Visual Exploration of Large Multidimensional Trajectory Data

A. Telea M. Behrisch

Abstract

Several visualisation methods have been recently proposed to aid a wide variety of users in the exploration of geographical trajectory, or trail, datasets. Such datasets consist of thousands up to millions of spatiotemporal trails that are also attributed by many additional data variables related to the identity of the tracked items, type of motion being recorded, data provenance, and more. As both data size and data dimensionality grow, finding efficient and effective ways to answer concrete questions, as well as discover unknown insights, from such data become increasingly important. We present an overview of recent information visualisation and visual analytics developments in this direction, with the aim of bridging the gap between technical developments in this area and actual users and use-cases that can benefit from them. In this overview, we discuss strengths, limitations, assumptions, and other important characteristics of such visualisation methods, so as to help domain experts find optimal methods for their given application contexts. We illustrate our discussion with several examples of visualisation of large-scale, real-world, trajectory datasets related to migration data and use-cases.

1 Introduction

The last decade has witnessed the rapid increase of data sources concerning many societal aspects, including migration. Such data sources, *e.g.*, RPC's admissions and arrivals [Refugee Processing Center, 2020] or the Migration Data Portal [MDP, 2020], provide increasingly rich and diverse data concerning the origins, categories, amounts, and paths followed by migrating individuals and groups. As such portals collect and aggregate data from a variety of sources and types, they provide an alternative, and important, source for migration researchers, policymakers, and the grand public for studying and understanding migration-related phenomena [Bilsborow, 2016].

However, data availability is only one of the necessary ingredients to support insight forming and decision making. The other key ingredient is the availability of *tools* allowing stakeholders to clean, analyse, and present data in ways that support answering their questions and completing their tasks. Such tools include statistical analysis, data mining, and, last but not least, data *visualisation*.

Visualisation tools applicable to migration data cover a variety of approaches. As geographical attributes, such as origin and destination of migration flows and paths taken in between are key to migration data, many visualisation approaches use a map-based presentation to encode spatial information, and overlay this with additional attributes on demand, such as types, sizes, and time of migratory flows [Gapminder Org., 2020]. However, efficiently and effectively visualising large amounts of migration data in this spatial metaphor is challenging. One of the key issues here is the *relational* nature of such data, which relates origins to destinations by paths. Displaying large datasets containing thousands of such paths or more, each one potentially annotated with multiple attributes, can easily create a high amount of clutter, which makes analysis hard or even impossible.

This chapter provides a practical overview of visualisation methods, techniques, and workflows for the exploration of large trajectory datasets such as present in migration data. The main aim is to provide data scientists and, at a wider level, researchers interested in studying migration data who do not have a visualisation background, with guidelines on how to choose and use existing visualisation tools and

techniques to study their trajectory-centric data. When the practitioner involved with migration data has a better understanding of the process of preparing and creating visualizations, he or she will be in a better position to understand the shortcomings, biases, and ways of deceiving the viewer by these visualizations – and thereby be able to avoid such issues.

This chapter is structured as follows. Section 2 introduces a generic model to store and manipulate migration-related – or more generally, trajectory-based – data. Next, the same section introduces several classes of visualisation methods for such data, outlining their advantages and limitations for specific tasks. After having discussed data and tools, Section 3 details the typical workflow that the data scientist follows from the moment when a new data source, or dataset, is made available up to and including the iterative and interactive exploration of created visualisations that is used to answer actual questions on the data. Section 4 unifies the discussion on data modelling, tools, and visual exploration outlining key open challenges related to the study of trajectory-based data, and also sketching directions for further reading and research.

2 Background

Designing and validating visualisations follows typically a so-called nested model [Munzner, 2009] consisting of four steps: (1) The *domain* problem is characterised by eliciting the terms and high-level tasks that are commonly used by specialists in the respective field (migration study, in our case). (2) The data that describes these tasks and terms are abstracted into a so-called *data model*. (3) The data and tasks elicited during the previous steps are mapped to *visual encodings*, i.e., mapping *data* to visual shapes that are to be rendered on the screen, and interaction design, i.e., mapping *tasks* to interactive operations that one can execute on the visual shapes. (4) The visual and interaction designs proposed in the previous step are subsequently implemented in concrete software *tools*.

Constructing visualisations ideally follows all the steps (1-4), which are executed iteratively several times so as to refine the understanding of the users' needs and thereby the creation of visualisations that optimally address these needs. Doing this, however, requires considerable amounts of effort and specialised knowledge, both in the problem domain and in visualisation, computer graphics, and interaction programming. Moreover, such paths require a *concrete*, specific problem with particular users and their needs. Detailing the design of such specific visualisations is possible, but less interesting for the wider public.

In this chapter, we aim to provide actionable knowledge to *non-specialist* users interested in understanding the *types* of visualisations that can help the study of migration data in general, as opposed to the design of *custom* visualisations for a specific problem. Moreover, we focus here on migration data that involves spatiotemporal *trails*. This data is the least supported by generic visualisation tools known to a wider public, and thus where our discussion of more advanced visualisation tools is of greatest added value. As such, we next only cover a subset of the entire four-step nested model, as follows. Section 2.1 outlines the problem domain, listing generic questions and tasks that users of trail-centric migration data want to address. Section 2.2 proposes a simple but generic model for representing trail-centric migration data that covers most use-cases from the problem domain. Section 2.3 presents several types of visualisation techniques that can handle the aforementioned data model, focusing mainly on techniques that are well proven and battle-tested within the visualisation community and which are supported by open-source implementations.

2.1 Problem domain

Characterising a problem domain for visualisation design starts by identifying the entities that are to be analysed [Munzner, 2009]. For migration data, these include origins and destination locations, *e.g.*, countries, migration time moments, gross migration amounts, and migrant population characteristics, *e.g.*,

profession, age group, or refugee status. Consequently, tasks imply answering questions revolving around these entities.

As outlined in Sec. 1, one of the particularly challenging aspects of migration data is that it is *spatial* and *relational* in nature – that is, it inherently consists of multiple spatial locations being *linked* by multi-attribute migratory flows. As such, we next focus only on tasks that take into account this relational nature. Other tasks, *e.g.*, that consider the migration data as a set of tables listing the values of entities to be analysed, are far simpler to address by using established charting tools for tabular data, *e.g.*, Tableau [Tableau Inc., 2020] or Google Charts [Google Inc., 2020], and are thus not discussed in this chapter. Specific questions that visualisation addresses related to migration data are outlined below:

- **Q1:** Where (between which origins and destinations) are the strongest migration flows?
- **Q1a:** Which attributes, e.g., trail characteristics, contribute to the success of a migration trail? And, how can migration flows be steered?
- **Q2:** Are there similar migration patterns over different space and/or time intervals?
- **Q2a:** Can we extrapolate root causes that lead to these migration flows at a specific point in time?
- **Q3:** Are there migration patterns having a specific structure, *e.g.*, migration from a country to a large set of neighbour countries?
- **Q4:** Is there a reversal of migration over specific geographical regions and, if so, during which time periods?
- **Q5:** Which reasons exist for emigrating from a specific origin? Which reasons exist for immigrating to a specific destination?

Figure 1 illustrates the above challenge of visualising trail-based data. We consider here a simple dataset that records the number of refugees by *origin* (leaving a country) and by *destination* (entering a country), for all countries, for the year 2000. Images (a) and (b) show these numbers, visualised using GapMinder [Gapminder Org., 2020], colour-coded on a blue-to-red colormap. For reference, the country populations are encoded by the disk sizes. The overall technique is known as a bubble chart. These images allow us to see some patterns, *e.g.*, that Europe and North America receive a significant share of refugees (warm colours, image (b)); Afghanistan has the highest refugees leaving it (small red dot in the center of image (a)); and Iran and Pakistan are the highest receivers of refugees (small red dots in the center of image (b)). However, we cannot, for instance, see where to the Afghan refugees are going or where come from the ones arriving in Iran and Pakistan. On a higher level, we cannot see any *displacement* patterns indicating actual flows of refugees between different parts of the world. Image (c) shows actual refugee flows: Origin-destination (OD) flows above a certain person count (selected by the user so as to diminish visual clutter) are drawn as straight lines linking the respective countries' centers, and coloured using a red (origin) to green (destination) gradient [Boyandin et al., 2010]. Some additional patterns become immediately visible, such as the net influx of refugees in Europe and North America (green lines ending there). We also see that these lines originate mainly from Central and South America, respectively Africa, the Middle East, and Central Asia. Image (d) further simplifies the visualisation by reducing visual clutter by 'bundling' spatially-close trails. This creates a more schematic view of the migration flows which shows more clearly the two main refugee streams arriving in North America and Canada, and also that the majority of refugees that leave Africa arrive in Northwest Europe. While kept simple on purpose, Figure 1 already illustrates both the difficulty, but also the added-value, of visualising migration trails as opposed to simple per-country aggregates.

From the above simple example, we see a few specific aspects of the questions that visualisation aims to address. First and foremost, these are less of a quantitative nature, but more of a qualitative nature.

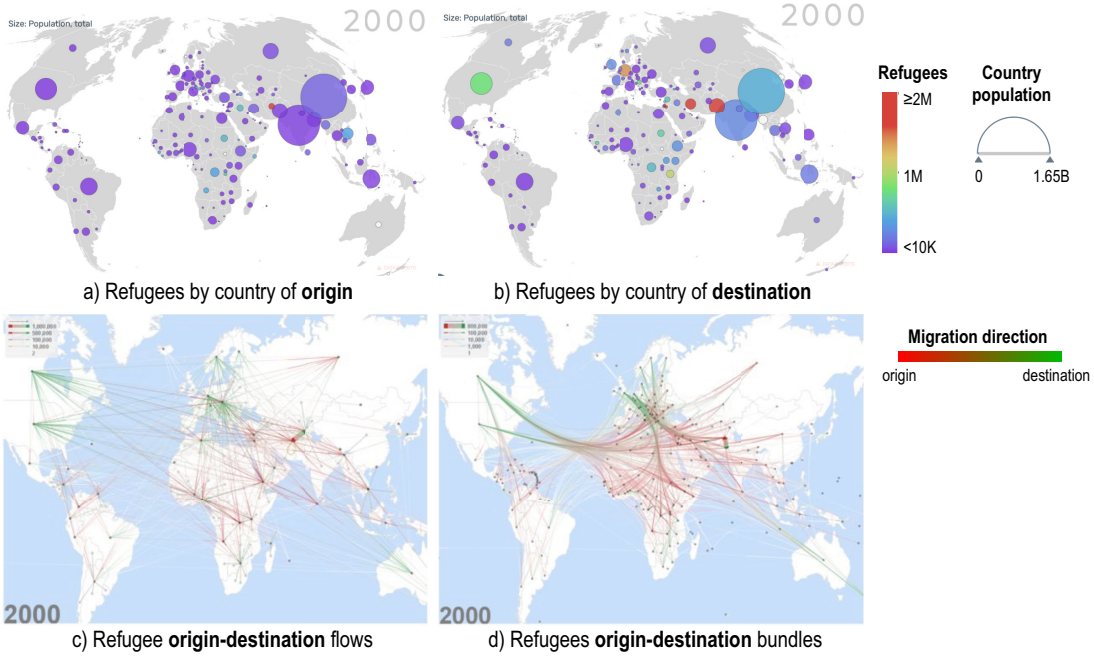


Figure 1: Refugee movements between origin and destination (asylum) countries in 2000 visualised with four methods. a,b) Refugee counts per origin, respectively destination countries shown in GapMinder [Gapminder Org., 2020]. c) Refugee flows depicted as straight lines (red=origin, green=destination) in JFlowMap [Boyandin, 2010]. d) Same image as c) but with simplified by trail bundling. See Sec. 2.1.

Indeed, questions that relate *strictly* to attribute values, *e.g.*, finding the range or average of a migration flow, or the point in time where a migration flow has peaked, can be accomplished using standard database tools, and benefit (far) less from visualisation. Simple visual depictions such as plain-text tables or bar charts can already support answering such questions. Conversely, questions which involve *multiple* data aspects (attributes), some of which are of spatiotemporal nature, and questions whose answers are not single values, but rather spatiotemporal *patterns*, are naturally better served by visualisation. For example, the easiest way to convey what a distribution looks like, is actually to *draw* it – especially in the case that its nature cannot be easily captured by a simple mathematical model. The qualitative aspect of the above questions – as opposed to quantitative aspects – is outlined by the presence of keywords such as 'where' (requires the description of potentially multiple locations), 'similar' (requires the description of multiple aspects that make two phenomena alike), 'patterns' (requires the description of multiple data aspects which, when occurring in a certain proportion, cause the appearance of what one calls a pattern), and 'structure' (requires the description of relationships between specific parts of one or several patterns). Other aspects that typically signal the qualitative aspect of questions – which, next, is best approached by visualisation – are the presence of descriptive terms which cannot be measured precisely on a quantitative scale, such as 'strong', 'scattered', 'important', or 'salient'.

A second point common to all the questions listed above is that they all relate to so-called indicators of mobility. Visualisation aims to answer questions concerning such indicators by encoding them into the attributes of the visualisation. For instance, one can color-code two instances of a country map by unemployment rates, respectively by with immigrant numbers, and thereby support answering questions concerning the correlation of the two indicators. We further detail how visualisation relates to questions concerning mobility indicators in the next two sections.

2.2 Data model

Preliminaries: Migration scholars use various forms of representing their data, such as a wealth of indicators to characterise migration and mobility, *e.g.* events, economic indicators, socio-cultural indicators, and geographic and climate-related data. In visualisation, data is typically represented by means of generic models that aim to capture its structure, typically in an as application-independent way as possible (to foster reusability of the developed visualisation techniques). To ease the learning task of the migration scholar or practitioner interested in (re)using visualisation techniques, we follow next the visualisation terminology of data modeling, thereby introducing and explaining terms and notations that such practitioners will likely encounter, and need to understand, when using visualisation tools.

Let $\mathbf{o}_i \in \mathbb{R}^2$ and $\mathbf{d}_i \in \mathbb{R}^2$ be pairs of points denoting the origin (start), respectively destination (end) points of a *journey*. By journey, we mean here the displacement, over a geographical map, of an entity (person, vehicle, or other object carrying information). Let $\mathbf{p}_i \subset \mathbb{R}^2$ denote the *path* being followed by this entity from origin to destination. Such paths are also called Origin-Destination (OD) paths. We distinguish two path types: *Straight-line* paths record only the tuple $(\mathbf{o}_i, \mathbf{d}_i)$, *i.e.*, provide no information on how and where the actual motion occurred. *Trails* record the actual position of the entity over time over its journey as a sequence $\mathbf{x}(t_i^0), \ldots, \mathbf{x}(t_i^N)$ of points $\mathbf{x} \in \mathbb{R}^2$ recorded at consecutive time instants t_i^j. Besides spatial information, OD paths typically also include other data attributes: Per-path attributes $\mathbf{a}(\mathbf{p}_i)$ are values associated to an entire path $\mathbf{p}_i$, *e.g.*, identity of the vehicle, type of vehicle, or cargo weight. Trail-based attributes $\mathbf{a}(\mathbf{x}_i^j)$ are values associated to actual sample points $\mathbf{x}_i^j = \mathbf{x}(t_i^j)$ along a trail, *e.g.*, speed and flying altitude of an aircraft. Attributes are most conveniently stored as (key, value) pairs, which allows different items (trails, sample points) to have different sets of attributes. Putting it all together, let $OD = \{\mathbf{o}_i, \mathbf{d}_i\}$ be the set of OD pairs under consideration; and let P be the set of straight-line paths or trails. An entire migration dataset is, thus, the tuple $D = (OD, P)$.

Figure 2a shows a simple schema (using UML-like notation) that allows storing migration data. Such a schema can be easily implemented using relational databases [Hoffman, 2003] or even plain-text file formats [Hurter et al., 2012].

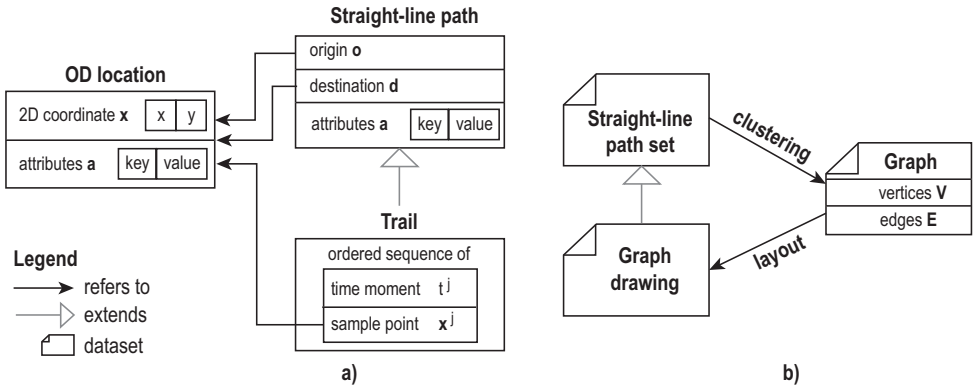
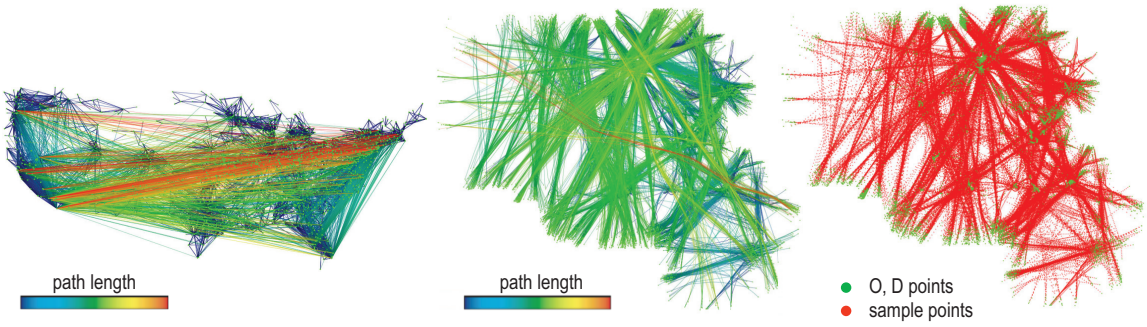


Figure 2: a) Schema for storing migration data. b) Relation of path-sets, graphs, and graph drawings.

Paths *vs* trails: Straight-line paths and trails typically co-exist in migration data visualisation and serve different purposes. Straight-line paths are necessary when one does not avail of actual location measurements along a trail, but only knows the O and D data; or when one wants to perform the analysis at a higher, more abstract, or more aggregated level. In contrast, trails allow finer-grained analyses, and are necessary when the underlying questions target actual motion patterns. Figure 3 shows examples of these two data types. Image (a) shows straight-line paths for a dataset of US migra-

tions [Holten and van Wijk, 2009]. Every path (9780 in total) shows the migration (relocation) of one person from one city (O) to another city (D) in the US. Paths are colour-coded by their length, to help understanding the visualisation. Image (b) shows a trail dataset containing 5255 civil-aircraft trajectories recorded by Air Traffic Control (ATC) over the French airspace during one week, also colour-coded by trail length [Hurter et al., 2014]. Trails have between 50 and 200 sample points. Image (c) shows the actual O and D points for these trails (green) and their sample points (red).



a) straight-line paths (US migrations) b) trails (French air traffic) c) O, D, and sample points (French air traffic)

Figure 3: Examples of straight-line path data (a) and trail data (b,c) [2]. Figures generated with the open-source CUBu software [van der Zwan et al., 2016].

Modelling time: The data model presented so far can integrate *time* in two different ways. *Sequence* models store consecutive snapshots D_i of the dataset D, recorded at different time instants i. Sequence models are used when one only avails of the recording time i of an entire dataset D_i – for example, we have several datasets as the one in Figure 3b recorded for several weeks over a year. They only allow comparing *global* movement patterns between two or more time moments. *Streaming* models store a single dataset D, in which the OD points of each path or trail $\mathbf{p}_i \in P$ have a time attribute. The lifetime of path $\mathbf{p}_i$ is thus the time interval $[t_i^0, t_i^N]$. Streaming models are used when one avails of specific recording times for each individual path or trail. When available, they allow finer-grained comparison of motion over different space and/or time ranges. In the following we will cover both time-independent data and streaming and sequence models, outlining visualisation techniques suitable for each of these data types. For a more detailed discussion on visualising temporal multivariate data, we refer to [Archambault et al., 2013].

Graphs, networks, trails: Migration data, and more generally trajectory data, is interchangeably referred to in different sources in the visualisation literature using the terms graphs, networks, and trails [von Landesberger et al., 2011, Archambault et al., 2013, Lhuillier et al., 2017]. Clarifying these terms and how they relate to each other is important, since some visualisation techniques (discussed next) are applicable to only certain data types (see also Figure 2b):

- *Trail* data has been described in the 'Paths *vs* trails' comparison. It models trajectories, or paths, of entities over (typically) two-dimensional Euclidean space; Straight-line OD paths are particular instances of trails containing only two points – the origin (O) and destination (D);
- *Graph* data models abstract relations (also called edges) between node pairs. Formally, $G = (V, E)$, with V being a vertex set and $E \subset V \times V$ being an edge set. Graphs are visualised by graph *drawings*, which are created by a so-called graph *layout* process. Graph drawings are also called

[2] The figures in this chapter are best viewed in full color.

node-link visualisations. A graph drawing can be using straight lines or curves, which corresponds to straight-line OD path sets, respectively trail sets;

- Graphs can be *constructed* from OD sets by clustering spatially close O, respectively, D points. Each point cluster c creates a vertex $\mathbf{v} \in V$. Paths linking two clusters c_1 and c_2 create an edge $\mathbf{e} \in E$ linking the clusters' respective vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ from V;
- *Networks* are, in visualisation, typically used as a synonym for the more theoretical term graphs.

2.3 Visualisation techniques

Arguably the key challenge when visualising geographic movement data is caused by the *size* of the dataset D. When this is too large – roughly, over a few hundred elements – *clutter* occurs due to the many overlapping and/or intersecting paths, as already visible in Figure 3a,b. Clutter further impedes accomplishing even basic tasks using the visualisation. As such, visualisation methods aim to reduce such clutter by various mechanisms, as follows.

Aggregating methods: These methods do not attempt to directly draw a dataset D. Rather, they simplify D into a dataset D' containing (far) fewer OD pairs and paths. When D' is under a certain size – typically a few tens or hundreds of items – it can be directly drawn with limited clutter, for example using straight-line drawing techniques (discussed next). Note that using straight-line drawing is not mandatory: When D' is small, custom techniques can be used to further reduce clutter, by routing (bending) the drawn paths to minimise overlap and/or intersection. Historically, this has been first done by hand-drawing the paths in the simplified path-set P' of D'. Figure 4a-d shows four examples of such visualisations created by the French cartographer Minard [Minard, 2020]. This design, where the width of the curved paths is used to encode a path attribute, is also known under the name *flow maps* or *Sankey diagrams*. Image (a) shows a single-trail flow map depicting Hannibal's advance into Gaul and Italy, with path width encoding Hannibal's army size. Image (b) refines this design to show Napoleon's campaign in Russia[3].

As no accurate location data is available, consecutive OD points are linked now by straight-line segments. colour encodes the advance (brown) *vs* retreat (black) paths. Image (c) generalises this design to show the single-origin, multiple-destination map of French wine exports, with export volume encoded in path width. In contrast to designs (a) and (b), paths are now curved to minimize overdraw – so the actual path shapes do not actually encode geographical information, a design we will encounter further on. Finally, image (d) shows a multiple-origin, multiple-destination dataset of people migration in 1858, with path width mapping the number of migrants, and colour encoding the continent of origin. Figure 4e shows a recent hand-drawn flow map depicting the intra-European migrations of 2006 [Hossmann et al., 2008]. Compared to the earlier Minard maps (Figure 4a-d), this design is simpler, as it uses so-called orthogonal (vertical and/or horizontal) path directions only. This follows studies in map visualisation that showed that reading such orthogonal layouts, also called 'metro map layouts', is easier than following layouts using paths drawn at arbitrary angles and/or using variable-angle bends [Wolff, 2007, Nöllenburg, 2014].

Several automatic aggregation methods for a path or trail dataset D exist. The most widespread, and easiest to use, aggregate O and D locations based on spatial proximity, thereby replacing clusters of densely located O or D points by single points, typically the cluster centroids. The aggregation radius determines the simplification degree. Alternative techniques use generic clustering methods such as k means [Luo et al., 2017]. After obtaining the simplified dataset D', this can be depicted using standard graph drawing algorithms, which can carefully optimise the curving of paths to minimise clutter [Tamassia, 2013]. Several standard libraries and tools exist that allow non-specialists to create such

[3]While Minard's visualisation is technically impressive, the *insights* it conveys are doubtable. Minard created this visualisation in a period when Napoleon's rule was politically questionable in France, using data sources which highly exaggerated Napoleon's losses in Russia [Tufte, 2002]. For a historically more accurate rendition, see [de Caulaincourt, 1933].

high-quality graph drawings [Gansner, 2020, Auber, 2004, Auber, 2020].

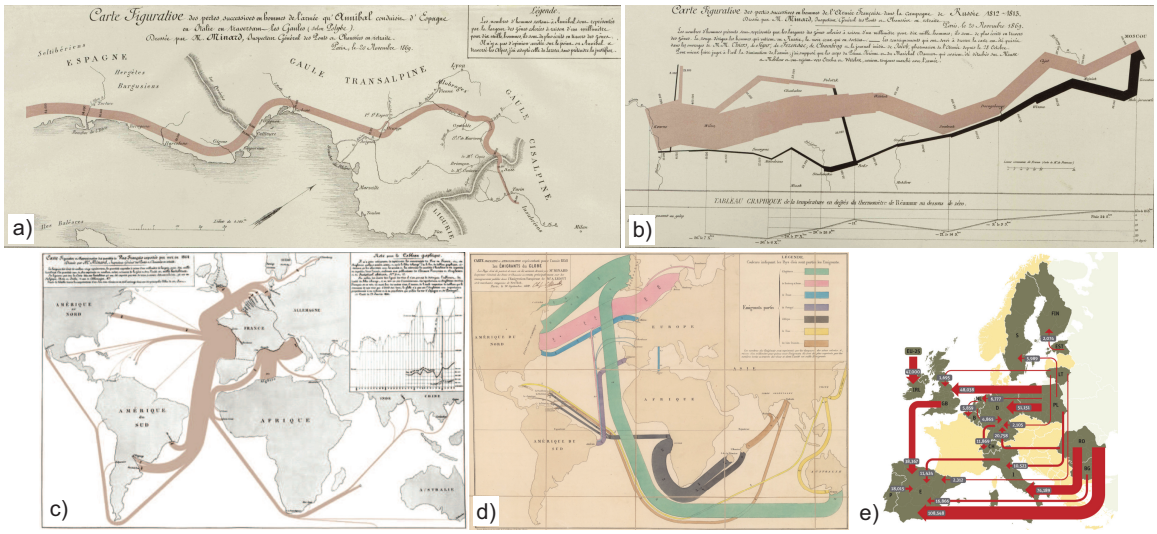


Figure 4: Aggregate visualisations of geographical movement data. Flow maps of a) Campaign of Hannibal (Minard, 1869). b) Campaign of Napoleon in Russia (Minard, 1869). c) French wine exports (Minard, 1864). d) Word migration map (Minard, 1862). e) Intra-European migrations in 2006 [Hossmann et al., 2008].

Density maps: These methods – also called heat maps – are motivated by the typical clutter created by directly drawing trails (Figure 3a). In contrast to the aggregating methods discussed above, they address this by aggregating the *drawing* of the dataset D rather than the actual *data* (trails). This is done by convolving the drawing of D with a Gaussian or Epanechnikov (parabolic) filter [van Liere and de Leeuw, 2003], a process known in image processing as Kernel Density Estimation (KDE) [Comaniciu and Meer, 2002]. The result is a density map that effectively merges trails closer than the width k of the filter, thereby simplifying the visualisation. Figure 5 shows this for the US migrations dataset in Figure 3a. Image (a) shows a naive computation of trail density, done by drawing the trails half-transparent. While dark regions indicate zones populated by more trails, close (but not exactly same-position) trails are not grouped together. Image (b) shows the same drawing as in (a), this time convolved (blurred) by a Gaussian filter. We see how close trails get visually merged into high-density zones. This effectively simplifies the visualisation in image space, the simplification level being given by the blurring filter radius. Image (c) shows the same density map as in (b), with density mapped to both colour and height. The dense group of trails that connects the Southwest to the Northeast of the US now becomes even more salient. Summarising, density maps address well Q1 (Sec. 2.1) and also remove small-scale clutter to create a simplified visualisation. For a formal discussion of KDE for trail visualisation, we further refer the reader to [Hurter, 2015].

Bundling methods: These methods share their motivation with the density maps described above, aiming to group similar trails to simplify the visualisation. In contrast to density maps, they however *deform* the trails to accomplish this. Given a trail set $D = (OD, P)$, bundling (1) finds trails $\mathbf{p}_i \in P$ that are similar to each other, and (2) deforms these so they become even closer spatially. Trail similarity is typically computed using *both* the trail positions $\mathbf{x}_i^j$ and trail attributes $\mathbf{a}(\mathbf{p}_i)$ and/or $\mathbf{a}(\mathbf{x}_i^j)$. Spatial trail similarity, *i.e.*, the term using the positions $\mathbf{x}^j$, is typically computed using Hausdorff distance [Rockafellar and Wetts, 2005, Lhuillier et al., 2017]. Attribute similarity is typically computed using Euclidean distance between the $\mathbf{a}(\mathbf{p}_i)$ values of different trails $\mathbf{p}_i$. The two terms (spatial and attribute) are typically merged by weighting [Telea and Ersoy, 2010].

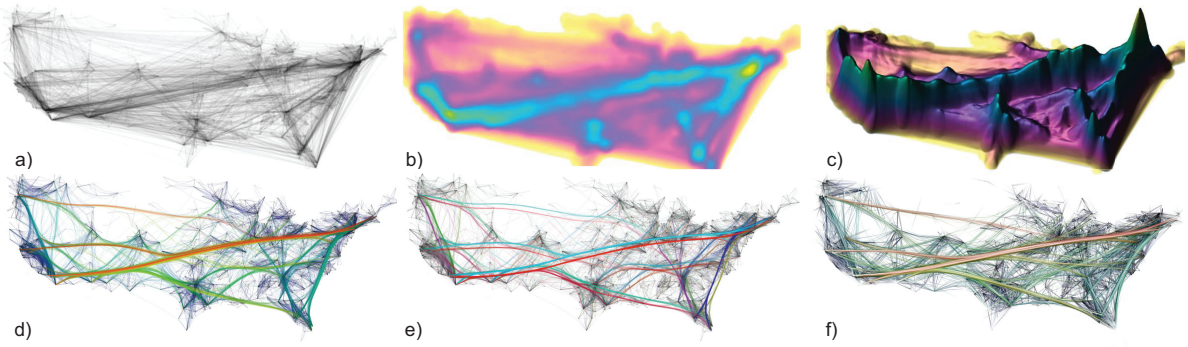


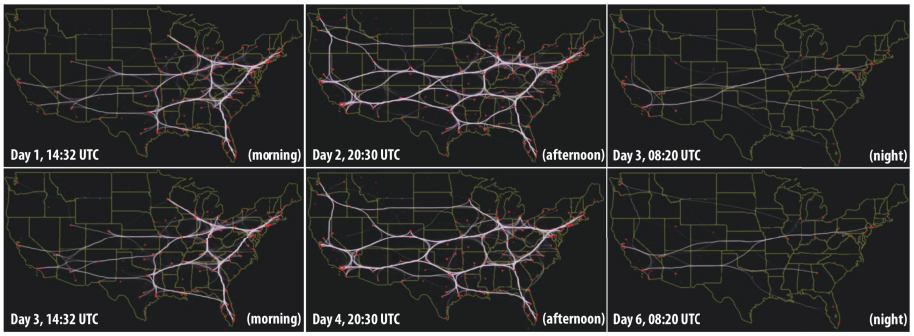
Figure 5: Visualisation of US migration dataset by several methods. From simple to involved: a) Blending of straight-line trails. b) colour-coded density map. c) Height and colour coded density map. d) Bundled trails using path-length colouring. e) Directional bundling. f) Pseudo-shading of bundles. Figures generated with the open-source CUBu software [van der Zwan et al., 2016].

Formally put, bundling is nothing but applying the mean shift aggregation principle, well known in image analysis [Comaniciu and Meer, 2002], to the drawing of trails. Intuitively put, bundling 'pulls' similar trails in the visualisation towards their local common centre, so that they emerge as compact groups separated by whitespace. This way, one can easily see the main flow patterns present in a trail dataset.

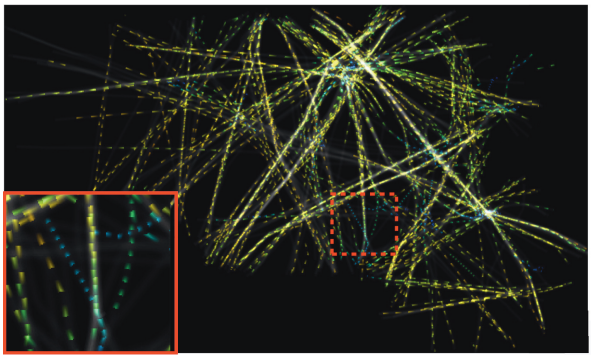
Tens of bundling methods exist in visualisation literature – for a recent survey, we refer to [Lhuillier et al., 2017]. From a practical end-user perspective, these differ mainly in terms of ease of use, computational speed, and ease of implementation. As such, we next highlight only those that we consider to be the most interesting for data scientists involved in migration studies (who are not experts in visualisation, and require easy-to-use, scalable, readily available, and predictable methods). From this perspective, two methods stand out: The Kernel Density Estimation Edge Bundling (KDEEB) method [Hurter et al., 2012] pioneered scalable and easy-to-use trail bundling. KDEEB is simple to explain: It computes a KDE density map of the trail set, and next advects (moves) trail sample points $\mathbf{x}^j$ upwards in the density gradient, until reaching its maximum, following the mean shift principle [Comaniciu and Meer, 2002]. The CUDA Universal Bundling (CUBu) method [van der Zwan et al., 2016] refines and generalises KDEEB to efficiently use graphics hardware (NVIDIA CUDA GPUs) to bundle millions of trails in under one second on modern computers, and also provides different bundling styles (subsuming most results of earlier methods) in a single implementation.

Figures 5d-f show three examples of trail bundling for the US migrations dataset created with CUBu (which subsumes KDEEB and earlier bundling methods). Image (d) colours trails by their length, similar to Figure 3a. We see here more clearly than in the unbundled image (Figure 3a) where the longest migration trails are located – Southwest to Northeast, coloured red. Also, spatial migration patterns become clearer than in Figure 3a – we see that most migrations happen on the horizontal East-West axis, except for the US coast, where salient North-South patterns exist. Image (b) shows the bundled trails separated by migration *direction*. That is, migration flows between the same O and D areas yield distinct, parallel, bundles. From this image, we see that migrations in both directions are balanced. Finally, image (f) shows the bundled migration trail-set with pseudo-shading, colour coded again by trail length. Bundles appear here as 3D tubes seen from above, thereby reducing interpretation problems when they cross (compare with Figure 5a). For instance, the Southwest-Northeast migration bundle appears more saliently in this image than in Figure 5a.

Techniques for time-dependent data: For visualising time-dependent migration data (either sequence or streaming, see Sec. 2.2, two main techniques classes exist. First, *small multiples* show separate visuali-



a) Six small-multiple snapshots from a visualization of US flights over a week.



b) Snapshot of a particle visualization of flights over Paris

Figure 6: Visualisation of dynamic motion data using (a) small multiples [Hurter et al., 2013] and (b) animation [Hurter et al., 2014]. Images generated by open source software (a) [Hurter et al., 2013] and (b) [van der Zwan et al., 2016].

sations of the snapshots D_i recorded at different time instants, side by side, using the same visualisation parameters, a design known in visualisation as *small multiples*. This way, users can compare the resulting images to spot (salient) differences and therefore infer (salient) changes. Figure 6(a) shows an example, where six snapshots $D_i, 1 \leq i \leq 6$ from a time-dependent dataset showing the travel of people using airlines in the US over six days is depicted using KDEEB bundling [Hurter et al., 2013] Comparing the snapshots shows how travel patterns change over time – for instance, we see mainly East-coast travel in the morning, travel over the entire map at noon, and mainly long-haul East-West flights during night, respectively. The key problem of small multiples is that it does not scale to more than a few time instants i. The alternative is to use *animation*, *i.e.*, depict how imaginary travellers 'flow' over the paths $\mathbf{p}_i \in P$ over time. This can be done by seeding all $\mathbf{p}_i$ with particles (points) and next animate these over the trajectories of $\mathbf{p}_i$, while at the same time show only paths whose lifetime $[t_i^0, t_i^N]$ encompasses the current (animation) time. Figure 6(b) shows a snapshot of such a particle visualisation depicting the air traffic over Paris at a given time moment in a streaming dataset [Hurter et al., 2014]. Animation demands less space than small multiples, but poses a higher burden on the user's memory to remember, and compare, spatial patterns occurring at different time moments. So far, the visualisation community advocates both small multiples and animation for showing time-dependent data, with no decisive arguments in favor, or against, one of these techniques. More examples of such techniques are given in [Scheepens et al., 2015]. Implementations of animation techniques are given in [Hurter, 2020].

Other visual encodings: So far, we have discussed visualisations based on the node-link metaphor. In this metaphor, several visual mappings are predefined: The coordinates of physical locations are mapped

to O and D points in the screen space $\mathbf{o}_i \in \mathbb{R}^2$, respectively $\mathbf{d}_i \in \mathbb{R}^2$, while the migration flow data attributes are mapped to one more multiple visual encodings applied to the paths or trails. Typical choices for these edge encodings are colour, thickness or transparency. Hence, node-link visualisations implicitly emphasise location relationships, making them an effective choice for contextualised displays, *e.g.*, maps.

However, also other visualisations for migration data exist, see Figure 7. Historically, these visualisations were designed to address some of the shortcomings of node-link metaphors. Ghoniem *et al.* [Ghoniem et al., 2004], for example, demonstrated that *matrix representations* outperform node-link ones for large or dense relational datasets on several graph analysis tasks. In our context, an (adjacency) matrix of a migration flow graph $G = (V, E)$ is a square matrix M where the cell m_{ij} captures information describing all edges $\mathbf{e}_k \in E$ between vertices $\mathbf{v}_i \in V$ and $\mathbf{v}_j \in V$. Hence, a matrix row or column depicts a node, while a matrix cell depicts edges between two given nodes. At the simplest level, $m_{ij} = 1$ indicates that at least such an edge exist, whereas $m_{ij} = 0$ tells that $\mathbf{v}_i$ and $\mathbf{v}_j$ are not directly connected in G. Figure 7a shows a simple (directed) graph and its equivalent matrix representation. Note that undirected graphs correspond to symmetric matrices. At a more refined level, m_{ij} can aggregate one, or even multiple, attributes $\mathbf{a}(\mathbf{e}_k)$ defined over all edges $\mathbf{e}_k$. Adjacency matrices shift the emphasis away from the spatial contextualisation of the nodes and paths – that is, one cannot use them to reason about the spatial location or relative position of nodes and edges. In contrast, matrices scale visually very well, as every edge-set $\mathbf{e}_k$ requires, in the limit, a single pixel to be shown. This allows graphs with thousands of nodes and millions of edges to be displayed on a typical computer screen. Also, matrices do not have any of the clutter and overdraw issues of node-link metaphors.

However, for path-related tasks, such as finding how any two vertices are connected (via paths formed by multiple edges), finding shortest routes, and also for contextualising the findings, matrix metaphors are significantly more demanding than node-link ones. Several visual extensions to the basic matrix metaphor aid this. Figure 7b shows one of these: Here, apart from showing edges in the matrix cells, these are drawn as curved arcs connecting the respective nodes both along rows and columns [Henry and Fekete, 2007]. The user can visually 'follow' a sequence of arcs to find paths that indirectly connect nodes. The MapTrix tool [Yang et al., 2017] re-embeds the geographic (spatial) context of migration flows into matrix displays (Figure 7c). This is done by connecting a matrix display (right on the figure) with a classical flow map display [Rae, 2009] (left in the figure). The connecting lines (gray in Figure 7c) link each row and column (node in the matrix display) with its geographic location on a map, and can also show additional path attributes. OD maps [Wood et al., 2010] (Figure 7d) show another way to preserve spatial context. In this visualisation, the map is subdivided by a grid. For each grid cell, a heat map shows the density of origins (O) of all flows ending in that cell. This way, users can compare patterns of incoming flows between the grid cells. The heat maps are displayed into their respective cells, thereby conveying information on the destinations (D).

Other methods target depicting additional, potentially time-dependent, attributes of flows. For example, Space-cuts [Buchmüller et al., 2016] (Figure 7e) distorts the geographic maps by artificially introducing cuts along spatial landmarks, such as streets, rail tracks, or rivers. Flow attributes are shown by rendering them in the space created by the cuts. Contextualised glyph designs [Sun et al., 2017] and interactive lenses [Krüger et al., 2013] are additional solutions to the data scalability problem. However, these solutions are technically more complex to implement and learn to use.

3 Creating visualisations

Key questions related to the techniques outlined in Sec. 2.3 are: How easy is it to create such visualisations for the non-technical user, and how should she proceed to do this? We aim to answer these questions next by detailing the steps of creating an end-to-end visualisation pipeline, from having a data source up to the visual exploration design.

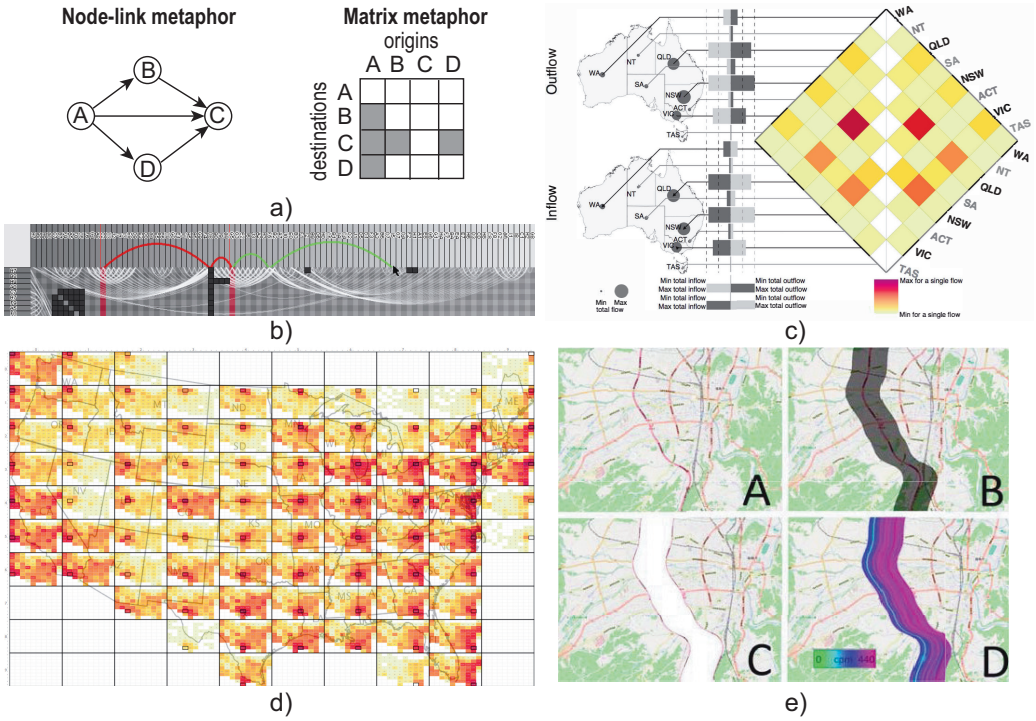


Figure 7: Other visualisation types for migration data. (a) Matrix metaphor. (b) MatLink [Yang et al., 2017]. (c) MapTrix [Yang et al., 2017]. (d) OD Maps [Wood et al., 2010]. (e) SpaceCuts [Buchmüller et al., 2016].

3.1 Data collection and curing

The first step, and arguably one of the most laborious, in constructing good visualisations is obtaining good *data*. By this, we mean a dataset D that strictly follows a variant of the schema in Figure 2. Obtaining such a D poses several challenges. Some major ones – including solutions for them – are as follows.

Attributes: The general schema discussed in Sec. 2.2 (Figure 2) poses no constraints on path attributes: Every path $\mathbf{p}_i$, and even every trail point $\mathbf{x}_i^j$, can have a variable number of attributes $\mathbf{a}$)($\mathbf{p}_i$), respectively $\mathbf{a}(\mathbf{x}_i^j)$. To handle such attributes, a *regularisation* pass is needed: First, for all attribute values for the same key, the *types* of all attribute-values $\mathbf{a}$ are found, scanning all their values over D, by examining their actual values. Based on their frequency, attributes $\mathbf{a}$ are typically classified as *quantitative* (real-valued numbers), *integral* (integer values), *ordinal* (values that allow ordering but whose absolute values do not capture extra semantics), and *categorical* (values which indicate different classes of objects). These types can be further refined, *e.g.*, categorical values can be split into plain text or URLs. This step is key to the subsequent aggregation and visualisation of attributes. After this step, every value $\mathbf{a}$ for a path or sample point (Figure 2) will have an associated *type*.

Normalisation: Comparing attributes of different types (determined in the previous step), *e.g.*, to compute path or trail similarity further needed for simplification (Sec. 2.2) requires normalising them. For numerical attributes, this is typically done by standardisation, *i.e.*, replacing every value $\mathbf{a}$ by $(\mathbf{a}-\overline{\mathbf{a}})/\sigma(\mathbf{a})$, where $\overline{\mathbf{a}}$ is the attribute's average, and $\sigma(\mathbf{a})$ is the attribute's standard deviation, over all its values in D.

Separately, handling trails $\mathbf{p}_i$ typically requires resampling these so that the spatial density of points $\mathbf{x}_i^j$ is roughly uniform over D. This is typically done by linear resampling $\mathbf{p}_i$, which involves also (typically linear) interpolation of the attributes $\mathbf{a}(\mathbf{x}_i^j)$.

Values: Real-world datasets often come with *missing* or *incorrect* values for the attributes $\mathbf{a}$. These need sorting out, since virtually all visualisation methods require consistent attribute-sets for all their samples $\mathbf{p}_i$ and/or $\mathbf{x}_i^j$. When such values are missing, they are usually replaced – in a process known as value imputation – by *averages* over the entire set of attribute values $\{\mathbf{a}(\mathbf{x})|\mathbf{x} \in D\}$ or by special 'undefined' values, if the attribute-type of $\mathbf{a}$ allows this. Incorrect values are treated similarly, *i.e.*, detected based on comparison with the expected range of $\mathbf{a}$, and replaced by averages or defaults if non conforming.

Normalisation and value imputation are two important sources of bias in interpreting migration and mobility data. When these operations are used – prior to creating visualisations – they should, next, be reported in the actual visualisation, *e.g.*, by means of suitable legends or captions.

3.2 Data simplification and filtering

Simplification and filtering are interchangeable terms for two operations: (1) Given a dataset D, how to reduce its amount of *sample points* $\mathbf{x}$ (size reduction); and (2) how to reduce its amount of *attributes* $\mathbf{a}$ (attribute reduction). These two simplification directions are orthogonal, and treated as follows.

Size reduction: This *speeds up* the creation and execution of visualisations, since fewer data items need to be drawn. Also, for node-link displays, it reduces clutter. More generally, size reduction allows the user to focus on the main, coarse-scale, patterns present in the data. Size reduction can be done by two main mechanisms. *Selection* picks a subset $D' \subset D$ of the data elements to be explored, based *e.g.* on specific values of attributes of interest, *e.g.*, migration flows starting from a given country, connecting two given areas, or taking place in a specific time period. Selection works well when one knows *in advance* which are the subsets of interest D' and is extremely easy to implement. Figure 6a is an example of selection, as it shows six subsets D' of the entire time-dependent dataset D, selected based on time ranges. *Aggregation*, in contrast, replaces D by a new dataset $\overline{D}$ by replacing elements (paths, origins, destinations) of D that are deemed similar to aggregate versions thereof. The simplest form of aggregation is averaging. For example, the hand-drawn visualisations in Figure 4 are obtained this way. Here, the designer has manually grouped all flows between locations of interest. Aggregation can be also done automatically for graph data with tools such as LGC [Fountoulakis et al., 2018] and Tulip [Auber, 2020] and for table-based data with tools such as Tableau [Tableau Inc., 2020]. Aggregation does not require the user to select a specific subset of the data. However, it requires selecting a suitable *level of simplification* for the entire dataset.

Attribute reduction: This *simplifies* the visualisation creation, since less data-per-item needs to be drawn. As for size reduction, this can be done by selection or aggregation. Selection picks a few attributes $\mathbf{a}$ from the entire available set, typically based on their keys (Sec. 2.2), and encodes these into different visual channels, such as size, colour, transparency, and position. For example, Figure 5d shows the US migration dataset with colour encoding trail length, and position encoding O and D coordinates, respectively. Since such visual channels are not independent, typically no more than three up to four different attributes $\mathbf{a}$ can be visualised simultaneously. Aggregation replaces subsets of attributes $\mathbf{a}$ by a single attribute, again using suitable methods, such as averaging. This is more challenging to do than aggregation for size reduction, since now the attributes to be aggregated can be of different *types* having also different *ranges*. It is hard to come up with generic guidelines on how to do attribute aggregation. However, a good starting point for practitioners is examining standardisation and one-hot-encoding techniques that are used since long in multidimensional data analysis. A good introductory textbook on this topic is [Jolliffe, 2002].

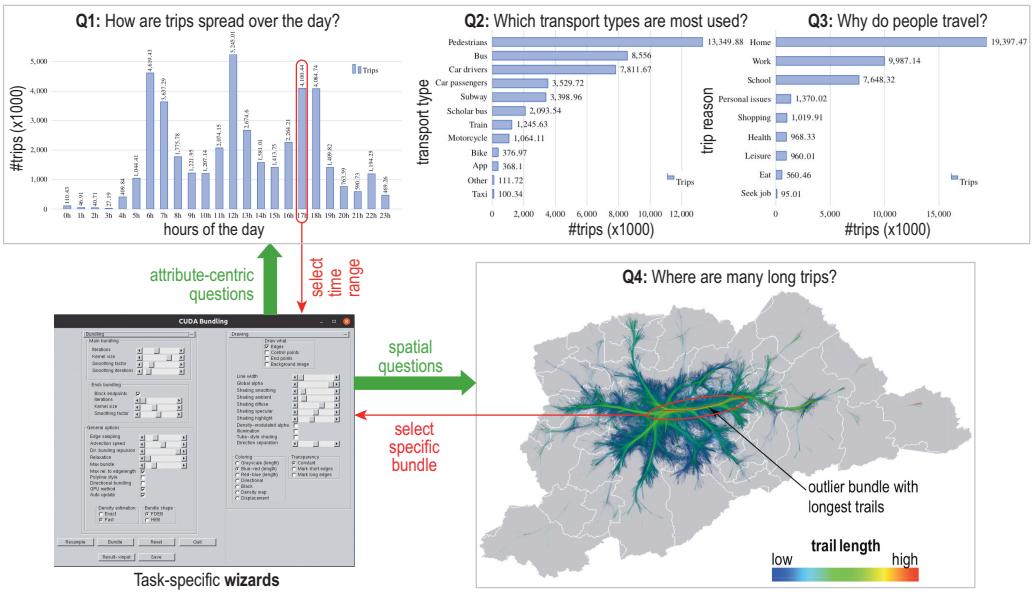


Figure 8: Visual analytics workflow for exploring commuter flow in the greater São Paulo area [Martins et al., 2020] constructed using the CUBu open-source tool [van der Zwan et al., 2016].

3.3 Designing the visual exploration

Having the suitably cleaned and selected data, the final step is to choose a suitable *set* of visual exploration techniques. By set, we mean here the fact that an effective visualisation never consists of a single, static, image that depicts data. Rather, several techniques are combined, via user interaction, to allow one to *explore* the data and answer specific questions or complete specific tasks. Designing an effective visualisation is a complex process. Nevertheless, several general and well-tested guidelines can be given for this, as follows.

Overview, zoom and filter, then details on demand: This concept, known also under the name of 'Shneiderman's visual exploration mantra' [Shneiderman, 1996], is almost invariably used by all visualisation tools. Since users typically do not know *where* to start their exploration, the visualisation starts by presenting a global overview of the entire dataset, computed *e.g.* using aggregation techniques (Sec. 3.2). This helps showing interesting spatial patterns, into which the user zooms; alternatively, one can filter out uninteresting data aspects from the overview to simplify its exploration. Next, the user selects the patterns of interest, and examines these in more detail. The process is typically iterated until the questions of interest are answered. Technically, this requires designing visualisations which (a) are tightly coupled with data selection and aggregation mechanisms; and (b) which allow spatial zooming-and-panning.

Visual analytics loop: At a higher level, Shneiderman's design enables the creation of so-called *visual analytics* (VA) solutions for exploring data. Simply put, these are visualisation-and-data-processing tools whose user interaction options (GUIs, direct manipulation) are designed so as to best reflect the user's typical *workflow*. That is, rather than offering all options to the user in a 'flat' GUI (which is confusing, since one does not know then which options to use and in which order), options are grouped in wizard-like GUI designs that address specific tasks. The user then selects one such wizard to start the data exploration. The obtained insights allow her to form a *hypothesis* related to the data. Next, other wizards are used to examine the (subsets of the) data in detail to confirm, infirm, or refine the hypothesis. The process loops

until one arrives at a confirmed hypothesis, or simpler put, answers to one's original questions.

Figure 8 shows the workflow of such a VA tool designed to explore the urban mobility data from the greater São Paulo (SP) area in Brazil [Martins et al., 2020]. The input dataset D contains over 42 million trips of commuting people in SP over a single day in 2017. Every trip is an OD path annotated with hour of travel, means of transportation (*e.g.*, by foot, bus, train, bike, car, or others), and trip reason (work, school, shopping, seek jobs, and others). Full details of this dataset, collected since 1967 by surveys, are given in [Metrô SP, 2018].

Mobility researchers in SP, including municipalities, want to understand the travel patterns to optimise transport. This implies answering a (wide) set of questions. VA can help here, as follows. The analyst loads the dataset D and first produces a number of aggregated charts showing the distributions or attributes a of D over the dataset. These simple, attribute-centric, questions can be readily answered by histogram bar charts, like those shown in Figure 8(top). There, three basic questions are answered: How are trips spread over the day (Q1); Which transport types are mostly used (Q2); and Why do people travel? The respective three histograms show that the answers to these questions are, respectively, 'around 6AM, 12AM, and 12AM-6PM' (which would confirm the standard hypothesis related to work start, lunch break, and work end in Brazil); "people mostly walk, then go by bus, then by car" (which is good to know since public transportation scores second-high, above private cars); and "people travel most to go home, go to work, and go to school" (which can provide hints as to how one can help improving overall transportation).

Apart from these attribute-centric questions, one can also do spatial questions. A key one is: How are trips spread over the SP area? Figure 8(bottom-right) shows this by using trail bundling [van der Zwan et al., 2016]. Here, trips are coloured by length, and opacity encodes bundled-trip density. We see a red outlier in the middle, signifying *many long trips* that go from West SP to East SP or conversely. Since opacity encodes trip density, the fact that a large part of the SP area is empty (gray) means that there are very few trips spanning peripheral regions. The 'core' of travel is within SP central, and the highest outlier (bottleneck) is the aforementioned West-*vs*-East trajectory. Hence, if planners want to improve the situation, they should focus on this trajectory.

Given these global insights, users can next select *subsets* of the data, *e.g.*, hours of the day, or particular trails. These are marked in Figure 8 in red. After selection, the VA process loops, but now only on the selected data, allowing users to pose more questions to understand *why* these events occur. Questions – thus, the creation of specific visualisations – are done by the GUI *wizards* (Figure 8 bottom-left). The entire VA lop (queries, marked green, followed by selections, marked red in Figure 8, followed by inspecting the newly created visualisations) repeats until one is satisfied with the obtained answers.

This VA example is, obviously, a simple one; for space constraints, we cannot refer to all existing options [van der Zwan et al., 2016, Martins et al., 2020]. Still, it captures the *essence* of designing VA solutions for mobility exploration – visualisations, ranging from simple/aggregated, to detailed ones, driven by user selection of data based on visualisation insights: The VA loop.

3.4 Putting it all together

A key challenge in implementing (interactive) visualisation systems is the availability of software tools. Visualisation tools are notoriously hard to replicate and/or implement, as underlying technology spans fields as diverse as algorithms and data structures, data mining and querying, computer graphics, image processing, computer vision, interactive techniques, and user interface design [Childs et al., 2013]. To assist the reader, Table 1 gives a few pointers to established freely available software tools that implement techniques discussed earlier in this chapter, indicating also how the software is available (open source or licence-based model). The rightmost column indicates the skill-set expected to use the software, *i.e.*, if it targets any (A) users or users with programming (P) skills.

[4]Tableau offers a free academic licence model.

Table 1: Software for processing and visualising (multivariate) motion data.

Name	Reference (including URL)	Functionality	Availability	Skills
GraphViz	[Gansner, 2020]	Small graph layout and rendering	open source	A
Tulip	[Auber, 2020]	large graph interactive visualisation	open source	A
KDEEB	[Hurter et al., 2012]	Bundling spatial trails (basic)	open source	P
CUBu	[van der Zwan et al., 2016]	Bundling spatial trails (extended)	open source	P
Particles	[Hurter et al., 2014]	Animating particles along spatial trails	open source	P
Local Graph Clustering	[Fountoulakis et al., 2018]	Graph simplification	open source	P
Sankey Diagrams	[Open Source, 2020]	Drawing Sankey diagrams	open source	A
Tableau	[Tableau Inc., 2020]	Data cleaning, selection, aggregation	commercial[4]	A

4 Discussion and Conclusion

We have presented a roadmap for selecting and implementing visualisation solutions for exploring multidimensional trail data, such as describing the motion of persons over space and time. We have outlined how to represent such data, which are the main types of visualisation methods, and how to design the assembly of an end-to-end visual exploration pipeline for such data. This provides, we believe, practical guidelines for researchers in different fields (especially those not close to Computer Science) to select, instantiate, and combine such methods to answer their questions.

Still, it is important to pinpoint several open questions regarding the state-of-the-art, or more precisely, what current tools can offer, in this respect:

Scalability: How to visually explore large trail-sets at interactive rates (millions of trails, hundreds of sample points, tens of attributes)? Bundling and data aggregation methods cover the first two points (number of trails and sample points). Aggregating multiple attributes is still an open question, given the fundamentally different types, and ranges, thereof. Given also the mentioned hard limit of visualising only a few attributes at a time (Sec. 3.2), this is, we believe, one of the key open issues in the field.

Interpretability: A visualisation conveys, by construction, a *simplified* view of the data D it receives. Hence, it makes needed simplifications. The question is: How do these simplifications affect the *interpretation* of the data, *i.e.*, the conclusions users will draw from it? A point-in-case is done by bundling: Whereas reducing occlusion in visualisations, it also *deforms* actual trails, thereby potentially misleading users who expect to see accurate spatial locations. Conveying the fact that (bundled) visualisations are necessary simplifications of the actual data is also an open challenge.

Quality: Given all inherent limitations of visualisation design discussed in this chapter, it is obvious that one cannot design the 'perfect' visualisation. Hence, ways (metrics) to gauge the quality of a visualisation are needed. For attributed trail data, these are quite scarce. For instance, we do not still have a theory, let alone metrics, to gauge the quality of a bundling [Lhuillier et al., 2017]. Hence, visualisations are most often evaluated by means of controlled studies. However, when deploying them to explore migration data, which is inherently sensitive and open to controversies, ground truth (required by controlled studies) is typically missing.

Replicability: The last but definitely not smallest issue in visualisation is replicability. Using a visualisation proposal presented in academic research (papers) implies being able to *exactly* replicate the setup presented by the authors. This is increasingly hard to do, even for visualisation professionals. The key reason is the increasing complexity of visualisation algorithms and methods, most of which are not available as open-source software. As such, the optimal solution for the interested practitioner is to reply on potentially less cutting-edge methods which are openly available (see *e.g.* Table 1).

Nevertheless, considering all these challenges, we believe that the current chapter has presented a

convincing set of use-cases, with supporting methodology and tooling, that will help migration scientists to explore, experiment with, adopt, and use visualisation methods for trail data, thereby enhancing their understanding and obtained insights from their respective datasets.

References

[Archambault et al., 2013] Archambault, D., Abello, J., Kennedy, J., Kobourov, S., Ma, K.-L., Miksch, S., Muelder, C., and Telea, A. (2013). Temporal multivariate networks. In *Multivariate Network Visualization*, pages 151–173. Springer.

[Auber, 2004] Auber, D. (2004). Tulip – a huge graph visualization framework. In Junger, M. and Mütze, P., editors, *Graph Drawing Software*, pages 105–126. Springer.

[Auber, 2020] Auber, D. (2020). Tulip graph visualization framework. `https://tulip.labri.fr`.

[Bilsborow, 2016] Bilsborow, R. E. (2016). The global need for better data on international migration and the special potential of household surveys. In *Proc. Improving Data on International Migration – Towards Agenda 2030 and the Global Compact on Migration*. GMDAC.

[Boyandin, 2010] Boyandin, I. (2010). JFlowMap flow map visualization tool. `https://code.google.com/archive/p/jflowmap`.

[Boyandin et al., 2010] Boyandin, I., Bertini, E., and Lalanne, D. (2010). Visualizing migration flows and their development in time: Flow maps and beyond. In *Proc. IEEE InfoVis – Doctoral Consortium*.

[Buchmüller et al., 2016] Buchmüller, J., Jäckle, D., Stoffel, F., and Keim, D. A. (2016). SpaceCuts: Making room for visualizations on maps. In Bertini, E., Elmqvist, N., and Wischgoll, T., editors, *Eurographics Conference on Visualization, EuroVis 2016, Short Papers, Groningen, The Netherlands, 6-10 June 2016*, pages 67–71. Eurographics Association. Open access DOI:10.2312/eurovisshort.20161163.

[Childs et al., 2013] Childs, H., Geveci, B., Schroeder, W., Meredith, J., amd Christopher Sewell, K. M., Kuhlen, T., and Bethel, E. W. (2013). Research challenges for visualization software. *Computer*, 46(5):34–42.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619.

[de Caulaincourt, 1933] de Caulaincourt, A. (1933). *Mémoires*. Eds. Plon.

[Fountoulakis et al., 2018] Fountoulakis, K., Meng, L., Gleich, D. F., and Mahoney, M. W. (2018). Local graph clustering software. `https://github.com/kfoynt/LocalGraphClustering`.

[Gansner, 2020] Gansner, E. (2020). GraphViz graph drawing toolkit. `https://www.graphviz.org`.

[Gapminder Org., 2020] Gapminder Org. (2020). Gapminder visualization tool. `https://www.gapminder.org/tools`.

[Ghoniem et al., 2004] Ghoniem, M., Fekete, J., and Castagliola, P. (2004). A comparison of the readability of graphs using node-link and matrix-based representations. In *Proc. IEEE InfoVis*, pages 17–24.

[Google Inc., 2020] Google Inc. (2020). Google charts API. `https://developers.google.com/chart`.

[Henry and Fekete, 2007] Henry, N. and Fekete, J.-D. (2007). MatLink: Enhanced Matrix Visualization for Analyzing Social Networks. In *IFIP TC 13 International Conference on Human-computer Interaction*, INTERACT'07, pages 288–302, Berlin, Heidelberg. Springer-Verlag.

[Hoffman, 2003] Hoffman, D. R. (2003). *Effective database design for geoscience professionals*. PennWell Corp.

[Holten and van Wijk, 2009] Holten, D. and van Wijk, J. J. (2009). Force-directed edge bundling for graph visualization. *Comput Graph Forum*, 28(3).

[Hossmann et al., 2008] Hossmann, I., Karsch, M., Klingholz, R., Köhncke, Y., Kröhnert, S., Pietschmann, C., and Sütterlin, S. (2008). *Europe's Demographic Future*. Berlin Institute for Population and Development. Open access at `https://www.berlin-institut.org`.

[Hurter, 2015] Hurter, C. (2015). *Image-Based Visualization: Interactive Multidimensional Data Exploration*. Morgan & Claypool.

[Hurter, 2020] Hurter, C. (2020). Implementation of particle systems for time-dependent trail visualization. Open source software at `http://recherche.enac.fr/~hurter/RealTimeBundling.html`.

[Hurter et al., 2014] Hurter, C., Ersoy, O., Fabrikant, S. I., Klein, T. R., and Telea, A. C. (2014). Bundled Visualization of Dynamic Graph and Trail Data. *IEEE TVCG*, 20(8):1141–1157.

[Hurter et al., 2012] Hurter, C., Ersoy, O., and Telea, A. (2012). Graph bundling by kernel density estimation. *Comput Graph Forum*, 31(3):865–874. Software available at `http://recherche.enac.fr/~hurter/KDEEB.html`.

[Hurter et al., 2013] Hurter, C., Ersoy, O., and Telea, A. (2013). Smooth bundling of large streaming and sequence graphs. In *Proc. IEEE PacificVis*, pages 41–48. Publicly available at `https://hal.archives-ouvertes.fr/hal-00878680`. Software available at `http://recherche.enac.fr/~hurter/RealTimeBundling.html`.

[Jolliffe, 2002] Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer. 2^{nd} edition.

[Krüger et al., 2013] Krüger, R., Thom, D., Wörner, M., Bosch, H., and Ertl, T. (2013). TrajectoryLenses – A set-based filtering and exploration technique for long-term trajectory data. *Comput Graph Forum*, 32(3):451–460.

[Lhuillier et al., 2017] Lhuillier, A., Hurter, C., and Telea, A. (2017). State of the Art in Edge and Trail Bundling Techniques. *Comput Graph Forum*, 36(3):619–645.

[Luo et al., 2017] Luo, D., Cats, O., and van Lint, H. (2017). Constructing transit origin-destination matrices with spatial clustering. *Transportation Research Record: J Transp Res Board*, 2652(1).

[Martins et al., 2020] Martins, T., Lago, N., de Souza, H., Santana, E., Telea, A., and Kon, F. (2020). Visualizing the structure of urban mobility with bundling: A case study of the city of São Paulo. In *Proc. SBRC (CoUrb Workshop)*.

[MDP, 2020] MDP (2020). Migration data portal. `https://migrationdataportal.org`.

[Metrô SP, 2018] Metrô SP (2018). *A mobilidade urbana da região metropolitana de São Paulo em detalhes*. Secretaria Estadual dos Transportes Metropolitanos e Companhia do Metropolitano de São Paulo, São Paulo.

[Minard, 2020] Minard, C.-J. (2020). Tableaux graphiques et cartes figuratives. Bibliothèque numérique patrimoniale des ponts et chaussées, `https://patrimoine.enpc.fr/document/ENPC01_Fol_10975`.

[Munzner, 2009] Munzner, T. (2009). A nested model for visualization design and validation. *IEEE TVCG*, 15(6):921–928.

[Nöllenburg, 2014] Nöllenburg, M. (2014). A survey on automated metro map layout methods. In *Proc. Schematic Mapping Workshop*.

[Open Source, 2020] Open Source (2020). Sankey diagram software. `http://www.sankey-diagrams.com/sankey-diagram-software`.

[Rae, 2009] Rae, A. (2009). From spatial interaction data to spatial interaction information? geovisualisation and spatial structures of migration from the 2001 uk census. *Computers, Environment and Urban Systems*, 33(3):161–178.

[Refugee Processing Center, 2020] Refugee Processing Center (2020). Refugee data – admissions and arrivals. `https://www.wrapsnet.org/admissions-and-arrivals`.

[Rockafellar and Wetts, 2005] Rockafellar, R. T. and Wetts, J. R. (2005). *Variational Analysis*. Springer.

[Scheepens et al., 2015] Scheepens, R., Hurter, C., van de Wetering, H., and Wijk, J. J. V. (2015). Visualization, selection, and analysis of traffic flows. *IEEE TVCG*, 22(1):379–388.

[Shneiderman, 1996] Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. IEEE Symp. on Visual Languages*, pages 336–343.

[Sun et al., 2017] Sun, G., Liang, R., Qu, H., and Wu, Y. (2017). Embedding spatio-temporal information into maps by route-zooming. *IEEE Trans Vis Comput Graph*, 23(5):1506–1519.

[Tableau Inc., 2020] Tableau Inc. (2020). Tableau data visualization tool. `https://www.tableau.com`.

[Tamassia, 2013] Tamassia, R. (2013). *Handbook of Graph Drawing and Visualization*. CRC Press.

[Telea and Ersoy, 2010] Telea, A. and Ersoy, O. (2010). Image-based edge bundles: Simplified visualization of large graphs. *Computer Graphics Forum*, 29(3).

[Tufte, 2002] Tufte, E. (2002). Minard's sources for the Napoleon campaing visualization. `https://www.edwardtufte.com/tufte/minard`.

[van der Zwan et al., 2016] van der Zwan, M., Codreanu, V., and Telea, A. (2016). CUBu: Universal Real-Time Bundling for Large Graphs. *IEEE TVCG*, 22(12):2550–2563. Software publicly available at `http://www.staff.science.uu.nl/~telea001/uploads/Software/CUBu`.

[van Liere and de Leeuw, 2003] van Liere, R. and de Leeuw, W. (2003). GraphSplatting: Visualizing graphs as continuous fields. *IEEE TVCG*, 2(9):206–212.

[von Landesberger et al., 2011] von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J., Fekete, J., and Fellner, D. (2011). Visual analysis of large graphs: State-of-the-art and future research challenges. *Comput Graph Forum*, 30(6):1719–1749.

[Wolff, 2007] Wolff, A. (2007). Drawing subway maps: A survey. *Informatik Forsch. Entw.*, 22:23–44.

[Wood et al., 2010] Wood, J., Dykes, J., and Slingsby, A. (2010). Visualisation of origins, destinations and flows with od maps. *The Cartographic Journal*, 47(2):117–129. Also publicly available at `https://openaccess.city.ac.uk/id/eprint/537`.

[Yang et al., 2017] Yang, Y., Dwyer, T., Goodwin, S., and Marriott, K. (2017). Many-to-many geographically-embedded flow visualisation: An evaluation. *IEEE Trans Vis. Comput Graph*, 23(1):411–420. Also publicly available at arXiv:1908.02052 [cs.HC].