Journal of Big Data

RESEARCH Open Access

Detecting taxi movements using Random Swap clustering and sequential pattern mining

Rami Ibrahim and M. Omair Shafiq*

*Correspondence:
omair_niit@yahoo.com;
omair.shafiq@carleton.ca
School of Information Technology, Carleton University, Ottawa, ON, Canada

Abstract

Moving objects such as people, animals, and vehicles have generated a large amount of spatiotemporal data by using location-capture technologies and mobile devices. This collected data needs to be processed, visualized and analyzed to transform raw trajectory data into useful knowledge. In this study, we build a system to deliver a set of traffic insights and recommendations by applying two techniques, clustering, and sequential pattern mining. This system has three stages, the first stage preprocesses and samples the dataset into 168 subsets, the second stage applies two clustering techniques, the hierarchical density-based spatial clustering (HDBSCAN) and the Random Swap clustering (RS). We compare these two clustering algorithms in terms of processing time and quality of clusters. In the comparative analysis, the Silhouette coefficient shows that RS clustering outperforms HDBSCAN in terms of clusters quality. Moreover, the analysis shows that RS outperforms K-means in terms of the mean of square error (MSE) reduction. After that, we use a Google Maps approach to label the traffic districts and apply sequential pattern mining to extract taxi trips flow. The system can detect 146 sequential patterns in different areas of the city. In the last stage, we visualize traffic clusters generated from the RS algorithm. Furthermore, we visualize the taxi trips heatmap per weekday and hour of the day in Porto city. This system can be integrated with the current traffic control applications to provide useful guidelines for taxi drivers, passengers, and transportation authorities.

Keywords: Random Swap, HDBSCAN, Sequential pattern mining

Introduction

Understanding the nature of trajectories can help in analyzing their behavior. A spatiotemporal trajectory is a time stamped sequence generated by tracking the location of a moving object [1]. This sequence is represented by a series of space and time instances. A vast number of real-life applications are using mobile services sensors and Global Position Systems (GPS) to collect trajectory data. They apply trajectory mining techniques to discover knowledge which provides useful information for social networks, transportation systems, and urban computing.

Several trajectory data mining techniques have been proposed such as pattern mining, clustering, classification, time series analysis, and anomaly detection. These techniques can help the city to detect road networks by tracing the movement of people in this city [2]. Trajectory mining techniques can translate collected geospatial coordinates and transform them into text information such as points of interest

(POI). Additionally, analyzing traffic flow can help authorities to understand factors influenced by this flow such as air quality, noise pollution, and peoples physical and mental health.

Individuals movement patterns can be detected using GPS devices [3]. Mining these patterns can uncover valuable information such as mode of transport (e.g., walk, bike, bus, car) and locations of significance. When the semantic description is provided along with trajectory data, activities associated with each location can be recognized like restaurants, shopping centers, and working places. In addition to the GPS sensors, Radio frequency identification (RFID) has been used. RFID is used in a vast number of applications such as warehouse management systems to monitor and control their operations [4]. RFID sensors can collect manufacturing data which can be useful for discovering trajectory patterns.

Despite the usage of trajectory mining in a wide range of application areas, there are some limitations which should be addressed. First, some studies conducted trajectory analysis in a reasonable granularity level (e.g., day, week). However, when a higher level of granularity is considered (e.g., hour, minute), more interesting patterns can be detected. Second, many studies relied on numerical features to uncover traffic movements behavior. Meanwhile, in our study, we use both numerical data (i.e., longitudinal) and textual data (i.e., city districts) to label our clusters and produce more interesting patterns. Adding the district dimension will enrich our analysis since taxi drivers and passengers can identify areas with various traffic conditions. Third, previous studies applied clustering algorithms such as K-means and DBSCAN. The main drawback of these algorithms is the lack of the ability to fine-tune the clusters once they are produced. Therefore, we apply a novel algorithm called Random Swap clustering (RS) [5] which proved to have a significant improvement for the quality of clusters. Generating the correct traffic clusters will improve the accuracy of detected patterns, thus, will provide taxi drivers and passengers with meaningful insights and recommendations.

Given these limitations, our main contribution is to establish a data-driven approach to enhance the traffic in Porto city by delivering a set of recommendations based on trajectory analysis. Unlike previous studies, our study extracts insights on a high granularity level where taxi trips are processed based on the day and hour they started. This study applied clustering for both origin and destination points to provide useful guidelines. We used two clustering techniques, HDBSCAN and RS clustering on the origin and destination points. We clustered 168 files as we set the granularity level to 1 h (i.e., 24 h $\times$ 7 days).

The novelty of our study is using a combination of efficient algorithms such as Random Swap (RS) clustering and sequential pattern mining (SPADE) to detect traffic in different districts and timestamps. Previous experiments proved that RS is more efficient than other algorithms such as K-means [6]. It produces clusters with higher quality as it minimizes the sum of square error (SSE) and the centroid index (CI). Moreover, we added more useful features to taxi trips dataset by using Google Maps to extract districts. This feature enables our analysis to detect areas with a high number of trips at any given day and hour, which makes it a multi-dimensional approach (i.e., space vs. time analysis).

Paper organization

The remaining of the paper is organized as follows. "Related work" section presents and discusses related work. "Methodology" section presents the main concepts and methodology of applying RS clustering and sequential pattern mining to produce insights and recommendations for taxi drivers and passengers, followed by results in "Results" section. "Discussion" section presents discussion on implications of our methodology and results. "Limitations and future work" section presents future work and enhancements. "Conclusions" section presents conclusions followed by acknowledgments and references.

Related work

Various trajectory mining techniques were proposed such as clustering and sequential pattern mining. For trajectory data clustering, some studies relied on textual features to cluster their datasets. They utilized semantic trajectories where textual description and photos are provided for each trajectory point. In [7], the authors generated patterns and regions of interest (ROI) by analyzing semantic trajectories. They introduced SimDB-SCAN clustering algorithm which they extracted from extending DBSCAN algorithm. They clustered areas based on their similarities and included Flickr photos of trajectory points for user preferences description. Efficient models combined trajectory points with regions of interest (ROI).

However, most studies relied on numerical data such as longitudinal coordinates to cluster trajectory points. In [8], the authors clustered origin and destination points for trips trajectories (OD). The purpose of their approach was to detect household travel patterns by identifying attractive places, then visualizing clusters based on spatial distribution and temporal direction. In [9], the authors proposed a spatial clustering algorithm called R-FDBSCAN. It is based on DBSCAN, but the size of clusters was controlled using an additional parameter *R*. They clustered the pickup points (i.e., origin) for different periods. The experiments showed that their algorithm had multiple advantages in terms of time performance and clustering outcome. By visualizing their clusters on the map of Beijing, they showed that pickup hotspots were highly synchronized with the daily regularity of the city residents. In [10], the author discovered attractive areas and interesting patterns by clustering pickup and drop-off points (i.e., origin and destination) in different periods. They analyzed the distribution of clusters in five main periods; then they measured the flow among clusters (i.e., inter-clusters flow). Finally, they extracted the level of attractiveness for each area and period. In [11], the authors proposed a new approach called density peaks clustering (DPC). The purpose was to discover hot spots by using density peaks feature. They combined their algorithm with image analysis techniques to improve the efficiency of DPC algorithm. Their experiments showed that DPC could be used to detect long term hot spots.

However, some previous studies clustered grids of data instead of clustering pickup and drop-off points. In [12], the authors divided trajectory dataset into smaller grids; then they applied DBSCAN clustering on each grid for each timestamp. They found a congestion level in certain areas. In [13], the authors proposed a grid-based clustering algorithm GRIDBSCAN to explore attractive areas. The GRIDBSCAN required two parameters, size of grid k and grid density threshold Λ. The model considered both spatial and temporal constraints. Their algorithm was able to extract the areas which had massive traffic and frequent activities. Some studies focused on the taxi driver instead of the regions and districts. In [14], the authors analyzed the impact of the taxi driver's behavior on his daily profit margin. They used K-means to group drivers based on their driving duration and distance. Then they studied these groups in terms of spatial and temporal driving patterns. The driving patterns were extracted using DBSCAN algorithm. Their experiments showed that high-profit drivers had more frequent trips than low-profit drivers. Moreover, high-profit drivers had more trips with duration.

Another trajectory mining technique is sequential pattern mining. It is used to discover repeated subsequences in a set of sequences. Time series data is an ordered form of sequences where data is represented over time. Trajectory data can be considered as a time series sequence which consists of a set of ordered longitudinal locations. In [15], the authors mentioned that a discretization process is required to perform sequential pattern mining. For example, when dealing with financial time series data, numerical amounts of money should be associated with symbols based on the defined intervals. In [16], the authors performed spatiotemporal trajectory region-of-interest (ROI) sequential pattern mining. They used Flickr photos for Queensland taken by tourists to represent each ROI point in a trajectory. They applied the sequential pattern mining (SPM) framework to mine the ROI dataset, and they were able to uncover exciting patterns in the east coast and Brisbane.

One essential technique of mining trajectory data is the data visualization. For example, traffic data has become a main part of human life as people tend to spend a decent time on roads every day [17]. Visualization of traffic data can provide a natural interpretation that integrates human capabilities. Taxi trajectories charts can explore congestion areas and traffic jams. Detected taxi flow patterns can produce more insights and recommendations to both taxi drivers and passengers. In [18], the authors provided more insights into human activity and patterns for New York City residents. They used taxi trips dataset along with a wide range of spatiotemporal queries which use origin and destination features. The end user could choose a sample of the taxi dataset and generate a spatial distribution of trips in different districts of New York City. In [19], the authors proposed a visualization model that described passenger status (e.g., vacant and occupied) and the speed of the vehicle. The model used MongoDB database to store taxi trips dataset. After that, they applied DBSCAN clustering to visualize passenger pickup points. Their model provided an interactive visualization for the taxi movements. In [20], the authors proposed a model to visualize taxi trips in Beijing. They relied on some factors like daily operation time, driver residence location, and driver rest periods. Their experiments showed that taxi traffic formed 20% of the Beijing traffic, and the vacant taxis rate was more than busy taxis rate. In our paper, visualization is the last stage of our system as we can notice in Fig. 1. We use map visualization for taxi clusters in different districts of Porto city. Moreover, we use spatiotemporal heatmaps to describe the traffic status in a given day and hour of the day. Additionally, we use R Shiny interactive visualization to generate sequential patterns based on the selection of the end user.

Some previous studies applied trajectory data mining techniques to build recommendation systems. For example, on taxi trips, services offered to passengers are not efficient as taxi drivers can experience traffic jams and congestions. The lack of guidelines and

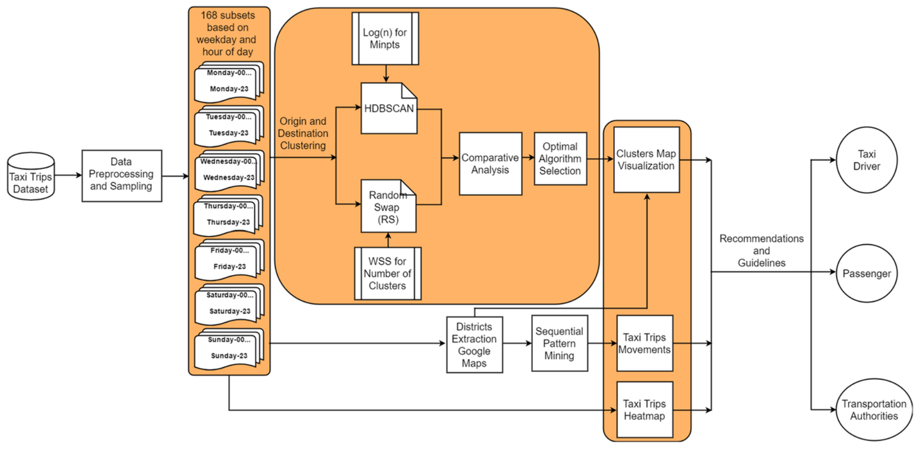

Taxi Trips Dataset
Data Preprocessing and Sampling
168 subsets based on weekday and hour of day
Monday-00... Monday-23
Tuesday-00... Tuesday-23
Wednesday-00... Wednesday-23
Thursday-00... Thursday-23
Friday-00... Friday-23
Saturday-00... Saturday-23
Sunday-00... Sunday-23
Origin and Destination Clustering
Log(n) for Minpts
HDBSCAN
Random Swap (RS)
WSS for Number of Clusters
Comparative Analysis
Optimal Algorithm Selection
Clusters Map Visualization
Districts Extraction Google Maps
Sequential Pattern Mining
Taxi Trips Movements
Taxi Trips Heatmap
Recommendations and Guidelines
Taxi Driver
Passenger
Transportation Authorities

Fig. 1 Three stages system architecture

Table 1 Existing taxi clustering techniques

Studies	Clustering algorithm	Features
Takimoto et al. [7]	SimDBSCAN	Clustered semantic trajectory
Qi and Liu [9]	R-FDBSCAN	Clustered pickup points with tuning the clusters size
Liu et al. [11]	DPC	Clustered pickup points with image analysis
Saptawati [12]	DBSCAN	Clustered grids of spatial data
Linjiang et al. [13]	GRIDBSCAN	Clustered grids of spatiotemporal data
Naji et al. [14]	K-means DBSCAN	Clustered drivers' profiles, then clustered driving patterns

recommendations could affect both driver and passenger; the taxi driver is not able to drop off the passenger on the planned time. Additionally, his car is consuming more gas while being stuck in high traffic areas. On the other hand, if a passenger is looking for a vacant taxi, it would be difficult to get one in the low traffic areas where there are a few numbers of taxis [21]. Authors proposed several techniques to build an effective recommendation system that can help both drivers and passengers such as clustering, classification, and regression prediction.

However, in our study, we perform clustering to detect traffic in origin and destination. Detecting the traffic in origin will help taxi drivers to identify the hot spots in the city; they can detect places where they can likely pick up passengers. While detecting traffic in trips destination can help passengers to identify hot areas where trips end, thus, they can find more vacant taxis to ride. Table 1 summarizes the related work in the field of taxi trips clustering techniques.

Methodology

This section presents our methodology design and details, and the dataset chosen.

System architecture

The final deliverable of our study is a set of recommendations and insights which can enhance the traffic by performing two main tasks, clustering taxi trips for a given district and period, and detecting taxi movements among different districts. Figure 1 describes our solution architecture. It shows that our solution has three stages; the first stage is the preprocessing and sampling stage. In this stage, we cleaned the dataset and extracted features required for our analysis such as weekday, hour of the day, and city districts. Moreover, we divided the main dataset into small subsets; each subset contained taxi trips that started in a specific weekday and hour of the day. As we had 7 weekdays and 24 h of the day, the outcome of this stage was 168 subfiles. The second stage had the two models we built to get more insights into the city traffic; these models were clustering and sequential pattern mining. As clustering and sequential pattern mining are unsupervised learning models, their output is a set of taxi movement patterns over Porto city districts. For the clustering, we applied two types of algorithms, a centroid-based algorithm called Random Swap (RS) and a density-based algorithm called HDBSCAN. After that, we conducted a comparative analysis for these two algorithms in terms of processing time and quality of clusters; then we selected the optimal algorithm to be used for the last stage. For the sequential pattern mining, we used a subset of the original dataset; then we extracted the flow of taxi trips over different districts in Porto city. The last stage was visualizing taxi trips based on outputs from stage two. We used Tableau Desktop [22] to visualize taxi trips clusters and plot them on Porto city map. Tableau maps were generated for both origin points and destination points.

Additionally, we visualized the flow of taxi trips with the arrows to display the direction of each flow by using Pereira's Porto city map [23]. The visualization of the traffic status was implemented by generating a heatmap for taxi trips on weekdays and hour of the day. Finally, we assume that the generated set of insights and recommendations can be useful for taxi drivers, passengers, and transportation authorities in Porto city.

Dataset description

In our study, we used a taxi trajectory dataset which represented 442 taxi trajectories in Porto city, Portugal [24]. This dataset contained all taxi trips traveled from June 2013 to June 2014. The original CSV file had 1.7 million trips with an approximate size of 2 GB. Each trip was a separate tuple with several attributes; trip id was the unique attribute used to identify each trip. The call type attribute defined the area where the taxi was requested like central area, random street, or taxi station. The phone number from which the cab was requested was stored in origin call attribute, and if the cab started from a station, the origin stand attribute described the point where the trips started.

The taxi driver could be identified using taxi id attribute, the start time of each trip was represented by a Unix timestamp attribute. The day type (i.e., weekday, holiday, or weekend) was represented by day-type attribute, and the missing attribute indicated the status of GPS reading. If the reading could not be captured, the missing attribute had a value of true; otherwise, if the reading was captured the attribute had a value of false. The attribute that described the trajectory of each trip was a polyline; this attribute stored a sequence of longitudinal coordinates. Each pair of values was a combination of longitude and latitude coordinates for a specific location. The time frequency for this attribute was 15 s, which meant that a new pair of coordinates (i.e., new longitude and latitude) was added every 15 s. The number of pairs in this attribute represented the length of the trip; therefore, if there were 40 pairs of coordinates in polyline attribute, then the duration was 40×15 s since the time frequency is 15 s. Thus, the trip duration was 600 s which meant that this trip lasted for 10 min.

Dataset preprocessing

We downloaded the taxi trajectory CSV file from the UCI Machine Learning Repository website [25]. After that, the SQLite DB browser [26] was used to import the CSV file as a table called taxi data. In order to include successfully captured GPS data, all rows with a true value missing attribute were deleted. To perform analytical techniques on this dataset, we converted the Unix timestamp attribute into windows timestamp with the following format "year–month–day hour–minute–second". Next was the extraction of the first and last pairs of the polyline attribute. The first pair represented the trip start location, and the last pair represented the trip end location. Next was the calculation of duration for each trip by multiplying the number of pairs with the time frequency (i.e., 15 s). In addition, the start time of each trip was extracted, weekday (e.g., Saturday, Sunday, etc.), month (e.g., Jan, Feb, Mar, etc.), and year. Based on our queries, the first trip in the dataset started on the 30th of June 2013 at 20:00, while the last trip started on the 30th of June 2014 at 19:59. Then, our previous calculations were added to the table as new attributes (i.e., trip start time, trip end time, duration, weekday, month, year, trip start location, trip end location). After preparing the final version of our table in SQLite DB browser, we extracted this table as a CSV file, then Delimit software [27] was used to browse the CSV file and partition the polyline attribute into smaller attributes.

The polyline attribute was divided into pairs; then each pair was separated into a single attribute where every single attribute represented a longitude or a latitude reading. The number of attributes generated from Delimit partitioning varied and was different for each trip. Therefore, longer trips had more attributes generated from the partitioning process. For example, if there was a trip with a duration of 5 min (i.e., 300 s), and with a time frequency of 15 s, we got 40 attributes generated from Delimit partitioning. However, if there was a trip with a duration of 30 min (i.e., 1800 s) and a time frequency of 15 s, then we got 240 generated attributes.

Dataset sampling

As the number of taxi trips in the dataset was 1.7 million trips, sampling was required for this dataset to apply trajectory mining techniques such as clustering and sequential pattern mining. When analyzing the whole dataset, these techniques can face limitations such as space and memory issues; they cannot handle the processing of this massive amount of taxi trips. Therefore, we extracted a subset from the original taxi trips dataset; this subset represented taxi trips which started in May. This month was chosen as it is one of the best times to visit Porto city because of the beautiful weather and cheap hotels.

Porto districts extraction

This dataset had longitude and latitude coordinates without having the description of the district for each location. To perform techniques like sequential pattern mining, we need to use the name of the district where the taxi is located. This approach used Google Maps [28] to divide Porto city into smaller rectangles. For the districts of Porto city, we relied on the Porto urban distinctions research conducted by Pereira in 2018 [23]. In his study, he divided Porto city into 18 various districts; he identified the location (i.e., central, periphery, Atlantic) and the nature of each district (i.e., historical). In Google Maps, the distance measurement feature was used to plot the corner points for each district, after that another point was plotted as Google Maps drew a line between these two points to calculate the distance between them. We kept plotting new points until we had rectangles of all districts in Porto city, each district was a rectangle area with four corners. In the end, there were 18 districts as shown in Fig. 2. To distinguish each district, we extracted the four corners coordinates for each district. For each plotted point on Google Maps, longitude and latitude coordinates of each corner were identified by using the right mouse click. As the city was divided into multiple rectangles, the assumption was to generate equiangular, which means a generation of a rectangle where each parallel side are equal in length. Therefore, it was said that a taxi belongs to a district if its current longitude lies between right and left longitude coordinates of the district, and its current latitude lies between lower and upper-latitude coordinates of the district. Table 2 describes the extracted districts and the longitude and latitude coordinates for each border (i.e., four borders for each district).

The final step was to scan the three CSV files and insert the district for each longitudinal coordinate according to the districts coordinates in Table 2. To accomplish this, we imported the districts coordinates into an Oracle table using Oracle Application Express [29]. Then the CSV files were stored in three Oracle tables.

After that, a PL/SQL script was created to scan all CSV files in a loop and compare each longitudinal coordinate with the border coordinates of each district. The trip coordinates were assigned to a district if they lied within its borders. If the trip coordinates did not lie within any district, then the value 'no district' was assigned to that point. Furthermore, there were few trip points which were lying in more than one district; this could be due to the manual rectangles plotting. There could be a slight overlap between rectangles (i.e., districts) when plotting the corners and drawing the borders in Google Maps. If there were more than one district, the trip point was assigned to any one of

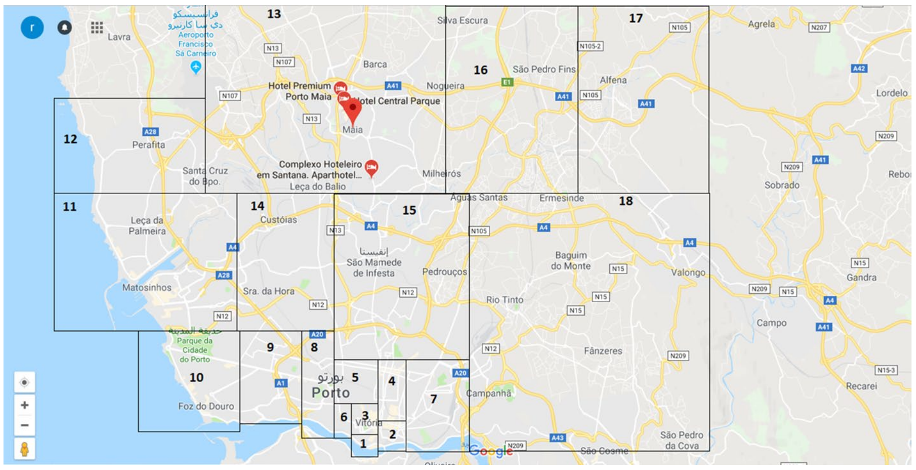


Fig. 2 Porto city districts as drawn in Google Maps

Table 2 Porto city districts border coordinates

District	Lower Lat.	Upper Lat.	Right long.	Left long.
Sao Nicolau	41.139357	41.143865	−8.611466	−8.621299
Se	41.139193	41.148208	−8.600352	−8.611466
Vitoria	41.144025	41.152879	−8.611251	−8.621298
Santo Ildefonso	41.148047	41.164788	−8.60056	−8.611249
Cedofeita	41.152879	41.165274	−8.611247	−8.628137
Miaragaia	41.14274	41.152881	−8.621298	−8.627924
Bonfim and Campnha	41.139827	41.165101	−8.577693	−8.600565
Massarelos	41.14435	41.17252	−8.627924	−8.639681
Lordelo Do Ouro	41.146765	41.172843	−8.639467	−8.661271
Foz Do Douro	41.144509	41.17284	−8.661271	−8.703385
Matosinhos	41.172523	41.210809	−8.664262	−8.732013
Perafita	41.210875	41.262566	−8.675873	−8.73124
Maia	41.210921	41.262463	−8.585343	−8.675948
Ramalde and Aldoar	41.172683	41.210647	−8.628359	−8.664262
Paranhos	41.164609	41.210949	−8.577473	−8.627925
Sao Pedro Fins	41.210791	41.263121	−8.540443	−8.58532
Alfena	41.21103	41.262843	−8.492282	−8.540443
Corujeiera and Roque	41.140778	41.211122	−8.491102	−8.57768
Vila Nova De Gaia	41.07919	41.136052	−8.571571	−8.67612

these districts. At the end of this approach, there were three tables with the following attributes, weekday, start time, longitudinal coordinates, and districts corresponding to each pair of longitudinal coordinates.

HDBSCAN clustering

After extracting the districts from longitudinal coordinates in the CSV files, the clustering technique was applied to categorize taxi trips based on Euclidean distance measure. In our study, we first used density-based clustering to detect taxi trips clusters in each subset. Additionally, spatial clustering was applied on both taxi trips origin points, and taxi trips destination points in the 168 subsets created in the first stage as shown in Fig. 1. Trips that started at the same hour on the same day were clustered together. Therefore, the original CSV file was sampled into small subsets (i.e., weekday vs. hour). Applying the clustering on the whole CSV file was improper as trips were starting at different periods. To prepare the CSV files for clustering, we eliminated taxi trips which had missing longitudinal coordinates using R sqldf package [30].

Our first clustering algorithm adopted a hierarchical density-based spatial clustering of applications with noise method called HDBSCAN [31]. This algorithm is based on the traditional density-based clustering algorithm DBSCAN [32]. Therefore, this algorithm is efficient in detecting random shapes like neighborhoods in the city. It can discover various density areas and can identify trips which are noise. The DBSCAN algorithm requires two initial parameters, the maximum distance between points and the minimum number of points in a cluster. Previous studies indicated that DBSCAN is sensitive to these parameters; improper values for initial parameters could cause a poor clustering outcome and would impact the size and number of clusters generated. Therefore, applying HDBSCAN will enhance the traditional DBSCAN since it requires only one initial parameter which is the minimum number of points in a cluster.

This algorithm calculates the reachability distance among points; then it builds a dendrogram for significant clusters. To implement the HDBSCAN algorithm, an R script [33] was created which reads each CSV subfile, the script used R hdbscan package [34]. The script scanned every pair of longitudinal coordinates separately. Each pair is a point in the dataset where the HDBSCAN was executed, in each iteration the script called hdbscan function which took two parameters, the CSV pair of coordinates and the minimum number of points *MinPts.* In terms of *MinPts* parameter, the optimal value was to be set by a domain expert, but since it was not applicable to work along with domain experts in our study, the *MinPts* was set to *log(n)*, where *n* is the number of points to be clustered. After executing the hdbscan function, the script extracted the cluster number *C* and stored it as a new attribute in each CSV files as shown in Fig. 3.

Random Swap (RS) clustering

We used the same 168 subsets generated from the first stage for our clustering. The second algorithm is a centroid-based; it is called Random Swap [5]. Random Swap (RS) is an efficient clustering algorithm that is used to fine-tune the location of K-means [6] centroids. Its primary purpose is to improve the clustering quality and generate the correct clusters by swapping centroids iteratively. For each iteration, RS swap centroids and calls K-means, then it measures the quality of clustering by relying on two main factors, the centroid index (CI) and the normalized sum of square errors (MSE). If the CI value is 0, this means the clustering outcome is accurate. If the value is greater than 0, it means that there are some missing clusters in some areas. Calculating CI require a ground truth table which was not available in our study. Therefore, we relied on normalized sum of square errors (MSE) to measure the quality of our clusters.

```
HDBSCAN Clustering
Input: 168 CSV files-origin (weekday vs. hour of the day)
168 CSV files-destination (weekday vs. hour of the day)
Output: clustered CSV files with an extra attribute, cluster
number C for each pair of coordinates
Algorithm: HDBSCAN (CSV files)
    1. Read the CSV file
    2. Iterate i from 1 to nPairCoordinates
    3. Extract longitude and latitude for pair i
    4. Count = number of nonempty records of
       longitude
    5. MinPts = Round of Log (Count)
    6. If MinPts = 1
       // The cluster cannot contain one point only
       Break;
       Else
       // Apply the clustering for this pair of coordinates
       Ci = HDBSCAN (longitude, latitude, MinPts)
    7. add Ci attribute to the CSV file
    8. Export a new CSV file for visualization
```

Fig. 3 HDBSCAN clustering Pseudo-Code, based on [31]

Table 3 RS customized parameters

Parameter	Value
Dataset	Taxi trips dataset
Clusters	*k* value from WSS
Iterations	5000

```
Random Swap (RS) Clustering
Input: 168 CSV files-origin (weekday vs. hour of the day)
168 CSV files-destination (weekday vs. hour of the day)
Output: clustered CSV files with an extra attribute, a cluster number C for each pair of coordinates
Algorithm: RS (CSV files, C, P)
    1.  Read the CSV file
    2.  K-means (CSV files, C, P, k)
    3.  Calculate MSE
    4.  Select Random centroids Ci
    5.  Select Random clusters Pi
    6.  Iterate j from 1 to T
    7.  Create new clusters based on selected Ci and Pi
    8.  K-means (CSV files, Ci, Pi, k)
    9.  If MSEi < MSE (clusters improved)
        Change centroids and clusters to Ci and Pi
        Return to step 2 with new clusters and centroids
        Else
        Exit with current clusters and centroids
    10. add C attribute to the CSV file
    11. Export a new CSV file for visualization
```

Fig. 4 RS clustering Pseudo-Code, based on [5]

The K-means algorithm required one parameter which is the number of clusters *k*. In our experiment, we visualized the optimal number of clusters by using within sum of squares average method (WSS) [35]. We created an R script that used a package called factoextra [36]. This package had a function called fviz_nbclust; this function read each subset and visualized the WSS optimal number of clusters plot. Then we relied on the Elbow method and the plot to choose the number of clusters where we observe that adding another cluster does not improve the WSS value. We used this value for *k* parameter in the K-means algorithm. The next step is to apply the RS algorithm, the R script for this algorithm was available on the University of Eastern Finland Machine Learning website [37]. We used the same R script in our experiments by changing the parameters described in Table 3.

As we could observe from Fig. 4, the RS will keep iterating and minimizing the normalized sum of square errors (MSE). It will keep iterating until no significant reduction is observed. It starts with removing a cluster and creating a new one in a different area; then it calculates MSE value for new clusters before applying K-means on this set of clusters. After that, it calculates the MSE value after every K-means execution and compares it to the MSE value before K-means execution. If the new value is less than the old value, it assigns the centroids and clusters to the new value and applies

K-means for another iteration. Otherwise, it will choose the old centroids and clusters; then it will exit.

In our study, we used the RS algorithm with K-means clustering. K-means is a well-known algorithm that is used on a wide scale; it calculates the centroid for each cluster and assigns each point to the closest cluster. It keeps calculating centroids and reassigning points to their new clusters until there is no change in clusters (i.e., convergence). However, RS can be used with any centroid-based clustering algorithms such as K-means++, x-means, and global K-means as we can observe from Fränti's study [5]. Moreover, the author applied the RS algorithm on different datasets with various dimensionality and scalability.

Results

This section presents results from the methods chosen and applied on the dataset.

Comparative analysis

After applying HDBSCAN and RS clustering on the 168 CSV files, we decide which clustering algorithm is optimal relying on a set of factors. These factors are quality of clusters, mean of sum of square error (MSE), and the processing time in seconds.

Quality of clusters

To measure the quality of clusters for both algorithms, we used the Silhouette coefficient [38]. This coefficient measures the average distance between a given point *p* and the rest of the points in the same cluster. Let us assume that this distance is *b(p)*, where *p* is the given point. After that, it measures the average distance between the point *p* and points from the nearest cluster, let us name it *a(p)*, where *p* is the given point. The Silhouette score is the difference between *b(p)* and *a(p)*, the last step is to normalize the score, this is done by dividing the difference by the maximum value of *a* and *b* as we can observe from the Silhouette coefficient equation:

$$s(p) = \frac{b(p) - a(p)}{\max\{a(p), b(p)\}}, \text{ where } -1 \leq s(p) \leq 1$$

The Silhouette coefficient indicates how much a point is close to its cluster. If the value is negative, it means the point is far from its cluster. If the value is zero, it means the point is located between two clusters. If the value is positive, it means the point is close to its cluster. Therefore, to have good quality clusters, we need $b(p) \gg a(p)$ since we want the Silhouette coefficient to be close to 1.

To extract the Silhouette coefficient for our clusters, we applied two R packages, cluster [39], and vegan [40]. We calculated the Silhouette coefficient 168 times for HDBSCAN algorithm and 168 times for RS algorithm. Furthermore, we repeated the calculation twice, once for taxi trips origin and once for taxi trips destination. The following table shows the factors we extracted as we executed our R scripts for both HDBSCAN and RS.

After calculating the Silhouette coefficient, we compared the scores for both algorithms HDBSCAN and RS. In our comparison, we considered both taxi trips origin and taxi trips destination. Figure 5 shows the Silhouette coefficient scores for HDBSCAN

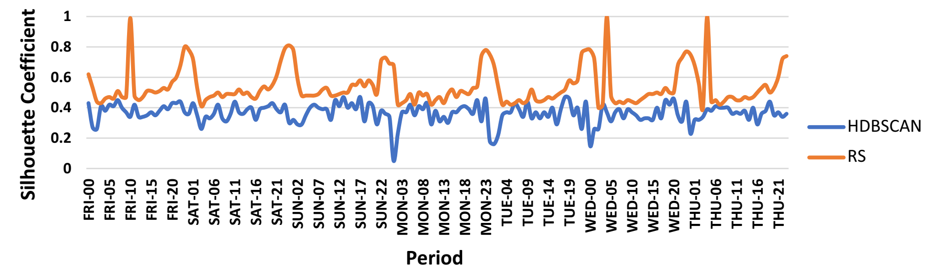


Fig. 5 Silhouette coefficient for trips origin

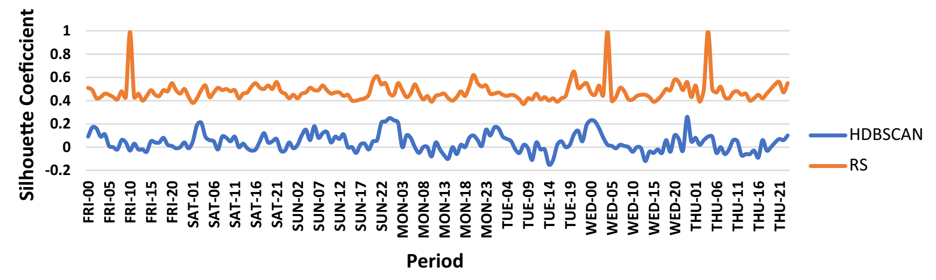


Fig. 6 Silhouette coefficient for trips destination

and RS for the 168 trips origin CSV files. As we can observe from the figure, the RS algorithm significantly outperformed the HDBSCAN as the quality of clusters was higher for most of the days and hours. We can notice that the RS clusters were robust for three periods, Friday 10:00 A.M., Wednesday 4:00 A.M., and Thursday 4:00 A.M. Other RS clusters were reasonable with a Silhouette coefficient varied between 0.45 and 0.8. Meanwhile, HDBSCAN clusters were weaker as their Silhouette coefficient ranged between 0.05 and 0.47. Figure 6 shows the Silhouette coefficient scores for HDBSCAN and RS for the 168 trips destination CSV files. We can notice from the figure that the RS algorithm significantly outperformed the HDBSCAN as the quality of clusters was higher for all days and hours. Additionally, we had robust clusters for the same periods, Friday 10:00 A.M., Wednesday 4:00 A.M., and Thursday 4:00 A.M.. Other RS clusters were reasonable with a Silhouette coefficient varied between 0.37 and 0.65. Meanwhile, HDBSCAN clusters were weak as some clusters had a negative Silhouette coefficient. The HDBSCAN coefficient values ranged between −0.15 and 0.26.

Mean of sum of square error (MSE)

As we know from statistics, this factor is a normalization for the sum of square error (SSE). It represents the amount of clustering error produced. A reduction in the MSE value means that the RS algorithm successfully improved the clustering output. MSE value can be calculated using the following equation:

$$MSE = \frac{SSE}{N.D} \text{where } SSE = \sum_{i=1}^{N} \|x_i - c_{pi}\|^2$$

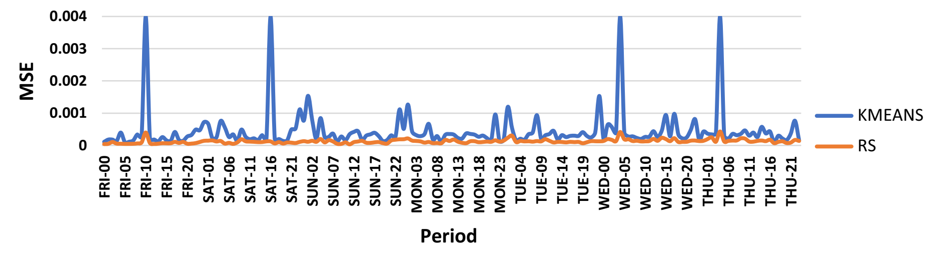


Fig. 7 MSE value for trips origin

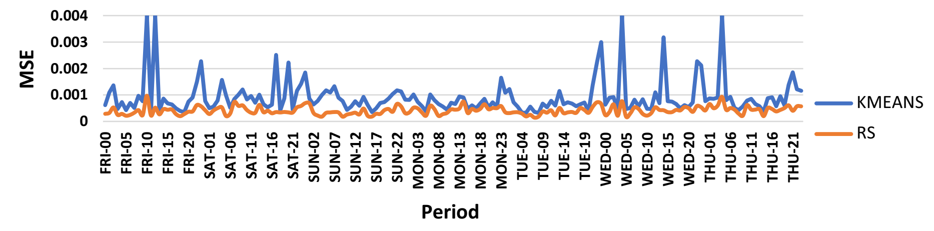


Fig. 8 MSE value for trips destination

where N is the number of data points in the dataset, x is the dataset, c is the centroid, and p is the partition.

The RS algorithm we executed used K-means as shown in the R script [37]. However, RS is a generic algorithm which can be used with any centroid clustering algorithm. When executing the RS algorithm in R, we set the iterations to 5000 as shown in Table 3. Additionally, we plot WSS using fviz_nbclust function; then we manually set the optimal value for the number of clusters k. To compare the MSE value before clustering and MSE value after clustering, we printed the MSE value twice. Once before executing the RS function and once after each iteration of the RS function. Figure 7 shows the MSE value before and after executing the RS algorithm for the 168 trips origin CSV files.

We can observe that MSE value was reduced significantly when applying RS algorithm along with K-means. The RS MSE values ranged from 0.000031 to 0.0000431. While K-means MSE values ranged from 0.000096 up to 0.307779. The RS algorithm successfully reduced the MSE value for every dataset clustered. Figure 8 shows the MSE value before and after executing the RS algorithm for the 168 trips destination CSV files. We can observe that the MSE value was reduced significantly when applying RS algorithm along with K-means. The RS MSE values ranged from 0.00014 to 0.000971. While K-means MSE values ranged from 0.000321 up to 0.308451. The RS algorithm successfully reduced the MSE value for every dataset clustered.

Processing time

We used proc.time function in our R scripts to calculate the processing time of each algorithm. It works as a stop-watch where it first initializes the starting time, then after we execute the algorithm, it calculates the ending time and subtracts the starting time from the ending time. The difference (i.e., elapsed time) is calculated in seconds. We extracted the processing time for the three algorithms HDBSCAN, K-means, and RS. In

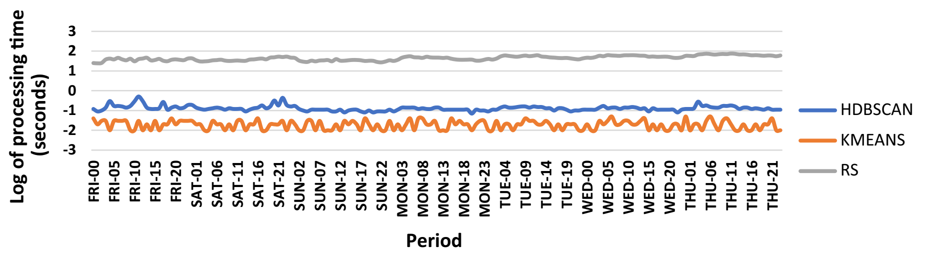


Fig. 9 Processing time for trips origin

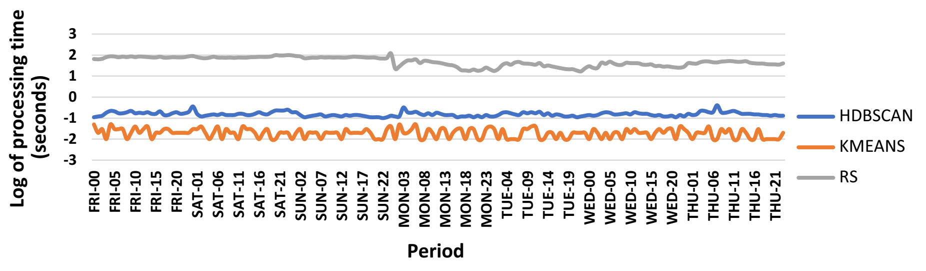


Fig. 10 Processing time for trips destination

addition, the execution time was calculated twice, once for taxi trips origin and once for taxi trips destination.

However, the processing time difference between the three algorithms was large. Therefore, we used the log-scale for processing time to reduce the variability in the visualization. Figure 9 shows the processing times of HDBSCAN, K-means, and RS for taxi trips origin CSV files. We can observe that the processing time for the RS algorithm was more significant than the other algorithms. This amount of time was due to the number of iterations made by RS algorithm to swap centroids and reduce the MSE value, thus, enhance the clustering outcome. Also, the number of iterations include both successful and unsuccessful swaps. The primary challenge for the RS algorithm is that the number of swaps is unknown and can depend on multiple factors such as the size of the neighborhood α and the number of clusters k. For the HDBSCAN and K-means, the processing time for K-means was less than HDBSCAN for all periods. Figure 10 shows the processing time of HDBSCAN, K-means, and RS for taxi trips destination CSV file. We can observe that the processing time for the RS algorithm was more significant than the other algorithms. This amount of time was due to the number of iterations made by RS algorithm. For the HDBSCAN and K-means, the processing time for K-means was less than HDBSCAN for all periods.

Clusters visualization (Tableau)

Based on the comparative analysis, it was apparent that the RS clustering algorithm was more efficient than other algorithms in terms of clusters quality. Therefore, we decided to use the RS algorithm for our cluster's visualizations. We used Tableau Desktop Software [22] to plot the taxi trips clusters on Porto city map. First, we created a connection and defined every CSV file as a data source for the Tableau workbook. Then we identified each pair of longitude and latitude coordinates as geographical attributes, then we created a new datasheet, and we plotted the longitude and latitude attributes as columns and rows respectively. To visualize the clusters on Porto city map, we added a new cluster column as an attribute in the datasheet; then we assigned the color property to this attribute to distinguish different clusters while viewing the map. in addition, we used the district feature we extracted previously in our maps, we added a new column for the district.

Figure 11 shows all taxi trips that started on Friday at 06:00 P.M. The map shows three clusters of trips and displays the names of the district in Porto city. The districts names were derived from the district feature which we extracted previously as shown in Table 2. We can notice that most taxi trips started from central areas (i.e., the blue cluster) like Se and Sao Nicolau. The yellow cluster indicated that a high number of trips began in west areas like Matosinhos, while the red cluster represented the trips which began in east areas like Bonfim and Campanha but with less density. Moreover, when the user picked one of the trips, the map displayed a white box with the longitudinal coordinates besides the area which provided more insights into the taxi trips clusters.

Figure 12 shows all taxi trips that ended on Monday at 06:00 A.M. The map shows two clusters of trips, the blue cluster which represented trips ended in central and eastern areas, and the yellow cluster which represented trips ended in west and north areas. We can notice that the blue cluster has a low density, while the yellow cluster has a high density for trips ending in Porto international airport. Table 4 shows details about extracted factors for each of the algorithms used.

Taxi trips heatmap

To help taxi drivers and passengers identify areas with high or less traffic, we created an R script to generate a heatmap for taxi trips based on their frequency. First, we calculated the number of trips for each period (i.e., weekday and hour of the day), then we imported the counts into an R script. After that, we used two R packages, ggthemes

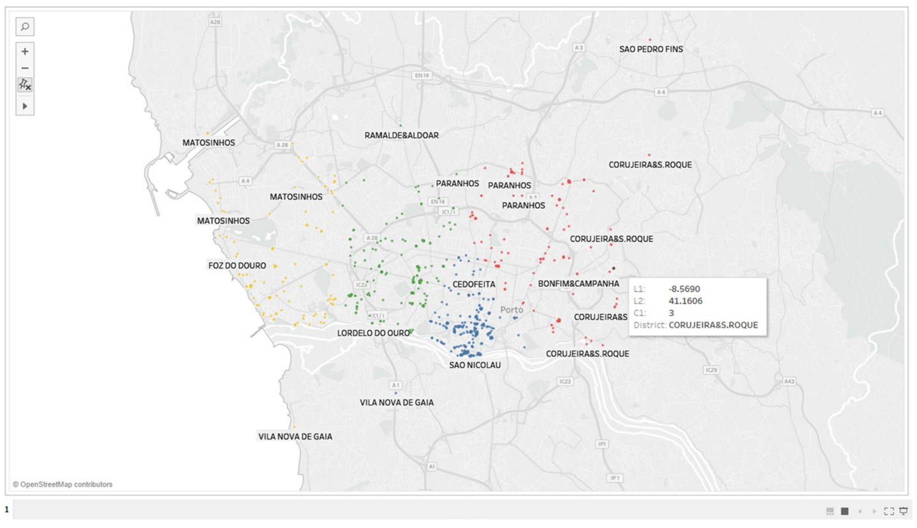


Fig. 11 Trips origin clusters on Friday at 6:00 P.M.

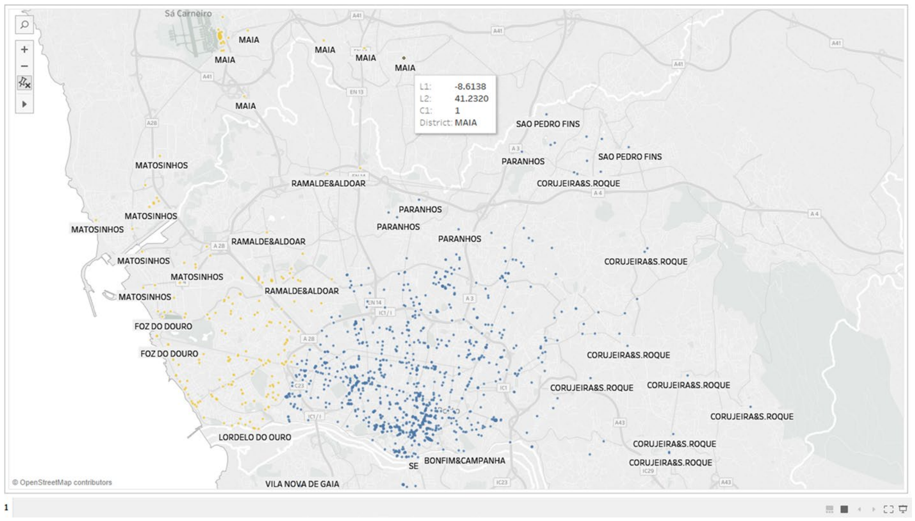
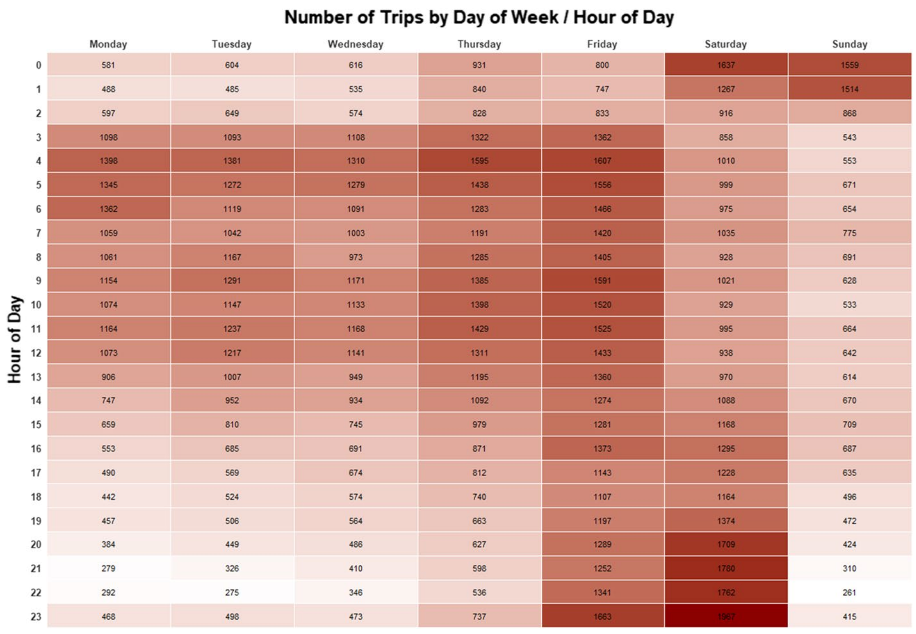


Fig. 12 Trips destination clusters on Monday at 6:00 A.M.

Table 4 Extracted factors for each algorithm

Algorithm	Processing time (seconds)	Silhouette coefficient	MSE
HDBSCAN [31]	Yes	Yes	No
RS [5]	Yes	Yes	Yes
K-means [6]	Yes	No	Yes



Hour of Day	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
0	581	604	616	931	800	1637	1559
1	488	485	535	840	747	1267	1514
2	597	649	574	828	833	916	868
3	1098	1093	1108	1322	1362	858	543
4	1398	1381	1310	1595	1607	1010	553
5	1345	1272	1279	1438	1556	999	671
6	1362	1119	1091	1283	1466	975	654
7	1059	1042	1003	1191	1420	1035	775
8	1061	1167	973	1285	1405	928	691
9	1154	1291	1171	1385	1591	1021	628
10	1074	1147	1133	1398	1520	929	533
11	1164	1237	1168	1429	1525	995	664
12	1073	1217	1141	1311	1433	938	642
13	906	1007	949	1195	1360	970	614
14	747	952	934	1092	1274	1088	670
15	659	810	745	979	1281	1168	709
16	553	685	691	871	1373	1295	687
17	490	569	674	812	1143	1228	635
18	442	524	574	740	1107	1164	496
19	457	506	564	663	1197	1374	472
20	384	449	486	627	1289	1709	424
21	279	326	410	598	1252	1780	310
22	292	275	346	536	1341	1762	261
23	468	498	473	737	1663	1967	415

Fig. 13 Heatmap for taxi trips

[41] package to create a friendly theme heatmap, and ggplot2 [42] package to generate the heatmap based on the selected theme and the taxi trips frequencies. We built one heatmap to represent both trips origins and trips destination as the number of trips was the same in both CSV files. Figure 13 shows the heatmap of taxi trips for each weekday and hour of the day. The darker the color was, the higher traffic we had for that period. We can notice that the highest number of trips was on Saturday in midnight hours (i.e., 20:00–00:00), this was likely because that most people were spending more time outside their homes as the weekend was coming ahead. Moreover, for the all weekdays except Friday, we can observe that most traffic occurred between early morning hours and afternoon hours (i.e., 3:00–14:00). On Friday, most of the day had high traffic starting from early morning hours until midnight.

Sequential pattern mining

To help the transportation authorities to get more insights into the flow of taxi trips among different districts, we applied the sequential pattern mining technique. Sequential pattern mining is used to discover patterns in datasets where data is displayed as a sequence. In our dataset, a set of taxi trips is considered a time series sequences of districts, each trip travels and navigates into a different district over time. Each time series sequence should have an alphabet which represents the items in that sequence; in our study our alphabet contained seven elements; each element was the name of a district in Porto city.

The first step in our analysis was to prepare the CSV file for sequential pattern mining. The CSV file had to be in a long format where each taxi trip sequence was split into subsequences; each subsequence had one event. All subsequences of a given trip had to be ordered in the CSV file according to their timestamp. For example, if there was a taxi trip which navigated through five districts, T1 = {Santo Ildefonso, Se, Vitoria, Miragaia, Vitoria}. Then this trip was displayed in the CSV file as follows:

In addition, in the CSV file, it was assumed that all events had a size of 1 as each district was a separate event. All trips with a duration of 5 min were extracted and converted from wide format into a long format as shown in Table 5. To convert the CSV file from wide to long format, we created a PL/SQL script which read all districts in a row, then inserted each district in that row as a new record in the CSV file. When adding districts, the script ignored repeated districts in the same row as many trips stayed in the same district for a while. The script avoided patterns which had repeated districts such as {Paranhos, Paranhos}. In the CSV file, some trips navigated over two districts, while

Table 5 Trip sequence distribution over time

Trip	Time	Event	Size
T1	1	Santo Ildefonso	1
T1	2	Se	1
T1	3	Vitoria	1
T1	4	Miragaia	1
T1	5	Vitoria	1

Table 6 Patterns generated by SPADE algorithm

Sequence	Support value
<{VITORIA}, {SE}>	0.115662
<{MASSARELOS}, {LORDELO DO OURO}>	0.099603
<{VITORIA}, {MIRAGAIA}>	0.098
<{SE}, {SANTOILDEFONSO}>	0.095
<{SE}, {VITORIA}>	0.094

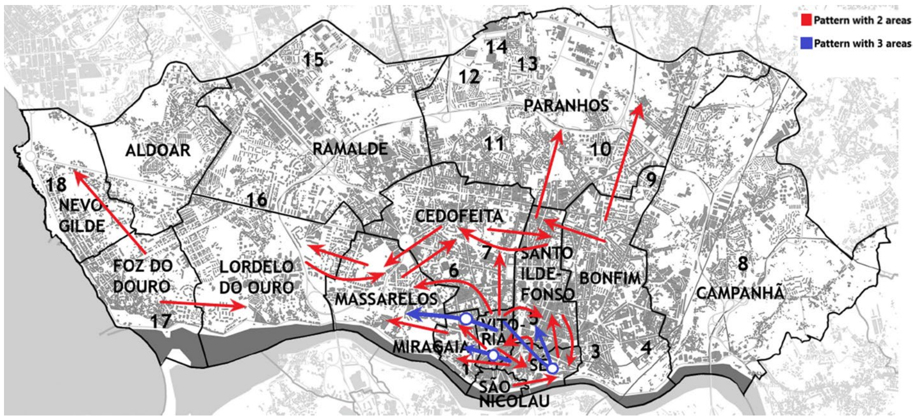


Fig. 14 Taxi trips flow over different districts in Porto city

other trips navigated over four districts. After preparing the CSV file, there was 5542 trips and 15,921 rows.

Next, we created an R script using arules [43–45] and arulesSequences [46] packages. The CSV file which had the taxi trips in the long format was imported, then the script called cspade function which was based on the sequential pattern discovery using equivalence classes (SPADE) algorithm [47]. SPADE algorithm builds a lattice tree for the time series sequence; then it performs three scans to decompose the lattice tree into smaller subtrees processed separately. After that, it generates patterns with a support value associated with each pattern. The cspade function took one parameter which was the minimum support value, if a generated sequence support value was greater than the minimum support value, then it was a frequent sequence (i.e., pattern). However, setting an improper value for minimum support value could cause poor performance in terms of results and execution time. For example, if the script sets a high minimum support value, this could generate a few patterns and ignore some significant patterns in the time series sequence. And if it sets a low minimum support value, the algorithm could generate many insignificant patterns.

In the R script, when we set the minimum support value to 0.01, the algorithm generated 146 patterns. In order to shortlist the generated patterns and visualize them on the Porto city map, we set the minimum support value to 0.04 which generated 41 patterns in the end. From 41 patterns, 14 patterns were single patterns with one district, 24 patterns were two-districts and 3 patterns were three-districts patterns.

The Table 6 shows the top five generated two-districts patterns with the highest support values.

Furthermore, to have a better understanding of patterns generated, we visualized 25 patterns with highest support values to analyze the flow of taxi trips among Porto city's districts. Figure 14 shows Porto city urban distinctions based on the author's work [23] along with our generated patterns. Each pattern was plotted as an arrow to define its flow and the navigation districts. The arrows in red represented trips patterns over two districts, and the blue arrows represented trip patterns over three districts. It was apparent that most of the patterns generated were trips with two districts. Moreover, most trips patterns were flowing in the Central areas like Sao Nicolau, Se, Vitoria, Miaragaia, Santo Ildefonso. It meant that these districts have a lot of facilities that keep people and tourists actively engaged. However, we identified some patterns heading North to Paranhos and West to Lordelo Do Ouro, Foz Do Douro, and Matosinhos.

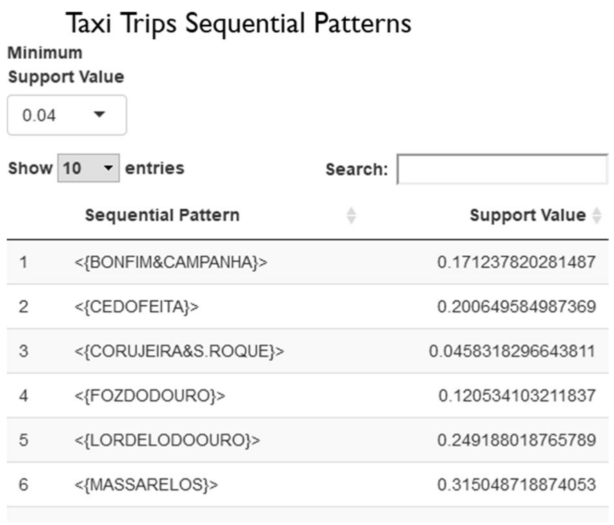
Interactive visualization analysis

To provide an interactive visualization which could help people to learn more from our dataset, and to give the chance to explore the dataset by passing different parameters, we created R shiny [48] slides for sequential pattern mining technique applied previously. By using R shiny, applications which run through web browsers can be created without the need to learn other programming languages.

For the sequential pattern mining, we took the same R script we created previously. However, the script used the shinyapp function which had two parts, ui, and server. The ui part defined what appeared to the user in the slides, and server part defined what the user wanted to do internally to visualize the data. In the ui part, usually, all fields (i.e., parameters) which were visible the first time we run our R shiny presentation were defined. The idea behind using R shiny was to create an interactive sequential pattern mining with dynamic parameters instead of static values. For instance, instead of passing a minimum support value of 0.04 to the cspade function as the script did previously, we could create an input parameter which takes any given value and retrieves sequential patterns accordingly. In this case, the user can run the SPADE algorithm several times, each time with a different support value.

This approach was applied by creating an input parameter in the ui part; the input field was a dropdown list where the user had the option pick one of the displayed support values which were 0.01, 0.02, 0.03, and 0.04.

After that, the script called the cspade function in the server part and passed the value of the input field created. It plotted the patterns and their support values in a table on the R shiny slide. Figure 15 shows the slides generated from R shiny script when the script runs the presentation. As it can be noticed from the slide, R shiny gave the user the ability to view any number of patterns per page. The default value for the minimum support value was 0.04, and once the script runs the presentation R shiny would display all the patterns with support value larger or equal to 0.04. Additionally, the user could sort patterns based on their support value in an ascending or descending order. And if the user wanted to search for a specific pattern, he could enter the district name in the search box.


Taxi Trips Sequential Patterns

Minimum Support Value: 0.04

Show 10 entries Search:

	Sequential Pattern	Support Value
1	<{BONFIM&CAMPANHA}>	0.171237820281487
2	<{CEDOFEITA}>	0.200649584987369
3	<{CORUJEIRA&S.ROQUE}>	0.0458318296643811
4	<{FOZDODOURO}>	0.120534103211837
5	<{LORDELODOOURO}>	0.249188018765789
6	<{MASSARELOS}>	0.315048718874053

Fig. 15 R shiny slide for sequential pattern mining

For example, if the user wanted to display all the patterns where Se district was part of, he could fill the search box with the value "Se", and press return to refresh the page and view the patterns again. Furthermore, when the user selected different support value from the list like 0.01, it was apparent that more patterns were generated, and the number of pages increased. This increase in pattern numbers was because of the cspade function in the server part; it generated more insignificant patterns when the user provided a low support value as discussed earlier.

Overall, R shiny interactive visualization could provide the ability to apply trajectory mining techniques like sequential pattern mining in a dynamic mode. Passing a parameter each time would implement the same method but with different behavior and outcome. Therefore, transforming data mining techniques into the interactive visualization mode could enhance the understanding of their mechanism.

Discussion

In this section, we explain how our system and results are beneficial to taxi drivers, passengers, and the transportation management authorities in Porto city.

Transportation recommendations and insights

Our system was able to identify the traffic in different districts and timeframes. In addition, it consisted of an efficient clustering algorithm (i.e., RS algorithm) which was able to detect traffic in Porto city districts. Moreover, RS was capable of visualizing traffic with various densities for all weekdays and hours of the day. For example, we could observe that more people are arriving at Porto international airport on Monday at 6:00 A.M. as shown in Fig. 12. Clustering and visualizing taxi trips data can provide useful guidelines for taxi drivers, passengers, and Porto transportation authorities. These guidelines can be utilized by different parties in the city to have more insights into the traffic flow.

In terms of taxi drivers, they can use the heatmap and clusters visualizations to avoid high traffic districts during their trip as they want to take the fastest route to drop off their passenger with no delays. Additionally, they can rely on the origin clusters visualizations (e.g., Fig. 11) to seek high traffic districts when they are vacant. Usually, high traffic districts of trips initiations mean that there is a high demand for taxis in that area; thus, taxi drivers can likely find passengers. In terms of passengers, the system should be able to decrease the waiting time. Our heatmap and destination clusters visualizations (e.g., Fig. 12) can help passengers to identify taxi availability in a specific area. For example, if the traffic status of trips destination is high (e.g., Porto international airport), this means that many taxis are ending their trips at that place. Thus, passengers are more likely to find vacant taxis in this area and timeframe.

Providing recommendations and guidelines for taxi drivers can save them time and effort. Additionally, It can maximize their profit by providing them with traffic status on a given route. Meanwhile, it can reduce the waiting time for passengers who are trying to find a vacant taxi.

Urban computing

Trajectory mining techniques can provide knowledge of traffic flow and taxi trips behavior in the city. This knowledge can be utilized to enhance the services provided to people. For example, sequential pattern mining technique used previously can help authorities to identify the taxi trips flow over various regions. High flow areas can be a good indication of the functionality of certain areas (e.g., educational, business, industrial, etc.). For example, most discovered patterns were located in the central city areas; this could be a strong sign that most facilities such as universities, shopping centers, hospitals, and restaurants are not distributed over Porto city districts. Therefore, it can be concluded that most places which attract people are in the central areas of Porto city, not in the north or west areas. This information can help transportation authorities to reduce traffic in the central areas by providing a reliable bus and subway services. Furthermore, they can plan to establish new infrastructures in urban areas to avoid road networks issues such as traffic congestions which wastes a lot of gas and cause serious environmental problems like air and noise pollution.

Human and machine intelligence combination

Our study used visual analytics in all applied techniques. Heatmaps were created to visualize the traffic for each district, weekday, and hour of the day. In clustering, Tableau maps were used to visualize trips clusters and their density. In sequential pattern mining, Porto city map was used to visualize taxi trips flow among various districts in the city. Furthermore, we applied R shiny interactive visualizations to show how the user can choose different parameters and get a better understanding of sequential patterns. Visualization is an essential technique for intelligent transportation systems; it can enhance the knowledge of moving vehicles and traffic data. Visualization analytics can provide a useful interpretation of traffic data which can help humans and decision makers in accident monitoring, route planning, and traffic jams reduction.

Limitations and future work

We applied three clustering algorithms in our system, HDBSCAN, K-means, and RS. The RS algorithm outperformed the HDBSCAN in terms of quality of clusters. Moreover, the RS algorithm outperformed the K-means in terms of MSE error reduction. However, the only drawback of the RS algorithm was the processing time. The RS algorithms iterated and calculated the MSE value repeatedly after every K-means execution until no significant reduction was detected. This process consumed time as we had many iterations depending on some factors like the size of the dataset and the number of clusters.

Multiple methods can be applied to reduce the processing time of the RS clustering algorithm. One method is to apply the RS algorithm with a different clustering algorithm like Fast K-means [49]. The RS algorithm is a generic centroid algorithm which swaps centroids to tune-up the generated clusters. In our experiments, we applied RS with the traditional K-means. In K-means, a full search was applied for every iteration. The K-means full search process involves the calculation of the distance between all points and their centroids in each iteration. Full search is done even if the distance between the points and their centroid is not changing. This type of search consumes time as it scans all points and centroids even if there were some static clusters (i.e., no centroid change). However, Fast K-means has a different mechanism. This algorithm measures the activity classification for its clusters in every iteration, after that it labels the points and centroids which have no change in distance as static. Then, in the next iteration, it performs a partial search for the active clusters only. Thus, in Fast K-means, the processing time will be improved as the distance calculations between points and centroids in static clusters will be skipped.

Another method to improve the RS processing time is to apply each iteration in parallel. In every RS iteration, the algorithm swaps the centroids then calls K-means clustering. We can reduce the computation of every K-means execution by utilizing some parallel processing platforms such as MapReduce. Moreover, instead of executing K-means on a single node, the MapReduce can execute K-means on thousands of nodes in parallel [50]. Thus, MapReduce will significantly improve the processing time for both K-means and RS clustering.

A significant challenge in our study was finding an approach to extract the description (i.e., district) for each longitudinal coordinate. Google services were used to extract the district based on the longitude and latitude of each trip. However, it was a time-consuming process, and it produced an imbalanced dataset where 90% of districts were assigned to "Porto". Therefore, we divided the Porto city into 18 equiangular by using Google Maps. Still, dividing the city into rectangular districts is not a practical approach as neighborhoods tend to have different shapes. A good approach is to rely on geometrical tools to divide the city into uniform shapes such as polygons.

Another promising approach is to build a robust interactive visualization application. By using R shiny, many slides of the presentation can be created where each slide represents a separate trajectory mining technique such as clustering and sequential pattern mining. We plan to create dynamic slides which can be fed by any online trajectory mining datasets. This application can help users to understand the mechanism of different techniques when they have the chance to interact with various visualizations.

Conclusions

Our system was built based on RS clustering and sequential pattern mining. In terms of sequential patterns, the system was able to detect 146 patterns. By minimizing the minimum support value to 0.04, we detected 41 patterns. Most of these patterns flew over the central districts of the city. In addition, three patterns had a sequence of 3 districts, while the rest of the patterns had a sequence of two districts.

We applied three clustering algorithms for taxi trips origin and destination points. These algorithms were HDBSCAN, K-means, and RS. The RS algorithm outperformed HDBSCAN in terms of the quality of clusters. The RS algorithm outperformed K-means in terms of the mean of square error (MSE) reduction. While the K-means processing time was less than HDBSCAN, the processing time for the RS clustering was much higher than both HDBSCAN and K-means. Visualizing taxi trips RS clusters on Porto city map and including the city districts was fruitful. We were able to detect some districts with high traffic such as Porto international airport and Matosinhos west area. The taxi trips heatmap was useful in identifying taxi trips behavior and periods with more traffic. For example, we could know that Saturday midnight hours are busy. Moreover, in terms of day time hours, it is busier in the weekdays while it is less busy on Sunday.

The proposed system can be used to provide vacant taxis with guidelines of the traffic status in areas where taxis start (origin). Furthermore, it can guide occupied taxis to avoid areas with high traffic (route prediction) which mitigates the traffic congestion issue. Moreover, this system can help passengers to identify areas where they can find vacant taxis (destination). Overall, this system shows the feasibility of applying trajectory data mining techniques on taxi trips dataset to enhance the road network services and reduce traffic jams in Porto city.

Abbreviations

RS: Random Swap clustering; GPS: Global Position Systems; RFID: radio frequency identification; ED: Euclidean distance; PPM: periodic pattern mining; ROI: region-of-interest; KNN: K-nearest neighbor; WSS: within sum of squares.

Acknowledgements

This research is supported by Carleton University, in Ottawa, ON, Canada.

Authors' contributions

Both authors read and approved the final manuscript.

Authors' information

Rami Ibrahim is a graduate student at the School of Information Technology, Carleton University. He has 10+ year of industry experience by working on database systems. His research interests include Data Analytics, Big Data, Database Systems. He has produced two research papers that got accepted recently in IEEE conferences.

M. Omair Shafiq is an Assistant Professor at the School of Information Technology, Carleton University. His research interests include Data Modeling, Big Data Analytics, Services Computing, Machine Learning and Cloud Computing. He received NSERC Postdoctoral Fellowship Award and Mitacs Elevate Postdoctoral Fellowship Award in 2015–2016 competition, NSERC Vanier CGS Scholarship in 2012, Alberta-Innovates Technology Futures (AITF) Scholarships for PhD and Master studies in 2011 and 2010, J.B. Hyne Research Innovation Award from University of Calgary in 2012, Departmental Research Award from University of Calgary in 2009 and 2010 and Teaching Excellence Award from University of Calgary in 2011. He has published over 50 peer-reviewed publications in journals, book chapters, conferences and workshops, served in technical program committee of over 30 conferences and workshops, co-organized more than 8 conference and workshops.

Funding

Carleton University

Availability of data and materials

We used publicly available dataset [24].

Competing interests

The authors declare that they have no competing interests.

Received: 9 November 2018 Accepted: 6 May 2019
Published online: 13 May 2019

References

1. Zheng Y. Trajectory data mining: an overview. ACM Trans Intell Syst Technol. 2015. https://doi.org/10.1145/2743025.
2. Zheng Y, Capra L, Wolfson O, Yang H. Urban computing: concepts, methodologies, and applications. ACM Trans Intell Syst Technol. 2014. https://doi.org/10.1145/2629592.
3. Lin M, Hsu WJ. Mining GPS data for mobility patterns: a survey. Perv Mobile Comput. 2013. https://doi.org/10.1016/j.pmcj.2013.06.005.
4. Zhong RY, Huang GQ, Lan S, Dai QY, Chen X, Zhang T. A big data approach for logistics trajectory discovery from RFID-enabled production data. Int J Prod Econ. 2015. https://doi.org/10.1016/j.ijpe.2015.02.014.
5. Fränti P. Efficiency of random swap clustering. J Big Data. 2018;5:13. https://doi.org/10.1186/s40537-018-0122-y.
6. Hartigan JA, Wong MA. Algorithm AS 136: a k-means clustering algorithm. J R Stat Soc. 1979;28(1):100–8.
7. Takimoto Y, Sugiura K, Ishikawa Y. Extraction of frequent patterns based on users' interests from semantic trajectories with photographs. 2017. IDEAS'17, July 12–14, 2017, Bristol, United Kingdom.
8. Feng MAO, Minhe JI, Ting LIU. Mining spatiotemporal patterns of urban dwellers from taxi trajectory data. Berlin Heidelberg: Higher Education Press and Springer-Verlag; 2015.
9. Qi H, Liu P. Mining taxi pick-up hotspots based on spatial clustering. 2018. In: IEEE SmartWorld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, 2018, p. 1711–17. https://doi.org/10.1109/smartworld.2018.002907.
10. Yue Y, Zhuang Y, Li Q, Mao Q. Mining time-dependent attractive areas and movement patterns from taxi trajectory data. 2008.
11. Liu D, Cheng SF, Yang Y. Density peaks clustering approach for discovering demand hot spots in city-scale taxi fleet dataset. In: 2015 IEEE 18th international conference on intelligent transportation systems, Las Palmas, 2015, p. 1831–6. https://doi.org/10.1109/itsc.2015.297.
12. Saptawati GA. Spatio-temporal mining to identify potential traffic congestion based on transportation mode. In: 2017 International conference on data and software engineering (ICoDSE).
13. Zheng L, Xia D, Zhao X, Tan L, Li H, Chen L, Liu W. Spatial–temporal travel pattern mining using massive taxi trajectory data. Physica A. 2018;501:24–41. https://doi.org/10.1016/j.physa.2018.02.064.
14. Naji HAH, Wu C, Zhang H. Understanding the impact of human mobility patterns on taxi drivers' profitability using clustering techniques: a case study in Wuhan, China. Information. 2017;8:67.
15. Fournier-Viger P, Lin JC, Kiran RU, Koh YS, Thomas R. A survey of sequential pattern mining. Data Sci Patt Recogn. 2017;1(1):54–77.
16. Bermingham L, Lee I. Spatio-temporal sequential pattern mining for tourism sciences. Procedia Comput Sci. 2014;29:379–89.
17. Chen W, Guo F, Wang FY. A survey of traffic data visualization. IEEE Trans Intell Trans Syst. 2015;16(6):2970–84.
18. Ferreira N, Poco J, Vo HT, Freire J, Silva CT. Visual exploration of big spatio-temporal urban data: a study of New York city taxi trips. IEEE Trans Visual Comput Graph. 2013;19(12):2149–58.
19. Xiong H, Chen L, Gui Z. A Web-based platform for visualizing spatiotemporal dynamics of big taxi data. In: International archives of the photogrammetry, remote sensing & spatial information sciences. 2017.
20. Jianqin Z, Peiyuan Q, Yingchao D, Mingyi D, Feng L. A space-time visualization analysis method for taxi operation in Beijing. J Vis Lang Comput. 2015. https://doi.org/10.1016/j.jvlc.2015.09.002.
21. Hu Y, Yang Y, Huang B. A comprehensive survey of recommendation system based on taxi GPS trajectory. In: International conference on services science. 2015. https://doi.org/10.1109/icss.2015.31.
22. Tableau Desktop version 2019.1 Copyright © 2019 Tableau Software Inc. http://www.tableau.com.
23. Pereira VB. Urban distinctions: class, culture and sociability in the city of Porto. Int J Urban Reg Res. 2018;42(1):126–37.
24. Moreira-Matias L, Gama J, Ferreira M, Mendes-Moreira J, Damas L. Predicting taxi–passenger demand using streaming data. IEEE Trans Intell Trans Syst. 2013;14(3):1393–402.
25. Dua D, Taniskidou E. UCI machine learning repository. https://archive.ics.uci.edu/ml/datasets/Taxi+Service+Trajectory+-+Prediction+Challenge,+ECML+PKDD+2015. Irvine: The University of California, School of Information and Computer Science; 2017.
26. DB Browser for SQLite version 3.9.1. http://sqlitebrowser.org/.
27. Delimit version 3.7.5 (x64), copyright (c) 2017. www.delimitware.com.
28. Google Maps. (2018). Porto, Portugal. Retrieved from https://www.google.ca/maps/place/porto+portugal.
29. Oracle Application Express (APEX) version 4.0.2.00.09, Oracle 11 g. Oracle Corporation, California, U.S. https://apex.oracle.com.
30. Grothendieck G. sqldf: Manipulate R data frames using SQL. R package version 0.4-11. https://CRAN.R-project.org/package=sqldf. 2017.
31. Campello RJ, Moulavi D, Sander J. Density-based clustering based on hierarchical density estimates. 2013. https://doi.org/10.1007/978-3-642-37456-2_14.
32. Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. New York: AAAI Press; 1996. p. 226–31.
33. R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/. 2018.

34. Hahsler M, Piekenbrock M. dbscan: Density based clustering of applications with noise (DBSCAN) and related algorithms. R package version 1.1-2. https://CRAN.Rproject.org/package=dbscan. 2018.
35. Thorndike RL, Psychometrika (1953) 18: 267. https://doi-org.proxy.library.carleton.ca/10.1007/BF02289263.
36. Kassambara A, Mundt F. Factoextra: extract and visualize the results of multivariate data analyses. R package version 1.0.5. https://CRAN.R-project.org/package=factoextra. 2017.
37. University of Eastern Finland, Finland. http://cs.uef.fi/sipu/soft/RS.R.
38. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987;20:53–65.
39. Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K. Cluster: cluster analysis basics and extensions. R package version 2.0.7-1. 2018.
40. Oksanen J, Blanchet FG, Kindt R, Legendre P, Minchin PR, O'hara RB, Simpson GL, Solymos P, Stevens MH, Wagner H, Oksanen MJ. vegan: community ecology package. R package version 2.5-4. https://CRAN.R-project.org/package=vegan. 2019.
41. Jeffrey B. Arnold. ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'. R package version 4.1.0. https://CRAN.R-project.org/package=ggthemes. 2019.
42. Wickham H. ggplot2: elegant graphics for data analysis. New York: Springer; 2016.
43. Hahsler M, Buchta C, Gruen B, Hornik K. Arules: mining association rules and frequent itemsets. R package version 1.6-1. https://CRAN.R-project.org/package=arules. 2018.
44. Hornik K, Grün B, Hahsler M. Arules—a computational environment for mining association rules and frequent item sets. J Stat Softw. 2005;14(15):1–25. https://doi.org/10.18637/jss.v014.i15.
45. Hahsler M, Chelluboina S, Hornik K, Buchta C. The arules R-package ecosystem: analyzing interesting patterns from large transaction datasets. J Mach Learn Res. 2011; 12:1977–1981. http://jmlr.csail.mit.edu/papers/v12/hahsler11a.html.
46. Buchta C, Hahsler M, Diaz D. Arulessequences: mining frequent sequences. R package version 0.2-20. https://CRAN.R-project.org/package=arulesSequences. 2018.
47. Zaki MJ. An efficient algorithm for mining frequent sequences. New York: SPADE; 2001.
48. Chang W, Cheng J, Allaire J, Xie Y, McPherson J. Shiny: web application framework for R. R package version 1.1.0. https://CRAN.R-project.org/package=shiny. 2018.
49. Kaukoranta T, Franti P, Nevalainen O. A fast exact GLA based on code vector activity detection. IEEE Trans Image Process. 2000;9(8):1337–42. https://doi.org/10.1109/83.855429.
50. Boukhdhir A, Lachiheb O, Gouider MS. An improved mapReduce design of kmeans for clustering very large datasets. In: 2015 IEEE/ACS 12th international conference of computer systems and applications (AICCSA), Marrakech, 2015, p. 1–6. https://doi.org/10.1109/aiccsa.2015.7507226.