BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

WEB APPLICATION FOR GRAPHICAL VISUALIZATION OF GEOSPATIAL TIME SERIES

WEBOVÁ APLIKACE PRO GRAFICKOU VIZUALIZACI GEOGRAFICKÝCH ČASOVÝCH ŘAD

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. KRYŠTOF RYKALA**
AUTOR PRÁCE

SUPERVISOR **Ing. JIŘÍ HYNEK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2021

Brno University of Technology
Faculty of Information Technology

Department of Information Systems (DIFS) Academic year 2020/2021

Master's Thesis Specification

Student: **Rykala Kryštof, Bc.**
Programme: Information Technology
Field of study: Information Systems
Title: **Web Application for Graphical Visualization of Geospatial Time Series**
Category: User Interfaces
Assignment:

1. Get acquainted with the principles and visualization of geospatial data developing in time. Study available map diagrams presenting geospatial data and analyze possibilities of how to map the data to the diagrams.
2. Analyze available technologies for visualization of geospatial data, including authoring systems and existing libraries (such as Leaflet, D3.js). Compare them with other solutions and each other. Propose a new/innovative solution.
3. Analyze the requirement on the visualization of geospatial data changing in time.
4. Design an application that allows the users to visualize geospatial data developing in time, which respects outcomes of items (2) and (3), focusing on generic data, e.g. network communication, epidemic spreading, aim at animation of the visualization
5. Implement the designed solution.
6. Test the result using selected geospatial time series. Propose future extensions.

Recommended literature:

- Kachlík, J.: Grafická vizualizace geografických dat síťového provozu. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.
- Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers/Elsevier, 2010, ISBN: 978-0-12-375030-3.
- Pea-Araya, V., et al.: A Comparison of Visualizations for Identifying Correlation over Space and Time. *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2019): 375-385.
- Leaflet: *Leaflet API reference* [online]. 2019 [cit. 2020-09-16]. Dostupné z: https://leafletjs.com/reference-1.7.1.html

Requirements for the semestral defence:

- Items 1 to 4.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Hynek Jiří, Ing., Ph.D.**
Head of Department: Kolář Dušan, doc. Dr. Ing.
Beginning of work: November 1, 2020
Submission deadline: May 19, 2021
Approval date: February 4, 2021

Abstract

Geospatial data has become an integral part of everyday life. The most common visualization of geospatial data is a static thematic map. However, static maps visualize only one time moment, or several of them aggregated together. Moreover, it is impossible to visualize geographical phenomena such as the evolution of different geographical features (e.g., the growth of carbon emissions or the spread of a virus) without time. This work aims to create a tool for creating visualizations of geospatial time series that allow data to be presented and studied. The solution is developed as a tool of the Geovisto application. The developed tool allows users to create custom visualizations and animations of geospatial time series while being configurable for use with general data. Users with programming knowledge can create new Geovisto tools (e.g., thematic maps) and use the time tool to implement animations themselves in the new thematic maps.

Abstrakt

Geografická data se stala nedílnou součástí každodenního života. Nejběžnější vizualizací geografických dat je statická tematická mapa. Ta ovšem vizualizuje pouze jeden časový okamžik, případně několik z nich, agregovaných dohromady. Bez času navíc není možné vizualizovat geografické jevy, jako je vývoj různých geografických charakteristik (např. růst emisí uhlíku nebo šíření viru). Cílem této práce je vytvoření nástroje pro vytváření vizualizací geografických časových řad, které umožní data prezentovat a studovat. Řešení je vyvinuto jako nástroj aplikace Geovisto. Vyvinutý nástroj umožňuje uživatelům vytvářet vlastní vizualizace a animace geografických časových řad a zároveň je konfigurovatelný pro použití s obecnými daty. Uživatelé se znalostí programování mohou vytvářet nové Geovisto nástroje (např. tematické mapy) a pomocí nástroje pro práci s časem sami implementovat animace do nových tematických map.

Keywords

geovisualizations, thematic maps, time series, animations, Geovisto, Leaflet

Klíčová slova

geovizualizace, tematické mapy, časové řady, animace, Geovisto, Leaflet

Reference

RYKALA, Kryštof. *Web Application for Graphical Visualization of Geospatial Time Series.* Brno, 2021. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jiří Hynek, Ph.D.

Rozšířený abstrakt

Geografická data se stala nedílnou součástí každodenního života. Nejběžnější vizualizací geografických dat je statická tematická mapa. Ta ovšem vizualizuje pouze jeden časový okamžik, případně několik z nich, agregovaných dohromady. Bez času navíc není možné vizualizovat geografické jevy, jako je vývoj různých geografických charakteristik (např. růst emisí uhlíku nebo šíření viru). Zavedení času do vizualizací je tedy důležité, aby bylo možno studovat a prezentovat geografické jevy nebo geografické časové řady obecně. Prezentace geografických časových řad je široce využívána zpravodajskými portály, které používají animované tematické mapy k prezentaci příběhu čtenářům, zato studiem geografických časových řad se většinou zabývají odborníci. Data lze studovat porovnáváním časových okamžiků nebo pozorováním vývoje dat v čase.

Pro vytvoření efektivní vizualizace geografických dat v čase je potřeba zvolit správný typ vizualizace. Existují tři hlavní přístupy: jedna statická mapa, série statických map a animované mapy. Statická mapa poskytuje pouze omezené možnosti vizualizace času. Je možno vizualizovat jeden časový okamžik, nebo několik agregovaných časových okamžiků. Rovněž je možno použít sérii statických map pro různé časové okamžiky. Tento přístup je efektivní pro porovnávání několika časových okamžiků, nicméně při větším počtu efektivita klesá. To je způsobeno kognitivním přetížením, které je způsobeno zahlcením uživatele příliš velkým množstvím informací. Nejefektivnějším způsobem vizualizace geografických časových řad jsou animované mapy. Animované mapy prezentují jednotlivé časové okamžiky vizualizací za sebou, takže je vždy vizualizován jen jeden časový okamžik. Ovšem kvůli kognitivnímu přetížení musí být animace krátká, popřípadě musí být k dispozici ovládací prvky pro ovládání toku animace.

Pro vytváření dynamických animovaných map je možné volit mezi několika přístupy. Prvním přístupem je použití programovací knihovny k vytvoření konkrétní vizualizace geografických dat od základu. Tyto knihovny obvykle nabízejí nástroje pro generování prvků SVG nebo použití plátna HTML, pomocí kterých je možno vytvořit tematické mapy. Ovšem nenabízejí téměř žádnou podporu pro vizualizaci časových řad. Druhým přístupem je použití knihoven pro tvorbu map, jako je například Leaflet, OpenLayers nebo Mapbox GL. Tyto knihovny poskytují hotové mapy a ovládací prvky pro jejich ovládání (např. zoom, změna pohledu kamery, atd.). Tyto knihovny je možno kombinovat s předešlými a vytvářet vlastní tematické mapy, které jsou zobrazeny jako vrstvy nad mapou. Posledním přístupem je použití autorského nástroje jako je například Grafana, Tableau nebo Mapbox Studio. Tyto nástroje nabízejí množství hotových tematických map, které se pomocí nich vytvářejí bez znalosti programování. Často také nabízejí podporu časové složky, ale tato řešení nejsou flexibilní a mají také malé možnosti animace. Kompromisem mezi programovou knihovnou a autorským nástrojem je knihovna Geovisto. Jedná se především o programovací knihovnu s možností použití jako autorský nástroj pro přípravu konfigurací. Hlavní myšlenkou Geovisto je prezentace stejných dat v různých perspektivách, ale v jedné vizualizaci s více datovými vrstvami. Základní vrstva využívá knihovnu Leaflet, která poskytuje interaktivní mapovou vrstvu.

Cílem této práce je vytvoření nástroje pro tvorbu vizualizací geografických časových řad, který pokryje dva hlavní případy užití – prezentaci a studium geografických dat. Při prezentování geografických časových řad jsou data prezentována pomocí animace, je tedy důležité umožnit uživatelům nastavit animaci podle jejich konkrétních potřeb. Mohou chtít změnit dobu trvání přechodu, délku časového kroku nebo přesunout pohled kamery na jinou entitu geografických dat (např. přesunout pohled kamery z USA na Evropu). Naopak při studiu dat není kladen takový důraz na animace. Hlavní důraz je kladen na to,

aby bylo možné data efektivně studovat a porovnávat. Proto je potřeba interaktivní časový přehrávač, který umožňuje procházet časem a možnosti předběžného zpracování dat před jejich vizualizací.

Řešení je vyvinuto jako nástroj aplikace Geovisto. Vyvinutý nástroj umožňuje uživatelům vytvářet vlastní vizualizace a animace geografických časových řad a zároveň je konfigurovatelný pro použití s obecnými daty. Nástroj poskytuje potřebné ovládací prvky pro manipulaci s časem a vytváření animovaných příběhů. Příběhy lze exportovat k uchování nebo k použití jinými uživateli. Rovněž byly rozšířeny existující nástroje Geovisto o podporu času. Zejména nástroje reprezentující tematické mapy implementující animace, které jsou vyvolány při časovém přechodu. Uživatelé se znalostí programování mohou vytvářet nové tematické mapy jako nástroje Geovisto a pomocí nástroje pro práci s časem sami implementovat animace do nové tematické mapy. Aplikaci Geovito s vyvinutým časovým nástrojem lze integrovat do stávajících aplikací nebo ji lze používat jako samostatnou webovou aplikaci.

Použití nástroje je demonstrováno na dvou vytvořených datových sadách, které simulují reálné použití nástroje. První datovou sadou jsou nové případy nakažení Covid-19 v Česku. Tato datová sada byla použita k testování studijního případu užití. Vizualizoval jsem data pomocí kartogramu a použil jsem nástroj čas k efektivnímu studiu dat a identifikaci trendů v nich (například korelace mezi denními novými případy a denními úmrtími). Druhým datovým souborem jsou světové emise uhlíku, který testoval prezentační případ užití.

Vytvořený nástroj pro vizualizaci geografických časových řad bude v budoucnu distribuován jako součást Geovisto. Uživatelé jej tedy budou moci využít pro tvorbu svých unikátních vizualizací nebo si mohou vyvinout vlastní nástroje a použít časový nástroj jako základ pro vizualizaci a animaci nových tematických map.

Web Application for Graphical Visualization of Geospatial Time Series

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Ing. Jiří Hynek, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Kryštof Rykala
May 13, 2021

Acknowledgements

I would like to thank Mr. Jiří Hynek for a great deal of support I have received throughout the writing of this Master's thesis.

Contents

Chapter 1

Introduction

Geospatial data has become an integral part of everyday life. The most common visualization of geospatial data is a static thematic map. However, static maps visualize only one time moment, or several of them aggregated together. Moreover, it is impossible to visualize geographical phenomena such as the evolution of different geographical features (e.g., the growth of carbon emissions or the spread of a virus) without time. Thus, introducing the time into the visualizations is essential to enable the possibility of studying and presenting geographical phenomena, or geospatial time series in general. Presenting geospatial time series is broadly used by news portals. They use animated thematic maps to present a story to the readers. On the other hand, studying the geospatial time series is mainly done by experts. Data can be studied by comparing time moments or by observing the evolution of the data in time.

The goal of this thesis is to design and implement a web application for visualizing geospatial time series. The application would allow users to work with generic geospatial data. They would be able to configure the visualizations and persist them for later use. Once the visualizations are configured, the application would provide controls to manipulate the time effectively. The controls would allow users to study the geospatial time series by navigating in specific time moments and comparing them. Users would be able to define stories, which would make it possible to configure transitions after specific times. Configuring the transitions would allow creating custom animations to present the geospatial time series, essentially telling a story.

Chapter 2 lays down a fundamental theory for visualizing geospatial data, including their temporal component. The theory also focuses on visualizing the change and possibilities of visualizing it using animated maps. Existing solutions to visualize geospatial time series are described in Chapter 3. This chapter describes programming solutions and authoring tools that can be used. The chapter also describes the Geovisto application, which is used for the final solution. Chapter 4 analyses real-world use cases of visualizing geospatial time series. Using the analyzed use cases, I design a tool to solve the problem. The design is the subject of Chapter 5. The implementation of the design is described in Chapter 6. The developed solution was tested using real-world datasets to test the use cases defined in the analysis. The testing is described in Chapter 7. Last but not least, the conclusion is reached in Chapter 8.

Chapter 2

Visualization of Geospatial Data

This chapter presents the theory needed to understand what geospatial data is, how to digitalize it, store it, and finally, how to visualize it. This chapter also provides an overview of commonly used thematic maps used to visualize the geospatial data. Last but not least, the chapter describes techniques used when visualizing geospatial data in time and takes a closer look at animated maps.

2.1 Geospatial Data

To understand what geospatial data is, it is better first to define what data is in general. Oxford dictionary defines data as facts or information, especially when examined and used to determine things or make decisions. In short, data is facts that describe some phenomenon. Nigel Walford [22] describes geospatial data in the following way: "Geography is concerned with the description and explanation of things occurring on or near the Earth's surface, therefore, geospatial data is facts related to features that are spatially referenced to this surface". In more simple words, they are facts about the world around us. One example of geospatial data is a simple blind map, Figure 2.1 shows geospatial data that describes Earth's continents.

Figure 2.1: Earth's blind map of continents. Each continent is one geospatial datum as it describes location and dimensions of the continent.[1]

Menno-Jan Kraak and Ferjan Ormeling [14] point out the main difference between the geospatial data and other data. The geospatial data has a specific location in space (spatial

[1] https://pixabay.com/vectors/world-map-earth-global-continents-306338/

address). Therefore, such objects (e.g., roads, fields, or mountains) and phenomena can be visualized (the visualization is called a map). These visualizations are necessary for studying geospatial data.

2.1.1 Components of Geospatial Data

Menno-Jan Kraak and Ferjan Ormeling [14] define three elementary questions that one can ask about geospatial data: *What?*, *Where?*, and *When?* (see (a) in Figure 2.2). By answering these questions, Kraak and Ormeling divide geospatial data into a locational component, an attribute component, and a temporal component. The locational component refers to the position and dimensions (geometrical aspects) of the phenomenon, while the attribute component refers to other (non-geometrical) characteristics of the geospatial data. The temporal component refers to the specific moment in time, for which both, the locational and attribute components, are valid. The components (answers to the questions) compose the nature of an object (see (b) in Figure 2.2). All three types of components of the object can have multiple characteristics (see (c) in Figure 2.2), for example, different coordinate systems, multiple variables, or different kinds of time.

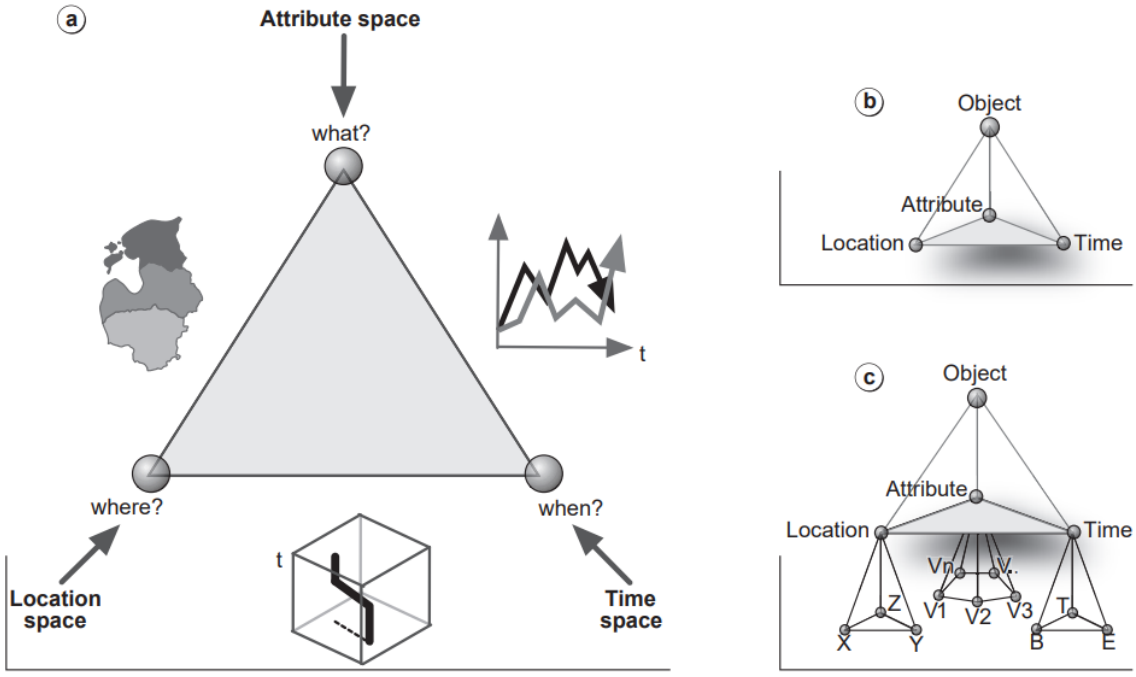

Figure 2.2: The components of geospatial data: (a) locational, attribute and temporal component, and the questions that they answer (where, what and when); (b) the nature of the object (object view); (c) detailed (multiple) characteristics of the data components. [14]

Locational Component

The locational component answers the *Where?* question, by giving the data its location in space (spatial address)—this is done by *georeferencing* them [14]. Linda L. Hill [11] defines the *georeferencing* as relating information to a geographic location. The informal way of referring to locations is done in ordinary discourse using *placenames*. The formal

representations use the longitude and the latitude coordinates, or other spatial referencing systems (e.g., footprints). *Footprints* refer to a specific spot or an area where something is located. The footprints are the basis for calculating distance and direction and for definitions of geographical relationships (e.g., overlap and containment). In order to take full advantage of placename referencing, it is needed to translate formal representations to informal ones. By translating the placenames to their formal representations, it is possible to map them and see the geographic patterns and associations. The translation is done by using gazetteers. Linda L. Hill [11] defines *gazetteers* as dictionaries of placenames that have geospatial footprints for the named location. Placenames within a gazetteer are unique and serve as the unique identifiers for gazetteer entries. An example of commonly used placenames is country names and codes, which are defined by standard ISO 3166 and are subject to Subsection 2.1.3.

Attribute Component

Attributes answer the *What?* questions: What is at a specific location and what are its characteristics? [14]. The attribute data can be composed of either *qualitative* data or *quantitative* data. Quantitative data can be measured and expressed by numerical value (e.g., air pressure, height above sea level, income or election results). Qualitative data, on the other hand, approximates and characterizes (e.g, language, soil characteristics, geological formations). The attributes can change in time, changing the objects they describe [14]; visualizations of these changes are described in Section 2.3. The attribute component is visualized by using thematic maps [14], these maps are the focus of Section 2.2.

Temporal Component

Without the temporal component, the maps can show only a single snapshot of data in time. To be able to successfully study geographical processes or events, the time has to be considered as well. The definition of time is not as simple as one might think. The Oxford Dictionary defines time as the indefinite continued progress of existence and events in the past, present, and future regarded as a whole. Kraak and Ferjan Ormeling [14] say that geosciences are all about events and change and they define two types of events. An event can be either continuous (e.g., changing temperatures at a meteorological station) or discrete (e.g., municipal boundary changes). They also say that time can be measured objectively or subjectively and provide the following examples: a train trip will take 1:30; a lecture is scheduled for one hour, but could be experienced as shorter or longer. Time can be considered either linear or cyclic [14]. An example of the linearity of time is society's system of counting years. Meanwhile, examples of cyclic times are returning days and nights, or seasons. The visualization of data with the temporal component is the focus of Section 2.3.

2.1.2 Representation of Geospatial Data

It is necessary to digitalize the real world geospatial data, in order to be able to work with it and to store it on a computer. Basic geometric shapes can be used to describe real world objects [14]. Markus Schneider [20] introduces *spatial data types*, which are used to model geometry and to represent geometric data in database systems. The literature [14, 19, 20] defines three spatial data types: point, line and area.

- **Point** is a location of a geographical object in space. The location is defined by a coordinate pair (x, y). Points are used to represent size-less geographical objects (e.g., cities, buildings, points of interest).
- **Line** (Figure 2.3) is a sequence of points representing its two end nodes and zero or more internal nodes (vertices). The line can be composed using multiple (smaller) line segments instead of only one line. A closed line forms a *polygon*, which is essential in defining an area. Lines are used to represent connections in space or a movement through space (e.g., roads, rivers, flight paths).

Figure 2.3: A line represented by two end points and three vertices. [19]

- **Area** (Figure 2.4) is represented by its boundaries. Area's boundaries are defined by one or more non-overlapping polygons. Areas are used to represent the partition of space (e.g., state borders, lakes, district, forest).

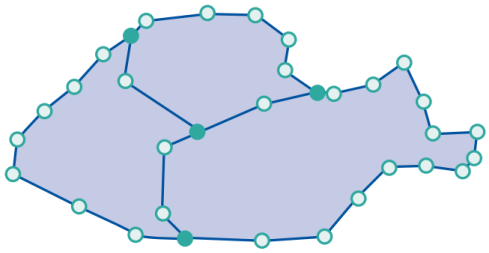

Figure 2.4: An area represented by three non-overlapping polygons, that form the area's boundaries. [19]

2.1.3 Storing Geospatial Data

When storing, the geospatial data is usually divided into locational data, attribute data and temporal data [14]. The locational data refers to the geometrical aspects. These geometrical aspects can be described by the geometric shapes defined in Subsection 2.1.2. On the web, the most widely used format for storing locational data is GeoJSON. GeoJSON is also widely used in JavaScript web-mapping libraries, JSON-based document databases, and web APIs [6]. The attribute data can be saved as a part of locational data. Another approach is to use gazetteers and store the attribute and locational data separately. The attribute data records are then mapped to their location using placename unique identifiers, e.g., country code (ISO 3166 [2]).

The temporal data defines the validity of both the geospatial data and the attribute data. The temporal component is described more in-depth in Subsection 2.1.1. The literature [14]

describes one problem with temporal data: It is not always clear what type of time is stored. Most prevalent is the world time (the moment an event took place), but there is also the database time (the moment an event was stored) and the display time (the moment an event was visualized, although this type relates more to visualizations, it is not usually stored).

GeoJSON

The RFC-7946 [6] defines GeoJSON in the following way: "GeoJSON is a geographical data interchange format based on JavaScript Object Notation (JSON)". It defines several types of JSON objects (GeoJSON types) and how they are combined to represent data about geographic features, their properties, and their spatial extents. GeoJSON uses a geographic coordinate reference system, World Geodetic System 1984, and units of decimal degrees. A simple example of GeoJSON can be seen below.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

The term *GeoJSON object types* refers to nine case-sensitive strings: Feature, FeatureCollection, and seven geometry object types. The geometry object type refers to seven case-sensitive strings: Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon, and GeometryCollection. The main difference between the geometry object types, besides GeometryCollection, is the type of the *coordinates* member. The differences between the GeoJSON object types are as follows:

- **Point**, the *coordinates* member is a single position.
- **MultiPoint**, the *coordinates* member is an array of positions.
- **LineString**, the *coordinates* member is an array of two or more positions.
- **MultiLineString**, the *coordinates* member is an array of LineString coordinate arrays.
- **Polygon**, the *coordinates* member must be an array of linear ring (closed LineString with four or more positions) coordinate arrays. If the array contains more than one linear ring, the first one must be the exterior ring, which represents the boundaries of the polygon. Any other linear ring in the array must be an interior ring, which represents holes within the boundaries.
- **MultiPolygon**, the *coordinates* member is an array of Polygon coordinate arrays.

- **GeometryCollection** has a member *geometries*, which value is an array of other GeoJSON geometry objects or the array can be empty. The GeometryCollection type does not have the *coordinates* member. Instead, the coordinates of all its parts belong to the collection.
- **Feature** object represents a spatially bounded thing. Every Feature object is a GeoJSON object. It has a *geometry* member, which contains a geometry object (e.g., Point, MultiPoint, etc.). A Feature object also has an optional *properties* member, which value is a JSON object (if set).
- **FeatureCollection** has a member *features*, which value is either an array of Feature objects defined above or an empty array.

Country Codes

ISO 3166 [2] is a standard that defines internationally recognized country codes and codes for their subdivisions. The defined codes can consist of letters and/or numbers (used for the country's subdivisions). The standard itself does not define the names of countries.

ISO 3166 consist of three parts [2]:

- The **country codes** can be represented either as a two-letter code (*alpha-2*), three-letter code (*alpha-3*) or three-digit numeric code (*numeric-3*). *Alpha-2* is recommended as the general-purpose code, while *alpha-3* resembles country name. The *numeric-3* can be useful if one need to avoid using Latin script. Examples of codes for country of Switzerland, *alpha-2* is *CH*, *alpha-3* is *CHE* and *numeric-3* code is *756*.
- The **codes for subdivisions** are represented as the *alpha-2* code for the country, followed by up to three characters. Example: code for Riau province of Indonesia is *ID-RI* and for Rivers province in Nigeria is *NG-RI*.
- The **formerly used codes** are four-letter codes (*alpha-4*).

The advantage of using country codes is saving time and reducing errors as it is unnecessary to use a country's name, which needs to be localized. The examples of usages of country codes are top-level domain names (e.g., ".fr" for France), identifying the country of origin of the bank in money transfers [2].

2.2 Visualizing Geospatial Data

Maps are the most efficient and effective way to transfer spatial information [19]. "Where do I find?" or "Where did it come from?" are both examples of questions one can ask about geospatial data. The geospatial data can also be visualized in a non-map form. However, the map provides the full picture by revealing spatial relations and patterns and offers the user an overview of the distribution of particular phenomena [19]. Board [5] defines a map as "a representation or abstraction of geographic reality. A tool for presenting geographic information in a way that is visual, digital or tactile".

Rolf A. de By [19] states that maps can also answer the question *"what?"* by informing about thematic attributes (attribute component) of the geographic objects. However, maps can answer these questions only in relation to the location. The last type of question that can be answered is *"when?"* which is a subject of Section 2.3.

Rolf A. de By [19] divides maps in *topographic* and *thematic* maps. A topographic map visualizes the Earth's surface as accurately as possible. The visualization can include infrastructure (e.g., railroads and roads), land use (e.g., vegetation and built-up area), and other geographical objects. Tennekes [21] says that thematic maps show spatial distributions and that the theme refers to the phenomena shown, which is often demographical, social, cultural, or economic. The thematic map types are listed and described in Subsection 2.2.2.

2.2.1 Mapping Data

When displaying data, it is necessary to map its dimensions onto the graphical visualization. The chosen graphical visualization usually has a limited number of dimensions that they can display, while data can have an unlimited number of dimensions. For example, data records in a database are only limited by memory, and the same goes for JSON format, which can have as many key-value pairs as necessary. Therefore, it is needed to map such records with unlimited dimensions onto diagrams that have only a limited number of them.

As an example, let us take data from Table 2.1. When changing the mapping of the dimensions and axis onto the diagrams, the communicated information of the diagram can change (Figure 2.5). Hence, it is necessary to choose the most appropriate mapping, which often requires domain knowledge and understanding of how the user will use the diagrams.

city	product	count
Prague	shirts	5
Warsaw	shirts	6
London	shirts	7
Prague	pants	3
Warsaw	pants	8
London	pants	4

Table 2.1: An example of mapping data dimensions onto diagram.

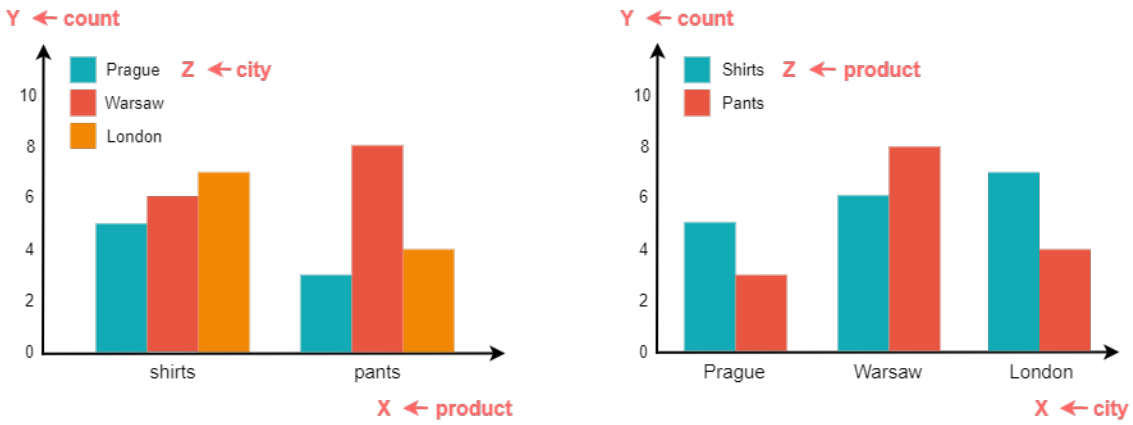

Figure 2.5: Two examples of mapping dimensions on axis of the chart.

Rolf A. de By [19] classifies the contents of a map in different basic categories: point symbols, line symbols, area symbols, and text symbols. The appearance of these symbols is characterized by their nature, meaning they change their visual appearance based on the mapped data. Although the combinations of visual appearances of the symbols are

limitless, Rolf A. de By [19] uses categorization defined by Bertin [4]. He distinguished six categories of visual variables: size, value (lightness), texture, color, orientation, and shape. By using these variables, it is possible to make one symbol different from another. De By [19] claims that visual variables stimulate the user's perception of the map and says: "What is perceived depends on the human capacity to see what belongs together (e.g., all red symbols represent danger), to see the order (e.g., the population density varies from low to high—represented by light and dark color tints, respectively), to perceive quantities (e.g., symbols changing in size with small symbols for small amounts), or to get an instant overview of the mapped theme". Figure 2.6 shows an example of mapping data onto a thematic map, where the attribute dimension is mapped to the size of the symbols. The mapping of temporal dimension is described in Subsection 2.3.1.

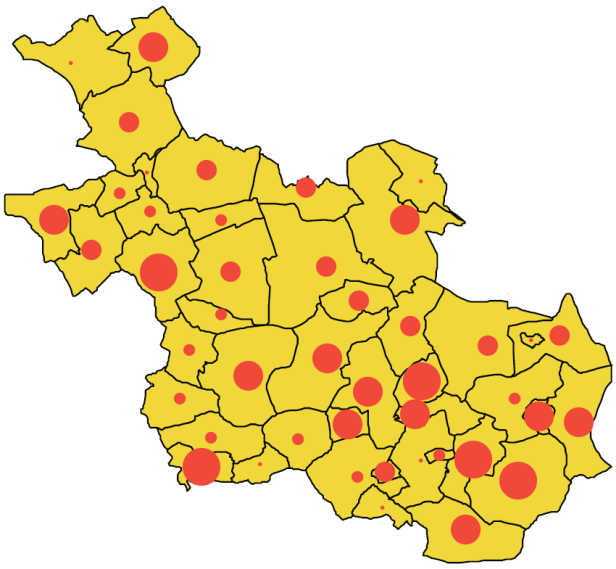

Figure 2.6: A thematic map with symbols. [19]

2.2.2 Thematic Maps

Thematic maps represent the distribution of particular themes (attributes), and they provide a geographic reference to the theme represented [19]. Simply put, they visualize the attributes in a location that they occur in.

This section describes mainly thematic maps that are used in the practical part of this thesis. Besides diagrams described in this section, the literature [10, 14] mentions other commonly used thematic maps such as isopleth maps, cartograms, isoline maps, dot maps, heatmaps, and many more.

Choropleth Map

A choropleth map displays the area data. The difference in areas is visualized by using different shading, colors, or patterns [10, 14]. Choropleths are generally suited for visualizing ratios, percents, or rates (e.g., population density or per-capita income) as opposed to absolute units (e.g., total dollars for an area) [10]. Data is often organized into class intervals as shown in Figure 2.7. Chien [7] recommends to have 3 to 7 data classes.

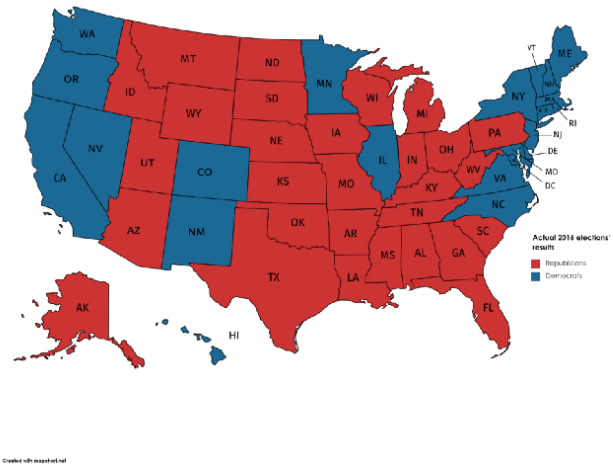

Figure 2.7: An example of the classified choropleth map displaying results of presidential elections.[2]

While grouping data into classes is still prevalent, *unclassed choropleths* offer a more accurate representation of data [13]. In this type of choropleth, each unique value is displayed in a unique style (color, shade, pattern). The disadvantage of these maps is that the user loses the sense of classification (since it is missing). It makes it harder to compare the data and to classify them mentally [1]. The example of unclassed choropleth is shown in Figure 2.8.

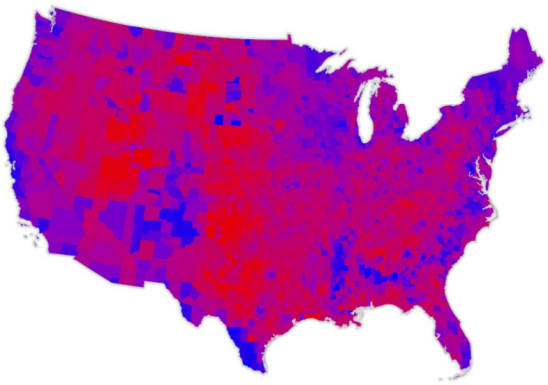

Figure 2.8: An example of the unclassed choropleth map displaying results of presidential elections.[3]

Symbol Map

Symbol maps bind thematic variables to the visual properties of symbols overlaid on top of the corresponding map features [16]. The symbols can be different in shape, orientation, or color. Kraak [14] distinguishes between figurative and geometrical symbols. Moreover, he explains that the figurative ones are used when associations might ease their recognition

[2]https://towardsdatascience.com/presidential-elections-forecast-19d366ea7945
[3]https://cartographicperspectives.org/index.php/journal/article/download/cp86-kelly/1576/

(domain-specific symbols like those used in meteorology). In contrast, geometrical ones are used for more abstract phenomena. Sometimes, miniature graphs are used as symbols; they can encode as many as 10 to 20 attributes [10].

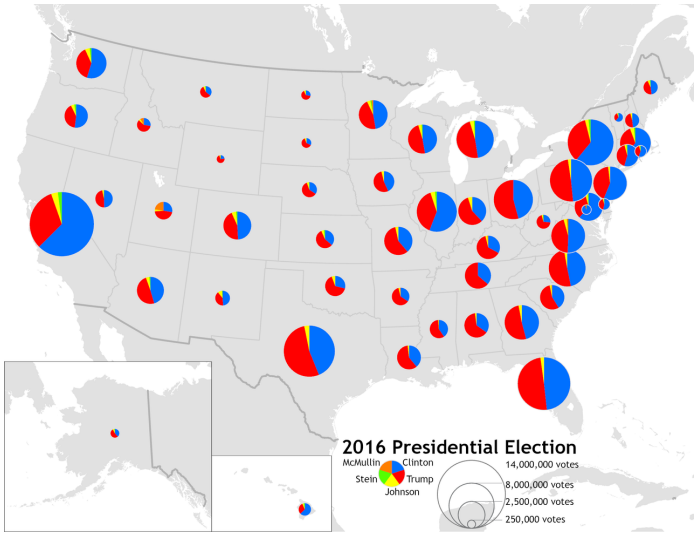

Figure 2.9: An example of the map with proportional symbols, where the used symbols are pie charts.[4]

The symbols can be scaled with respect to the value they display [1]. This variation of symbol map is called **proportional symbols map**. An example of this thematic map is shown in Figure 2.9. Few [9] points out that many proportionally scaled symbols do not work well with maps (e.g., bars and lines), as it can be challenging for the user to compare the symbols which are not beside one another. He also points out that many maps do not have enough room to scale the symbols, which results in overlapping symbols and unreadable maps.

Connection Map

A connection map shows relations between two locations in a map. The literature [10, 14, 23] describes following characteristics of connection maps. Each connection consists of a source node and a target node forming an edge. One or more source nodes can be linked to many target nodes by edges. The edges can also depict direction by using arrow symbols. An example of the connection map is showed in Figure 2.10.

Zhou [23] talks about a *visual clutter* problem in connections maps which is caused by dense edges. The visual clutter can be reduced by techniques, such as filtering, sampling, and clustering, but recently the most common clutter reduction technique has been edge bundling.

2.2.3 Thematic Layers

Rinzivillo [18] provides an example and describes thematic layers in the following way. Real-world data contains many different geographical objects. A region can be composed

[4] https://en.wikipedia.org/wiki/Proportional_symbol_map
[5] https://www.data-to-viz.com/story/MapConnection_files/figure-html/unnamed-chunk-2-1.png

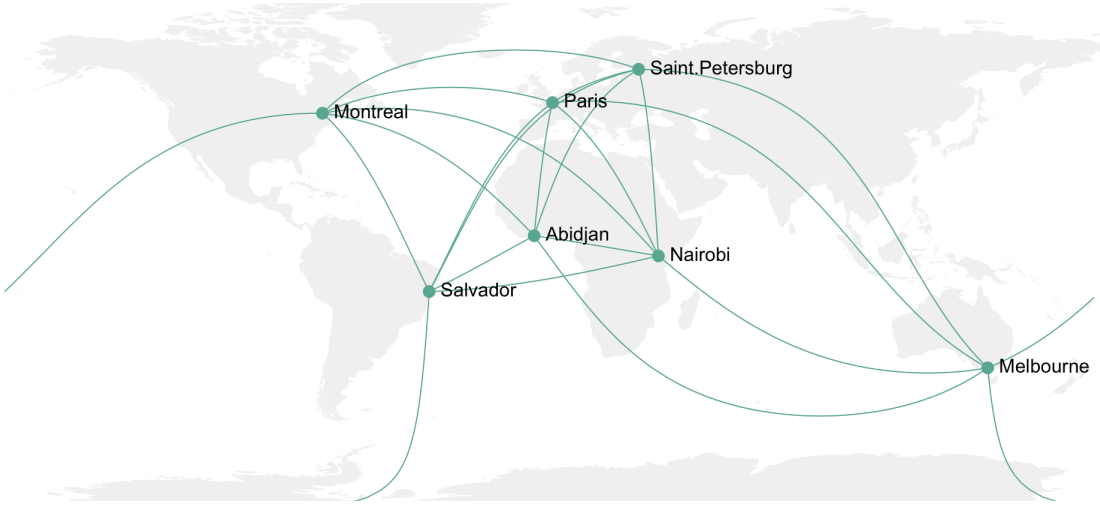

Figure 2.10: An example of a basic connection map.[5]

of cities, road networks, buildings, hydrographic information, etc. GIS typically handles this thematic division by organizing data in layers. Then, each layer contains information about one thematic type. For example, one layer contains information about the road network while another contains information about rivers and lakes. Although different data is visualized using different thematic layers, they are integrated together using geographic locations. By using the geographic location, the layers can be overlayed or spatially joined. Each layer comprises two types of data: locational data and attribute data (Subsection 2.1.1). The locational data describes the location of the objects, and the attribute data specifies the characteristics of the data.

2.3 Visualizing Time

Most visualizations display only a single point in time, but to study geographical processes, events, and phenomena, one should also consider time [14]. Examples of such phenomena are tracing the diffusion of diseases or understanding the regionally varied impacts of global climate change. By including the temporal component in the visualizations, the visualizations can help answer questions related to time. Kraak [14] uses MacEachren's questions concerning such geospatial data with a temporal component. MacEachren classified these questions into seven query types.

- if? or whether?: addresses the existence of an entity
- when?: location of the entity in time
- how long?: duration of the entity
- how often?: the temporal texture of the entity
- how fast?: rate of change of the entity
- what order?: the sequence of entities

- do entities occur together?: synchronization

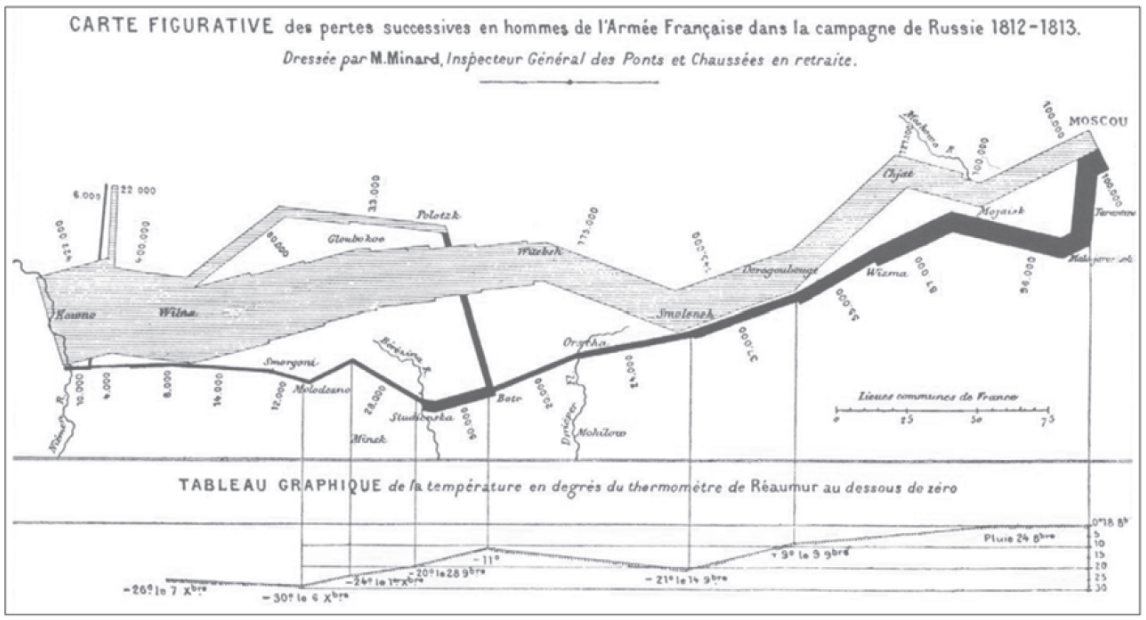

Figure 2.11: Map by Charles Joseph Minard in 1861, visualizing Napoléon's invasion of Russia in 1812–1813 in geographic and temporal dimensions. [14]

Good visualizations should be able to answer temporal questions defined. The literature [11, 14] gives Minard's map, showing Napoléon's invasion of Russia (Figure 2.11), as an example of an innovative design that captures change. This map illustrates the power of a space-time visualization to convey information and is praised by Edward R. Tufte [14].

2.3.1 Mapping Change

Mapping the time dimension means mapping the change. Both Rolf A. de By [19] and Kraak [14] describe three types of changes in time: the change in geometry, the change in an attribute, and combination of both. He also provides examples of these types of changes. An example of changing geometry in time would be the evolving coastline of the Netherlands (Figure 2.12), the location of Europe's national boundaries, and the position of fronts on weather maps. Examples of changes of attributes are changes of a parcel's owner or changes in road traffic intensity. Last but not least, the example for the combination of both geometry and attribute changes would be urban growth, where the urban boundaries expand and the land-use shifts from rural to urban. Besides these types of changes, Rolf A. de By [19] defines one more type of change, the change in an existence of a feature, such as an appearance or disappearance. A real-world example of this change is an iceberg that disappears after it melts.

Given the defined types of change, the literature [8, 14, 19] distinguishes between three temporal cartographic techniques: a single static map, series of static maps, and animated maps:

- **A single static map**: Specific graphic variables and symbols are used to show the change in order to represent an event. In Figure 2.13 (a) colors are used to represent time. Darker colors visualize older and lighter colors parts of the city.

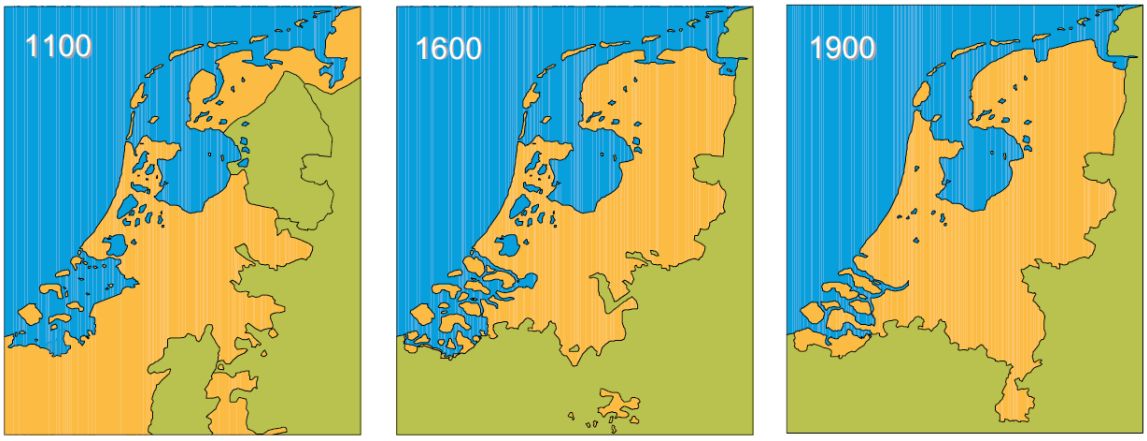

Figure 2.12: Series of static maps of evolving coastline of Netherlands. [19]

- **Series of static maps**: A single map in the series represents one snapshot in time. Together, the maps visualize the process of change as shown in Figure 2.13 (b). The change is perceived by the succession of individual maps depicting the situation in successive snapshots. The number of snapshot images should be low as it is difficult for a human to follow long series of maps.
- **Animated map**: The change is perceived to happen in a single image. One image is a snapshot in time which represents one frame in an animation. The animation displays several snapshots after each other, just like a movie with successive frames. Figure 2.13 (c) shows a simulation of the animation where one image represents one snapshot (a frame of the animation). The animated map can have either interactive or non-interactive animations. One example of non-interactive animations would be gifs (repeating sequence of images) often used on the web. The interactive animated maps are the subject of Subsection 2.3.2.

2.3.2 Animated Maps

Animated maps have become increasingly popular in recent years and became widespread through the Internet. Dodge [8] claims that, unlike static maps, animated maps seem suited for visualization of the change between moments and that static maps are often insufficient when it comes to representing time because they directly represent geographic behaviors or processes. Animated maps can incorporate time explicitly. Therefore, they are better suited for visualization of the change, and cartographers tend to exploit their potential.

The basic goal of the cartographic animation is the depiction of change. The literature [8, 14] divides cartographic animations into two subcategories: temporal and non-temporal animations. Dodge [8] describes *temporal animation* as "animation that deals with the depiction of dynamic events in chronological order and depicts the actual passage of time in the world". Examples of temporal animations are: changes in the coastline of Netherlands (Figure 2.12), the spread of diseases and population growth [14]. The *non-temporal animation* uses animation time to show the change in spatial relationships or to clarify geometrical or attribute characteristics of spatial phenomena [14]. Examples of non-temporal animations are: change in perspective of the map (e.g., from 2D map into 3D map) or animation of camera motion (e.g., fly-bys or fly-throughs in 3D maps) [8].

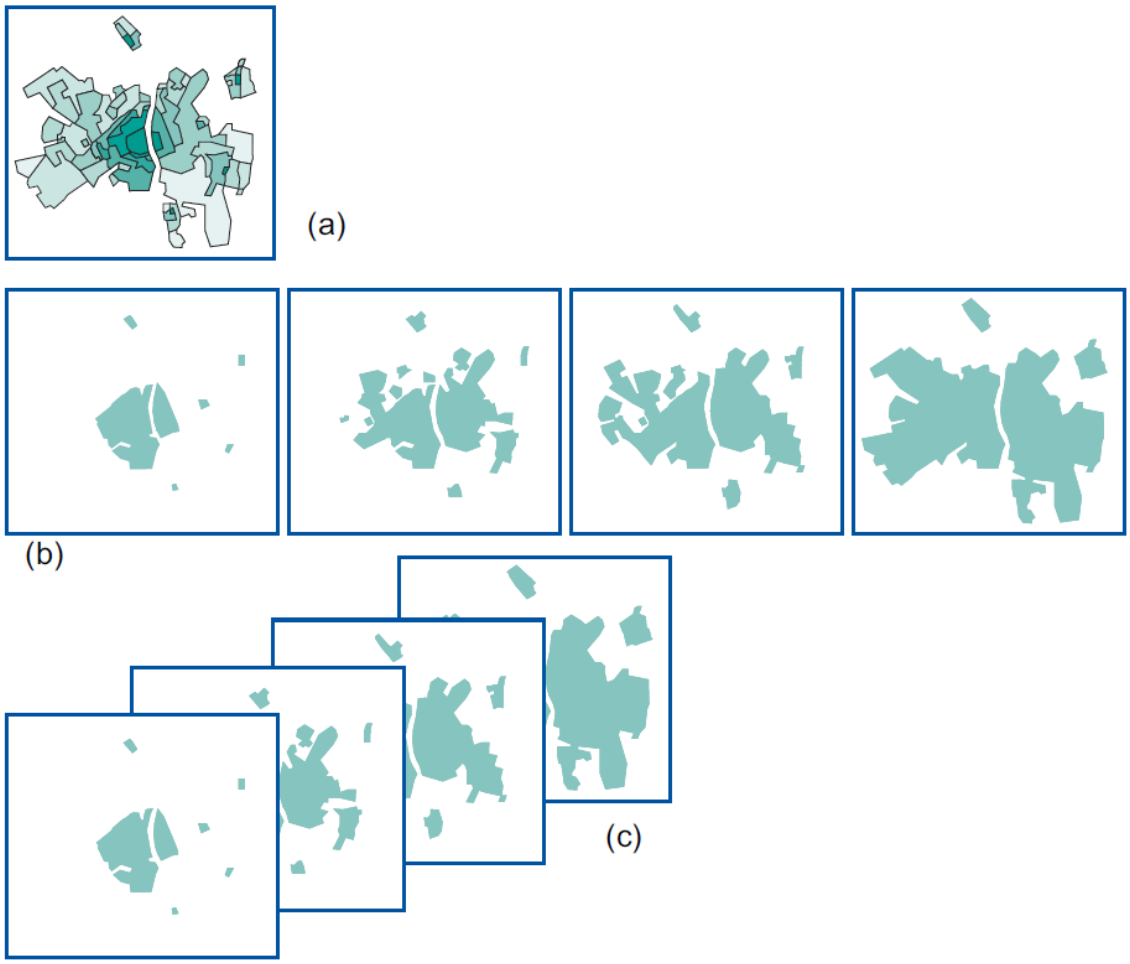

Figure 2.13: Mapping change; example of the urban growth of the city of Maastricht: (a) single static map, in which colors represent age of the city areas; (b) series static of maps; (c) animation (each simulates represents one frame). [19]

Characteristics of Animations

Animated maps present information over time. Thus they have an additional representational dimension that can be used to display information [8]. Temporal animated maps have a *temporal scale*. Dodge [8] defines the temporal scale as the ratio between real-world time and animation time. He gives an example that five years of data shown in a 10-second animation would have a temporal scale of 1:157 million. Most animated maps keep the same temporal scale for the whole animation time. However, it is possible to build animations that vary their temporal scale as they are played (this brings focus on important moments) [8].

Dodge [8] uses the claim of Andrienko and Gatalsky [3] that the best understanding may be achieved when the animation is under user control, and the geospatial data can be explored in a variety of ways. Dodge's proposed solution is achieved by visualizing the passage of time alongside the map through an *interactive temporal lengend*. The advantage of interactive graphical temporal legend is that it can communicate at a glance both the currently displayed moment (e.g., 3/5/1995 11:30) and the relation of that moment to the

entire dataset (e.g., beginning and end of the animation). Kraak [14] defines interactive controls of the temporal legend, which are required as a minimum to use the animated temporal maps effectively: forward, backward, slow, fast, and pause. Nowadays the geographical applications usually use media-player-like controls as the interactive temporal legend [19].

Animation Variables

When designing an animated map, one needs to make sure that the viewer understands the development or trend of the phenomena [14]. The literature [15, 17] defines the following animation variables that can be changed in order to design the change in time. The animation variables are size, shape, position, speed, viewpoint, distance, scene, texture, pattern, shading, and color.

- **Size**: When an area of a map (e.g., country or region) or a symbol (e.g., circle) changes its size it represents a change in value. The smaller the size is, the smaller the value is, and vice-versa.
- **Shape**: Parts of a map can change their shape. Animation can show the change of country's border in time or display a difference between different map projections (by blending between their shapes).
- **Position**: By changing the position of a symbol in a map, it is possible to show the change of location. An animation can depict a movement from point A to point B.
- **Speed**: The speed of a transition or movement can depict the rate of change.
- **Viewpoint**: A change in the viewer's angle of view can highlight a particular part of a map as part of an animation and bring the observer's attention to parts of a map with more significance.
- **Distance**: The change in the distance in which the viewer sees the scene may be interpreted as a change in scale.
- **Scene**: Visual effects are used to indicate a transition in an animation from one state to another.
- **Texture, Pattern, Shading, Color**: These variables are most commonly used to depict a change in value. These may also be used to highlight a part of a map.

Pitfalls of Animated Maps

The literature [8, 14, 16] often mentions perceptual and cognitive limitations of the animated maps. Menno-Jan Kraak [14] mentions the problem called *cognitive overload*. This problem is caused when the viewer of the animations is overwhelmed by the number of frames they need to process, and because of this, it makes it hard for the viewer to remember what he saw. He also mentions the problem where the viewer may fail to notice important changes during the animation. Dodge [8] points out the *split attention* problem. This problem occurs when the viewer switches attention between the legend and the map, causing him to miss important cues or information.

Chapter 3

Existing Tools for Geospatial Visualizations

When it comes to visualizing geospatial data, including their temporal component, many tools and solutions already exist. The creation of visualizations can be generally divided into two categories based on their approaches. The first one uses a **programming library** and the second one uses an existing **authoring tool**. While the programming libraries provide the most flexibility, they require programming knowledge and more effort to construct geospatial visualizations. On the other hand, the authoring tools usually do not require programming knowledge. However, they provide less flexibility as the users can create only geospatial visualizations that the specific authoring tool provides.

3.1 Programming Libraries

Programming libraries offer a highly customizable way to visualize geospatial data but come with the requirement of having programming skills. There are many libraries for creating thematic maps, the majority being developed in JavaScript for web development, and those are the ones that this section focuses on. The programming libraries can be further distinguished into three subcategories: libraries that provide tools for creating SVG/HTML elements, charting libraries, and geospatial frameworks.

3.1.1 Libraries for Creating and Manipulating SVG/HTML

Libraries for creating and manipulating SVG/HTML provide tools for creating the visualizations from scratch. Such libraries provide API for rendering SVG elements or drawing on canvas. They are the most flexible but require the most programming skills and also math knowledge. One popular option of such a library is D3.js[1] (D3).

D3 provides functions for Data Object Model (DOM) manipulation and data transformation. Data is bound to the DOM and then visualized by generating SVG elements. D3 does not provide any thematic maps out of the box. Thematic maps are created by generating SVG for all the map elements. To work with geospatial data effectively, D3 provides functions for handling GeoJSON or map projections tools (g3-geo, d3-geo-projection).

[1] https://d3js.org/

3.1.2 Charting Libraries

Charting libraries provide ready-made thematic maps. However, programming knowledge is still required. A big disadvantage of these libraries is limited flexibility and map interactivity. There are generally two types of charting libraries based on the approach they take. The first approach is imperative (e.g., HighCharts[2]) and the other one declarative (e.g., Vega[3], Nivo[4]). These libraries are easy to use, but they usually come with only basic thematic maps (e.g., choropleths), and none of the mentioned libraries come with support for time series.

3.1.3 Geospatial Frameworks

Geospatial frameworks often provide basic map tools (e.g., zoom and camera position). When using such tools, one can focus on creating only the geospatial visualizations and not the map itself. Examples of such geospatial frameworks are Leaflet[5], Google Maps API[6] or OpenLayers[7]. All of them are based on the same principle of thematic layers (Subsection 2.2.3). When one wants to create a thematic map, such as a choropleth, he needs to implement a new layer that is displayed on top of a rendered base map. The thematic layer can be created in other libraries such as the mentioned D3. These libraries provide the most flexibility and functionality when working with geographical data, as they allow creating pretty much any thematic map imaginable. Figure 3.1 shows a choropleth made on top of Leaflet map.

An alternative to previously mentioned geospatial frameworks is Mapbox GL[8]. The difference in Mapbox GL is that the library uses WebGL. WebGL is used to render interactive maps at a high frame rate. Although using WebGL can lead to performance issues as the rendering is done on the client side. Mapbox GL does not come with support for time series, but the users are able to implement the solution themselves. This is done using filters and creating a custom control for the timeline. An example of a map with a trivial time-player component is shown in Figure 3.2.

3.2 Authoring Tools

Authoring tools are already developed solutions. One of the advantages is that they usually require no programming and math knowledge. However, there is a learning curve when using these tools. The difference between authoring tools is how the users map the data onto diagrams and in the configuration possibilities that the given tool provides. The authoring tools usually come as a standalone application, so they offer minimal options for customizing or extending them (e.g., the users can use only predefined thematic maps).

[2] https://www.highcharts.com/
[3] https://vega.github.io/
[4] https://nivo.rocks/
[5] https://leafletjs.com/
[6] https://developers.google.com/maps
[7] https://openlayers.org/
[8] https://mapbox.com/

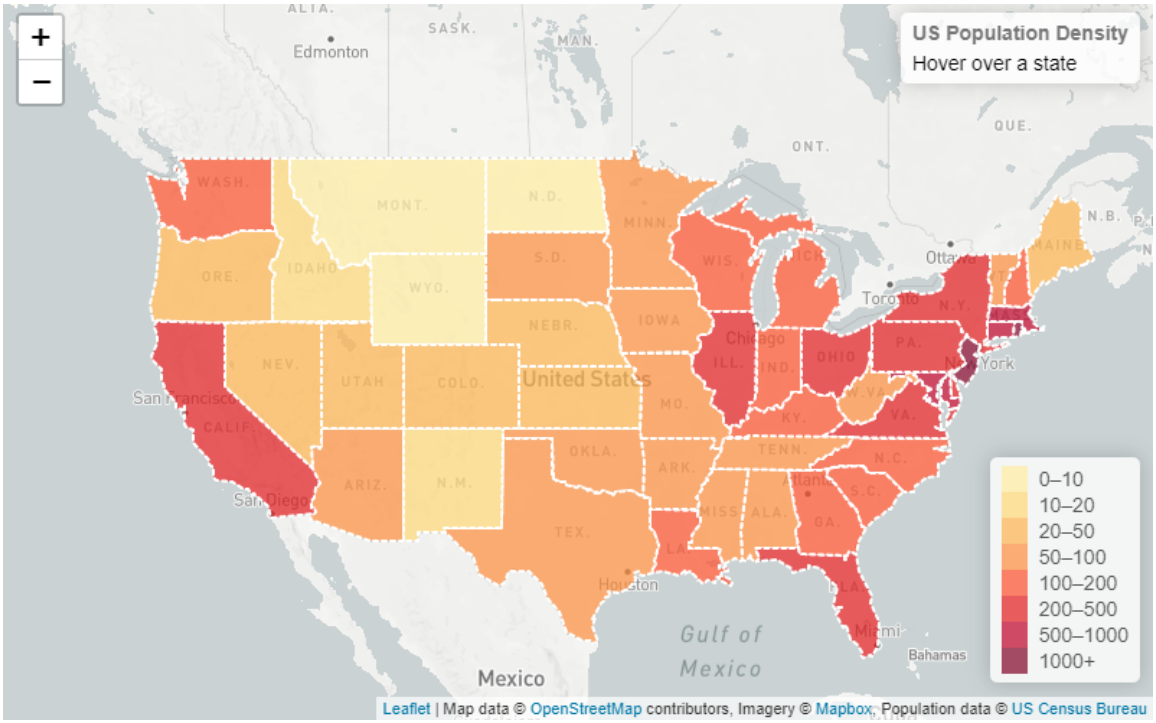

Figure 3.1: The choropleth made on top of Leaflet map.[9]

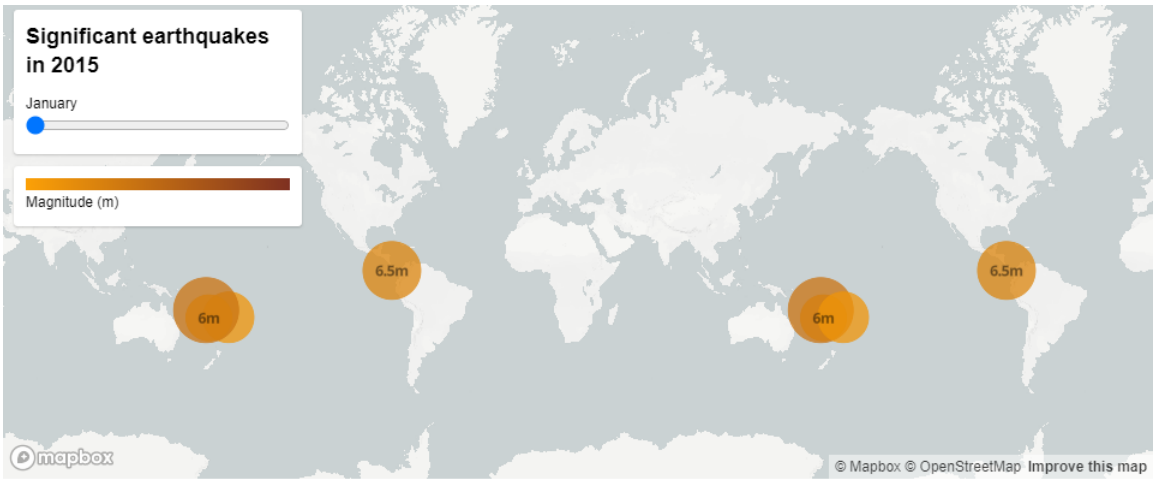

Figure 3.2: The example of a thematic map created in Mapbox GL. The example has also basic time-player control.[10]

3.2.1 Microsoft Excel

The simplest example of an authoring tool can be Microsoft Excel. It supports mapping tabular data onto simple thematic maps, such as choropleth. Even though the creation of simple static maps is easy, a significant disadvantage is their limited customizability. They

[9] https://leafletjs.com/examples/choropleth/
[10] https://docs.mapbox.com/mapbox-gl-js/example/timeline-animation/

usually offer only limited options, such as color change and a few thematic maps. Microsoft Excel offers no support for visualizing the temporal component besides simple static maps.

3.2.2 Tableau

Tableau is a commercial authoring tool that allows generating visualizations such as dashboards or stylesheets. Tableau requires no programming knowledge to create data visualizations as it provides a wide scale of predefined thematic maps. Mapping data onto the thematic maps is done in a drag and drop manner. Tableau also offers support for temporal animated maps (Figure 3.3), but the options are limited. However, the animated maps allow to configure the mapping of a time dimension and choosing a time length of one step.

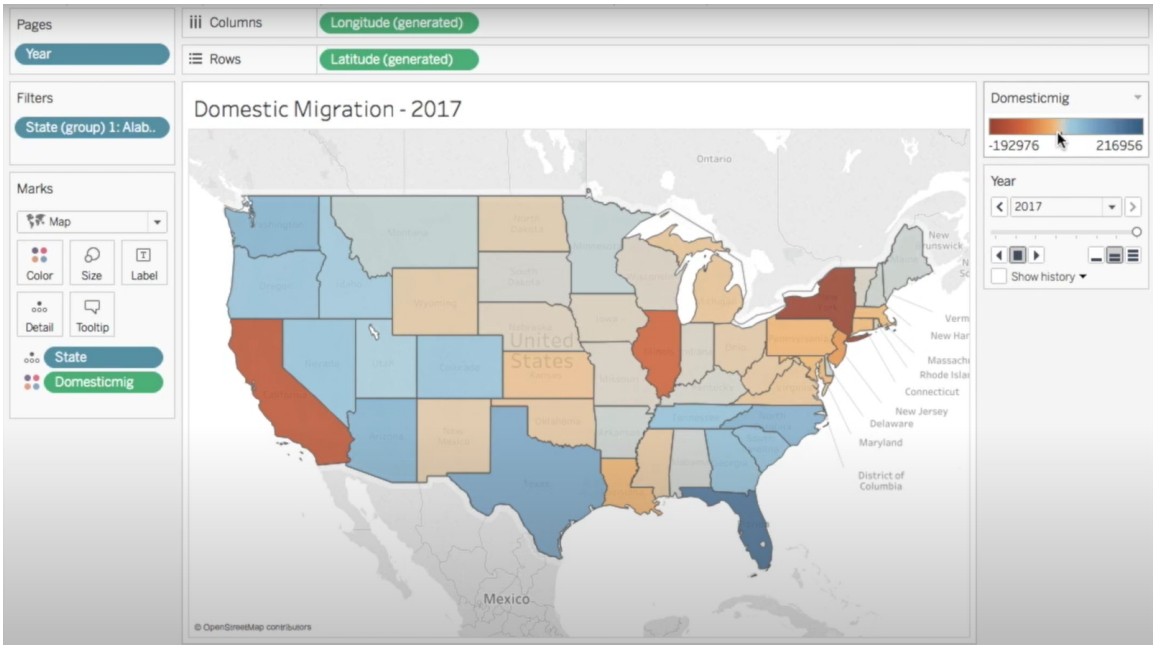

Figure 3.3: The example of Tableau animated map.[11]

3.2.3 Grafana

Grafana is an open-source web application for visualizing data using charts, graphs, and alerts. While Grafana is mainly a tool for creating interactive dashboards, it also supports creating thematic maps. However, thematic maps are not available out of the box but are available via Grafana plug-in system. A disadvantage of Grafana is that the users cannot upload their custom data, but the data is fed into Grafana via configured data source (e.g., InfluxDB[12]). The data mapping is done via creating a query that can be configured using Grafana's graphical query creator. As mentioned, Grafana comes with no support for thematic maps out of the box, and the same holds for creating animated maps. Although, this functionality can be added using the plug-in system. One example of such a plug-in is GeoLoop[13], an example of Grafana's panel using this plug-in is shown in Figure 3.4.

[11] https://youtu.be/7-sVGqwJyQ8?t=329
[12] https://www.influxdata.com/
[13] https://grafana.com/grafana/plugins/citilogics-geoloop-panel

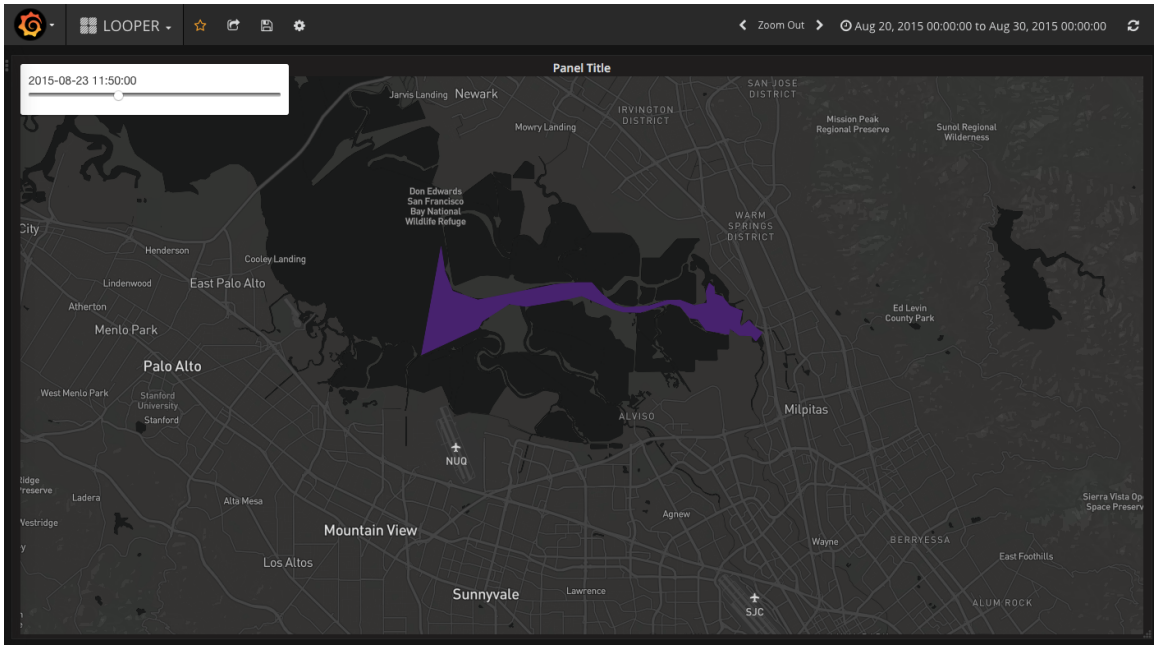

Figure 3.4: The example of GeoLoop plugin.[13]

3.2.4 Mapbox Studio

Mapbox Studio is an application for creating custom map styles and managing geospatial data. Users can create new layers on top of the map. It is possible to create custom thematic maps, such as choropleth. Mapbox Studio supports various formats: Shapefiles, GeoJSON, CSV, and more. Uploaded data is converted into vector tiles, which is the Mapbox format for handling data. Mapbox Studio includes a dataset editor, which allows the users to manage and edit datasets. It is possible to reuse existing component styles. The disadvantage of creating the thematic maps is that the users need to configure everything by themselves. For example, when creating the choropleth, users need to specify the ranges of all the classes. Unfortunately, Mapbox Studio can only be used for creating the component styles. The component styles can be exported for use in Mapbox's SDKs and APIs, embedded into source code, or exported as images. Mapbox Studio does not support visualizing geospatial time series, but the users can create the thematic maps and implement the support for a temporal component using Mapbox GL.

3.3 Geovisto

Geovisto is a toolkit for generic geospatial data visualization. It is a compromise between the two previously described approaches. It is primarily a programming library with the possibility of being used as an authoring tool for preparing configurations. The main idea of Geovisto is presenting the same data in different perspectives, but in one visualization with multiple data layers [12]. The base layer uses the Leaflet library to provide an interactive map layer. All of the other visual layers that are used for rendering thematic maps are rendered on top of this base layer as SVG elements.

The application specifies polygons and their centroids of geographic regions. Each of these polygons is associated with a unique identifier. The identifier of the polygon is used

in map layers' configurations to map the data to specific geographic objects. By default, the application uses the specification of world countries. The programmer can replace this configuration with a custom GeoJSON file to specify custom regions and centroids of the map.

The advantage of Geovisto is that it allows regular users to create geospatial visualizations with no programming knowledge and configure them using the web application. Moreover, Geovisto itself can be extended by developing a custom extension, such as an additional thematic map layer.

Geovisto offers limited support for visualizing geospatial time series. The possibility to visualize geospatial time series can be done by filtering specific time moments using the filter tool, which is then visualized.

3.3.1 Architecture

Geovisto has a modular structure. It consists of a core module and several tool modules. The modular structure is achieved using *events*. The core dispatches events to which the tools can subscribe. This way, each module is independent and has no dependencies on other modules. The core is also responsible for handling the life-cycle of Geovisto, e.g., it creates and initializes specified tools. It processes input data, initializes, and creates the Leaflet map and its state. The tools represent controls, toolbars, map layers, but they can also do additional data manipulations. The tools implement a standardized interface and subscribe to events. They can be enabled or disabled and configured in a control panel in the Geovisto application. Geovisto takes four types of inputs:

- Input data: The input data is mapped to dimensions of the visualizations. The supported format is GeoJSON.
- Polygons and centroids description: Descriptions of map regions. The polygons are used by choropleth to map the input data to its regions, the connection map uses the centroids to map the nodes, and the marker map uses them to map the markers.
- Properties: Properties allow to programmatically change the behavior of Geovisto, such as map state, provide own tools or redefine the behavior of map or tools.
- Configuration: The configuration allows regular users to override the default state of the map or the tools. The configuration can be exported during the run time of the application.

3.3.2 Thematic Maps

Currently, Geovisto provides three thematic maps (users can implement new ones as an additional tool): choropleth, connection map, and marker map. All three of them enabled at the same time are shown in Figure 3.5.

The thematic maps are customizable within the side panel in the Geovisto application and can be enabled or disabled. This allows creating a unique view of the data according to specific needs. The thematic maps use the polygons and centroids provided by the user. This allows users to create visualizations of the whole world or only of regions of a specific country. Since the thematic maps are implemented as Geovisto tools, the users can create new thematic maps and use them in Geovisto.

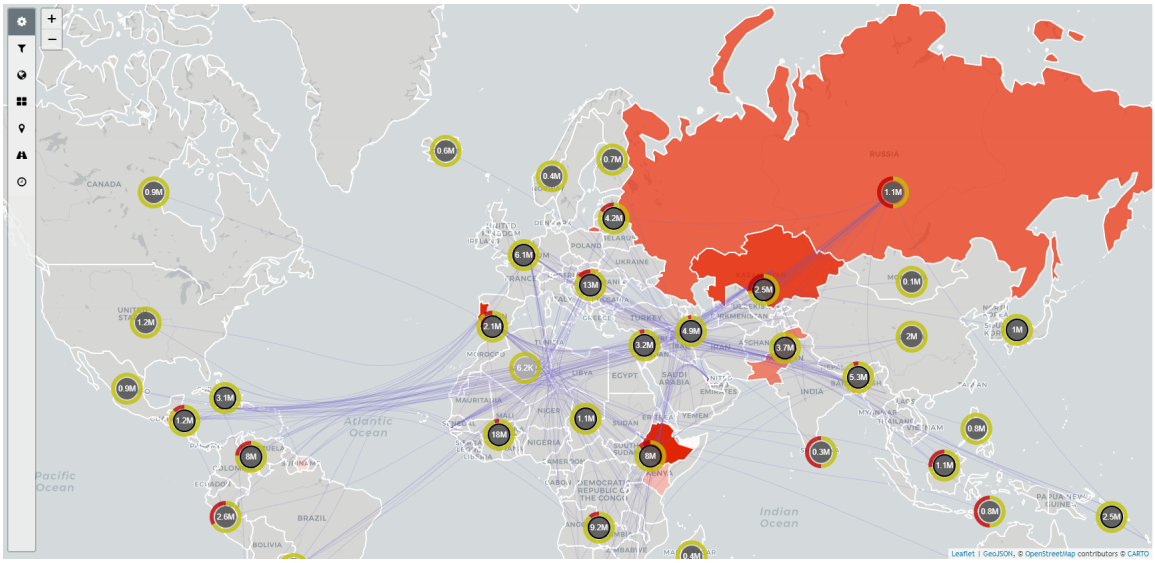

Figure 3.5: The thematic map layers of the Geovisto application. At the moment Geovisto has three integrated thematic maps: marker map, choropleth and connection map.

Chapter 4

Analysis

This chapter analyses the problem of visualizing geospatial time series from the user's perspective. The first section of this chapter introduces general use cases that potential users might face. The second section analyses the problem from the technical standpoint, defining the technical requirements to solve the use cases. Then, the last section of this chapter looks at existing tools, whether and how they solve the requirements defined in this chapter.

4.1 Usage Examples

This tool is aimed to be used by the general public and is not aimed at one specific group of users. Therefore, it is not possible to identify specific personas. However, it is possible to define two general use cases in which this tool will be used: presenting data and studying data. Both of these use cases solve different problems and are used by different types of user groups.

4.1.1 Studying Data

Studying of geographical phenomena is mainly done by experts in their particular fields. For example, data representing import and export trends of the world countries would be generally studied by economists. The evolution of carbon emissions would be a point of study for meteorologists.

When studying the data, there is not such an emphasis on the animations. The main focus is to be able to study and to compare the data effectively. Therefore, there is the need for an interactive time-player that allows navigating through time and pre-processing options of the data before they are visualized. Sometimes, the whole time range of the data might not be relevant, and the users might want to select only a specific time range. Other times, the users might want to aggregate the data to view the data in different time granularities, such as days, weeks, months, and others.

The studying use case requirements can be summed up into having high configurability, customizability, and practical tools for manipulating time. This allows creating a wide variety of different visualizations of one dataset.

Real World Example

A real-world example is the evolution of global carbon emissions. Carbon emissions started around the year 1750 in the UK. That is approximately 250 years of data. In this case, the users need an easy way to navigate such a large amount of data to study the phenomenon effectively. It is also vital for the users to limit the data to study only the interval that is relevant for their studies.

4.1.2 Presenting Data

Another general use case is presenting the geospatial time series. One example of a user group included in this use case is journalists. They will be used as an example to describe this use case.

Journalists present facts. Presenting facts in the context of geospatial time series means describing the state of some geographical phenomenon in time. In order to do this, journalists want to prepare a view of the data for each time moment and define transitions between them. Defining view of the geospatial time series is essentially creating animation that tells a story. Then, the generic use case for journalists is having some geographical phenomenon that they want to present. Therefore, they set up and configure the thematic maps and create an animation that tells a story. Since the main focus of this use case is presenting the data in animations, it is essential to let the users configure the animation to their specific needs. They might want to change the transition duration, time step length, or shift the focus of the view to a different entity of the geospatial data (e.g., shift focus from the USA to Europe).

Real world example

One example of visualizing geospatial time series is visualizing disease spread. Figure 4.1 shows a visualization of daily new cases of Covid-19 in Czechia. The time-player, located in the top-left corner, manipulates and navigates through time. When the time-player is in play mode, the time automatically flows, and the choropleth map changes its values of the attributes of its entities based on the data in the particular time moment.

4.2 Functional Requirements

Having analyzed the use case examples described, I identified functional requirements that are necessary to create an efficient tool. The requirements can be categorized in the following way: time component, processing generic data, visualizing change, animation configuration, and platform requirements.

4.2.1 Time Component

When working with the geospatial time series, the most important part is the temporal component. It is needed to create the timeline from the data that the users want to visualize. For example, when presenting data, the time should be in an ordered sequence, and the animation played from time 0 till the end time. On the other hand, when studying the data, it is convenient to choose the time moments manually, meaning the users can switch through the time in an ad hoc manner.

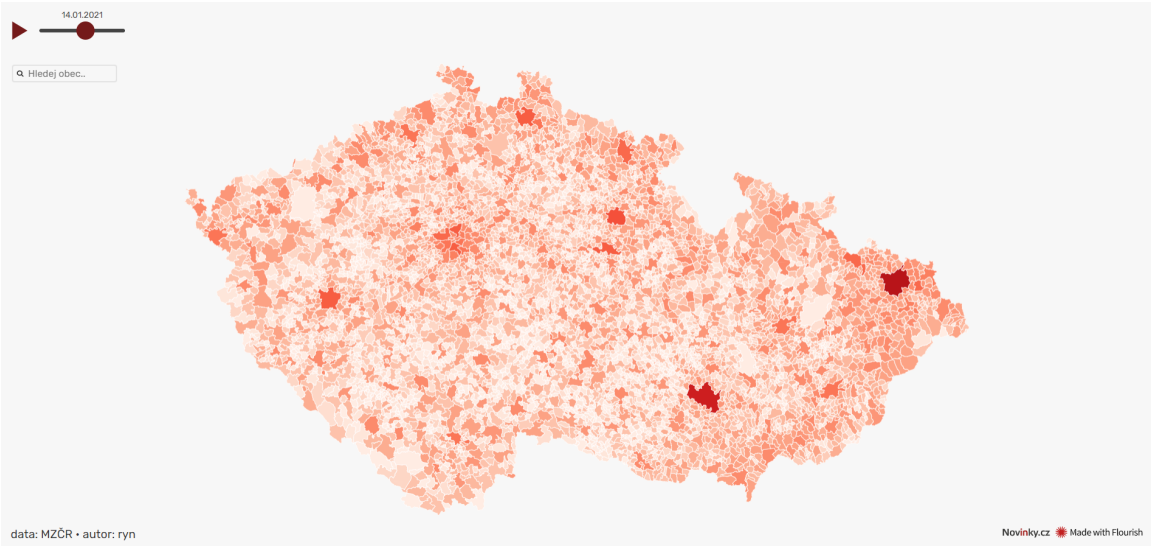

Figure 4.1: The choropleth map with a time player of daily Covid-19 new cases in Czechia. Picture taken from visualization published on the news portal—www.novinky.cz.[1]

Another common issue is that the users are not interested in the whole time period of the data. Sometimes the users need or want to select a specific time range within the whole time interval of the data. This provides more flexibility as they can filter out unnecessary information.

4.2.2 Processing Generic Data

One of the assumptions when studying data is that the users already have the data. Let us assume the data are in JSON format. The users usually want to do as little pre-processing of the data as possible. Therefore, the tool should provide a way to map the dimensions of the time and the thematic maps. An example of this can be additional parsing of the time, such as choosing a specific time window of the data or just mapping the time dimension of the data.

It is convenient to export and store the configuration of the set up visualizations and animations. Once the configuration is exported, it can be used for setting up the visualizations and animations from the exported one. Thus, it is possible to share the configured visualizations and animations between users of the application or store them and reuse them.

4.2.3 Visualizing Change

One important aspect when animating the change is the transition itself. While the transition animation can be visually appealing, the main focus should be on the ability of the transition to communicate what type of change happened.

In the context of geospatial data, the change is happening either within or between geospatial entities. Two types of changes that can be identified are as follows:

- **Attribute of an entity changes**, for example, the population of a country changes each year.

[1] https://flo.uri.sh/visualisation/4034499/embed

- **Relationship between entities changes**, the example of this is an import of a product from country A to country B. Such a relationship can be named as A imports to B. When country A stops importing to country B, the relationship stops existing, and vice-versa.

4.2.4 Animation Configuration

In order to make the animation more interactive and descriptive, the users should be able to configure it. For example, sometimes, there should be a focus on a particular part of the map. A real-world example can be the spread of a disease that starts in some countries and then spreads to the surrounding countries. In this case, the data are only in a specific region of the map. Hence, it might be helpful to be able to specify camera viewpoints for different time moments.

Furthermore, the users should be able to specify parameters of the transitions, such as transition length or transition delay. That allows them to emphasize essential changes or lower cognitive overhead if too many changes happen at once.

4.2.5 Tool Usage

The idea is to create a programming library that would allow users to create custom user interfaces for geospatial visualizations. Furthermore, the library should be abstract and extensible to allow users to implement new thematic maps. This programming library should also have support for working with time, implemented as a tool of the programming library. The tool should provide UI components that would be used to configure it. Although, the configuration could also be done programmatically by creating a configuration file.

4.3 Existing Tools

Programming libraries and geospatial frameworks introduced in Chapter 3 do not provide the support for visualizing geospatial time series out of the box. However, it is possible to use them and to create a custom-made solution that would satisfy all the specified requirements in this chapter.

The authoring tools described in 3.2 offer little support for visualizing geospatial time series. **Tableu** allows mapping of the time dimension and provides a time-player so that the users can play the time series in a sequence. This time-player also allows selecting a specific time instance. **Grafana** does not provide support for time series. However, its functionality can be extended using plugins. Granafa's plugin GeoLoop offers support for visualizing time series. Overall, these two authoring tools offer very basic support for the visualization of geospatial time series. The time-player functionality is limited, the configuration of animations is also very basic, and offers minimal options. **Mapbox Studio** comes with no time-player tool, and the users have to use Mapbox GL (which is a geospatial framework) to develop a custom solution. Although all of the mentioned authoring tools offer very limited support for visualizing geospatial time series, they allow creating static maps to visualize specific time moments of the time series.

Geovisto also does not provide any support for visualizing geospatial time series. Moreover, as with other authoring tools, it provides only the possibility to create a static map for a specific time moment. The Geovisto functionality can be extended using tools (plugins). Users with programming knowledge are able to create custom tools by themselves.

4.4 Conclusion

This chapter introduced general use cases in which the tool for visualizing geospatial time series can be used. The chapter also lists technical requirements, which were defined using the general use case scenarios.

From the analysis of the existing tools, none would satisfy all the specified requirements. Therefore, it is needed to develop a custom solution. Tableau is not open for extension, so it is not possible to use it. Grafana can be extended using plugins. However, Grafana is not suitable for creating custom visualizations for custom input data. Another possibility is to use Geovisto. Geovisto provides UI for users to create custom visualizations. It is also open for extension using custom tools, which work as plugins. Thus, making it possible to create a tool for time support. And although it is possible to use programming library (e.g., Leaflet or MapboxGL) by using Geovisto and developing custom tool for it, it is not necessary to develop the solution from scratch.

Chapter 5

Design

In this chapter, I will design a solution to address the defined technical requirements from Chapter 4. The core of the designed solution is a clock. The clock ticks in an interval and can be controlled through the interactive time-player. Another UI component of the designed solution is the control panel to configure parameters of the time and the animations. The design also describes how the change should be animated, user-created stories, and the data model of the solution.

5.1 Clock

The application needs an analogy to a clock. The clock ticks in a specified time interval, updating its current time. The sequence of time instances, over which the clock ticks, can be created as follows:

- **A sequence of time instances**: The time is created from timestamps of the data records by putting them into an ordered sequence. The order is from the lowest timestamp to the highest one. The step between the time instances does not have to be of the same value.
- **Real time**: It takes the lowest and highest timestamp from the records and generates all time instances in this interval. The users are able to select the granularity of the instances (e.g., minute, hour, day, week) and an aggregation function to aggregate the values in these generated time intervals. Using real-time in combination with the aggregation function allows creating different views of the data, such as visualizing the change daily or monthly.

The clock is also responsible for propagating the current time into other components, such as the map that visualizes the data and the time-player component.

5.2 UI Components

The UI components provide users with controls to manipulate the clock and configure the clock's parameters, visualizations, and animations.

5.2.1 Control Panel

Some parameters of the application need to be configurable by the users. This should be done from the central place in the application, such as a control panel. The configurable parameters allow users to create custom animations or views that would suit their specific use cases. Figure 5.1 shows a mockup of the whole application, with a control panel that allows the users to configure the parameters defined in this section.

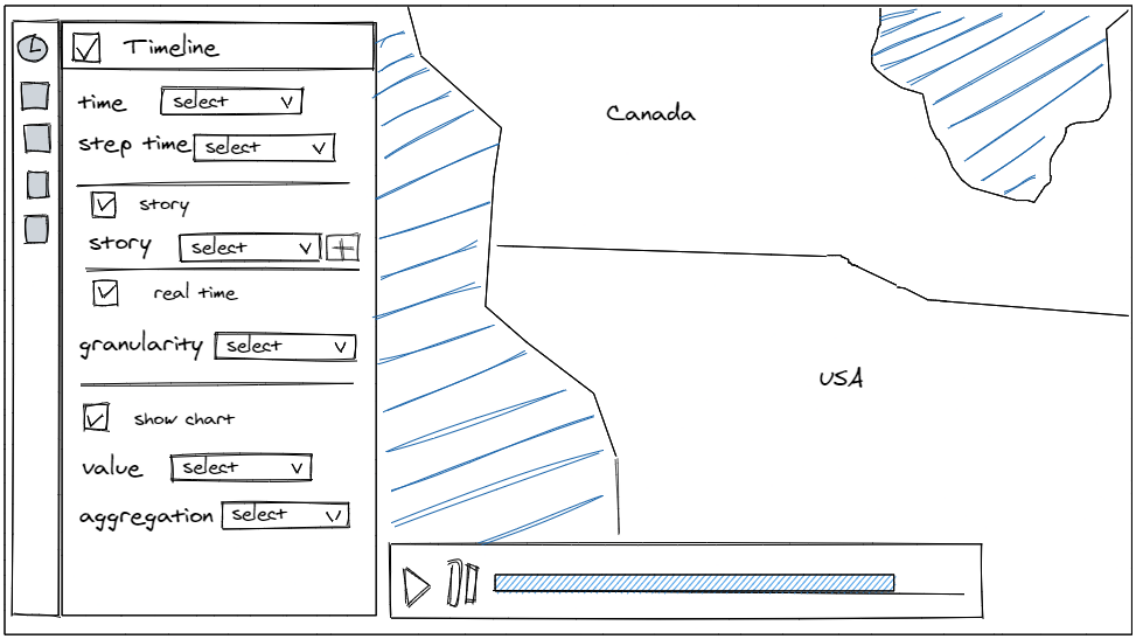

Figure 5.1: The mockup of the application. The mockup includes a control panel (for configuring the thematic maps and the time clock), an interactive time-player and a map layer.

Time Dimension

It is needed to map a time dimension from the input data. This can be done using a select component that will list all dimensions of the data. The users should be able to choose only the dimensions that can be converted to date type. Therefore, the input data needs to be parsed and valid time dimensions identified, so that they can be listed.

Time Step Length

The time step specifies the time interval between each tick of the clock. This can be overridden for a specific time moment when using stories, which is described in Section 5.3.

Animation Transition Duration

The animation transition duration defines how long the change from one state to another state takes. The value can be overridden for a specific time moment when using stories.

Data Aggregation Strategy

This corresponds to the characteristics described in Section 5.1. The users can choose between two values. If the real-time value is enabled, it generates the time instances between the start and end times. The frequency of the generated time instances can be selected in *granularity* select (minute, hour, day, week, month, year). Also, the *aggregation function* can be selected. The data with the same time is aggregated based on the specified function.

Chart

It represents a logical value indicating whether a chart should be rendered or not. Users can map the value dimension that is to be visualized in the chart. They can also choose the *aggregation function* (sum or average) that is used to aggregate the values of the same time. The chart provides another channel of communicating information about the data, such as overall trends. As shown in Figure 5.2 the chart is rendered inside the time-player tool.

5.2.2 Interactive Time-player Tool

The time-player tool is used to control the time and to work with the timeline so that the users have more flexibility in examining specific time instances.

The time-player should have the following controls:

- play, pause: used to control the automatic flow of time
- slider representing the time sequence: used to select specific time instance and also to reflect the selected time instance; the slider is interactive, and when playing, it is changing with each tick of the clock
- slider to select the time range: used to select start and end times of the time series

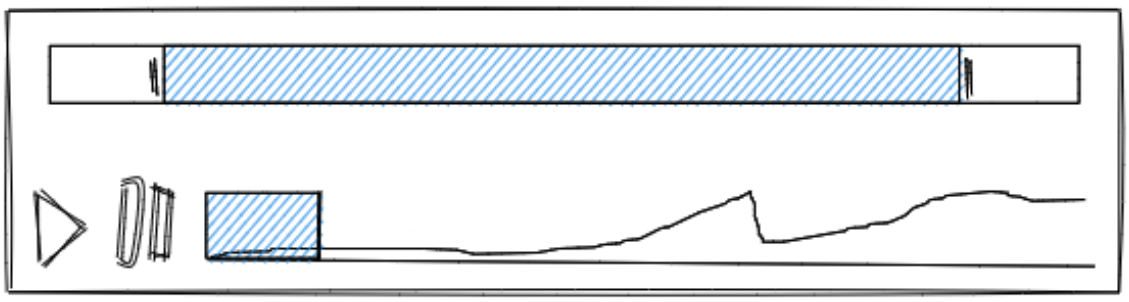

Figure 5.2: The mockup of the time-player tool. The time-player consist of the main interactive timeline which also displays chart. In the upper half is a time range selector.

5.3 Stories

The stories allow the users to configure the transitions of the animations for specific time steps. This makes it possible for the users to tell a story with the created animations.

The stories are snapshots of customized views in specific time moments with configured transitions into the following views.

When creating the story, the users should be able to save the current camera position. The camera position consists of longitude, latitude, and zoom. If the transition is made between the views with a different camera position, the camera transition of the view should be animated. The users should be able to configure how long the animation takes and the value of the time step. Last but not least, the users need a way to define how long the transition animation of the thematic maps will take, including the transition delay.

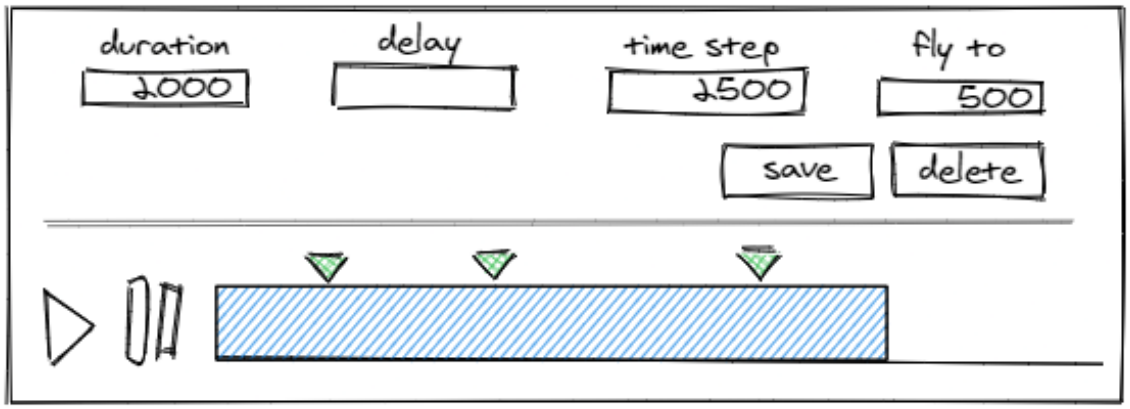

Figure 5.3: The mockup of the time-player tool with the configuration panel managing a story. The green triangles above the timeline indicate the times of the story snapshots.

Once the users specify all the parameters for the current view, they should be able to capture the state of the view in the current time step. When the animation plays and the time switches to a time step that contains a story snapshot, the transition animation will fire off with the defined parameters.

5.4 Animating Change

The most important part is to animate the change between each time moment. As analyzed in Subsection 4.2.3, two types of changes can occur: change in entity's attribute or change in the relationship between two entities. By designing the stories, another change needs to be animated and that is the change of camera position in the map.

5.4.1 Animating Change of Entity's Attribute

An attribute of an entity can be represented in numerous ways. To name a few, the value can be represented by texture, pattern, shading, and color. All these are also defined as animation variables in Figure 2.3.2. Therefore, animating the change of attribute means animating the change of these attributes.

As an example, choropleth uses color to visualize attribute values. When the value of an attribute changes, it is needed to animate the change of the color as shown in Figure 5.4.

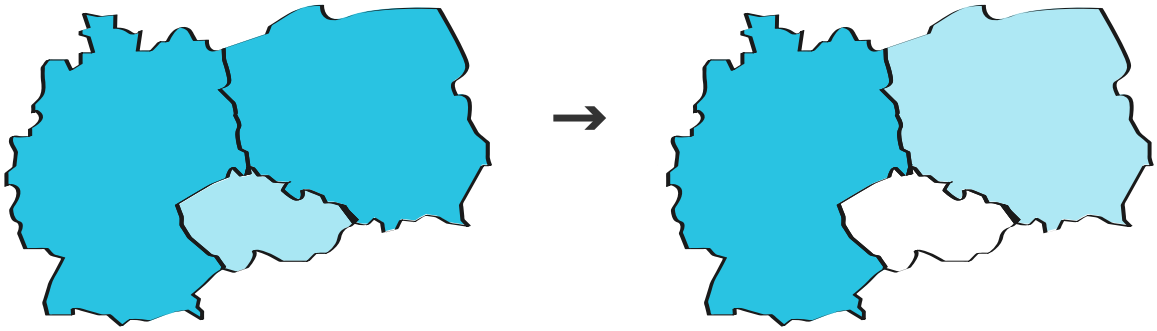

Figure 5.4: The visualization of the change in a country attribute value. The figure on the left presents the state before the change—all countries (Czechia, Germany, and Poland) have an attribute with a value. The figure on the right shows the state after the change. Germany has the same value, Poland has a lower value, and Czechia has no value. The different value of color opacity represents the change.

5.4.2 Animating Change in Relationship

The relationship between two geospatial entities is generally visualized using a connection map. The connection consists of source and target points. The connection between *source* and *target* represent the relationship. The relationship can be: country A imports to country B, cyberattack was done from country A to country B, the virus spread from country A to country B, etc.

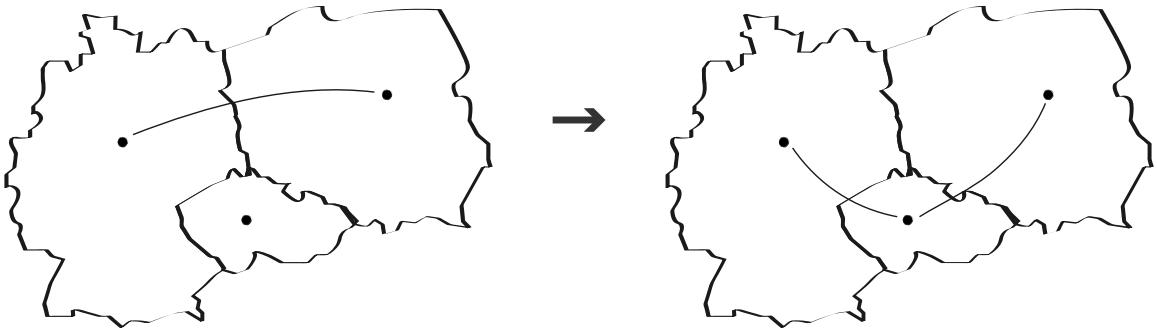

Figure 5.5: The visualization of the change in the relationship between countries. The figure on the left presents the state before the change—the relationship between Germany and Poland. The figure on the right (after the change) shows the change in the relationship—the relationship between Germany and Poland stops existing, and Czechia has new relationships with Germany and Poland.

There are two types of changes that can occur: the relationship starts existing or stops existing. A simple example of these two types of changes is visualized in Figure 5.5. To animate the change, it is needed to animate the creation and removal of the relationship. That can be done by removing and adding the relationship to the visualization. One thing that can help the users notice what changes occurred is to use the *opacity* property of the connection. By fading out the connection on removal and vice-versa, the users see what change is happening, thus lowering the cognitive overload.

5.4.3 Animating Camera Viewpoint

Since it is possible to define different camera viewpoints in the stories, it is important to animate the change. Such animation is non-temporal (as defined in Chapter 2). By animating the change, it lowers the cognitive overload, as the users have a better overview of the camera's current position. If the change was not animated, the users would have to realize the new camera's position themselves.

The camera position in a map consists of three parameters: latitude, longitude, and zoom. The three parameters reflect the animation variables *distance* and *viewpoint* (defined in Chapter 2). The most fitting animation to depict the change of viewpoint and distance is *fly to*. The *fly to* animation directly communicates the change of the camera's position. At the same time, the users do not lose the context of where the camera currently points to, which lowers the cognitive overload.

5.5 Data Model

When the users create a configuration of the application, they should be able to export and persist it. The exported configuration is stored in a JSON file. The configuration should include all the configurable parameters set from the control panel specified in Subsection 5.2.1. It should also include the configured stories.

The configurable parameters are stored as an object in the property named *configurableParameters*, each property of the object directly represents one configurable parameter from Subsection 5.2.1. The example of the JSON object of the configurable parameters is as follows:

```
"configurableParameters": {
    "timePath": string,
    "stepTimeLength": string,
    "transitionDuration": string,
    "storyEnabled": boolean,
    "storyName": string,
    "realTimeEnabled": boolean,
    "granularity": "HOUR" | "DAY" | "WEEK" | "MONTH" | "YEAR",
    "chartEnabled": boolean,
    "chartValuePath": string,
    "chartAggregationFn": string
}
```

The *stories* property is an array of objects where each object is one configured story. Each story must have the `name` property (identifies the stories) and the `states` property (holds the story's states in an array). The story state consist of unique timestamp (`time` property specifies for which time the state is used), camera viewpoint (the camera viewpoint is defined by: `zoom`, `latitude` and `longtitude`) and last but not least, the transition parameters (`stepTimeLength`, `flyToDuration`, `transitionDelay` and `transitionDuration`). The example of the JSON object of the stories is as follows:

```
"stories": [{
    "name": string,
    "states": [{
```

```
        "time": string,
        "zoom": number,
        "latitude": number,
        "longitude": number,
        "stepTimeLength": number,
        "flyToDuration": number,
        "transitionDelay": number,
        "transitionDuration": number
    }]
}]
```

The exported configuration can be reused by importing it into the application. This allows users to create the view and stories and export their configuration. The configuration can be either persisted for later use or can be shared with other users.

Chapter 6

Implementation

The solution for visualizing geospatial time series is developed as a tool for the Geovisto application. This chapter focuses mainly on the parts related to the tool and not the whole Geovisto application. This chapter describes how the solution was implemented from the description of the used technologies, the tool's UI components, and the tool's core functionality. Last but not least, the animating of the thematic maps is described.

6.1 Used Technologies

JavaScript is the language used for the implementation. Although parts of the implementation are in TypesScript because at the moment of the development, the whole Geovisto application was being refactored from JavaScript to TypeScript.

Geovisto uses the geospatial framework Leaflet to create the map and its controls. Therefore, this technology is transitively used by this tool. In this case, the Leaflet is used to create the UI control elements of the map and also, to some extent, in the Geovisto tools representing thematic maps.

The interactive time-player component is implemented in React. The component is added to the map as a Leaflet control. Library *react-compound-slider*[1] is used to create the time sliders, which work as time-player controls.

For the work with the time, *date-fns*[2] library is used. This library provides date utility functions for manipulating time.

D3.js (D3) is used for the creation of the thematic maps and their transitions. D3 is a programming library that allows the creation of visualizations of data. More about how D3 is used is described in Section 6.5.

6.2 Tool Architecture

The tool consist of three logical parts, the next sections describe each of them:

- **Tool UI** (Section 6.3): The tool consists of two UI components: the sidebar and the time-player. The sidebar component allows users to configure the time tool and initialize the time. The users are able to control the time through the time-player component. The time-player component allows configuring the time range and the stories.

[1] https://www.npmjs.com/package/react-compound-slider
[2] https://date-fns.org/

- **Tool Core** (Section 6.4): The core of the tool provides all the functionality that is needed to visualize geospatial time series. The main part of the core is the time clock. The clock ticks in an interval updating the Geovisto application data. When the data changes other tools are notified and handle the change.
- **Animations** (Section 6.5): The visualization of the change of time is not encapsulated in this tool. Each Geovisto tool is responsible for reflecting the change of the time itself. The animations are implemented as parts of existing tools representing thematic maps.

The overall architecture of the tool is shown in Figure 6.1. The class diagram visualizes dependencies between classes of the tool. The class `TimelineService` serves as the internal clock.

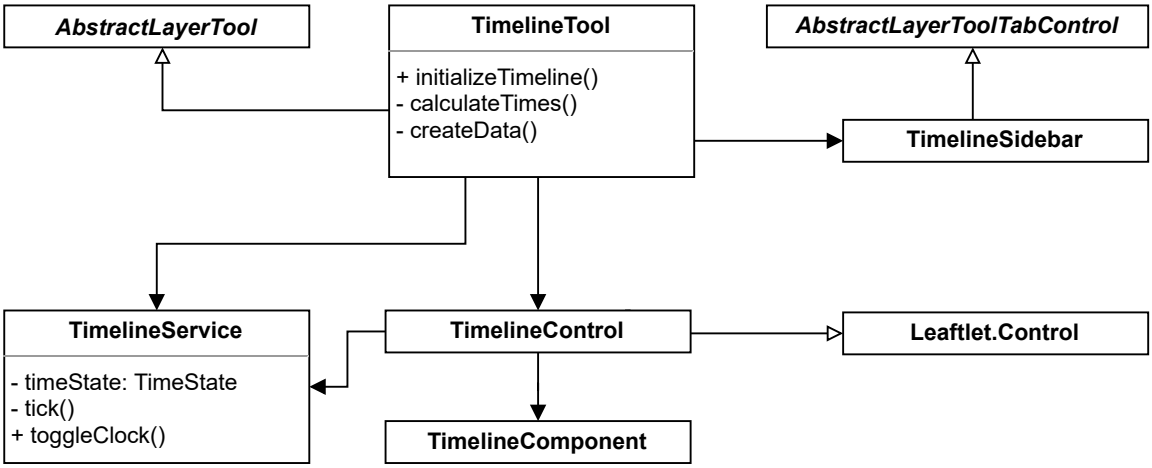

Figure 6.1: The overall architecture of the TimelineTool. The TimelineService serves as an internal clock. The diagram visualizes dependencies between classes of the tool.

6.3 Tool UI

The main UI components of the implemented tool are the interactive time-player and the sidebar. The interactive time-player component is used to control the time flow of the animations and the sidebar is used to configure the time.

6.3.1 Interactive Time-player

The interactive time-player is the main component for controlling the time in the application. It consists of four parts: time-player, range selector, chart, and story configurator. The expanded time-player component with all of its functionality enabled is shown in Figure 6.2.

The interactive time-player is implemented as a React component. Both the time-player and the range selector are created using *react-compound-slider* library, which provides unstyled slider component. The sliders represent time axis, allowing users to select particular times.

The React component is created and rendered in the `TimelineComponent` class. By wrapping the React component, it is possible to use it in an object-oriented way. The

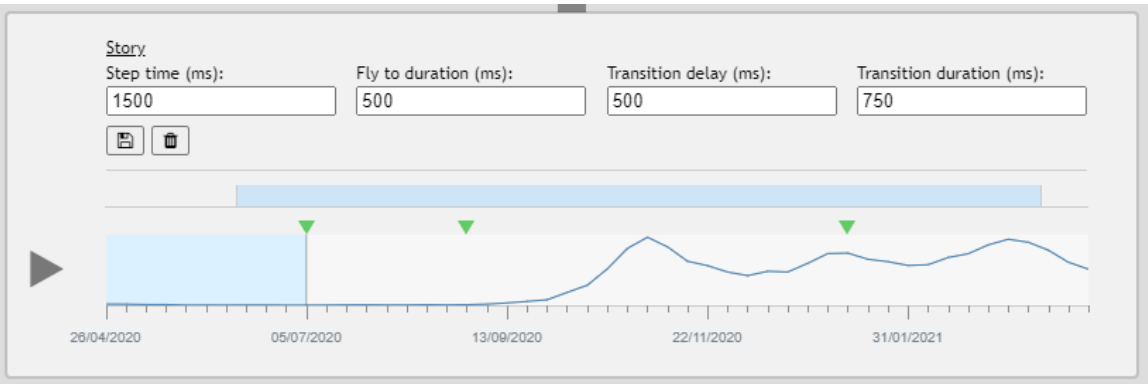

Figure 6.2: The interactive time-player with all its features enabled. It consists of four parts: time-player, range selector, chart and story configurator.

`TimelineComponent` exposes property setters, which mirror the properties of the React component. When a setter is called the React component is re-rendered. Moreover, by encapsulating the rendering of the component into the class, it is possible to replace the underlying component with custom solution. The `TimelineComponent` is used as a Leaflet control, which makes it possible to add it into the Leaflet map.

Time-player

The time-player displays all the time instances in a time-player-like fashion. The time-player allows for the manual selection of a specific time moment. On the manual selection, the time is propagated to other parts of the application. The automatic time flow can be controlled with the play and pause buttons. In the Figure 6.2 the *play* button is displayed and the time flow is stopped. When the *play* is clicked, the button changes to the *pause* button, and the automatic time flow is started.

The time-player also displays time labels for the time instances. The format of the time labels is dependent on the real-time aggregation function specified in the sidebar.

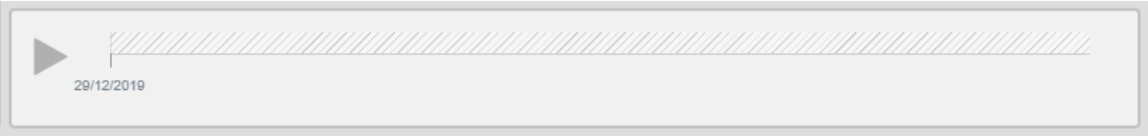

Figure 6.3: The time-player for only one existing time instance. The time-player is disabled and only shows the one time instance.

If only one time instance exists for the whole time, the interactive timeline is disabled and only shows the one time instance. This edge-case is shown in Figure 6.3.

Range Selector

The time range selector allows users to select a specific time range. The UI component displays the whole time range and the users can select two specific times. The two selected times form the time range.

If users select a new interval, and the current time of the clock is outside of this interval, the current time changes to either the lowest or the highest time from the range (depending on which one is closer).

Chart

The users have the possibility to display additional information using a chart. The chart is rendered in the background of the interactive time-player. *D3* library is used to render the chart as an SVG. The *x axis* of the chart is the time and *y axis* is the value. The value of each of the time moments is displayed in a tooltip on hover (shown in Figure 6.4).

Figure 6.4: The time-player with enabled chart. The tooltip displays a value for the hovered time instance.

Story Configurator

The story configurator is displayed when a story is enabled and selected. The configurator consists of four number inputs: time step length, fly to duration, transition delay, and transition duration. Users can create a new story state by selecting the specific time instance, for which they want to create the story state. After selecting the time instance and configuring the inputs the story state can be saved by clicking on the *save* button. On save, the new state is saved (or updated) in the story configuration. Which time instances have story state configured is indicated by green triangles above the time-player (can be seen in Figure 7.4). Users can delete any of the existing story states by selecting the time instance of the story state and clicking on the *delete* button.

6.3.2 Sidebar

The sidebar provides controls to enable and disable the timeline tool and also to configure it. The configuration is done using a form in the sidebar. The final implementation of the sidebar is shown in Figure 6.5.

The sidebar has a button *Apply*. This button is enabled if the form inputs are valid. On click, the time in the application is initialized and the time-player component is created and displayed.

Geovisto allows tools to implement custom sidebar tabs, which are added into the Geovisto sidebar (shown in Figure 6.5). The custom sidebar tab is implemented in TimelineToolTabControl. This class extends an abstract class provided by Geovisto (AbstractLayerToolTabControl). The content of the sidebar is created in the `getTabContent` method. By extending the class and implementing its methods, the sidebar tab is created and added into the Geovisto sidebar.

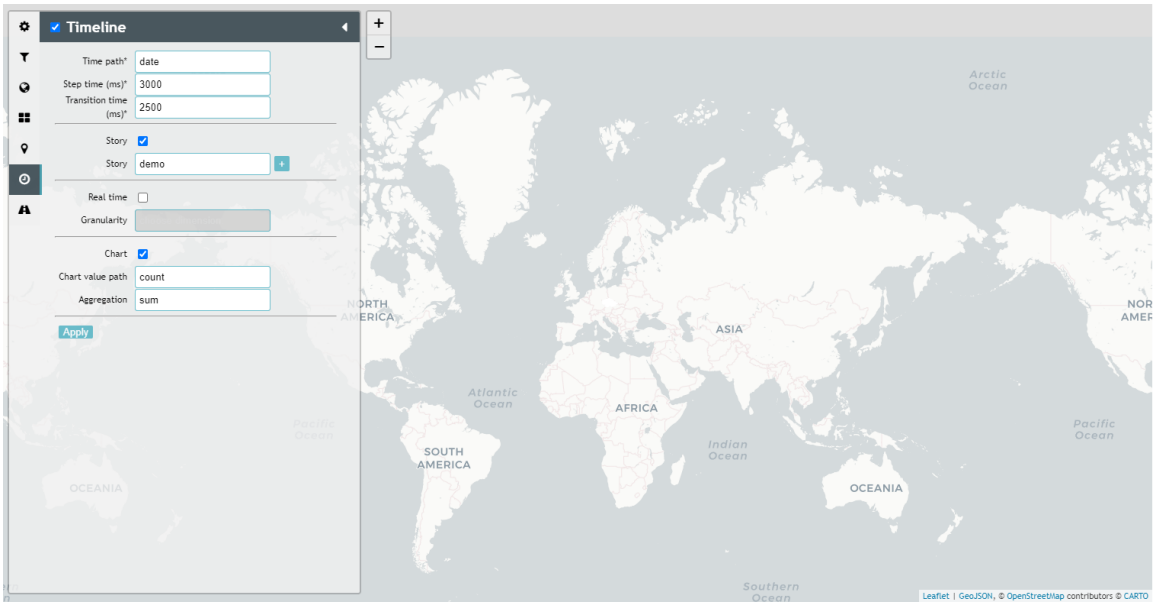

Figure 6.5: The screenshot of the Geovisto application with the opened timeline tool sidebar tab. The form in the sidebar is valid and the *Apply* button is enabled.

Configurable Values

The sidebar allows to configure the following values:

- time path: specifies dimension used to create the time, can be nested
- time step length: the value of time step length in milliseconds
- transition duration: the value of transition animation duration in milliseconds
- enable story: if enabled, users can select the story
- story: users can select an existing story or add a new one
- enable real-time: enables the real-time mode; if enabled, users can select the real-time granularity
- real-time granularity: select that specifies the granularity; items of the select are *hour, day, week, month, and year*
- enable chart: specifies if the chart is rendered in the timeline component
- chart value path: specifies dimension that is visualized in the chart
- chart values aggregation function: user can select the aggregation function that is used to aggregate values with the same time; function can be *sum* or *average*

Mapping Time Dimension

The time path select is populated with dimensions of which all records can be converted into valid JavaScript `Date` type. The valid time dimensions are identified by taking all values of a dimension. The values are iterated and being converted to `Date` one by one. If any of the conversions fail, the dimension is flagged as invalid and is not used.

Adding New Story

The story select is populated with the existing stories from the input configuration. Users can also create a new story by inputting the name in the select input and clicking on "+" button. This creates and selects a new story.

6.4 Tool Core

The time tool core implements the required functionality for handling time. The main important part is the clock. First, it is needed to create the times from the input data. This data is then used to initialize the clock. The clock, when started, ticks in an interval and periodically updates the data, that is to be visualized.

6.4.1 Time Initialization

The time initialization is started when users click on the *Apply* button in the tool's sidebar. The initialization phase is responsible for creating the timeline and preparing the data, but also for creating the clock and the tool's control component.

Creating the Timeline

In order to create the timeline, first, it is necessary to get all the unique times from the input data for the specified dimension. These unique times are ordered in an ascending order to create the timeline. At this point, if the real-time is disabled the ordered sequence of unique times is used. Otherwise, a time interval is created by taking the lowest and the highest time. After that, the times within the interval are generated. The time step between the times in the interval depends on the configuration of the real-time granularity in the sidebar.

In both cases, the times in the sequence are represented by the number of milliseconds since the Unix Epoch, this is done using `Date.prototype.getTime()`. The reason behind that is that when working with the timestamps (e.g., comparing them, using them as keys in objects, etc.) it is easier to work with a *number* than with a *Date* object.

Preparing Time Data

To effectively work with the input data records, the data records are preprocessed. The main reason behind this is the performance, as it would be necessary with each tick of the clock to filter the data records for the specific time moment.

The input data records are grouped by their unique time stamps and stored in a `Map` object. The keys of the `Map` object are timestamps in milliseconds, the `Map` is more convenient than `array` when accessing the data randomly (e.g., when manually switching time).

If the real-time is enabled, it is not possible to group the data records because the times in the timeline can be different from the timestamps of the data records. In this case, let us assume we have an ordered sequence of unique time stamps $I = (i_0, i_1, .., i_n)$. The data record is assigned to the timestamp i_n if *data timestamp* $\in [i_n, i_{(n+1)})$.

6.4.2 Time Clock

The responsibility of the time clock is to keep the state of the time and to provide the functionality to manipulate the clock. The state consists of the current time index, but the

state also has the start and end time index. The reason for storing the times as indexes is that it is easier to manipulate the time state. The actual time is then found by extracting the time from the specified index from the ordered sequence of times (this sequence is described in Subsection 6.4.1).

The start and end times represent the time range that can be set from the time-player component.

Time Tick

When the clock is started the time starts ticking. In the context of this tool, the tick of a clock means incrementing the current time index by one. That essentially changes the time to the next one in the time sequence. The clock ticks until it reaches the end time or until the clock is manually stopped.

The periodicity of the clock time tick is determined based on the parameter *time step length*, which is set in the sidebar. However, if the users set specific time step length for the current time in the story, then, the time step length of the story is used as the timeout before the next clock time tick.

6.4.3 Map State

The map state consists of the data that is currently used by the map and the tools. The state is stored in the class `GeovistoMapState`. The state has two properties (that are relevant to the time tool): filtered data and current data.

- Filtered data: Filtered data are processed input data. They are a superset of the current data and are set on initialization of `GeovistoMap` and by calling `updateData` method with `options.redraw` set to true.
- Current data: Subset of filtered data, is used for example by time tool when setting current data for a specific time moment. The current data are used by some tools to create transition animations.

6.4.4 Updating Data

When the current time of the clock changes, it is necessary to update the Geovisto data and notify other tools. The update of data happens on each time tick and manual change through the time-player.

Geovisto uses `DataChangeEvent` to indicate that its data in the `GeovistoMapState` has changed. This event is used also in this solution, but the event includes additional information, which is needed for creating the animations in other tools. The additional information is:

- `redraw`: Specifies the nature of the data change. When redraw is set to true, the tools rerender their visualizations from scratch. But when redraw is set to false it indicates *update* mode.
- `transitionDelay`: Specifies the delay before the transition animation is triggered. It can be overridden by defining different value in a story.
- `transitionDuration`: Specifies how long the transition takes, by default is 0. Users can specify this value either in the sidebar or in a story.

By extending the `DataChangeEvent` it is possible to encapsulate the animations within each specific tool. Section 6.5 describes how each thematic map layer handles this event.

6.5 Animations

All the thematic map layers subscribe to `DataChangeEvent`. Depending on whether the event specifies `redraw` or not, the layers either redraw the whole layer or just update the existing one. The difference between these two modes is subtle, but it is necessary to distinguish between them to create fluent transition animations.

6.5.1 Camera Viewpoint

The change of the camera viewpoint can be set when using the stories. Each state of the story can have a different camera viewpoint. When the next time instance has defined a different camera viewpoint it is necessary to animate the change. If not animated, the camera viewpoint would change instantly, which could lead to confusion what is the new viewpoint.

The Leaflet map API has a function `flyTo`. This function allows animatedly changing the camera viewpoint. The animation flies from the existing viewpoint to the new one. The fly to animation is triggered when the time changes. The duration of the fly to animation can be configured in the story configurator.

6.5.2 Connection Map

The connection map is implemented in the class `ConnectionLayerTool`. The connections are represented by SVG paths rendered between the source and the target nodes.

The connection layer uses the edge-bundling algorithm to bundle the connections. The bundling reduces visual clutter and is the responsibility of the class `D3PathForceSimulator`. It takes the nodes and connections and prepares the list paths which can be bent by the D3 force simulation.

The connection layer stores all currently rendered connection SVG paths in the property `connectionsPaths`. This property is an object, the keys are IDs of the country nodes with an existing connection. The currently rendered connections are used when updating the connections.

Redraw

When redrawing, the connections and the nodes are created from the current data. After that, the `D3PathForceSimulator` creates the SVG path definitions. Then, the paths are iterated over, rendering each path using D3 and storing each path in the `connectionsPaths` under the path ID.

After the paths are rendered, method `run` of class `D3PathForceSimulator` is called. This method runs the simulator, bundling the connections (rendered paths).

Update

When the `DataChangeEvent` is triggered the new connections and nodes are created from the current data. The transition duration and the transition delay (value is zero by default) from the `DataChangeEvent` are used to configure the transition animations.

When updating the connections, the newly created paths are merged with the currently rendered ones (stored in `connectionsPaths`). Although, if the paths that are currently rendered do not exist in the newly created ones, their value is set to an empty array. The reason behind this is to update the currently rendered connections correctly. If they do not exist in the newly created ones, they are updated with an empty array, which essentially removes them. This is a behavior of D3 which is used for rendering and updating the connections.

`D3.enter()` returns elements that need to be added. This way it is possible to target newly created connections and animate the addition. The animation is done in the following way. The new element is added but it has set `stroke-opacity` to 0. This way the element already exists and is rendered but is not visible. Then, the opacity is transitioned to value 0.4, creating a fade-in animation.

```
this.connectionsPaths[id].enter()
    .append("path")
    .attr("d", projectionPathFunction)
    .attr("class", `leaflet-layer-connection ${id}`)
    .style("stroke-opacity", 0)
    .transition()
    .delay(transitionDelay)
    .duration(transitionDuration)
    .style("stroke-opacity", 0.4)
```

The removal of the connections is handled in a similar way. Calling `D3.exit()` specifies the behavior for the connections that are to be removed. If the connection is to be removed, the connection fades out and is removed only after the transition is finished. The fade out transition is created by changing the opacity to 0.

```
this.connectionsPaths[id].exit()
    .transition()
    .delay(transitionDelay)
    .duration(transitionDuration)
    .style("stroke-opacity", 0)
    .remove()
```

After the update of the connections, the `D3ForceSimulator.run` method is called to bundle the rendered connections. The bundling is done while the connections are still in transition (fading in or fading out). This way the user does not see the animation of the bundling, but only a smooth transition between two states as the connections appear in the correct position.

If there is a transition delay set, the `D3ForceSimulator.run` is set as callback in setTimeout. The `setTimeout` has the timeout set to the transition delay.

Animated Direction

The user can turn on (in the sidebar) animation of the direction of the connections. This functionality is implemented using CSS styles. When the connection SVG is created, and the animate direction option is enabled, a CSS property `stroke-dasharray` is set to *"10 ,4"*. The `stroke-dasharray` is set on each SVG element representing the connections. This style makes the connections appear as dashed lines. Then, the offset is incremented in

an interval. The interval timeout is set from the sidebar. By incrementing the offset, the dashes in the dashed lines appear as moving. This depicts the direction of the connection from the source node to the target node.

6.5.3 Marker Map

The marker map visualizes markers on the map using a donut chart. Each region of the map has a marker (if the region has a value). The created markers are stored in the tool state. The reason for this is that when updating the markers, it is also needed to update the markers which may not exist in the current data. There are two types of markers:

- Base markers: The base markers are displayed on top of each map region. Their values consist only from the values of the region they represent.
- Cluster markers: These markers are displayed instead of other markers they cluster. The value of the cluster marker consists of the aggregated values of its child markers (the markers of the cluster).

It is not necessary to create any new markers when updating, because all of the markers are already created. However, it is needed to update the markers' values and the displayed donut chart. The update of the markers is done in the following order:

1. The values and donut charts of the base markers are updated. The base markers are updated even if they are not currently visible because the cluster markers use their current values.
2. All the cluster markers are flagged for an update. They are updated the next time they are displayed. E.g., no cluster marker is displayed when the map is zoomed in (no region is clustered), then, if the map is zoomed out the cluster markers are created and displayed. If they would not be flagged for an update, they would not recalculate their data and would not update their donut chart for the updated values.
3. All the currently visible cluster markers are updated immediately. It is necessary to explicitly update the visible cluster markers even if they are flagged for an update. If the cluster marker is flagged for an update, it is only updated the next time its creation is triggered.

The update of the marker is done in two phases. First, its values and tooltip are updated, then the donut chart is updated. The update of the donut chart is animated and is done using D3 functions. The following code snippet shows the D3 functions that animate the change of the donut chart:

```
this._svgGroup
    .datum(Object.entries(this.options.values.subvalues))
    .selectAll("path")
    .data(pie)
    .transition()
    .delay(transitionDelay)
    .duration(transitionDuration)
    .attrTween("d", function(newData) {
```

```
        const i = d3.interpolate(this._currentData, newData);
        this._currentData = i(0);
        return (t) => arc(i(t));
    });
```

The `transition` function specifies that the change should be animated. The `delay` and `duration` functions configure the transition parameters. The animation itself is defined in the `attrTween` function. The `attrTween` function returns a tween for the attribute transition, in this case the `d` attribute. The attribute `d` defines the SVG lines of the donut chart arcs. The returned `tween` function gets called for each time step of the transition, returning a new definition of the donut chart arcs. Each time step, the `d` attribute is updated with the new arc values of the given time step.

6.5.4 Choropleth

The choropleth layer is created using `L.geoJSON` from the configured polygons. The choropleth is classified and the difference in classes is visualized using different opacities for each class. The choropleth classes are recalculated on each redraw. There are two possible modes when creating the classes that users can choose in the choropleth sidebar:

- Scaling style: Supports four functions (median, absolute, relative, irrelative). In this mode, the choropleth classes are created from the current time. That means that the min/max is calculated on each time step only from the current data and not from the whole time range.
- Custom min/max: Users can specify the min and max values used for calculating the classes. This way it is possible to create classes that will reflect data within the whole time range and not just the current time.

After creating the classes, each SVG item of the geoJSON layer is assigned `fill` and `opacity` styles. On `DataChangeEvent` the choropleth items are updated. The difference between redraw and update is in the delay and the transition. When redrawing the transition is done instantly, while when updating the transition parameters are specified in the `DataChangeEvent`. The animation is done using CSS transitions. Each SVG item has set CSS style property `transition-property` to *"fill, fill-opacity"*, this specifies which CSS properties will use transitions. The transition itself is done in two steps. In the first step, for each SVG item the `transition-duration` and `transition-delay` are set to the values passed in the event's `options` object. These two properties configure the transition. After that, the `fill-opacity` is changed to reflect the current value.

Chapter 7

Testing

The testing of the developed tool has been divided into two phases. At first, I tested if the tool works in the Geovisto application since the tool was developed as a Geovisto tool. After that, I created three datasets with real-world examples and tested the tool itself.

7.1 Tool Integration in Geovisto

The Geovisto application is not a stand-alone app. It is developed as a NPM package and will be distributed via the NPM repository in the future. *Storybook*[1] was used to test the Geovisto. Storybook allows the development of UI components and pages in an isolation without the need to deploy the application. The Geovisto package exports React component to use the Geovisto application. This React component allows configuring the Geovisto through React properties. The configuration allows to specify tools that are used by the Geovisto. An example of a configured React Geovisto component is as follows:

```
<ReactGeovistoMap
    polygons={this.state.polygons}
    centroids={this.state.centroids}
    data={new FlattenedMapData(this.state.data)}
    config={new BasicMapConfig(this.state.config)}
    tools={new ToolsManager([
        new TimelineTool({ id: "timeline" }),
        new SidebarTool({ id: "sidebar" }),
        new ChoroplethLayerTool({ id: "choropleth" }),
        new MarkerLayerTool({ id: "marker-map" }),
    ])}
/>
```

In this examples, the `polygons`, `centroids`, `data` and `config` properties specify the data users can input. All of those properties are connected with input fields that were added into Storybook. Thus, making it possible to test the application with different inputs. Geovisto can be configured to use different tools. The tools are specified in the `tools` property. In this particular example, the timeline tool is used together with the sidebar tool and two thematic maps—the choropleth and the marker map.

[1] https://storybook.js.org/

For testing purposes, I have successfully created the new Geovisto application in the Storybook. The Geovisto application uses the time tool, which was configured through the React properties. The configured Storybook was used to emulate the usage of the solution as a standalone web application. A screenshot of the configured Storybook is shown in Figure 7.1.

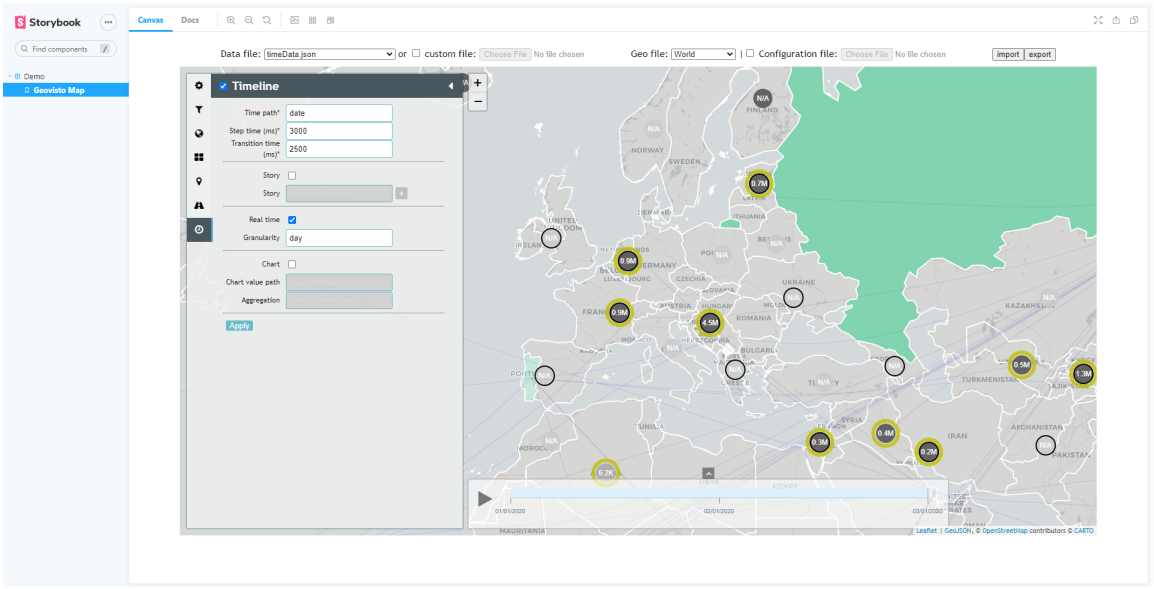

Figure 7.1: A screenshot of the Geovisto visualisation of world carbon emissions. The visualization uses choropleth and marker layer. The time is active and set to the last time instance.

7.2 Use Cases

The time tool was tested by using the use cases described in Chapter 4. For testing purposes, I created three datasets. The carbon emissions and Covid-19 datasets are based on real data, while the cyber attacks dataset was manually created.

7.2.1 Carbon Emissions

For the presentational use case, the carbon emissions dataset is applied. This dataset includes carbon emissions since the year 1750. This dataset uses the choropleth layer to show the value of carbon emissions and the marker layer to visualize the type of emissions (e.g., solid fuel, gas fuel, etc.). The configured visualization, including the time-player with the configured story, is shown in Figure 7.2. For this dataset, I created an animation story. The story shifts the camera view to different parts of the world, depending on which countries start to produce carbon emissions. For example, in 1750, only the UK was producing emissions. However, at the end of the century, the USA also started to produce emissions, so the camera shifts the focus to the USA. The created story was exported in a configuration file, and it was evaluated that it is possible to share it and import it back into the Geovisto application. Since there are different viewpoints for different times, the camera performs the *fly to* animation. It was necessary to configure the transition delay

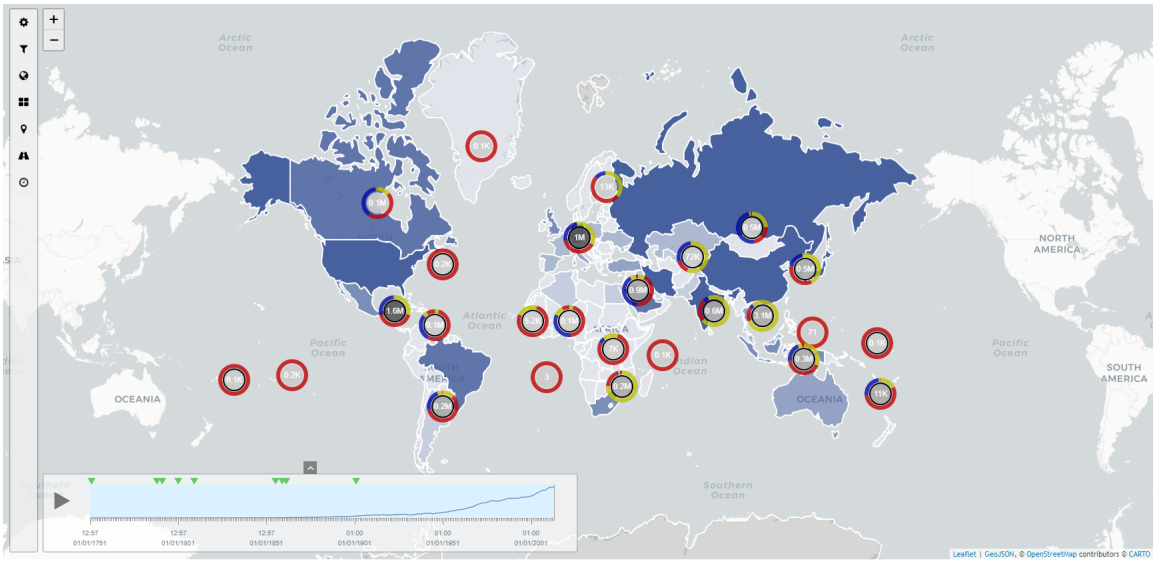

Figure 7.2: A screenshot of the Geovisto visualisation of world carbon emissions. The visualization uses choropleth and marker layer. The time is active and set to the last time instance.

and duration of the animations, as when the *fly to* animation was too fast, it was hard to keep track of the camera position. I used the range selector to visualize only the 20th century because, before the 20th century, there are not many countries producing emissions, so there is a long period without "interesting" data.

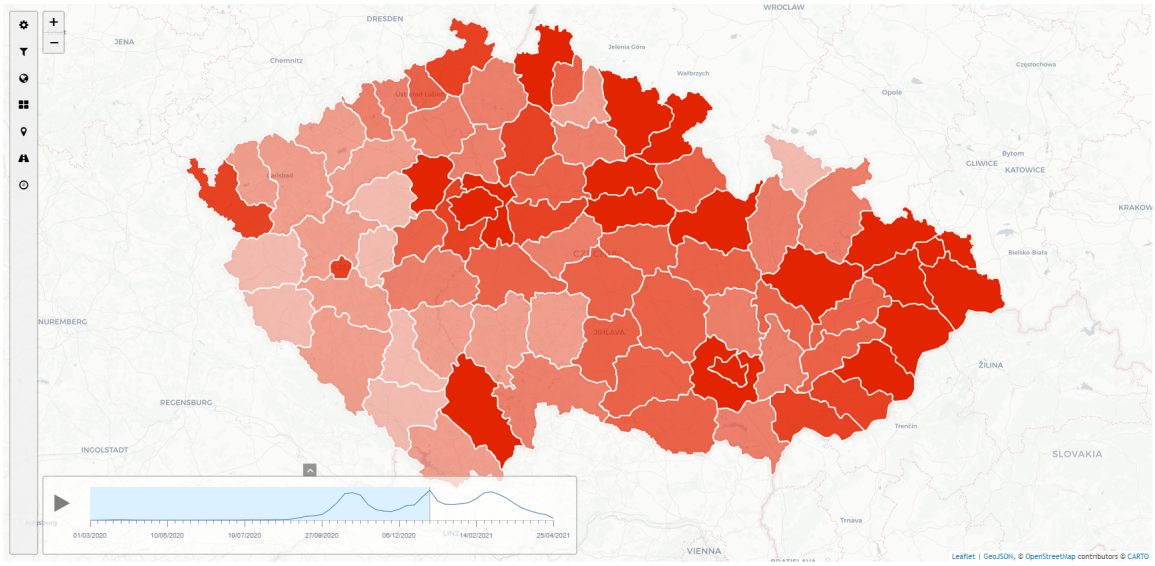

Figure 7.3: A screenshot of the Geovisto visualisation of daily new Covid-19 cases in Czechia. The visualization uses choropleth and has time enabled. The time-player displays a chart of daily deaths.

Change of Country Borders

The limitation I have encountered when testing the dataset was the change of country borders or country existence. An example of this is the split of Czechoslovakia into Czechia and Slovakia. This is the limitation of bigger datasets since, as of now, neither Geovisto nor the time tool supports this dynamics. It is possible to provide the polygons and centroids in the configuration, but it is impossible to swap them during the animations.

7.2.2 Covid-19

The Covid-19 dataset is used to demonstrate the studying use case. The dataset includes daily Covid-19 data in the Czech regions from the beginning of the pandemic until the end of April 2021. Since this dataset does not display world data, it was necessary to create custom geographical polygons and centroids for the regions. The dataset includes data for the territory of Czechia. Hence the custom polygons and centroids had to be created for these regions.

The configured visualization is shown in Figure 7.3. The choropleth was configured to display the number of new cases. No other tool is used but the choropleth and the time tool. The goal of this dataset was to study the evolution of the pandemic in Czechia, identify trends or correlations, and compare different periods of the pandemic. When testing, I tried different aggregation functions. They allowed me to study daily, weekly, monthly and yearly cases. By studying the data, I was able to identify a direct correlation between the total number of daily cases and daily deaths. I was also able to identify the most affected regions. As shown in the time-player chart (Figure 7.3), there were three peaks of the daily new cases.

7.2.3 Cyber Attacks

This dataset was used during the implementation phase as this dataset has all the data necessary to display all three thematic maps. Sources and targets of cyber-attacks are visualized in the connection map. The state of the attacks is visualized using the marker layer, and the number of attacks is visualized using the choropleth. When testing, all the thematic maps were enabled, as shown in Figure 7.4. I also enabled the filter tool, which allows focusing only on the specific state of the attacks. The visualizations worked as expected, and there were no issues while working with the time tool.

7.3 Limitations

During the testing, I encountered a few limitations. Solving these limitations would improve the time tool and the Geovisto application in general. The limitations and ideas for improvements are as follows:

- When the users create some story (e.g., animations), the only way to share them is to export the configuration. Other users can use this configuration, but they need to import it into the Geovisto application. In some cases, the users might want to share the animations with others who do not use the Geovisto application. This can be solved by exporting the configured animations into one of the video formats or GIF format.

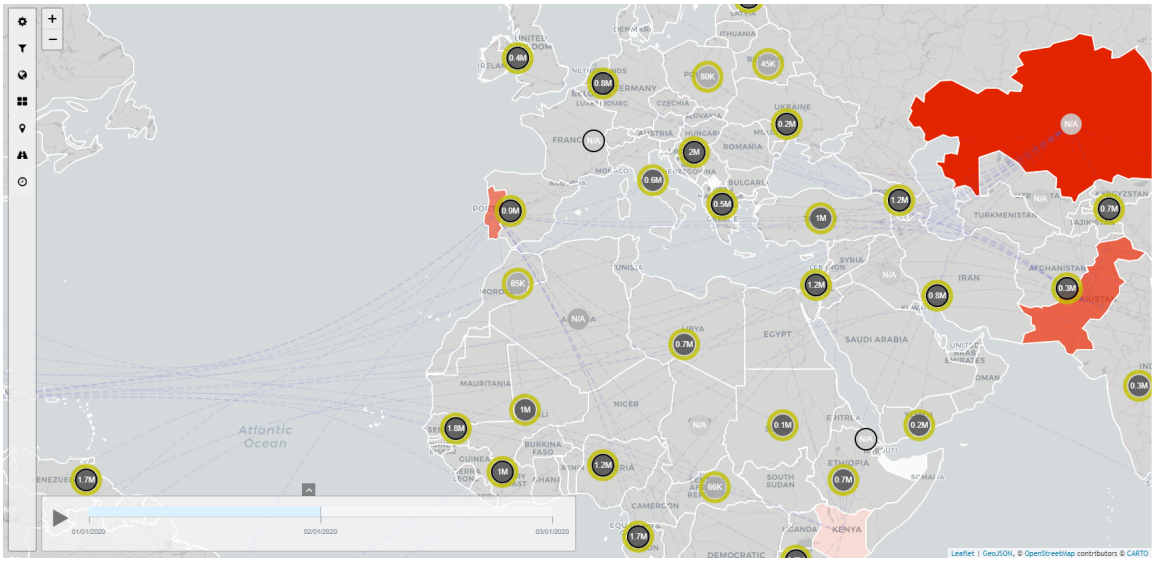

Figure 7.4: A screenshot of the Geovisto visualisation of daily cyber attacks between world countries. The visualization uses all Geovisto thematic maps—choropleth, marker map and connection map. The time is enabled and the data are filtered using the Geovisto filter tool, to display only mitigated attacks.

- When datasets span over a long time period, there is a problem with regions that change their boundaries (e.g., Czechoslovakia split into Czechia and Slovakia). In order to solve this, the users need to import more than one definition of polygons and centroids. Then, these polygons and centroids could be swapped in specific time moments.
- I noticed that when importing large datasets, there is a need to pre-process the data, which could take a long time. The idea would be to move these computations from the client-side to improve the performance and the user experience.
- Visualizing geospatial time series in animations, while effective, may not be the best solution for all use cases. The multiple static maps might be a better solution when comparing a few time instances. As of now, the users can open the Geovisto application several times and configure each instance of the application as one static map. It would be more convenient if the users could configure several map views in one instance of the application.

Chapter 8

Conclusion

The goal of this thesis was to create a tool for visualizing geospatial time series. There are many existing solutions to visualize geospatial data. Users can use existing authoring tools. However, those have limited support for working with the geospatial time series. It is possible to develop a custom solution for the visualizations, but programming knowledge is needed. Therefore, the goal was to develop a tool that would allow users to create custom visualizations of geospatial time series without programming knowledge.

I developed the solution as a Geovisto application tool. The developed Geovisto tool allows users to create custom visualizations and animations of geospatial time series while also being configurable for use with generic data. The tool provides necessary controls to manipulate the time and to create animated stories. The stories can be exported to be persisted or to be used by other users. Users with programming knowledge can develop new thematic maps as Geovisto tools and use them with the time tool to implement the animations in the new thematic map themselves. The Geovisto application, with the developed time tool, can be integrated into existing applications or can be used as a standalone web application.

I have created two datasets to demonstrate the usage of the tool in real-world use cases. The first dataset is daily new cases of Covid-19 in Czechia. This dataset was used to test the studying use case. I visualized the data using choropleth and used the time tool to effectively study the data and identify trends (such as the correlation between daily new cases and daily deaths). The second dataset is world carbon emissions which tested the presentational use case. During testing, I have encountered a few limitations such as slow pre-processing of large datasets and not being able to change polygons' GeoJSON definitions when playing stories. However, they do not affect the solution negatively. I have also come up with few improvements that would improve the user experience, such as an easier way for the users to persist and share the configurations or provide different visualizations of the time series (such as static maps).

I have successfully created a solution for visualizing geospatial time series. This tool will be distributed as part of the Geovisto NPM package in the future. Thus, users will be able to use it to create their unique visualizations or develop their own tools and use the time tool as a base for visualizing and animating new thematic maps.

Bibliography

[1] *Cartography guide: a short, friendly guide to basic principles of map design* [online]. [cit. 2020-22-12]. Available at: https://www.axismaps.com/guide.

[2] *The International Standard for country codes and codes for their subdivisions* [online]. [cit. 2020-25-12]. Available at: https://www.iso.org/iso-3166-country-codes.html.

[3] Andrienko, N., Andrienko, G. and Gatalsky, P. Supporting visual exploration of object movement. In: *Proceedings of the working conference on Advanced visual interfaces*. 2000, p. 217–220.

[4] Bertin, J. Semiology graphique. *Mouton, Den Haag*. 1967.

[5] Board, C. Report of the working group on cartographic definitions. *Cartographic Journal*. 1992, vol. 29, p. 65–69.

[6] Butler, H., Daly, M., Doyle, A. et al. *The GeoJSON Format* [RFC 7946]. RFC Editor, august 2016. DOI: 10.17487/RFC7946. Available at: https://rfc-editor.org/rfc/rfc7946.txt.

[7] Chien, T.-W., Wang, H.-Y., Hsu, C.-F. et al. Choropleth map legend design for visualizing the most influential areas in article citation disparities: a bibliometric study. *Medicine*. Wolters Kluwer Health. 2019, vol. 98, no. 41.

[8] Dodge, M., McDerby, M. and Turner, M. *Geographic visualization: concepts, tools and applications*. John Wiley & Sons, 2011.

[9] Few, S. and Edge, P. Introduction to geographical data visualization. *Visual Business Intelligence Newsletter*. 2009, p. 1–11.

[10] Harris, R. L. *Information graphics: A comprehensive illustrated reference*. Oxford University Press, USA, 1999.

[11] Hill, L. L. *Georeferencing: The geographic associations of information*. Mit Press, 2009.

[12] Hynek., J., Kachlík., J. and Rusňák., V. Geovisto: A Toolkit for Generic Geospatial Data Visualization. In: INSTICC. *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: IVAPP,*. SciTePress, 2021, p. 101–111. DOI: 10.5220/0010260401010111. ISBN 978-989-758-488-6.

[13] Kelly, B. Review of Unclassed Choropleth Mapping. *Cartographic Perspectives*. Nov. 2017, no. 86, p. 30–35. DOI: 10.14714/CP86.1424. Available at: https://cartographicperspectives.org/index.php/journal/article/view/cp86-kelly.

[14] Kraak, M.-J. and Ormeling, F. *Cartography: visualization of geospatial data*. Routledge, 2009.

[15] Ogao, P. J. and Kraak, M.-J. Defining visualization operations for temporal cartographic animation design. *International journal of applied earth observation and geoinformation*. Elsevier. 2002, vol. 4, no. 1, p. 23–31.

[16] Peña Araya, V., Pietriga, E. and Bezerianos, A. A Comparison of Visualizations for Identifying Correlation over Space and Time. *IEEE Transactions on Visualization and Computer Graphics*. IEEE. 2019, vol. 26, no. 1, p. 375–385.

[17] Peterson, M. P. Spatial visualization through cartographic animation: Theory and practice. In: *GIS/LIS*. 1994, p. 619–628.

[18] Rinzivillo, S., Turini, F., Bogorny, V. et al. Knowledge discovery from geographical data. In: *Mobility, Data Mining and Privacy*. Springer, 2008, p. 243–265.

[19] Rolf, A., By, R. de et al. Principles of geographic information systems. *The International Institute for Aerospace Survey and Earth Sciences (ITC), Hengelosestraat*. 2001, vol. 99.

[20] Schneider, M. Spatial data types: Conceptual foundation for the design and implementation of spatial Database systems and GIS. In: *Proceedings of 6th International Symposium on Spatial Databases*. 1999.

[21] Tennekes, M. Tmap: Thematic Maps in R. *Journal of Statistical Software*. 2018, vol. 84, no. 6, p. 1–39.

[22] Walford, N. *Geographical data: characteristics and sources*. John Wiley & Sons, 2002.

[23] Zhou, H., Panpan Xu et al. Edge bundling in information visualization. *Tsinghua Science and Technology*. 2013, vol. 18, no. 2, p. 145–156. DOI: 10.1109/TST.2013.6509098.