Treemob

Expressive mobility data representation through tree-based structures

Dipartimento di Filologia, Letteratura e Linguistica
Corso di laurea Magistrale in Informatica Umanistica
Tesi di Laurea Magistrale

Candidata Marta Fioravanti
Relatori Prof. Riccardo Guidotti, Prof. Salvatore Rinzivillo
Anno accademico 2020 / 2021

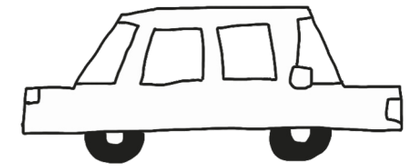
In Supremæ Dignitatis 1343
Università di Pisa

Università di Pisa

Dipartimento di Filologia, Letteratura e Linguistica
Corso di laurea Magistrale in Informatica Umanistica
Tesi di Laurea Magistrale

Treemob

Expressive mobility data representation through tree-based structures



Candidata
Marta Fioravanti
Relatori
Prof. Riccardo Guidotti
Prof. Salvatore RInzivillo
Anno accademico
2020 / 2021

Content

“[...]We must maintain a humanist view of data, relying on our own faculties to tell a story. Second, to improve the discourse surrounding data, we must disavow our fascination with the intricate and complicated by learning how to throw things out.

To provide some background, my own work is focused on visualization of complex data sets, whether the human genome, millions of words of text, or truly important things like showing which American baseball teams are over-paying for star players. But my assumption is that whether you think it's relevant or not, working with ever-larger data sets will remain the domain of design for years to come, as we continue to be inundated with more sources of information, and proportionally fewer ways to handle and understand them.”

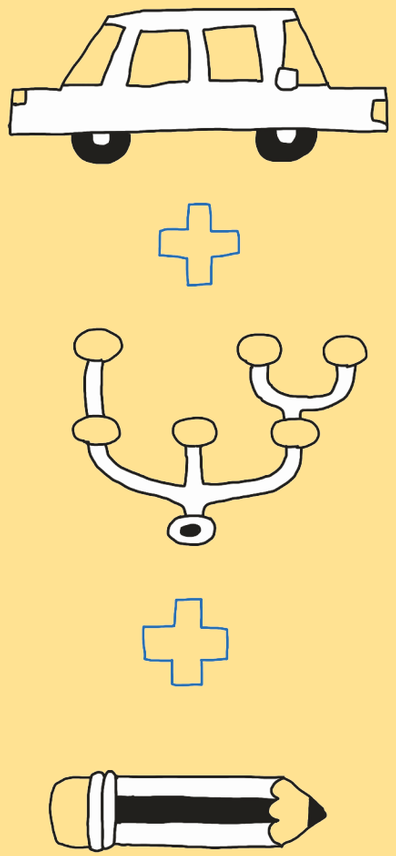
Ben Fry (2009)[1]

Abstract

This thesis is about finding expressive representations about the personal mobility of a user, intending the term expressive both from the point of view of information and of user-centred design, with the aim of allowing people to access their data, understand it and gain awareness about their behaviours, starting from the assumption that expressiveness is not necessarily related to complex systems.

We embrace the individual perspective on movement data and we present a new methodology to represent and study mobility through the aid of tree-shaped structures. We present as a complement a suite of tools named Treemob, which gathers all the methods and classes to perform an analysis, in Python language, and an example of analysis that could be conducted starting from tree-shaped personal mobility data. The proposed experiments can be viewed as a boilerplate for new and deeper studies trying to answer different questions and to explore different contexts.

1. Introduction



In our more and more tracked life, we produce an enormous amount of personal data representing our daily activities and behaviours, including information about the way we move in the space we live[2]. This data includes spatio-temporal information, like the locations we traverse, and the velocity and duration of the travel.

The quantity of the information we produce every day is huge as well as raw and needs some kind of organisation to be useful, but after some preprocessing it starts to reveal how many kinds of knowledge we can extract from it, mainly if we think about Personal Data Analytics[3].

An effective Personal Mobility Profile should organise information in such a way that is feasible for the computational analysis but at the same time that is accessible and understandable for the user.

Global Positioning System (GPS) and Global System for Mobile Communications (GSM)[2] data creates the opportunity to study human and non-human behaviours in a spatio-temporal view. This data has been already used to study traffic patterns, as well as urban planning, sustainable mobility, the design of the public transportation systems or even the mining of attractive locations.

For example, Andrienko et. al. (2011)[4] propose a methodology that exploits visual analytics and clustering techniques to discover the relevant geographic places of a territory by extracting the most important events in a large set of trajectories traversing that zone. Another study, by Larcom et. al. (2017)[5], focuses on observing the travellers' search for a new optimal path after their usual route becomes impassable due to a strike.

These two examples show two of the many different kinds of information we can extract from mobility data: this field combines the approaches of sociological studies with more quantitative methods, and tries to discover mathematical patterns in human behaviour. Mobility data is a powerful tool to understand new facets of human nature and to unveil collective mechanisms at different scales, and will be fundamental to develop environment-friendly solutions in the next few years. A virtuous example of this kind of project is Berlin Autofrei (car-free)[6], which aims to enlarge the pedestrian area of the city and to reduce by consequence the usage of the individual motorised transport. To make similar initiatives feasible, an a-priori planning of the population's mobility needs is necessary, in order to offer adequate alternative public transport services or parallel infrastructures (for example cycle paths) in replacement of the existing ones.

While in the beginning this mobility data analysis was predominantly focused on global mobility, and by consequence considered the trajectories of many users at a time[7,8], recent studies have highlighted the potentiality of individual models[9]; in fact, using a microscope view on the user's digital traces allows the scientists to detect systematic behaviours of the single and highlight patterns that are invisible in a collective perspective. One of the most representative studies that used the individual mobility network studies is the one by Pappalardo et. al. (2015)[10], which revealed the presence of two main classes of travellers: the returners and the explorers. The former group tends to move in a limited set of locations, that by consequence occur very frequently; the latter instead is characterised by the impossibility to be reduced to few locations. The importance of this paper is also due to its interconnection with social studies, since it discusses how these two classes differently contribute to the spread of information, social interactions and diseases.

Another genre of contribution is from Trasarti et. al. (2011)[11], where the focus is on solving practical problems like carpooling recommendation through the analysis of the personal mobility agenda of each user and trying to cluster similar routines to be matched.

Individual models also open new possibilities in terms of user-centred data, since they are suitable to be used in a shared network where each single person can choose how much and which kind of data to make public. This is fundamental since individual patterns alone are often insufficient for the analysis: the comparison between individual data is fundamental, as well as the combination of personal models together with global models.

Another important aspect to be considered is the sensitivity of the movement data: the analyst should always be aware of the ethical issues which could emerge while working with very personal information. In order to preserve the user's privacy rights, a possible solution could be an infrastructure which allows the user to view their data and choose what to share about it. This infrastructure, called a Personal Mobility Profile[3], not only improves privacy but is also able to trigger the user's awareness about their behaviours, that, in the case of mobility, could also mean to build more consciousness about the user's behaviours relative to the pollution they produce and other social themes the researchers want to point the attention to.

The aim of this thesis is to find an efficient and semantically expressive representation of personal mobility data exploiting tree structures, taking de facto a quite unexplored route; in fact, a large number of studies[3, 15, 16] in this field use network structures to perform the analysis, because they well represent spatial relations, and they can also be adapted to store temporal data.

We also complement the analytical work with the development of a visual analytics tool we appositely designed and realised to understand and explain the results obtained, and with a Python module containing all the methods we implemented to generate the mobility tree shaped data.

The final objective is to perform unsupervised learning on the generated trees to find different kinds of mobility habits, and to compare the results obtained observing two different cities, Pisa and Florence.

To do so, we defined a variation of prefix tree containing the trajectories of the user and we rooted it under the most frequent location

detected. Then, we chose as a dissimilarity measure the unordered tree edit distance, a particular type of tree edit distance that works with trees whose branches are not ordered from left to right. Once having computed the distance matrix, we tried different clustering algorithms both on the tree data directly and on their vectorial representation which describes some basic structural attributes.

Contrary to expectations, we did not observe a set of different types of user, but rather a quite dense and unique group, accompanied with a small fraction of noise, no matter the clustering algorithm, nor the input data type, or the cleaning pipeline.

This work is organised in the following structure: we first describe, in the Background section, the main theoretical concepts related to mobility data, trees and graphs, and visual analytics; once these ideas are exposed, in the Related Work we overview a selection of researches that we found useful, or that well explain the field we are operating in; then, in the Problem Formulation and in the Methodology sections we explain the theoretical and operative steps and choices we made to organise the data and to visualise them; after that, in Experiments, we explain which kind of dataset we used, the cleaning pipeline we followed, and of course the results we obtained. The Conclusion, lastly, synthesises the major achievements of this work.

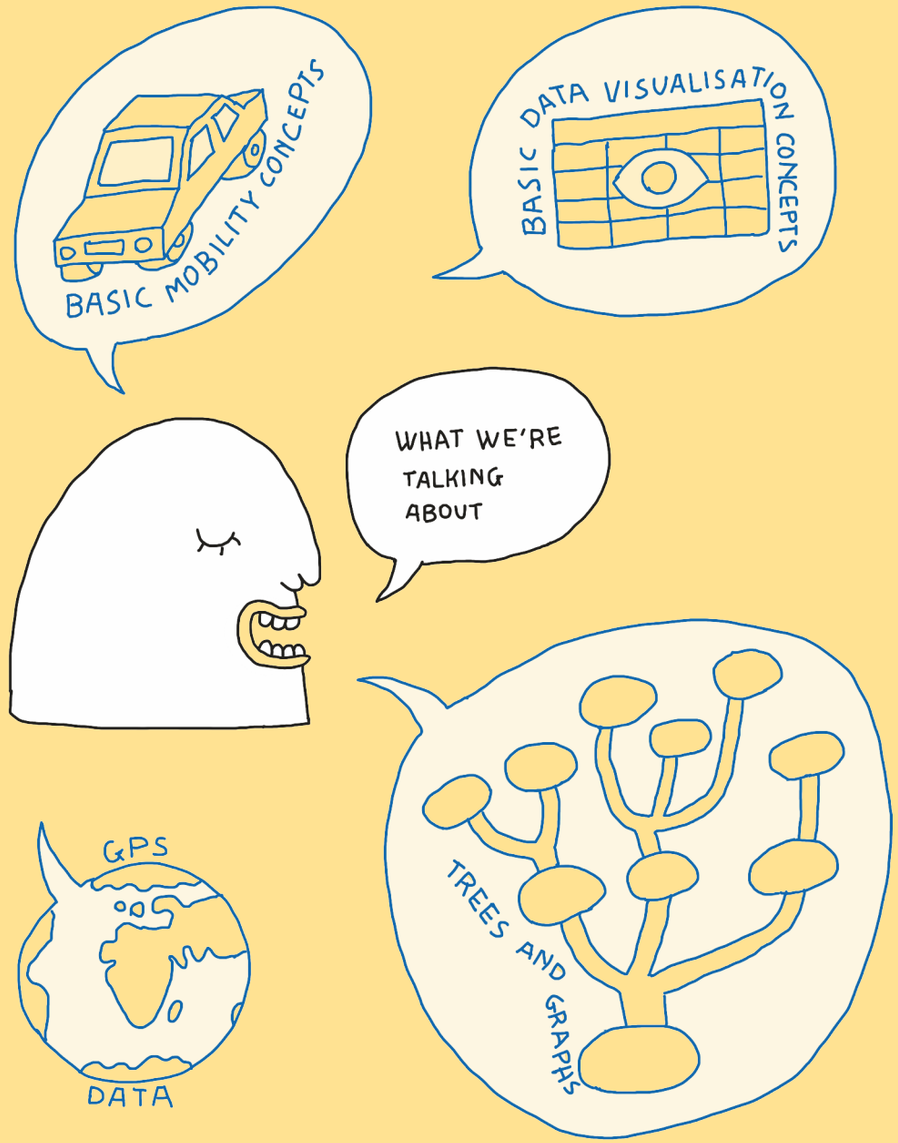
1.1 Note to the visual apparat

One of our priorities during our work has been to make the content of this thesis as much accessible as possible: the editorial product we designed tries to be organically organised and to provide information at different levels of complexity and technicality, to let all kinds of readers to grasp the main concepts behind this work. For this reason, we believed adding some graphic and illustrated support could make the addressed topics more interesting and engaging. The role of the visual content in this thesis is then not only demonstrative (for example, used for showing the results) but also narrative, so as to tell the whole theoretical process that brought to the results. We strongly believe in the central role of the image in scientific communication and that illustrations can bring an important contribution, far from being a mere esthetical complement, but rather being an active part of the development of the discourse. The prejudice which sees the usage of images as less noble with respect to what is commonly defined as writing is as deep-rooted as it is unfounded[12].

We use visual language more than we think, because it is often more functional that plain text; and the visual systems we build are themselves a form of writing, with different functions from typographical writing, but not for that should be considered of less importance. Visual systems are often able to break the plain text limitations to convey messages in a nonlinear yet more direct way. As well explained by the semiologist Roy Harris in his work Rethinking Writing: *from an integrational point of view, the written communication is a form of communication in which the contextualised integration is based on the majority of cases on a visual frame and on visual analogies*[13].

Combining the potentialities of typographical and visual writing we hope to offer a clear and maybe also pleasant way to access the information we want to communicate, according to the principle of synsemia, the *conscious display of the elements in the space in order to communicate through the spatial relations between them*[14].

2. Background



In this section we introduce the theoretical elements related to mobility data analysis.

2.1 Basic mobility concepts

2.1.1 Mobility data structure

The core element constituting a mobility dataset is the trajectory. A trajectory is an ordered sequence of points traversed by the user in a specific moment[17].

Formally we define a trajectory as a series of spatio-temporal points $t_s = <p_1, ..., p_n>$ each of them in the form of a triple $pi = (lon_i, lat_i, t_i)$ where lon_i and lat_i are respectively the longitude and the latitude of the location traversed, while t_i is the timestamp of the event. As already said, a trajectory is chronologically ordered, so $\forall\, 1 \leqslant i \leqslant n$: $ts_i < ts_{i+1}$; so, the i^{th} point of the trajectory ts is identificated as *ts[i]*, while its spatio-temporal coordinates are *ts[i].lon*, *ts[i].lat*, and ts*[i].t*.

A mobility dataset is then a set of trajectories $T = <t_1, ..., t_m>$.

2.1.2 Personal and collective mobility data

We talk about collective mobility data when the trajectories of multiple users are considered simultaneously to build a single model. This could be useful to study cities' viability, for example highlighting the areas and the temporal intervals more subject to flocks or to incidents. Global pattern mining is based on the assumption that studying users' behaviours in mass can effectively produce well-defined classes sufficiently accurate to represent a mobility environment. However, global patterns are not always able to correctly generalise such complex systems like human movements, because of the large variety of behaviours that can occur.

To overcome this issue, personal models were ideated. There are many kinds of personal models each responding to specific research needs. Trasarti et al. (2011)[18] propose a mobility profile extraction pipeline aiming to detect the habits of the user through trajectory pattern extraction. The idea is to aggregate trajectories that are temporally and spatially similar by setting distance thresholds in both the dimensions, and to remove unusual paths. Once these *trip groups* are extracted, their medoid is computed to obtain a prototype of each habitudinary travel; the set of these *routines,* then, represents the mobility profile.

Another way to extract the personal mobility model is the individual mobility network (IMN)[17]; the IMN is obtained by organising in a directed graph the regular locations L_u^R, the irregular locations L_u^i, and the movements M_u of a user, obtaining a network $G_u=(V, E)$ consisting in the set of vertices $V = L_u^R \cup L_u^i$ and the edges $E = M_u$. The graph will consist in a set of trajectories linking regular locations that correspond to the abstraction of the user's habits, and in a set of noise nodes and edges that are their infrequent travels.

Finally, there exist combined models, whose aim is to exploit the strengths of both the approaches by creating hybrid models.

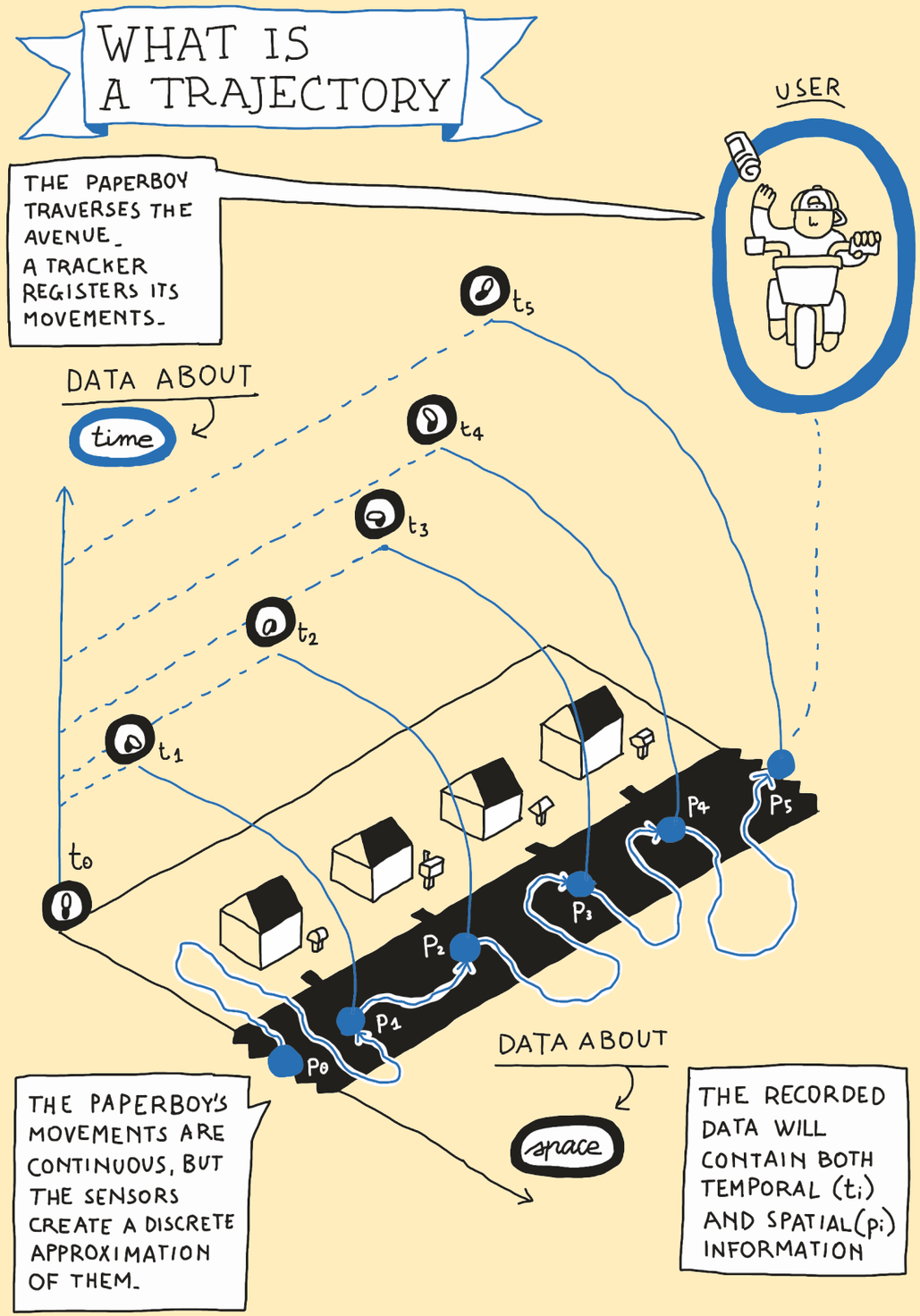


Fig. 1 (right): An illustrated representation of mobility data, taking into consideration a trajectory.

2.2 Trees and graphs

One way to represent and display a set T of trajectories is using a graph. A mobility graph is a directed graph consisting in a set of nodes $N = <n_1, ..., n_n>$ representing the locations appearing in T, and a set of edges $E = <e_1, ..., e_n>$ where each $e_i = <n_a, n_b>$ corresponds to a path from location a to location b. The mobility graphs are often weighted, meaning that to each node and to each edge is associated a weight value corresponding for example to the frequency of its appearance.

2.2.1 Prefix trees

A prefix tree is a data structure predominantly used in information retrieval and in particular for text data because of its capacity to condense huge amounts of data in a simpler structure.

A prefix tree is a variation of a hash tree and can be formalised as a triple $PT = (V, E, root)$, where V is the set of nodes (locations), E is the set of edges connecting nodes (path from one location to another) and root is the virtual root node on the top of the tree, which does not correspond to any location ($root \notin V$). The main characteristic of a prefix tree is that for each node, the set of its siblings does not contain duplicates: this means that all the sequences with a common prefix are grouped in one single branch of the tree; by consequence, two strings $s_1 = <A, B, D, E>$ and $s_2 = [A, B, D, F, G]$ will share the same branch $A \rightarrow B \rightarrow D$; at this point, there will be a split to E and another to F and then G.

2.2.2 Spanning graphs

We put the spanning graphs under this section because as explained in the next chapter this structure was then converted into a tree-shaped one.

A spanning graph is a subgraph G' of a generally edge-weighted graph G which preserves the length of shortest paths in G, up to some error distortion. A spanner is an uncycled graph obtained from a graph with cycles by extracting the optimal paths connecting each node, and by consequence providing a simplification of the original graph. In general, we refer to minimum spanning graphs, because we aim to find the shortest or cheapest path to reach a location from one another. In the case of personal mobility data approximation, on the contrary, we could be interested in representing the most frequent connections, using by consequence a maximum spanning graph. In practice, the method of extraction remains invariant but we assign weights to paths corresponding to the inverse of their frequency, and so consider less optimal the rarest paths.

2.3 Basic data visualisation concepts

Since we consider the visual output of this thesis fundamental to understand and appreciate the concepts theorised, we dedicate this section to describe the principles behind the data visualisation discipline.

Data visualisation is not an exact science: many studies have been proposed in order to discover the exact mechanisms behind human visual perception and information encoding, and the result was not a set of dogmas but rather some principles and good practices.

The artisanality of this discipline should not be considered a defect but rather an opportunity to mould the information in such a way that it is understandable by the target we are referring to, and so that the visual output immediately communicates the various shades of the data we are studying. From our point of view, data visualisation is fundamental to make the scientific knowledge accessible and to help the divulgation outside the research field. Intuitive tools can help to build trust in the technology.

In the following paragraphs, we will review the most important theoretical contributions behind data visualisation, so that the reader can better understand the visual choices we did while designing the visual output of this thesis.

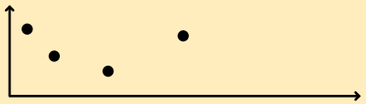
Jacques Bertin is considered one of the fathers of data visualisation thanks to his work Sémiologie Graphique (1967)[19], where he rationalised the elements used to represent statistical information mainly in cartographic applications. Bertin defined the so-called visual variables and described how each element of a representation is more or less suitable to convey a certain type of information. These variables are: the position of the element in space, the shape, the size, the value, the colour, the orientation and the texture. Their combination constitutes a complex visual code able to transmit multiple information in a relatively small amount of space. For position is intended the coordinates of an element in the page, and for size its length or area; the shape is the form given to the element, and could be quite variable; for value we intend its darkness while the colour refers to the hue; finally, the orientation is the amount of change in the alignment of the element, while the texture is the variation in grain of its fill. The core point of Bertin is that each of these variables can be exploited to represent numerical values, so for example a longer bar in a chart corresponds to a greater value.

Another important figure in the data visualisation field is Edward Tufte. In his work The Visual Display of Quantitative Information (1983)[20] Tufte defines the *data-to-ink ratio*, arguing that a good representation minimises the amount of ink used to show a particular data. The proportion is computed by comparing the total amount of ink used to print the graph and the quantity actually useful to represent data, so the objective, according to Tufte, is to remove all kinds of redundancy and of non-data elements. This is of course a radical approach to information visualisation and there is no evidence that minimalist plots are more understandable nor that are preferred[21]. For example, Bateman et. al. (2010)[22] have pointed out that a certain amount of decoration and of redundancy helps to memorise the information in a chart, and the field of data journalism often takes advantage of the combination between illustration and diagrams[23].

What is important about Tufte's philosophy is the will to convey the information in the cleanest way and by consequence to be as

Bertin's visual variables

Position
x, y location



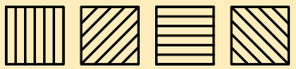
Size
length, area, width or height

Shape
there are infinite shapes

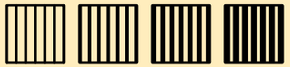
Value
light to dark

Colour
hue

Orientation
the aligment / rotation



Texture
the grain

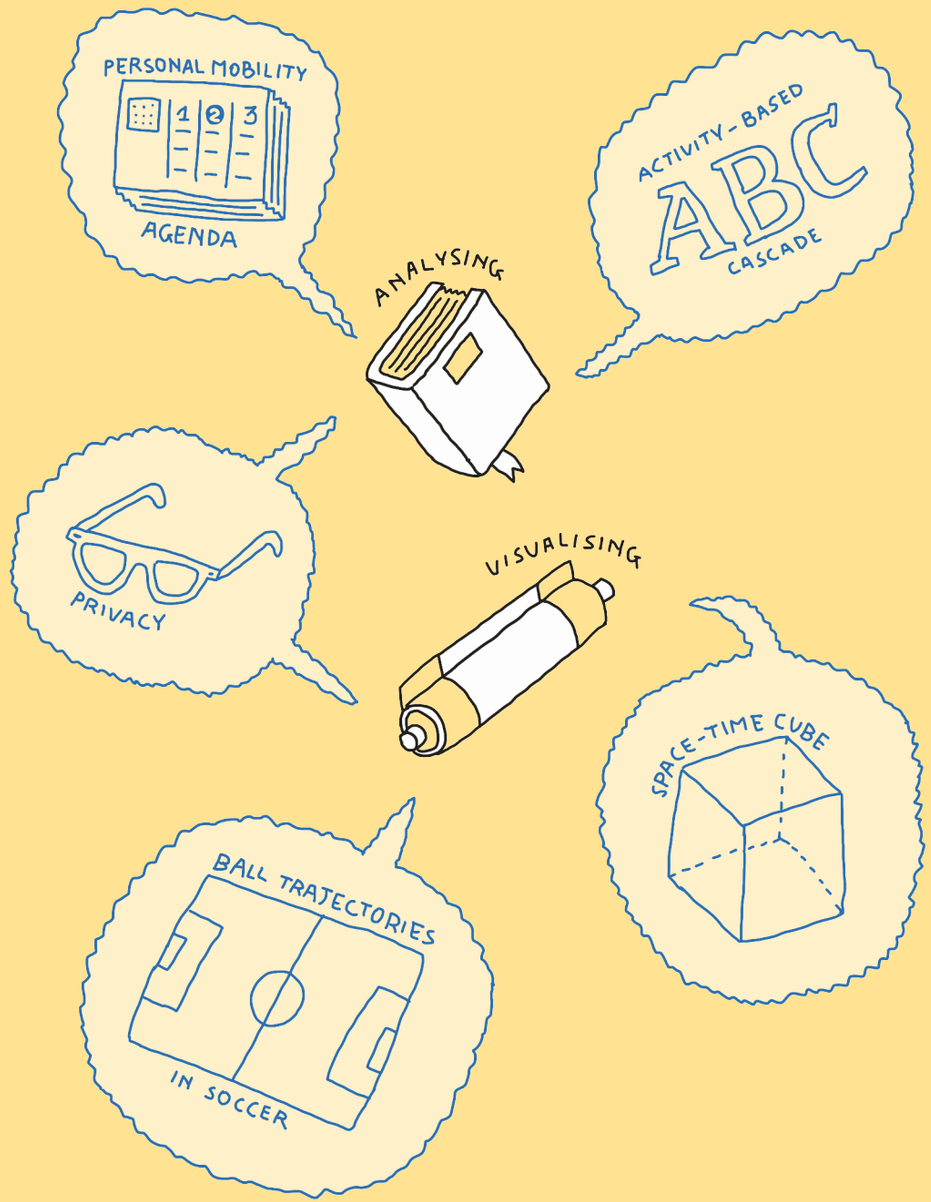


scientific as possible. Nowadays, we are conscious that biases also emerge before the data representation, in the way we classify the records and we organise them. Also, a small level of redundancy in the visualisation could help to read it more easily: as humans, we do not perceive in a mechanical way, so the repetition of the same concept in different ways makes us faster in understanding a concept. For a complete review about the subjectiveness of scientific images see Objectivity, by Daston and Galison (2007)[24]. However, we recognise to Tufte the will to not interfere in the representation with useless elements, that could become misleading.

Ceneda et al. (2020)[25] well explain how difficult and artisanal the process of designing a visual analytics tool is, pointing out that *guidance* (the process of easing the information extraction from complex data) is context dependent. Also, they see visual analytics as a method to fill knowledge gaps, that, they discuss, can arise from many reasons. Another important point stated by the authors is that data visualisation is almost the last step of the designing process: first, we need to understand the analysis goals, figure out what kinds of knowledge gaps there could occur, and then iteratively prototype and receive feedback from the users. This means that one solution could work in a specific case but could be unhelpful in similar situations.

Fig. 2 (left): Bertin's visual variables.

3. Related work



In this section we review the literature about our main areas of interest, that is mobility data representation and analysis, and mobility data visualisation.

3.1 Mobility data shaping and analysis

Rinzivillo et al. (2014)[26] have studied a cascading classification approach to detect salient activities from individual mobility networks. The main idea of this work is to extract semantic information from the Individual Mobility Network of the user starting from an annotated dataset of GPS tracked activities through a procedure called Activity-Based Cascade (ABC). The ABC consists in splitting the multi-class classification into a set of binary classifiers organised in a random forest which progressively determine which type of activity a set of data represents. Beside being interesting to perform supervised learning on mobility data, we found useful to see how the feature selection process has been resolved: the authors, in fact, explain they considered both the so-called trip features and the *network features*; the former are relative to the temporal and topological characteristics of a travel, while the other basically inherited from the network analysis field.

Guidotti et al. (2017)[27] propose to represent personal mobility through an agenda that summarises the movement habits of the user during a cyclic period, like a week. This is an interesting approach to predict the user's location on a specific day at a given time.

The personal mobility agenda is in fact defined on a certain time granularity (τ) and respect to a specific observed time interval, and it consists in a unique trajectory for the whole day, with a location at each *clock tick* τ. Obviously, it is not always possible to get the user's location at the a τ: some kind of interpolation is needed.

To reproduce the personal mobility agenda, the authors extract the stops using the algorithm TOSCA[28], the locations, and the movements of the user, and from them create a mobility network as a directed graph, where the nodes correspond to the locations and the links represent the movements. Then, the regular locations are extracted, and by consequence a cleaner movement dataset is obtained and an initial model can be obtained. The paper presents a procedure named RAMA (Routinary Actions Mobility Agenda) to extract the agenda using that model: for convention, the agenda starts at midnight, and the initial location is the most likely at that time given the observed data. Then, for each interval τ, we need to add a simulated position checking if the user is generally moving at that time. If this is the case, the last stop is considered as the current location, otherwise we need to interpolate it with some kind of function.
The interesting aspect of this work is the aim to create a discrete and understandable model, simplifying the structure of the mobility network and adding the temporal dimension.

The last work we want to report is from Zhao et al. (2020)[29] and is about privacy preservation of trajectory data. The authors use differential privacy combined with data arranged into prefix trees to create representations that preserve the spatio-temporal characteristics of the trajectories, simplify the mobility data structure, and hide sensitive data through the usage of laplacian noise. This paper was crucial for this thesis to figure out how to arrange trajectories in a simple and standard architecture.

3.2 Visual analytics for mobility data

In this section we present some examples of visual analytics applications that we find relevant for various reasons.

For the VAST 2013 conference, Andrienko et al.[30] propose a visual analytics methodology for analysing movement behaviours of groups of users. The resulting interactive tool is extremely complete in terms of analysis possibilities but at the cost of being quite complex. The first concept introduced in their work is the mapping of the movements not only on a geographical basis (in two dimensions) but also in a *space-time cube*, exploiting the third dimension to represent movements during time. The next step is to

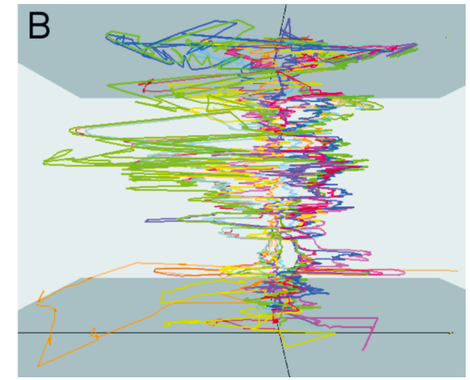


Fig. 3: Some user's trajectories in the space-time cube. Andrienko et. al. (2013)

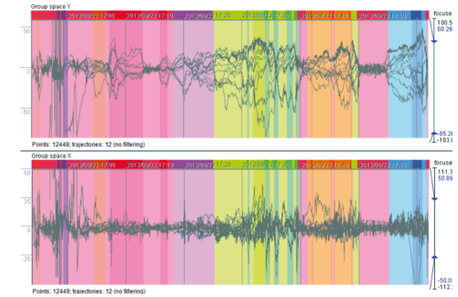


Fig. 4: the trajectories represented as variation in the movement's direction. Andrienko et. al. (2013)

transform the first two dimensions into a more abstract space which encodes the information about the direction of the movement, rather than the exact geographical coordinates. The resulting projection allows the analyst to study individual trajectories or groups of similar trajectories. The tool also provides for each user the information about how it fits in the group they have been assigned, by showing the density of their positions with respect to the group, or also the participation of the same user to different clusters. This tool is an example of how many different kinds of analysis perspectives are available starting from the same dataset, and that geospatial representation is not the only solution. The main problem with the tool is the amount of variables it has to handle, and the consequent complexity of the graphical user interface: the visual outputs often need an extensive explanation and could be hard to use outside the research field, for example for communicating the results of a study to the institutions or to a larger public.

Another work from Andrienko et al. (2021)[31] we selected is a very abstract representation aiming to hide the most sensitive

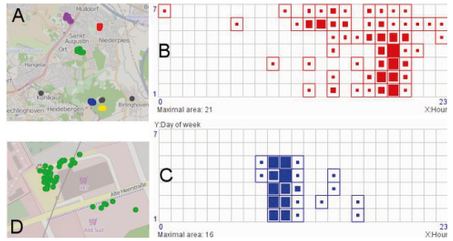
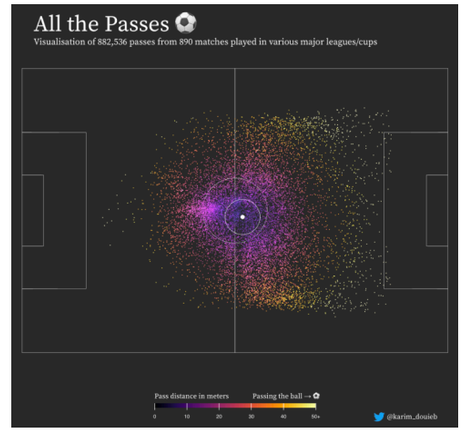
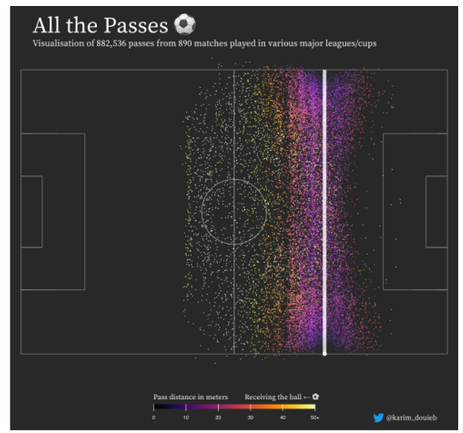


Fig. 5: Frequently visited places of a user plotted on a temporal map, where the rows correspond to the hours of a day, while the columns to the days in a week.
Andrienko et. al. (2021)

information which could be extracted from mobility data. In fact, as they discuss: [*it is easy to identify a*] *person's home and workplaces and other frequently visited places by interpreting spatial and temporal patterns of the person's moves and stops from the positions of the human common sense*. The authors propose a couple of two-dimensional histograms, divided in cells, one for the detected home-location and the other for the secondary location (for example, the workplace), whose rows correspond to the hours in a day and the columns represent the seven days of a week; the more a cell is filled, the higher the probability for the user to be in the location in that time. The simplicity and intuitivity of the visualisation and its independence from the geographical space are the strength of the output. Its only issue is the difficulty of comparison of the probabilities, since their value is represented through the size of a square area, that is not easily readable for humans, and since each square is centred in the cell, making even more difficult to discern the quantitative difference. However, this visualisation is an excellent qualitative starting point for more in-depth diagrams, and it is easily explainable to the general public.

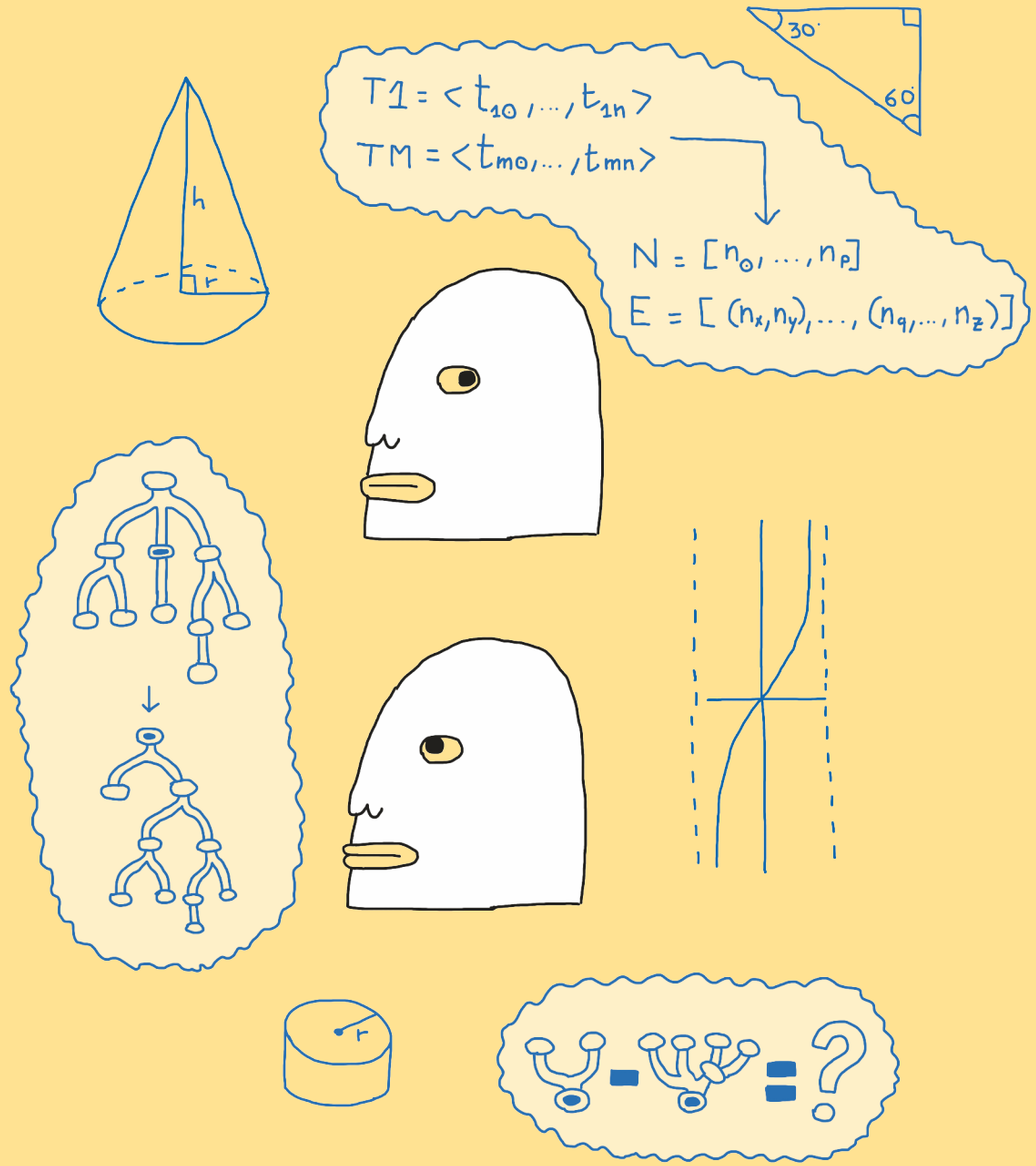




Figg. 6 (top), 7 (bottom): All the ball passes arriving and starting from a [x, y] point or from a specific x value in the field.
Douieb (2021)

A totally different paradigm is proposed by Douieb (2021)[32] to represent a different kind of mobility data: football passes in the play area. In this case, the focus is completely on the locations, while who hit or received the ball remains unspecified. The result is a flow visualisation which allows the user to focus on a specific coordinate or on a specific axis, or to simply look at all the passes simultaneously, and to visually discover the game patterns. A simple interface allows customising the visualisation's variables; also in this case the output is highly qualitative, but at the same time rapidly understandable. Possible implementations could be to use a slider to visualise the progress of the action, rather than an animation, and maybe to provide some statistics, like for example the median distance of the incoming or outcoming balls; in addition, the interface could be improved to filter data, rather than only to change visual variables.

4. Problem formulation



In this section, we describe how we can transform the set of trajectories of the user u $M_u = <t_o, ..., t_n>$ into a tree $T = \{N, E\}$, where $N = <n_o, ..., n_p>$ is the set of nodes corresponding to the traversed locations, and $E = <e_o, ..., e_m>$ is the set of edges connecting two nodes, so that there is no cycle. We also explore how to convert the personal data M_u into a unique vector V and to transform a tree into a vector $V(T)$ describing its architecture.

4.1 Representing mobility data through trees

In this section, we present the two data structures we experimented with: prefix trees and spanning trees.

4.1.1 Prefix trees

Mapping trajectories into a prefix tree is a quite straight-forward process, since we treat each point as the letter of a string. Since we are considering similar trajectories beyond their temporal appearance, we can simplify each point of a trajectory to contain only spatial data; in addition, to better represent the same geographical places in the structure, we assign to each tuple $g_i=(lon_i, lat_i)$ a unique identifier $j \in J$ (a collection of geographical points). So, the trajectory points previously described as $p_i = (lon_i, lat_i, t_i)$ are now simplified as $p_i = j_i$. The resulting tree will be composed by the nodes $N = <n_o, ..., n_n>$, the labels $L = <l_1, ..., l_n>$ assigned to each node, and the edges $E = <e_o, ..., e_m>$. The nodes are artificial structures associated with a unique identifier, and contain the information of the represented location in their label; the same label value can be present in different nodes, in various positions in the tree. For clarity, we expose through an example the creation process starting from a simple set of data. Let D be the database of trajectories of a user U, and be it equal to $D_u = \{ t_o: < A, B, C >, t_1: <A, B, D >, t_2: < A, E, F >, t_3: < B, E, G>, t_4: < B, A, G >, t_5 : < C, A, B >\}$. The first step is to initialise the virtual root of the tree as an empty node. The siblings of root will be A, B and C, since these are the starting locations of all the trajectories. The node $root \rightarrow A$ will have B and E as siblings, because these points can be reached after having traversed the virtual root and A; the siblings of B, instead, will be E and A, while C will only have A as successor.

Note that in the siblings of $root \rightarrow A$ does not appear G because this point is not directly reachable from A as a starting trajectory location.

We have obtained a compact yet powerful representation of the user's movements. One of the aspects to be taken into account is how the location information is decentralised among the tree: many nodes correspond to the same spot and each of them provides insights about it according to a specific pattern of movement.

4.1.2 Spanning tree

To map a trajectory dataset into a spanning graph, we simply create a node for each unique location L observed, because the same location cannot occur more than once. Then, using the Kruskal algorithm (but there are many others available), we determine the optimal edges according to the user's movement graph so that each point is connected with the others, at least indirectly. Since we are interested in the most frequent paths, we use as a weighting function the inverse of their frequency, since light weights are prioritised.

The resulting representation synthesises the user's mobility focusing on the connections between each location. Contrary to the prefix tree, in this structure there is a one-to-one correspondence between nodes and locations, giving place to a more condensed representation. At the same time, if in the prefix tree the chronological order of the locations in the trajectories is respected, in the case of the spanning tree we have an important summarisation of the paths, because we only keep the links with the best fitness. As we will repeat in other circumstances, we do not consider this feature an issue, since the focus of this work is on the shape of a person's mobility, rather than only on the sequences of places they visit.

4.1.3 Two perspectives on the same phenomenon

We saw two different approaches to represent movements through trees; in the next lines we describe the different details they highlight. A prefix tree is more focused on the ordered sequence of points traversed: from its root, in fact, are generated the nodes corresponding to the locations where, according to the dataset, the user starts their trips. Following a branch, all the nodes with more than one successor represent possible practicable routes given the previously visited locations.

A spanning tree is instead a more abstract representation of the user's movements: its branches, in fact, do not necessarily correspond to real trips but are a synthesis of the user's habits. This architecture tries to organise hierarchically the mobility so that more traversed routes are central, while the other have a more accessory role.

4.2 Representing mobility data through vectors

The second path we explored was to represent the mobility data as vectors, because such a representation is suitable for a large number of analysis methods. We tried to preserve the same principles we adopted when creating mobility trees, so the will to study the shape of the mobility, and to be as indirectly related as possible to the geographical coordinates information.

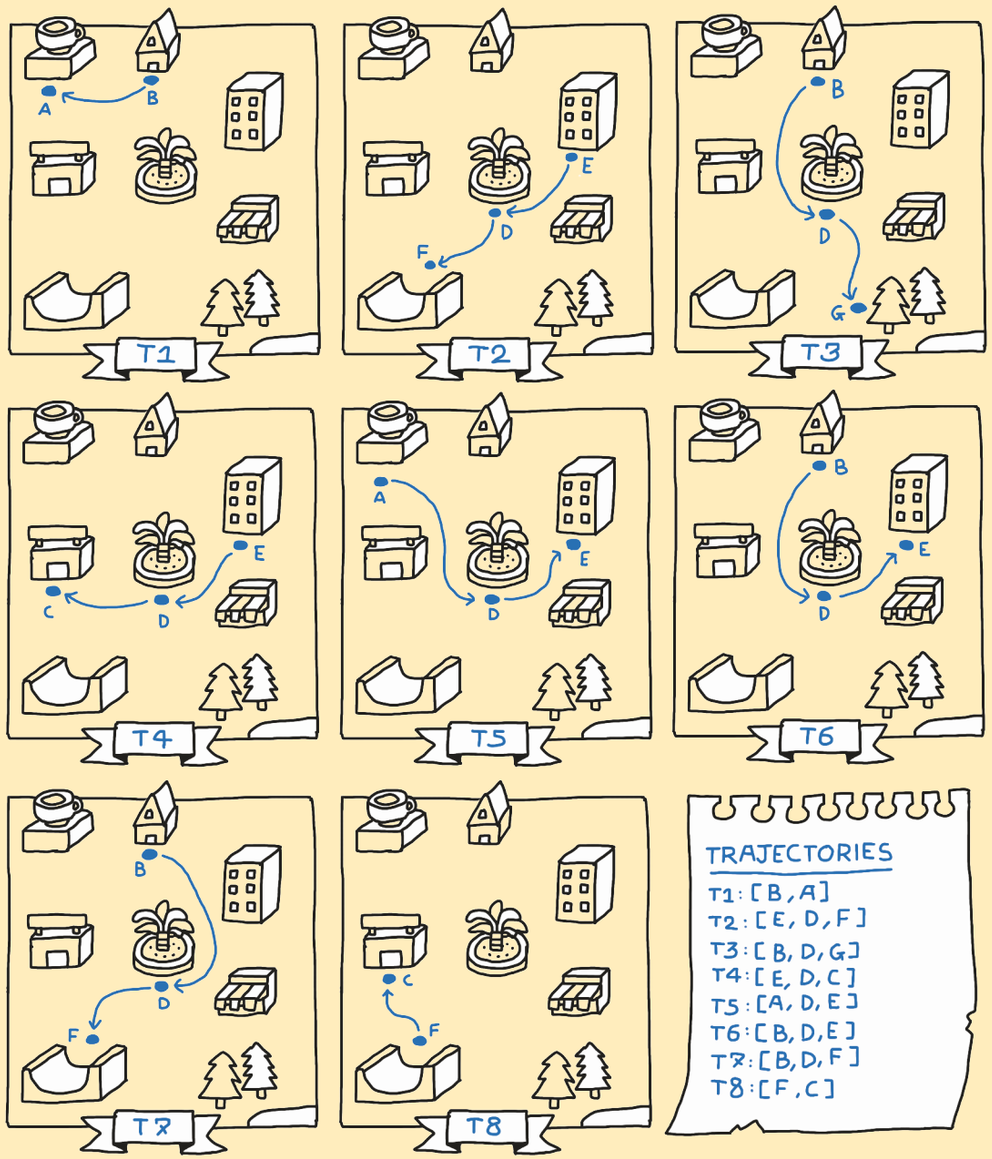
The theoretical basis for building mobility vectorial representation has been the concept of TF-IDF. The TF-IDF (Term Frequency - Inverse Document Frequency) is a measure primarily used in information retrieval and it consists in the ratio between the probability of one term in a document (TF) and the logarithm of the total number of documents divided by the number of documents where the term appears (IDF). By analogy, in our case the locations correspond to the words, while the users are intended as the documents.

The TF-IDF is useful to understand which locations characterise one person's mobility respect to the others, because location common for many users will get a low value.

Fig. 8 (next two pages): How to build a mobility prefix tree from trajectory data.

MOBILITY PREFIX TREE

THE DATASET



TRAJECTORIES

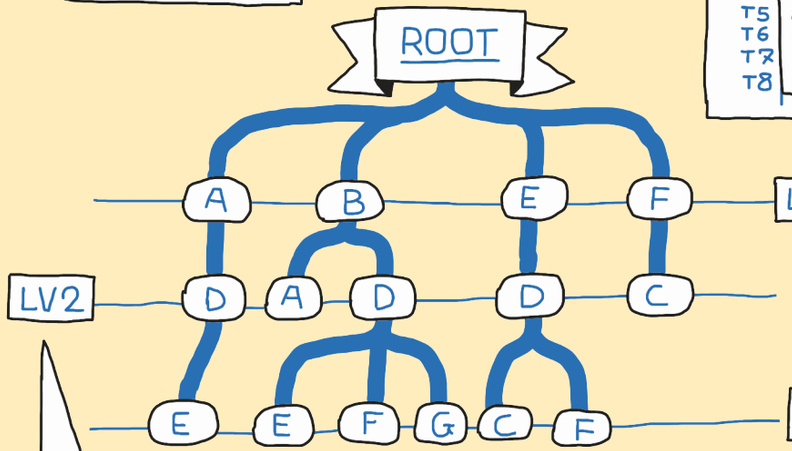
T1: [B, A]
T2: [E, D, F]
T3: [B, D, G]
T4: [E, D, C]
T5: [A, D, E]
T6: [B, D, E]
T7: [B, D, F]
T8: [F, C]

BUILDING THE TREE

SINCE THERE COULD EXIST MANY POSSIBLE STARTING LOCATIONS, WE CREATE A VIRTUAL NODE AS THE ROOT OF THE PREFIX TREE

ACCORDING TO THE DATASET, THE USER'S STARTING LOCATIONS ARE: A, B, E, AND F

	0	1	2
T1	B	A	
T2	E	D	F
T3	B	D	G
T4	E	D	C
T5	A	D	E
T6	B	D	E
T7	B	D	F
T8	F	C	



ACCORDING TO THE STARTING LOCATIONS WE DERIVE THE FOLLOWING ONES

A	0	1	2
T5	A	D	E

B	0	1	2
T1	B	A	
T3	B	D	G
T6	B	D	E
T7	B	D	F

E	0	1	2
T2	E	D	F
T4	E	D	C

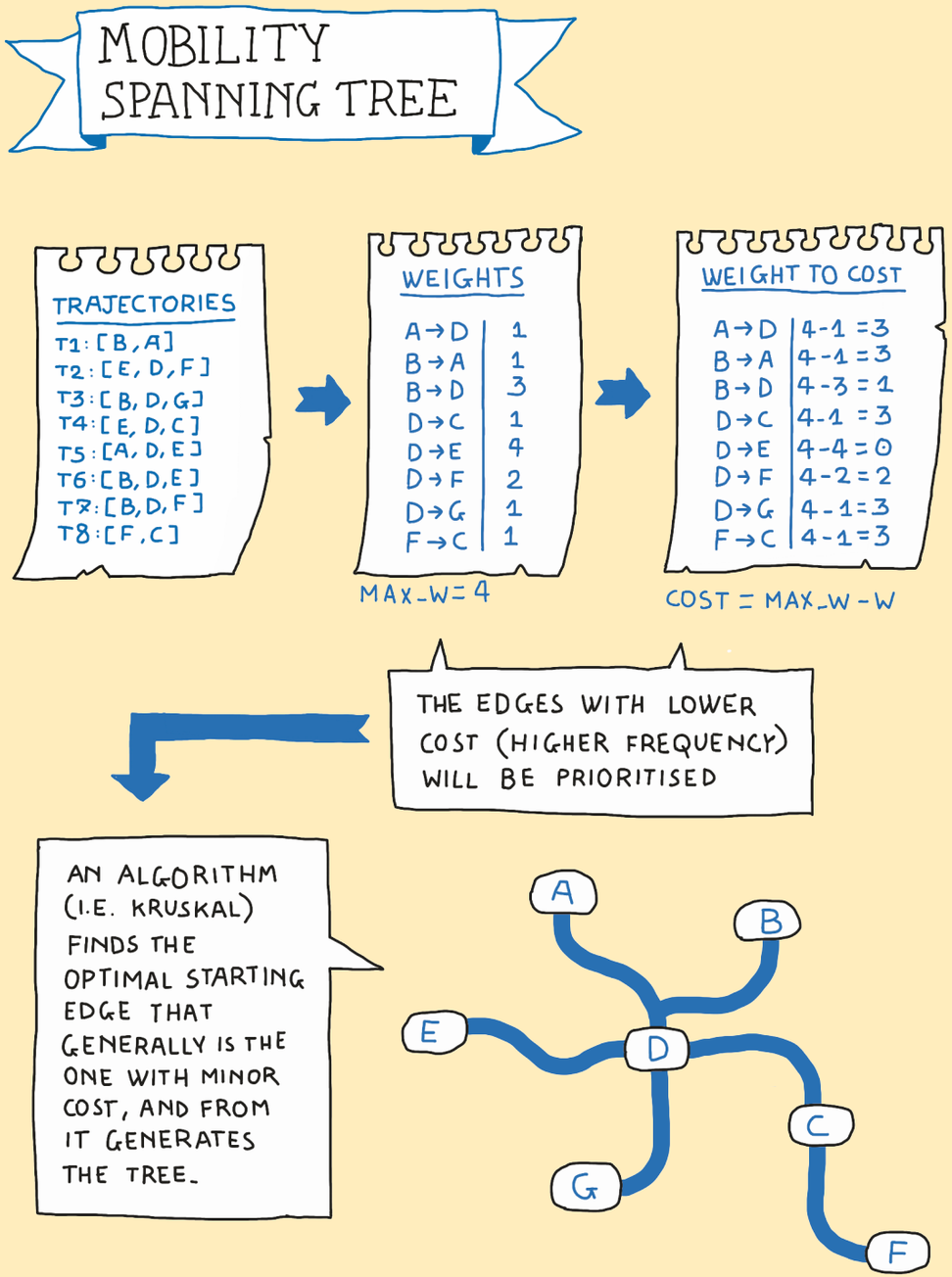
F	0	1	2
T8	F	C	

WE PROCEED FOLLOWING THE SAME LOGIC

A	0	1	2
T5	A	D	E

B	0	1	2
T1	B	A	
T3	B	D	G
T6	B	D	E
T7	B	D	F

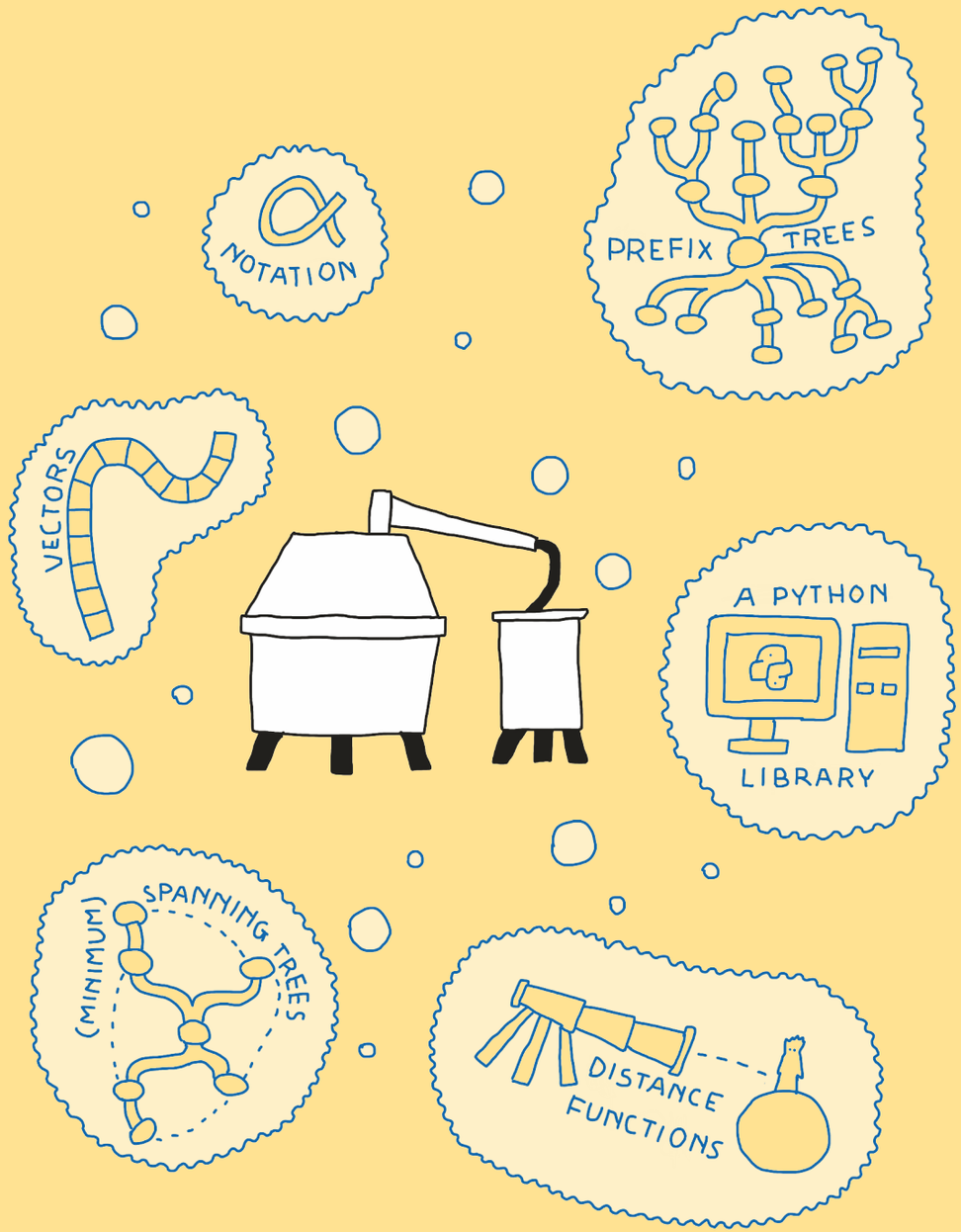
E	0	1	2
T2	E	D	F
T4	E	D	C



Another way to represent mobility data through vectors we hypothesised was to extract some features describing the mobility trees and use them as features for unsupervised learning. In the Methodology section both methods are explained.

Fig. 9 (left): How to build a mobility spanning tree from trajectory data.

5. Methodology



In this section, we present the theorical and technical solutions we ideated to transform the already mentioned trees and vectors into really semantically valuable representations. In particular, the main question we have tried to answer has been: what will be the role of the root of a tree, and which kind of relations are we trying to encode both in trees and vectors?

5.1 Trajectory notation

In order to keep the data comprehensible when organised in prefix trees, we added two artificial points to each trajectory, an apex '^' at the beginning and a dollar '$' as the final point. This notation allows us to keep track of the direction of the trajectory when, as we will explain later, the list of traversed points is bent on a location fulcrum. This transformation is not necessary for spanning trees and for vector representations.

5.2 Prefix trees

The first reflection we made about organising trajectories in a tree was that such a structure is perfect to focus the analysis around a specific location, set as the root. This implies the choice of a selection strategy for the designed location, that, in our case, was the most frequent in the user's trajectory dataset but many other valid alternatives are possible depending on the analysis' purpose.
Once the root location has been determined, the canonical prefix tree should be rotated around it. However, this is not straightforward, since the same location ID can occur in many nodes of the tree; also, it is more expensive in terms of computation to rotate an existing tree instead of building it from scratch.

Since in a prefix tree a root location can occur in different nodes, there are no determined strategies to perform the rotation. In the preliminary phase, we designed three kinds of rotations, we evaluated the possibilities each of them led, and opted for one over them.

The first strategy was the most straightforward: we select one node among all the ones whose label corresponded to the root location, also defining a selection criteria; this becomes the root and the tree is simply hanging on it. We discarded this option quite early because the representation does not add so many advantages: first of all, the root location is still repeated in multiple nodes among the tree; then, we need to preserve the original virtual root empty node somewhere in the structure, meaning some branches can contain edges with different directions.

The second strategy was designed to reduce the root location to one single occurrence. In order to do that, we ideated a tree where under the virtual root with no location assigned were connected siblings: the first corresponds to the root location (on the left); the second is an empty node, named ^, which serves as virtual root for all the trajectories not containing the root location. By consequence, the new rotated prefix tree contained two internal prefix trees, one for the trajectories touching the root location and one for the others, that we named orphan paths. The advantages of this representation are the centralisation of the root location, the uniformity of the branches' edges directions and the well-defined separation of the trajectories passing from the root from the ones not traversing it. The major issue, instead, is that there is no distinction from incoming and outcoming paths when observing the root, and that it is not possible to reconstruct the original trajectory: in fact, there is no connection between the incoming path of a trajectory and the outcoming path. However, this representation can be used to observe in a global way the movements around the root, shifting the focus from the whole trajectory to patterns related to an important location.

The last architecture we designed was meant to have all paths sharing the same direction. As in the previous one, under the virtual root lie the root location node and the ^ empty node. In this case, however, the paths connected to the root location are only the ones that are exiting from it; the incoming paths are under ^, and their terminal node is labelled as the root location. Even if this architecture is the most formally uniform, since all the edges follow a chronological order, the information representation is sparse, since we have multiple occurrences of the root location; nevertheless, the structure is cleaner than the first option. We define this architecture as a meeting point between the two structures described before.

The first architecture has been discarded soon because of the variability of its structure, which could impede a comparison between users. We then chose for the second, because thanks to the unique appearance of the root location it enabled us to imagine different kinds of comparison: for the third architecture, in fact, it seemed more difficult and expensive to keep track of all the occurrences of the root location in the leaves under ^.

After some experiments we decided to simplify the structure described above not only to reduce the computation complexity but also to make more organic the analysis: in fact, if a location was decided to be the fulcrum of the user's mobility, trajectories not traversing it could augment the noise and appear distractive. Of course, an interesting research path could be to relate the orphans' branch with the main one, but it is out of the objectives of this work.

The final structure of the rotated prefix tree is then originated by a virtual root that gives origin to a single node: the location root, from which all the trajectories are developed.

Note that in this structure we can observe a partial loss of information, since there is no way to know how an entering path continues, unless we encode this feature in some way. We decided this issue could not be too problematic, since we were interested in understanding the general shape of the user's mobility, and we did not intend to observe the single trajectories one by one.

5.3 Spanning trees

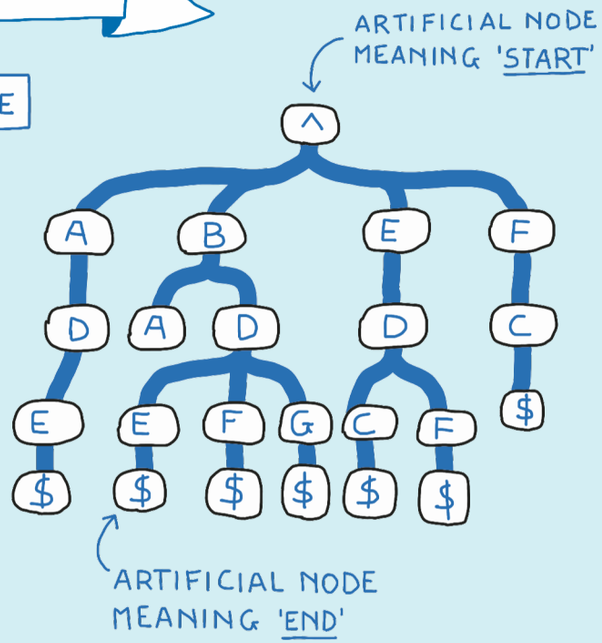
As we did for the prefix trees, we decided to transform the spanning graphs into rooted trees. At a first glance, this could seem an incoherent idea, because a tree is a directed graph, while a spanning graph is undirected.

Fig 10 (next pages): Three strategies to rotate a mobility prefix tree

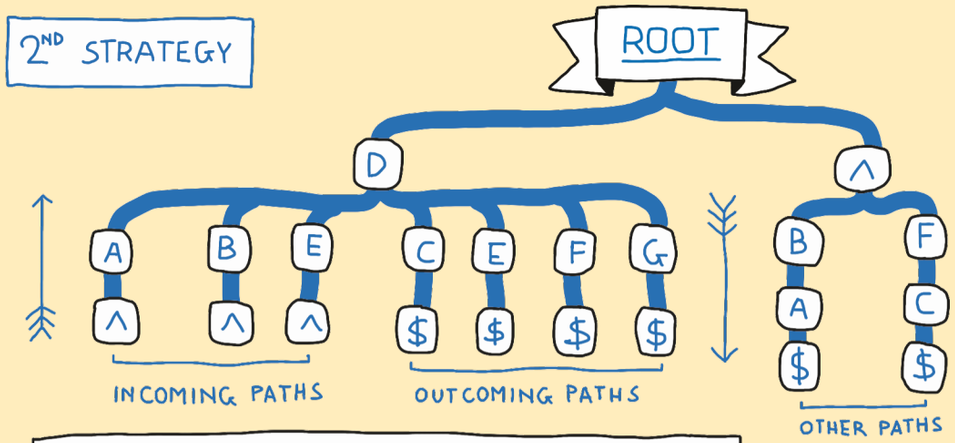
ROTATING A PREFIX TREE

THE ORIGINAL TREE

TRAJECTORIES
T1: [B, A]
T2: [E, D, F]
T3: [B, D, G]
T4: [E, D, C]
T5: [A, D, E]
T6: [B, D, E]
T7: [B, D, F]
T8: [F, C]



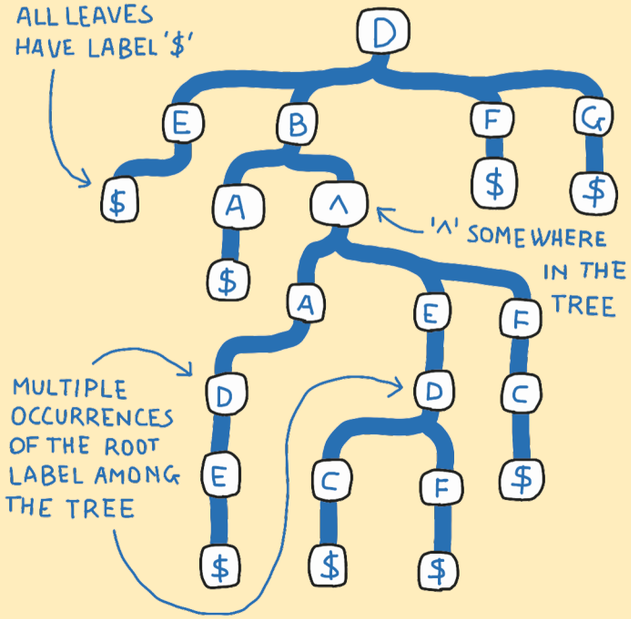
2ND STRATEGY



UNDER 'ROOT' WE ALWAYS HAVE 2 SIBLINGS: THE MOST FREQUENT LOCATION AND '^'.
UNDER THE FORMER WE FIND ALL THE TRAJECTORIES CONTAINING THAT LOCATION, DIVIDED IN 2 SUBPATHS → EACH, CORRESPONDING TO THE INCOMING & THE OUTCOMING TRIP.
UNDER '^' THERE ARE ALL THE TRAJECTORIES NOT CONTAINING THE MOST FREQUENT LOCATION.

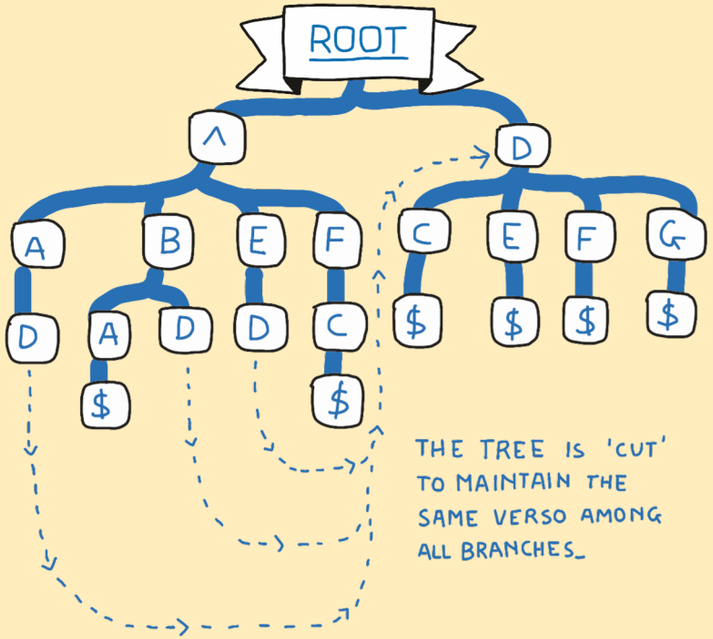
1ST STRATEGY

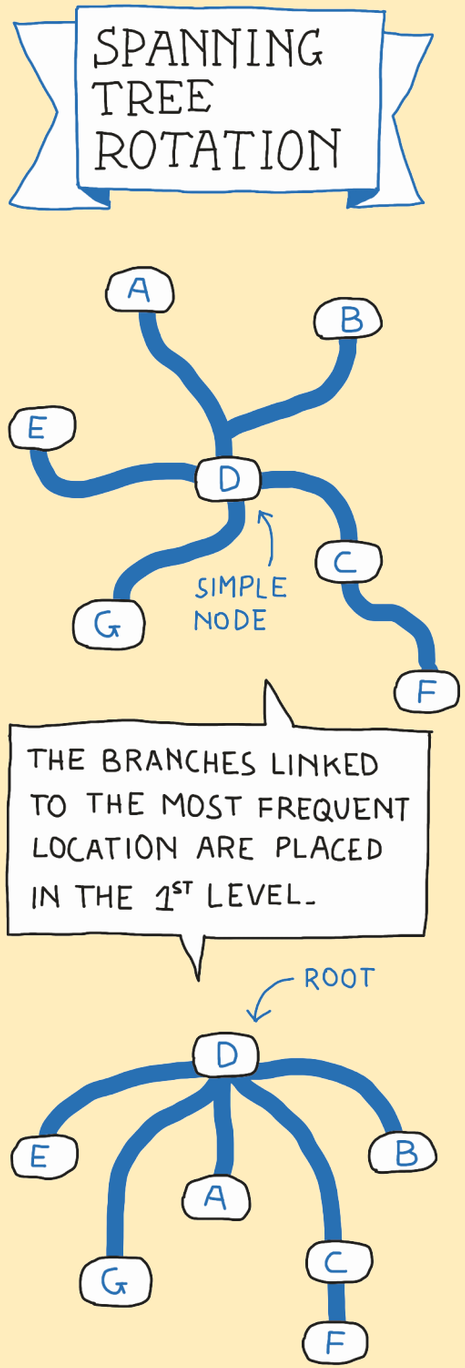
ONE NODE LABELLED WITH THE ID OF THE MOST FREQUENT LOCATION IS SELECTED AS ROOT.
IT NEEDS A CRITERION OF SELECTION OF THE ROOT NODE AMONG ALL THE ONES WITH THE SAME LABEL.



3RD STRATEGY

ALL THE BRANCHES HAVE AT THEIR TOP THE LEAST RECENT TRAVERSED POINTS OF A TRAJECTORY.
BRANCHES NOT ENDING WITH '$' CONTINUE IN THE SUBTREE UNDER THE MOST FREQUENT LOCATION.





We decided to perform this transformation because we are interested in the shape of the movements around a location, instead of on the effective links between places. Transforming the spanning graph into a rooted tree gives us another representation of a user's mobility, more focused on the locations instead of on the sequences of visited points, as in the prefix trees.

5.4 Vectors

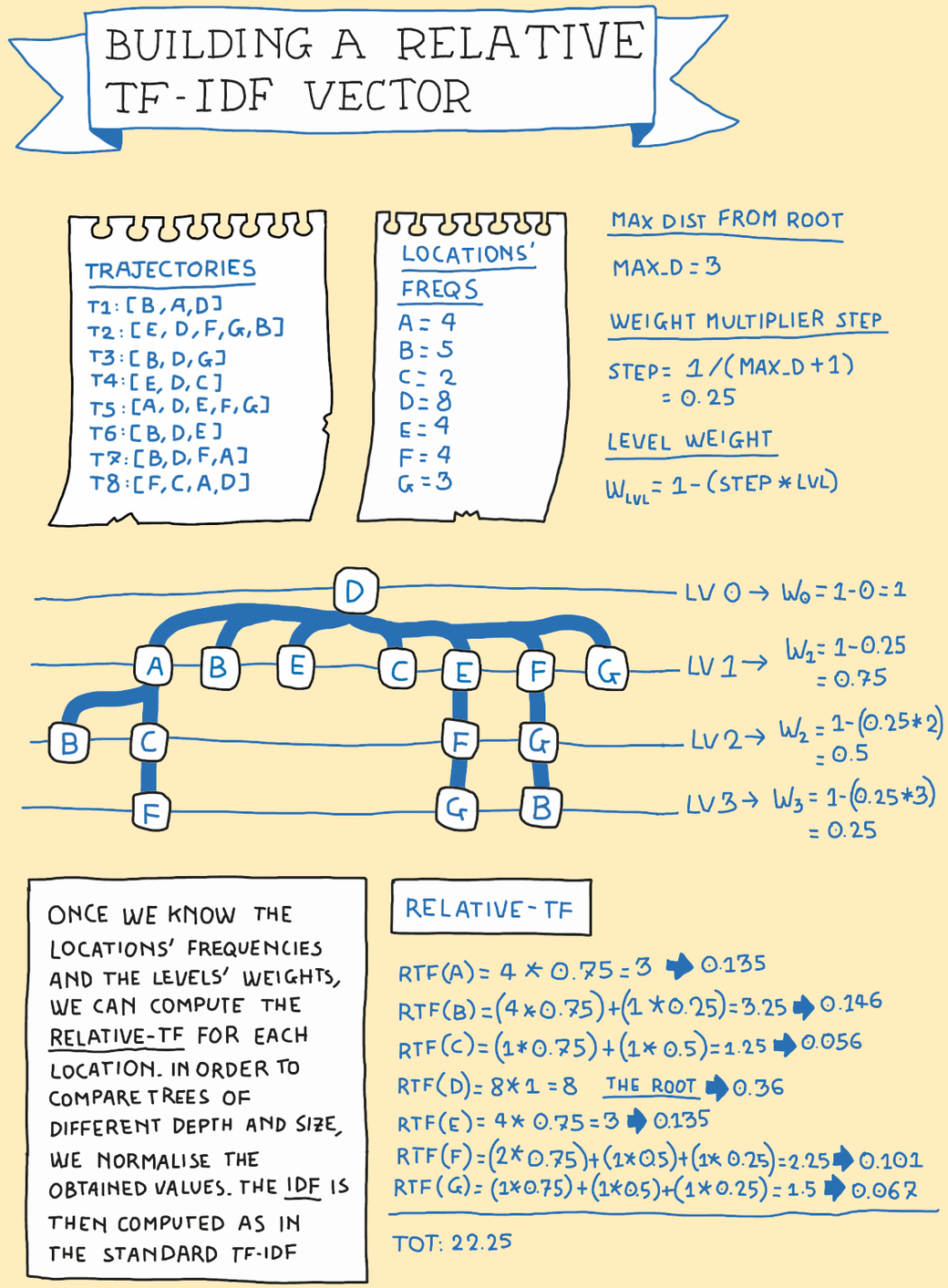
5.4.1 TF-IDF inspired method

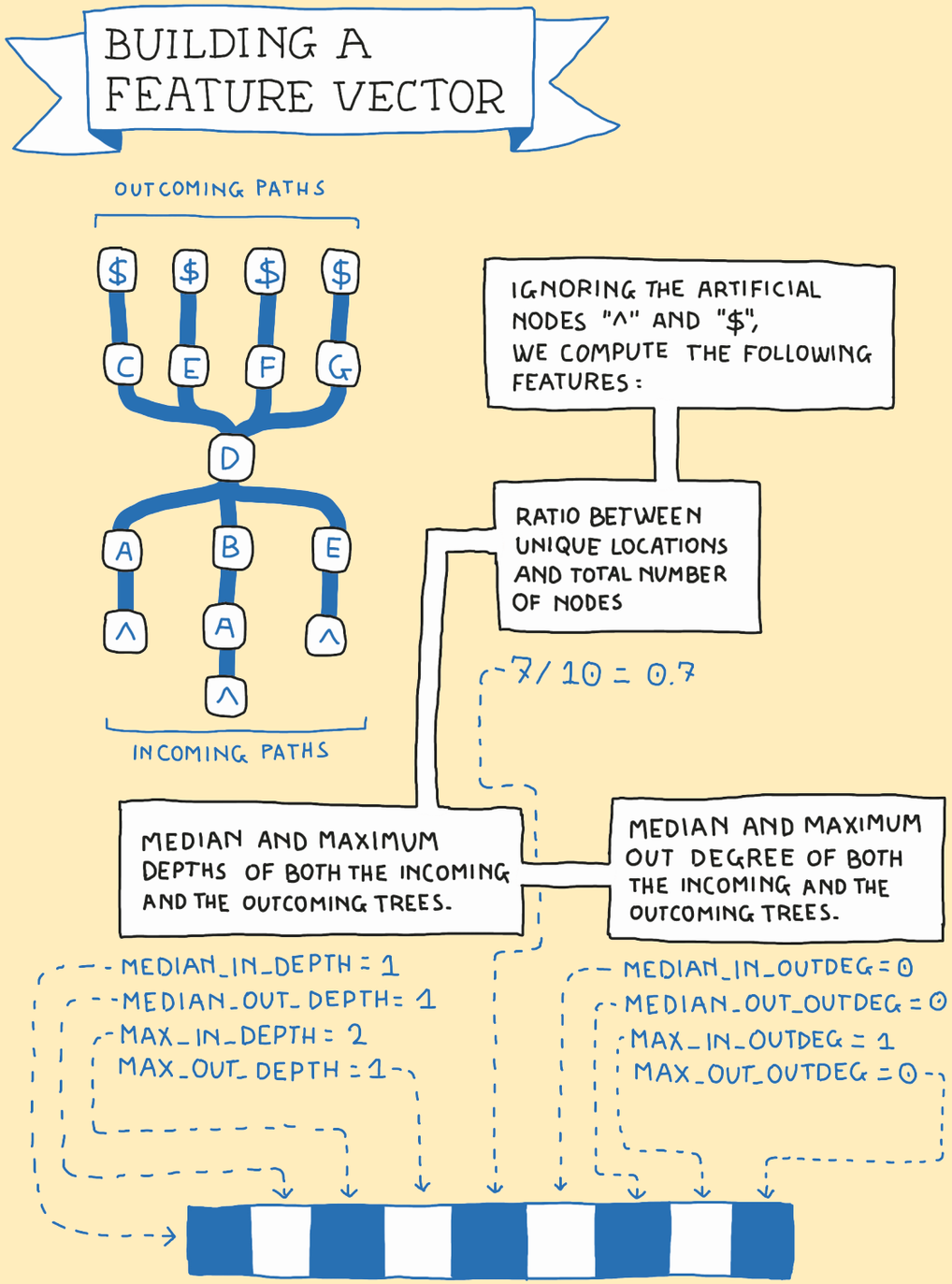
Due to the lack of literature about clustering techniques between trees (instead of among the same tree), we also ideated a backup technique to represent mobility through vectors. Of course, changing the datatype, the approach has been different, to better exploit the characteristics of the different data structure.

We started from the idea to create a vector for each user, where each position corresponds to the TF-IDF value of each possible location in the dataset, aware that a feature selection step would be necessary due to the highly probable sparseness of the representation.

We imagined a way to combine this metric with the general philosophy we followed while creating the prefix trees, that is to give importance to the relation between the locations and a root, and we named it relative-TF-IDF. In this case, the TF does not only represent how frequently a location occurs, but it is also weighted according to its position with respect to the most frequent

Fig. 11 (left): How to rotate a mobility spanning tree.
Fig. 12 (right): How to create a relative-TF-IDF vector.





location. Concretely, we consider d, the depth of the tree, $w_{level} = -(depth_{level} * 1/d)$, the weight assigned to each level of distance (in points) from a location to the root, and then assign the weight to each location as $w(location) = i \forall\ 1*w(level(i))$, where each i is a node labelled as the considered location.

This representation could by consequence highlight which are the most central places in the user's mobility, adding some semantic to the standard TF-IDF measure.

5.4.2 Feature-driven method

The second way to transform movement data into vectors started from the mobility trees we described above. Essentially, the idea is to extract some features describing them, and organise these values into vectors. The selected features take inspiration from the user profiling field in text analytics, and in particular from the work of Dell'Orletta et. al. (2013)[33] who consider the dependency trees of the sentences.

The resulting vectors had a total of nine attributes: the median depth of the incoming tree, and the median depth of the outcoming tree; the maximum depth of the incoming tree, and the maximum depth of the outcoming tree; the ratio between the number of unique locations and the total number of nodes, considering the whole tree; the median out degree of the incoming tree, and the same of the outcoming; and finally the maximum out degree of the incoming tree, and of the outcoming tree.

Fig 13 (left): How to create a feature vector from a mobility prefix tree.

5.5 Distance functions

To compare the trees and the vectors generated from a mobility dataset including many users, i. e. to look for clusters of similar users, we need to define some criterion to determine how much similar or dissimilar they are. If for vectors many options are available (like euclidean, Manhattan, and Minkowsky), for trees the choice is less obvious. In this subsection, we describe the distance measures we considered.

5.5.1 Tree distance functions

Data analysis on tree-shaped structures cannot rely on the distance functions used for vectors. In this section, we take an overview of the literature about the measures capable of determining how similar or dissimilar are two trees.

The tree edit distance is a dynamic programming technique which calculates the distance between two trees as the minimum number of operations needed to convert one of the two into the other. The allowed edits are: add (insert a new node), delete (remove a node) and relabel (change the value of a node). In the most general case, each of these operations have a cost of one each, even if in some cases they are customised for the research's needs (i. e. to penalise certain configurations)[34]. Formally, the tree edit distance is defined as $\delta(T_1, T_2) = \min\{\gamma(M) \mid (M, T_1, T_2)\}$. Many algorithms are available to compute the optimal set of edit operations.

The major problem with this approach is that it is ordered from left to right. This means that two similar trajectories in the two trees could not be detected if their position with

respect to the first path on the left is not equal. For this reason we decided to create an ordering criterion to test this metric, which is explained in the following paragraph. At the same time, we also found an unordered variation of the tree edit distance which performs the comparisons of the branches in sets instead of sequences[34]; this version also has the advantage of being computed in polynomial time, beating the standard version that requires exponential time instead.

For what concerns the ordering mechanism of the trees, we decided to position the most heavy branches in terms of weight: by consequence, the prefixes with higher frequency could be found in the left side of the tree. Also, since this structure was extremely interesting for studying the mere structure of the tree, we decided to remove the locations' labels from the nodes: in this way, two trees were comparable only for the user's habits instead of for the visited locations.

5.6 Treemob: the Python module

All the mentioned structures are available in the Python module we created[35]. We took care of creating inline documentation in the code, so that it is reusable for future works. In the Appendix the reader can find the full description of all the methods and classes we implemented.

5.7 The visual representation of mobility trees

Data visualisation is fundamental for both the data exploration step and the communication of the research outcomes. For this reason, we studied a visual system[36] to represent and explore the mobility trees obtained through the mentioned methodologies.

We decided to implement it in the Javascript library D3.js, a standard tool for data visualisation. This choice allowed us to design interactive modules and to have control on all the aspects of the representation: in fact, starting from simple snippets in the Observable platform, we were able to create complex and customised visualisations.

The output is then a web page describing different aspects of the user's personal mobility tree. We organised each component in a hierarchical order, so that the most important (and more general) elements are suddenly accessible, while the more specific ones are visible in a second moment. This is coherent with the Shneiderman's Visualization Mantra[37].

All the graphical choices we made were driven by the aim to communicate complex information in the most accessible way. We adopted a rigorous colour system and we added only functional elements to the visualisation. To increase the understandability of each component, we also added a brief description explaining what it represents and how to read it.

Fig. 14: The mobility prefix tree in the classical shape, accompaied by the barchart representing the amount of terminal locations for each level.

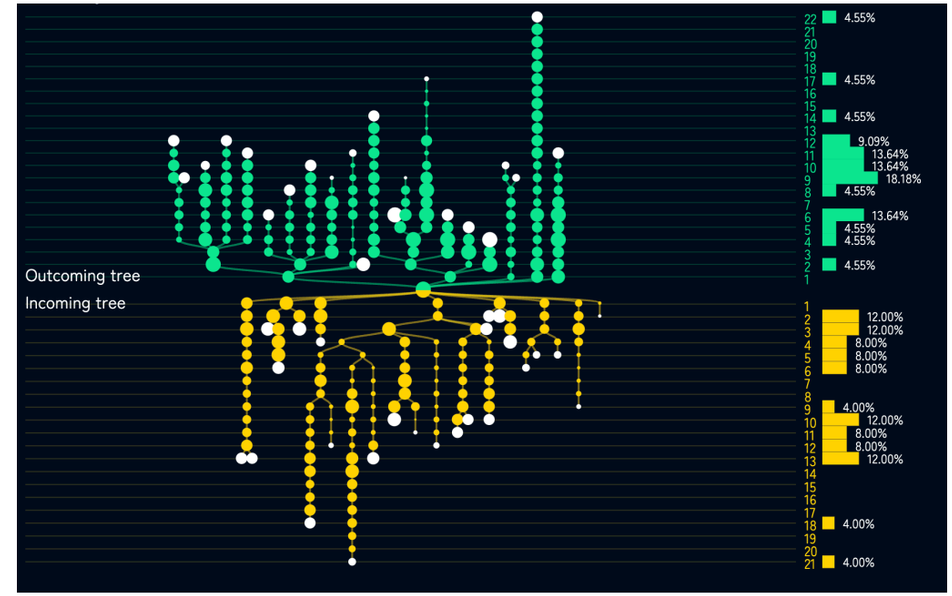
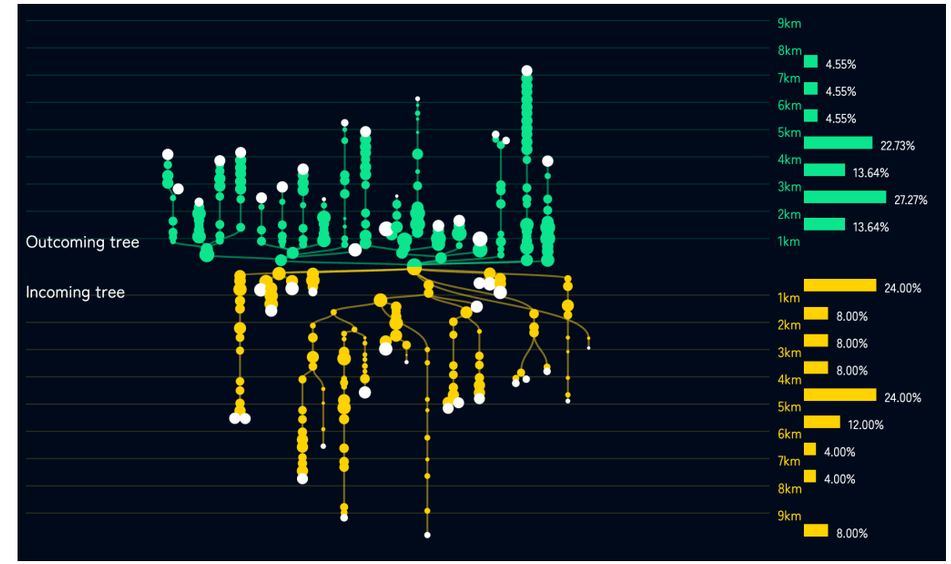


Fig. 15: The mobility prefix tree mapped on the distance travelled with respect to the root, accompained by the barchart representing the amount of leaves for each km from the root.



The first output we designed has been, of course, the user's mobility tree. From an information architecture point of view, we adopted the most feasible representation, that is, the node-link layout, also named the tree-shaped structure. This choice has been of course automatic, since the starting data was organised in the same way. We adopted the botanical analogy, placing at the bottom the incoming trajectories and at the top the outcoming ones. Being the most important component of the visualisation, the tree is the first element we can see in the page. So, each node corresponds to a location, and each link is a path from one place to another. To highlight the important locations, we scaled each node according to its frequency in the user's trajectory dataset; also, when a node is hovered, only the other nodes in the tree corresponding to the same location are highlighted, allowing to discover patterns in the user's habits.

The canonical tree representation is set in such a way that each node is positioned in a layer, according to how far in terms of steps is from the root. This means that each link, corresponding to the path from a location node to another, has the same length as all the others in the tree, regardless of the distance travelled. This could be quite limitant in a user experience point of view, because we are not able to see if there are longer or shorter paths. For this reason, we also proposed a variation of this representation where the length of each link is proportional to the distance in kilometres between the two locations linked by the edge.

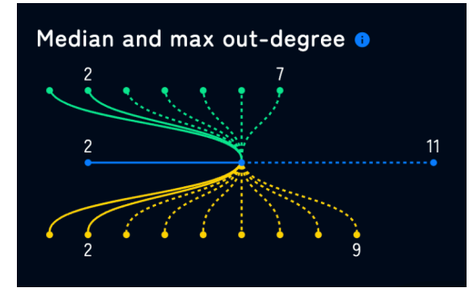
At the right of the tree component we placed a leaf exploration tool which shows what percentage of leaves appear at a certain depth.

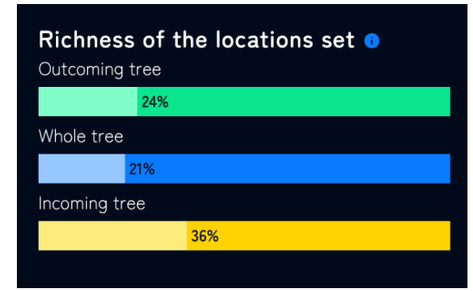
This information reveals another aspect of the user's mobility, that is how far they generally travel. We used barcharts as the representation technique for how simply and clearly they represent discrete information.

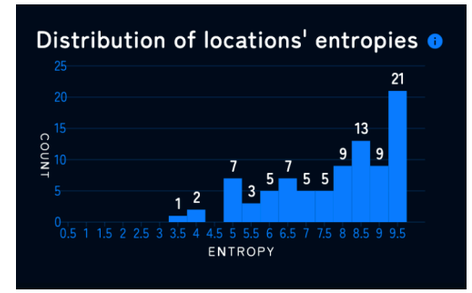
In the bottom of the page, we then introduced other components describing more specific features of the mobility tree.

Firstly, there is the out-degree diagram, showing the median and the maximum number of connections exiting from the nodes. The information inside this element is organised in such a way that we can distinguish the out-degree of the incoming tree, of the outcoming tree and also of the whole tree. We used the same colour code of the tree diagram, by consequence the former is yellowish, the second is green, and the latter is blue. From the point of view of the shape, we re-used the tree-structure to maintain the same information encoding: this uniformity should increase the understandability of the graph.

This diagram is useful to numerically understand how fragmented are the branches of the tree: for example, we could see that after reaching the root location, there could be multiple and various different travels, maybe suggesting that the user lives in a peripheral area and needs to reach the city to work, go to the gym, and go shopping.







Figg. 16, 17, 18 (from top to bottom): The median and maximum out-degree visualisation; the barchart representing the ratio between locations and total nodes; the distribution of the locations' entropies.

The next component we implemented represents the richness of the location set, meaning the proportion between the number of different locations encountered and the number of all the traversed points. This gives information about how much habitual is the user: if the richness is low, in fact, it means that the same locations are traversed quite frequently; on the contrary, an elevated richness means that the user explores many different locations, and rarely returns on the same routes. In this case, we adopted a variation of the barchart that clearly displays the proportion between the total and the location set. Maintaining the same colour system, we created a bar for the incoming, the outcoming and for the whole tree.

The last asset of the tool is the histogram of the local entropies of the locations. The entropy is the measure of information encoded by an element in a system, so in this case how much information a location carries with respect to the user's mobility. Its distribution by consequence serves to capture the complexity of the personal movement data of the user. The component is reactive with the tree visualisation: if a node is hovered, the histogram bin corresponding to the location's entropy value is highlighted, so that it is possible to make a one-to-one inspection.

An immediately visible characteristic of our data visualisation is the absence of a map, which could seem strange, since we are representing information that is highly connected with the geographical dimension. This was a choice driven by the aim to offer the analyst a tool that is respectful for the user's privacy, and to demonstrate how many strategies we can find with the limit of anonymization. Following the example of some of the visualisations presented in the introductory sections of this thesis we took the apparent limitations established by ethical concerns as an opportunity to tell

something new, or at least from a different perspective, about personal mobility data.

5.8 Preliminary results

5.8.1 Feasibility tests

Before starting the analysis, we ran several tests to measure how much time was required by each method to be computed. Essentially, we took a batch of users from a subset in the analysis dataset, in particular selecting the ones with more trajectories: this allowed us to make a pessimist estimation of the computational cost of the operations, and to select only the feasible techniques.

We immediately noticed that the basic operation required minimal time, considering the heaviest tree, with 681 trajectories, 5428 locations and an average trajectory length of 84.9 points, over 30 tests: the generation took in mean less than half a second, and also the rotation was not obstructive, since it took at most 1.1 seconds. Considering that the data was not cleaned, and that the user itself was a borderline example, we proceeded with the other tests without further explorations.

We performed deeper tests on the other operations: for each user, we built four different prefix trees, each of them considering a random sample of 25%, 50%, 75% and 100% of their trajectories. The aim of this test was also to have an idea of how much the amount of data affects the performance. We realised that the standard tree edit distance was unsuitable for a real-world problem with a lot of users to analyse, since our tree ordering algorithm took in mean from 103.4 to 560 seconds, and the distance function ranged between 10.5 and 353 seconds. If the ordering could be optimised in some way, there was no solution for the distance measure.

Tree ordering	
Percentage of data	Mean secs (30 tests)
25%	103.4099
50%	204.7914
75%	256.842
100%	559.8551

Ordered tree edit distance	
Percentage of data	Mean secs (30 tests)
25%	10.4926
50%	43.6611
75%	83.3877
100%	353.147

Unordered tree edit distance	
Percentage of data	Mean secs (30 tests)
25%	4.3097
50%	16.0067
75%	26.2233
100%	128.5133

Tables 1, 2, 3: The measured times are a pessimistic extimation of the methods' performances since the trees used for the tests were the deepest and no preprocessing has been performed on them.

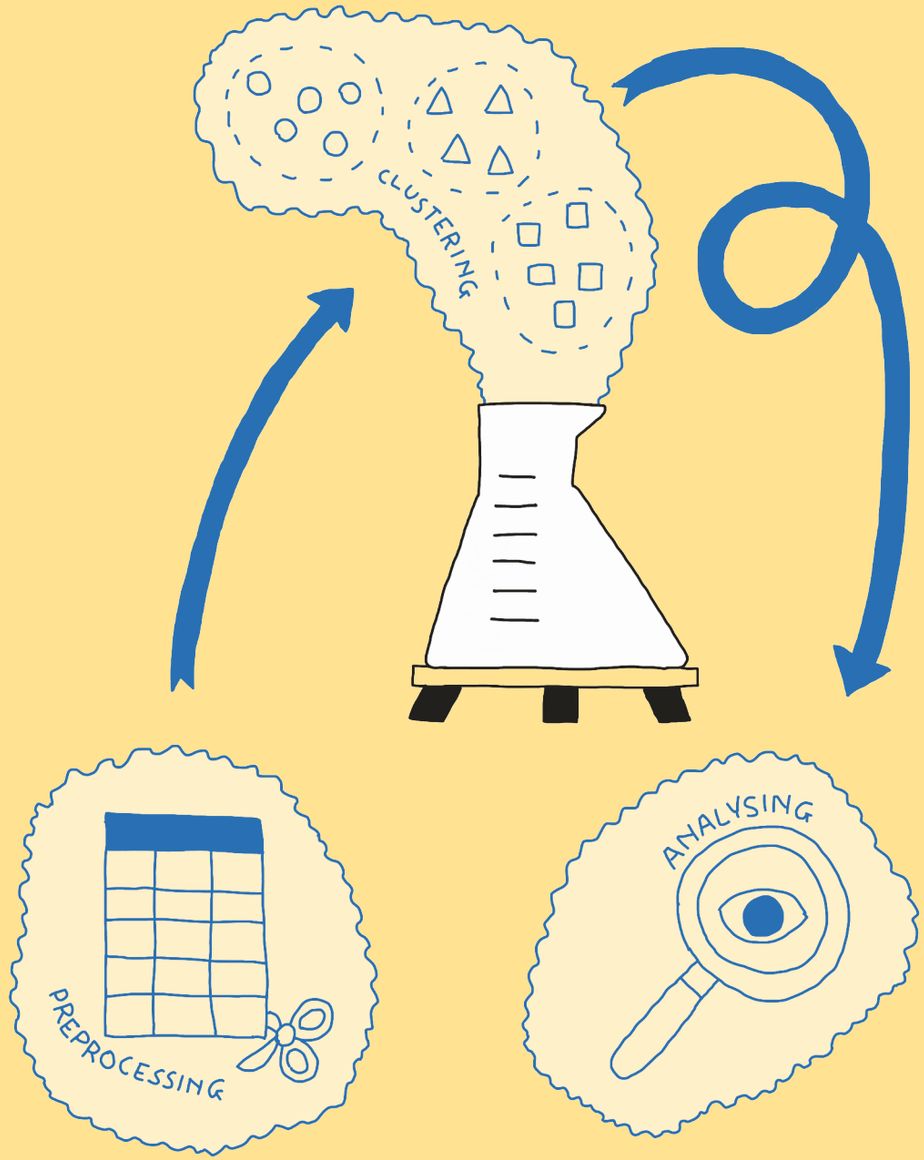
Instead, the unordered edit distance in the first tests took from 4.3 to 128.5 seconds, but later in the analysis we found a way to optimise the adjacency matrix structure required by the algorithm and the mean on the real data dropped to 0.4 seconds, also thanks to the data cleaning and selection step.

5.8.2 Project management reflections

Due to the large amount of techniques we theorised in the preliminary phase, we had to decide whether covering all of them in the analysis, and providing broad but superficial results, or focus on a set of them and make a deeper research, even if not complete for what concerns the tested methods.

We decided to go for the second route, because we were interested in testing the limits of the tree-shaped representation and because the same methodology we adopted for one case can be easily transposed for other cases. By consequence, we selected the prefix tree structure instead of the spanning tree, and the feature-based representation for the vectors, instead of the TF-IDF based one; for what concerns trees, we believe that the chosen architecture could be more understandable and better lent itself to being narrated; with regard to the vectors, we thought the second option was more coherent with the rest of the analysis, since it could provide another perspective on the same problem, but given the same premises.

6. Experiments



The aim of our experiments was to discover groups of people linked by the same mobility habits, and to verify if different cities showed different movement patterns. To do so, we performed several tests using a set of clustering algorithms, both directly on the prefix trees, the structure we selected, and on their vectorial abstraction; for the latter case, we also searched for optimal projections in less dimensional spaces, with different dimensionality reduction techniques. To compare the cities, we controlled through the z-test if there was any statistically difference between the distributions of the vector attributes, i. e. the median and maximum depth of both the incoming and the outcoming subtree, their median and maximum nodes' out-degree, and finally the ratio between number of unique locations and the total amount of nodes.

To perform the experiments, we used a dataset named *octoscana*. Its records were obtained by recording the movements of private vehicles with on-board GPS receivers, which allow the user to have a discount on the insurance in exchange for their data. The dataset consists of three typologies of table: the *seed*, the *trajlinks* and the *trajstats*. *Seed* contains all the locations' IDs, and associates them to the geographic coordinates they correspond to, so the table has three columns: *locationID*, *longitude*, and *latitude*. *Trajlinks* contains the information about each trajectory in terms of traversed points, and is composed by four columns: *userID*, *trajectoryID*, *pointID* and *locationID*; *userID* is the anonymised identifier for a vehicle, *trajectoryID* is an integer assigned to each trajectory, and *pointID* orders each point of a trajectory from 0 to the length of the travel; *locationID* is the same location identifier of the seed file. Finally, *trajstat* contains a general description for all the trajectories, like the date of the travel, the duration and the distance covered; it has nine columns: *userID*, like in the other tables, *trajectoryID*, *length* (in metres), *numPoints* that is the number of locations traversed, *startTime*, *endTime*, *duration* (in seconds), *startCell* and *endCell*, that are the boundary locations IDs.

During the analysis step, we opted for a vertical approach, focused on the most promising technique, rather than on a horizontal and probably superficial overview of all the methods theorised in the previous chapter. By consequence we decided to focus on the prefix tree representation since it seemed the most unexplored, exotic, and radical, in addition to apparently being the most difficult in terms of analysis: in fact, if

octoscana - seed		
Column name	Description	Type
ID	The ID of the location	qualitative
long	The longitude of the location	deg
lat	The latitude of the location	deg

octoscana - trajlinks		
Column name	Description	Type
ID	The user's ID	qualitative
traj_ID	The ordinal index of a user's trajectory	ordinal
point_ID	The ordinal index for each point in each trajectory (always starting from 0)	ordinal
location_ID	The ID of the location in the octoscana-seed file	qualitative

octoscana - trajstats		
Column name	Description	Type
ID	The user's ID	qualitative
traj_ID	The ordinal index of a user's specific trajectory	ordinal
length	The amount of metres travelled in this specific trajectory	continuous
num_points	The number of locations traversed in this specific trajectory	discrete
start_time	The time at which the first location of this trajectory was detected	timestamp
end_time	The time at which the last location of this trajectory was detected	timestamp
start_cell	The location ID at which the first location of this trajectory was detected	qualitative
end_cell	The location ID at which the last location of this trajectory was detected	qualitative

Tables 4 (left), 5, 6 (top): The octoscana dataset's tables and their attributes.

for vector pattern mining many techniques are available, and spanning trees are more adaptable to graph analysis, tree-shaped data are a quite overlooked horizon. Also, from a scientific communication point of view, we saw the potentiality of the tree analogy, and how many data storytelling opportunities we could exploit. We chose the more difficult and less explored path in order to evaluate the feasibility of the representation and the major issues the analyst could encounter while working with it.

6.1 Preprocessing

Due to the huge amount of locations and trajectories generally present for each user, we need to consider some data cleaning techniques to reduce on one hand the computational time and on the other the noise that could affect the analysis.

6.1.1 Choice of the locations

We decided to focus on two cities, Pisa and Florence, as analysis prototypes, aiming to find which techniques are the most useful and to understand the limits of the selected representations. The two cities differ both in extension and in number of users, by consequence constitute a good starting point to detect potential differences.

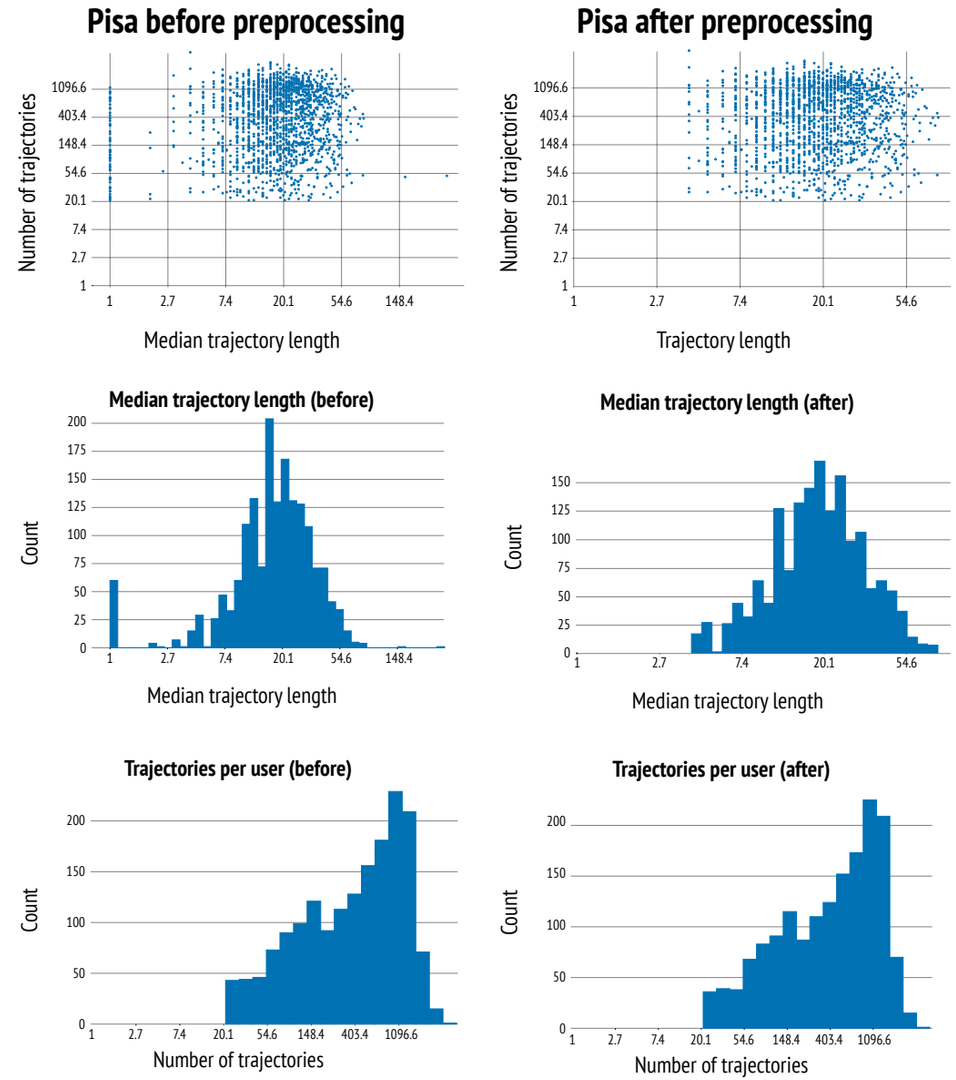
From the octoscana dataset, then, we selected a square perimeter of coordinates surrounding the city and including the adjacent area, assuming that not all the users live inside its exact boundaries.

6.1.2 Users and trajectories pruning

The preprocessing of each user's dataset required multiple passages to ensure to discard outlier trajectories and all the profiles with not enough data to be compared with the others.

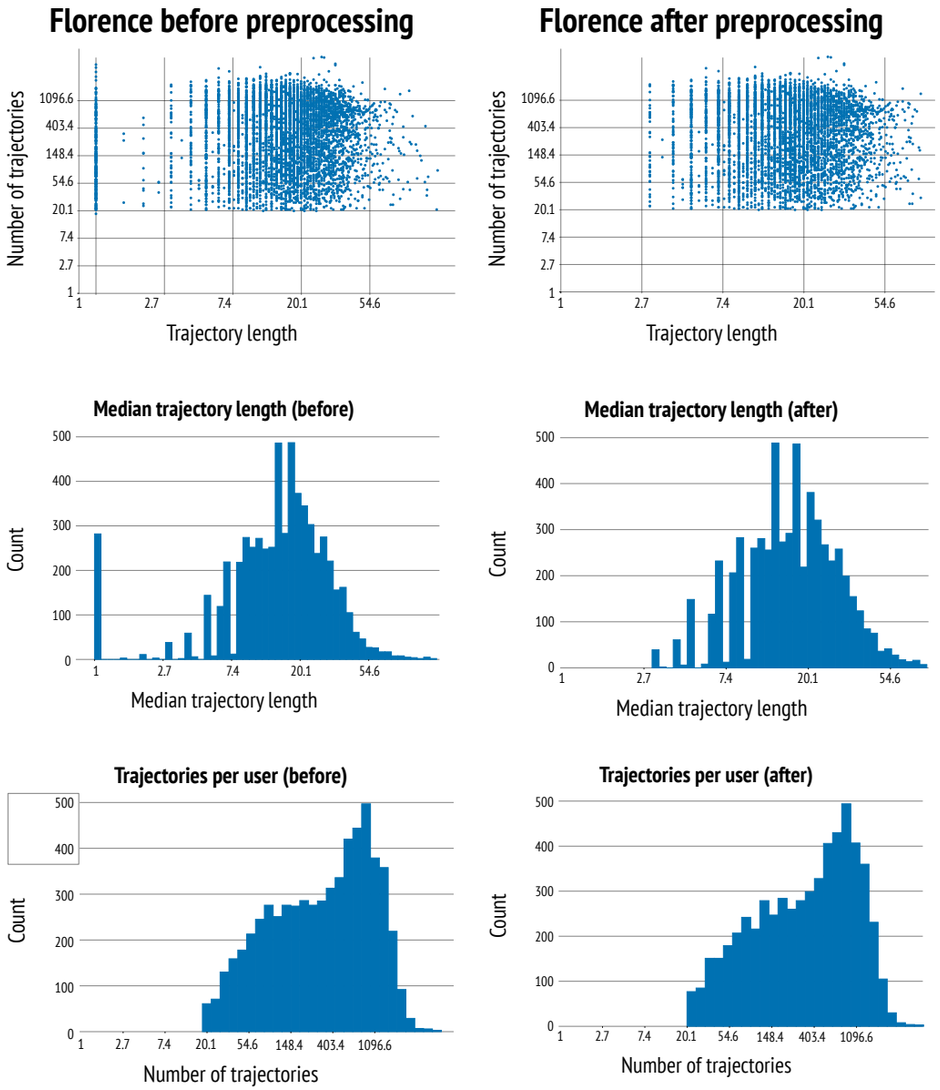
First of all, once we selected in the seed file the locations inside the perimeter, we selected the users whose mobility is developed at least by 65% in the city. To calculate this percentage, we copied the user's trajectories dropping all the locations not inside the perimeter, and compared the sizes of the two datasets: if the clean dataset's size was at least the 65% of the original, we kept the user. We chose this strategy in order not to penalise the user's whose mobility is reducible to few locations in the city, and in order not to promote infrequent travels traversing most of the city's locations. Then, we proceeded with a user-specific data cleaning: we deleted all the outlier trajectories in terms of length. To do so, we calculated the distribution of the points traversed in each travel, and deleted the outliers according to the IQR test. We chose to rely on the trajectory length in terms of traversed locations instead of in terms of travelled metres because, thanks to the data tessellation, we knew that each point had a distance of around 200m from the adjacent ones; we considered it a good approximation. Once we did this, it was time to delete global outliers: we extracted for each user the quantity of trajectories and their median length, and, after we converted them to the logarithmic scale, we re-performed the IQR test on the two distributions and deleted the outliers.

For what concerns the dataset of Pisa, we started with 1895 users, that became 1636 after the cleaning. In the starting dataset, we observed a distribution of the number of trajectories per user with a mean of 588.79, a median of 387, and a standard deviation of 705.1; once cleaned, these values turned to 637.1, 501, and 548.7. The distribution of the lengths of each trajectory, instead, started with a mean of 29.15, a median of 18, and a standard distribution of 51.02, and ended with those values becoming 23.73, 17, and



Figg. 19 (left), 20 (right): The Pisa dataset before and after the preprocessing: focus on the median trajectories' length and on the number of trajectories for each user.

24.5. The original number of users in the dataset of Firenze, instead, was 7155, and became 5762 after the whole cleaning. The initial distribution of the number of trajectories per user had mean 487.47, median 255, and standard deviation 841.48, values that became 553 , 386, and 532.697 after the cleaning. Instead, the distribution of the lengths of each trajectory had originally a mean of 25.67, a median of 14, and a standard



Figg. 21 (left), 22 (right): The Florence dataset before and after the preprocessing: focus on the median trajectories' length and on the number of trajectories per user.

distribution of 49.42, values that became 20.468, 14, and 22.795.

Ending the cleaning process, we proceeded with the users' tree generation. Since the amount of data and its complexity, we slightly simplified the trajectories using the Ramer–Douglas–Peucker (RDP) algorithm; RDP approximates a path deleting the points if they do not differ more than a certain threshold, in terms of direction, from the previous and the following points. We found that an epsilon of 0.001 provided a good approximation without losing too much information. In fact, we also tried to use 0.005 and 0.0005, but the final results were conceptually equal, so we opted for the middle way, with not too much approximation and at the same time with faster distance calculations during the unsupervised learning phase.

Starting from these clean data, we generated the mobility prefix tree for each user.

6.1.3 Distance matrix generation

Since clustering algorithms are generally expensive in terms of the amount of needed computations, because they iteratively need to calculate distances between points, we decided to pre-compute the distance matrix between trees, using the already mentioned unordered tree edit distance.

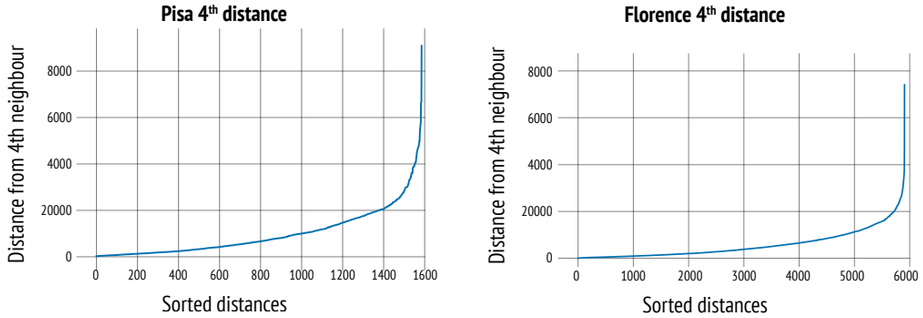
6.2 Choice of the clustering algorithms

The first algorithm we thought of was the k-medoid because on one hand we needed to find a representative tree for each cluster but mainly because tree data is not suitable to be used with algorithms that use artificial points as centroids. We performed a grid-search on the k measuring the silhouette value for each value between 2 and 30, for 30 different random medoid initialisations. We also considered the amount of samples in each cluster and the silhouette of each single cluster, in order to have a more detailed view of the problem. We also tested DBSCAN and OPTICS as density-based approaches, as well as many hierarchical clustering algorithms such as single, complete, and centroid linkage.

6.2.1 Clustering trees

Both for Pisa and for Florence, the K-medoid clustering with unordered tree edit distance resulted in one prevalent cluster with mean high silhouette (on average, 0.71 for the former and 0.62 for the latter), and other sparse and fragmented clusters with negative silhouette, no matter the value of k; however, in general the best results were obtained with k equal to 4. We observed the same results with different cleaning techniques and with different RDP epsilons, and we kept the configuration that brought the higher silhouette, that is the result of the already explained preprocessing. For Pisa, the main cluster contained on average 1481.5 users, while the one for Florence 2552, suggesting that a large number of people seems to have similar movement habits.

To validate these results, we also ran DBSCAN and OPTICS, and we observed the same phenomenon. To select the epsilon to be used in DBSCAN, we plotted the distance from the ith neighbour, and since in a range of i between 2 and 8 there were no significant



Figg. 23 (left), 24 (right): The 4th distance plot of the two cities used to run the DBSCAN algorithm.

differences, we kept the 4th neighbour distance, as suggested by the authors of the algorithm; for Pisa, the epsilon was equal to 3000, for Florence to 2000. DBSCAN agglomerated the 96% of the samples of Pisa in a unique cluster, and the 97% of the ones of Florence. Comparable results were obtained using OPTICS.

Finally, also the hierarchical clustering, in all the tested techniques, showed the presence of a unique cluster generated from a predominant subsection of the whole dataset.

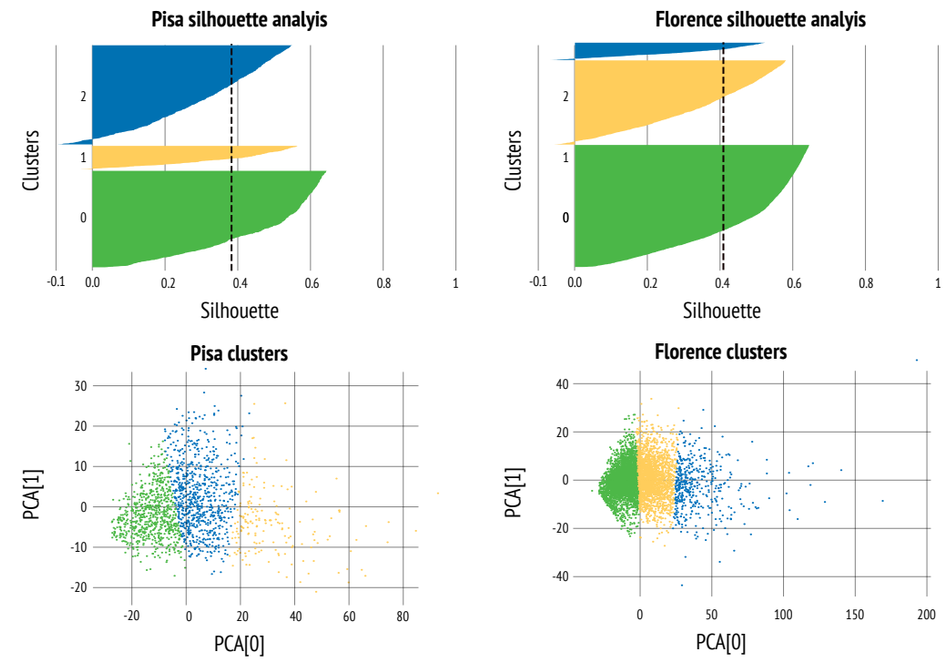
6.2.3 Clustering vectors

The next step of the analysis consisted in performing the clustering on the vectorial abstractions of the prefix trees, aiming to discover more general patterns in the mobility of the users, and completely ignoring the geographical dimension of the movements. In this case, we only performed k-means combined with some feature projection techniques.

For both cities, we performed a grid search on the number of clusters (k), considering a range between 2 and 6, and we computed the average silhouette of each configuration; the test was repeated for: the vectors with standardised values, their projection through independent component analysis (ICA), principal component analysis (PCA), with a number of features from 2 to 4, and multidimensional scaling.

Both for Pisa and for Florence the best results were obtained through a 2-means on the 2-features projected on the principal component, with a silhouette of 0.45 for the former and a 0.49 for the latter. However, in all the configurations we tested, we noticed that no meaningful cluster was obtained: also the higher results were divisions of a bigger dense agglomerate of points, corresponding to the majority of the users.

For this reason, we decided to pick the second best result, that is a 3-means on the 2-dimensional PCA, to see if any interesting difference emerged in this space, and to provide three prototypes of users, to see if some difference occurs, despite the presence of a unique big and dense group. We visualised the trees which were closer to each cluster's centroid, for both cities: we name



Figg. 25 (left), 26 (right): The results of the 3-means on the vectors projected in two dimensions through PCA.

the prototypes of Pisa P0, P1 and P2, and the ones of Florence F0, F1, and F2.

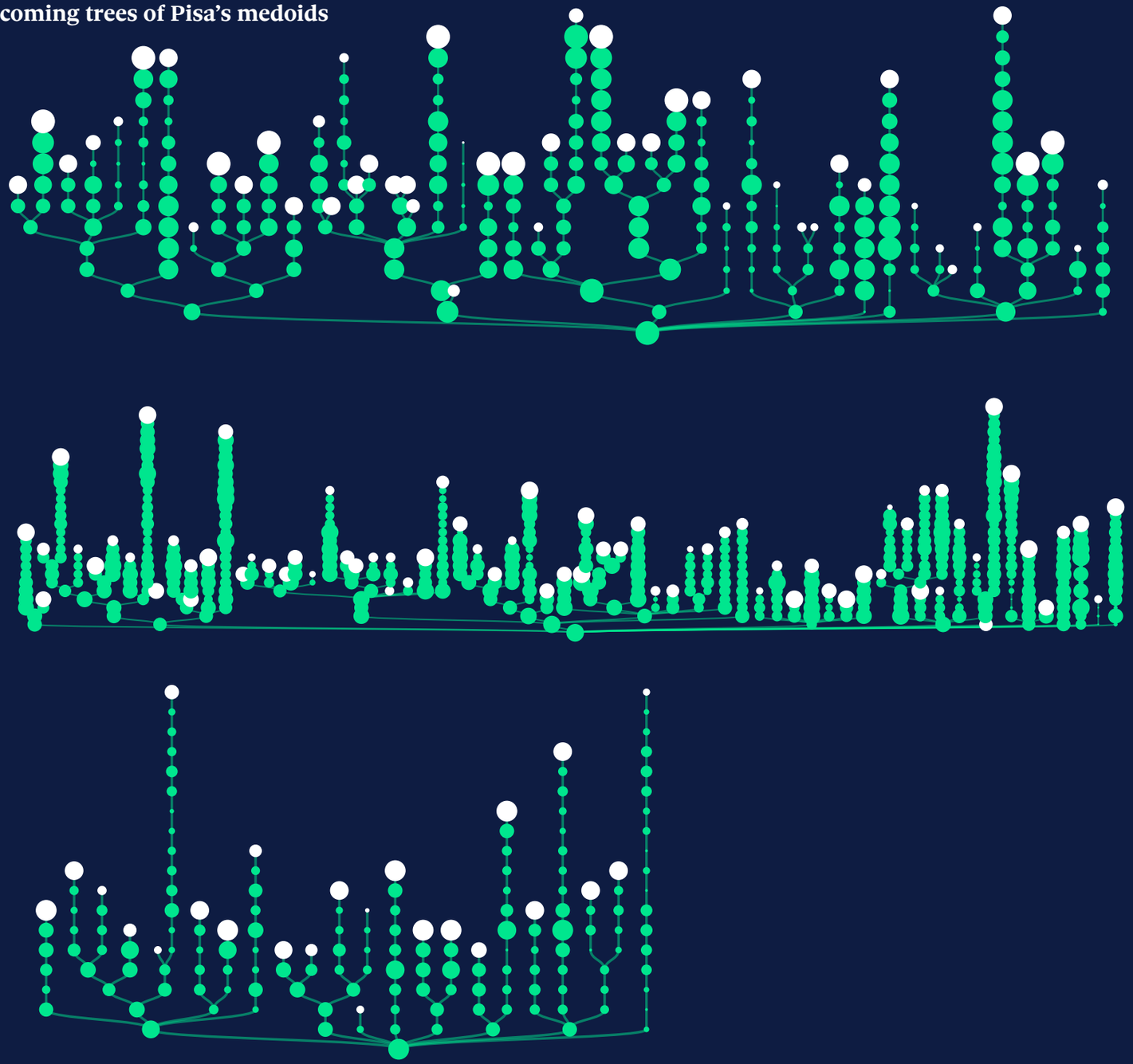
For Pisa, the most visible difference is the frequency of the nodes, which highlights three different phenomena: in P0, we see branches containing both quite frequent and rare locations (the bigger and the smaller nodes); instead, P1 is characterised by generally big nodes, while in P2 we see generally infrequent locations. To interpret these visual differences, we can also read the values relative to the richness of the location set, as well as take a look at the distribution of the nodes' entropies.

We see that P1 has the lower location richness: this is coherent with the generally frequent nodes, meaning that the user generally visits the same locations in the same order; also, since the variance of the nodes sizes is quite small, meaning similar frequencies among all the locations, we also observe the higher values of entropy.

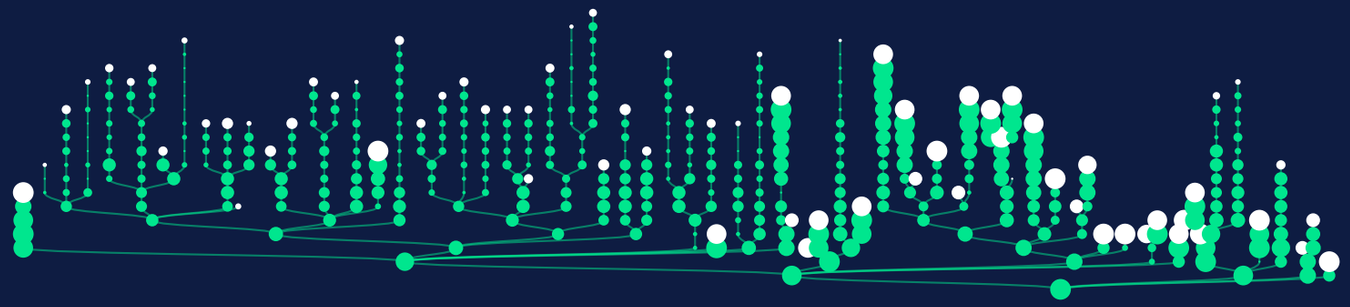
On the contrary, P2 is richer in terms of locations, and this is also validated by the smaller radius of the nodes: the user seems to be less regular in the choice of the paths to

Fig. 27 (next two pages), 28 (following two pages): The outcoming trees of the medoid users of Pisa P0, P1, and P2, and Florence, F0, F1, F2. We preferred to show only the topmost part of the tree, corresponding to the outcoming paths, to allow the user better appreciating the details. The incoming trees present analog traits.

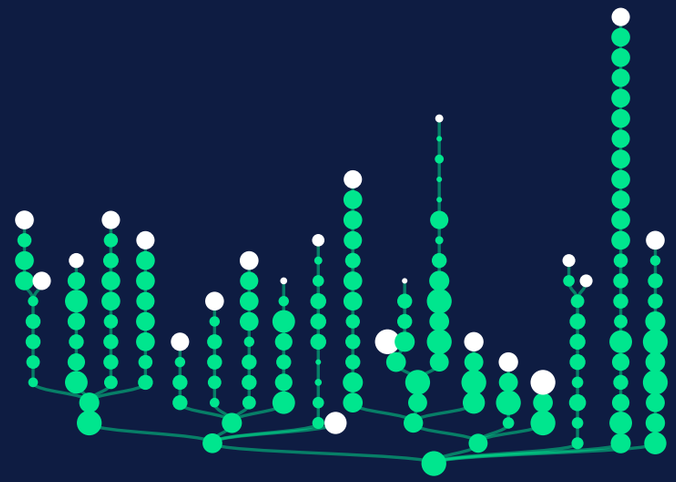
Outcoming trees of Pisa's medoids



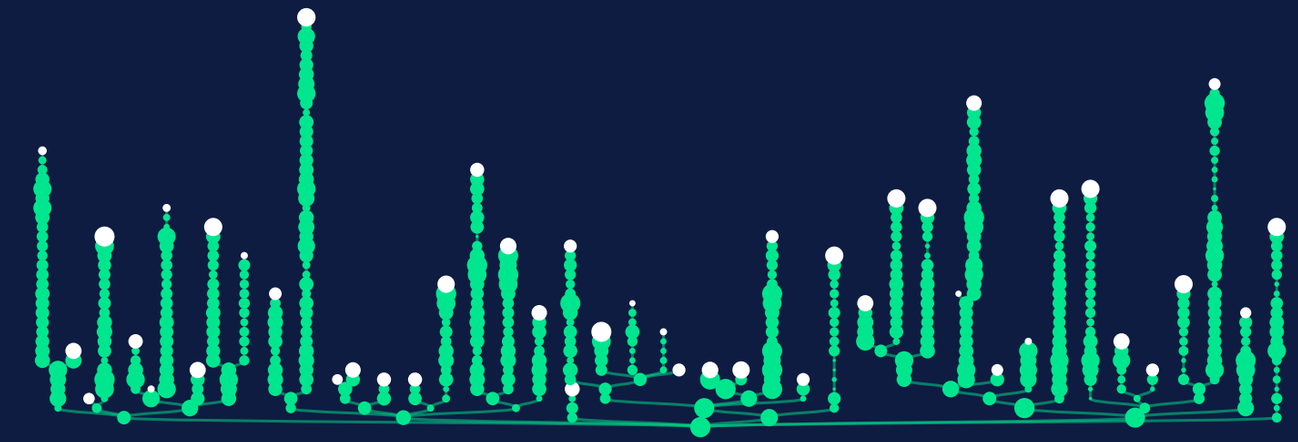
Outcoming trees of Florence's medoids



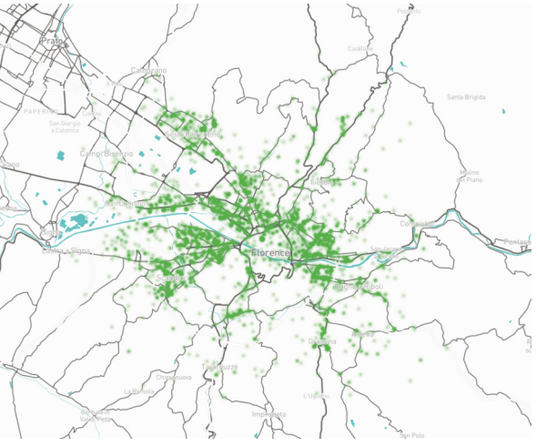
F0

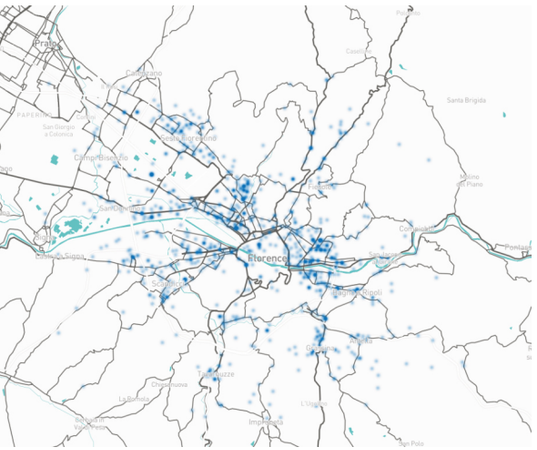


F1



F2





Figg. 29, 30, 31 (from top to bottom): The coordinates of the root locations of each user of the Florence dataset, divided by cluster and plotted on the map. Pisa's output is analog: no geographical pattern emerged.

cross. Since the locations' frequencies are slightly more variable, we observe a lower maximum entropy.

P0, finally, seems to be in a certain sense at a midway between P1 and P2: if we can notice quite frequent locations, we also observe very rare ones, so the frequency variance is higher than in the two other prototypes. Also the richness of the location set is close to the mean of the ones of P1 and P2. A very interesting detail in P0 is the presence of a very thick branch which probably identifies the routinary travel the user does to go to work.

The three prototypes seem not to differ too much in terms of node out degree.

When looking at the visualisation mapped on the cumulative distance travelled in each path, we do not see great differences in terms of maximum kilometres travelled in one single trip. Also, the three prototypes have a similar distribution of the distance of the leaves: in all cases, we see the majority of travels ending within the first seven kilometres, and the more we are distant from the root, the few possible travels (and by consequence, leaves) we can observe.

For what concerns Florence, we see less visible differences regarding the locations' frequencies, excluding F0, where we notice more variance and by consequence a slightly greater maximum value of entropy. The three prototypes are similar in terms of richness of the location set and in terms of out-degree. In Florence, it seems to be more discriminant the amount of possible prefixes: we see that F1 has a quite reduced tree, while F0's root has a very large number of successors, meaning a lot of variety of prefixes; F2 is larger with respect to F1, but it results quite small if compared with F0.

By looking at the distances travelled, we can see that they are quite diversified, contrary to what we observed in the users Pisa. Also, the distribution of the number of leaves (start or stop locations) with respect to the distance to the root is less graduated: it seems there are particular ranges of distances that are usually travelled.

Another qualitative test we performed was to plot the root of each user dividing them by cluster on the map of each city to verify if different areas of the clustering corresponded to different urban zones. Neither in Pisa nor in Florence seems to exist a geographical distinction of the groups.

6.3 Comparing the two cities' distributions

Another analysis we performed was to compare the characteristics of the trees belonging to the two cities. To do it, we considered the features used to encode the trees into the vectors used for the previous step. We performed a z-test for each dimension after having standardised the values. Even if for some attributes the test indicated statistically different means, we do not take these results as too reliable, since some distributions resulted quite skewed: by looking at the table 7, in fact, we can see that in these cases the difference between the means is quite small; the greater difference seems to be in the outcoming trees, where we have a difference of approximately two units in the out-degree and of 0.15 in their median depth. As shown in the plots, however, we are not looking at a blatant difference.

6.4 Interpretation

Against original expectations we were unable to find distinct categories of users given the tree-shaped data and the derived vectorial representation. By observing the most central sample of each cluster in both cities, however, we noticed few characteristics that distinguish each of them. This could mean that the data space is well graduated, i. e. there are intermediate steps between one type of user and another. This hypothesis seems to be supported by the fact that the results don't change if the preprocessing, the algorithm, or the data structure is modified.

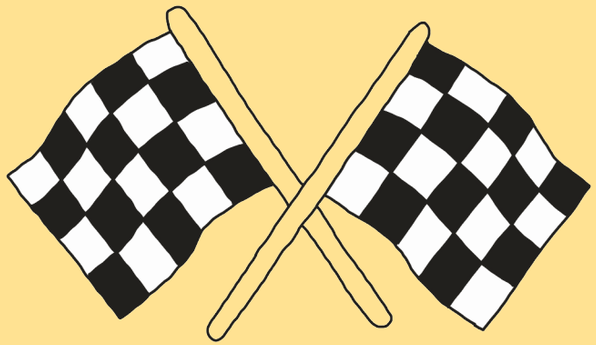
The most interesting factor that emerged has been the fact that it seems these clusters are distinguished by the regularity of their paths: between the observed users, in fact, there is at least one who has quite equi-probable nodes among all the branches, while the others present very frequent locations opposed to very rare ones. This characteristic, that is better visible in Pisa, seems to be analogous to the idea of the returners and the explorers suggested by Pappalardo et. al. (2015)[10].

Also, the fact that there is no correspondence between the root's position of the users and their belonging to one specific cluster is a good clue suggesting that the grouping is not biassed by the traversed locations.

Table 7: The results of the z-test on the vectors' attributes of the two cities. The test was performed after standardising the values.

Comparing Pisa's and Florence's distributions				
Attribute	Test statistic	P-value	Mean Pisa	Mean Florence
Median depth incoming tree	1.606	0.108	8.0767	7.898
Median depth outcoming tree	-2.531	0.011	1.559	1.699
Max depth incoming tree	0.94	0.347	24.493	24.181
Max depth outcoming tree	1.937	0.053	20.287	19.576
Locations / nodes ratio	-3.946	0.0007	0.15	0.164
Median incoming tree out-deg	0.533	0.594	2	2
Median outcoming tree out-deg	0.571	0.568	1.955	1.952
Max incoming tree out-deg	0.571	0	14.392	12.666
Max outcoming tree out-deg	4.854	0.00001	9.833	9.148

7. Conclusions



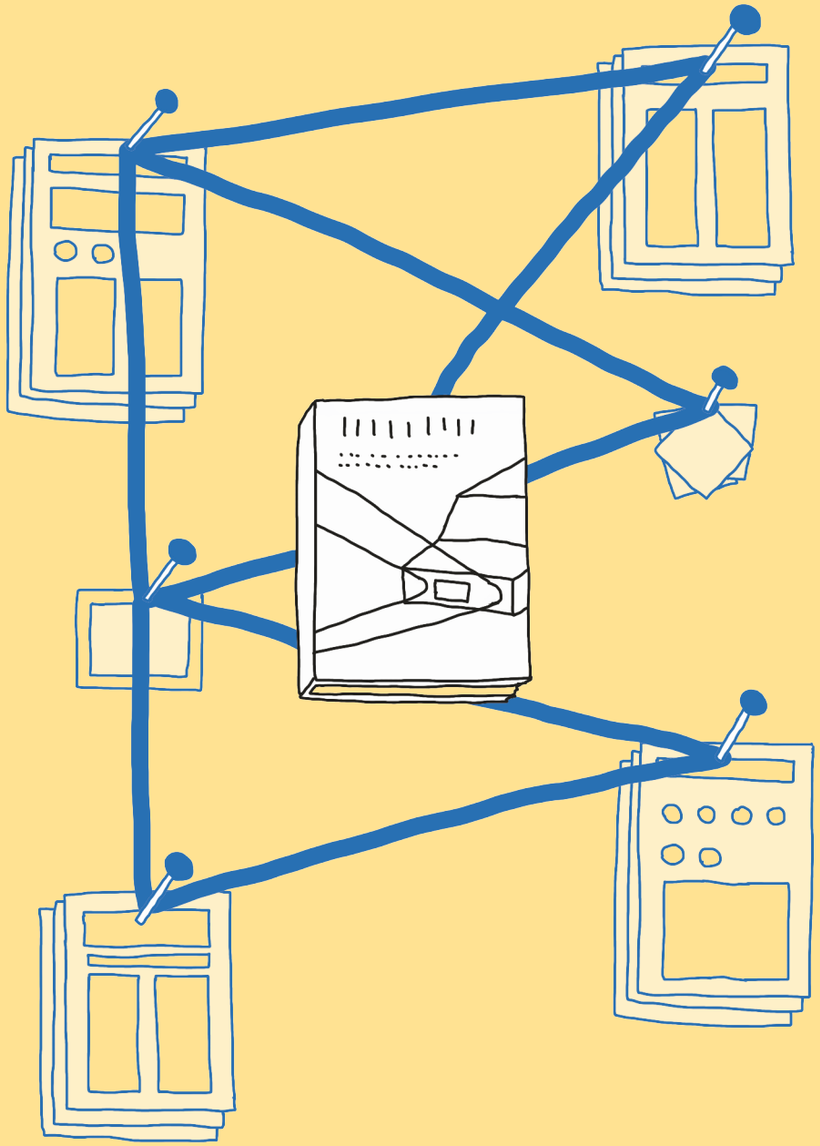
In this thesis we explored how we can map personal mobility data into tree structures. In particular, we focused on a variation of the prefix tree that is rotated under the most frequent location. As a distance function between them, we used the tree unordered edit distance, a special type of tree edit distance that allows us to compare them even if their branches are not ordered from left to right. We also created a vectorial description of the trees, which captures its structural characteristics. For the analysis, we focused on the cities of Pisa and Florence, and we exploited unsupervised learning with the aim to find different types of user based on both the tree and the vector representations. The results were consistent in all the tests, in spite of the algorithm, the data type, and the preprocessing, and showed the presence of a unique, dense group, with a small percentage of noise.

For what concerns future developments, with the help of the visualisation tool, and setting as root semantically different locations, the scientist can try to understand in depth how humans move, maybe also comparing trees generated from different sets of data belonging to the same person: for example, we could compare the working-week tree with the week-end tree, or the winter and the summer tree.

Also, we could try to compare cities from different countries, or the trees generated from other vehicles besides cars, and see if the results we obtained change. Another aspect we did not explore is the clustering inside a unique tree, to understand if there are recurrent patterns within the user's movements.

To sum up, this work suggests the potentialities arising while thinking about mobility in the shape of trees. The results we obtained seem to confirm the analysis of Pappalardo et. al (2015)[10] which see a dichotomy between returners and explorers.

References



[1] Fry, B. (2009). Learning from Lombardi. url (https://benfry.com/exd09/)

[2] Giannotti F., Pedreschi D. (2008). Mobility, data mining and privacy: geographic knowledge discovery. Heidelberg: Springer; p. 189–211.

[3] Barth, D., Bellahsene, S., & Kloul, L. (2011). Mobility Prediction Using Mobile User Profiles. 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, 286-294.

[4] Andrienko, G., Andrienko, N., Hurter, C., Rinzivillo, S., Wrobel, S. (2011). From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data. In: Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST) 2011; IEEE Computer (pp. 161–170). Society Press.

[5] Larcom, S., Rauch, F., Willems, T., (2017). The Benefits of Forced Experimentation: Striking Evidence from the London Underground Network, The Quarterly Journal of Economics, Volume 132, Issue 4, November 2017, Pages 2019–2055, https://doi.org/10.1093/qje/qjx020

[6] Volksentscheid Berlin Autofrei (2020). https://volksentscheid-berlin-autofrei.de/index.php?lang=en, visited on March 2022.

[7] Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D. (2007). Trajectory pattern mining. KDD '07.

[8] Andrienko, G., Andrienko, N., Rinzivillo, S., Nanni, M., Pedreschi, D., Giannotti, F. (2009). Interactive Visual Clustering of Large Collections of Trajectories. VAST.

[9] Guidotti, R., Trasarti, R., Nanni, M., & Giannotti, F. (2015). Towards User-Centric Data Management: Individual Mobility Analytics for Collective Services. In Proceedings of the Fourth ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (pp. 80–83). Association for Computing Machinery.

[10] Pappalardo, L., Simini, F., Rinzivillo, S. et al. (2015). Returners and explorers dichotomy in human mobility. Nat Commun 6, 8166. https://doi.org/10.1038/ncomms9166.

[11] Trasarti, R., Pinelli, F., Nanni, M., & Giannotti, F. (2011). Mining Mobility User Profiles for Car Pooling. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1190–1198). Association for Computing Machinery.

[12] Lussu, G. (1999), La lettera uccide, Roma, Stampa Alternativa.

[13] Harris, R. (2000). Rethinking Writing, London, Athlone.

[14] Perondi, L. (2012). Sinsemie, Scritture nello spazio.

[15] Jeung, H., Yiu, M. L., Zhou, X., & Jensen, C. S. (2010). Path prediction and predictive range querying in road network databases. VLDB Journal, 19(4), 585-602. DOI: https://doi.org/10.1007/s00778-010-0181-y

[16] Zhou, H., Hirasawa, K. 2019. Spatiotemporal traffic network analysis: technology and applications. Knowl. Inf. Syst. 60, 1 (July 2019), 25–61. DOI: https://doi.org/10.1007/s10115-018-1225-7

[17] Riccardo, G. (2017). Personal data analytics - capturing human behaviour to improve self-awareness and personal services through individual and collective knowledge, Ph.D. dissertation, University of Pisa.

[18] Trasarti, R., Pinelli, F., Nanni, M., & Giannotti, F. (2011). Mining Mobility User Profiles for Car Pooling. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1190–1198). Association for Computing Machinery.

[19] Bertin, J. (1967). Sémiologie graphique, Paris, Mouton/Gauthier-Villars.

[20] Tufte, E. R. (1983), The Visual Display of Quantitative Information (2nd ed.), Cheshire, CT: Graphics Press.

[21] Ihad Inbar, Noam Tractinsky and Joachim Meyer. Minimalism in information visualization: attitudes towards maximizing the data-ink ratio. DOI: http://portal.acm.org/citation.cfm?id=1362587.

[22] Bateman, S., Regan L. Mandryk, C. G., Genest, A., McDine, D., Brooks, C. (2010). Useful junk? the effects of visual embellishment on comprehension and memorability of charts. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10). Association for Computing Machinery, New York, NY, USA, 2573–2582. DOI: https://doi.org/10.1145/1753326.1753716

[23] Chalabi, M., & Gray, J. (2021). Sketching With Data. In L. Bounegru & J. Grey (Eds.), The Data Journalism Handbook: Towards A Critical Data Practice (pp. 174-181). Amsterdam University Press. DOI:10.1017/9789048542079.026

[24] Daston, L., & Galison, P. (2007). Objectivity. Zone Books.

[25] Ceneda, D., Andrienko, N., Andrienko, G., Gschwandtner, T., Miksch, S., Piccolotto, N., et al. (2020). Guide me in analysis: A framework for guidance designers, Computer

Graphics Forum, vol. 39, no. 6, pp. 269-288.

[26] S. Rinzivillo, L. Gabrielli, M. Nanni, L. Pappalardo, D. Pedreschi and F. Giannotti (2014). The purpose of motion: Learning activities from Individual Mobility Networks, International Conference on Data Science and Advanced Analytics (DSAA), 2014, pp. 312-318. DOI: 10.1109/DSAA.2014.7058090.

[27] Guidotti, R., Trasarti, R., Nanni, M., Giannotti, F., Pedreschi, D. (2017). There's a Path for Everyone: A Data-Driven Personal Model Reproducing Mobility Agendas. In 2017 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2017, Tokyo, Japan, October 19-21, 2017 (pp. 303–312). IEEE.

[28] Guidotti, R., Trasarti, R., & Nanni, M. (2015). TOSCA: Two-Steps Clustering Algorithm for Personal Locations Detection. In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. Association for Computing Machinery.

[29] Zhao, X., Pi, D., & Chen, J. (2020). Novel trajectory privacy-preserving method based on prefix tree using differential privacy. Knowl. Based Syst., 198, 105940.

[30] Andrienko, N., Andrienko, G., Barrett, L. , Dostie, M., Henzi, P. (2013) Space Transformation for Understanding Group Movement, in IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 12, pp. 2169-2178, Dec. 2013. DOI: 10.1109/TVCG.2013.193.

[31] Andrienko, Gennady & Andrienko, Natalia. (2012). Privacy Issues in Geospatial Visual Analytics. DOI: 10.1007/978-3-642-24198-7_16.

[32] Douieb, K. (2021, June 9). All the passes. Observable, https://observablehq.com/@karimdouieb/all-the-passes. Last visited April 2022

[33] Dell'Orletta, F., Montemagni, S., Venturi, G. (2013). Linguistic Profiling of Texts Across Textual Genres and Readability Levels. An Exploratory Study on Italian Fictional Prose. In Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013, pages 189–197, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

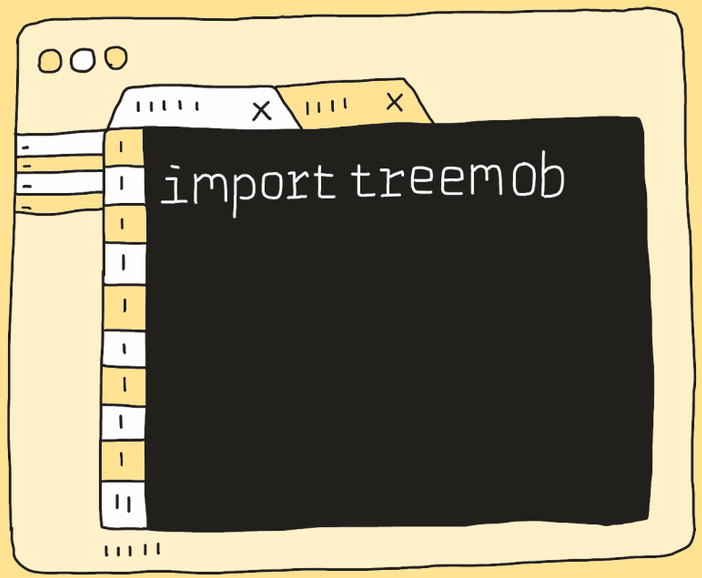
[34] Bille, P. (2005). A survey on tree edit distance and related problems, Theoretical Computer Science, Volume 337, Issues 1–3, 2005, Pages 217-239, ISSN 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2004.12.030

[35] Fioravanti, M. (2021). Treemob, a Python library for tree-shaped representation of mobility data. Url: https://github.com/sblbl/treemob (last visited: April 2022)

[36] Fioravanti, M., (2021). Treemob Visualisation. Url: https://github.com/Sblbl/treemob_viz. (last visited: April 2022)

[37] Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations, Proceedings 1996 IEEE Symposium on Visual Languages, 1996, pp. 336-343. DOI: 10.1109/VL.1996.545307.

Appendix



This appendix contains the description of the classes and the methods developed for the Treemob Python module we wrote to perform our analysis. The full code is available at the link https://github.com/sblbl/treemob.

We start with the PrefixTree class and its methods, then we pass to the SpanningTree class, for which we do not repeat the methods that are equal to the ones available in PrefixTree. Finally, we review all the useful functions used for the data preparation, archiviation, and mining.

treemob.PrefixTree: the class				
Description	The prefix tree data structure adapted to personal mobility data.			
Initial parameters	Description	Type	Optional	Default
trajectories	The list of trajectories belonging to a user. Each trajectory is a list of location IDs	list	False	-
root	The location ID of the designated root	str	False	-
nodes_weights	A dictionary in form {locationID : weight}	dict	True	None
edges_weights	A dictionary in form {(from, to) : weight}.	dict	True	None
Internal variable name	Description	Type		
in_trajs	The portion of each trajectory from the start point to the root	list		
out_trajs	The portion of each trajectory from the root to the end point	list		
base_tree	The prefix tree including in and out paths	nx.DiGraph		
base_labels	{nodeID : label} in the base_tree	dict		
in_tree	The prefix tree including the incoming paths	nx.DiGraph		
in_labels	{nodeID : label} in the in_tree	dict		
out_tree	The prefix tree including the otcoming paths	nx.DiGraph		
out_labels	{nodeID : label} in the out_tree	dict		
brackets	When the function to_brackets is run, it stores the tree in brackets format	dict		

treemob.PrefixTree.show				
Description	Plots the tree.			
Input variable name	Description	Type	Optional	Default
with_labels	Whether to show the locationID in the nodes	bool	True	True
tree	The tree to show	str: ['base_tree', 'in_tree, out_tree']	True	'base_tree'
Returns	None			

treemob.PrefixTree.to_json				
Description	Converts the tree in json format.			
Input variable name	Description	Type	Optional	Default
tree	The tree to convert	str: ['base_tree', 'in_tree, out_tree']	True	'base_tree'
Returns	tree: dict			

treemob.PrefixTree.to_brackets				
Description	Converts the tree in bracket format.			
Input variable name	Description	Type	Optional	Default
tree	The tree to convert	str: ['base_tree', 'in_tree, out_tree']	True	'base_tree'
Returns	tree: dict			

treemob.PrefixTree.lookup_matrix				
Description	Creates the lookup matrix of the tree.			
Input variable name	Description	Type	Optional	Default
tree	The internal tree to convert	str: ['base_tree', 'in_tree, out_tree']	True	'base_tree'
Returns	tree: dict			

treemob.PrefixTree.add_nodes_weights				
Description	Assigns to each node a weight.			
Input variable name	Description	Type	Optional	Default
tree	The tree to modify	str: ['base_tree', 'in_tree, out_tree']	True	'base_tree'
default_val	The value of the weight if not defined in self.n_weights	int	True	1
Returns	None			

treemob.PrefixTree.add_edges_weights				
Description	Assigns each edge from one node labeled A to one node labeled B a weight. Edges between couples of nodes with the same couple of labels have the same weight.			
Input variable name	Description	Type	Optional	Default
tree	The tree to modify	str: ['base_tree', 'in_tree, out_tree']	True	'base_tree'
default_val	The value of the weight if not defined in self.e_weights	int	True	1
Returns	None			

treemob.SpanningTree: the class				
Description	The prefix tree data structure adapted to personal mobility data.			
Initial parameters	Description	Type	Optional	Default
trajectories	The list of trajectories belonging to a user. Each trajectory is a list of location IDs	list	False	-
root	The location ID of the designated root	str	False	-
nodes_weights	A dictionary in form {locationID : weight}	dict	True	None
edges_weights	A dictionary in form {(from, to) : weight}.	dict	True	None
method	Wherether to include all the points or only the start and end points	string: ['all', 'stops']	True	'all'
Internal variable name	Description	Type		
trajs	The portion of each trajectory from the start point to the root	list		
root	The location ID of the root	str		
graph	The spanning graph	networkx.Graph		
tree	The rooted spanning tree	nx.DiGraph		
brackets	When the function to_brackets is run, it stores the tree in brackets format	dict		

treemob.save_tree				
Description	Saves a PrefixTree or SpanningTree object in a .pkl file.			
Input variable name	Description	Type	Optional	Default
obj	The tree to be saved	treemob.PrefixTree or SpanningTree	False	-
filename	The name of the file that will be saved (no extension is needed)	str	False	-
Returns	None			

treemob.load_tree				
Description	Loads a .pkl file containing a PrefixTree or SpanningTree.			
Input variable name	Description	Type	Optional	Default
filename	The name of the file to be opened (no extension is needed)	str	False	-
Returns	tree: treemob.PrefixTree or treemob.SpanningTree			

treemob.generate_trajs				
Description	Loads a .pkl file containing a PrefixTree or SpanningTree.			
Input variable name	Description	Type	Optional	Default
trajs	The table with df.columns containing ['traj', 'cell'], where 'traj' is the ID of the trajectory and 'cell' is the ID of the traversed location	pandas.DataFrame	False	-
Returns	trajs: list of trajectories			

treemob.get_link_weights				
Description	Returns the weight of each available link from two locations in the user's trajectories set.			
Input variable name	Description	Type	Optional	Default
trajs	The set of trajectories from whom to compute the weights	list	False	-
Returns	dict:{(location_a, location_b): value}			

treemob.get_loc_weights				
Description	Returns the frequency of each location in the user's trajectories set.			
Input variable name	Description	Type	Optional	Default
trajs	The set of trajectories from whom to compute the weights	list	False	-
Returns	dict {location_a: value}			

treemob.most_frequent_loc				
Description	Computes the most frequent location of a trajectory set.			
Input variable name	Description	Type	Optional	Default
data	The node weights from which to extract the locations	list	False	-
Returns	str: the id of the most traversed location			

treemob.order				
Description	This function orders the tree's branches from left to right according to their weight (the sum of the weights of its links). This passage is fundamental to obtain valid results when using the basic tree edit distance, since it proceeds in an ordered way.			
Input variable name	Description	Type	Optional	Default
unord_tree	The PrefixTree to order	treemob.PrefixTree	False	-
Returns	networkx.DiGraph			

treemob.edist_adj				
Description	Computes the adjacency matrix of the tree in the format required by edist.			
Input variable name	Description	Type	Optional	Default
tree	A PrefixTree	networkx.DiGraph	False	-
nodes	The nodes to be considered	str	False	-
Returns	list			

treemob.preprocess_trajs				
Description	Cleans a user's trajectories by using RDP, then finds the root as the most frequent location and finally deletes all trajectories not containing it.			
Input variable name	Description	Type	Optional	Default
trajs	The list of trajectories, being each a list of location IDs	list	False	-
loc_ids	The locations' IDs	list	False	-
loc_longs	The locations' longitudes	list	False	-
loc_lats	The locations' latitudes	list	False	-
epsilon	The RDP epsilon value	float	False	-
Returns	list: the clean trajectories			

treemob.tree_to_vec				
Description	Represents a tree through a vector with the following attributes 0. median in_tree depth 1. median out_tree depth 2. max in_tree depth 3. max out_tree depth 4. location / nodes ratio (whole tree) 5. median in_tree out_degree 6. median out_tree out_degree 7. max in_tree out_degree 8. max out_tree out_degree			
Input variable name	Description	Type	Optional	Default
tree	The tree to convert	treemob.PrefixTree	False	-
Returns	list: the vectorial representation of the tree			

treemob.tree_to_tfidf				
Description	Vectorises the users' trajectories dataset assigning to each location a weight corresponding to its importance in the user's personal mobility.			
Input variable name	Description	Type	Optional	Default
trajs	The trajectories of all the users	list	False	-
Returns	list: the vectorial representation of all the users			

treemob.tree_to_relative_tfidf				
Description	Vectorises the users' trajectories dataset assigning to each location a weight corresponding to its importance in the user's personal mobility.			
Input variable name	Description	Type	Optional	Default
trajs	The trajectories of all the users	list	False	-
Returns	list: the vectorial representation of all the users			

Acknoledgements

The first person I want to thank is of course Gigi, who always encouraged and supported me in all the things I like to do, and who doesn't judge me for my hearty breakfasts or if I spend a lot of time talking about programming and skateboarding.

The second thank is due to my family, that left me free to make my choices and to follow my passions, even if they are quite incomprehensible to them. An honourable mention to my grandparents who really love me.

A very special thank goes to my tutors Riccardo Guidotti and Salvatore Rinzivillo. Thank you because of a lot of reasons: first of all, for the time you dedicated to me, that was a lot, and always stimulating, but also because I felt that you really cared about my work and because you really helped me to give a shape to my intuitions.

I also want to thank the guys of oio studio, that believed in me and hired me while I was studying. I feel less guilty for having stalked you for about a year :) Also, I'm really happy with the community around us, which allows me to learn something new and cool every day. I'm looking forward to meeting you in person (also to verify you are real).

O course, I really want to thank the other guys, the ones of QZR, Arnaldo, Simone, Salvo, Luha, Lorenz, and again Gigi, who always welcome me in their studio, and give me a desktop and limitless coffee every day, in addition to a healthy dose of jingles and memes to be repeated in loop for an undefined amount of time. Salvo, I know you don't appreciate the sound of my 3D printer, but I don't care.

Then, there are Dani, Chiara, and Beatrice, from whom I have learned a lot, and with whom a cat picture is never inappropriate (but, honestly, is there any case where it is inappropriate?).

Another person I want to mention is Marta, my insider. She knows why I thank her *smirk*.

Now, a glance at my past: I want to thank all the people that during my three years at ISIA Urbino asked me to write simple programs or assemble Arduino circuits, and, before that, the ones who helped me to learn programming. Also, thank you again to Gigi, who prevented me to create a programming language for the BA thesis: without you, I would be working on it, yet.

Finally, I don't forget Wakame, that physically stayed very close to me every time I was working from home: I would not be surprised to find your fur in the digital version of this thesis too.