UNIVERSITÀ DI PISA

Master's Degree Course in Humanities Computing

REPORT

Visual Analytics framework for borders of human mobility

Candidate

Ahmed Adeyemo

Supervisor

Prof. Salvatore Rinzivillo

Co-supervisor

Prof. Dino Pedreschi

Academic Year 2017/2018

Abstract

This thesis illustrates and discusses the design and development procedures of, a small web framework for interactive and visual analytics applied on a real-life situation in particular the borders of human mobility. The availability of massive network and mobility data from diverse domains fostered the analysis of human behaviours and interactions. This data availability leads to challenges in the knowledge discovery community. Several different analysis have been performed on the traces of human trajectories, such as understanding the real borders of human mobility or mining social interactions derived from mobility and vice versa.

The Web Application **Mobility Borders** tackles the objective of giving the data analyst the opportunity and instruments to visually explore and interact with the borders of human mobility data with the end goal of acquiring knowledge and discovering new insights. In the pages, ahead this thesis discusses all the steps made to the development of the final framework, considering all aspects which include technological and analytical elements. Particularly, these steps include the presentation of a theoretical framework which illustrates the abstract model of the data, the formalization of the analytical functions based on this model, the design and the implementation of the framework. A real-life situation is taken in consideration by utilizing a large dataset of human trajectories, a GPS dataset from more than 150k vehicles in Italy(Tuscany). Some examples of analysis performed on the data are also provided in order to show the potential of the analytical tool. The resulting web framework is built using several technologies, such as Node.js, HTML, CSS and D3.js, the JavaScript library which is nowadays the standard for web-based data visualization projects.

Contents

1 Introduction 1

2 Related works 4
2.1 Mobility Data 5
2.2 Data Mining 6
2.3 Mobility Data Mining 8
2.3.1 Local trajectory patterns 8
2.3.2 Global trajectory models 14
2.4 Visual Analytics of Movement (Methods, tools, and procedures) 23
2.4.1 Visualizing trajectories 24
2.4.2 Looking inside trajectories 31
2.4.3 Bird's-eye view on movement: generalization and aggregation 33
2.5 A quick view on Network-Based Analysis of Human Mobility 39

3 Web Application 45
3.1 Data Visualization 45
3.1.1 Interactive Visualization 46
3.2 Technologies 48
3.2.1 D3.js 49
3.2.2 Node.js 53
3.2.3 TopoJSON 54
3.2.4 HTML 5 55
3.2.5 CSS3 57
3.3 Architecture 57
3.3.1 Data input 59
3.3.2 Data pre-processing 62
3.3.3 Final data format 71

4 Mobility Borders 73
4.1 Map Visualization 74

4.2 Control Panel 76

4.3 Analysis 78

5 Conclusion & future Developments 81

Webliography 82

Bibliography 84

Chapter I

Introduction

In this report, we present Mobility Borders, a web framework built for visual exploration of the borders of human mobility designed with the sole purpose providing an interactive experience to explore mobility data

The analysis of movement has been fostered by the wide spread diffusion of wireless technologies, such as the satellite-enabled Global Positioning System (GPS) and the mobile networks data that refer to which cell tower a phone, carried and used by a person was connecting. This results in a huge quantity of data about tens of thousands of people moving along millions of trajectories. This big mobility data provides a new and powerful social microscope, which can help us understand human mobility, and discover the hidden patterns, outliers and models that characterize the trajectories humans follow during their daily activity. This direction of research has recently attracted scientists from diverse disciplines, being not only a major intellectual challenge, but also a domain of great importance that spreads in various sectors such as urban planning, sustainable mobility, transporting engineering, public health, and economic forecasting. The European Discovery project GeoPKDD (Geopraphic Privacy-aware Knowledge Discovery and Delivery), started in 2005, is a precursor in mining human mobility data, which developed various analytical and mining methods for spatio-temporal data just like our case.

In the real world, different events may dramatically change how people move on the territory. Such events may be unpredictable or not frequent, like natural disasters, but most of them are not. The most natural and predictable event is the transition between working and non-working days. During Saturdays and Sundays, people usually abandon their working mobility routines for different paths, obeying to completely different criteria. Another example may be organized human social events, like manifestations in a particular town or sport events.

The aim of this thesis is not only to prove that to mine human mobility and extract from it useful knowledge is necessary to take into account some elements, but it's to show that with the technology advancements in recent years we can interact and explore mobility data in a way that's never been done before giving the opportunity to discover new insights in the human mobility behaviour.

Mobility Borders is a web framework that has been designed and developed with the sole purpose of extracting information from a spatio-temporal aspect confined to our geography to visually interact and gain new insights on human mobility data.

The main contributions are:

- Definition of a data pre-processing methodology to automatically transform original data for analysis and visualization.
- Design and implementation aspects of a user interface using todays web technologies.
- Development of a dynamic, interactive, data visualization using a combination of today's technologies.

The rest of the thesis is organised as follows. In Chapter II we present the related works regarding background topics which are necessary to comprehend the subject of the project and give a complete view of Network-Based Analysis of Human Mobility[1]. Afterwards, we give an extensive overview on how the application is structured and developed, presenting the technologies used, the data pre-processing

[1] Base subject for the web application development

phase and the general architecture (chapter III). Following this, we explain the design choices we made for the user interface design, focusing on how to convey the information to the analyst and the interaction provided. (chapter IV).

Chapter II
Related works

As anticipated in the introduction, there are several works in the field of human trajectories data mining. In this chapter the following pages present the related works regarding trajectories analysis and network-based analysis of human mobility and before going into details its best to give some ground explanation to comprehend the subject, which is defining data mining/mobility data mining applied to spatio-temporal data (section 2.1). Sub sequentially it is also necessary to give an overview of visual analytics elements on Mobility data, which is a field of visual analytics that deals with interactive visualization of movement data (section 2.2). Finally, we can Focus on the works that regard mobility in general by presenting in the final part of the chapter Mobility analytics and visual instruments applied on mobility analytics.

Today, in our extremely complex social systems of the gigantic areas of the 21st century, the observation of the movement patterns and behavioural models of people is used by traffic engineers and city managers to reason about mobility and its sustainability and to support decision makers with trustable knowledge. This very knowledge is precious for an urban planner e.g. to localise new services etc. At a small spatial scale, movement in contexts such as a shopping area or a natural park is an interesting subject of investigation, either commercial purposes, as in geo marketing, or for improving the quality of service.

In all the cases listed above, two key problems occur:

- 1 – How to *collect mobility* data which is often complex and chaotic, social or natural systems made of large populations of moving entities.
- 2 – How to turn this data into *mobility knowledge*, i.e. into useful models and patterns that abstract away from the individual and shed light on collective movement behaviour, pertaining to groups of individuals that it is worth putting into evidence.

Today with the growth of technology a chance to get closer to the dream is offered, by the convergence of two factors or better said two movements:

- The *mobility data* made available by wireless and mobile communication technologies
- *Data mining* - methods for extracting models and patterns from (large) volumes of data.

2.1 Mobility Data

In our everyday actions, the way we move and live, leave digital traces in the information systems of the organisations that provide services through wireless communication networks for mobile appliances. The potential value of this information recording the human activities in a territory is becoming real, because of the increasing pervasiveness and positioning accuracy. The number of mobile phone users worldwide today is 8.3 billion (more than the planets population), with regions, such as Italy, where the number of mobile phones is exceeding the number of inhabitants; in other regions, especially developing countries, the number are still increasing at high speed. On the other hand, the location technologies, such as GSM, UMTS & 4G, currently used by wireless phone operators are capable of providing an increasingly better estimate of a user’s location, while the integration of various positioning technologies proceeds: GPS-equipped mobile devices can transmit trajectories to some service provider (and the

European satellite positioning system Galileo may improve precision and pervasiveness in the future.

The consequence of this scenario is that human activity in a territory may be tracked, not necessarily on purpose, but simply as a side effect of the ubiquitous services provided to mobile users. Thus, the wireless phone network, designed to provide mobile communication, can also be viewed as an infrastructure to gather mobility data, if used to record the location of its users at different times and locations. The wireless networks, whose localisation precision increases while new location-based and context-based services are offered to mobile users, are becoming the very foundation of our territory - in particular, our towns – capable of sensing and, possibly, recording our movements. From this perspective, we have today a chance of collecting and storing mobility data of unprecedented quantity, quality and timeliness at a very low cost.

2.2 Data Mining

Data mining is the process of automatically discovering useful information in large data repositories. Often, traditional data analysis tools and techniques cannot be used because of the massive quantity of data gathered by the automated collection tools, such as point-of-sale, Web logs from e-commerce portals, earth observation data from satellites, genomic data, Sometimes, the non-traditional nature of the data implies that ordinary data analysis techniques are not applicable.

The three most popular and important data mining techniques are predictive modelling, cluster analysis and association analysis

- *Predictive Modelling*: in this technique the goal is to develop classification models, that are able to predict the value of a class label (or target variable) as a function of other variables (explanatory variables); the model is built from historical observation where the class label of each sample is known: once constructed, a classification model is used to predict the class label of new samples whose class is unknown, as in forecasting whether a patient has a given disease based on the results of medical tests.

- *Association analysis*: also, called *pattern discovery*, the objective is to discover patterns that define or describe a strong correlation among features in the data or associations among features that occur frequently in the data.
- Cluster analysis: the goal is to partition a data set into groups of closely related data in such a way that the observations belonging to the same group, or cluster are similar to each other, while the observations belonging to different clusters are not.

Data mining is a step of Knowledge discovery in databases, also known as KDD process for converting raw data into useful knowledge. The KDD process consists of a series of transformation steps:

- Data *pre-processing*, which transforms the raw source data into an appropriate form for the subsequent analysis.
- *Data mining*, which transforms the prepared data into patterns or models: classification, clustering models, association patterns, etc.
- *Post processing* of data mining results, which assesses validity and usefulness of the extracted patterns and models, and presents interesting knowledge to the final users by adding an appropriate visual element.

2.3 Mobility Data Mining

Mobility data mining is, therefore, an area of research aimed at the analysis of mobility data by means of appropriate patterns and models extracted by efficient algorithms; it also aims at creating a discovery process meant for the analysis of mobility with reference to geography, at appropriate scales and granularity. In fact, movement always occurs in a given physical space, whose key semantic features are usually represented by geographical maps; this puts us in the position where the geographical background knowledge about a territory is always essential in understanding and analysing mobility in such territory.

Mobility data mining methods can be divided in two mail classes (Nanni 2013): local patterns and global models. Local patterns identify behaviours and regularities that involve only a small subset of trajectories, meanwhile the global models aim to provide a general characterization of the whole dataset of trajectories, whit the sole goal of defining general laws that regulate data.

2.3.1 Local trajectory patterns

The mobility data mining literature offers several examples of trajectory patterns that can be discovered from trajectory data. Despite this vast variety, most proposal actually respect two basic rules: first, a pattern is interesting (and therefore extracted) only if it is frequent and therefore it involves several trajectories[2], secondly a pattern must always describe the movement in space of objects involved, and not only a spatial or highly abstracted spatial features.

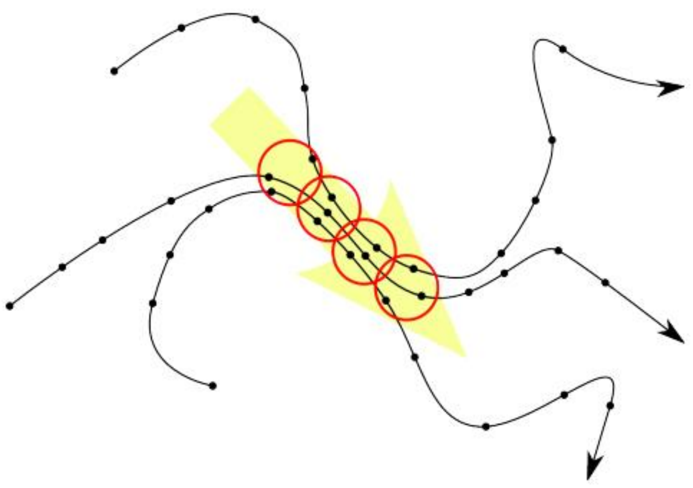
While a trajectory pattern always describes a behaviour that is followed by several moving objects, we can choose whether they should do so together (i.e., during the period), at different moments yet with same timings (i.e., there can be a time shift between the moving objects), or in any way, with no constraints on time.

Absolute time

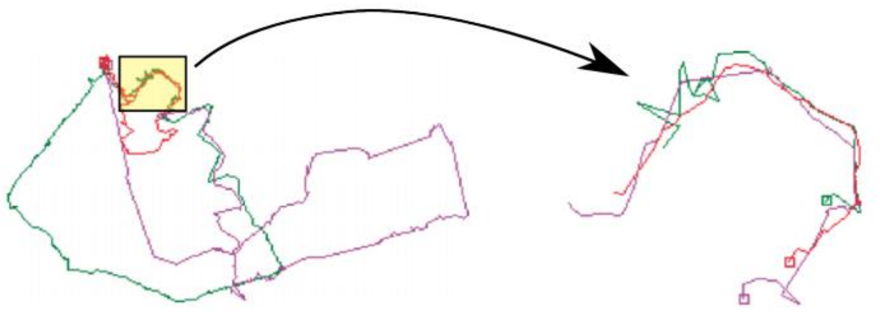
The basic questions that come up when analysing moving objects trajectories is the following: *Are there groups of object that move together for some time?* For instance, in the domain of animal monitoring suck kind of patterns would help to identify possible aggregations, such as herds or simple families, as well as predator-prey relations. In human mobility, similar patterns might indicate groups of people moving together on purpose or forced by external factors, e.g. a traffic jam, where cars are forced to stay close to each other for a long-time period. It's also obvious that larger group indicate that the phenomenon is significant and not a pure coincidence.

The simplest form of a trajectory pattern in literature that answers the question posed above is the trajectory flock. Just as the name suggests, it is a group of moving objects that satisfy three constraints:

- A spatial proximity constraint: within the whole duration of the flock, all its members must be located within a disk of radius r – possibly a different one at each time instant, i.e. the disk moves to follow the flock;
- A minimum duration constraint: the flock duration must be at least k time units;
- A frequency constraint: the flock must contain at least m members.



As you can see in the picture above Figure 1 shows an example of flock, where three trajectories meet at some point (5th time unit), stay close to each other for a period (4 consecutive time units) and then separate (9th time unit).



In **Figure 2** we have an example extracted from a real dataset that contains GPS tracks of tourists in a recreational zone (Dwingelderveld National Park, in Netherland). The left side of the figure depicts 3 trajectories that were involved in the flock, while the right side shows in detail only the segments of trajectories that create the flock

The general concept of moving together or forming a group are implemented by the flock's framework in the simplest way possible: the objects are required to be very close to each other during all the duration of the flock. It's important to be noted that a group might appear also under different conditions. One of these alternatives requires that at each timestamp the objects form a cluster. A notable example is moving clusters, a form of pattern that at each time stamp groups objects by means of density-based clustering (Kalnis, Mamoulis, and Bakiras 2005). Beside their sheer size, the groups formed through this process can also have a relatively large extension and an arbitrary shape. In several contexts this can be useful, for instance in analysing vehicle trajectories, since the road network simply forces large groups of cars to distribute themselves along the roads therefore creating a cluster with snake-like shape, instead of freely agglomerate around a centre which would instead yield a compact, spherical-shape cluster.

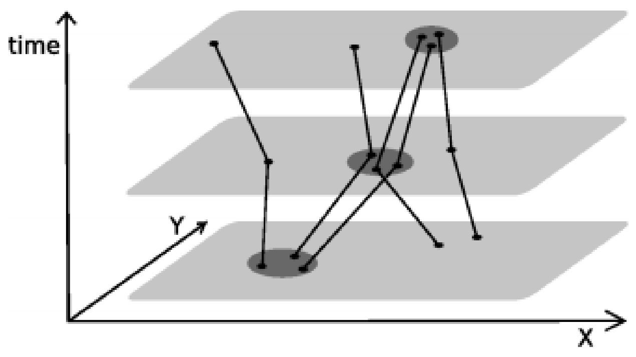


Figure 3 is a visual example of a moving cluster over three time units

Another perk that characterizes moving cluster is the fact that at the population of objects involved in the pattern can change gradually over time: the only constrained requirement is that at each timestamp a cluster exists, and that when moving from different timestamp the population shared by the corresponding spatial clusters is larger than a given fraction. A visual example is shown in Figure 3: at each *time slice* a cluster is found, formed by three objects, and any pair of consecutive clusters share two over three objects. One element that affects both patterns illustrated is the fact that they describe continuous portions of time. For example, if a group that usually moves compactly gets dispersed for a short time and later gets compact again, both flocks and moving clusters will generally result into two different and disconnected patterns, this introduces the *before* and the *after* temporary dispersion

Using relative time

In some frameworks, the moving objects we are examining might act in an akin way, even if they are not together. For instance, similar daily routines might lead several individuals to drive their car along the same routes, even if they have home at very different hours of the day. Or tourists that visit a city in different days of the year might actually visit in the same way – for instance by visiting the same places in the same order and spending approximately the same amount of time on them – because they share interests and attitude.

To evaluate this category of examples we have to introduce a new concept to discover or answer implicit questions like: *Are there groups of objects that perform a sequence of movements, with similar timings though possibly during completely different moments?*

Patterns like flocks and moving clusters can provide some answers to the question. This question leads to the introduction of T-Patterns (Trajectory Patterns) which are defined as a sequence of spatial locations with typical transition times. T-patterns are parametric on 3 factors:

- The set of spatial regions to be used to form patterns e.g., spatial extension of a Railway Station
- The minimum support threshold, corresponding to the minimum size of the population that forms the pattern
- Time tolerance τ, that determines the way transition times are aggregated (*transition times that differ less than τ will be considered compatible, and therefore can be joined to form a common typical transition time.*)

T-patterns do not specify any route among two consecutive regions described instead a travel time is specified, which approximates the (similar) travel time of each individual trajectory represented by a pattern.

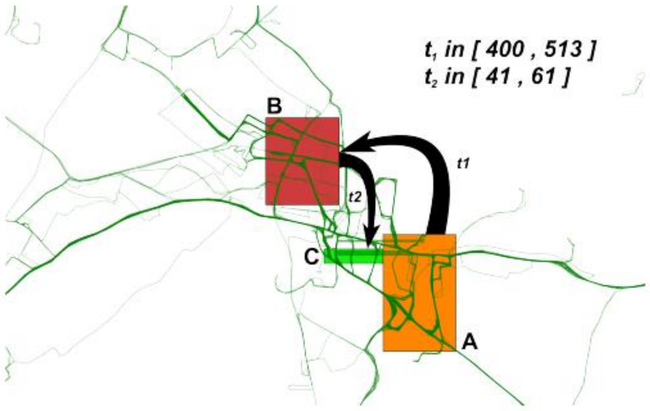


Figure above is an example of T-pattern on vehicle data, which

describes the movement of a fleet of trucks. It shows that exists a consistent flow of vehicles from region A to region B, and then back to C, which is close to the origin. It also indicates that the time from A to B (t1 in the figure) is around ten times bigger than C to the origin.

Not using time

Another real-life situation would be to see if there any typical routes followed by significant portions of the population: *Are there groups of objects that perform a common route (or segment of route), regardless of when and how fast they move?* To simplify the question, we can just say we are interested in the path an individual follows without any time dimension or means of transportation.

To answer the question above we introduce some definitions of patterns provided by mobility data mining known as spatio-temporal sequential pattern. The process consists in two steps[3]:

- Segments of trajectories are grouped based on their distance and direction, in such a way that each group is well described by a single representative segment
- Consecutive segments are joined to form the pattern

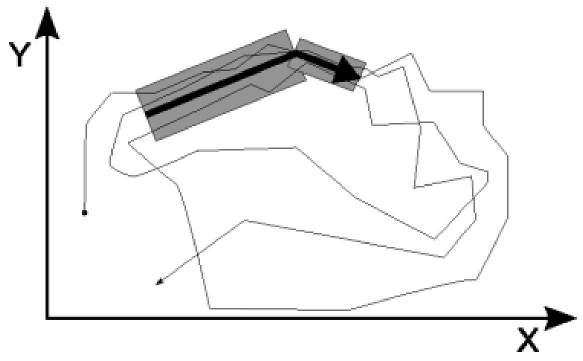


Figure 5 Visual representation of a spatio-temporal sequential pattern

(it is possible to see how the second part of the pattern is tighter than the first one, i.e., the trajectory segments in represents are more compact)

2.3.2 Global trajectory models

A frequent requirement in data analysis is to understand which are the laws and rules that drive the behaviour of the objects that are being monitored. This refers to the (global) trajectory models, and in this field they are three important representative classes of problems:

- Dividing trajectories into homogeneous groups
- Learning rules to label any arbitrary trajectory with some tag, to be chosen among a set of predefined classes

[3] The original proposal of this pattern considers a single, long input trajectory. However, the same concepts can be easily extended to multiple trajectories

- Predicting where an arbitrary trajectory will move next

In the pages below we will introduce and discuss these notions

Trajectory Clustering

Clustering is defined as the process of creating groups of objects that are similar to each other, while keeping separated those that are much different. We can define this operation in one-word *portioning*. However, there are exceptions to this general definition exits and are quite rare.

In the field of data mining its quite common to find clustering methods for simple data, on the other hand applied on trajectories it's not so simple. Trajectories are complex objects, and many traditional clustering methods are tightly bound to simple and standard data type they were developed for. To be able to apply these methods and adaptation has to be undertaken in a trajectory-oriented way.

Generic methods with trajectory distances. Several clustering methods in the data mining literature are actually clustering schemata that can be applied to any data type, provided that a notion of similarity or distance between objects is given. For this reason, they are commonly referred to as *distance-based methods*. The key point is that such methods do not look at the inner structure of data, and simply try to create groups that exhibit small distances between its members. All the knowledge about the structure of the data and their semantics is encapsulated in the distance function provided, which summarizes this knowledge through single numerical values – the distances between pairs of objects; the algorithm itself, then, combines such summaries to form groups by following some specific strategy.

To give an idea of the range of alternative clustering schemata available in literature, we mentioned three very common ones: k-means, hierarchical clustering, density-based clustering.

K-means tries to partition all input objects into k clusters, where k is a parameter given by the user. The method starts from a random partitioning and then performs several iterations to progressively refine it. During an iteration, k-means first computes a centroid for each cluster, i.e., a representative object that lies in the perfect centre of the cluster5), then re-assigns each object to the centroid that is closest to it. Such iterative process stops when convergence (perfect or approximate) is reached.

Hierarchical clustering methods try to organize objects in a multilevel structure of clusters and sub-clusters. The idea is that under tight proximity requirements, several small and specific clusters might be obtained, while loosening the requirements some clusters might be merged together into larger and more general ones. For instance, agglomerative methods start from a set of extremely small clusters – one singleton for each input object – and iteratively selects and merge together the pair of clusters that are most similar. At each iteration, then, the number of clusters decreases of one unit, and the process ends when only one huge cluster is obtained, containing all objects. The final output will be a data structure called dendogram, represented as a tree where each singleton cluster is a leaf, and each cluster is a node having as children the two sub-clusters that originated it through merging.

Finally, *density-based clustering*, is aimed to form maximal, crowded (i.e., dense) groups of objects, thus not limiting the cluster extension or its shape and, in some cases, putting together also couples of very dissimilar objects. Depending on the type of analysis to perform an appropriate similarity function has to be chosen. Examples include the following:

- Spatial route: the spatial shape of the trajectory is considered, and two trajectories that follow a similar path from start to end, will result in a low distance.
- Spatio-temporal route: time is considered; thus two trajectories will be similar when they approximately move together.
- Spatial starts, ends, or both: two trajectories are compared based only on their starting or ending points or in some cases the combination of them.

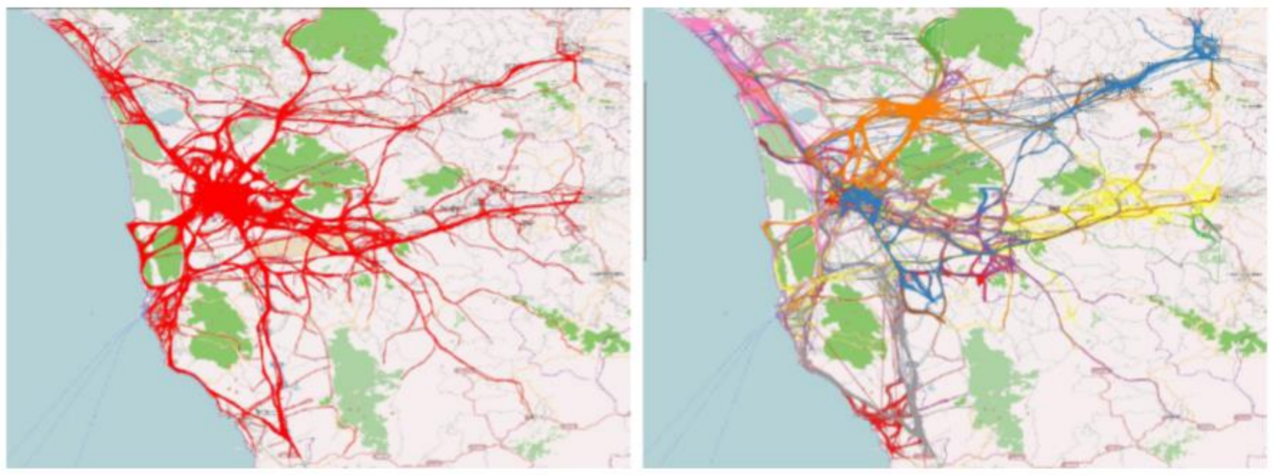


Figure 6 shows a Sample trajectory clustering on a real dataset, obtained using a density-based clustering schema, and a spatial route distance function.

A corresponding approach to clustering, consists in algorithms that try to better exploit the nature and inner structure of a trajectory data. From a technical point of view, that usually translates to deeply re-adapt some existing solution in order to accommodate the characteristics of trajectory data. One important family of solutions makes use of standard probabilistic modelling tools. A very early example was provided by mixture models-based clustering of trajectories (Gaffney and Smyth 1999). The basic idea is not dissimilar from k-means: we assume that the data actually forms a set of k groups, and each group can be summarized by means of a representative object. The difference is that now the representative is a probability distribution of trajectories that fits well with the trajectories in its cluster. The key point in this approach, obviously, is exactly how a probability distribution of trajectories can be defined (and fitted on a dataset). In short, the solution adopted computes a central representative trajectory plus a random Gaussian noise around it, then, the closeness of a trajectory from the cluster centre is simply computed as its likelihood, i.e., the probability that it was generated from the central trajectory adding some Gaussian noise. Another well-known statistical tool often adopted when dealing with trajectories are Hidden Markov Models (HMMs). The basic approach, here, consists in modelling a trajectory as a sequence of transitions between spatial areas. Then, a cluster of trajectories is modelled by means of a Markov model i.e. the set of transition probabilities between all possible pairs of regions, that better fits the trajectories. Moreover, the precise position that a trajectory is expected to occupy within each spatial region is also modelled through a probability distribution. The clustering problem, then, consists in finding a set of HMMs (the clusters), such that each of them fits well with a subset of the trajectories (Mlıch and Chmelar 2008). Other examples of trajectory-oriented clustering methods can arise by adding novel dimensions to the clustering problem. In the literature it was investigated the problem of finding clusters by means of a distance-based clustering method (more exactly, a density-based one, though a similar process might be easily replicated for other approaches) when it is not known in advance the time interval to consider for clustering. For instance, we might expect that rush hours in urban traffic data exhibit cluster structures that are better defined than what happens in random periods of the day. The problem, then, becomes to find both the optimal time interval (rush hours were just a guess to be confirmed) and the corresponding optimal cluster structure. The solution proposed, named time-focused trajectory clustering, adopts a trajectory distance computed as the average spatial distance between the trajectories within a given time interval, which is a parameter of the distance. Then, for each time interval T, the algorithm can be run focusing on the trajectory segments laying within T. The quality of the resulting clusters is evaluated in terms of their density, and a heuristic is provided to explore only a reasonable subset of the possible values of T (Figure 7) (Nanni and Pedreschi 2006).

Trajectory classification

Trajectory classification is defined as the process of predicting the class labels of moving objects based on their trajectories and other features(knowledge). In several contexts such knowledge is available, more exactly in the form of a set of predefined classes and a set of objects that are already labelled with the class they belong to – the so called *training set.* The problem, here, becomes to find rules that classify new objects in way that is rational with prior knowledge.

The simplest solution is called *K-nearest* neighbours approach which directly compares each new trajectory t against the training set, and finds the k labelled trajectories that are close t. The idea rolls the fact that the more similar are two trajectories, the more likely they belong to the same class.

Approaching the problem from a different viewpoint, each class involved in the classification problem could be modelled through a probabilistic model that is fitted to the available trajectories in the class. Then, each new trajectory can be assigned to the class whose model most likely generated it. Similarly, to what we have seen with clustering, hidden Markov models are a common choice to do it. As compared to clustering, the problem is now simplified, since the association trajectories ↔ classes are known apriori. Behind the probabilistic framework they operate in, HMMs essentially aggregate trajectories based on their overall shape, again assuming that similar trajectories have better chances of belonging to the same class.

Another way to classify trajectories is based on a traditional two-steps approach: first extract a set of discriminative features by a preliminary analysis of the trajectories; then, use such features – that can be expressed as a database tuple or a vector – to train any existent standard classification model for vector or relational data. The first step requires to understand which characteristics of the trajectories appear to better predict which class each trajectory belongs to. One straightforward approach might consist in calculating a predefined set of measures expected to be informative enough for the task. For instance, aggregates such as average speed of the trajectory, its length, duration, average acceleration and diameter of the covered region might be used. Other more sophisticated solutions might instead try to extract finer aspects of the movement, tuned to calculate only the most useful ones. A proposal of this kind can be found in literature with the name TraClass, a method which heavily relies on a trajectory clustering step (Lee et al. 2008). Once a vector of features has been computed for each trajectory, we can choose any generic, vector-based classification algorithm. One representative example are decision trees: the resulting classification model has the structure of a tree, whose internal nodes represent tests on the features of the object to classify, and the leaves indicate the class to associate to the objects.

Trajectory location prediction

Predicting a categorical variable related to a trajectory can be seen as the main problem of Trajectory classification. However, prediction is related to the temporal evolution of variables. They are modelling tools that are able to model the sequential evolution of objects they describe. In literature we have the *WhereNext* approach that basically extract T-patterns from a training dataset of trajectories and combine them into a tree structure similar to a prefix-tree. In particular, each root-to-node path corresponds to a T-pattern, and root-to-leaf paths correspond to maximal patterns. Figure 7 shows a sample prediction tree, condensing 12 patterns, 7 of which are maximal (one per leaf).

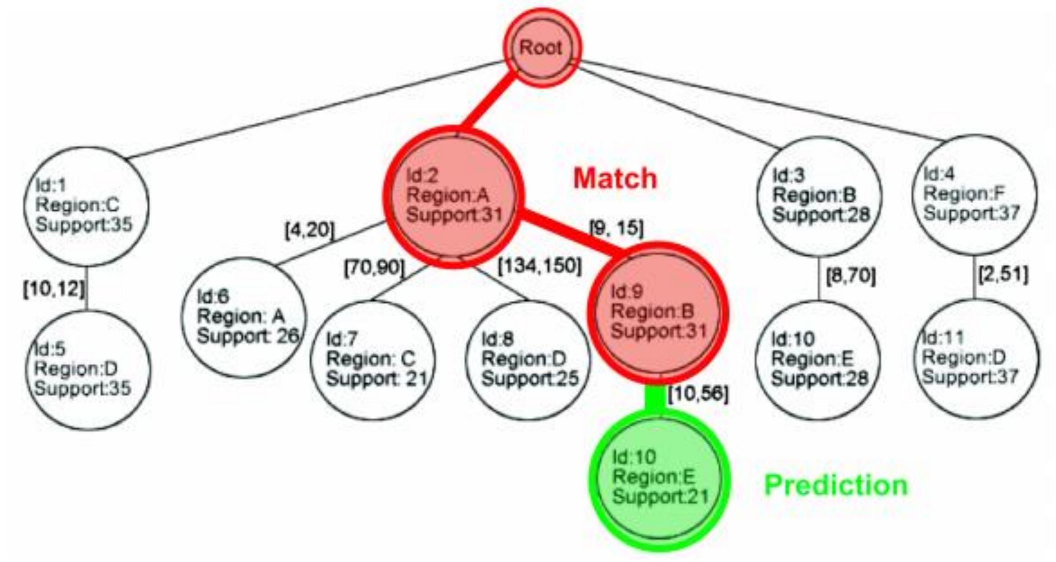


Figure 7 Sample Prediction tree produced by WhereNext.

WhereNext has a few peculiar features that distinguish it from most alternative approaches:

- The next location prediction is equipped also with a temporal delay
- If there is no good match between trajectories and prediction tree, no prediction is provided
- The location prediction occurs in terms of regions and no single spatial points

Trajectory Outliers

Clustering is the process in trying to fit each object in data into some category, although the data analyst is exactly interested in those objects that deviate from the rest of the dataset known as outliers. The search for an outlier means the detection of some feature or pattern.

A method for discovering trajectory outliers consists in adopting a density-based clustering perspective, and therefore computing the number of neighbours of each trajectory over a reasonably large neighbourhood (Those that have few neighbours are considered as outliers).

Privacy

In today's data collection of data lies a little path from opportunities to threats: We are aware that, on the basis of this scenario, there lies a flaw of potentially dramatic consequences, the fact that the donors of mobility data are the citizens, and making these data publicly available for the mentioned purposes would put at risk our own privacy. In other words, the personal mobility data, gathered by wireless networks, are extremely sensitive information; their disclosure may represent a brutal violation of the privacy protection rights, established in increasingly more laws and regulations internationally.

A genuine positivist researcher, with an unlimited trust in science and progress, may observe that, for the mobility-related analytical purposes, knowing the exact identity of individuals is not needed: anonymous data are enough to reconstruct aggregate movement behaviour, pertaining to whole groups of people, not to individual persons. This line of reasoning is also coherent with existing data protection regulations, such as that of the European Union, which states that personal data, once made anonymous, are not subject any longer to the restrictions of the privacy law. Unfortunately, this is not so easy: the problem is that anonymity means making reasonably impossible there-identification, i.e. the link age between the personal data of an individual and the identity of the individual itself. Therefore, transforming the data in such a way to guarantee anonymity is hard: as some realistic examples show, supposedly anonymous data sets can leave unexpected doors open too malicious re-identification attacks.

2.4 Visual Analytics of Movement (Methods, tools, and procedures)

The main idea of visual analytics is to develop knowledge, methods, technologies and practice that take advantage and combine the strengths of human and electronic data processing. We define visualization as the means through which humans and computer cooperate using their distinct capabilities for the most effective results. It is essential for analysing phenomena and processes unfolding in geographical space.

Today the analysis of movement is currently a hot topic in visual analytics, it's a combination of cartography established techniques for representing movements, armies, explores, time geography with techniques for user-display interaction.

The aim of visual analytics is to make our way of processing data and information clear and transparent for analytical purposes. The means of visualization provides ways to communicate information rather than looking at the results only.

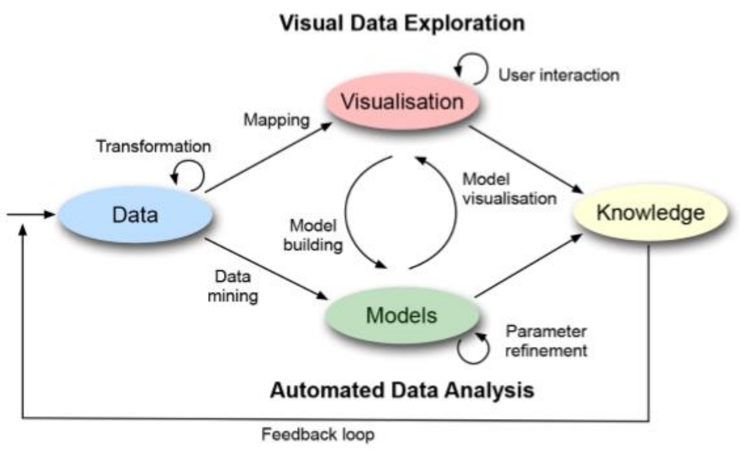


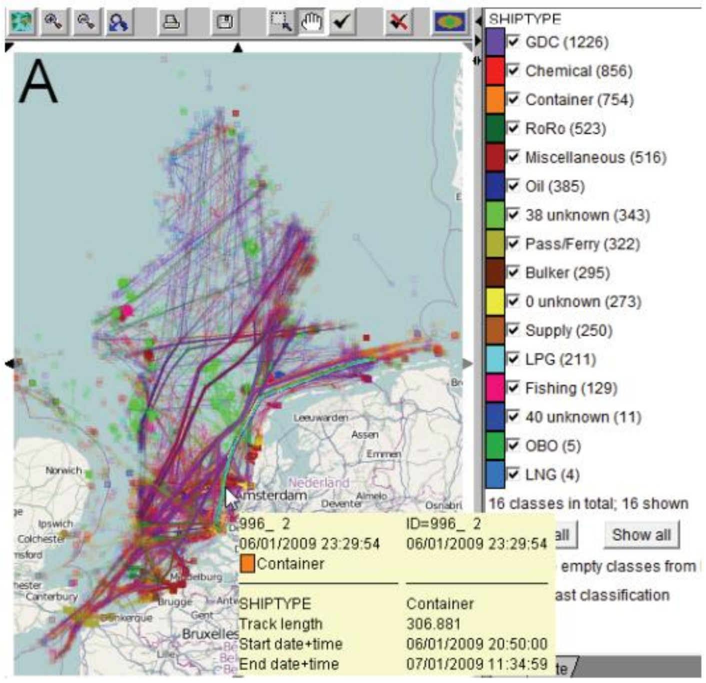
Figure 8 shows the narrow integration of visual and automatic data analysis methods with database technology.

2.4.1 Visualizing trajectories

The most common type of display are static and animated maps and interactive space-time cubes with linear symbols representing trajectories.

Common interaction techniques facilitating visual exploration of trajectories and related data include manipulations of the view (zooming, shifting, rotation, changing the visibility and rendering order of different layers etc.), manipulation of data representation, manipulations of the content, and interactions with display elements, for example accessing different views. These are generic techniques used for various types of data, not only for movement data. They have become standard in information visualization and visual analytics; most of the existing software tools include them. In addition to these generic techniques, there are some interaction techniques specifically designed for trajectories, including selection of trajectories with particular shapes by sketching.

In the Figure below, there is a map with entire trajectories of ships represented by lines with a 10% opacity.



996_ 2	ID=996_ 2
06/01/2009 23:29:54	06/01/2009 23:29:54
🟧 Container	
SHIPTYPE	Container
Track length	306.881
Start date+time	06/01/2009 20:50:00
End date+time	07/01/2009 11:34:59

Figure 9a: An interactive map with trajectories of ships shown as lines with 10% opacity

The small vacant and filled squares indicate respectively the start and end positions of the trajectories. By executing the mouse-pointing on the map, the user accesses a much detailed information about the trajectories. Lines are coloured according to the ship type. The legend on the right servers as a filter to hide/show ship types from the view and focus on the others.

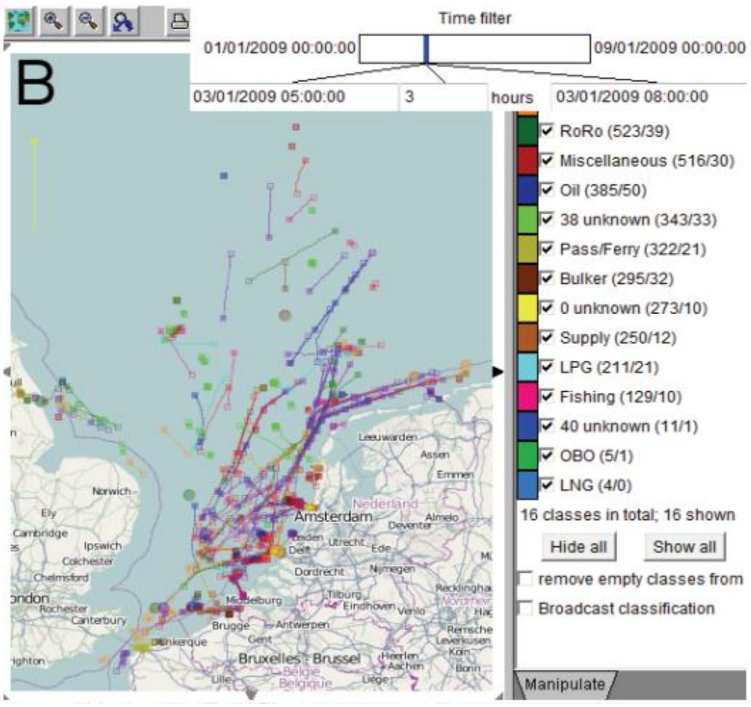
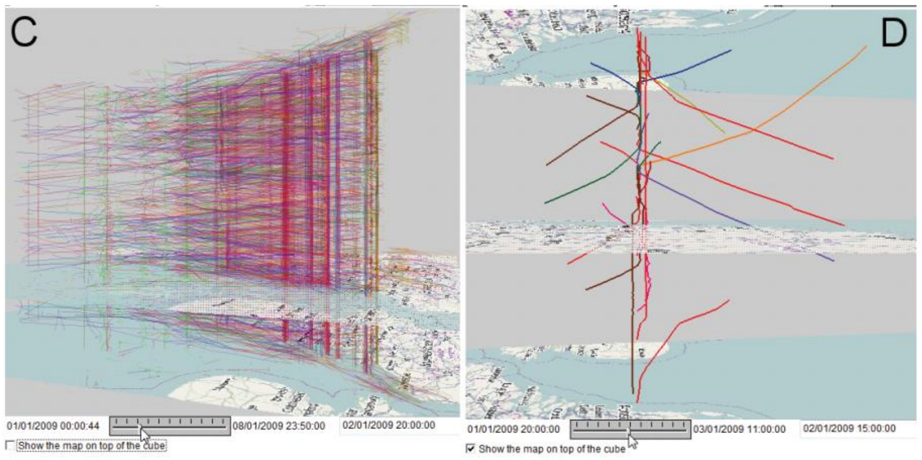


Figure 9b: Map animation through a time filter. Positions and movements of ships during a 3-hours' time interval are visible.

In Fig. 9b, map animations trough time filter is demonstrated. The user can select a time window of a desired length and move this window forwards of backwards in time by dragging the slider. The map in response to these actions show only a fragment of the trajectories that were made during the current time window



In figure C & D, there is an interactive space-time cube (STC) with all ship trajectories, below the map there is map plane that can be moved with the cube to select time moments. In the figure, the position of the movable plane corresponds to the time 20:00 on 02/01/2009 while the whole time range of the data is from 01/01/2009; 00:00 till 08/01/2009; 23:50. Meanwhile in the second figure STC represents selected trajectories.

All the examples seen above may suffer from visual clutter and occlusions. This untidiness can be reduced by reducing the opacity of the symbols.

Clustering trajectories

In visual analytics, clustering is a popular technique used in handling large quantity of data. The habitually existence of clustering methods are contained in interactive visual interfaces, supporting not only visual inspection but often interactive clustering results.

Spatio-temporal data introduce new dimensions and novel issues in performing a clustering task. In movement data, for a give object we have a triple *(x_i,y_i,t_i)* for each time instant t_i . These set of triples for a given object may be viewed as function fid. *Time -> space*, which assigns a spatial location to the object for each time moment.

The extensions of existing clustering methods to the spatio-temporal domain can be classified into two approaches: feature-based models and direct geometric models (Kriegel et al. 2003). In the first case, the objects of the dataset are projected to a multidimensional vector space and their similarities are computed in the new projection. In the second model, the spatial and temporal characteristics of the objects are directly used for defining the distances. We follow the second approach by developing a library of distance functions that compare the trajectories according too various spatial and spatio-temporal properties an analyst may be interested to consider. The properties include the origins of the trips, the destinations, the start and end times, the routes (where a route is characterized in terms of the geometric shape of the footprint and its spatial position and orientation), and the dynamics of the movement (which may be defined in terms of the relative times when different positions have been reached and/or the variation of the speed throughout the trip).

To visualize a trajectory on a map display, we draw a polygonal line connecting its successive positions. The starting position is marked by a small hollow square and the ending position by a larger filled square (Figure 10a). Trajectories can also be visualized in a space-time cube where the vertical dimension represents time. However, when multiple trajectories are visualized in this way, both the map and the cube get severely cluttered and therefore illegible. A possible approach to solving this problem is based on clustering. The idea is to group trajectories by similarity using clustering techniques (Figure 10b) and then represent each group in a summarized way without drawing the individual trajectories (Figure 10c).

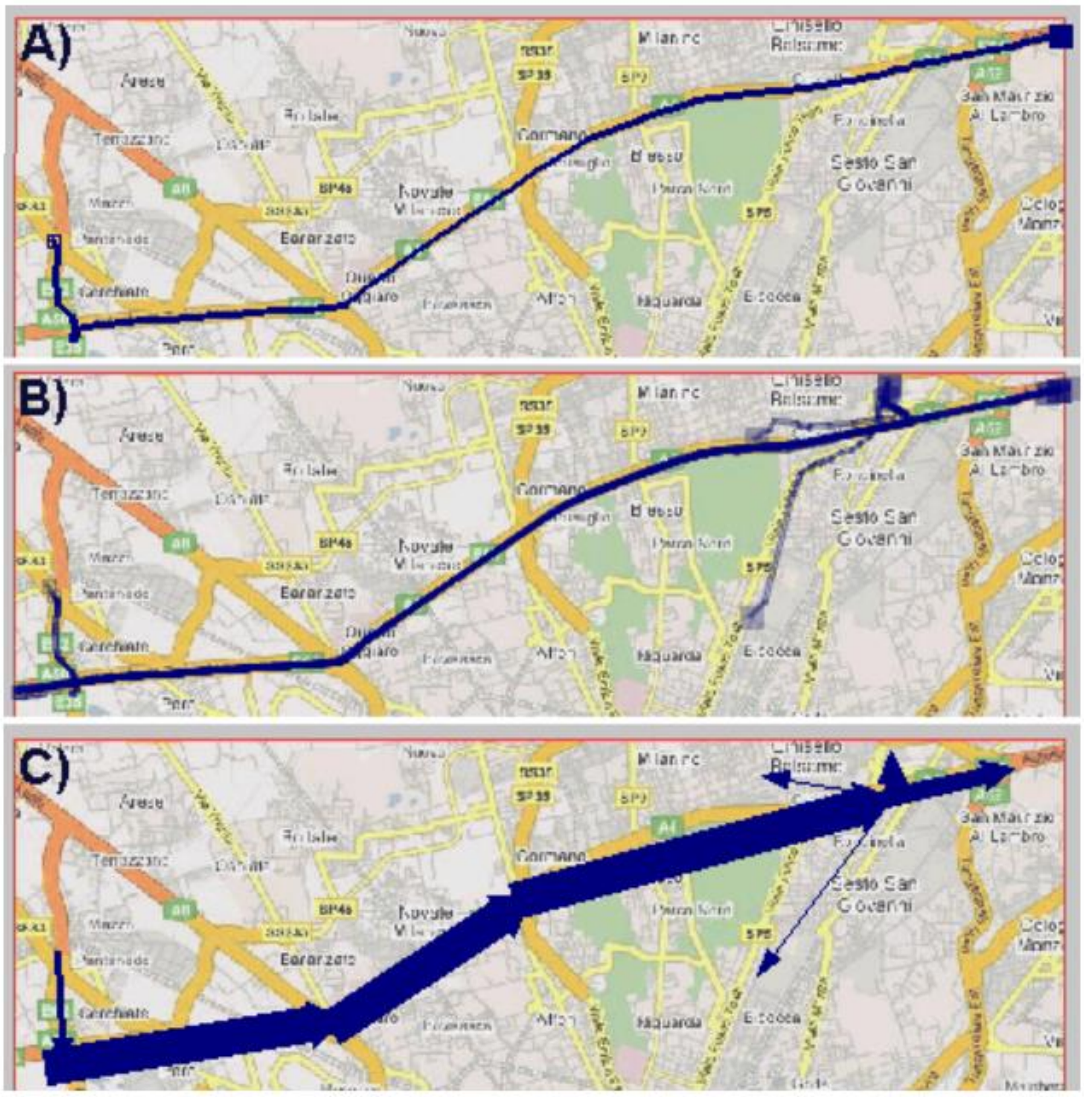


Figure 10. A) Visualization of a single trajectory on a map. B) A cluster of trajectories with similar routes; each trajectory is drawn with 30% opacity. C) A summarized representation of the cluster.

Transforming times in trajectories

Comparison of dynamic properties of trajectories using STC, time graph, or other temporal display is very difficult when the trajectories are far from each other because their representations are also distant in the display view. This problem can be resolved or alleviated by transforming times in trajectories.

Two classes of transformations are suggested:

- Transformations that reflect the cycle nature of time, which means trajectories can be projected in time to a single year / season / month/ week / day and so on. This allows to study movement patterns related to temporal cycles
- Transformations with respect to the single trajectories. Trajectories can be shifted in time to a common start on end time. This simplifies the comparison of dynamic properties of the trajectories

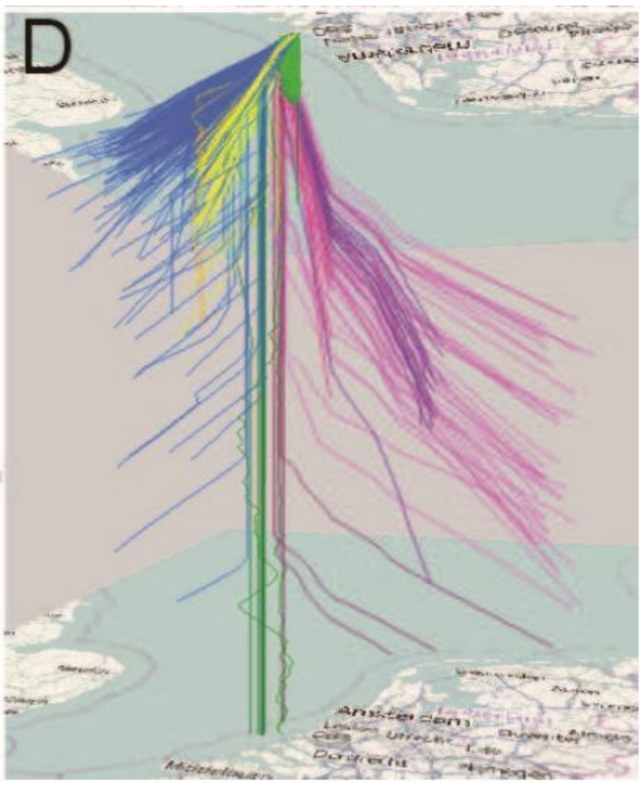


Figure 10d. Clusters by route similarity are shown in a STC; the noise is excluded

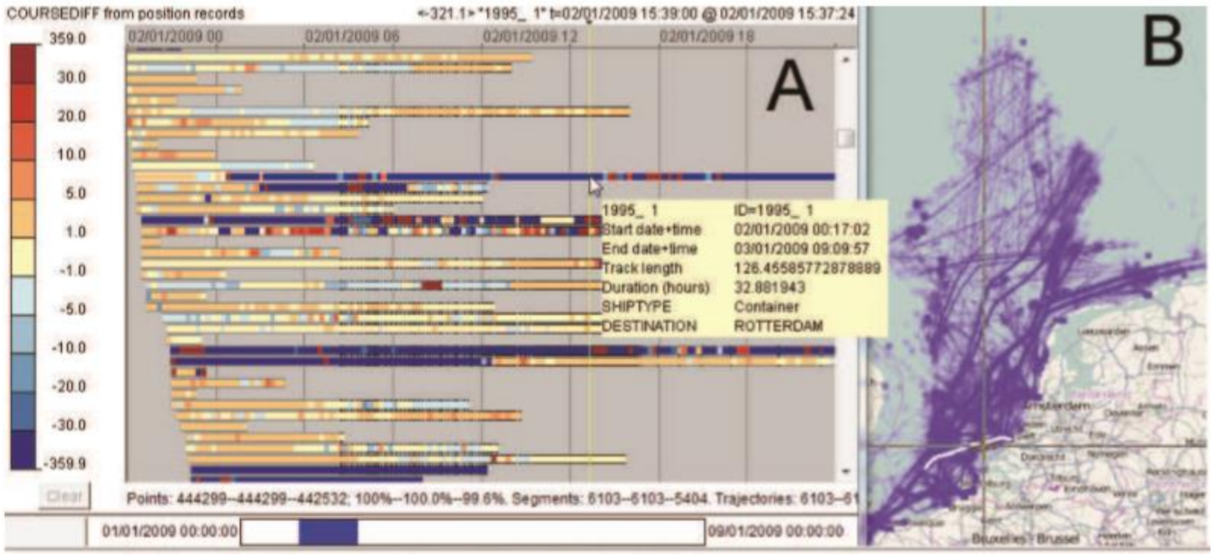
An example of time-transformed trajectories is show in figure above. The STC shows route-based clusters of ship trajectories ending at Rotterdam. The times in the trajectories have been transformed so that all trajectories have a common end time. The STC in Fig. 2 shows us that some ships just stayed all the time near Rotterdam, others approached

Rotterdam and then stayed close to it for time intervals of different lengths, and the remaining ships moved towards Rotterdam with different speeds without stops. Doing such a comparison is hardly possible when the trajectories are positioned in the STC according to their original times.

2.4.2 Looking inside trajectories

The methods described previously deal with trajectories as whole, treating them like atomic objects. In this section we consider methods operating on the level of points and segments of trajectories. These methods visualize and analyse the variation of movement and other dynamic features associated with trajectory position or segments.

The most obvious way to visualize position-related attributes is by dividing the lines or band representing trajectories on a map or in a 3D display. The attributes can be represented by colouring or shading of the segments or by symbols (graphs). It should also be noted that representing dynamic attributes of trajectories on a map or in a STC may be ineffective due to display clutter and occlusions, especially when trajectories on their parts overlap in space.



1995_ 1	ID=1995_ 1
Start date+time	02/01/2009 00:17:02
End date+time	03/01/2009 09:09:57
Track length	126.45585772878889
Duration (hours)	32.881943
SHIPTYPE	Container
DESTINATION	ROTTERDAM

Points: 444299--444299--442532; 100%--100.0%--99.6%. Segments: 6103--6103--5404. Trajectories: 6103--61

A trajectory wall represents trajectories by segmented bands stacked on top of a cartographic map

Figure A demonstrates a time bars display, where the horizontal dimension represents time, each trajectory is represented by a horizontal bar such that the length of bar corresponds to the start time and duration of the trajectory. Meanwhile the vertical dimension is used to arrange the bars. Colouring the bars encodes values of some user-selected dynamic attribute associated with the positions in the trajectories (this is a user selected attribute). The display in **Fig. 11A** represents the values of the attribute "course difference" expressing the difference between the ship heading and its actual movement direction in each trajectory point. The shades of blue and red represent negative and positive differences, respectively. Darker shades correspond to higher absolute values of the differences. Light yellow is used for values around zero. The legend on the left of the display explains the colour coding.

A disadvantage of temporal displays of trajectory attributes, such as time graph and time bars, is that they lack spatial information. To alleviate this, temporal displays are linked to spatial displays through interactive techniques. An example is shown in Fig. 11A and B. The mouse cursor in Fig. 11A is positioned on one of the bars. In Fig. 3B, the trajectory represented by the bar is highlighted on a map display. The intersection of the horizontal and vertical lines marks the spatial location corresponding to the position of the mouse cursor in the time bars display. This means that the ship was in this location at the time selected by the mouse cursor.

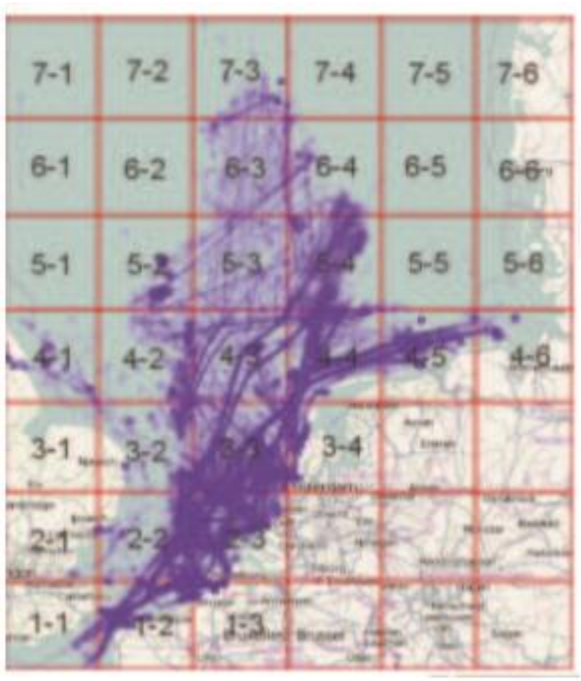
2.4.3 Bird's-eye view on movement: generalization and aggregation

Generalization and aggregation enable an overall view of the spatial and temporal distribution of multiple movements, which is hard to gain from displays showing individual trajectories. Aggregation is helpful in dealing with large amounts of data. There are two major groups of analysis task supported by aggregation

- *Investigating the presence of moving objects* in different locations in space and the temporal variation of the presence.
- *Investigation of the flows of moving objects* between spatial locations and the temporal variation of the flows.

Analysing presence and density

A moving object in a location can be characterized in terms of the count of different objects that visited that location, the count of visits and the total average time spent in the location during some time interval. A part from stats, movements, or activities can be interesting. To obtain these measures, movement data has to be aggregated spatially into continuous density surfaces fields or discrete grids.



The figure above shows density fields built using kernels with different radii can be combined into one field to expose simultaneously large-scale patterns and fine features. Density fields are visualized on a map using colour coding and/or shading by means of an illumination model. A more recent paper suggests a library of methods and a scripting-based architecture for creation, transformation, combination, and enhancement of movement density fields. This approach allows an expert user to involve domain knowledge in the process of building density fields. Mountain further processes density surfaces generated from movement data to extract their topological features: peaks, pits, ridges, saddles, etc.

Presence and density maps do not convey information about the movement directions. Brillinger et al. (Brillinger et al. 2004) use arrow symbols to represent the prevailing movement directions in cells of a discrete grid. The lengths and widths of the symbols are proportional to the average speed and the number of moving objects, respectively. This approach is suitable when the movement is mostly coherent, i.e., objects moving closely in space tend to have

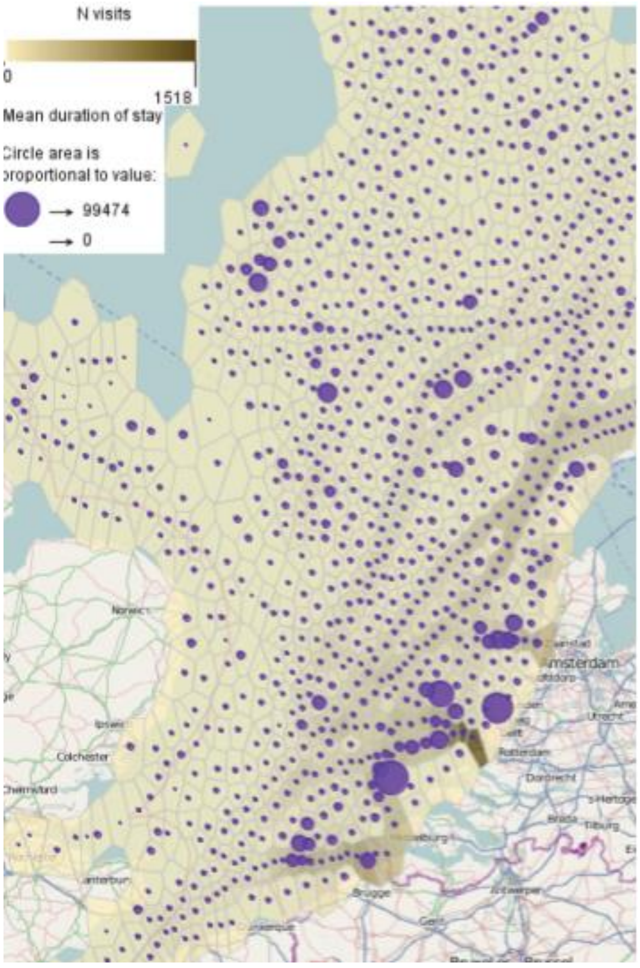


Figure: Spatial aggregation by cells of a discrete grid: counts of cell visits are shown by shading and mean durations of the visits by areas of circles. This figure is taken from (Andrienko and Andrienki 2012).

The same direction, which is not always the case. Movements in different directions can be represented by directional diagrams positioned on a map within grid cells. A diagram is made of bars aligned in a radial or circular layout and oriented in different compass directions, as in a wind rose. The lengths of the bars are proportional to the counts of objects that moved in the respective direction, or to their average speeds, or to any other numeric statistics associated with the directions. This approach combines two kinds of aggregation: by spatial positions and by movement directions. To combine spatial aggregation with aggregation by several arbitrary attributes, spatially-ordered tree maps can be used.

To investigate the temporal changes of object presence and related attributes across the space, spatial aggregation is combined with temporal aggregation, which can also be continuous or discrete.

For discrete temporal aggregation, time is divided into intervals. Depending on the application and analysis goals, the analyst may consider time as a line (i.e. linearly ordered set of moments) or as cycle, e.g., daily, weekly, or yearly.

Accordingly, the time intervals for the aggregation are defined on the line or within the chosen cycle. The combination of discrete temporal aggregation with continuous spatial aggregation gives a sequence of density surfaces, one per each time interval, which can be visualized by animated density maps. It is also possible to compute differences between two surfaces and visualize them on a map, to see changes occurring over time (this technique is known as a change map). The combination of discrete temporal aggregation with discrete spatial aggregation produces one or more aggregate attribute values for each combination of space compartment (e.g., grid cell) and time interval. In other words, each space compartment receives one or more time series of aggregate attribute values. Visualization by animated density/presence maps and change maps is possible as in the case of continuous surfaces.

It is also possible to combine presence/density maps with time series displays such as a time graph or temporal histogram. Zhao et al. (Zhao, Forer, and Harvey 2008) build circular temporal histograms to explore the dependency of movement behaviours on temporal cycles. They also suggest a visualization technique called ring map, a variant of a circular histogram where aggregate values are shown by colouring and shading of ring segments rather than by bar lengths. Multiple concentric rings can represent aggregation according to an additional attribute, for example, activity performed by the moving objects.

When the number of the space compartments is big and time series are long, it may be difficult to explore the spatio-temporal distribution of object presence using only visual and interactive techniques. It is reasonable to cluster the compartments by similarity of the respective time series and analyse the temporal variation cluster-wise, i.e., investigate the attribute dynamics within the clusters and do comparisons between clusters.

Tracing flows

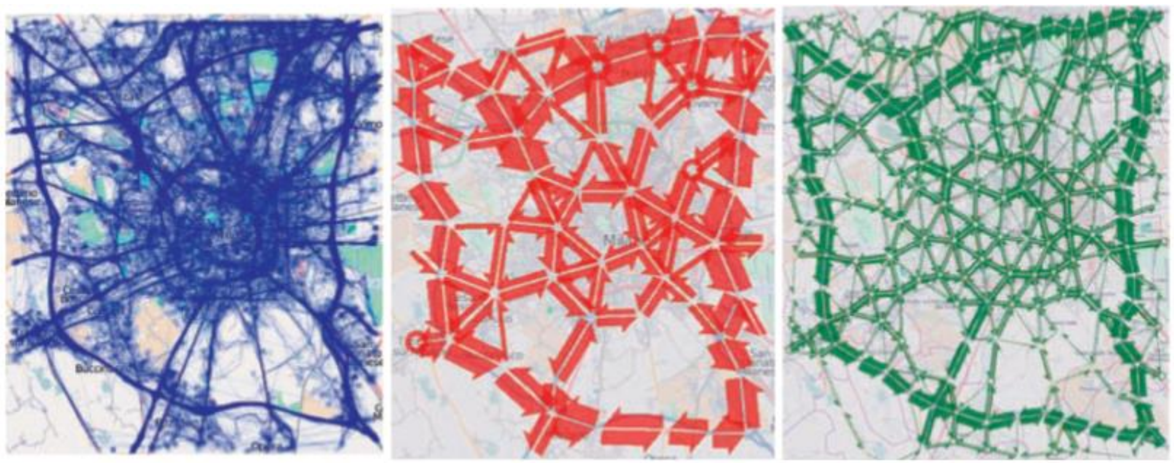
In the previous section, we have considered spatial aggregation of movement data by locations. Now we consider another method of aggregation which is by pairing locations: having two location A & B, we can summarize the moves. This results in a number of transitions, number of different objects that moved from A to B, with the canonical stats. This leads us in introducing the term "flow" used to indicate the aggregated movements between locations. The amount of movement, that is the count of moving objects, may be considered "flow magnitude".

There are two possible ways to aggregate trajectories into flows assuming that each trajectory represents a full trip of a moving object from some origin to some destination, the trajectories can be aggregated by origin-destination pairs. A well-known representation of the resulting aggregates is the *origin-destination matrix* also known as the (*OD matrix*) where the rows and columns correspond to the locations and the cells contain aggregate values. OD matrices are often represented graphically as matrices with shaded or coloured cells. A disadvantage of the matrix display is the lack of spatial context.

An alternative way to visualize flows is the “flow map” where flows are represented by straight or curved lines or arrows connecting locations; the flow magnitude are represented by proportional widths and/or colouring or shading of the symbols.

The other possible way of transforming trajectories to flows is to represent each trajectory as a sequence of transitions between all visited locations along the path and aggregate the transitions from all trajectories. Even movement data positions may be aggregated so that only neighbouring locations are linked by flows. These flows can be represented on a flow map without intersections and occlusions.

Andrienko and Andrienko (Adrienko and Adrienko 2011) described a way to define suitable places for aggregating movement data in the cases when the positions of moving agents are specified only by numeric coordinates lacking any semantics, i.e. no predefined areas are given. The suggested method extracts significant points from the trajectories, groups them by spatial proximity, and the uses the centres of the groups as generating points for a Voronoi tessellation of the territory. The resulting Voronoi cells are used as places for the division of the trajectories into segments. For each ordered pair of places, the trajectory segments starting in the first place and ending in the second place are aggregated. The quality of the generalization is measured and improved if necessary. The degree of the generalization depends on the sizes of the cells which, in turn, depend on the spatial extents of the point groups. The desired spatial extent (radius) is a parameter of the method. The system visualizes the quality measures and provides tools for adjusting the data abstraction level in spatial aggregation of movement data. It is possible to increase or decrease the abstraction level on the whole territory and to do local adjustments in selected parts of the territory.



Flow maps based on fine, medium, and coarse territory divisions obtained automatically. This figure is taken from (Adrienko and Adrienko 2011).

2.5 A quick view on Network-Based Analysis of Human Mobility

In the real world, different events may dramatically change how people move on the territory. Such events may be unpredictable, like natural disasters, but most of them are not. The most natural regular and predictable event is the transition between working and non-working days. For example, during Saturdays and Sundays, people usually abandon their working mobility routines for different paths, obeying to complete different criteria, that can be considered outliers. Another example may be organized human social events like concerts manifestations in particular location.

In this section an extensive description will take place showing the base framework and ideas used to develop the application and all spatial and temporal analysis of Human Mobility.

Community Discovery

A very important piece of the base framework is the application of clustering algorithm on the network of trajectories taken into account. To make this clear in this section the problem of community discovery in complex networks will be introduced with an adopted solution.

An extensive survey, providing more background about community discovery, can be found in [5]. From the 5 we know that clustering algorithms can provide extremely different results, according to their definition of what is a community in complex network.

Clustering algorithms enable the multi-level identification of clusters-in-a-cluster, and they are defined “hierarchical”. With this type of clustering algorithms, we can explore each cluster at several levels and possibly choose the level. This is a critical function for mobility networks, as in this scenario it is necessary to explore borders at different granularity levels: conglomerates of cities, cities and even neighbourhoods.

Among the hierarchical clustering algorithms available in the literature, infomap has been used because it’s one of the best performing non-overlapping clustering algorithms.

The infomap algorithm is based on a combination of information theoretic techniques and random walks on a graph as a path for information flows in the real system and decomposes the network into clusters by compressing a description of the probability flow. The algorithm looks for a cluster partition M in tm clusters so as to minimize the expected description length of a random walk. The intuition behind the Infomap approach for the random walks compression is the following:

The best way to compress the paths is to describe them with a prefix and suffix. Each node that is part of the same cluster M of the previous node is described only with is suffix, otherwise with prefix and suffix. Then the same suffixes are reused in different cities. The optimal division in different prefixes represent the optimal community partition. We can formally present the theory behind Infomap.

The expected description length, given a partition M, is given by:

$$L(M) = qH(Q) + \sum_{i=1}^{m} p_i H(P_i).$$

L(M) is made up of two terms: the first is the entropy of the movements between clusters and the second is the entropy of movements within clusters. The entropy associated to the description of the n states of a random variable X that occur with probabilities

$$p_i \; is \; H(X) = -\sum_{1}^{n} p_i log_2 p_i.$$

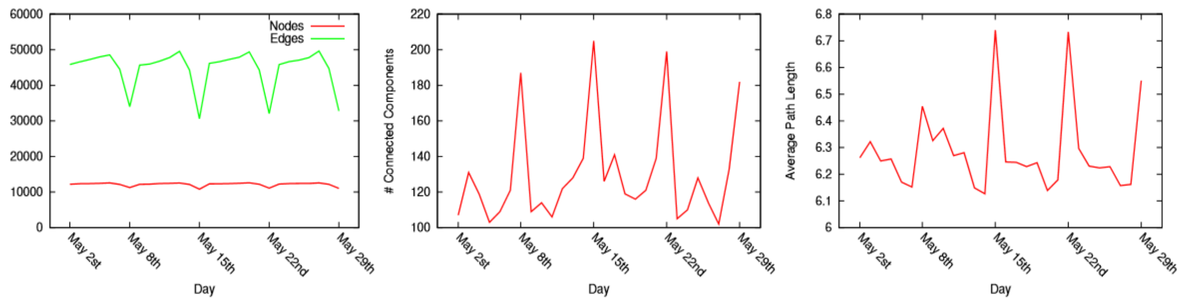
In (1) entropy is weighted by probabilities with which they occur in the particular partitioning. More precisely, q is the probability that the random walk jumps from a cluster to another on any given step and p_i is the fraction of within-community movements that occur in community i plus the probability of exiting module i. Accordingly, *H(Q)* is the entropy of clusters names, or city names in our intuition presented before, and *H(Pi)* the entropy of movements within cluster i, the street name in our example, including the exit from it. Since trying any possible partition in order to minimize *L(M)* is inefficient and intractable, the algorithm uses a deterministic greedy search and then refines the results with a simulated annealing approach.

Temporal Dimension

In this section, we explore the temporal issues of the application of complex network analysis to mobility data. As a proxy of human mobility, we used a dataset of spatio-temporal trajectories of private cars consisting of around 10M trips performed by 150,000 vehicles. The GPS tracks were collected by Octo Telematics S.p.A., a company that manages on-board GPS devices and data collection for the car insurance industry. Each trajectory is represented as a time-ordered sequence of tuples (*id, x, y, t*), where *id* is the anonymized car identifier, x and y are the latitude a longitude coordinates collected during a period of one month, from the 1st of May to the 31st 2011. The GPS device automatically starts collecting the positions when the car is turned on and it stops when its turned off. The log is transmitted to the server via GPRS connection. Octo Telematics serves the 2% of registered vehicles in Italy, in our collection, they collected the traces of the vehicles circulating in a bounding box containing Tuscany Region during the period of observation.

To apply complex network analysis on mobility data we first generalize the spatio-temporal positions by means of spatial tessellation. This is already a challenge per se, and we deal more in deep with it in following pages. Moving forward we focus on the origin and destination of each travel of each vehicle. Using the spatial tessellation provided by ISTAT, the statistical bureau in Italy, we associate each origin (destination) to the census sector where the corresponding travel began (ended).

After this generalization step we can model human mobility by means of a graph where the nodes represent the census sectors and each edge where the nodes represent the census sectors and each edge represent the set of travels starting and ending within the corresponding census sector. In particular, and edge connecting nodes *v1* and *v2* is weighted with the number of travels starting from the sector associated to *v1* and ending at the sector associated with *v2*. However, since we are interested in studying the temporal evolution of the extracted network, we extracted networks at different time intervals. In general, our method consists in selecting only the trajectories "alive" in the time period of study.



(a) Number of nodes/edges. (b) Number of components. (c) Average path length.

Figure: Some statistics for the daily network snapshots.

A temporal interval was fixed for one day, and daily snapshots of the movements graphs. In the figure above we represent some of the basic statistics of these daily networks. We can see that there are remarkable differences between weekday and weekend networks (noting that May 8th, 15th,22nd and 29th 2011 were Sundays). Saturdays and Sundays networks usually have lees edges, somewhere between 62-75% of the edges of a weekday Fig (a); they have more components, i.e. the networks are more fragmented, with areas not connecting at all to each other Figure (b); and finally, their average path length is significantly higher, may 8th presents a lower peak, but the whole preceding week was lower than the following, due to the fact of Italian national holiday of May 1st Figure(c).

This brings us to the conclusion that we can expect different results from the weekdays and weekend networks, as their topology is significantly different. Thus, we considered three distinct intervals for each week:

weekdays, i.e. from Monday to Friday, weekends Saturday and Sunday, and the whole week obtaining 12 networks for the four weeks considered. All 12 networks have gone through the application on the infomap Algorithm that return Hierarchical form which has been outputted into file.tree[4]. This outputted data will be the base of our data input for the application.

[4] File extension for hierarchical data

Chapter III

Web Application

In the light of the topics and data structures addressed in the previous chapters, and due to the amount of data that had to be managed and processed, various choices have been made to build an application that is dynamic and interactive as possible taking into account some computing and efficiency factors. The gap between the data and the final web application has to be filled with various solutions and technologies.

This section illustrates the architecture and all the technologies used to build the application from the front-end with the use of D3.js for producing dynamic interactive data visualization for the web by manipulating documents based on data to the back-end with the aid of Node.js a server-side platform used in this context to process data, reliving the client of the all the unnecessary computation and processing aspects. This is done due to the fact that client-side scripts are parsed and then processed on the client's browser, if scripts are not well write taking care of performance optimization issues they can introduce inefficiencies.

Due to some nda conditions, we could not implement a basic structure for data processing for the fact that raw data cannot be publicly broadcast to the web, this is done to preserve the privacy of the company clients.

3.1 Data Visualization

Data visualization is the presentation of data in a graphical format. Visualization helps people see things that were not obvious to them before. Even when data volumes are very large, patterns can be spotted quickly and easily. Visualization conveys information in a universal manner and makes it simple to share with others. It gives people the opportunity to ask themselves and others “Do you see what I see?” A big advantage to data visualization is its ability to scale with large amount of data; a traditional electronic spreadsheet cannot visually represent the information due to software limitations. In addition, if printed, the spreadsheets would be very long, and thus in both cases taking time and resources to analyse. Data visualization presents data in a way that anyone, from the expert to a novice can easily interpret, saving time, money, and energy. Because of the way our brain processes information, it is faster for people to grasp the meaning of data when they are displayed in charts and graphs rather than poring over piles of spreadsheets or reading pages and pages of reports on data.

3.1.1 Interactive Visualization

Data visualization is the presentation of data in a graphical format. Visualization helps people see things that were not obvious to them before. Even when data volumes are very large, patterns can be spotted quickly and easily. Visualization conveys information in a universal manner and makes it simple to share with others. It gives people the opportunity to ask themselves and others “Do you see what I see?” A big advantage to data visualization is its ability to scale with large amount of data; a traditional electronic spreadsheet cannot visually represent the information due to software limitations. In addition, if printed, the spreadsheets would be very long, and thus in both cases taking time and resources to analyse. Data visualization presents data in a way that anyone, from the expert to a novice can easily interpret, saving time, money, and energy. Because of the way our brain processes information, it is faster for people to grasp the meaning of data when they are displayed in charts and graphs rather than poring over piles of spreadsheets or reading pages and pages of reports on data.

Dynamic, interactive visualization can empower people to explore data for them-selves. All design patterns found today in most interactive visualization have changed since when Ben Shneiderman[5] proposes a "Visual Information-Seeking Mantra": Overview first, zoom and filter; then details-on-demand. This combination of functions is successful because it makes the data accessible to different audiences from those who are merely browsing or exploring the data, to those who approach the visualization with a specific question in search of an answer.

An interactive visualization that offers an overview of the data alongside tool for "drilling down" into the details may successfully fulfil many roles at once, addressing different concerns of the different audiences, from those new to the subject matter to those already deeply familiar with the data.

Interactivity can also encourage engagement with the data in ways that static images cannot. With animated transitions and well-crafted interfaces, some visualizations can make exploring data feel more like playing a game. Interactive visualization can be a great medium for engaging an audience who might not otherwise care about the topic or data at hand.

Visualizations are not truly visual unless they are seen. Getting the application out there for others to see is very critical, and the quickest way to do so is publishing on the web. Working with the web standard technologies means that we can reach a broad audience from the novice to the experienced, regardless of the operating system.

[5] https://en.wikipedia.org/wiki/Ben_Shneiderman

3.2 Technologies

Just like all application in the world this application is built using HTML, CSS and JavaScript. These components work as follow:

- **HTML** (Hyper Text Markup Language) is a markup language for describing web documents and therefore it defines the structure of static web page content.
- **CSS** (Cascading Style Sheets) is a stylesheet language that describes the presentation of an HTML (or XML) document, thus CSS describes how elements must be rendered.
- **JavaScript** (often shortened to JS) is a lightweight dynamic scripting language that runs on the web client, i.e. the browser. It is used to make web pages interactive. Since it runs on the client, it allows to have different and changing content after the page has been loaded depending on the user input, the environmental conditions, or other variables.
- ***SVG*** for the creation of maps and two-dimensional graphics. Scalable Vector Graphics is an XML based vector image format for two-dimensional graphics that supports Interactivity.
- ***JQuery*** for events handling and animations. JQuery is an Open Source JavaScript library that simplifies navigation in a document.

In addition, other frameworks, libraries, and plug-ins are used when necessary. The following subsections describe the main tools used and explain why they were chosen and their role in the application development.

3.2.1 D3.js

D3 also referred to as D3 [6]or d3.js is a JavaScript library for creating data visualizations. But that kind of undersells its.

The abbreviation references the tool's full name, Data-driven Documents. The data is provided by you, and the documents are web-based documents, meaning anything can be rendered by a web browser, such as HTML and SVG. D3 does the driving in the sense that it connects the data to the documents.

A huge benefit of how D3 exposes the designer directly to the web page is that the existing technology in the browser can be leveraged without having to create a whole new plotting language. This appears both when selecting elements, which is performed using CSS selectors (users of **JQuery** will find many of the idioms underlying D3 very familiar), and when styling elements, which is performed using normal CSS. This allows the designer to use the existing tools that have been developed for web design—most notably Firefox's Firebug and Chrome's Developer Tools. Instead of creating a traditional visualization toolkit, which typically places a heavy wrapper between the designer and the web page, D3 is focused on providing helper functions to deal with mundane tasks, such as creating axes and axis ticks, or advanced tasks such as laying out graph visualizations or chord diagrams. This means that, once over D3's initial learning curve, the designer is opened up to a very rich world of modern, interactive and animated data visualization.

Fundamentally, D3 is an elegant piece of software that facilitates generation and manipulation of web documents with data. It does this by:

- *Loading* data into the browser's memory

[6] https://d3js.org/

- *Binding* data to elements within the document, creating new elements as needed

- *Transforming* those elements by interpreting each element's bound datum and settings its visual properties accordingly

- *Transitioning* elements between states in response to user input

Learning to use D3 is simply a process of learning the syntax used to tell how we want to bind and load data, and transform and transition elements.

The transformation is the most important step, as this is where the mapping happens. D3 provides a structure for applying these transformations. Should larger values make taller bars or brighter circle? Will Clusters be sorted on the x-axis by age or category? What colour palette is used to fill in countries on a world map? These visual decisions are up to us.

In a sense, D3 is a specialized JavaScript library that allows you to create amazing data visualizations using a simpler (data driven) approach by leveraging existing web standards. D3.js was created by Mike Bostock (http://bost.ocks.org/mike/) and superseded his previous work on a different JavaScript data visualization library called Protovis. For more information on how D3 was created and on the theory that influenced both Protovis and D3.js, please check out links in the following information box. Here in this book we will focus more on how to use D3.js to power your visualization. Initially, some aspects of D3 may be a bit confusing due to its different approach to data visualization using JavaScript. I hope that over the course of this book, a large number of topics, both basic and advanced, will make you comfortable and effective with D3. Once properly understood, D3 can improve our productivity and expressiveness with data visualizations by orders of magnitude.

Information Visualization Systems: D3 includes a collection of helper modules that sit on top of the selection-based kernel. In the last few years, researchers have developed a variety of toolkits for facilitating visualization design. One class of framework provides a hierarchy of visualization components so that new visualizations are introduced either by authoring new components or sub classing existing ones. A second class of framework explicitly instantiates the InfoVis Reference Model using a set of composable operators for data management, visual encoding, and interaction. Though new combinations of operators can enable customized views, most novel visualizations require programmers to author completely new operators. Thus, both classes of framework work well when visualization creators have software engineering expertise, but are prohibitive to more general audiences such as web designers. Instead, when developing Protovis, Jeff Heer and Michael Bostock have advocated for declarative, domain specific languages for visualization design. By decoupling specification from execution details, declarative systems allow language users to focus on the specifics of their application domain, while freeing language developers to optimize processing. Similar to Protovis, D3 provides a declarative framework for mapping data to visual elements. However, unlike Protovis and other grammar-based declarative models (such as ggplot214), D3 does not strictly impose a toolkit-specific lexicon of graphical marks. Instead, D3 directly maps data attributes to elements in the document object model. Whether or not this design move is advantageous depends on context: many programming environments do not provide a standardized scenegraph abstraction. Moreover, toolkit specific scenegraph abstractions have compelling benefits as custom abstractions can facilitate portability (cross-platform deployment) and performance optimization. A curated lexicon of graphical marks can also improve notational efficiency. Consequently, when developing D3, Bostock and Heer maintain that toolkit-specific representations continue to be an important component of many visualization models, and they address this with D3's helper modules. The technical constraints and entrenched standards of the web have led them to a different approach for browser-based visualization. The browser environment does not provide the same optimization opportunities as compiled programming languages; instead, the overhead of mapping an internal representation to the DOM introduces performance bottlenecks. Intermediate representations can also complicate debugging, as the mapping between code (written in terms of abstract graphical marks) and inspect able output (e.g., SVG elements in the DOM) is often unclear. Custom abstractions may additionally limit expressiveness: they must be revisited to take advantage of new browser features and due to encapsulation may be unable to exploit supporting technologies such as CSS. D3 is designed to sidestep these problems and complement web standards. It also introduces features that may inform other visualization frameworks: query-driven selection and data binding to scene-graph elements, document transformation as an atomic operation, and immediate property evaluation semantics.

How does D3 work? D3's atomic operand is the selection: a filtered set of elements queried from the current document's DOM. Operators act on selections, modifying content. Data joins bind input data to elements, enabling functional operators that depend on data, and producing enter and exit sub-selections for the creation and destruction of elements in correspondence with data. While operators apply instantaneously by default, animated transitions interpolate attributes and styles smoothly over time. Special operators called event handlers respond to user input and enable interaction. Numerous helper modules, such as layouts and scales, simplify common visualization task. Learning the syntax used to tell it how you want it to load and bind data, and transform and transition elements. By using explicit transformations of a native representation, D3 can avoid unnecessary computation (transformations can be limited to selected attributes) and reduce overhead (the DOM is modified directly, eliminating the indirection of an intermediate scenegraph). The design decisions improve performance compared to a higher-level framework such as Protovis[7]. D3's core language provides an efficient foundation for specifying rich, dynamic visualizations on the web.

3.2.2 Node.js

Node.js [8]is a server-side platform built on Google Chrome's JavaScript engine (V8 Engine) for easy building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. It's to be noted that all Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Following are some of the important features that make Node.js the first choice of software architects.

- *Asynchronous* and Event Driven − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism

[7] http://mbostock.github.io/protovis/docs/
[8] https://nodejs.org/en/

of Events of Node.js helps the server to get a response from the previous API call.

- *Very Fast* – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- *Single Threaded but Highly Scalable* – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- *No Buffering* – Node.js applications never buffer any data. These applications simply output the data in chunks.
- *License* – Node.js is released under the MIT license.

The use of Node.js in the back-end architecture was obvious due to the fact that its ideal for JSON APIs based Application and I/O bound Applications

3.2.3 TopoJSON

TopoJSON is an extension of GeoJSON that encodes topology. Rather than representing geometries discretely, geometries in TopoJSON files are stitched together from shared line segments called arcs. This technique is similar to Matt Bloch's MapShaper and the Arc/Info Export format, .e00.

TopoJSON eliminates redundancy, allowing related geometries to be stored efficiently in the same file. For example, the shared boundary between California and Nevada is represented only once, rather than being duplicated for both states. A single TopoJSON file can contain multiple feature collections without duplication, such as states and counties. Or, a TopoJSON file can efficiently represent both polygons (for fill) and boundaries (for stroke) as two feature collections that share the same arc mesh. See How To Infer Topology for a visual explanation of how TopoJSON works. See Command-Line Cartography for an introduction to TopoJSON and related tools.

To further reduce file size, TopoJSON can use quantized delta-encoding for integer coordinates. This is similar to rounding coordinate values (e.g., LilJSON), but with greater efficiency and control over loss of information. And like GeoJSON, TopoJSON files are easily modified in a text editor and amenable to gzip compression.

As a result, TopoJSON is substantially more compact than GeoJSON, frequently offering a reduction of 80% or more even without simplification. Yet encoding topology also has numerous useful applications for maps and visualization above! It allows topology-preserving shape simplification, which ensures that adjacent features remain connected after simplification; this applies even across feature collections, such as simultaneous consistent simplification of state and county boundaries. Topology can also be used for Dorling or hexagonal cartograms, as well as other techniques that need shared boundary information such as automatic map colouring.

3.2.4 HTML 5

HTML5 is a mark-up language used from structuring and presenting content for the World Wide Web and a core technology of the internet. It is the fifth iteration of the HTML standard created in 1990; its core aims have to improve the language with support for the latest multimedia standards, while keeping it easily readable by human and consistently understood by computers and devices.

The definition of the specifications is managed by the WC3 (World Wide Web Consortium) and the WHATWG (Web Hypertext Application Technology Working Group). The innovation introduced in HTML5 compared to the previous iterations are aimed in particular at improving the structure defined as the mark-up, the characteristics of yield, defined by the style guidelines, and content of a web page, defined by the actual text. In addition, the HTML 5 provides support for local storage of large amount of data downloaded from the browser, to enable the use of web-based applications (such as mailboxes of Google or other similar services) even in the absence of Internet connection. In particular, these are the innovations:

- The rules for the structuring of the text into chapters, paragraphs and sections are tightened, control elements for navigation are introduced;
- The controls for the electronic modules are improved and extended;
- Specific elements are introduced for the control of multimedia content (video/ audio);
- Few elements have been removed or deprecated such as items that have no actual use;
- Canvas is supported and lets you use JavaScript to create animations and bitmap graphics;
- Introduction of geolocation24, due to a strong expansion of mobile operating system (Such as Android and iOS);
- Standardization of JavaScript programs called Web Workers, and ability to use some sites offline;

- Changing the doctype* and complex, with a simple <! DOCTYPE html>;

3.2.5 CSS3

CSS3 is a technology in the final stage of development whose aim is to provide tools to generate graphical effects, applying directives to the stylistic elements in a web page. The specifications related to CSS3 defined by the WC3 (World Wide Web Consortium), have three basic characteristics:

- The core language is left intact, and the validation of documents in CSS2 or CSS3 continues to be valid;
- The number of properties increased from 115 to 276
- The WC3 has organized the specification into modules, each of which covers a specific area of CSS. Everyone has a life of its own and a different degree of progress towards his or her definition.
- Finally, CSS3 should present solutions for the correction of bugs generated rendering different browser engines such as internet Explorer.

3.3 Architecture

In the light of the topics addressed in the previous chapters and the amount of data that had to be managed for the final target, various choices have been made to make the application dynamic and interactive as possible. In this section, we explain all the steps for data processing giving a detailed look at the data available (subsection 4.3.1). We need to understand which information the data contains and how it is structured. Accordingly, we need to understand if this data needs to be cleaned/filtered or if it needs some other pre-processing (subsection 4.3.2). Finally, we need to understand what is the most appropriated structure for the data we will load in the application, and eventually change the original data structure and its file format (subsection 4.3.3).

The architecture of the application from here on will be known as the backend, all operation that run in the backend are done using Node.js an open-source, cross-platform JavaScript runtime environment for developing a diverse variety of tools and applications.

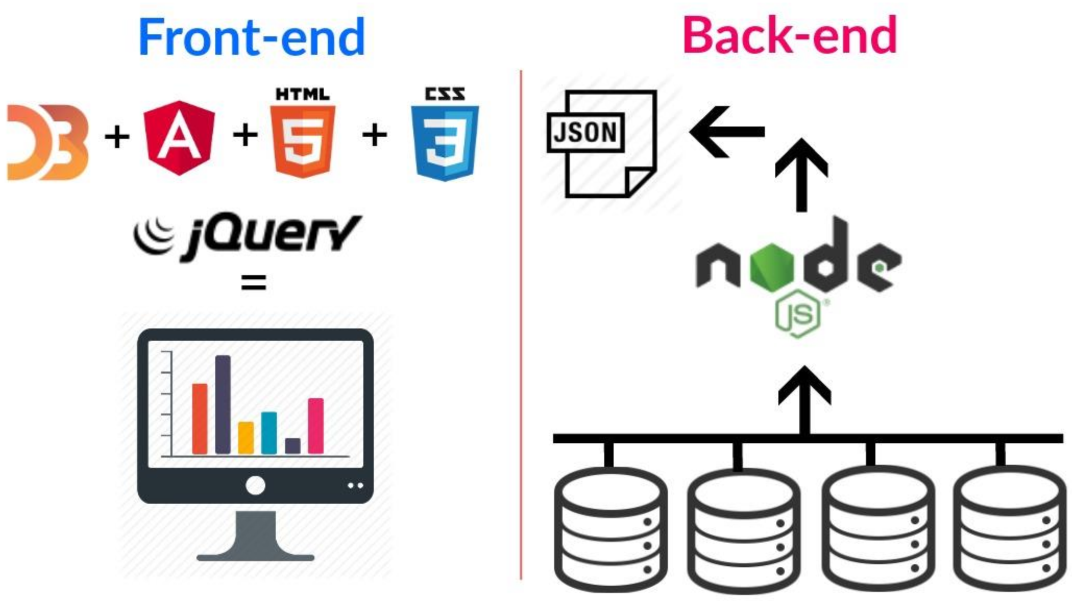
Due to some *nda* [9]conditions, we could not implement a basic structure for data processing for the fact that raw data cannot be published in other to preserve users' privacy.

The structure of the application is divided in 2 sides:

- ***Back-end***: The backend is an application that serves indirectly in support of the front-end, in our case the backend does all the pre-processing activities to our input data.

- ***Front-end***: Also, known as the client-side, the side of the application in which generally HTML, CSS and JavaScript are combined to produce and interface where the user can interact directly. It also needs to be noted that D3.js has been heavily adopted for the front-end.

[9] A non-disclosure agreement



The **figure** above illustrates the various technologies used by the Front-end and the workflow of data in the back-end.

3.3.1 Data input

As discussed in the previous pages, we specified that the base for our input data are 12 file.tree[10] that represent the spatio-temporal trajectories of private cars consisting of around 10M trips performed by 150,000 vehicles. All 12 files cover a time period of a month from the 1st of May to the 31st year 2011. The month as been subsequently divided into 4 weeks, in which a single week was also divided into a temporal dimension giving us the following scheme:

[10] A FileTree represents a hierarchy of files. It extends FileCollection to add hierarchy query and manipulation methods.

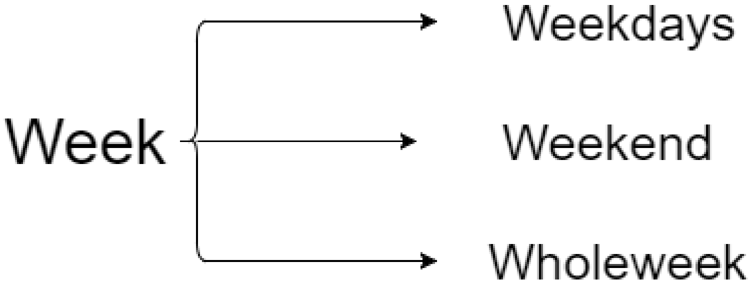
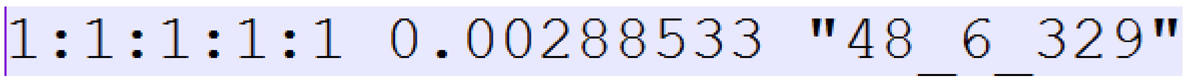


Figure above illustrates the temporal dimension of each file in a single week

In this spatio-temporal data each record is a branch or leave of the hierarchical structure and each position contains the following field:



```
1:1:1:1:1 0.00288533 "48_6_329"
```

Figure. The format of a record in a file tree

- 1:1:1: 1:1 also known as the ID, identification for Hierarchical structure, for example the first number on the left indicates the highest level of the tree. Moving forward we descend to the lower ones.
- 0.00288533 is the fitting measure of the element, all sections are ordered in a descending manner by this value.
- "48_6_329" This is a reference code assigned by ISTAT [11]to indicate a geographical zone or area, for example 48 indicates the regione(region) and the order two values indicate e sub-levels in the region which are comune and sezione.

This data alone is not enough for a visual representation, looking at the format it's clear that we do not have any kind of geometric indication

[11] The **Italian National Institute of Statistics** (Italian: ***Istituto Nazionale di Statistica***; **Istat**) is the main producer of official statistics in Italy.Its activities include the census of population, economic censuses and a number of social, economic and environmental surveys and analyses.

of the position of a single record, the solution and idea to this problem is adding geometric information to a single record enabling us to pinpoint each record and collocate it somewhere. To execute this we decided to get geometric information based on the reference code in the file.tree.

To do this we introduced a shapefile [12]of the region Tuscany, which is a document that essentially contains geometric information about an area such as boundaries of geographic areas, and points within those area, unfortunately their content are not plain text and, therefore, are very hard to read. The shapefile format grew out of the community of geographers, cartographers, and scientists using Geographic Information Software. To be able to utilize this shapefile of Tuscany, we converted it to a format much easier to access such as GeoJSON.

GeoJSON is a JSON-based standard for encoding geodata for web applications.

Convert to GeoJSON

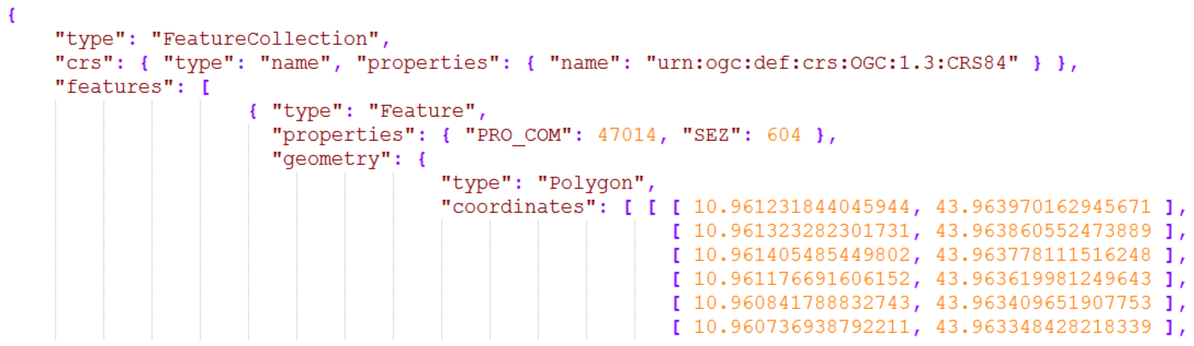
Having a shapefile, our end goal is to get a GeoJSON file that can be accessed and be processed with the file.tree by our back-end application to produce a base data that can be visually represented by the front-end with a very low load on the client.

With the aid of Qgis [13]and ArcMap [14]two cross-platform free and open-source desktops geographic information system (GIS) applications that provide data viewing, editing, and analysis we can obtain a JSON file, the process is straight forward:

[12] Shapefile acquired online from ISTAT portal, data is based on a 2001 Census
[13] http://www.qgis.org/it/site/
[14] http://desktop.arcgis.com/en/arcmap/

- We use ArcMap to change the coordinates system from meters to the most common coordinates such as longitude & latitude.
- With the aid of Qgis we can use its exporting features to remove the unwanted information output a GeoJSON file.

Below you can find the output of the conversion (reformatted and greatly abbreviated here):



```
{
   "type": "FeatureCollection",
   "crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
   "features": [
                  { "type": "Feature",
                    "properties": { "PRO_COM": 47014, "SEZ": 604 },
                    "geometry": {
                                   "type": "Polygon",
                                   "coordinates": [ [ [ 10.961231844045944, 43.963970162945671 ],
                                                      [ 10.961323282301731, 43.963860552473889 ],
                                                      [ 10.961405485449802, 43.963778111516248 ],
                                                      [ 10.961176691606152, 43.963619981249643 ],
                                                      [ 10.960841788832743, 43.963409651907753 ],
                                                      [ 10.960736938792211, 43.963348428218339 ],
```

In a typical GeoJSON style, we can see, first of all, that this is all on giant object. The object has a type of `FeatureCollection`, followed by `features`, which is an array of individual `Feature` objects. Each one of these `Features` represent a sezione(section) of Tuscany, you can actually see in the properties the reference `SEZ`.

The most interesting part is under `geometry`. This is where the features `type` is specified, followed by the many coordinates that constitute the feature's boundary. Within the `coordinates` are sets of longitude/latitude pairs, each one as a small, two-value array.

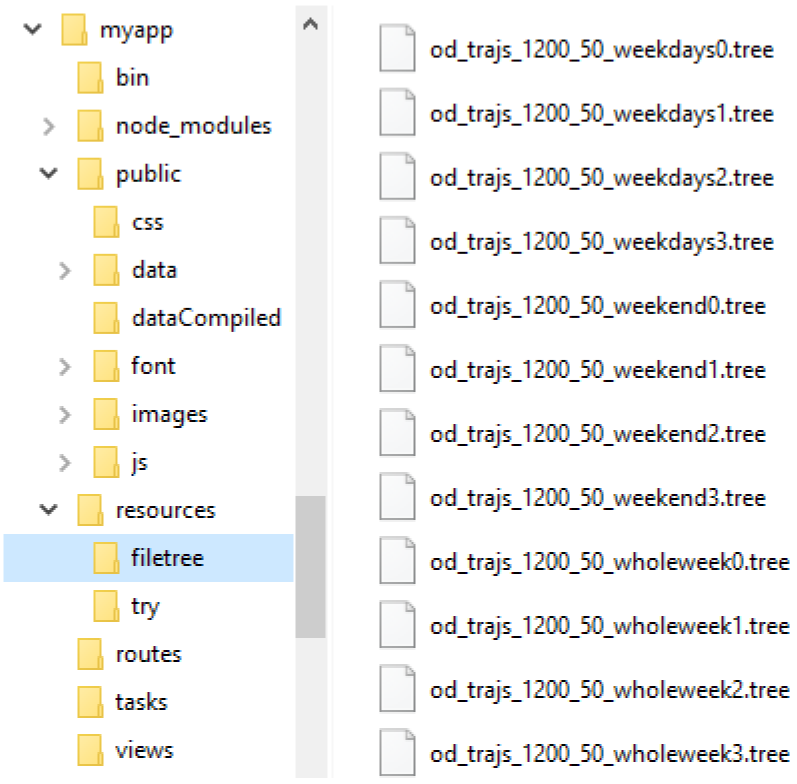
The longitude and latitude together constitute an enormous grid that encircles Tuscany. This is convenient cause we can map data value to display values.

3.3.2 Data pre-processing

The previous subsection described the base information and data needed for a visual representation. Knowing the data at hand is filled with the information needed and ready to be processed, in this section we explain how the data is transformed into something suitable for visualization and analysis. For the fact that D3.js is our framework for visually representing data we have to carefully consider the cons. In fact, D3 loads data into the browser memory therefore the size of the data is important, a smaller file can result in a loading time reduction hence gaining in performance. Keeping this in mind our objective is to:

- Remove data we don't need
- Merge each file.tree with the geometries into a new file extension to obtain data that can be processed easily by D3.
- Output data that can be easily be processed without loading the client.

Without wasting time, we shall now explain how the back-end application process our input data.



```
myapp                          od_trajs_1200_50_weekdays0.tree
  bin                          od_trajs_1200_50_weekdays1.tree
  node_modules                 od_trajs_1200_50_weekdays2.tree
  public                       od_trajs_1200_50_weekdays3.tree
    css                        od_trajs_1200_50_weekend0.tree
    data                       od_trajs_1200_50_weekend1.tree
    dataCompiled               od_trajs_1200_50_weekend2.tree
    font                       od_trajs_1200_50_weekend3.tree
    images                     od_trajs_1200_50_wholeweek0.tree
    js                         od_trajs_1200_50_wholeweek1.tree
  resources                    od_trajs_1200_50_wholeweek2.tree
    filetree                   od_trajs_1200_50_wholeweek3.tree
    try
  routes
  tasks
  views
```

Figure: Visual structure of the Back-end side

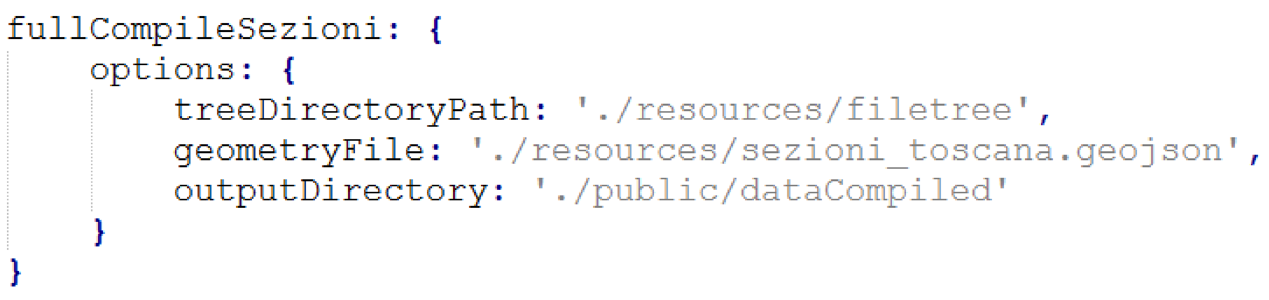
The figure above illustrates the structure of the back-end, as you can see on the left we have a hierarchical view of the folders, the main folder on which we should concentrate is *filetree* in which all the file.trees for the month of May 2011 are stored, in fact you see them on the right side of the figure above. Giving a quick look at the right side of the picture you will easily notice that each file refers to the 3 categories of a week that is weekdays (Monday - Friday), weekends (Saturday & Sunday) and the whole week.

Since we have a group of data to be processed, the choice for node.js was also based on the fact of having various modules that can run multiple task, in our case processing multiple files in on step.

Using Grunt[15] a JavaScript Task runner, we can simplify its definition to automation, it basically gives us the opportunity of performing repetitive task in a single step.

In our case the idea is to merge the file trees with a GeoJSON containing geometries and finally convert the files into a folder in one go to be visualized in a second moment by the front-end.

Going in to the details the first step was done by creating a Gruntilefile that contained all the information to define out task of processing data.



```
fullCompileSezioni: {
    options: {
        treeDirectoryPath: './resources/filetree',
        geometryFile: './resources/sezioni_toscana.geojson',
        outputDirectory: './public/dataCompiled'
    }
}
```

As you can see above, in the Gruntfile.js the task `fullCompileSezioni` is defined, in this task we add 3 options that are simply the directory in

[15] http://gruntjs.com/

which the tree files a located `treeDirectoryPath`, the json file that contains our geometry (`geometryFile`) and the output folder (`outputDirectory`), in which the merged files will be saved once merged and converted.

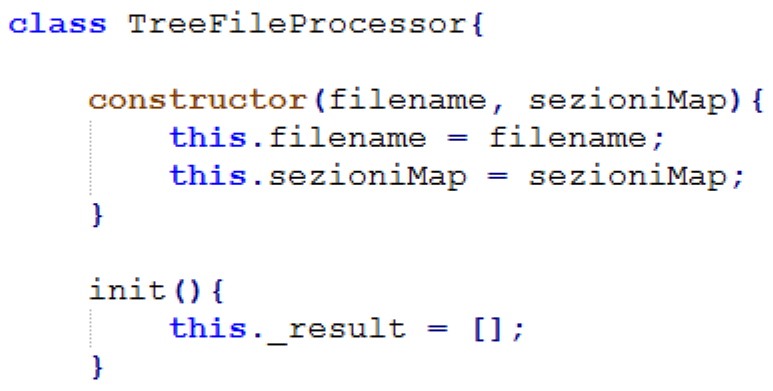
Moving to the `fullCompileSezioni`, for the fact that any file under node.js is alone and not part of a global domain, the only way to include a function is to use `require`.

With `require` we can include any line of code of a function or library implicitly

```
// Library within node js
const fs = require('fs');
const stream = require('stream');
const readline = require('readline');
const path = require('path');

// Libraries download with npm install
const async = require('async');
```

As you can see we have included various libraries



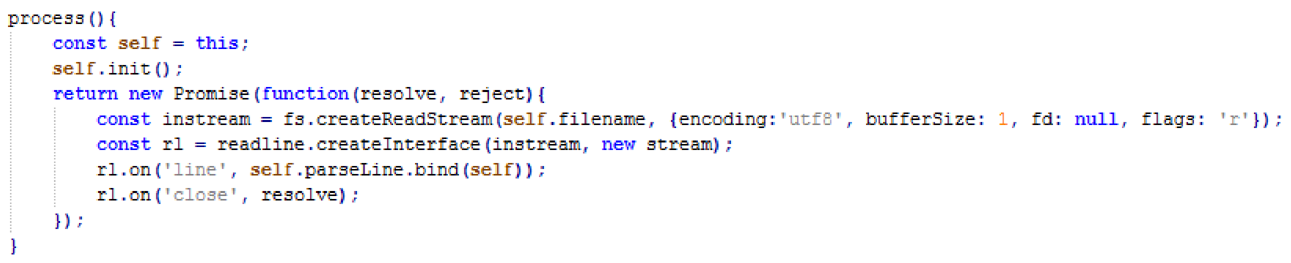
```
class TreeFileProcessor{

    constructor(filename, sezioniMap){
        this.filename = filename;
        this.sezioniMap = sezioniMap;
    }

    init(){
        this._result = [];
    }
```

The `TreeFileProcessor` is a class that includes all operations of the process, as a constructor it takes the `filename` (the path of the tree file) that has to be processed and the Map of the sezioni(sections).

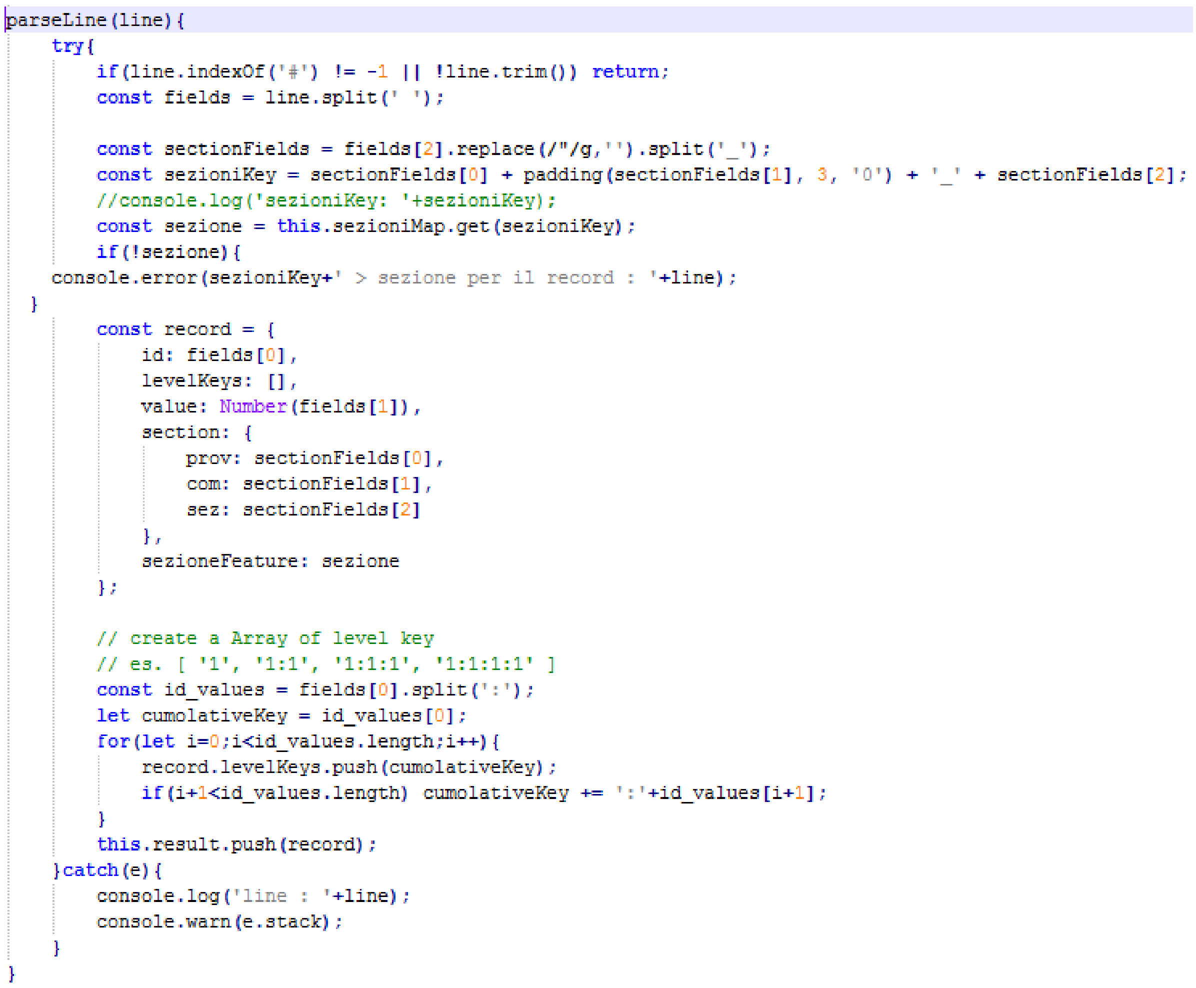
In `result` we put the parsing output of the file and with `init` we initialize an empty array.



```
process(){
    const self = this;
    self.init();
    return new Promise(function(resolve, reject){
        const instream = fs.createReadStream(self.filename, {encoding:'utf8', bufferSize: 1, fd: null, flags: 'r'});
        const rl = readline.createInterface(instream, new stream);
        rl.on('line', self.parseLine.bind(self));
        rl.on('close', resolve);
    });
}
```

`Process()` just starts the process that begins filling `result` with records that have been parsed from the fileTree. Note the term `promise`, this was introduced to make sure the reading of files from the disk is done in an asynchronous way, going into to the details, we create a stream with `fs.createReadStream` and then use it with `readline`, this simply generates 2 asynchronous events ("`line`", "`close`"). Line is called for every file read while close comes at the end. When `close` is emitted we call the resolve of the Promise. All this is done to able to multi-thread because JavaScript is single thread, applying this we can run a process on multiple files without waiting for each one to finish before starting a new one.

On the event of `'line'` of the stream, we pass our function `parseLine` , you can see it in the figure below



```
parseLine(line){
    try{
        if(line.indexOf('#') != -1 || !line.trim()) return;
        const fields = line.split(' ');

        const sectionFields = fields[2].replace(/"/g,'').split('_');
        const sezioniKey = sectionFields[0] + padding(sectionFields[1], 3, '0') + '_' + sectionFields[2];
        //console.log('sezioniKey: '+sezioniKey);
        const sezione = this.sezioniMap.get(sezioniKey);
        if(!sezione){
    console.error(sezioniKey+' > sezione per il record : '+line);
  }
        const record = {
            id: fields[0],
            levelKeys: [],
            value: Number(fields[1]),
            section: {
                prov: sectionFields[0],
                com: sectionFields[1],
                sez: sectionFields[2]
            },
            sezioneFeature: sezione
        };

        // create a Array of level key
        // es. [ '1', '1:1', '1:1:1', '1:1:1:1' ]
        const id_values = fields[0].split(':');
        let cumolativeKey = id_values[0];
        for(let i=0;i<id_values.length;i++){
            record.levelKeys.push(cumolativeKey);
            if(i+1<id_values.length) cumolativeKey += ':'+id_values[i+1];
        }
        this.result.push(record);
    }catch(e){
        console.log('line : '+line);
        console.warn(e.stack);
    }
}
```

`ParseLine` takes a single line of the fileTree and transforms it into a JavaScript object adding the geometry of the corresponding section(sezioni), this also creates an array containing all the levels.

In map, we have `chiavesezione => geometria sezione,` the object created is added to the result of the array with :

```
this.result.push(record);
```

Process is closed by a `try{}catch{},` in the possible event of a problem during the parsing, for example the line does not have the correct format that we need for merging.

The final part of this scripts is a crucial section, because we now have to put the result of the parsing together, this is done to relieve the front-end of processing every single element. By doing this the can read a set of information merged together rather than reading every single file incurring in performance issues.

Clusterize is the function that takes the result of the parsing and creates various sets(groups), the parameters are:

```
clusterize(level, threshold){
        const self = this;
        return new Promise(function(resolve, reject){
```

- **Level**: the file tree comes with levels to explore the data, so it was obvious to add the level to be able to filter the levels data we want to group.
- **Threshold**: A threshold below which sets are discarded.

The first line of the function is to "save" the reference of the current class, because inside the `Promise` **`on this`** refers to the object of the `Promise` created previously.

```
const clusters = new Map();
const maxSezioneCluster = new Map();
```

`cluster` saves all the sets, meanwhile in `maxSezioneCluster` will go all the big sets determined by a `sezioni` so that in a second moment we can delete a `sezioni` from set making sure that is not assigned to any other set except the biggest one.

```
async.forEach(self.result, function(record, eachcb){
```

`Async.forEach` [16]is an asynchronous function like database calls, this is very useful to control the flow of the operations within the function.

Without extending further details we shall explain what happens for every record in result. In Clusterize we determine the id of the cluster associated to the record also considering the level, using the array of all possible level, by doing this we create a key.

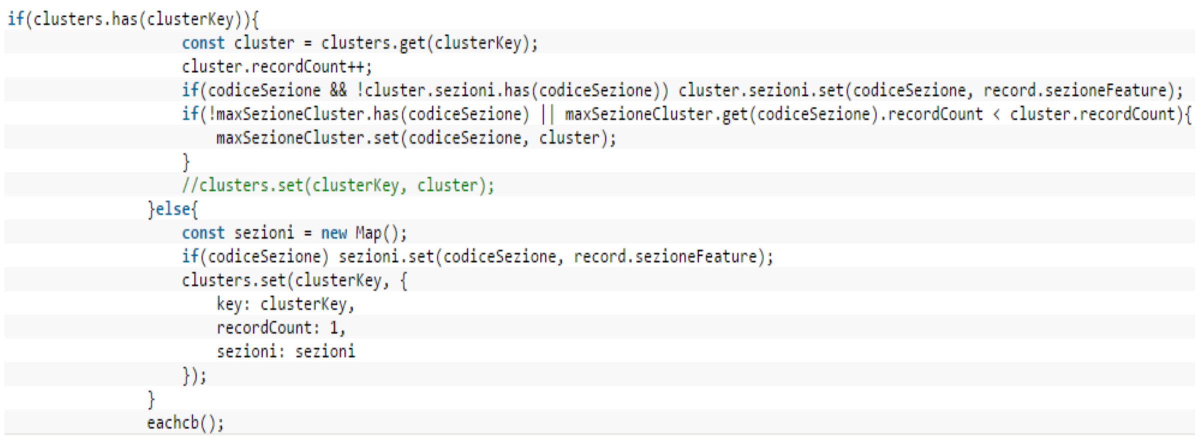
```
let clusterKey = null;
```

[16] https://caolan.github.io/async/

```
      if ( record.levelKeys.length-level<0 || level<1 ) clusterKey = record.levelKeys[0];
                  else clusterKey = record.levelKeys[record.levelKeys.length - level];
                  if (clusterKey == null) return eachcb();

                  let codiceSezione = null;
                  if(record.sezioneFeature) codiceSezione = record.section.sez;
```

The array of the level will be in this form ex. ['1', '1:1', '1:1:1', '1:1:1:1'] this means that level 1 will have the id's '1:1:1:1' and the lower levels will be -1 and so on.



```
if(clusters.has(clusterKey)){
                    const cluster = clusters.get(clusterKey);
                    cluster.recordCount++;
                    if(codiceSezione && !cluster.sezioni.has(codiceSezione)) cluster.sezioni.set(codiceSezione, record.sezioneFeature);
                    if(!maxSezioneCluster.has(codiceSezione) || maxSezioneCluster.get(codiceSezione).recordCount < cluster.recordCount){
                        maxSezioneCluster.set(codiceSezione, cluster);
                    }
                    //clusters.set(clusterKey, cluster);
                }else{
                    const sezioni = new Map();
                    if(codiceSezione) sezioni.set(codiceSezione, record.sezioneFeature);
                    clusters.set(clusterKey, {
                        key: clusterKey,
                        recordCount: 1,
                        sezioni: sezioni
                    });
                }
                eachcb();
```

A control has also been added to make sure that in the unlikely case we need to view level 6 and the record doesn't have a 6th level we consider the first level as the id. The id gives us 2 possibilities, and that is the cluster with the id is in the map or not, if its present we increase a counter that serves us as an indicator of the set size(cluster).

We also update the `maxSezioneCluster` determining if it's the largest set, in the case the cluster never existed, we create on and insert it in Map, its structure is very simple: **`id`**, **`counter`** and the list of the **`sezioni`**.

Cleared that with `eachCb` we directly inform the `async` that all the operation on the current object are done, and when all `eachCB` [17]are done `async` we then execute the next function.

```
// remove not interesting cluster and convert to Array
```

[17] CallBacks

```
clusters.forEach(function(cluster, key, map){

   // remove a sezione from all cluster except the cluster with max recordCount
   cluster.sezioni.forEach(function(sezione, key, map){
         const codiceSezione = parseInt(sezione.properties.SEZ).toString();
         if(maxSezioneCluster.get(codiceSezione) != cluster) map.delete(key);
    });


// remove a cluster if recordCount is less than threshold or id does not have sezioni
 if(cluster.recordCount<threshold || cluster.sezioni.size == 0){
     map.delete(key);
 }else{
     cluster.sezioni = Array.from(cluster.sezioni.values());
 }
});
 resolve(Array.from(clusters.values()));
           });
```

Moving forward we delete from the cluster the sezioni in the largest set if the current cluster is not the largest, this is done to avoid having duplicate sezioni in different cluster creating and overlapping effect, afterwards we control the `threshold`, making sure that **if** the sets is too small or doesn't have sezioni we can delete it **else** the Map will be converted to an Array. This part is very important because as stated in the pages above a JSON is a container of arrays.

```
resolve(Array.from(clusters.values()));
```

Moving to resolve, the result will be passed to the next function which is the final part of the script.

```
writeFile(outputDirectory, suffix, data){
        const self = this;
        return new Promise(function(resolve, reject){
            let outputname = self.filename.split('_');
            outputname = outputname[outputname.length-1];
            const outputPath = path.join(outputDirectory, outputname.split('.')[0] +'_'+suffix+'.json');
            const dataJSON = JSON.stringify(data);
            fs.writeFile(outputPath, dataJSON, 'utf8', (err) => {
                if (err) return reject(err);
                console.log('File Saved : '+outputPath);
                resolve();
            });
        });
    }
```

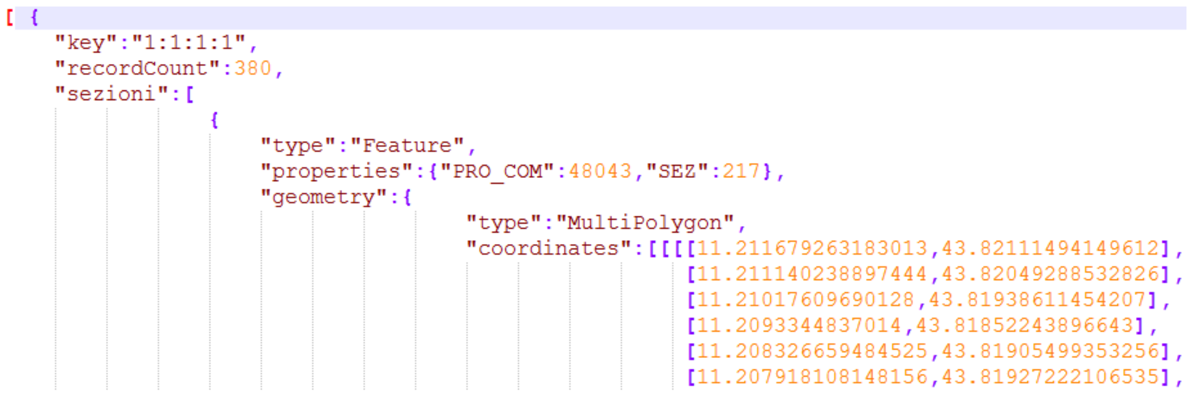
`WriteFile()` only writes the parameter data to a file , with `outputDirectory` we choose a folder where all the new files go, meanwhile suffix is for the name of the file, in fact to avoid confusion all new files will have the same name from which they originated plus a suffix.

The data object is converted in JSON with `JSON.stringify` e and written with `fs.writeFile`, with this all functions have been described.

3.3.3 Final data format

After a detailed look at how the back-end works, we can now have a look at how the final data format looks like.

This data will be managed by D3.js in the front-end.



```
[ {
    "key":"1:1:1:1",
    "recordCount":380,
    "sezioni":[
                {
                    "type":"Feature",
                    "properties":{"PRO_COM":48043,"SEZ":217},
                    "geometry":{
                                "type":"MultiPolygon",
                                "coordinates":[[[[11.211679263183013,43.82111494149612],
                                                 [11.211140238897444,43.82049288532826],
                                                 [11.21017609690128,43.81938611454207],
                                                 [11.2093344837014,43.81852243896643],
                                                 [11.208326659484525,43.81905499353256],
                                                 [11.207918108148156,43.81927222106535],
```

As you can see this file resembles the GeoJSON seen previously, but its slightly different. First of all, the file extension is in JSON (JavaScript Object Notation) which is a lightweight data-interchange format. Note the term **lightweight**, like a previously stated load and weight are of essence to the application. JSON is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON uses JavaScript syntax, but the JSON format is text only, just like XML, thus, JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the

C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. Text can be read and used as a data format by any programming language. These properties make JSON an ideal data-interchange language and an easier-to-use alternative to XML. JSON is built on two structures:

- a collection of name/value pairs. In various languages, this is realized as an object, record, dictionary, hash table, keyed list, or associative array.
- an ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures. Secondly we now have `key:”1:1:1:1` that indicates the level, `recordCount` is the number of `sezioni`(sections) in this set and `sezioni` includes all merged `sezioni` from the tree file and all `coordinates` respectively.

Chapter IV

Mobility Borders

In the previous chapters, we have extensively described all notions needed in one way or the other to understand the field in which we are operating, presenting the data on which information as to be extracted and conveyed to the end user. This final chapter tackles the last piece of the puzzle and that is how to convey information to the end user. Just as described in the previous chapter various technologies have been adopted for this purpose such as D3.js, and a complete state of the art scouting has been done to provide the best solution possible.

The development of the application is based on various studies of user interaction, in the sense that based on what the user wants to see the application has to create dynamically [18]the result.

In this chapter the application interface is presented showing an example of analysis that can be done with real data. The available data refers to a large dataset of human trajectories, a GPS dataset from more than 150k vehicles in Italy(Tuscany) covering the time period from May 1st to 31st year 2011. The process leading to this is straight forward, our data in input (file trees + GeoJSON) are merged and process by the backend in to a *.json* with geometries. Done with that, the data is passed to the front-end to be visualized with D3.js and various technologies. The visualization panel provided in Mobility Borders is essentially a map visualization panel, a statistic panel and a control panel that allows to control the type of data displayed.

[18] Characterized by a continuous change, activity or progress

4.1 Map Visualization

The map visualization is divided in two layers, that represent the area of study and that is the region of Tuscany, For the fact that we humans are visual attracted creatures, visualizing data is the best way to convey information during the analytical process. Since our data is based on GPS data it was obvious that our choice for data visualization would be a map. Data is beautiful, and geographic data is even more visual pattern oriented than most. These characteristics are lost when the data is contained in spreadsheets. With maps, we can create a journey of discovery, like our case by grouping locations, or by being able to drill down further into a specific location, maps encourage further discovery and interaction.

Just think about it when someone says "*Hey, look at this*" it's amazing what a new insight can spark, this provides new findings and perspective encourages collaboration, and before we know we people or the analyst asking the right questions about the data.

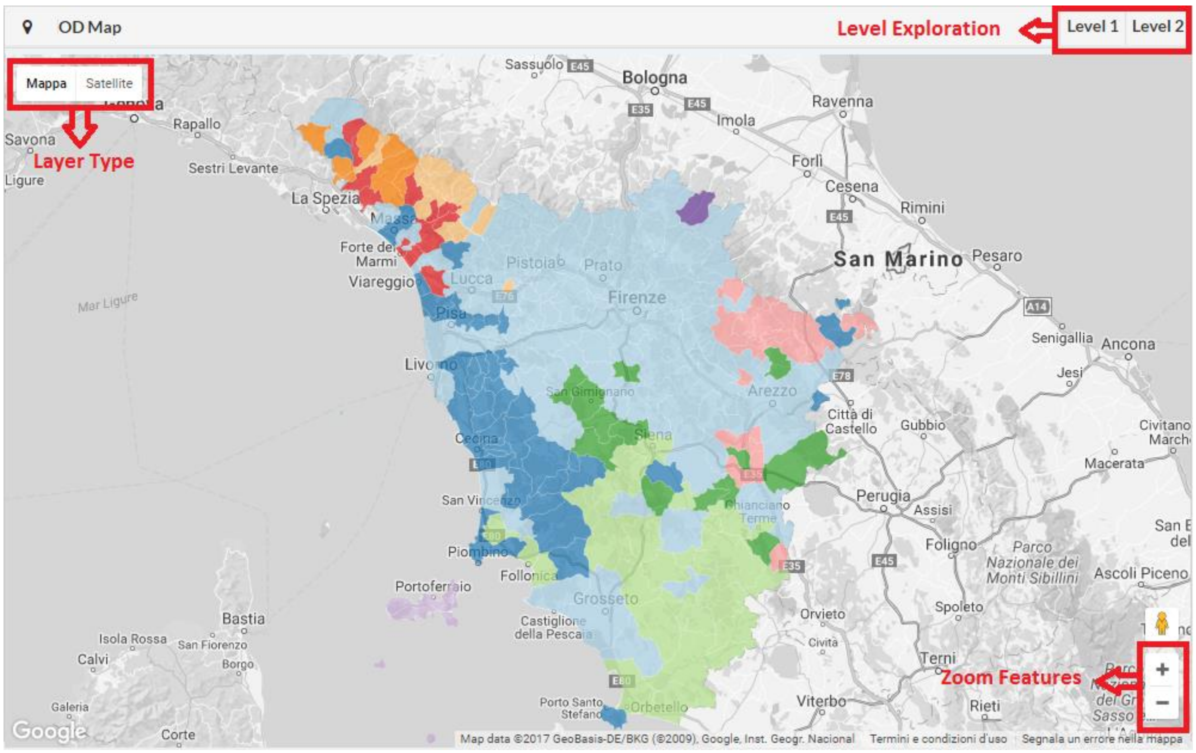
Like stated in the previous paragraphs 2 layer of map visualization have been adopted, A choropleth and Google Maps API[19].

A choropleth refers to a geomap [20]whose areas are filled in with different values (light or dark) or colour to reflect associated data values. The US, so called "red state, blue state" choropleth maps showing the Republican and Democratic leanings is an example.

The Google Maps API allows the embedding of Google Maps onto web pages of outside developers, using a simple JavaScript interface or a Flash interface. It is designed to work on both mobile devices as well as traditional desktop browser applications. This choice was used to give a real layer of confrontation with a D3 generate choropleth.

[19] https://developers.google.com/maps/documentation/

[20] map of a country, continent, or region map, with colours and values assigned to specific regions.



The figure above shows the result of the 2 maps overlaid, as you can see in the foreground the choropleth is place with the colour of the areas indicating a cluster, meanwhile in the background a real google is integrated giving us a way to actually identify areas in the cluster. If you are familiar with google maps, you will notice that the basic control can be found on our map as well.

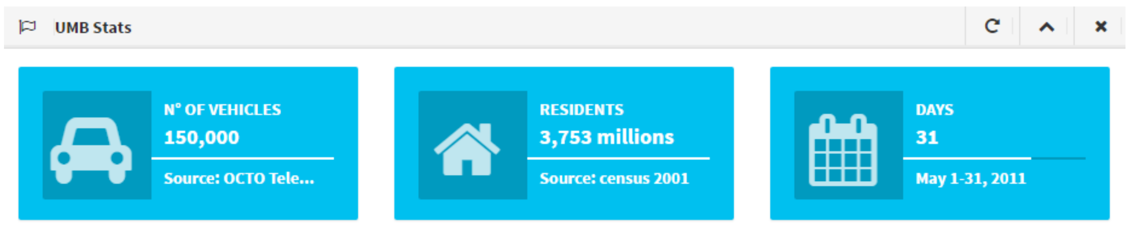
The main controls on the map are:

- *Level Exploration*: This enables us to visualize the clusters determined by level 1 & 2 of the Infomap hierarchy merged. json file. By default, all layers are set to level 1, the maximum drill down level is up to 3[21].
- *Zoom Features*: The initial resolution at which to display the map is set to 8, with this feature we are able to increase the level of detail to a higher resolution. Giving us the opportunity see better areas (comuni /sezioni) contained in a cluster.

[21] For demonstration purposes, only 2 are available

- *Layer Type:* The Google Maps JavaScript API manages the presentation of objects within layers by rendering their constituent items into one object (typically a tile overlay) and displaying them as the map's viewport changes. Layers may also alter the presentation layer of the map itself, slightly altering the base tiles in a fashion consistent with the layer. In the figure above we selected a RoadMap with saturation value -100 & lightness of 10 to guarantee e major visibility.

Another visual feature of lower interest is the UMB stats panel, in this panel all the basic stat on the input data is reported.



N° OF VEHICLES
150,000
Source: OCTO Tele...

RESIDENTS
3,753 millions
Source: census 2001

DAYS
31
May 1-31, 2011

Figure: Stats Panel

4.2 Control Panel

Another visual component of the web application is the *Control Panel*, following the UIX guidelines [22]we implemented a control panel that makes the transition of information seamless, using buttons for page navigation and data transition has been a common trend for year. In the application, it works well because we need to organize similar tools under the same umbrella, using both horizontal and vertical tab menus. This helps the user determine what he or she wants to visualize.

[22] https://uxdesign.cc/

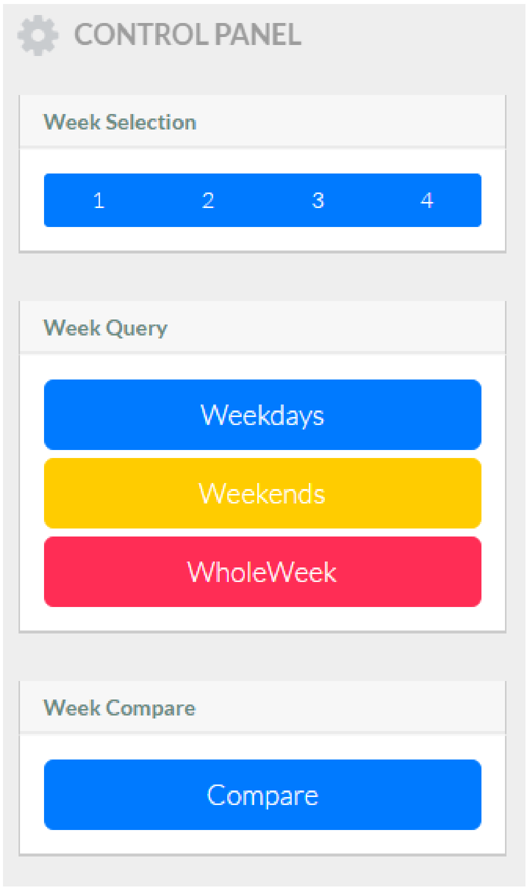


Figure: Control Panel

Looking at the figure above you will notice that all features have been grouped in to 3 different tabs:

- *Week Selection*: The data in input is divided in 4 week, so it's a simple consequence that we have 4 buttons, the week Query below and the level selection all depend on the week selection.

- *Week Query*: a direct dependant of the week selection, this provides a level of filtering, clusters on the map can be visualized weekdays (Mon - Friday), weekends (Sat - Sun) and the whole week (Mon - Sun).

- *Week Compare*: This feature was added in the final stages of development to provide a multi-view layout to compare the clustering effect in different weeks.

4.3 Analysis

In the following pages, we present some examples of analysis that can be done on the real data using Mobility

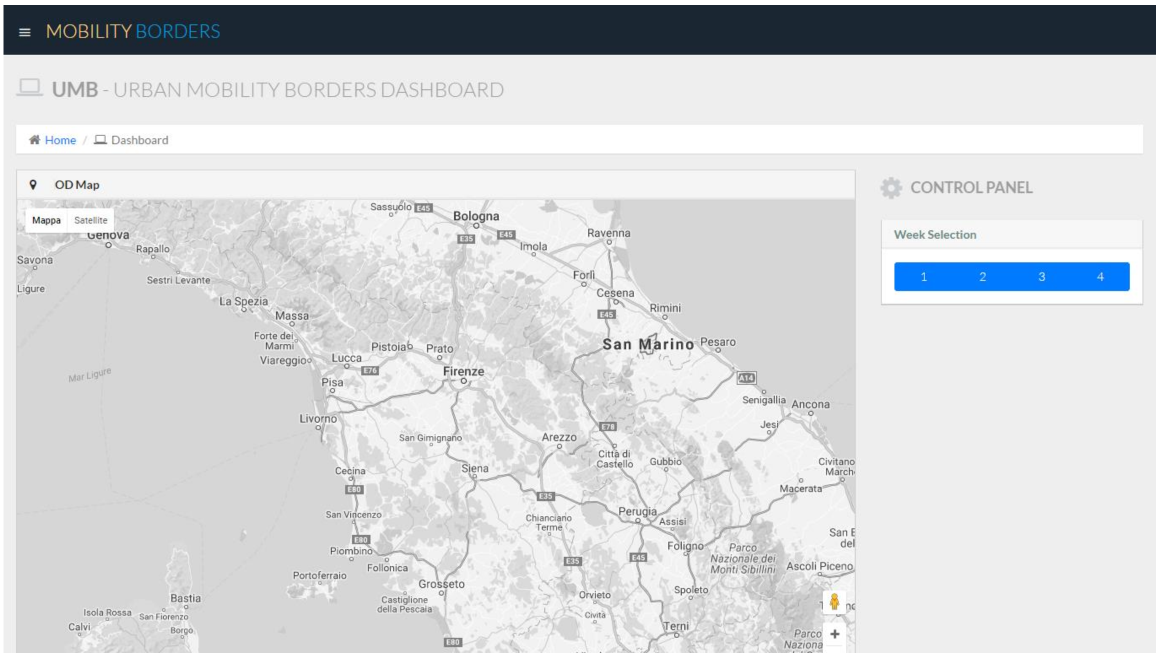


Figure: The Final User interface after back-end processing (* cluster areas colours are assigned casually)

Once the data is sent and loaded to the front-end, we get an empty map layer. By selecting a week in the control panel, we load the data to be visualized.

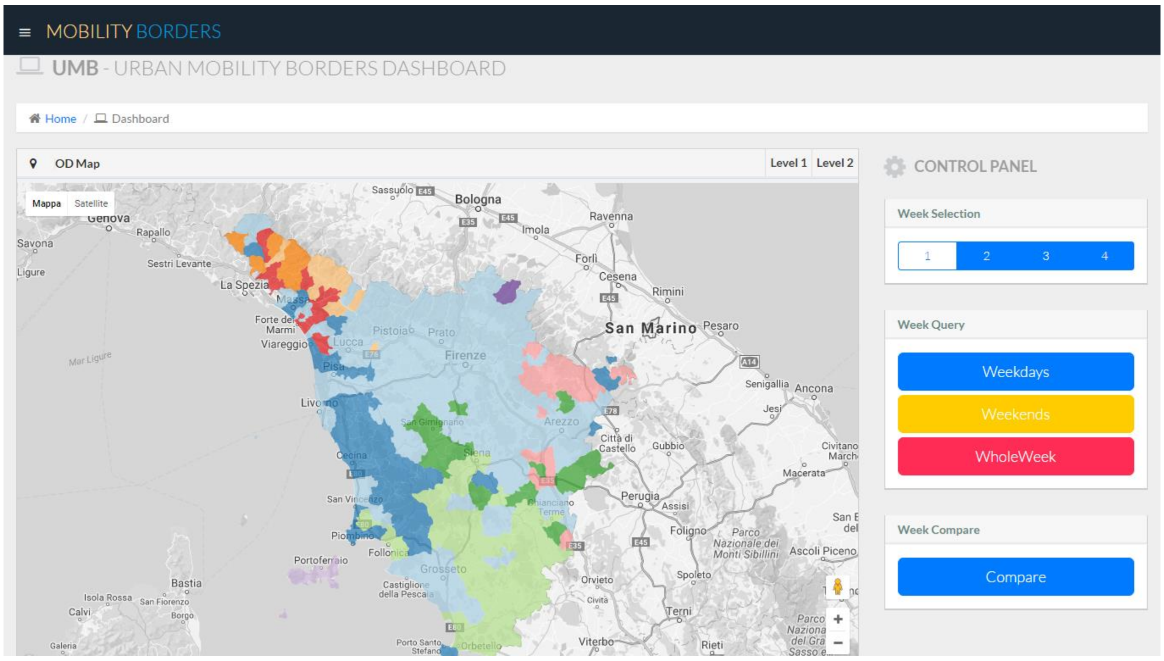
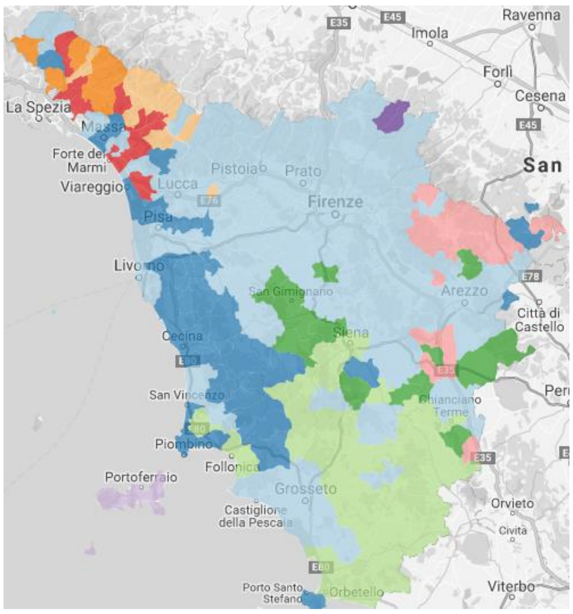


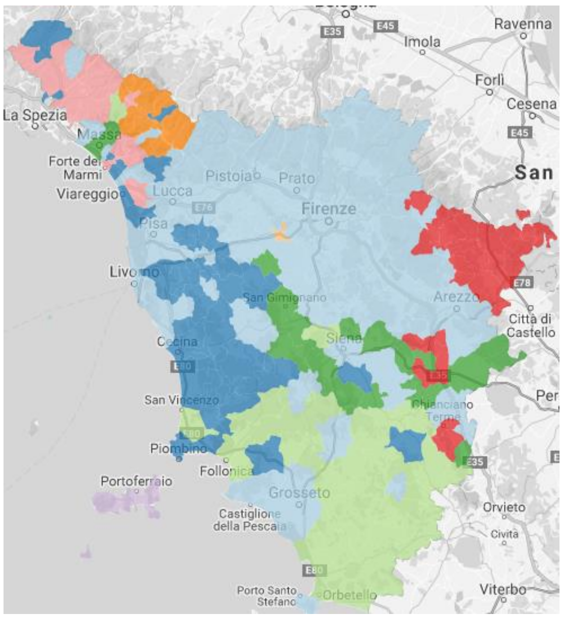
Figure: User Interface with loaded data

Based on week selection, the front-end elaborates the data and extracts the requested information. As you can see in the figure above on the empty Google Maps layer appears (in this case) the map of Tuscany with coloured area that represent the created cluster. Taking a closer look, we can note different clusters for week 1(weekdays) exactly 9 of which the biggest cluster is that of the area that covers Firenze, Pistoia, Prato and parts of Arezzo and Siena, this can come as a surprise but in reality, it as to be noted that the 1st of May in Italy is a public holiday, so we can suppose that during a public celebration registered user with Octo Telematics made trips to pass time in the capital of Tuscany[23]. Florence being one of the biggest city would have attracted a high number of people thus being a major cluster.

Another very interesting observation would be looking at the evolution of cluster taking into account the whole week of the 4 weeks.



Weekdays **week 1**

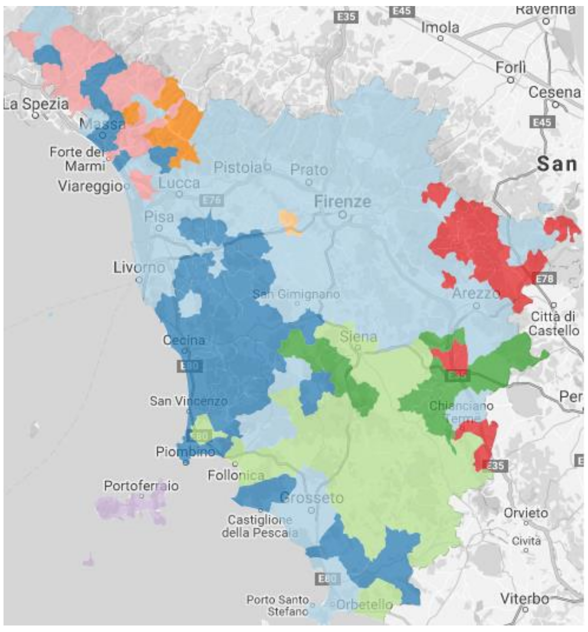


Weekdays **week 2**

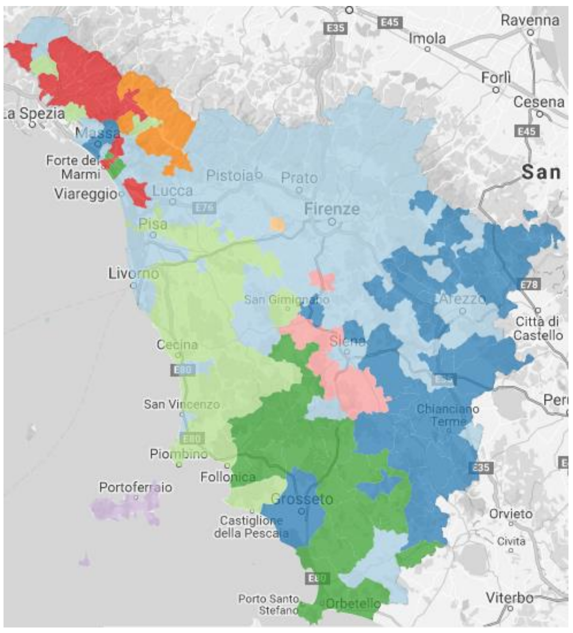
Confronting week 1 & week 2 we can notice that the cluster of Firenze is uncharted this is due to the fact that in the year 2011 there were many events in Firenze, especially concerts of Italian and international stars. It would be obvious to assume that the flux and movement would be directed

[23] During that year shops where open and also concerts where organized.

manly to that end. This would in part justify the dominance even thou the representation is an approximation of the areas in Tuscany. Examining other clusters, Grosseto is another major area in Tuscany, the representation is partial in our result and it extends much more than the original administration limit. Arezzo stands in-between Siena & Firenze, the cluster created is the worse representation of all the provinces.



Weekdays **week 3**



Weekdays **week 4**

Looking at the evolution from the first 2 weeks to the couple of week 3 and 4, it pops instantly to the eye that week 4 resembles in part the region of Tuscany, you can actually distinguish the provinces of Grosseto, Arezzo, Mass-carrara, and in part Siena. In all four weeks Firenze incorporates many areas in an attempt to clearly state that the destination of most trips from our analysed set of data is Firenze.

Chapter V

Conclusion & future Developments

In this report, all fundamental aspects in the development of the project Mobility Borders have been addressed. Starting from ground knowledge on various topics needed to comprehend the field in which we are operating to data visualization notions. The application will be briefly demoed to show its potential.

On the cause of feature development's, it has to be noted that the running application is a prototype version that has to be expanded and developed further. The various type of data based on community discovery algorithms apart from Infomap that can be introduced to have a more detailed look at how our borders can be reconstructed from different models. Another essential feature development will be the integration of different visual interaction techniques to improve the overall experience of the user. A very important aspect that has to be noted is the detailed fidelity of the of the visual result, in the running application we have an approximation of the real result cause by the use of comuni based JSON rather than a detailed version such as the sezioni.

The development of this project required the combination of technical skills and humanistic competencies, in order to create a product that responded in one way or the other to final objective. In particular, many technologies have been used such as Node.js and JavaScript as well as techniques for user interface development, needed for communicating information to the user.

Webliography

- **W3C, HTML5**
  http://www.w3schools.com/html/html5_intro.asp

- **W3C, CSS3**
  http://www.w3schools.com/css3/default.asp

- **W3C, PHP**
  http://www.w3schools.com/php/default.asp

- **PHP**
  http://php.net/
  http://www.oracle.com/technetwork/topics/php/underground-phporacle-manual-098250.html

- **jQuery**
  http://jquery.com

- **JavaScript**
  http://www.w3schools.com/js/

- **Node.js**
  https://nodejs.org/it

http://gruntjs.com/getting-started

http://expressjs.com/it/

- **TopoJSON**

  https://github.com/topojson/topojson

- **Grunt.js**

  http://gruntjs.com/

- **D3.js**

  http://d3js.org

  https://github.com/mbostock/d3/wiki

  http://alignedleft.com/tutorials/d3/

Bibliography

- Adrienko, Natalia and Gennady Adrienko (2011). “Spatial generalization and aggregation of massive movement data”. In: vol. 17. 2. IEEE, pp. 205–219.

- Andrienko, Gennady et al. (2009). “Interactive visual clustering of large collections of trajectories”. In: IEEE Symposium on Visual Analytics Science and Technology (VAST) 2009. IEEE, pp. 3–10.

- Andrienko, Natalia and Gennady Andrienko (2012). “Visual analytics of movement: An overview of methods, tools and procedures”. In: Information Visualization.

- Andrienko, Natalia and Gennady Andrienko (2013). “A visual analytics framework for spatio-temporal analysis and modelling”. In: Data Mining and Knowledge Discovery 27.1, pp. 55–83.

- Bostock, Michael, Vadim Ogievetsky, and Jeffrey Heer (2011). “D3 data-driven documents”. In: IEEE Transactions on Visualization and Computer Graphics 17.12, pp. 2301–2309.

- Bostock, Michael and Jeffrey Heer (2009). “Protovis: A graphical toolkit for visualization”. In: IEEE Transactions on Visualization and Computer Graphics 15.6, pp. 1121–1128.

- C. Ratti, S. Sobolevsky, F. Calabrese, C. Andris, J. Reades, M. Martino, R. Claxton, and S. H. Strogatz, “Redrawing the map of great britain from a network of human interactions," PLoS ONE, vol. 5, no. 12, pp. e14248+, Dec. 2010.

- S. Rinzivillo, S. Mainardi, F. Pezzoni, M. Coscia, D. Pedreschi, and F. Giannotti, "Discovering the geographical borders of human mobility," in Kunstliche Intelligenz, 2012, p. In press.

- M. Rosvall and C. T. Bergstrom, "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems," PLoS ONE, no. 6(4), p. e18209, Apr. 2011.

- R. Trasarti, F. Pinelli, M. Nanni, and F. Giannotti, "Mining mobility user profiles for Carpooling," in KDDM, 2011, pp. 1190-1168
- Scott Murray 2012. *Interactive Data Visualization for the Web*. O'Reilly

- Julie Steele, Noah Iliinsky. *Beautiful Visualization Looking at Data through the Eyes of Experts*, O'Reilly

- Giannotti, Fosca and Dino Pedreschi (2008). *Mobility, data mining and privacy*. Springer

- Kraak M-J, Huisman O. Beyond exploratory visualization of space time paths. In: Miller HJ, Han J (Eds). Geographic data mining and knowledge discovery. Second edition. Taylor & Francis: London, 2009; 431-443

- Guo H, Wang Z, Yu B, Zhao H, Yuan X. TripVista: Triple Perspective Visual Trajectory Analytics and its application on microscopic traffic data at a road intersection. In: Proceedings of the Pacific Visualization Symposium PacificVis 2011, IEEE, 2011; 163-170.

- Spretke D, Janetzko H, Mansmann F, Bak P, Kranstauber B, Mueller M. Exploration through Enrichment: A Visual Analytics Approach for Animal Movement. In Proceedings of 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2011), 2011; 421-424.

- Keim D, Andrienko G, Fekete J-D, Görg C, Kohlhammer J, Melancon G. Visual Analytics: Definition, Process, and Challenges. In: Kerren A, Stasko JT, Fekete J-D, North C (Eds). Information Visualization – Human-Centered Issues and Perspectives. Volume 4950 of LNCS State-of-the-Art Survey, Springer: Berlin, 2008; 154-175.

- Willems N, van de Wetering H, van Wijk JJ. Visualization of vessel movements. Computer Graphics Forum (CGF) 2009; 28(3): 959-966.