenance to the system.

- **For which cities the system is collecting data?**

  Collections about Car2Go contain data for: Wien, Washington DC, Vancouver, Twin Cities, Toronto, Torino, Stuttgart, Seattle, San Diego, Roma, Rheinland, Portland, New York City, Munchen, Montreal, Milano, Madrid, Hamburg, Frankfurt, Firenze, Denver, Columbus, Calgary, Berlin, Austin, Amsterdam.

  Collections about Enjoy contain data for: Bologna, Catania, Firenze, Milano, Roma, Torino.

- **When the collection started? When the collection ended?**

  To identify when data started to be collected, the Booking collections of Car2Go and Enjoy were visited, by ordering all the documents in descending order by 'init_time', which is the Unix timestamp of the booking. About **Car2go**, data started to be collected on December 13^{rd} 2016 at 17:38:23 UTC and they ended to be collected on January 31^{st} 2018 at 13:11:33 UTC. About **Enjoy**, data started to be collected on May 5^{th} 2017 at 15:06:21 UTC and they ended to be collected on June 10^{th} 2019 at 17:16:20 UTC.

- **What about the timezone of the timestamps?**

  The timestamp that can be recovered by the field 'init_time' of MongoDB documents is the Unix timestamp, which is the number of seconds that have been spent since January 1^{st} 1970 at UTC. This can also be demonstrated by the fact that the first booked car of Car2Go was in Vancouver and the recovered date from the field 'init_date' was on December 13^{rd} 2016 at 09:38:23 Local Time (UTC-8 hours). For this reason, it has been paid attention during the analysis to consider the local time.

- **How many cars are available in each city?**

  There were **1153** cars available in **Milano**, **976** in **Calgary** and **339** in **Amsterdam**.

- **How many bookings have been recorded on the November 2017 in each city?**

  There were **183197** bookings in **Milano**, **85475** in **Calgary** and **39479** in **Amsterdam**.

- **How many bookings have also the alternative transportation modes recorded in each city?**

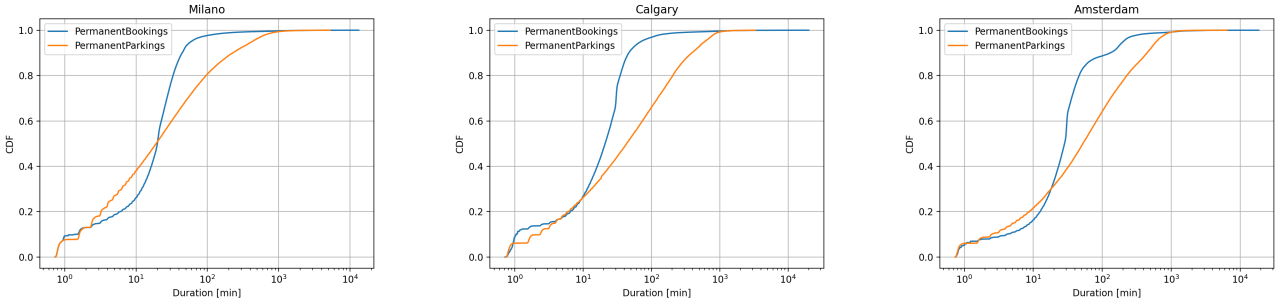
  Alternative transportation modes for bookings were available only for **Milano**. They were **729171**. For **Calgary** and **Amsterdam** they were not available.

2 Analysis of the data

In this part of the laboratory the analysis of the data will be limited only to the month of November 2017 and to three cities: Milano, Amsterdam and Calgary.

2.1 Cumulative distribution function of booking/parking durations

First of all, the duration was computed inside the MongoDB query by subtracting the final and initial timestamps and dividing it by 60, so to obtain the value in minutes. Then the data was grouped by duration and sorted in ascending order. Finally, the number of rentals for each duration were cumulated and divided by the total number so to obtain the CDF.

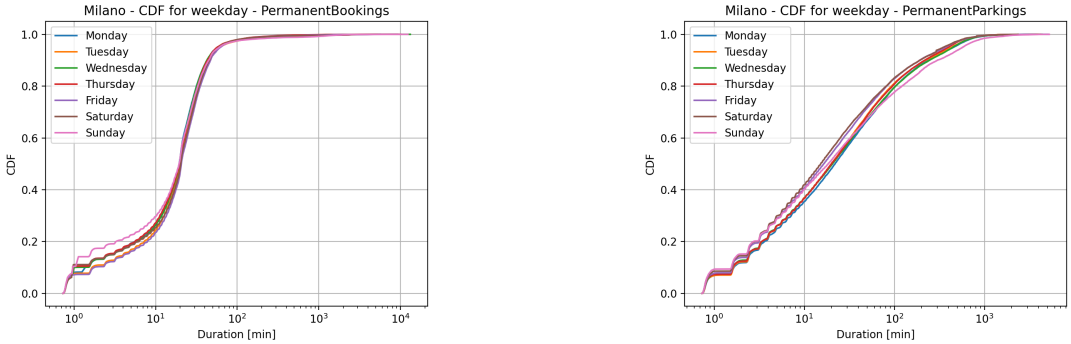


(a) Milano (b) Calgary (c) Amsterdam

Figure 1: Cumulative distribution function

As in can be seen in Figure 1, the CDF of bookings data is a little longer than the one of parkings for all the cities but both show very high values, which are probably outliers, as it is very unlikely that a booking/parking lasts more than a few hours. These values were probably recorded as a consequence of system errors, accidents or maintenance events, which were erroneously counted as bookings/parkings.

The CDF of bookings shows a similar behavior for Milano and Calgary, while for Amsterdam it can be seen that there's a lower percentage of short bookings (below 10 minutes). On the other end, the distribution of the parkings shows a similar behavior for Calgary and Amsterdam, while Milano shows that more than almost half of them last less than 10 minutes. This means that in Milano there is a higher usage of the system on average, so cars usually don't stay parked for a long time.



(a) Milano - Bookings (b) Milano - Parkings

Figure 2: Cumulative distribution function per weekday

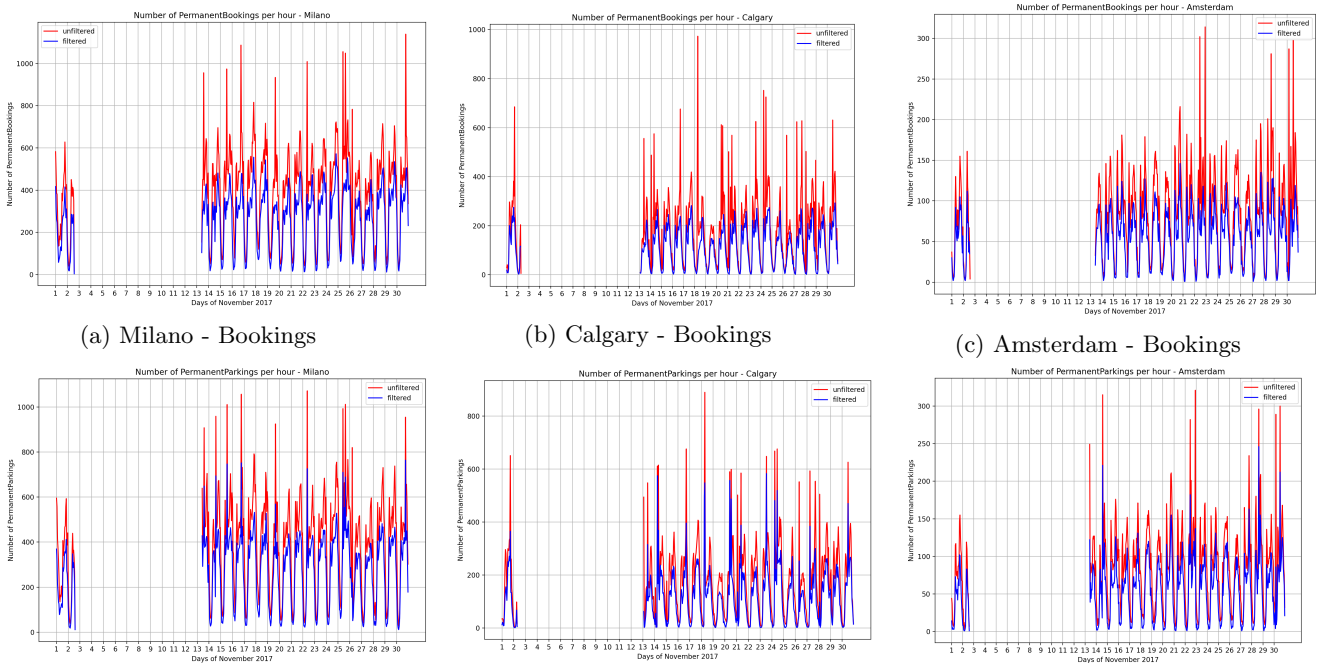
Figure 2 shows the cumulative distribution function per weekday for the city of Milano, both for bookings and parkings durations. It can be seen that during weekends (Saturdays and Sundays) there is a higher percentage of bookings and parkings which last a short amount of time, probably because people are going out for recreational activities, so they are willing to pay to use a more comfortable mean of transportation, even for very short distances. Another reason could be that during weekends people tend to go back home late at night when the public transport system is not available. Thus, the system is used more also for shorter distances.

2.2 System utilization over time and filtering outliers criteria

Until this task, the obtained figures had outliers which elongate lines in the graphs. In this part, the system utilization was analyzed by aggregating bookings and parkings per hour of the day. As it can be seen from Figure 3, where the number of rentals in progress for each city was plotted, there was a lack of data from November 3rd to November 12th, probably due to an error in the data collection system. The red lines in Figure 3 illustrate unfiltered version. The unusual peak points are visible signs of outliers in the dataset and useful filtering conditions were implemented to remove them. Such filters were generated according to different criteria by considering time and coordinates of bookings/parkings.

The first of the two filters was on the time measurements: bookings/parkings lasting less than 3 minutes were filtered out because unlikely. Probably, they were consequence of some errors in the system. Additionally, bookings/parkings having a duration of 180 minutes or more were filtered out as well, because of same reasons. Another filtering technique was performed over the positions of bookings: if final and initial coordinates of a rental were same, these points were counted as outliers too. In other words, rentals were considered as valid only if the origin coordinates and destination coordinates were different from each other.

After all these filters were applied, the final results were plotted with both the filtered and unfiltered data on same plots as reported in Figure 3. The first three figures illustrate the filtering on bookings: smoother graphs for filtered data with respect to unfiltered data were obtained. On the other hand, the filtering success on parking collection was not the same as the one on bookings, because the collection about parking had only a possible filtering criterion about the duration. As it is visible from Figures 3d, 3e, 3f, the difference between filtered and unfiltered data was less evident when compared with bookings graphs.



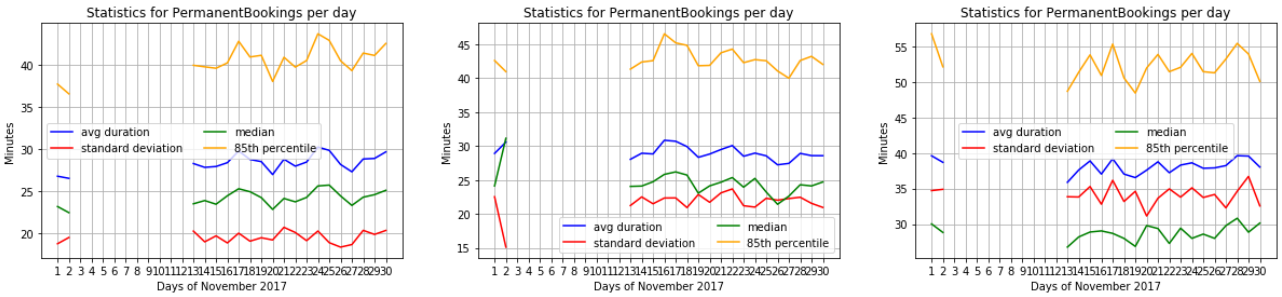
(a) Milano - Bookings (b) Calgary - Bookings (c) Amsterdam - Bookings

(d) Milano - Parkings (e) Calgary - Parkings (f) Amsterdam - Parkings

Figure 3: System utilization

2.3 Rentals statistics

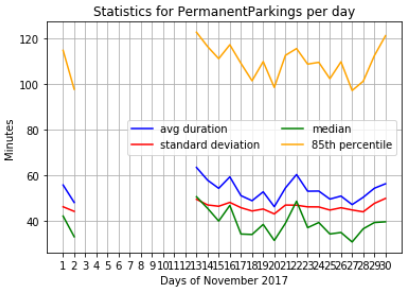
After removing outliers from datasets, average, median, standard deviation and 85th percentiles of the booking and parking duration over time were calculated. Their trends are shown in Figure 4. Amsterdam has vividly the highest average booking duration when compared with Milano and Calgary. It can be easily interpreted that the filtering was successfully performed for the bookings, because the standard deviation value is small and both median and average duration follow the same trend. This behavior is similar over all 3 cities as expected. If there were outliers, median and the average would be different. The average could be affected more from outliers with respect to the median but filtering also prevents this problem. Statistics have a slight variation over time and periodicity can be found. For example, a periodic spike in all the statistics can be spotted around the beginning of the weekends (November 17 - November 24), especially for the city of Milano.



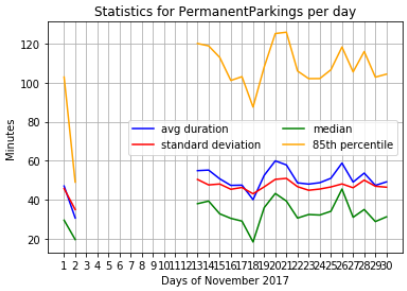
(a) Milano - Bookings

(b) Calgary - Bookings

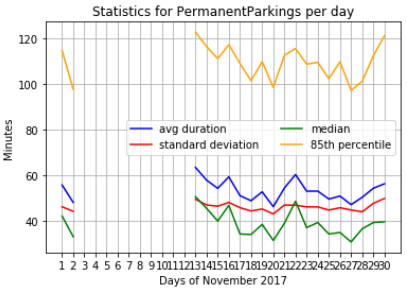
(c) Amsterdam - Bookings



(d) Milano - Parkings



(e) Calgary - Parkings



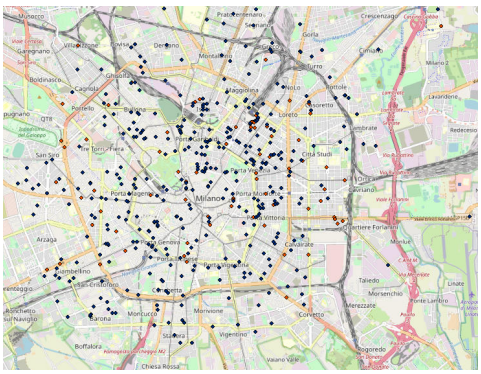
(f) Amsterdam - Parkings

Figure 4: Rentals Statistics

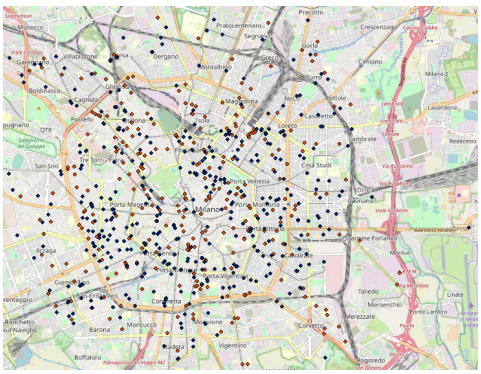
Figures 4d,4e,4f show the statistics for parkings. Median and average values seem similar again and also follow the same trend thanks to the filtering. However, 85th percentile is more variant due to the high standard deviation with respect to bookings. Indeed, in bookings, the standard deviation changes in between 15 and 35; whereas, in parkings it varies in between 40 and 50. As it was mentioned before, filtering was more successful in booking data and that's why high standard deviation is obtained for parking data. Moreover, the reason for which the standard deviation is higher for parkings with respect to bookings could also be because of the fact that they are two different actions: if one rents a car, then he/she needs to stop sooner or later, but if a car is parked somewhere outside the city, where few people are using the service, it's going to stay there longer. Additionally, parkings are influenced by the fact that the usage is quite different during the day and the night, so cars stay parked more during the night.

3 Rentals location analysis

In this part of the lab, only the city of Milano was taken into consideration. First of all, the parking location were plotted for two different periods of the day (Figure 5): from 8:00 to 9:00 and from 18:00 to 19:00, and for two reference days: a holiday (November 1st) and a working day (November 15th). In order to plot points on the map of Milano, ArcGIS was used in conjunction with OpenStreetMap.



(a) Holiday (November 1st)
Red Points: 8:00-9:00
Blue Points: 18:00-19:00

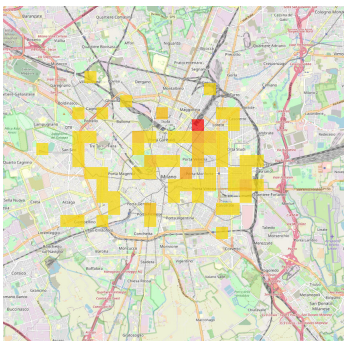


(b) Working day (November 15th)
Red Points: 8:00-9:00
Blue Points: 18:00-19:00

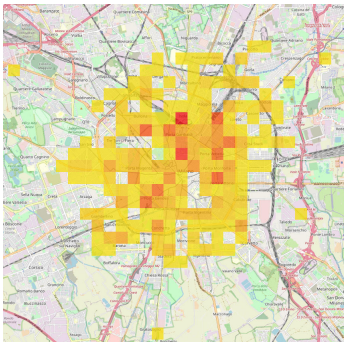
Figure 5: Parking locations

In order to better understand the distribution of parkings on the map, the density of parked cars per zone was computed by dividing the service area in squares measuring 500 m x 500 m. As it can be observed in Figure 6, on November 1st from 8:00 to 9:00, the most frequented zone is the one in proximity of the Central Station, which is actually always quite crowded. In fact, on the same day but from 18:00 to 19:00, the density of cars in that same area is still quite high but, differently from the morning, there are also other zones which are highly frequented, especially in the city center, like the ones close to Porta Garibaldi and Porta Genova.

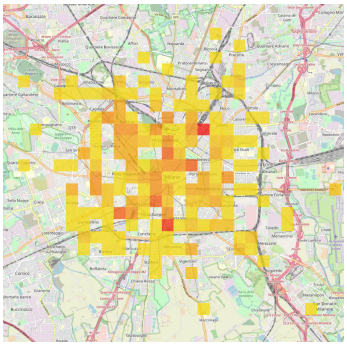
Also on November 15th, which was a regular working day, it can be seen that the zone close to the Central Station is still one of the most frequented, alongside with the zone close to Porta Vigentina, in the southern part of the city. On the same day, in the evening, the zone around Porta Garibaldi starts to get more crowded, since in that area there are many restaurants and bars, where people go after work.



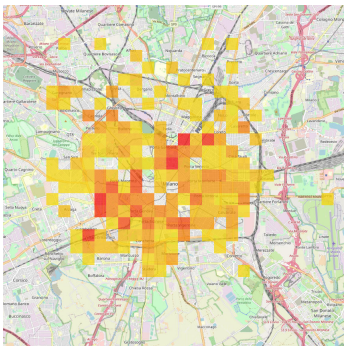
(a) Holiday 8-9 (November 1st)



(b) Holiday 18-19 (November 1st)



(c) Working day 8-9 (November 15th)



(d) Working day 18-19 (November 15th)

Figure 6: Parking location heatmaps

Finally, the O-D matrix, i.e. the matrix containing in each cell the number of trips from the origin O to the destination D, was computed, always using the division in square zones measuring 500 m x 500 m and limiting the analysis to the rentals of November 1st between 9:00 and 23:00. As shown in the heatmap of Figure 7, each zone was assigned an ID and the total trips between IDs was computed.

In order to have a better understanding of the most common paths, two additional plots were generated using ArcGIS: the first one (Figure 8a) shows all the trips from different zones of the city; the second one (Figure 8b) shows the most frequent paths, focusing on the city center. The dots represented in the two figures are the centroids of the 500 m x 500 m zones.

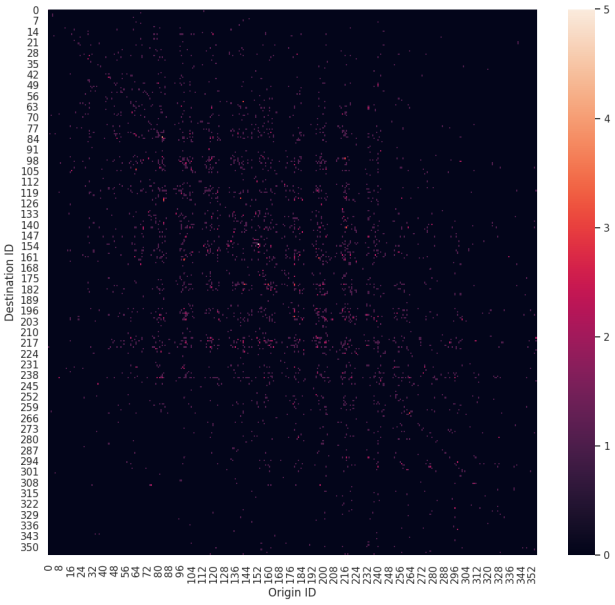
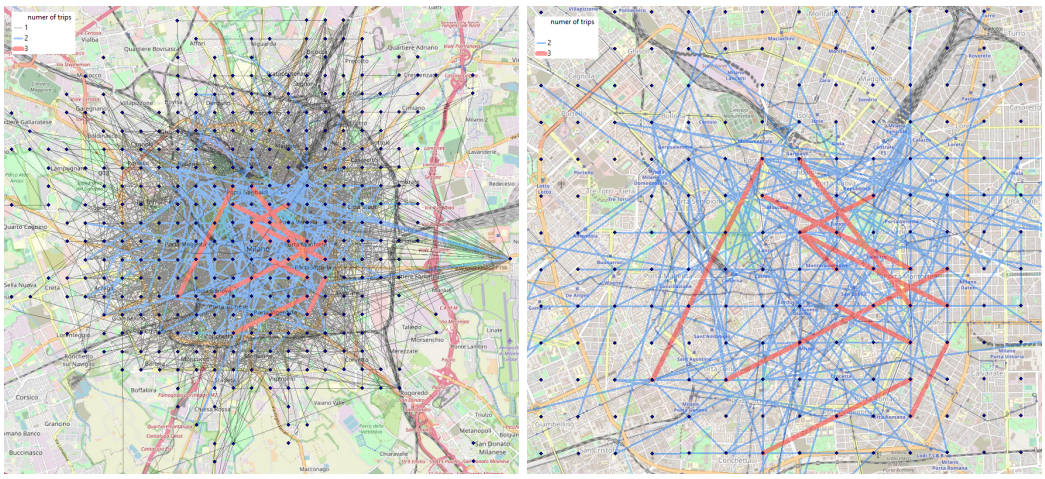


Figure 7: O-D matrix



(a) All trips (b) Most common trips

Figure 8: Trips on November 1st from 9 to 23

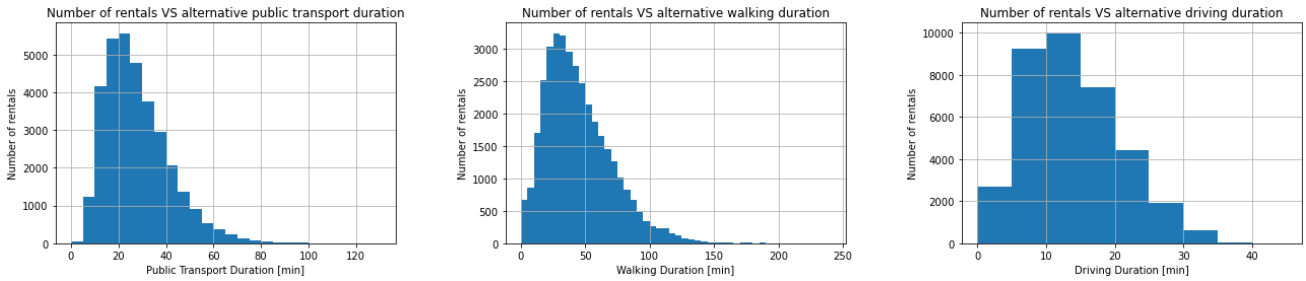
Looking at the most common paths, it is possible to spot some interesting points. First of all it is evident that the parking near the airport Linate (centre-right of Figure 8a) is the origin/destination of many trips, but most path only registered one trip for the day analyzed. Moreover, on that day, in the seaplane base of Milano, which is near the airport, there was also an international race of cyclocross. This may also be a reason of those trips. Another interesting fact that can be spotted, looking at the Figure 8b, is that the zone of the graveyard had many trips as origin/destination, because it is the moment of the year where most people visit it, since November 2^{nd} is the day in which deceased people are celebrated. Since the day in analysis was a holiday, it can also be seen that zones near parks and stations were quite frequented.

4 Optional Task: Probability of a rental with respect to the availability of other transport means

The last goal of the lab was to correlate the probability of a rental with the availability of other transport means, which were public transport, walking and driving. They were available only for two cities: Milano and Torino and the chosen one was Milano.

In Figure 9 histograms showing the number of rentals with respect to trips duration with alternative transportation systems are reported. To get these three graphs, same filters used for tasks in Section 2.2 were applied during data extraction and the duration was divided into bins of 5 minutes each. Finally, the number of rentals for each bin was computed.

As it can be observed, although the means of transport is different, all the three different figures have same trends: when the duration of the alternative transport system increases, the number of rentals increases in turn. This happens until the duration takes a certain number of minutes, in particular: 25 minutes for public transport, 30 minutes for walking and 15 minutes for alternative driving systems. Afterwards, the rentals start to decrease. This decrement is quite obvious, since the probability of having a so long trip with different alternative transport means in a city is low: there are no walking trips lasting for more than 40-50 minutes for example. Moreover, the probability of renting a car for a trip of more than 80 minutes in a city is low. In conclusion, it can be said that: longer the rental is, lower the probability of renting a car is.



(a) Alternative Public Transport duration (b) Alternative walking duration (c) Alternative driving duration

Figure 9: Number of rentals with respect to alternative transportation systems

A Step 1

A.1 Connection to Database

```
import pymongo as pm
from datetime import datetime

client = pm.MongoClient('bigdatadb.polito.it',
                        ssl=True,
                        authSource = 'carsharing',
                        tlsAllowInvalidCertificates=True)
db = client['carsharing']
db.authenticate('ictts', 'Ictts16!')
```

A.2 How many documents are present in each collection?

```
collectionName = ['ActiveBookings','ActiveParkings','PermanentBookings',
                  'PermanentParkings','enjoy_ActiveBookings',
                  'enjoy_ActiveParkings','enjoy_PermanentBookings',
                  'enjoy_PermanentParkings']

for i in collectionName:
    collection=db.get_collection(i)
    print(i + ": " + str(collection.estimated_document_count()))
```

A.3 For which cities the system is collecting data?

```
print('Cities Car2Go: ' +
      str(db.get_collection('PermanentBookings').distinct("city")))
print('Cities Enjoy: ' +
      str(db.get_collection('enjoy_PermanentBookings').distinct("city")))
```

A.4 When the collection started? When the collection ended?

```
timestamps_sorted_car2go = db.get_collection('PermanentBookings').find().sort(
    'init_time', pm.DESCENDING).distinct('init_time')
start_car2go=timestamps_sorted_car2go[-1]
end_car2go=timestamps_sorted_car2go[0]

timestamps_sorted_enjoy = db.get_collection('enjoy_PermanentBookings').find().sort(
    'init_time', pm.DESCENDING).distinct('init_time')
start_enjoy=timestamps_sorted_enjoy[-1]
end_enjoy=timestamps_sorted_enjoy[0]

ts_car2go = int(start_car2go) # initial time
te_car2go = int(end_car2go) #ending time

ts_enjoy = int(start_enjoy) # initial time
te_enjoy = int(end_enjoy) #ending time

print("UTC time of the first car collected in car2go")
print(datetime.utcfromtimestamp(ts_car2go).strftime('%Y-%m-%d %H:%M:%S'))

print("UTC time of the last car collected in car2go")
print(datetime.utcfromtimestamp(te_car2go).strftime('%Y-%m-%d %H:%M:%S'))

less = db.get_collection('PermanentBookings').find_one({'init_time':start_car2go})
date = less.get('init_date')
```

```
city_car2go = less.get('city')
print("Local time of the first car collected for car2go in " + str(city_car2go) + ": ")
print(date)

print("UTC time of the first car collected in enjoy")
print(datetime.utcfromtimestamp(ts_enjoy).strftime('%Y-%m-%d %H:%M:%S'))
print("UTC time of the last car collected in enjoy")
print(datetime.utcfromtimestamp(te_enjoy).strftime('%Y-%m-%d %H:%M:%S'))

less_enjoy = db.get_collection('enjoy_PermanentBookings').find_one({
                                                        'init_time':start_enjoy
                                                    })
city_enjoy = less_enjoy.get('city')
date_enjoy = less_enjoy.get('init_date')
print("Local time of the first car collected for enjoy in " + str(city_enjoy) + ": ")
print(date_enjoy)
```

A.5 How many cars are available in each city?

```
cities = ['Milano','Calgary','Amsterdam']
start = datetime(2017,11,1,0,0,0)#1novembre 2017 unixtime: 1509494400
end = datetime(2017,11,30,23,59,59) #30 novembre 2017 unixtime: 1512086399
for c in cities:
    car=db.PermanentBookings.distinct("plate", {"city": c})
    print("Cars in " + c + ": " + str(len(car)))
```

A.6 How many bookings have been recorded on the November 2017 in each city?

```
    bookings = db.PermanentBookings.count_documents(
        {"$and": [
                {"city":c},
                {"init_date": {"$gte":start} },
                {"final_date": {"$lte":end} }
                ]
            }
        )
    print("Booked cars in november 2017 in " + c + " are: " + str(bookings))
```

A.7 How many bookings have also the alternative transportation modes recorded in each city?

```
    alternative = db.PermanentBookings.count_documents(
                    {"$and": [ {"city":c},
                    {"$or":  [ {"walking.distance":{"$ne":-1} },
                               {"driving.distance":{"$ne":-1} },
                               {"public_transport.distance":{"$ne":-1} }
                               ]
                     }
                    ]
                     }
                    )
    print("Alternative transportation mode for " + c + ": " + str(alternative))
```

B Step 2

B.1 Cumulative distribution function

```
start = datetime(2017,11,1,0,0,0)
end = datetime(2017,11,30,23,59,59)

cities = ["Milano", "Calgary", "Amsterdam"]
collections = ["PermanentBookings", "PermanentParkings"]

for c in cities:
    for collection in collections:
        duration = db.get_collection(collection).aggregate(
            [
                { "$match" : {"$and": [{"city": c },
                                       {"init_date": {"$gte":start}},
                                       {"final_date": {"$lte":end}}
                                      ]
                              }
                },
                {
                    "$project": {
                        "_id": 1,
                        "city": 1,
                        "duration": { "$divide": [ { "$subtract": ["$final_time",
                                                                   "$init_time"]
                                                   },
                                                   60
                                                  ]
                                    },
                    }
                },
                { "$group": {
                        "_id": "$duration",
                        "tot_rentals": {"$sum": 1}
                }
                },
                { "$sort": {"_id": 1} }
            ]
        )

        value_id = []
        value_totrentals = []
        for i in list(duration):
            value_id.append(i["_id"])
            value_totrentals.append(i["tot_rentals"])

        tot_totrentals = sum(value_totrentals)

        cumulative_totrentals = [value_totrentals[0]]
        temp = 0
        for i in range(1,len(value_totrentals)):
            temp = cumulative_totrentals[i-1] + value_totrentals[i]
            cumulative_totrentals.append(temp)

        cumulative_totrentals = np.divide(cumulative_totrentals, tot_totrentals)

        plt.plot(value_id,cumulative_totrentals)
        plt.xscale("log")
        plt.xlabel("Duration [min]")
        plt.ylabel("CDF")

    plt.title(c)
    plt.grid()
    plt.legend(collections)
```

```
    plt.show()

days = [1,2,3,4,5,6,7]
days_labels = ["Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday", "Sunday"]

for city in cities:
    for collection in collections:
        for day in days:
            duration = db.get_collection(collection).aggregate(
                    [
                        { "$match" : {"$and": [ { "city": city },
                                                { "init_date": { "$gte": start } },
                                                { "final_date": { "$lte": end } },
                                              ]
                                    }
                        },
                        {
                            "$project": {
                                "_id": 1,
                                "city": 1,
                                "weekday": { "$isoDayOfWeek": "$init_date" },
                                "duration": { "$divide": [ { "$subtract": ["$final_time",
                                                                           "$init_time"
                                                                          ]
                                                           },
                                                           60
                                                         ]
                                            },
                            }
                        },
                        { "$match" : {"weekday": day}},
                        { "$group": {
                                "_id": "$duration",
                                "tot_rentals": {"$sum": 1}
                        }
                        },
                        { "$sort": {"_id": 1} }
                    ]
                )

            value_id = []
            value_totrentals = []
            for i in list(duration):
                value_id.append(i["_id"])
                value_totrentals.append(i["tot_rentals"])

            tot_totrentals = sum(value_totrentals)

            cumulative_totrentals = [value_totrentals[0]]
            temp = 0
            for i in range(1,len(value_totrentals)):
                temp = cumulative_totrentals[i-1] + value_totrentals[i]
                cumulative_totrentals.append(temp)

            cumulative_totrentals = np.divide(cumulative_totrentals, tot_totrentals)

            plt.plot(value_id,cumulative_totrentals)
            plt.xscale("log")
            plt.xlabel("Duration [min]")
            plt.ylabel("CDF")
```

```
        plt.title(f"{city} - CDF for weekday - {collection}")
        plt.grid()
        plt.legend(days_labels)
        plt.show()
```

B.2 System utilization

```
start = datetime(2017,11,1,0,0,0)
end = datetime(2017,11,30,23,59,59)

cities = ["Milano", "Calgary", "Amsterdam"]
collections = ["PermanentBookings", "PermanentParkings"]

for city in cities:
    for collection in collections:
        cars_per_hour = db.get_collection(collection).aggregate(
            [
                { "$match" : {"$and": [ { "city": city },
                                        { "init_date": { "$gte":start } },
                                        { "final_date": { "$lte":end } },
                                      ]
                              }
                },
                { "$project": {
                        "_id": 1,
                        "city": 1,
                        "date_parts": { "$dateToParts": { "date": "$init_date" } },
                    }
                },
                { "$group": {
                        "_id": {
                          "day": "$date_parts.day",
                          "hour": "$date_parts.hour"
                        },
                        "tot_rentals": {"$sum": 1}
                }
                },
                { "$sort": {"_id": 1} }
            ]
        )

        value_id = []
        value_totrentals = []
        value_id_2 = []
        value_totrentals_2 = []

        for i in list(cars_per_hour):
            if i["_id"]["day"] < 3:
                value_id.append(i["_id"]["day"] + i["_id"]["hour"]/24)
                value_totrentals.append(i["tot_rentals"])
            elif i["_id"]["day"] > 12:
                value_id_2.append(i["_id"]["day"] + i["_id"]["hour"]/24)
                value_totrentals_2.append(i["tot_rentals"])

        plt.plot(value_id,value_totrentals, color="red",label='unfiltered')
        plt.plot(value_id_2,value_totrentals_2, color='red')

        cars_per_hour_filtered = db.get_collection(collection).aggregate(
            [
```

```
        { "$match" : {"$and": [ { "city": city },
                                { "init_date": { "$gte": start } },
                                { "final_date": { "$lte": end } },
                              ]
                    }
        },
        { "$project": {
              "_id": 1,
              "city": 1,
              "moved": { "$cond": [
                 { "$ne": ["$origin_destination", "$undefined"] },
                 { "$ne": [
                    {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
                    {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
                 ]},
                 True
              ]},
              "duration": { "$divide": [ { "$subtract": ["$final_time",
                                                         "$init_time"
                                                        ]
                                         },
                                         60
                                       ]
                          },
              "date_parts": { "$dateToParts": { "date": "$init_date" } },
           }
        },
        { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                                 { "duration": { "$lte": 180 } },
                                 { "moved": True }
                                ]
                     }
        },
        { "$group": {
              "_id": {
                "day": "$date_parts.day",
                "hour": "$date_parts.hour"
              },
              "tot_rentals": {"$sum": 1}
        }
        },
        { "$sort": {"_id": 1} }
    ]
)

value_id = []
value_totrentals = []
value_id_2 = []
value_totrentals_2 = []

for i in list(cars_per_hour_filtered):
    if i["_id"]["day"]<3:
        value_id.append(i["_id"]["day"] + i["_id"]["hour"]/24)
        value_totrentals.append(i["tot_rentals"])

    elif i["_id"]["day"]>12:
        value_id_2.append(i["_id"]["day"] + i["_id"]["hour"]/24)
        value_totrentals_2.append(i["tot_rentals"])

plt.plot(value_id,value_totrentals,color='blue',label='filtered')
plt.plot(value_id_2,value_totrentals_2,color='blue')
```

```
        plt.xticks(np.arange(1, 31, step=1))
        plt.xlabel("Days of November 2017")
        plt.ylabel("Number of " + collection)
        plt.grid()
        plt.legend()
        plt.title(f"Number of {collection} per hour - {city}")
        plt.show()
```

B.3 Rentals statistics analysis

```
start = datetime(2017,11,1,0,0,0)
end = datetime(2017,11,30,23,59,59)

cities = ["Milano", "Calgary", "Amsterdam"]
collections = ["PermanentBookings", "PermanentParkings"]

for city in cities:
    for collection in collections:
    avg_duration_per_day = db.get_collection(collection).aggregate(
            [
                { "$match" : {"$and": [ { "city": city },
                                        { "init_date": { "$gte":start } },
                                        { "final_date": { "$lte":end } },
                                    ]
                            }
                },
                { "$project": {
                    "_id": 1,
                    "city": 1,
                    "moved": { "$cond": [
                        { "$ne": ["$origin_destination", "$undefined"] },
                        { "$ne": [
                            {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
                            {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
                        ]},
                        True
                    ]},
                    "duration": { "$divide": [ { "$subtract": ["$final_time",
                                                               "$init_time"
                                                              ]
                                                },
                                                60
                                              ]
                                },
                    "date_parts": { "$dateToParts": { "date": "$init_date" } },
                }
                },
                { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                                         { "duration": { "$lte": 180 } },
                                         { "moved": True }
                                         ]
                            }
                },
                { "$group": {
                    "_id": "$date_parts.day",
                    "avg_duration": {"$avg": "$duration"},
                    "std_dev": {"$stdDevPop": "$duration"}
                }
                },
```

```
            { "$sort": {"_id": 1} }
        ]
    )

value_id = []
value_duration = []
value_std = []

value_id_2 = []
value_duration_2 = []
value_std_2 = []


for i in list(avg_duration_per_day):
    if i["_id"]<3:
        value_id.append(i["_id"])
        value_duration.append(i["avg_duration"])
        value_std.append(i["std_dev"])
    elif i["_id"]>12:
        value_id_2.append(i["_id"])
        value_duration_2.append(i["avg_duration"])
        value_std_2.append(i["std_dev"])


plt.figure()
plt.plot(value_id,value_duration,color='blue',label='avg_duration')
plt.plot(value_id_2,value_duration_2,color='blue')
plt.plot(value_id,value_std,color='red',label='standard_deviation')
plt.plot(value_id_2,value_std_2,color='red')

median_duration_per_day = db.get_collection(collection).aggregate(
        [
            { "$match" : {"$and": [ { "city": city },
                                    { "init_date": { "$gte":start } },
                                    { "final_date": { "$lte":end } },
                                ]
                        }
            },
            { "$project": {
                "_id": 1,
                "city": 1,
                "moved": { "$cond": [
                    { "$ne": ["$origin_destination", "$undefined"] },
                    { "$ne": [
                        {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
                        {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
                    ]},
                    True,
                ]},
                "duration": { "$divide": [ { "$subtract": ["$final_time",
                                                           "$init_time"
                                                          ]
                                           },
                                           60
                                         ]
                            },
                "date_parts": { "$dateToParts": { "date": "$init_date" } }
            }
            },
            { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                                     { "duration": { "$lte": 180 } },
```

```
                        { "moved": True }
                        ]
                }
},
{ "$group": {
    "_id": "$date_parts.day",
    "count": { "$sum": 1 },
    "values": {
        "$push": "$duration"
    }
}
},
{
"$unwind": "$values"
},
{ "$sort": {"values": 1} },
{
"$project": {
    "_id": 1,
    "count": 1,
    "values": 1,
    "midpoint": {
        "$divide": [
            { "$subtract": [ "$count", 1 ] }, 2
        ]
    }
}
},
{
"$project": {
    "_id": 1,
    "count": 1,
    "values": 1,
    "midpoint": 1,
    "high": {
        "$ceil": "$midpoint"
    },
    "low": {
        "$floor": "$midpoint"
    }
}
},
{
"$group": {
    "_id": "$_id",
    "values": {
        "$push": "$values"
    },
    "high": {
        "$avg": "$high"
    },
    "low": {
        "$avg": "$low"
    }
}
},
{
"$project": {
    "_id": 1,
    "beginValue": {
        "$arrayElemAt": ["$values" , "$high"]
```

```
                } ,
                "endValue": {
                    "$arrayElemAt": ["$values" , "$low"]
                },
                "85_percentile": {"$arrayElemAt": ["$values", {"$floor":
                                    {"$multiply": [0.85,
                                                   {"$size": "$values"}
                                                   ]
                                    }
                                   }
                                  ]
                                 }
            }
            },
            {
            "$project": {
                "_id": 1,
                "median": {
                    "$avg": ["$beginValue" , "$endValue"]
                },
                "85_percentile": 1
            }
            },
            { "$sort": {"_id": 1} }
        ]
    )



value_median = []
value_perc = []
value_median_2 = []
value_perc_2 = []

for i in list(median_duration_per_day):
    if i["_id"]<3:
        value_median.append(i["median"])
        value_perc.append(i["85_percentile"])
    elif i["_id"]>12:
        value_median_2.append(i["median"])
        value_perc_2.append(i["85_percentile"])


plt.plot(value_id,value_median,color='green',label='median')
plt.plot(value_id_2,value_median_2,color='green')
plt.plot(value_id, value_perc, color='orange',label='85th_percentile')
plt.plot(value_id_2, value_perc_2, color='orange')

plt.xticks(np.arange(1, 31, step=1))
plt.xlabel("Days_of_November_2017")
plt.ylabel("Minutes")
plt.grid()
plt.legend(ncol=2)
plt.title("Statistics_for_" + collection + "_per_day")
plt.show()
```

B.4 Rentals location analysis

B.4.1 Parking positions

```
start = datetime(2017,11,1,8,0,0)
```

```
end = datetime(2017,11,1,8,59,59)

parkings_per_hour = db.get_collection("PermanentBookings").aggregate(
        [
        { "$match" : {"$and": [ { "city": "Milano" },
                                { "final_date": { "$lte":end } },
                                ]
                      }
        },
        { "$project": {
                "_id": 1,
                "init_long": {"$arrayElemAt": [ {"$arrayElemAt": [
                    "$origin_destination.coordinates", 0]}, 0
                  ]
                },
                "init_lat": {"$arrayElemAt": [ {"$arrayElemAt": [
                    "$origin_destination.coordinates", 0]}, 1
                  ]
                },
                "final_long": {"$arrayElemAt": [ {"$arrayElemAt": [
                    "$origin_destination.coordinates", 1]}, 0
                  ]
                },
                "final_lat": {"$arrayElemAt": [ {"$arrayElemAt": [
                    "$origin_destination.coordinates", 1]}, 1
                  ]
                },
                "moved": { "$ne": [
                    {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
                    {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
                  ]
                },
                "duration": { "$divide": [ { "$subtract": ["$final_time",
                                                           "$init_time"
                                                          ]
                                           },
                                           60
                                         ]
                            },
            }
        },
        { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                                { "duration": { "$lte": 180 } },
                                { "moved": True }
                                ]
                      }
        },
        ]
    )

with open("./1st_8-9_parkings.csv", "w") as pks:
    pks.write("latitude,longitude")
    for park in (list(parkings_per_hour)):
        pks.write(f"\n{park['init_lat']},{park['init_long']}
                  \n{park['final_lat']},{park['final_long']}")
```

B.4.2 O-D Matrix

```
start = datetime(2017,11,1,9,0,0)
end = datetime(2017,11,1,23,0,0)
```

```
bookings_per_hour = db.get_collection("PermanentBookings").aggregate(
        [
        { "$match" : {"$and": [ { "city": "Milano" },
                                { "init_date": { "$gte":start } },
                                { "final_date": { "$lte":end } },
                                ]
                    }
        },
        { "$project": {
                "_id": 1,
                "init_long": {"$arrayElemAt": [ {"$arrayElemAt": [
                    "$origin_destination.coordinates", 0]}, 0
                  ]
                },
                "init_lat": {"$arrayElemAt": [ {"$arrayElemAt": [
                    "$origin_destination.coordinates", 0]}, 1
                  ]
                },
                "final_long": {"$arrayElemAt": [ {"$arrayElemAt": [
                    "$origin_destination.coordinates", 1]}, 0
                  ]
                },
                "final_lat": {"$arrayElemAt": [ {"$arrayElemAt": [
                    "$origin_destination.coordinates", 1]}, 1
                  ]
                },
                "moved": { "$ne": [
                    {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
                    {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
                  ]
                },
                "duration": { "$divide": [ { "$subtract": ["$final_time",
                                                           "$init_time"
                                                          ]
                                           },
                                           60
                                         ]
                            },
            }
        },
        { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                                { "duration": { "$lte": 180 } },
                                { "moved": True }
                                ]
                    }
        },
        ]
    )

min_max_coord = db.get_collection('PermanentBookings').aggregate(
         [
            { "$match" : {"$and": [ { "city": "Milano" },
                                    { "init_date": { "$gte": start } },
                                    { "final_date": { "$lte": end } },
                                  ]
                        }
            },
            { "$project": {
                    "_id": 1,
                    "loc": 1,
```

```
                "origin": {"$arrayElemAt": [ "$origin_destination.coordinates",
                                             0
                                           ]
                          },
                "destination": {"$arrayElemAt": [ "$origin_destination.coordinates",
                                                  1
                                                ]
                               },
                "moved": { "$ne": [
                  {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
                  {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
                ]
              },
              "duration": { "$divide": [ { "$subtract": ["$final_time",
                                                         "$init_time"
                                                        ]
                                         },
                                         60
                                       ]
                          },
            }
        },
        { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                                 { "duration": { "$lte": 180 } },
                                 { "moved": True }
                                 ]
                     }
        },
        { "$group": {
              "_id": "$date_parts.hour",
              "count": { "$sum": 1 },
              "init_min_long": {"$min": {"$arrayElemAt": ["$origin", 0]}},
              "init_min_lat": {"$min": {"$arrayElemAt": ["$origin", 1]}},
              "final_min_long": {"$min": {"$arrayElemAt": ["$destination", 0]}},
              "final_min_lat": {"$min": {"$arrayElemAt": ["$destination", 1]}},
            }
        },
    ]
  )

for m in list(min_max_coord):
    if (m["init_min_long"] < m["final_min_long"]):
        min_long = m["init_min_long"]
    else:
        min_long = m["final_min_long"]
    if (m["init_min_lat"] < m["final_min_lat"]):
        min_lat = m["init_min_lat"]
    else:
        min_lat = m["final_min_lat"]

bookings_docs = list(bookings_per_hour)

r_earth = 6378 #km
with open("./9_23_bookings.csv", "w") as bks:
    bks.write("origin_zone_lat,origin_zone_long,dest_zone_lat,dest_zone_long")
    for park in bookings_docs:
        init_lat_id = floor((park['init_lat'] - min_lat) /
                            ((0.5 / r_earth) * (180 / pi)))
        init_long_id = floor((park['init_long'] - min_long) /
                             ((0.5 / r_earth) * (180 / pi) /
                              cos(park['init_lat'] * pi/180)))
```

```
        final_lat_id = floor((park['final_lat'] - min_lat) /
                             ((0.5 / r_earth) * (180 / pi)))
        final_long_id = floor((park['final_long'] - min_long) /
                              ((0.5 / r_earth) * (180 / pi) /
                               cos(park['final_lat'] * pi/180)))
        init_zone_lat = min_lat +
                              ((0.25 / r_earth) * (180 / pi)) +
                              init_lat_id * ((0.5 / r_earth) * (180 / pi))
        init_zone_long = min_long +
                               ((0.25 / r_earth) * (180 / pi) /
                                cos(init_zone_lat * pi/180)) +
                               init_long_id *
                               ((0.5 / r_earth) * (180 / pi) /
                                cos(init_zone_lat * pi/180))
        final_zone_lat = min_lat +
                               ((0.25 / r_earth) * (180 / pi)) +
                               final_lat_id * ((0.5 / r_earth) * (180 / pi))
        final_zone_long = min_long +
                                ((0.25 / r_earth) * (180 / pi) /
                                 cos(final_zone_lat * pi/180)) +
                                final_long_id *
                                ((0.5 / r_earth) * (180 / pi) /
                                 cos(final_zone_lat * pi/180))
        bks.write(f"\n{init_zone_lat},{init_zone_long},
                    {final_zone_lat},{final_zone_long}")
    bks.close()

df = pd.read_csv('./9_23_bookings.csv')

df_grouped = df.groupby(['origin_zone_lat', 'origin_zone_long', 'dest_zone_lat',
                         'dest_zone_long']).size().reset_index(name='count')
print(df_grouped)

coords_list = []
with open('heatmap.csv', 'w') as hm:
    hm.write("coords,origin_id")
    new_id = 0
    found = False
    for i in range(len(df_grouped)):
        coords = f"{df_grouped['origin_zone_long'][i]}-
                    {df_grouped['origin_zone_lat'][i]}"
        for j in range(len(coords_list)):
            origin_coords = coords_list[j][0]
            if origin_coords == coords:
              id = coords_list[j][1]
              found = True
              break
        if not found:
          id = new_id
          new_id += 1
          coords_list.append((coords,id))
        hm.write(f"\n{coords},{id}")
        found = False
    hm.close()

hm_df = pd.read_csv('./heatmap.csv')
with open('heatmap_new.csv', 'w') as hm:
    hm.write("origin_id,dest_id,count")
    id = len(hm_df)
    new_id = len(hm_df)
    for i in range(len(df_grouped)):
```

```
        dest_coords = f"{df_grouped['dest_zone_long'][i]}-
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{df_grouped['dest_zone_lat'][i]}"
        for j in range(len(hm_df)):
            origin_coords = hm_df['coords'][j]
            if origin_coords == dest_coords:
              id = hm_df['origin_id'][j]
              break
        if id >= len(hm_df):
          id = new_id
          new_id += 1
        hm.write(f"\n{hm_df['origin_id'][i]},{id},{df_grouped['count'][i]}")
    hm.close()

df_grouped.index = np.arange(1, len(df_grouped)+1)
df_grouped.to_csv('./filtered_9_23_bookings_grouped.csv', index_label='OID')

hm_df = pd.read_csv('./heatmap_new.csv')

if hm_df['origin_id'].max() > hm_df['dest_id'].max():
  size = hm_df['origin_id'].max() + 1
else:
  size = hm_df['dest_id'].max() + 1

max_count = hm_df['count'].max()

hm = np.zeros((size,size), dtype=int)

for i in range(len(hm_df)):
  hm[hm_df['origin_id'][i],hm_df['dest_id'][i]] += hm_df['count'][i]

sns.set()
fig = plt.figure(figsize=(12,11), dpi= 100)
ax = sns.heatmap(hm, vmin=0, vmax=max_count)
plt.xlabel('Origin␣ID')
plt.ylabel('Destination␣ID')
plt.show()
```

B.5 Optional Task

```
start = datetime(2017,11,1,0,0,0) #1 novembre 2017
end = datetime(2017,11,30,23,59,59)
alternative_transp_filtered = db.get_collection('PermanentBookings').aggregate(
[
  { "$match" : {"$and": [ { "city": "Milano" },
                          { "init_date": { "$gte": start } },
                          { "final_date": { "$lte": end } },
                          { "$or":  [ {"walking.duration":{"$ne":-1} },
                                      {"driving.duration":{"$ne":-1} },
                                      {"public_transport.duration":{"$ne":-1} }
                                      ]
                            }
                        ]
              }
  },
  { "$project": {
        "_id": 1,
        "city": 1,
        "moved": { "$ne": [
              {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
              {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
```

```
            ]
        },
        "duration": { "$divide":[{"$subtract":["$final_time","$init_time"]},60]},
        "alternative_duration_tr":{"$divide":["$public_transport.duration",60]}
      }
  },
  { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                           { "duration": { "$lte": 180 } },
                           { "moved": True },
                           { "alternative_duration_tr": {"$gte": 0} }
                         ]
               }
  },
  { "$sort": {"alternative_duration_tr": 1} }
]
  )


alt_tra_filt = list(alternative_transp_filtered)

alternative_duration_pt = []
for ad in alt_tra_filt:
    alternative_duration_pt.append(ad["alternative_duration_tr"])

max_ = myround(alternative_duration_pt[-1],5)
x = np.arange(0,max_+1,step=5)
plt.figure()
plt.grid()
plt.xlabel("Public Transport Duration [min]")
plt.ylabel("Number of rentals")
plt.title("Number of rentals VS alternative public transport duration")
n, bins, patches = plt.hist(alternative_duration_pt, bins=x)

#%% Walking
alternative_transp_filtered = db.get_collection('PermanentBookings').aggregate(
[
  { "$match" : {"$and": [ { "city": "Milano" },
                          { "init_date": { "$gte": start } },
                          { "final_date": { "$lte": end } },
                          { "$or":  [ {"walking.duration":{"$ne":-1} },
                                      {"driving.duration":{"$ne":-1} },
                                      {"public_transport.duration":{"$ne":-1} }
                                      ]
                            }
                        ]
              }
  },
  { "$project": {
        "_id": 1,
        "city": 1,
        "moved": { "$ne": [
              {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
              {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
            ]
        },
        "duration": { "$divide":[{"$subtract":["$final_time","$init_time"]},60]},
        "alternative_duration_tr":{"$divide":["$walking.duration",60] }
      }
  },
  { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                           { "duration": { "$lte": 180 } },
```

```
                                { "moved": True },
                                { "alternative_duration_tr": {"$gte": 1} }
                            ]
                    }
    },
    { "$sort": {"alternative_duration_tr": 1} }
]
    )


alt_tra_filt = list(alternative_transp_filtered)

alternative_duration_walking = []
for ad in alt_tra_filt:
    alternative_duration_walking.append(ad["alternative_duration_tr"])

max_ = myround(alternative_duration_walking[-1],5)
x = np.arange(0,max_+1,step=5)
plt.figure()
plt.grid()
plt.xlabel("Walking Duration [min]")
plt.ylabel("Number of rentals")
plt.title("Number of rentals VS alternative walking duration")
plt.hist(alternative_duration_walking, bins=x)
```

#%%Driving

```
alternative_transp_filtered = db.get_collection('PermanentBookings').aggregate(
[
    { "$match" : {"$and": [ { "city": "Milano" },
                            { "init_date": { "$gte": start } },
                            { "final_date": { "$lte": end } },
                            { "$or":  [ {"walking.duration":{"$ne":-1} },
                                        {"driving.duration":{"$ne":-1} },
                                        {"public_transport.duration":{"$ne":-1} }
                                        ]
                            }
                          ]
                }
    },
    { "$project": {
            "_id": 1,
            "city": 1,
            "moved": { "$ne": [
                    {"$arrayElemAt": [ "$origin_destination.coordinates", 0]},
                    {"$arrayElemAt": [ "$origin_destination.coordinates", 1]}
                ]
            },
            "duration": { "$divide":[{"$subtract":["$final_time","$init_time"]},60]},
            "alternative_duration_tr":{"$divide":["$driving.duration",60]}
        }
    },
    { "$match" : { "$and": [ { "duration": { "$gte": 3 } },
                            { "duration": { "$lte": 180 } },
                            { "moved": True },
                            { "alternative_duration_tr": {"$gte": 1} }
                          ]
                }
```

```
  },
  { "$sort": {"alternative_duration_tr": 1} }
]
  )


alt_tra_filt = list(alternative_transp_filtered)

alternative_duration_driving = []
for ad in alt_tra_filt:
    alternative_duration_driving.append(ad["alternative_duration_tr"])

max_ = myround(alternative_duration_driving[-1],5)
x = np.arange(0,max_+1,step=5)
plt.figure()
plt.grid()
plt.xlabel("Driving Duration [min]")
plt.ylabel("Number of rentals")
plt.title("Number of rentals VS alternative driving duration")
plt.hist(alternative_duration_driving, bins=x)
```