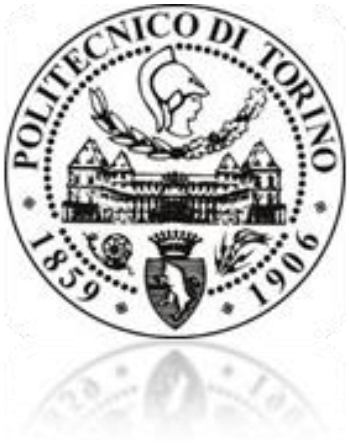


ICT in Transport Systems

Professor: Marco Mellia

Pinky Kumawat (s264813)
Mersedeh kooshki (s250399)
Zahra Arshadi (s246592)

2018/2019

Laboratory 1: Carsharing Analytics

Introduction

This laboratory considers Carsharing data collected from some websites like Car2go and Enjoy in Italy, and made available through MongoDB database. The goal is to work on real data, trying to extract useful information specific to the application domain (Transport Systems) and get used data pre-processing and filtering.

Step1-Preliminary data analysis

In the first step, we worked with the access of the data using MongoDB and made some analysis to investigate the collections. Carsharing system is divided into four collections for Car2go and Enjoy which are updated in real time: ActiveBookings, ActiveParkings, PermanentBookings, and PermanentParkings.

How many documents are present in each collection?

Car2go		Enjoy	
collections	**Number of document**	**collections**	**Number of document**
ActiveBooking	8743	ActiveBookings	1025
ActiveParkings	4790	ActiveParkings	1867
PermanentBookings	28180508	PermanentBookings	5264764
PermanentParkings	28312676	PermanentParkings	5297253

Why the number of documents in PermanentParkings and PermanentBooking is similar?

Actually they are related to each other, we know that the number of cars is constant so when a car is booked we have expected to park it. It is obvious that some missing values will happen so the numbers are not completely matched.

For which cities the system is collecting data?

"Amsterdam","Austin","Berlin","Calgary","Columbus","Denver","Firenze","Frankfurt","Hamburg","Madrid","Milano","Montreal","Munchen", "New York City", "Portland" ,"Rhineland" ,"Roma", "Seattle", "Stuttgart", "Torino", "Toronto", "Vancouver", "Washington DC", Wien".

If we try for "PermanentBookings" the result has two cities more than "ActiveBookings": "San Diego", "Twin Cities". Hence, totally the system is collecting data for 26 cities.

When the collection started? When the collection ended?

The result: **0** which is for Vancouver and unixtime is **2016-12-13 09:37:38.000Z**, and ended on **2019-01-02T19:11:27Z** at Catania. We have to write query for "Activebookings" collection to get the correct answer, by the way it is changing by time passes.

What about the timezone of the timestamps?

- **Timestamps**: It is a server time which system uses to save data on the server.
- **Timezone**: It is a local time which is different for each cities but it is human readable.

Now we have to consider each city of our group (Torino and Firenze):

How many cars are available in each city?

We wrote query on the following collections for each cities and then we sum up the values to get the result. In order to avoid repetitive cars we didn't use PermanentBookings or Parkings.

City	ActiveBookings	ActiveParkings	enjoy_ ActiveBookings	enjoy_ ActiveParkings	Sum
Torino	101	312	249	174	836
Firenze	---	---	20	75	95

How many bookings have been recorded on December 2017 in each city?

City	Car2go	Enjoy
Torino	93800	61680
Firenze	----	14017

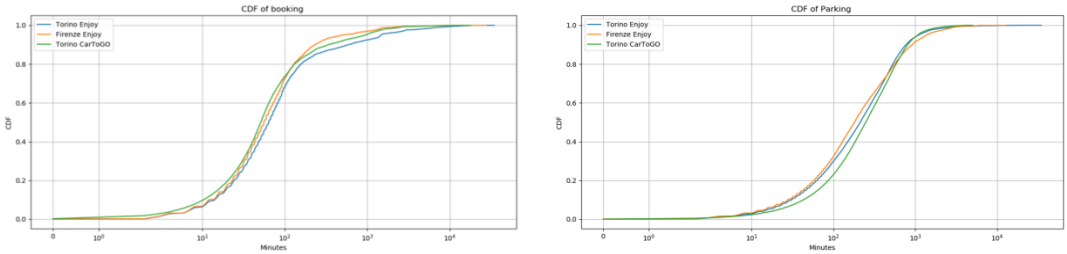
How many bookings have also the alternative transportation modes recorded in each city?

City	Car2go	Enjoy
Torino	873240	926662
Firenze	----	271314

Step2-Analysis of data

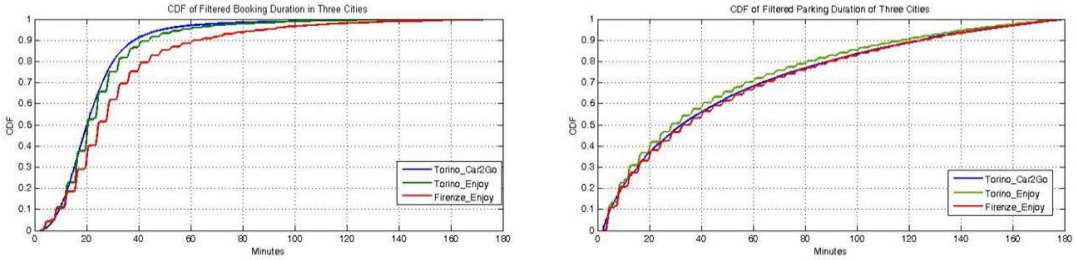
By considering each city of our group, period of time October 2017, and time series (city, timestamp, duration, locations), we were asked to do further analysis and displayed the results by plot.

Task1- Cumulative Distribution Function of Booking/Parking duration



(a) CDF plot of non-filtered booking duration (b) CDF plot of non-filtered parking duration

Figure 1.1: CDF of Booking and Parking Duration (Non Filtered)



(a) CDF plot of Filtered Booking Duration (b) CDF plot of Filtered Parking Duration

Figure 1.2: CDF of Booking and Parking Duration (Filtered)

The above figure 1.1 shows the CDF relative to booking and parking duration (minutes with log scale) without applying filter (on duration and move). It can be observed that on average the booking duration is high and some bookings lasts for more than 160 minutes that can be considered as outliers. Comparing the above CDF plots for all three situations displays: Torino (Car2Go) has longer CDF than two others, and Firenze (Enjoy) has smallest one. This means the cars were booked for longer period in Torino (Car2Go) and comparatively for smaller period in Firenze, one of the expected reasons of this difference can be the different size of territorial and urban areas in these cities while the other reason can be car service availability. For Parking, the CDF behavior of all three situations is quite similar and different from bookings. In addition, the parking duration seems longer (lasts 180 minutes) that means some cars were parked for long time and not used. Comparing all three scenarios Torino (Enjoy) parking duration is more and Firenze (Enjoy) has less parking duration. The presence of outliers affects the curves, which can be clearly seen from the plots in figure 1.2 (filtered one) by comparing it with the pervious figure (1.1).

Point (c) - from below figures, it is clearly visible that the CDF is quite similar and does not change over time.

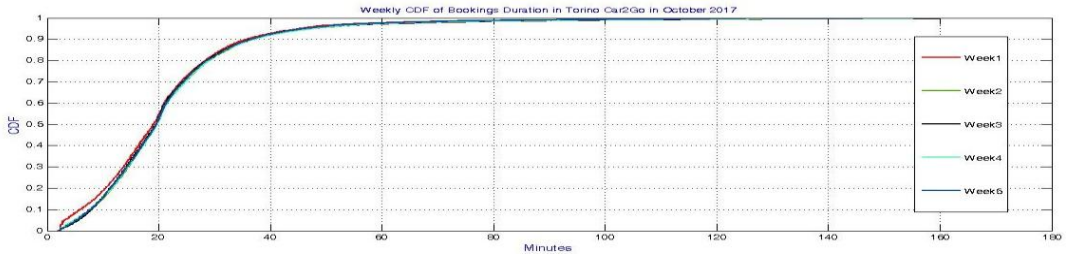


Figure 1.3: CDF of Booking Duration per each week of data

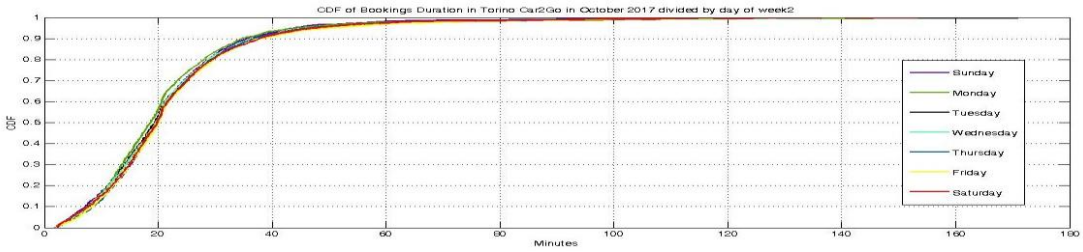
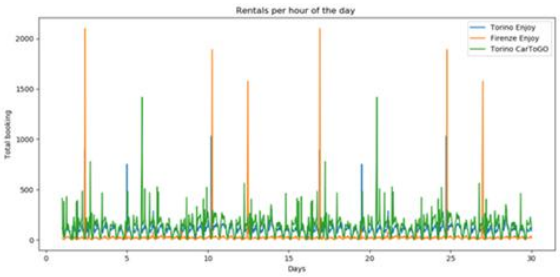


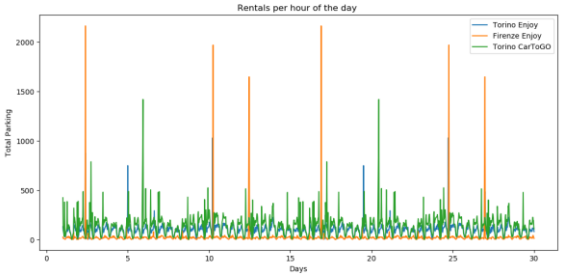
Figure 1.4: CDF of Booking Duration per each day of week

Task2 - System utilization over time: without filtering data

In this task, we have to estimate booking and parking for every days of the month and every hour of a day during chosen period time, based on the cities of our group. As you can see in the below figures 2.1 there are some unusual peaks which belongs to Firenze-enjoy for some special days while almost the other days follow the same pattern. On average, we have one strange peak in each week of the month that can be due to the outliers or some



(a) Bookings per hour of the days of the month

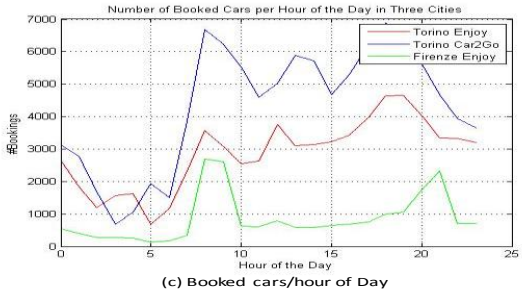


(b) Parkings per hour of the days of month

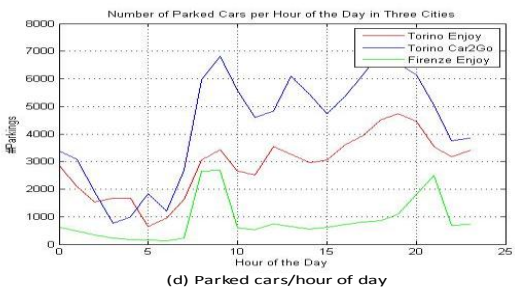
difficulty in enjoy server.

Figure 2.1: Booked/Parked Cars per hour of the days of the month (Non Filtered)

From the figure 2.2, it can be observed that the cars were booked more during peak hours (morning 6-10 and evening 16-20) of the days. In addition, there is quite similarity between the graph of booked and parked cars. The more number of booked and parked cars are specified for the city Torino (Car2Go) and in two other cities comparatively found less. Some peaks are notified in the graphs because of outliers/system errors, which are filtered in task 4.



(c) Booked cars/hour of Day



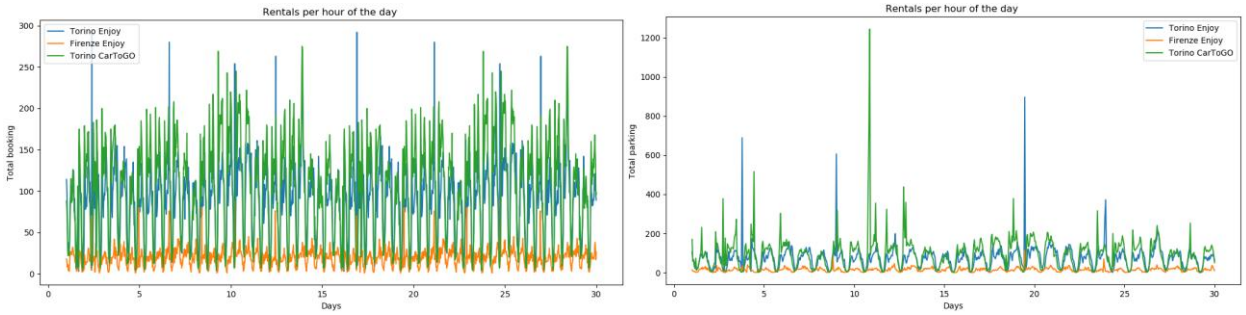
(d) Parked cars/hour of day

Figure 2.2: Rentals per hour of the day (Non Filtered)

Task 3 – Filtering Criteria

We applied two filtering criteria to filter possible outliers on the booking and parking that is already discussed in task 1. One filter is applied on the movement, where we discarded the cars that did not moved by comparing the origin and destination coordinates which should not be equal to satisfy the movement condition. Other filter criterion is duration where we discarded the duration less than 2 minutes and greater than 180 minutes. The possibility of duration less than 2 minutes can be that the bookings may be cancelled and the duration more than 180 minutes possibly can be result of system error or may be some critical situations like repairing/accidents/maintenance.

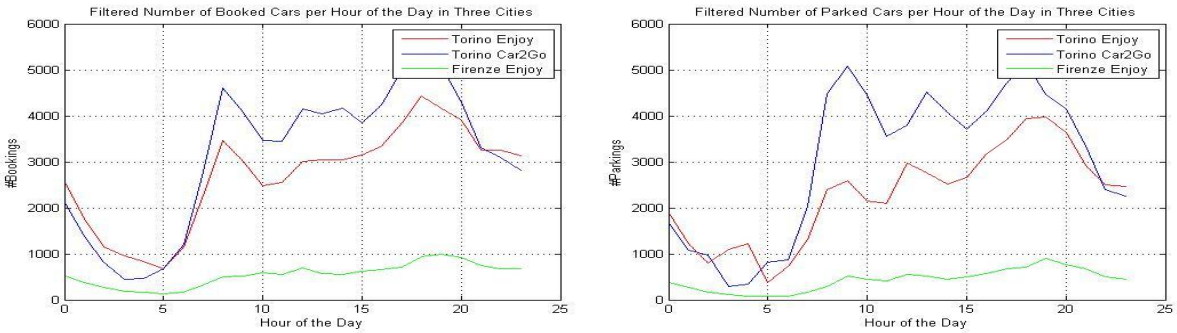
Task4 – System Utilization over time: with filtering data



(a) Filtered Bookings per hour of the days of the month

(b) Filtered Parkings per hour of the days of the month

Figure 4.1: Booked/Parked Cars per hour of the days of the month (Filtered)



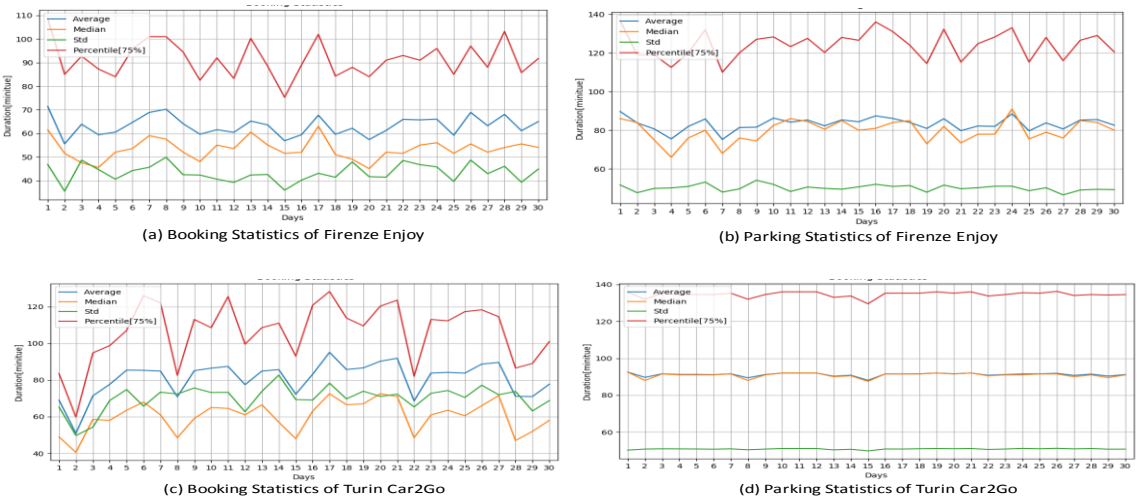
(c) Filtered Booked cars/ hour of day

(d) Filtered Parked cars/hour of day

Figure 4.2: Rentals per hour of the day (Filtered)

Considering same data in task 2, we applied filter on that to remove the outliers. Although the graph pattern is quite similar but the height of peaks decreased now because of filtering outliers, where we can observe the actual number of booked and parked cars.

Task5 – Average, Median, Standard Deviation, and Percentiles of the Booking/Parking duration over time

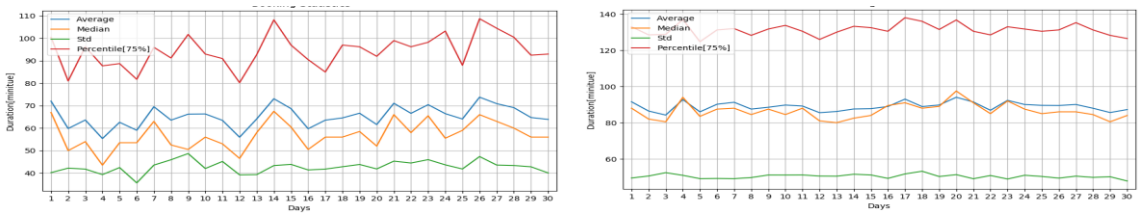


(a) Booking Statistics of Firenze Enjoy

(b) Parking Statistics of Firenze Enjoy

(c) Booking Statistics of Turin Car2Go

(d) Parking Statistics of Turin Car2Go



(e) Booking Statistics of Turin Enjoy (f) Parking Statistics of Turin Enjoy

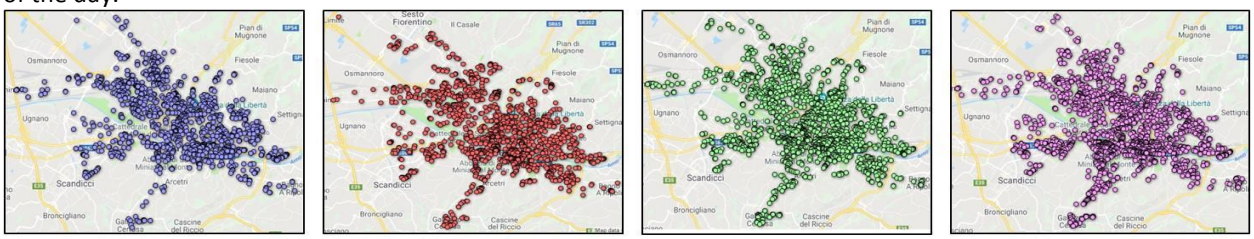
Figure 5: Statistics of Booking and Parking duration over time

In this task by applying filter on data we computed average, median, standard deviation, and percentiles (75th) of the booking and parking duration over time per each day of our period time. The booking statistics change over time but the parking statistics not changing more. Mostly the fluctuations in the peaks are on weekdays when people moving more for their trips' purposes (work, study, etc). Comparing all average car bookings, people use more car2Go service in Turin rather than in Turin (Enjoy) and Firenze (Enjoy). Also the percentile (75th) of Turin (Car2Go) fluctuates more and notified some peaks - upwards and falls, the upper peaks shows that cars are booked more than 120 minutes (on days 6th, 7th, 11th, 17th, 21th) and the fall is (lowest duration) 60 minutes. Also for booking statistics the dispersion among duration for Torino (Car2Go) is less by comparing two other situations. From parking statistics it can be observed that almost average parking duration in all the three scenarios are equal (around more than 80 minutes) also superimposing on median line.

Task6 – Parking position of cars and OD visualization

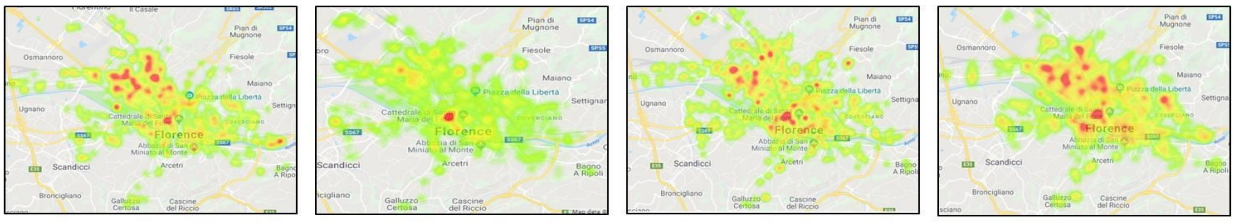
For this task, we consider the parking position of cars of Enjoy in Firenze.

Step (a) - To plot the parking position of cars, we computed the density of cars during 4 time slot division of hours of the day.



(a) 0-5 hours (b) 6-11 hours (c) 12-17 hours (d) 18-23 hours

Figure 6.1: Parking position of cars in different times



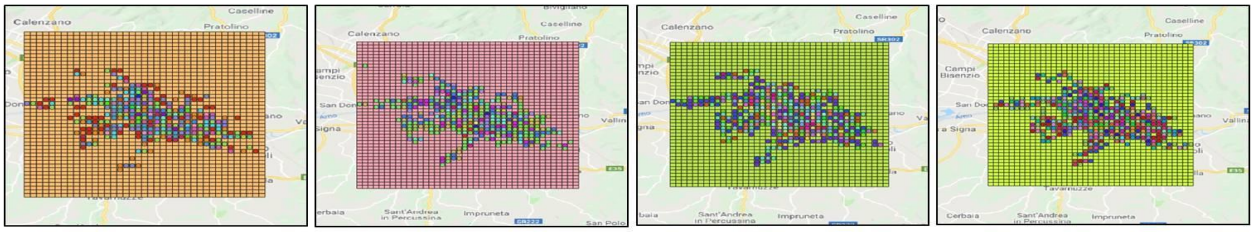
(a) 0-5 hours (b) 6-11 hours (c) 12-17 hours (d) 18-23 hours

Figure 6.2: Parking position of cars in different times (Heatmap)

Above figures represents that during period 0-5 hours less cars are spread and during 6-11 hours cars are more scattered because people started moving to fulfill their purpose ('work', 'study', 'leisure', etc.) and mostly hotspots are seen around 'Cattedrale di Santa' and 'Maria del Fiore'. During afternoon (12-17 hours) cars are also scattered and many hotspots are seen around the city. In evening and night (18-23 hour) cars are less scattered and the parking density is high at some hotspots showing red in figure (d) because people parked the cars and returned to home or doing some rest.

Step (b) – In this we divide the Firenze city area into a simple squared grid of 500m*500m to compute the density of cars in each area by assigning different color to each square during different times. More dense areas were

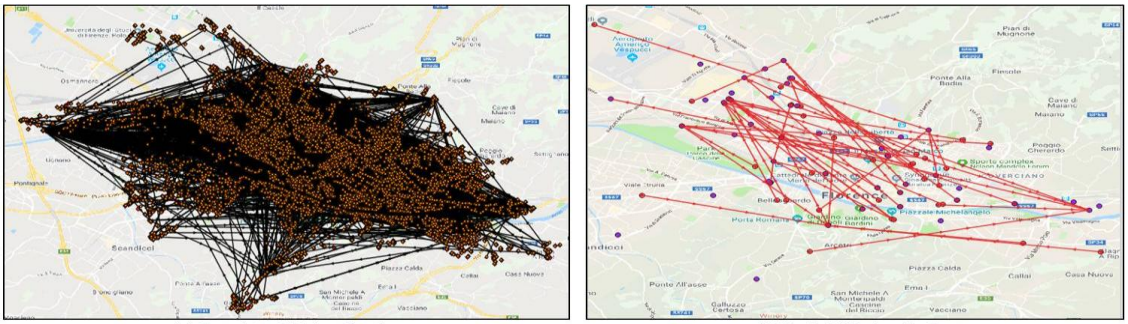
recorded around the city centre during peak hours 6-11 and 12-17, mostly around 'Cattedrale di Santa' and 'Maria del Fiore'.



(a) 0-5 hours (b) 6-11 hours (c) 12-17 hours (d) 18-23 hours

Figure 6.3: Position of parked cars with squared grid of 500m*500m in different time

Step (c) – In this we computed the OD matrix and visualize the origin-destination points using QGIS. We also take a sample of some origin points from a square and some destination points from other square to clearly visualize the origin and destination matrix and rentals in that areas.

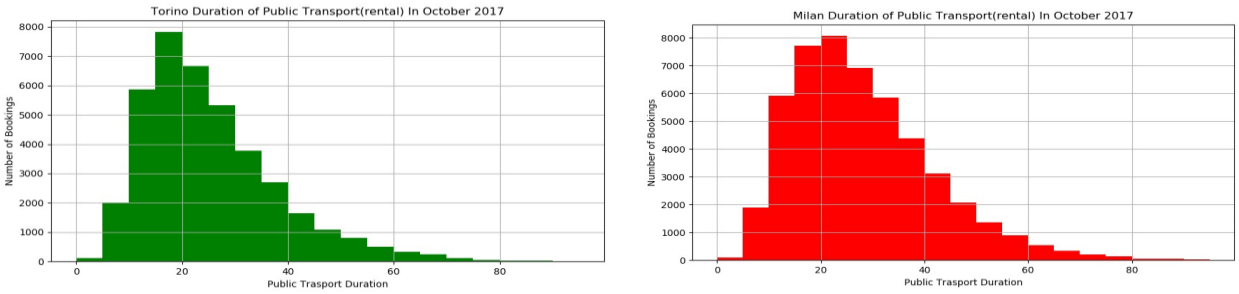


(a) Complete OD Visualization (b) Partial OD visualization

Figure 6.4: OD Visualization

Task7 – Correlation of the probability of a rental with the availability of other transport means

In this task, we extract valid rentals while there is also the data for alternative transport means (public transport). For public transport systems, we divided the duration into bins of 5 minutes, computed the rentals for each bin, and plotted them to show the correlation of the probability of a rental with the availability of public transport. In the figure 7.1 the number of rentals increase when public transport duration is between 15 till 25 minutes, and for the time<15 and time>25 minutes the number of rentals is reduced which means for the low duration or long duration people prefer do not use the car rental services.



(c) Rentals with public transport (Torino) (d) Rentals with public transport (Torino)

Figure 7.1: Histogram of Rentals with alternative Transport option – Public Transport

Laboratory 2: Most interested in using car sharing

Introduction

In the second laboratory we want to do a survey on car sharing's users and for this reason we do a study based on the data analyzed during the first laboratory. The goal is to understand which are the most likely type of users of car sharing system.

IMQ dataset:

IMQ is our dataset which stored in MongoDB and it contains data collected with phone interviews, which is the 2 months of rental data for Enjoy and Car2Go with Origin/Destination, and this dataset Covers 185 zones of Piedmont and we only consider 23 zones which are related to Turin. This dataset consist trips from Monday to Friday and users' information:

1) Gender (male/female)
2) Age (11-19, 20-49, 50-64, 65+)
3) Motivation (Go to work, working reason, study, shopping, bring someone, cures or medical visit, sport and leisure, going back home, visiting relatives or friends, others, going back home on the day of interview)

Methodology:

The first step is to create OD matrixes based on Carsharing dataset, for car2go, enjoy, both (Car2Go+Enjoy) and during this step we divided dataset into some proper subset which are all day (all hours), weekend (all hours), weekday morning (from 0 to 12) and afternoon (from 16 to 20) and at the end we have 12 different OD matrixes extracted. For next step we tried to calculate the distances between each of our scenarios and the IMQ matrix but before doing this we normalized our data in OD matrixes by dividing each cells with the sum of all the trips. Finally for each matrix we estimate the distances based on the following formula:

$$d(A,B) = \sum_{i=0}^{n}\sum_{j=0}^{n} |a_{ij} - b_{ij}|$$

Equation 1- Distance Equation

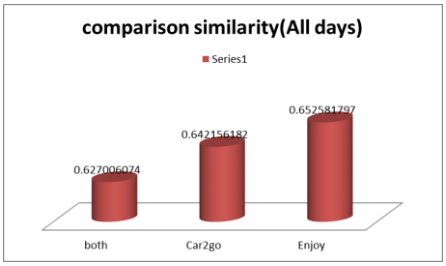
Some important aspects that we considered are:

- We removed age more than 65 years from our survey because of lowest similarity.
- We considered all hours of a day for all day and weekend scenarios.
- We decided to display all 11 motivations in one column as all and then mention only to the motivations which can be more useful in the case of comparison and we combined these two motivation (Go to work and, Working reason) as a one and put it in work column.

First scenario: All days

Gender			Age				Motivation							Result		
All	Male	Female	All	11 to 19	20 to 49	50 to 64	All	work	study	Cures or medical visit	Sport and leisure	Going back home	Other	Car2go	enjoy	All
*			*				*							0.667043448	0.68553428	0.654752737
	*		*				*							0.658336323	0.71218488	0.647287363
		*	*				*							0.686256895	0.74473831	0.713389724
	*				*	*	*							0.649038054	0.66564575	0.640598552
*					*	*	*							0.667106679	0.69020062	0.659240031
*					*	*		*	*	*	*	*	*	0.642622273	0.65931226	0.629185474
*					*	*		*	*	*	*	*		0.642462847	0.65911563	0.629548461
*			*					*	*	*	*	*		0.642156182	0.65460836	0.627006074
*			*					*	*	*		*		0.650560095	0.65258188	0.630789374

As you can see in the table above, we considered different scenarios based on our chosen criteria, filtering for all days of a week (both weekday and weekend) and we calculated the distance of each matrix with IMQ OD matrix to figure out their similarities. The values as the results display the difference level for each service (car2go, enjoy and both) with IMQ data compared to the others. Since the smallest outcome value will be due to the more similarity, so obviously the values with red color are represented most resemblance scenarios in every services with IMQ data. In addition, by comparing the best results of each column (each service and both) together, we will understand that best matching is belongs to both services together.



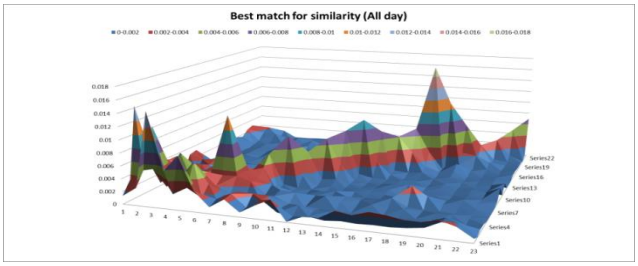


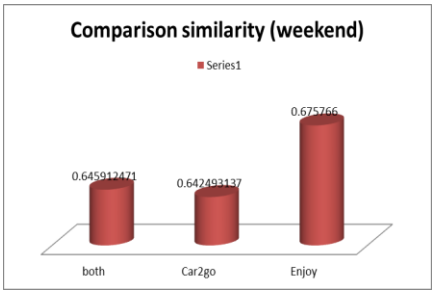
Figure1-a) comparison similarity for all day **Figure1-b) best match for similarity**

The figure1-a shows distance of optimal match from the IMQ OD matrix. From our point of view, the least similarity is for the situations with all motivations and on average for enjoy service this similarity is less than car2go service. To make it more clear, we tried to show you the comparison among three categories (car2go, Enjoy and both) by the figure so as you can see the highest similarity is belongs to both services together and enjoy service has the lowest similarity with IMQ OD matrix. It should be noted that this result is only belongs to all day survey.

Second Scenario: Weekend

Gender			Age				Motivation							Result		
All	Male	Female	All	11 to 19	20 to 49	50 to 64	All	work	study	Cures or medical visit	Sport and leisure	Going back home	Other	Car2go	enjoy	All
*			*				*							0.668131192	0.704797798	0.671275018
	*		*				*							0.663539050	0.696549006	0.665414867
		*	*				*							0.720605391	0.764305232	0.729346028
	*				*	*	*							0.657425741	0.693590474	0.661712828
*					*	*	*							0.670499858	0.709333691	0.676674125
*					*	*		*	*	*	*	*	*	0.643469827	0.680787039	0.647782097
*					*	*		*	*	*	*	*		0.644252833	0.680268522	0.647885041
*			*					*	*	*	*	*		0.642493137	0.677085401	0.645912471
*			*					*	*	*		*		0.646491431	0.675766988	0.647792065

In the second table, we made a study with different scenarios only for weekend days and observed some changes comparing the results of all days. This time best matching with IMQ OD matrix is for car2go service in the first column and enjoy has the high differences with IMQ matrix in comparison with two others.



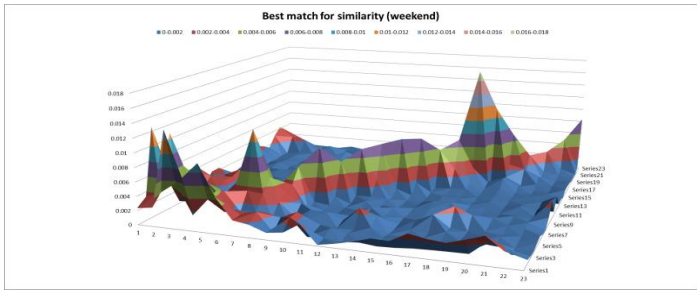


Figure 2-a) comparison similarity for weekend **Figure 2-b) best match for similarity**

The figure 2-a shows distance of optimal match from the IMQ OD matrix. It is obvious that still the least similarity is for the situations with all motivations and on average for Enjoy service this similarity is less than the two other ones. Actually, highest similarity is belongs to car2go service and enjoy service has the lowest similarity with IMQ OD matrix. It should be noted that this result is only belongs to weekend survey.

Third Scenario: Weekday morning

Gender			Age				Motivation							Result		
All	Male	Female	All	11 to 19	20 to 49	50 to 64	All	work	study	Cures or medical visit	Sport and leisure	Going back home	Other	Car2go	enjoy	All
*			*				*							0.733683275	0.727907225	0.702301497
	*		*				*							0.732780876	0.701735873	0.691798651
		*	*				*							0.783579212	0.786755053	0.754447026
	*				*	*	*							0.723312767	0.694326751	0.681197663
*					*	*	*							0.734903298	0.733607981	0.704260652
*					*	*		*	*	*	*	*	*	0.713185002	0.690778792	0.678369268
*					*	*		*	*	*	*	*		0.713208209	0.691390852	0.678410687
*			*					*	*	*	*	*		0.712410183	0.700163178	0.678322216
*			*					*	*	*		*		0.722830956	0.701118879	0.687096321

The third table displays different scenarios during weekdays-morning and the main point here is the increase in value of the results in all the services and scenarios rather than two situations before (all days and weekend) which shows highest differences with IMQ OD matrix compare to previous ones. Anyway, in this table best matching is for both services together (car2go and enjoy) and car2go has the less similarity with IMQ OD matrix in comparison with two others.

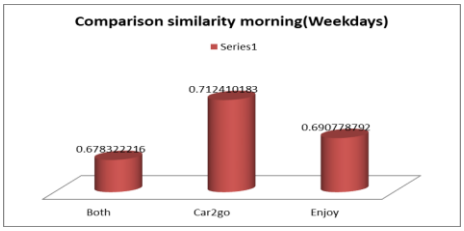


Figure 3-a) comparison similarity for weekday

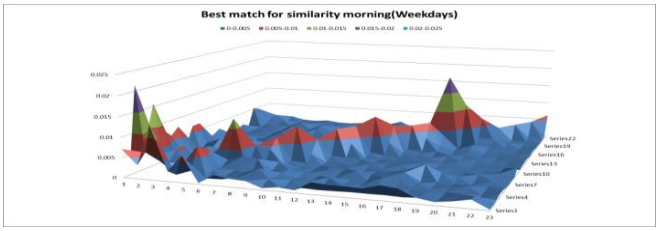


Figure 3-b) best match for similarity (weekday)

The figure 3-a shows distance of optimal match from the IMQ OD matrix. It is obvious that still the least similarity is for the situations with all motivations and on average for car2go service, this similarity is less than the two other ones. The range of numbers in this plot shows the increment of differences in the result value about matching with OD matrix in IMQ dataset. The same as other figures in other situations we described before this figure also shows comparison similarity but for weekday- morning. Highest similarity is belongs to both services together and car2go service has the lowest similarity with IMQ OD matrix.

Forth scenario: Weekday afternoon

Gender			Age				Motivation							Result		
All	Male	Female	All	11 to 19	20 to 49	50 to 64	All	work	study	Cures or medical visit	Sport and leisure	Going back home	Other	Car2go	enjoy	All
*			*				*							0.666678144	0.705373699	0.654036951
	*		*				*							0.661927848	0.689866875	0.64546039
		*	*				*							0.721538227	0.765976012	0.712000245
	*				*	*	*							0.664181848	0.692482843	0.648513603
*					*	*	*							0.674335088	0.713825745	0.662477033
*					*	*		*	*	*	*	*	*	0.64846169	0.681541332	0.636180727
*					*	*		*	*	*	*	*		0.649055481	0.681857222	0.636505236
*			*					*	*	*	*	*		0.644856979	0.672704905	0.628961532
*			*					*	*	*		*		0.640858151	0.665874885	0.627043061

Actually, in this table best matching is for both services together (car2go and enjoy) and enjoy has the less similarity with IMQ OD matrix in comparison with two others. By comparing weekday morning and afternoon, we understand that weekday afternoon has fewer differences with IMQ data compared to weekday morning.

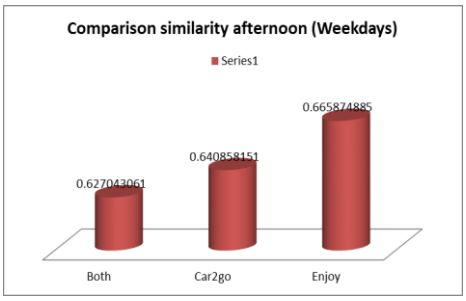


Figure 4-a) comparison similarity for weekday

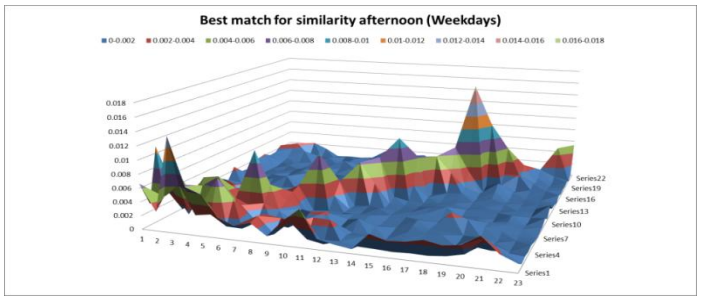


Figure 4-b) best match for similarity (weekday

The final plot (Figure 4-a) shows distance of optimal match from the IMQ OD matrix. For enjoy service, this similarity is less than the two other ones while both services has the best matching with OD matrix in IMQ dataset. Final figure 4-b, which is comparison similarity for weekday- afternoon shows that Highest similarity is belongs to both services together and enjoy service has the lowest similarity with IMQ OD matrix.

Conclusion

In this laboratory we compared OD matrixes that come from two different data sets and according to differences between them, we concluded that:

- Female has less similarity comparing men because of higher differences.
- People in range from 11 to 64 get the best fit according to differences.
- The main purposes to take the trips are: Go to work, Working reason, Study, Cures or medical visit, Sport and leisure, Going back home, others.
- By comparing morning and afternoon, sport purpose gets less similarity in the afternoon toward to morning situation.

Finally we should notice that IMQ dataset is of year 2013 (5 years ago) and it is can be possible that we find better results with latest data.

Code Appendix

Step 1 – Preliminary data analysis

1.1 - How many documents are present in each collection?

```
db.collection.find().count()
```

1.3 - For which cities the system is collecting data?

```
db.ActiveBookings.distinct(“city”)
```

1.4 - When the collection started? When the collection ended?

```
db.collection.find({},{init_date:1,_id:0}).sort({init_date:1}).limit(1)
db.collection.find({},{final_date:1,_id:0}).sort({final_date:-1}).limit(1)
```

1.6 - How many cars are available in each city?

```
db.collection.distinct("plate", {city: "Torino"}).length
```

1.7 - How many bookings have been recorded on the December 2017 in each city?

```
db.collection.find({init_date: {"$gte":new Date("12/25/2017"), "$lt":new Date("12/26/2017")}, city:
"Torino"}).count()
```

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

```
db.collection.find({city:"Torino", "public_transport.distance": {$ne: -1}}).count()
```

Step 2 – Analysis of the data

Task – 1

***CDF: Non Filtered**

```
var startDate = new ISODate("2017-10-01T00:00:00") //consider starting time of october 2017
var endDate = new ISODate("2017-10-31T23:59:59") //consider ending time of october 2017 in variable endDate
var results = db.collection.aggregate([
  {
   $match: { //applying filter to select what fields we need, here the cities and duration of october 2017
          $or: [{city:"Torino"}],
          init_date: {$gte: startDate, $lte: endDate }
         }
  },
  {
    $project: {//extract position,time and duration of parking
          _id: 0,
          city: 1,
          moved: {$ne:[
                {$arrayElemAt: ["$origin_destination.coordinates", 0]},//returns the element at the specified array
index
                {$arrayElemAt: ["$origin_destination.coordinates", 1]}//
               ]
            },

          duration: {$divide: [{$subtract: ["$final_time","$init_time"]},60 ]},//calculating duration of bookings in
minutes
          init_time:1,
          final_time:1
          }
  },
  {
    $sort: {
          "id.city":1,
          "duration":1
         }
  }
```

```
  ])
  while (results.hasNext())
  {
     var a = results.next()
    print(a["city"],a["duration"])
  }
```

***CDF: Filtered**

```
var startDate = new ISODate("2017-10-01T00:00:00") //consider starting time of october 2017
var endDate = new ISODate("2017-10-31T23:59:59") //consider ending time of october 2017 in variable endDate
var results = db.collection.aggregate([
  {
     $match: { //applying filter to select what fields we need, here the cities and duration of october 2017
           $or: [{city:"Torino"}],
           init_date: {$gte: startDate, $lte: endDate }
          }
  },
  {
     $project: {//extract position,time and duration of parking
           _id: 0,
           city: 1,
           moved: {$ne:[
                 {$arrayElemAt: ["$origin_destination.coordinates", 0]},//returns the element at the specified array
index
                 {$arrayElemAt: ["$origin_destination.coordinates", 1]}//
                ]
            },

           duration: {$divide: [{$subtract: ["$final_time","$init_time"]},60 ]},//calculating duration of bookings in
minutes
           init_time:1,
           final_time:1
           }
  },
  {
     $match: {//filter only actual bookings for which cars moved and the most likely duration
           moved: true, //car must have mooved because the origin destination is not equal and it is true
           duration: {$lte: 180, $gt: 2}//the booking duration is atmost 3 hours(180 minutes) and less than 2
minutes
         }
  },
  {
     $sort: {
           "id.city":1,
           "duration":1
         }
  }
  ])
  while (results.hasNext())
  {
     var a = results.next()
    print(a["city"],a["duration"])
```

```
  }
```

***CDF: per each week of data**

```
//Aggregate per each week of data to calculate CDF - task_1_C
var startDate = new ISODate("2017-10-01T00:00:00")//consider start time of october 2017
var endDate = new ISODate("2017-10-31T23:59:59")//consider end time of october 2017
var results = db.PermanentBookings.aggregate([
  {
    $match:{//filter on time
          $or:[{city: "Torino"}],
          init_date: {$gte:startDate, $lte: endDate} //considering period of october 2017
         }
  },
  {//considering time series: extract position, duration and time
    $project: {
              _id: 0,
              moved : {$ne: [
                        {$arrayElemAt: ["$origin_destination.coordinates",0]}, //gets the origin coordinates
                        {$arrayElemAt: ["$origin_destination.coordinates",1]} //gets the destination coordinates
                       ]
                    },
              duration: {$divide:[{$subtract:["$final_time","$init_time"]},60]},//calculating booking duration from
seconds in minutes
             // hourofDay: {$hour: "$init_date"}, //returns the hour for a date (in our case october 2017) as a
number between 0 and 23
             // daysofmonth: {$dayOfMonth: "$init_date"}//returns the days of the month (in our case october
2017) as a number between 1 and 31
              week:{$week: "$init_date"}//returns the weeks of the month october 2017
            }
  },
  {
    $match: {//filter only actual bookings for which cars moved
              moved: true, //car must have mooved because the origin destination is not equal and it is true
              duration: {$lte: 180, $gt: 2}//the booking duration is atmost 3 hours(180 minutes) and less than 2
minutes
          }
  },
  {
    $group: {
          _id: {city:"$city", week:"$week"},
          No_of_Bookings :{$sum:1}
          }
  },
  {
    $sort: {
          // "daysofmonth":1, //sort day in increasing order
           "_id.week":1, //sort hours of the day in increasing order
           "No_of_Bookings":1
    }
  }
  ])
  while(results.hasNext()){
    var a = results.next()
```

```
    print(a["_id"]["week"],a["No_of_Bookings"])
  }
```

***CDF: per each day of week**

```
var startDate = new ISODate("2017-10-01T00:00:00") //consider starting time of october 2017
var endDate = new ISODate("2017-10-31T23:59:59") //consider ending time of october 2017 in variable endDate
var results = db.collection.aggregate([
  {
    $match: { //applying filter to select what fields we need, here the cities and duration of october 2017
          $or: [{city:"Torino"}],
          init_date: {$gte: startDate, $lte: endDate }
         }
  },
  {
    $project: {//extract position,time and duration of parking
           _id: 0,
           city: 1,
           duration: {$divide: [{$subtract: ["$final_time","$init_time"]},60 ]},//calculating duration of bookings in
minutes
           init_time:1,
           final_time:1,
           week: {$week: "$init_date"},
           day:{$dayOfWeek: "$init_date"}
           }
  },
  {
    $match: {//filter only the most likely duration
             duration: {$lte: 180, $gt: 2},//the booking duration is atmost 3 hours(180 minutes) and less than 2
minutes
             week: {$eq:41},//extract second week's booking duration only of october 2017
             day:{$eq:1}
         }
  },
  {
    $sort: {
          "duration":1
        }
  }
  ])
  while (results.hasNext())
  {
    var a = results.next()
   print(a["day"],a["duration"])
  }
```

Task 2 – System Utilization over time

***rentals per hour of the day (Non Filtered)**

```
var startDate = new ISODate("2017-10-01T00:00:00")//consider start time of october 2017
var endDate = new ISODate("2017-10-31T23:59:59")//consider end time of october 2017
var results = db.collection.aggregate([
  {
    $match:{//filter on time
```

```
            $or:[{city: "Torino"}],
            init_date: {$gte:startDate, $lte: endDate} //considering period of october 2017
           }
   },
   {//considering time series: extract position, duration and time
      $project: {
                _id: 0,
                moved : {$ne: [
                            {$arrayElemAt: ["$origin_destination.coordinates",0]}, //gets the origin coordinates
                            {$arrayElemAt: ["$origin_destination.coordinates",1]} //gets the destination coordinates
                          ]
                        },
                duration: {$divide:[{$subtract:["$final_time","$init_time"]},60]},//calculating booking duration from
seconds in minutes
                hourofDay: {$hour: "$init_date"}, //returns the hour for a date (in our case october 2017) as a number
between 0 and 23
             }
   },
   {
      $group: {
            _id: {city:"$city", hourofDay:"$hourofDay"},
            No_of_Bookings :{$sum:1}
           }
   },
   {
      $sort: {
            "_id.hourofDay":1, //sort hours of the day in increasing order
            "No_of_Bookings":1
      }
   }
   ])
   while(results.hasNext()){
     var a = results.next()
     print(a["_id"]["hourofDay"],a["No_of_Bookings"])
   }
```

***Rentals per hour of the days of the month (Non Filtered)**

```
//Number of Bookings per hours of the day of Torino Enjoy in month October 2017
var startDate = new ISODate("2017-10-01T00:00:00")//consider start time of october 2017
var endDate = new ISODate("2017-10-31T23:59:59")//consider end time of october 2017
var results = db.PermanentBookings.aggregate([
   {
      $match:{//filter on time and city
            $or:[{city: "Torino"}],
            init_date: {$gte:startDate, $lte: endDate} //considering period of october 2017
           }
   },
   {//considering time series: extract position, duration and time
      $project: {
                _id: 0,
                moved : {$ne: [
                            {$arrayElemAt: ["$origin_destination.coordinates",0]}, //gets the origin coordinates
                            {$arrayElemAt: ["$origin_destination.coordinates",1]} //gets the destination coordinates
```

```
                        ]
                    },
                duration: {$divide:[{$subtract:["$final_time","$init_time"]},60]},//calculating booking duration from
seconds in minutes
                daysofmonth: {$dayOfMonth: "$init_date"}//returns the days of the month (in our case october 2017)
as a number between 1 and 31
             }
   },
   {
      $group: {
            _id: {city:"$city", daysofmonth:"$daysofmonth"},
            No_of_Bookings :{$sum:1}
           }
   },
   {
      $sort: {
            "_id.daysofmonth":1, //sort hours of the day in increasing order
            "No_of_Bookings":1
      }
   }
   ])
   while(results.hasNext()){
     var a = results.next()
     print(a["_id"]["daysofmonth"],a["No_of_Bookings"])
   }
```

***Matlab code for plots: Rentals per hour of the day**

```
figure;
plot(Hour,TE,'r');hold on;plot(Hour,TC,'b');plot(Hour,FE,'g');hold off;legend('Torino Enjoy','Torino Car2Go','Firenze
Enjoy')
xlabel('Hour of the Day')
ylabel('#Bookings')
title('Filtered Number of Booked Cars per Hour of the Day in Three Cities')
grid on;
```

***Python code for rentals per hour of the days of month**

```
from matplotlib import pyplot
import json
import numpy as np
import os
import pandas as pd
if __name__ == '__main__':
   cities=['Fi','To']
#    for city in cities:
   folder="F:\ICT\ICT in transport\lab\python\lab1.2.2\Parking"
   bookings = folder + '/' + 'To.csv'
   data=pd.read_csv(bookings, sep=" ",index_col=3, header=None)
   folder="F:\ICT\ICT in transport\lab\python\lab1.2.2\Parking"
   bookings_Fi = folder + '/' + 'Fi.csv'
   data_Fi=pd.read_csv(bookings_Fi, sep=" ",index_col=3, header=None)
   folder="F:\ICT\ICT in transport\lab\python\lab1.2.2\Parking"
   bookings_TO = folder + '/' + 'Tocar2go.csv'
   data_To=pd.read_csv(bookings_TO, sep=" ",index_col=3, header=None)
```

```
x = np.linspace(1, 30, 1424 )
y= np.linspace(1, 30, 1400 )
z= np.linspace(1, 30, 1424 )
pyplot.plot(x,data.iloc[:,2] ,label='Torino Enjoy')
pyplot.plot(y,data_Fi.iloc[:,2] ,label='Firenze Enjoy')
pyplot.plot(z,data_To.iloc[:,2] ,label='Torino CarToGO')
pyplot.xlabel('Days')
pyplot.ylabel('Total booking')
pyplot.title('Rentals per hour of the day ' )
pyplot.legend()
pyplot.show()
```

Task 4 – System Utilization over time: rentals per hour of the day (Filtered)

```
var startDate = new ISODate("2017-10-01T00:00:00")//consider start time of october 2017
var endDate = new ISODate("2017-10-31T23:59:59")//consider end time of october 2017
var results = db.PermanentBookings.aggregate([
  {
    $match:{//filter on time
          $or:[{city: "Torino"}],
          init_date: {$gte:startDate, $lte: endDate} //considering period of october 2017
         }
  },
  {//considering time series: extract position, duration and time

    $project: {
            _id: 0,
            moved : {$ne: [
                    {$arrayElemAt: ["$origin_destination.coordinates",0]}, //gets the origin coordinates
                    {$arrayElemAt: ["$origin_destination.coordinates",1]} //gets the destination coordinates
                   ]
                },
            duration: {$divide:[{$subtract:["$final_time","$init_time"]},60]},//calculating booking duration from
seconds in minutes
            hourofDay: {$hour: "$init_date"}, //returns the hour for a date (in our case october 2017) as a number
between 0 and 23
          }
  },
  {
    $match: {//filter only actual bookings for which cars moved
            moved: true, //car must have mooved because the origin destination is not equal and it is true
            duration: {$lte: 180, $gt: 2}//the booking duration is atmost 3 hours(180 minutes) and less than 2
minutes
        }
  },
  {
    $group: {
          _id: {city:"$city", hourofDay:"$hourofDay"},
          No_of_Bookings :{$sum:1}
          }
  },
  {
    $sort: {
          "_id.hourofDay":1, //sort hours of the day in increasing order
```

```
            "No_of_Bookings":1

        }
    }
    ])
    while(results.hasNext()){
      var a = results.next()
      print(a["_id"]["hourofDay"],a["No_of_Bookings"])
    }
```

*** Rentals per hour of the days of the month (Filtered)**

```
//Number of Bookings per hours of the day of Torino Enjoy in October 2017
var startDate = new ISODate("2017-10-01T00:00:00")//consider start time of october 2017
var endDate = new ISODate("2017-10-31T23:59:59")//consider end time of october 2017
var results = db.PermanentBookings.aggregate([
    {
       $match:{//filter on time and city
              $or:[{city: "Torino"}],
              init_date: {$gte:startDate, $lte: endDate} //considering period of october 2017
             }
    },
    {//considering time series: extract position, duration and time
       $project: {
                  _id: 0,
                  moved : {$ne: [
                               {$arrayElemAt: ["$origin_destination.coordinates",0]}, //gets the origin coordinates
                               {$arrayElemAt: ["$origin_destination.coordinates",1]} //gets the destination coordinates
                              ]
                           },
                  duration: {$divide:[{$subtract:["$final_time","$init_time"]},60]},//calculating booking duration from
seconds in minutes
                 daysofmonth: {$dayOfMonth: "$init_date"}//returns the days of the month (in our case october 2017)
as a number between 1 and 31
                }
    },
    {
      $match: {//filter only actual bookings for which cars moved
                  moved: true, //car must have mooved because the origin destination is not equal and it is true
                  duration: {$lte: 180, $gt: 2}//the booking duration is atmost 3 hours(180 minutes) and less than 2
minutes
              }
   },
    {
       $group: {
              _id: {city:"$city", daysofmonth:"$daysofmonth"},
              No_of_Bookings :{$sum:1}
                 }
    },
    {
       $sort: {
              "_id.daysofmonth":1, //sort hours of the day in increasing order
              "No_of_Bookings":1
       }
```

```
}
])
while(results.hasNext()){
var a = results.next()
print(a["_id"]["daysofmonth"],a["No_of_Bookings"])
}
```

Task 5 – Booking/Parking Statistics: Avg, Std, Median, Percentile
***MongoDB Query**

```
var startDate = new ISODate("2017-10-01T00:00:00")
var startUnixTime = startDate.getTime() /1000
var endDate = new ISODate("2017-10-30T23:59:59")
var endUnixTime = endDate.getTime() / 1000
NextTime=(-2*3600)
var result_Enjoy = db.PermanentBookings.aggregate([
  {
    $match: {
      city: "Torino",
      init_time: { $gte: startUnixTime+NextTime ,$lte: endUnixTime+NextTime}
        }
  },
    {
    $project: {
      _id: 0,
      city: 1,
      moved: { $ne: [
        {$arrayElemAt: [ "$origin_destination.coordinates", 0]},
        {$arrayElemAt: [ "$origin_destination.coordinates", 1]}
        ]
      },
      duration:{$floor: {$divide: [{$subtract: ["$final_time", "$init_time"]}, 60]}},
      init_time: 1,
      init_date:1,
      Day:{$dayOfMonth: "$init_date"}
    }
  },
   {
    $match: {
       moved: true,
       duration: {$lte: 180, $gt: 3}
    }
  },
  {
    $group:
    {
       _id:{ Day:"$Day",duration:"$duration"},
      total_booking:{$sum:1}
    }
  },
  {
    $sort:{
      "_id.Day":1,
      "_id.duration":1
```

```
        }
    }])
while (result_Enjoy.hasNext())
{
   o = result_Enjoy.next()
   print (o["_id"]["Day"],o["_id"]["duration"])
}
```

***Python Code for Plots: Booking/Parking Statistics**

```
import matplotlib.pyplot as plt
import numpy as np
import msvcrt as m
import pylab
import pandas as pd
if __name__ == '__main__':
   average = []
   median = []
   standard_deviation = []
   percentile = []
   booking = {}
   folder="F:\ICT\ICT in transport\lab\python\lab1.2.5\Booking"
   Parking = folder + '/' + 'carTo.csv'
   file = open(Parking,"r")
   data = file.readlines()[2:-1]
   file.close()
   for i in data:
      day, duration = [a.strip() for a in i.split(' ')]
      booking.setdefault(int(day), []).append(float(duration))
   for key in booking.keys():
      val = booking[key]
      average.append(np.mean(val))
      median.append(np.median(val))
      standard_deviation.append(np.std(val))
      percentile.append(np.percentile(val, 75))
   fig1 = plt.figure(1)
   plt.figure(figsize=(10, 5))
   plt.title(" Booking Statistics")
   plt.xlabel("Days")
   plt.ylabel("Duration[minitue]")
   plt.xlim(0.5,30.5)
   dim = np.arange(1, 31, 1);
   plt.grid(True)
   plt.plot(dim,average, label="Average")
   plt.plot(dim,median, label="Median")
   plt.plot(dim,standard_deviation, label="Std")
   plt.plot(dim,percentile, label="Percentile[75%]")
   plt.xticks(dim)
   plt.legend(loc=2)
   fig1.show()
   pylab.show()
   m.getch()
```

Task 6 - Position of parked cars in different hours of the days

```
var startDate = new ISODate("2017-10-01T00:00:00")//consider start time of october 2017
var startUnixTime = startDate.getTime() /1000
var endDate = new ISODate("2017-10-31T23:59:59")//consider end time of october 2017
var endUnixTime = endDate.getTime() / 1000
var range = 3600
var results = db.enjoy_PermanentParkings.aggregate([
  {
    $match:{//filter on time and city
          $or:[{city: "Firenze"}],
          init_date: {$gte:startDate, $lte: endDate} //considering period of october 2017
         }
  },
  {//considering time series: extract position, duration and time
    $project: {
              _id: 0,
              city: 1,
              duration: {$divide:[{$subtract:["$final_time","$init_time"]},60]},//calculating booking duration from
seconds in minutes
              hourofDay: {$hour: "$init_date"}, //returns the hour for a date (in our case october 2017) as a number
between 0 and 23
             daysofmonth: {$dayOfMonth: "$init_date"},//returns the days of the month (in our case october 2017)
as a number between 1 and 31
              loc:1,
              latitude:{$arrayElemAt: ["$loc.coordinates",1]},
              longitude:{$arrayElemAt: ["$loc.coordinates",0]}
    }
    },
  {
    $match:  {
      duration:{$gt:2}
    }
  },
  {
    $sort:{
      "daysofmonth":1,
      "hourofDay":1
    }
  }
  ])
  while(results.hasNext()){
       var a = results.next()
       print(a['daysofmonth'],a['hourofDay'],a['latitude'],a['longitude'])
    }
```

Task 7 – Correlation of the probability of a rental with the availability of other transport means
***Public Transport**

```
var startDate = new ISODate("2017-10-01T00:00:00")
var startUnixTime = startDate.getTime() /1000
var endDate = new ISODate("2017-10-30T23:59:59")
var endUnixTime = endDate.getTime() / 1000
Next_time = -2*60*60
var result = db.PermanentBookings.aggregate([
```

```
   {
    $match: {
        city: "Torino",
        init_time: { $gte: startUnixTime+Next_time, $lte: endUnixTime+Next_time},
        "public_transport.duration":{$ne:-1}
        //"walking.duration":{$ne:-1}
          }
   },
     {
     $project: {
        _id: 0,
        city: 1,
        moved: { $ne: [
          {$arrayElemAt: [ "$origin_destination.coordinates", 0]},
          {$arrayElemAt: [ "$origin_destination.coordinates", 1]}
          ]
        },
        duration: { $floor:{$divide: [ { $subtract: ["$final_time", "$init_time"] }, 60 ] }},
        init_date:1,
        init_time: 1,
        hourDay: {$hour: "$init_date"},
        day: {$dayOfMonth: "$init_date"},
        ptDduration:{$floor:{$divide:["$public_transport.duration",60]}}
        //walking_duration : {$floor:{$divide:["$walking.duration",60]}}
     }
    },
     {
      $match: {
         moved: true,
         duration: {$lte: 180, $gt: 2}
      }
    },
{
  $sort:{
     "day":1,
     "hourDay":1
     }
   }
 ])
while(result.hasNext()) {
   o= result.next()
   print (o["duration"],o["ptDduration"])
}
```

Lab2

***First scenario: all day**

```
Var zone=[“TorinoZonesArray.geojson”]
for(i=0;i<zone.length;i++)
   for(j=0;j<zone.length;j++){
var result=db.ictts_PermanentBookings.aggregate([
{
   $project:{hour:{$hour:"$init_date"},
           day:{$dayOfWeek:"$init_date"},
           init_loc:1,
           final_loc:1
           }
},
{
   $match:{
          init_loc:{$geoWithin:{
                  $geometry:{
                    "type":"MultiPolygon",
                    "coordinates":zone[i]}
                    }
                    },
          final_loc:{$geoWithin:{
                   $geometry:{
                     "type":"MultiPolygon",
                     "coordinates":zone[j]}
                     }
                     },

        }
},

{$count:"total"}

])
if(result.hasNext())
{
   o = result.next()
   print (i,",",j,",",o["total"])

} else
{print(i,",",j,",",0)}}
```

***Second scenario: weekend**

```
Var zone=[“TorinoZonesArray.geojson”]
for(i=0;i<zone.length;i++)
   for(j=0;j<zone.length;j++){
var result=db.ictts_PermanentBookings.aggregate([
{
   $project:{hour:{$hour:"$init_date"},
           day:{$dayOfWeek:"$init_date"},
           init_loc:1,
           final_loc:1
           }
```

```
},
{
   $match:{
          init_loc:{$geoWithin:{
                  $geometry:{
                    "type":"MultiPolygon",
                    "coordinates":zone[i]}
                    }
                    },
          final_loc:{$geoWithin:{
                   $geometry:{
                     "type":"MultiPolygon",
                     "coordinates":zone[j]}
                     }
                     },
          $or : [ { day : 1 }, { day : 7 } ]
         }
},

{$count:"total"}

])
if(result.hasNext())
{
   o = result.next()
   print (i,",",j,",",o["total"])

} else
{print(i,",",j,",",0)}}
```

***Third scenario: weekday (morning/afternoon)**

```
Var zone=[“TorinoZonesArray.geojson”]
for(i=0;i<zone.length;i++)
for(j=0;j<zone.length;j++){
var result=db.ictts_PermanentBookings.aggregate([
{
   $project:{hour:{$hour:"$init_date"},
           day:{$dayOfWeek:"$init_date"},
           init_loc:1,
           final_loc:1
           }
},
{
   $match:{
          init_loc:{$geoWithin:{
                  $geometry:{
                    "type":"MultiPolygon",
                    "coordinates":zone[i]}
                    }
                    },
          final_loc:{$geoWithin:{
                   $geometry:{
                     "type":"MultiPolygon",
```

```
                    "coordinates":zone[j]}
                    }
                    },
          hour:{$gte:0,$lt:12},
          day:{$gte:2,$lt:5}
         }
},

{$count:"total"}

])
if(result.hasNext())
{
   o = result.next()
   print (i,",",j,",",o["total"])

} else
{print(i,",",j,",",0)}}
```