TU WIEN Informatics

Visual Comparison of Spatial Deviations via Geospatial Slicing

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Stefan Meusburger
Matrikelnummer 01526330

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. M. Eduard Gröller
Mitwirkung: Dipl.-Ing. Dr.techn. Johannes Sorger

Wien, 28. Oktober 2020

Stefan Meusburger	M. Eduard Gröller

TU WIEN Informatics

Visual Comparison of Spatial Deviations via Geospatial Slicing

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Medical Informatics

by

Stefan Meusburger
Registration Number 01526330

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. M. Eduard Gröller
Assistance: Dipl.-Ing. Dr.techn. Johannes Sorger

Vienna, 28th October, 2020

Stefan Meusburger	M. Eduard Gröller

Erklärung zur Verfassung der Arbeit

Stefan Meusburger

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 28. Oktober 2020

Stefan Meusburger

Kurzfassung

Datenwissenschaftler analysieren Patientendaten, um Korrelationen zwischen Krankheitsbildern und Umweltfaktoren wie Lärm und Luftverschmutzung zu identifizieren. Die große Anzahl an Attributen führt zu vielen möglichen paarweisen Korrelationsmodellen. Um die Qualität dieser vollständig zu analysieren, ist es wichtig, Fehler in der Modellausgabe, das heißt die Residuen, lokal zu vergleichen. Die große Anzahl an Datenpunkten erschwert dies jedoch. Darüber hinaus ist es wichtig, Korrelationen zwischen Geodatenwerten und Attributen zu identifizieren, um neue Modelle zu erstellen und deren Qualitätsindikatoren entlang der geografischen Achsen zu analysieren. Niedrige Residuen der Modelle und geringe Schwankungen entlang der räumlichen Attributachsen sind Indikatoren für hochwertige Modelle. Ziel dieser Bachelorarbeit ist es, eine spezielle Visualisierung im Kontext einer Geodatenkarte zu entwerfen und implementieren, welche Datenwissenschaftlern dazu dient, Attribute und die Qualität von Korrelationsmodellen lokal und global zu vergleichen. Wir haben es mit Geodaten mit Attributvektoren zu tun. Daher müssen wir mehrere Attribute gleichzeitig in einem geografischen Kontext vergleichen. Um dies zu erreichen, wird ein neues Konzept namens Geospatial-Slicing eingeführt. Beim Geospatial-Slicing werden die Daten als Volumen behandelt, wobei die x- und y-Achse Längen- und Breitengrade darstellen und die z-Achse entweder Attribute oder Modellqualitätsindikatoren darstellt. Krankheitsbilder können mit mehreren Umweltfaktoren korrelieren. Daher werden die Werte dieser Attribute oder Modellqualitätsindikatoren für einen bestimmten Bereich in gestapelten Histogrammbalken dargestellt. Das resultierende Volumen kann dann geschnitten werden, um dessen Querschnitt zu zeigen. Dadurch können mehrere Attribute benachbarter Datenpunkte verglichen werden. In einigen Fällen kann nicht nur die räumliche Verteilung von Attributen oder Modellqualitätsindikatoren von Interesse sein. Es kann auch aufschlussreich sein, wie sich Attribute oder Modellqualitäten entlang nicht räumlicher Attributachsen ändern. Dies ermöglicht die Erörterung von Fragen wie "Wie ändert sich die Modellqualität für verschiedene Werte der Bevölkerungsdichte und Arbeitslosigkeit?". Um dies zu ermöglichen, wird eine zweite Ansicht implementiert. Hier repräsentieren die x- und y-Achse Attribute anstelle von Längen- und Breitengraden. Die Implementierung ist webbasiert und wird in deck.gl realisiert, einer Open-Source-Javascript-Bibliothek für hardwarebeschleunigte kartenbasierte Visualisierung. Um unsere Ergebnisse auszuwerten, wurde die Implementierung einem Datenwissenschaftler vorgestellt. Seine Einschätzung war, dass die Visualisierung ein nützliches Werkzeug ist, um Teilmengen der Daten zu analysieren.

Abstract

Data scientists are analysing patient data, to identify the correlations between disease patterns and local environmental factors, like noise and air pollution. The large number of different attributes leads to many possible pairwise correlation models. To fully analyse the quality of these correlation models, it is important to locally compare errors in the model output, i.e., the local residuals. However, the large number of data points complicates this task. Additionally, it is important to identify correlations between geospatial values and attributes to build new models and to analyse the model quality indicators along geospatial attribute axes. Low residuals of the models and little fluctuation of them along geospatial attribute axes are indicators for high-quality models. Therefore, the goal of this bachelors thesis is to design and implement a specialized visualization in the context of a geospatial map, which serves as a tool for data scientists to locally and globally compare the attributes and the quality of correlation models. We are dealing with geospatial samples with attribute vectors. Thus, we need to compare multiple attributes simultaneously in a geospatial context. To accomplish this, a new concept named geospatial slicing is introduced. Geospatial slicing treats the data as a volume, where the x- and y-axis represent longitude and latitude, the z-axis represents either multiple attributes or model quality indicators. Disease patterns can correlate to multiple environmental factors. Hence, the values of these attributes or model quality indicators for a specific area are depicted in histogram bars, which are stacked on top of each other. The resulting volume can then be sliced to show its cross-section. This way, multiple attributes of neighbouring data points can be compared. In some cases, not only the geospatial distribution of attribute vectors or model quality indicators may be of interest. It can also be revealing to see how attributes or model quality changes along non-spatial attribute axes. This enables the investigation of questions such as "How does the model quality change for different levels of population density and unemployment?". In order to enable the analysis of these data volumes not only in a geospatial context, but also in reference to non-spatial attribute ranges, a second view is implemented. In the second view the x- and y-axis represent attributes instead of longitude and latitude. The implementation is web-based and carried out in deck.gl, which is an open source javascript library for hardware accelerated map-based visualization. To evaluate our results, we showcased the implementation to a data scientist. His assessment was that the visualization is a useful tool to analyse subsets of the data.

Contents

Kurzfassung **vii**

Abstract **ix**

Contents **xi**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Statement 1
1.3 Aim of the Work 2
1.4 Methodology 2
1.5 Structure of Thesis 3

2 Related Work **5**
2.1 Geospatial Lenses 5
2.2 Other Multivariate Geospatial Data Visualization Techniques 8

3 Background **15**
3.1 Linear Regression 15
3.2 Volume Slicing 16
3.3 Considerations for Visual Comparison 16

4 Proposed Solution **19**
4.1 Geospatial Slicing 19
4.2 Data 20
4.3 Data Normalization 20
4.4 Sorting by Variance 21
4.5 Aggregation 21
4.6 Interface 22

5 Implementation **29**
5.1 Data Loading 30
5.2 Data Normalization 30
5.3 Sorting 31

5.4 Slicing 32
5.5 Layer Rendering 32

6 Results **35**
6.1 Performance 35
6.2 Feedback 36
6.3 General Discussion 37

7 Conclusion and Future Work **39**

List of Figures **41**

List of Tables **45**

List of Algorithms **47**

Bibliography **49**

CHAPTER 1

Introduction

1.1 Motivation

Our health and diseases are heavily influenced by the environment. The more we understand which environmental factors influence disease occurrences, the better we can understand how to decrease them [PM14]. To this end, data scientists are analysing patient data, to identify the correlations between disease patterns and local environmental factors, like noise and air pollution [RKK+13]. Especially for infectious diseases environmental factors can be linked to the spread of these diseases [YKHKT18]. But also other disease occurrences are affected by the environment. For instance, Wei et al. [WLFY16] in their study about the correlation of the living environment and children's health problems in Dailan show a correlation between children's allergic diseases and their living environment.

1.2 Problem Statement

The analysed datasets are geospatially distributed. The data points are spread across a specific region for instance Austria in form of a grid. Each square of the grid depicts one data point. These data points reflect the measurements for their respective area. To analyse these factors in detail we need a high resolution, which leads to dense sampling points and therefore a big number of them. Many different environmental factors could affect our health. Thus, the datasets have a large number of data points and attributes per data point. The numbers can reach up to hundreds of thousands data points, with each having an attribute vector of up to 100 measurements. For that reason, comparing geospatial variations of attributes in detail is difficult. Additionally, when comparing local model quality indicators, it is hard to find correlation models that consider the entire data. For a full analysis of the quality of the correlation models it is important to locally analyse and compare their deviations and attributes.

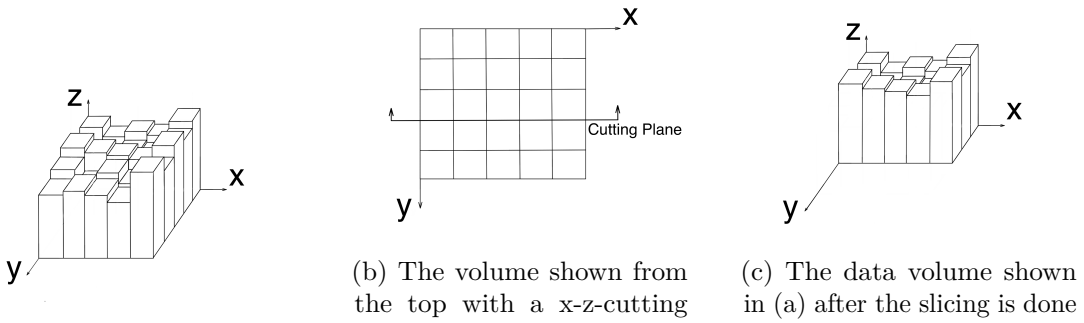
1.3 Aim of the Work

The goal of this bachelor thesis is to implement a novel visualization that should enable data scientists to analyse and locally as well as globally compare the deviations of correlation models and attributes of the data points.

1.4 Methodology

To accomplish the defined goal, we need a visualization technique that enables users to analyse areas in detail and also offers them an overview of the whole data. In medical visualization MRI- or CT-Scans are often displayed as 3D volume rendering. Additionally to the 3D visualization there usually is a 2D slice-based view. These two views complement each other. While the 2D visualization provides a detailed view of a specific region, the 3D visualization gives an overview of the whole scan and adds insight to where in the volume the slice is [DPL+11], [TMS+06]. In our case, the data does not have three spatial dimensions, but we can depict non-spatial attributes along the third spatial axis. Figure 1.1 shows the basic idea of geospatial slicing. The x- and y-axis depict longitude and latitude. The attributes or different model quality indicators are depicted in form of histogram bars along the z-axis. Figure 1.1a shows the data volume before slicing. Figure 1.1b shows the same volume from the top with a x-z-cutting plane. This cutting plane can be moved along the y-axis. In Figure 1.1c the data volume after slicing is shown. To add the possibility to compare multiple attributes, the user can select multiple attributes whose histogram bars are stacked in the data points. The total height of a histogram bar either encodes the aggregated attribute value or the aggregated model quality for the respective grid area, depending on whether attributes or model qualities are selected. By observing the height differences on the map, the data scientists get a global overview of the overall characteristics of either the attribute values or the model qualities of the selected models. This aspect of our visualization can be used to identify possible correlation models. For instance, the diabetes prevalence is higher in the east compared to the west of Austria. This values can be explored in our tool and compared to other attributes like noise, air pollution, population density, availability or proximity to medical services. If one of these attributes have a similar distribution from east to west, the data scientists can externally generate a linear regression model based on this similarity. We discuss linear regression in Section 3.1. The quality indicators of the resulting model, usually the residuals, can then be analysed in our tool. Little fluctuation of these residuals along the geospatial attribute axes are indicators for high-quality models, while a low-quality model has high fluctuations and for instance big differences of the residuals between conurbations and less densely populated areas. To be able to analyse the residuals locally in detail, the visualized volume can be sliced to yield a cross-section view of the volume. To slice the data we move the x-z- or the y-z-cutting-plane along their respective axis. Meaning that the x-z-cutting-plane is moved along the y-axis and the y-z-cutting-plane along the x-axis. This can be seen in Figure 1.1 for the x-z-cutting plane. By slicing, we reveal a cross-section of the volume. This enables the user to compare

multiple attributes of neighbouring data points. We call this visualization technique Geospatial Slicing. In some cases, not only the geospatial distribution of attribute vectors or model quality indicators can be of interest. It can be revealing to see how attributes or model quality changes along non-spatial attribute axes. This enables the investigation of questions such as "How does the model quality change for different levels of population density and unemployment?". In order to enable this, a second view is implemented. In the second view the x- and y-axis therefore represent attributes instead of longitude and latitude.



(a) The data volume before it is sliced

(b) The volume shown from the top with a x-z-cutting plane. This cutting plane can be moved along the y-axis.

(c) The data volume shown in (a) after the slicing is done with the cutting plane shown in (b).

Figure 1.1: Illustrating the principle idea of geospatial slicing.

The implementation of the visualization is web-based and carried out in deck.gl [dec] which is an open source javascript library for hardware accelerated map-based visualization.

This thesis is carried out in collaboration with the Complexity Science Hub Vienna. There, data scientists are investigating the correlation between disease patterns and environmental factors like noise and air pollution. This is done by building models using linear regression and analysing them. This collaboration ensures that the implemented tool fits the needs of the involved data scientists. To evaluate the implementation it was showcased to a data scientist with whom subsequently an interview was conducted. He identified drawbacks of our implementation and stated that the visualization is a useful tool to analyse subsets of the data.

1.5 Structure of Thesis

This thesis is structured as follows: Chapter 2 gives a quick overview of other geospatial lenses and other multivariate geospatial data visualization techniques. In Chapter 3 we discuss the background of this thesis and take a closer look at volume visualization and linear regression. Then, in Chapter 4 our proposed solution is explained in detail. The implementation is described in Chapter 5. We move on with the results of this thesis and the shortcomings of the implementation in Chapter 6 and conclude in Chapter 7 with a discussion of possible future work.

CHAPTER 2

Related Work

In their extended survey of interactive lenses for visualization Tominski et al. [TGK+17] compared multiple different types of interactive lenses in the context of information visualization. They described lenses as an important class of methods to support interactive multifaceted data exploration and that the idea behind them is to provide on demand an alternative visual representation of the data underlying a local area of the screen. The discussed lenses are also categorized. In the next section, we take a closer look at the lenses categorized as geospatial.

2.1 Geospatial Lenses

Appert et al. [ACP10] explore the limitation of magnification lenses using focus+context techniques by exploring the quantization problem in their work. They describe the quantization problem as "the mismatch between visual and motor precision in the magnified region". To solve this problem they introduce three new interaction techniques, *Speed*, *Key*, and *Ring*. *Speed* maps the precision of the lens to the input device's speed. This relies on the assumption that the input device's speed is high if navigating large distances, while a low speed is used for precise adjustments. This can be seen in Figure 2.1. The *Key* technique uses a context speed mode and a focus speed mode. The context speed mode is used for navigating large distances and the focus speed is used to perform precise navigation. To switch between these modes an additional key like SHIFT is used. The *Ring* technique can be imagined as a large rigid ring, like a bracelet on a flat surface. The Ring is moved by putting a finger into it that touches the surface, hence pulling the ring. In this case the Ring is the focus region of the lens and the cursor is the finger.

Spindler et al. [STSD10] introduced in their work tangible interactions or tangible views as an alternative to the traditional input devices like mouse and keyboard and visual feedback given on desktop displays. With tangible views the data gets projected onto a tabletop and rather than restricting the interaction to the display surface alone, the

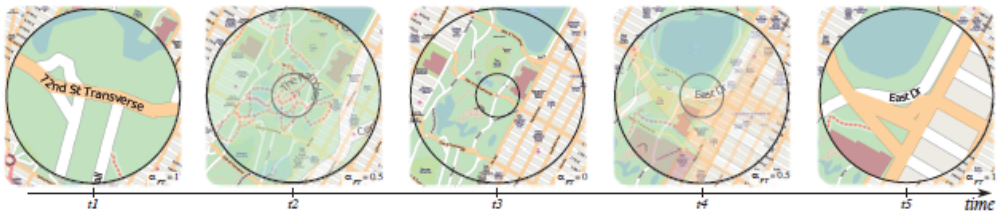


Figure 2.1: As speed increases, the speed-coupled blending lens smoothly fades into the context (from t1 to t3), and gradually fades back in when the target has been reached (t4 and t5). The inner circle fades in as the lens fades out; it delimits which region of the context gets magnified in the lens. The magnification factor remains constant [ACP10].

three-dimensional space above the tabletop is used for interaction with multiple displays. These displays show a visualization corresponding to their own position in the three-dimensional space above the tabletop. To compare different aspects of the data in a specific point the tangible views can be horizontally or vertically frozen. For instance Figure 2.2 shows two tangible views horizontally frozen. In this case the tabletop's x- and y-axis shows the spatial context as a geographic map and along the z-axis the 12 months of the year are mapped. Depending on the distance to the tabletop the views show data for a specific month. By horizontally freezing different months for this specific region, they can be compared.

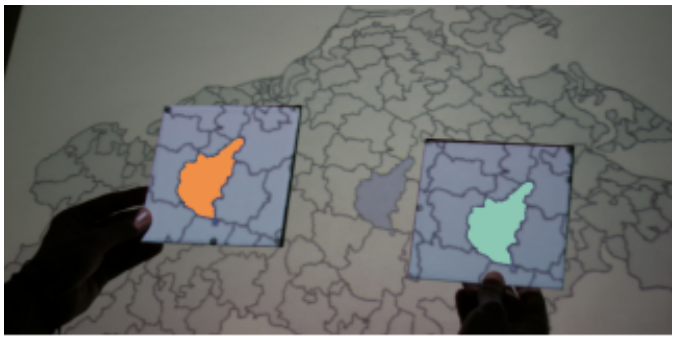


Figure 2.2: After locking the focus of two tangible views to the same location by horizontally freezing, users can visually compare between the two views by lifting or lowering them simultaneously [STSD10].

Color scales are often used to represent data. Analysing a subset of the data with the maximum or minimum values of the data set outside of the subset can lead to a small amount of the color scale being used. To overcome this limitation, Elmqvist et al. [EDF10] intodruced the Color Lens. The Color Lens inspects the lens contents in data space and optimizes the color scale for the lens. In Figure 2.3 this technique is used on a geospatial scatterplot of US Census Data. In the bar on top of the zoomed in views the maximum and minimum of data points inside this subset are shown. Additionally, the optimized color scale mapped to the values is shown beneath.

Figure 2.3: Multiscale navigation in US Census data, involving successive zooms into both space and color domains [EDF10].

Hurter et al. [HTE11] introduced a technique for interactive exploration of multivariate relational data in form of the MoleView. Given a spatial visualization of the data like a scatterplot or a graph, the lens allows users to keep selected data in the area of interest unchanged, while the rest of the data is deformed towards the edge of the lens. Figure 2.4 shows the lens applied to a color-coded image of the traffic in Lisbon at night. Here the green hues, which show relatively slow moving vehicles, are selected. All the non-green parts are pushed away from the focus. Therefore, the spatial map context is preserved for the data of interest.

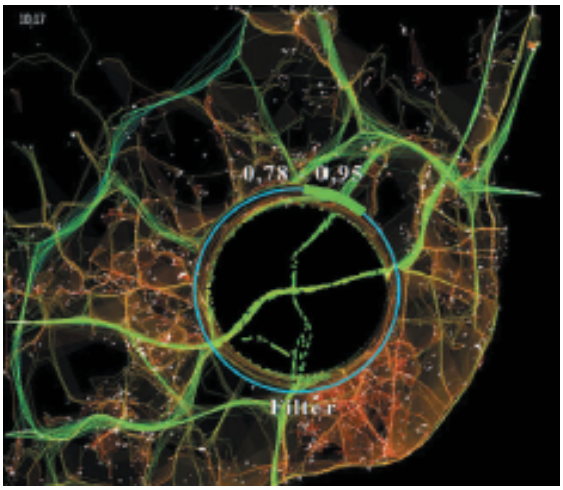


Figure 2.4: Element-based MoleView applied to color-mapped traffic speed image [HTE11].

Butkiewicz et al. [BDW+08] introduced the usage of geospatial probes in information

visualization in their study. By presenting this technique they want to overcome the limitations of traditional geospatial information visualizations of restricting the user only to a single perspective. If zoomed out, local trends become surpressed and if zoomed in spatial awareness is lost. They define a probe as "a pair consisting of a user-defined region-of-interest and a pane containing any variety of information visualizations coordinated to depict and interact with the data within that region-of interest". The user defines these probes by selecting a region-of-interest (either with a focal point and a specific radius or by manually selecting irregularly shaped regions) and then chooses the location of the visualization pane on the main geospatial visualization. Figure 2.5 shows this technique with four probes simultaneously.

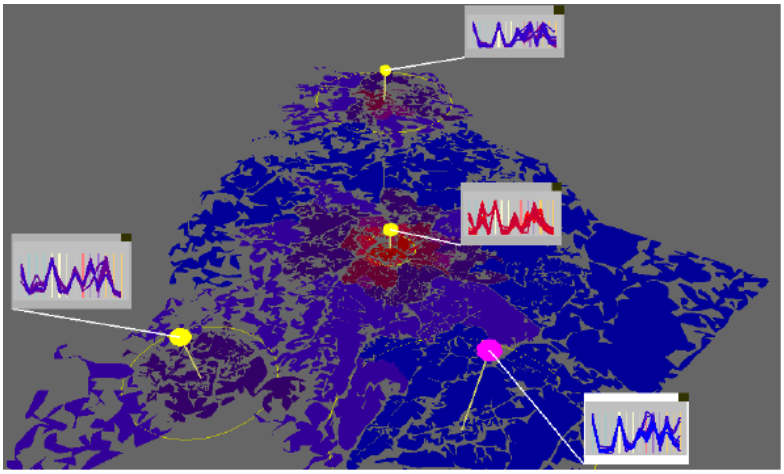


Figure 2.5: Four probes with focal point and radius are applied to a geospatial map. In the panes, a detailed view of the data in the area of the respective probe is shown. The multiple probes allow the user to compare the data of the different areas of interest [BDW+08].

2.2 Other Multivariate Geospatial Data Visualization Techniques

Maps are commonly used to visually depict multivariate geospatial data. But, with the geospatial dimensions occupying two of the spatial dimensions, the possibilities to visualize multiple attributes are limited. Commonly used techniques for visualization of single attributes on a map are choropleth maps or heatmaps as shown in Figure 2.6 and Figure 2.7, where color is used to portray the data [KK04]. By using different aspects of color like lightness, saturation and hue, multiple attributes can be displayed concurrently leading to multivariate visualizations. But, the possible number of depicted attributes is limited [TSH+14].

Another approach to make sense of multiple attributes in a geospatial context, is using interactive techniques. Here, the data is often visualized with a basic technique like a

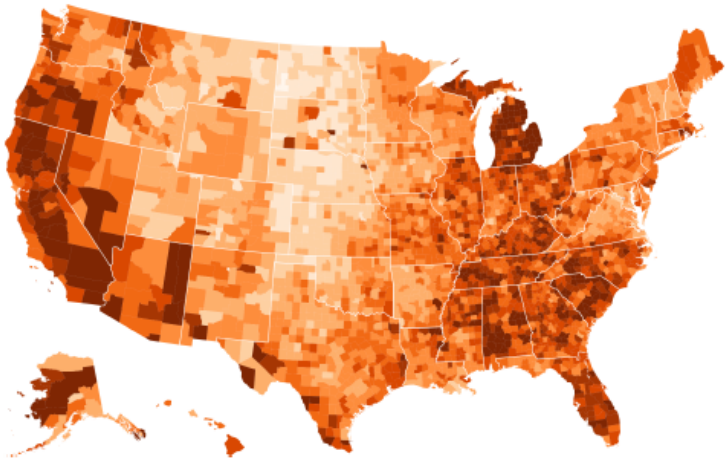


Figure 2.6: Choropleth map showing US unemployment rates of different regions from 2008, where a darker color means a higher unemployment rate [ZWC+16].

Figure 2.7: Visualizing hotspots in a city using a heatmap. The locations with a large number of vehicles passing by are shown in red [ZWC+16].

point-based visualization as shown in Figure 2.8 or a heatmap as shown in Figure 2.7. Additionally, to this basic visualization, a non-geospatial depiction of the data based on user input is supplied. So, the additional visualization can show the data in more detail for specific regions. For instance, the formerly discussed work of Butkiewicz et al. [BDW+08] uses this technique. Another example is the Community Health Map introduced by Sopan et al. [SNK+12]. This specific work addresses a similar subject as this thesis. In this case, the basic visualization is a choropleth map. Based on user input, the data can be analysed in more detail with tables showing multiple attributes for multiple regions or bar charts. This can be seen in Figure 2.9. Turkay et al. [TSH+14] also use an interactive technique in their study. They introduce attribute signatures, which are interactively crafted graphics that show the geographic variability of statistics of attributes. This concept can be seen in Figure 2.10. They enable the user to visually explore the extent

Figure 2.8: Example of point-based visualization of locations: Pickups (blue) and drop-offs (orange) of taxi trips in Manhattan from 7 to 8 am on May 1, 2011 are labeled by colored points [ZWC+16].

of dependency between the attributes and geography. Those graphics can be represented in different ways, ranging from single sparklines to multi-bar charts.

Another method to visualize geospatial data in the context of a map is to use the third spatial dimension to display attributes. This can be accomplished by depicting the data in bars on top of the map as shown in Figure 2.11. In their study Seipel et al. [SC12] compared the usage of 2D and 3D bar charts in a geospatial context. These charts can be seen in Figue 2.11. They conducted a user study to gain insight into which of the two visualizations is more user-friendly. The participants were required to carry out multiple analytical tasks, both on a 2D and 3D geospatial visualization. The conclusion of their experiments showed no significant difference in speed and accuracy, if using 2D visualizations opposed to 3D visualizations. Therefore, we decided to use a 3D visualization for our implementation. A large number of bars can lead to adjacent bars obscuring each other. A 3D navigation does not fully solve this problem but by being able to rotate the view in all three dimensions provides the user with a better spatial awareness.

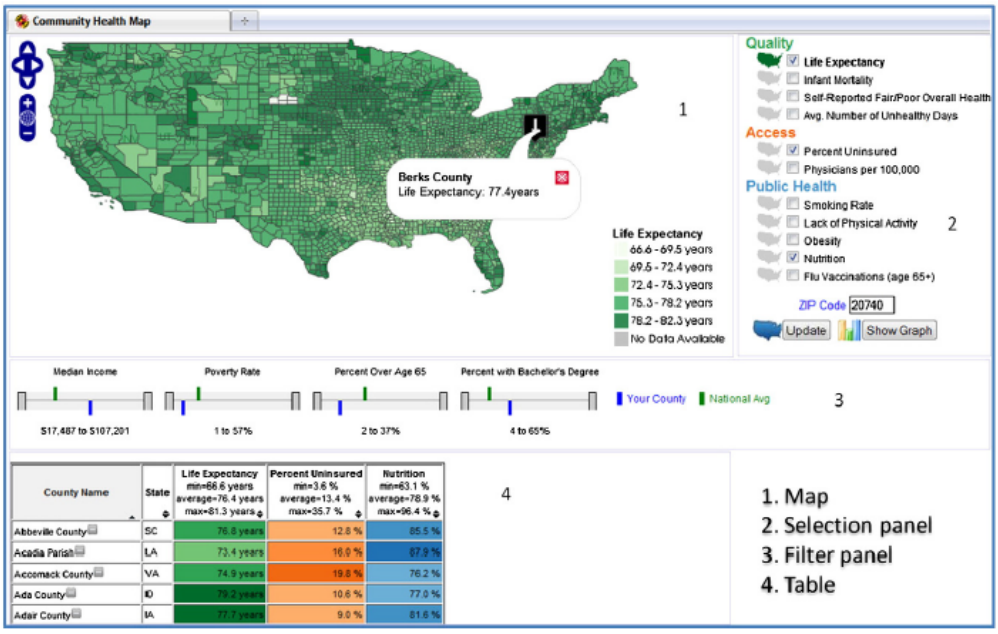


Figure 2.9: The Community Health Map interface consists of a map (top), hinted double-sided sliders for map and table filtering (middle), a color coded table (bottom), and the selection panel (right). The selection panel allows users to click the link to render a layer on the map, or check boxes to display a table with the selected variables [SNK+12].

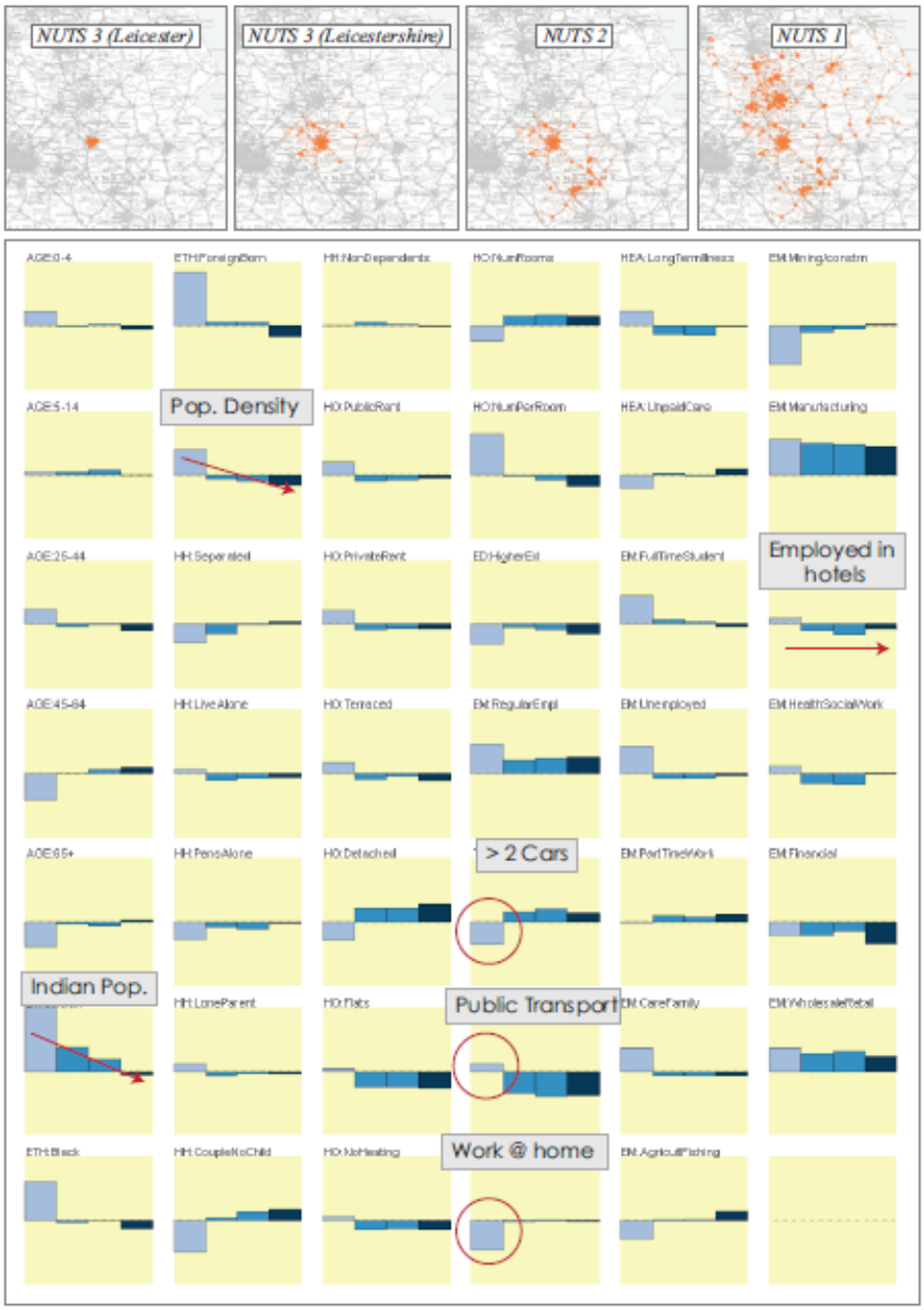
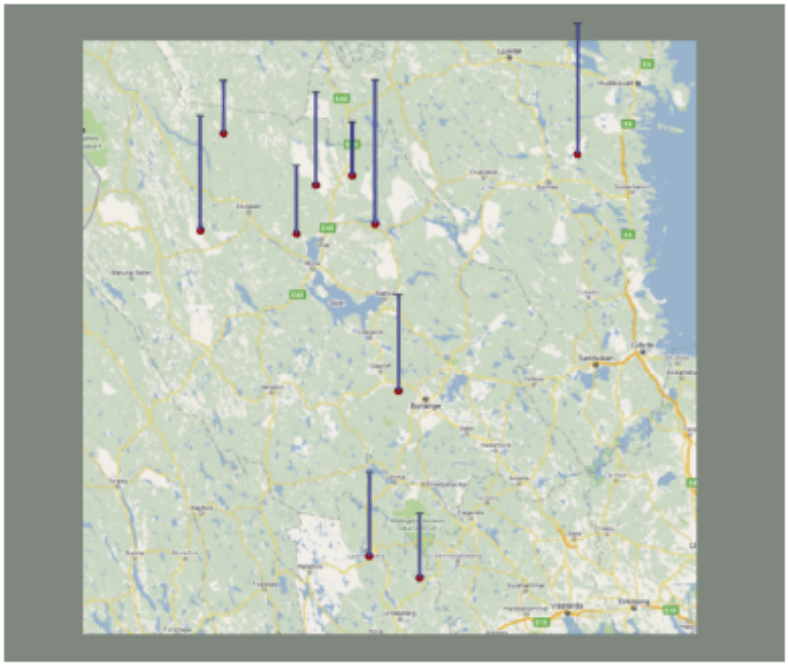
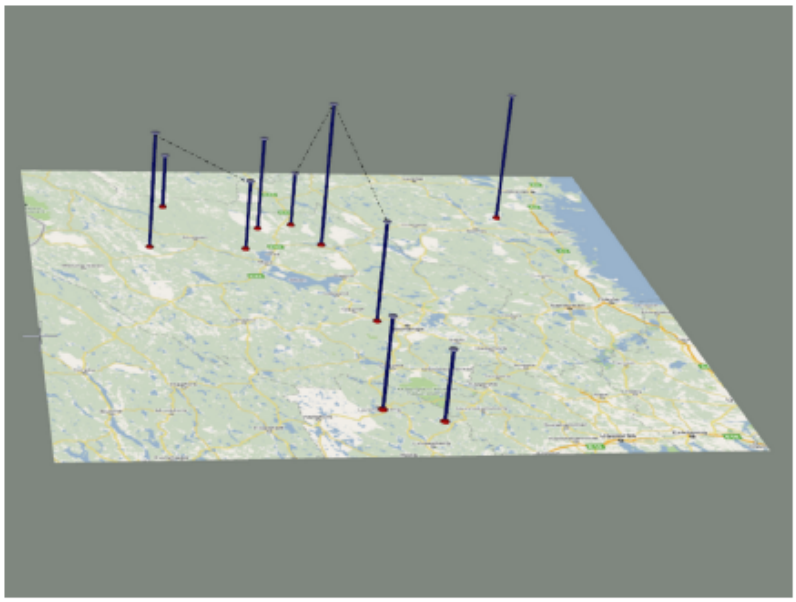


Figure 2.10: Comparing statistics for four different levels of NUTS (Nomenclature of Territorial Units for Statistics) using attribute signatures with multi-bar charts [TSH+14].



(a)



(b)

Figure 2.11: A stimulus containing ten bars as used in the experiment. Stimuli were represented and visualized in 2D (a) or visualized as a 3D slanted map with 3D bars (b). Imaginary lines between bar ends in sub-figure (b) are here indicated only for illustration purposes and they were not visible in the actual experiment [SC12].

CHAPTER 3

Background

The goal of this thesis is to design and implement a visualization that serves as a tool for data scientists to locally and globally compare attributes and the quality of correlation models in a geospatial context. Such correlation models are based on linear regression. So, to fully understand the requirements for such a tool, we need to take a closer look at linear regression. Additionally, our visualization is based on slicing a volume visualization. That is why in this chapter we will also take a closer look at linear regression and volume slicing.

3.1 Linear Regression

In statistics, linear regression is a linear technique to model the correlation between a dependent variable (response variable) and one or more independent variables (explanatory variables). In the case of the data scientists, the response variable is disease occurrences and the explanatory variables are environmental factors. The response variable is modelled as a linear function of the explanatory variables and a random error. The parameters of the function are determined based on a predefined method. Most commonly used is the least squares principle. In this case the parameters of the function are defined to lead to the smallest value of added squares of the residuals of the data points. The residuals are the vertical distances between the data points and the function line. This means that we minimize the squared error for this regression model. The resulting function can help to predict the response value based on the explanatory values. The residuals can be analysed to identify the quality of the regression model. Little fluctuations of them along the geospatial attribte axes are indicators for high-quality models, while a low-quality model has high fluctuations and for instance big differences of the residuals between conurbations and less densely populated areas [YS09].

3.2 Volume Slicing

To visualize our data, we decided to use a volume representation superimposed on a map. Volume visualization is an often used technique in medical and industrial image analysis. Magnetic Resonance Imaging (MRI) or 3D Computed Tomography (3D CT) use different image acquisition modalities to generate volumetric images, where each sample point in the volume represents one or more attributes measured at that location [DPL+11]. This data can then either be segmented and observed in 3D or analysed slice by slice to interactively identify spatial attribute variations that are hidden in the volume. Although, a purely slice based representation can lead to the loss of spatial awareness. By combining 3D visualization and 2D slice-based views they complement each other. While the 2D view provides a detailed view of a specific region, the 3D visualization gives an overview of the whole data and adds spatial awareness [TMS+06]. We want to combine these two advantages in our implementation and therefore use a volume visualization and enable the user to slice it.

3.3 Considerations for Visual Comparison

In their study of Considerations for Visual Comparison Gleicher et al. [GAW+11] discusses the issues if implementing a tool for visual comparison and provides a strategy to avoid these issues. The proposed strategy is divided into four sections. We will present these sections and show how they are applied to our implementation:

- *Identify the Comparative* ***Elements***: There are two common elements of comparison, a set of targets and an action performed on the relationships among these targets. In our case, the set of targets are the data points with each containing multiple attributes and model quality indicators. The action performed on the relationships among these targets are identifying the relationships among them and thus, finding new possible correlation models.

- *Identify the Comparative* ***Challenges***: The difficulty of comparison grows with three factors: the number of items to compare, the size or complexity of the items, and the size or complexity of the relationships between items. The large number of data points in our data set and the large number of attributes for each of them lead to comparative challenges.

- *Identify a Comparative* ***Strategy***: There are different strategies to compare data. The used strategy is dependent on the challenges identified. These strategies can be summarized into three categories: scanning sequentially, selecting a subset, and summarization. Due to the challenges identified, we need to make use of two of these strategies in our implementation. We use selecting a subset and summarization strategies. By aggregating the data points into a grid we summarize the data. Selecting a subset is used in two different ways in our implementation. On the one

hand, only selected attributes are visualized and on the other hand the slicing also leads to a subset of the data.

- *Identify a Comparative* ***Design***: The visual designs for comparison fall into three categories: juxtaposition, superposition, and explicit encoding. In our implementation we use juxtaposition and superposition. While the different data points that need to be compared are situated next to each other and are thus juxtapositioned, the different attributes are visualized on top of each other and are therefore in superposition.

CHAPTER 4

Proposed Solution

The goal of this bachelors thesis is to design and implement a specialized visualization in the context of a geospatial map, which serves as a tool for data scientists to locally compare the attributes and the quality of correlation models. Additionally, it should provide the user with an overview of the whole data set to identify trends and correlations of different attributes. To accomplish this, we introduce a novel visualization technique called Geospatial Slicing.

4.1 Geospatial Slicing

For this technique, we treat the data as a volume. Therefore, the longitude and latitude are mapped to the x- and y-axis. On the z-axis, the stacked attributes are depicted. The data points are then visualized by histogram bars stacked on top of each other. Each value of a specific attribute is depicted by one of the stacked histogram bars. Each data point contains longitude and latitude coordinates. The stacked bars are visualized in 3D on top of a map at their respective coordinates, enabling users to compare attributes based on their position as well as their variations across position. For instance, if a disease is known to have a higher prevalence in a specific region, the map can be helpful to identify environmental attributes with a similar distribution and thus higher values in this specific region. This can lead to new correlation models between the disease pattern and the environmental attribute.

Placing the 3D bars on a map, very quickly causes occlusion of the lower stack, only allowing a user to view the general height of the stacked bars. To alleviate this issue, we introduce geospatial slicing. The slicing is comparable to the slicing in medical visualizations of CT- or MRI-Scans [TMS+06]. The volume is sliced with the x-z- and y-z-plane. Figure 1.1 shows this principle with an x-z-cutting plane. Figure 1.1a shows the volume without slicing. Figure 1.1b shows the volume from the top with a x-z-cutting plane. This cutting plane can be moved along the y-axis. Figure 1.1c shows

the data volume after the slicing. The corresponding cross-section enables the user to locally compare multiple attributes or model deviations on that revealed latitudinal or longitudinal coordinates. Zooming in on this cross-section provides the user with a detailed view to compare the neighbouring data. If zoomed out the user has an overview of the whole data set to observe global variations of the stacked attribute vectors and to identify the regions where detailed local analysis is necessary.

For the data scientists not only comparing the geospatially neighbouring data points is interesting, but also comparing neighboring points in "attribute space". In this case, the height of the bar shows the error of the model at the respective attribute value. This enables the users to see if the local model quality indicators correlate with certain attribute ranges, for instance with high or low noise. For this reason, we provide a second view, where the x- and y-axis are not longitude and latitude but attributes. The z-axis still depicts stacked attributes, resulting in a volume, which we can slice in the same way as mentioned above to observe the cross-section. We will refer to this second view in the following as the abstract view and to the other view as the geospatial view.

4.2 Data

The analysed dataset is geospatially distributed. The data points are spread across Austria in the form of a grid. The grid consists of 250 by 250 meter squares and each square represents one data point. These data points reflect the measurements for their respective area. Therefore, each data point needs to contain a longitude and latitude value for them to be displayed on the map. Additionally to longitude and latitude, every data point has an attribute vector. The values of this vector either depict local correlation-model quality-indicators like the residuals or other attributes like environmental factors or disease occurrences. In our case, there are 178,957 data points with each having a 92 dimensional attribute vector. This big amount of data can be challenging to visualize and influences design decisions.

4.3 Data Normalization

The values of the different attributes can have different units and the range of these values can differ from attribute to attribute. Thus, to enable the comparison of attribute variations, all values are normalized before encoding them in a stacked bar. We offer two ways to normalize the data: local and global normalization. In each case, the Equation (4.1) is used. Depending on the used normalization, z_{max} changes. In the local normalization, each attribute is normalized separately. This means that the maximum of this attribute gets mapped to one and zero gets mapped to zero. In our case, the data set does not contain negative values. A possible solution to handle negative values is to map the minimum value to zero. We did not use this approach, due to the ratios between the bars being distorted. Meaning that a bar that is double the height of another bar does not mean that the value displayed by that bar is double the value of the other

bar. The local normalization is useful to compare the variation of attributes whose values are not in the same range or have different units. It makes the heights of the bars comparable and the user can easily identify which attributes are high in certain areas. In the Equation (4.1) this means that z_n is the normalized value, z is the original value and z_{max} is the maximum value of the attribute that z is a value of. This means that a specific height of a bar can depict different values depending on the represented attribute.

$$z_n = \frac{z}{z_{max}} \tag{4.1}$$

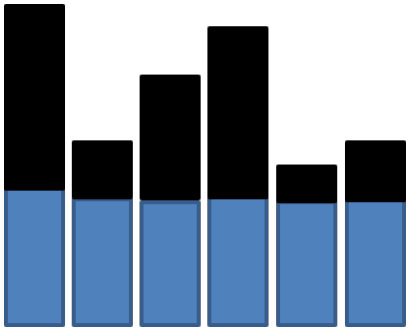
As mentioned above, in the global normalization Equation (4.1) is used as well, but z_{max} changes. Global normalization normalizes the attributes using the global maximum across all attributes. Zero still is mapped to zero. As mentioned above, we are not dealing with negative values, but the aforementioned solution of mapping the minimum value to zero is also applicable for the global normalization. This normalization only makes sense if all attributes have the same unit and are in a similar range. It is useful to compare attributes that are similar, like the percentage of infected people across different age groups. Due to the values being in the same range, the values depicted on the z-axis can refer to the values of the attributes in their respective dimensions. This is not the case for the global normalization, where the values on the z-axis depict fractions of the maximum value of the respective attribute.

4.4 Sorting by Variance

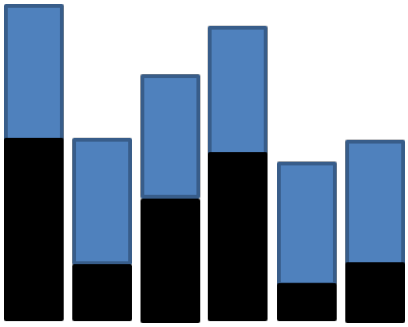
The stacking of the different attributes can lead to the attributes on the higher layers having irregular baselines, if the attributes on lower layers have a high variance. This can be seen in Figure 4.1b. These irregular baselines make it hard to compare the values of the attribute of the top layer. In order to to minimize this effect, we sort our visualized attributes by variance. Figure 4.1 shows how sorting by variance can affect the visualization. In our case, the attribute with the lowest variance will be at the bottom layer as shown in Figure 4.1a. This leads to the attributes on higher layers still having similar baselines. The closer the different baselines of the values of one attribute are, the easier are these values to compare. The opposite is shown in Figure 4.1b. Here the attribute with the high variance is depicted at the bottom layer. Hence, the baselines of the bars in the top layer differ. This leads to the top layer not being easily comparable. Still, with multiple attributes stacked on top of each other, the values of the attribute depicted at the top layer will be harder to compare as opposed to the bottom layer.

4.5 Aggregation

We initally visualized the data at its original resolution of 250 by 250 meter per data point. But, the big amount of data points and attributes lead to difficulties. We tested the performance in these two regards with the hardware configurations shown in Table 6.1



(a) The attribute with the lower variance is pictured in the bottom layer. This leads to the values of the attribute on the top layer with the high variance still being comparable, because they have similar baselines.



(b) The attribute with the higher variance is pictured in the bottom layer. This leads to the values of the attribute on the top layer with the low variance harder to compare, because they have different baselines.

Figure 4.1: Sorting attributes by variance

on a computer running Windows 10. With our data set of 178,957 data points and a single visualized attribute, the frame rate dropped below one frame per second when moving the camera. Therefore, we give users the option to spatially aggregate their data on the fly, if exploration at high resolution is not feasible. This is accomplished with a slider input that changes the cell size of the histogram bars. This leads to multiple data points per histogram bar. In our case, the data points are not necessarily equally distributed. This means that after the aggregation not all histogram bars include the same number of data points. To still keep the aggregated values comparable (per certain area), each bar encodes the average of the aggregated values. This aggregation is done by deck.gl and will be described in more detail in Chapter 5.

4.6 Interface

Figure 4.2 shows our interface, currently in the geospatial view with three selected attributes and no slicing. It consists of the stacked bars on top of the map and three control panels: the axis controls shown in Figure 4.4a in the top left corner, the bar controls shown in Figure 4.4b in the top right corner, and the selectable attributes shown in Figure 4.4c in the bottom right corner.

In the Attributes Control panel shown in Figure 4.4c, the user can select the attributes to visualize on the map. If an attribute gets selected or deselected, a new sorting is triggered. Each attribute has an assigned color in which it is depicted. To have distinguishable colors, we use a color scale tested with the Chroma.js Color Palette Helper [col]. The color scale ranges from yellow to dark green. The used colors are equally distributed along this scale. The color corresponding to the attribute is shown next to the selected attribute.

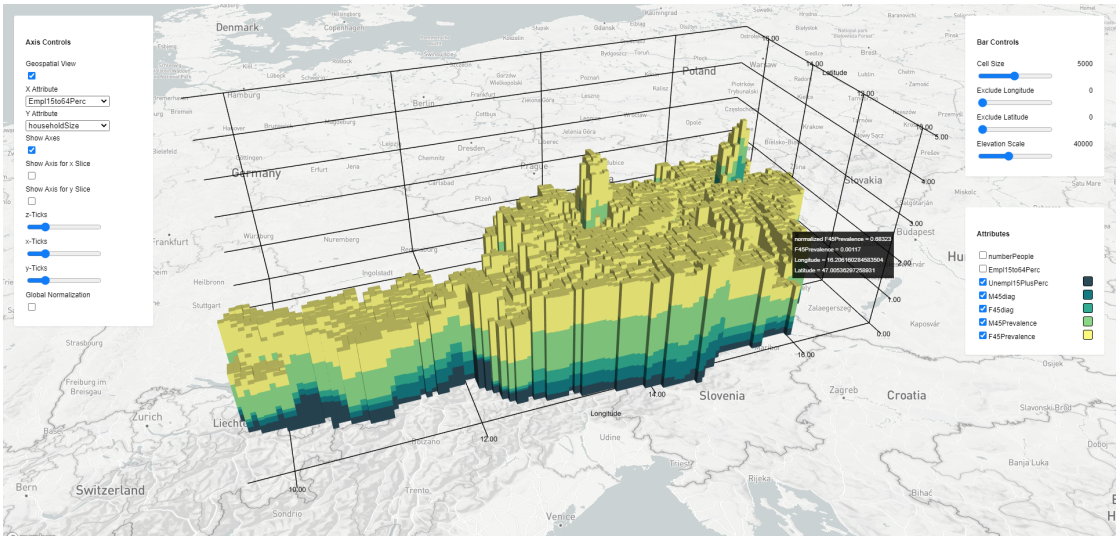
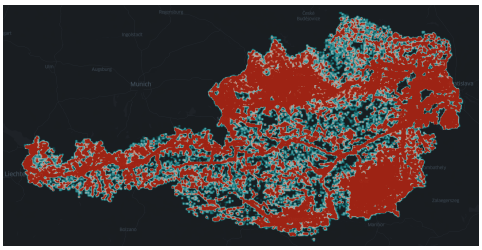


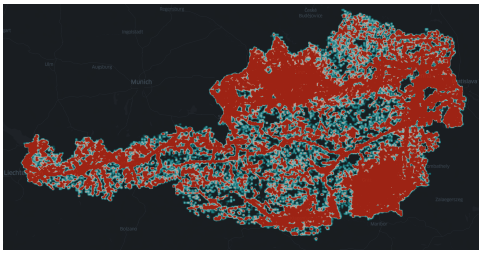
Figure 4.2: Geospatial view showing percentage of unemployed people at the age of 15 years or older (dark blue), the number of male people at the age between 45 and 59 diagnosed with diabetes (blue), the number of female people at the age between 45 and 59 diagnosed with diabetes (light blue), the prevalence of diabetes for male people at the age between 45 and 59 (green) and the prevalence of diabetes for female people at the age between 45 and 59 (yellow). A label is displayed at the bar the cursor was hovering over, showing the values of the respective bar. Figure 4.3 shows the same attributes visualized in juxtapositioned choropleth maps. With five or more attributes it can get tidy to locally compare attributes on a juxtapositioned maps. Our visualization simplifies this task.

The Bar Controls shown in Figure 4.4b support the following operations: By moving the Cell Size slider the base area of each bar is changed and therefore the spatial aggregation is adjusted. Exclude Longitude and Exclude Latitude are the sliders for the slicing. The labels of these sliders are depending on the name of the attribute depicted on the respective axes. For instance, if the geospatial view is shown, then the labels read Exclude Longitude and Exclude Latitude. If the abstract view is shown and the attribute depicted on the x-axis is householdSize, then the slider for the x-axis reads Exclude householdSize. Moving these sliders moves the slicing plane along their respective axis as shown in Figure 4.5. The Elevation Scale is a multiplier for the height of the bars and changes the bars and the height of the axis accordingly.

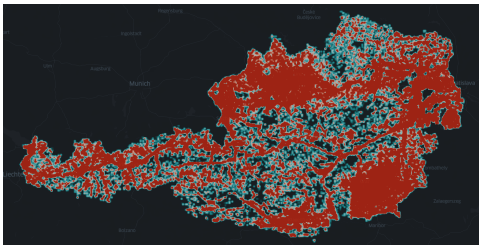
The Axis Controls shown in Figure 4.4a provide the user with settings to change the axis. With the check box Geospatial View, the user can change between the geospatial view shown in Figure 4.2 and the abstract view shown in Figure 4.6. In the abstract view, the x- and y-axis do not depict longitude and latitude, but selectable attributes. These attributes are selected in the X Attribute and Y Attribute drop down list for their respective axis. The values shown at the ticks of the axes also update to values in



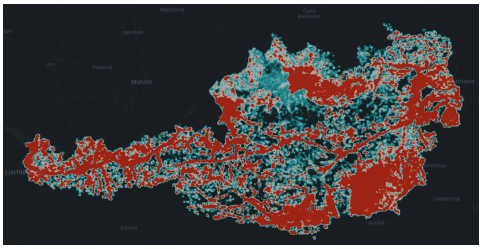
(a) Number of female people at the age between 45 and 59 diagnosed with diabetes



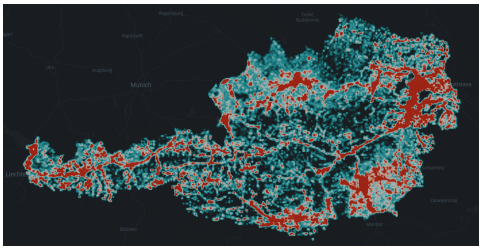
(b) Number of male people at the age between 45 and 59 diagnosed with diabetes



(c) Prevalence of diabetes for female people at the age between 45 and 59



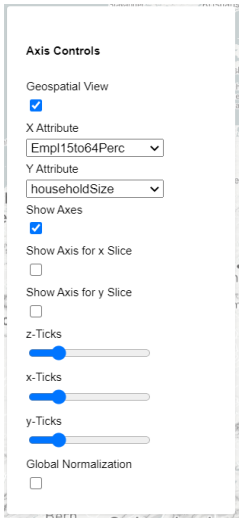
(d) Prevalence of diabetes for male people at the age between 45 and 59



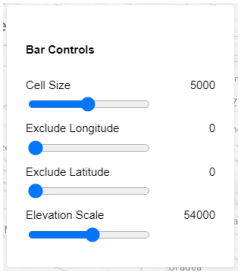
(e) Percentage of unemployed people at the age of 15 years or older

Figure 4.3: Figure (a) to (e) showing choropleth maps with the same data as is depicted in Figure 4.2. The red colored regions have high values, while the blue colored regions have low values. For the black regions there are no data points. These maps were created with the kepler.gl [kep] demo.

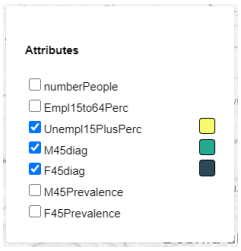
the range of the selected attribute. If the Show Axes check box is ticked, the axes at the bottom, back, and right are shown as in Figure 4.2. These axes can be hidden by deselecting the check box. To help the user to analyse the height of the bars at the slicing cross-section, the user can display axes at the slicing plane as shown in Figure 4.5 for the basic view and in Figure 4.7 for the second view. This is done by selecting the check boxes Show Axis for x Slice and Show Axis for y Slice. These axes move with their respective slicing planes as the slice sliders are moved. To adjust the axes to the users needs, the user can also change the number of ticks shown on each axis with the z-, x-, and y-axis sliders. To change between the different normalizations mentioned in Chapter 4.3 the user can check and uncheck the global normalization check box. Changing this, triggers



(a) In the Axis Controls panel the user can change between the geospatial and abstract view, the attributes depicted along the x- and y-axis in the abstract view can be changed, the axis can be turned on and off, the number of ticks along a specific axis can be changed and the user can switch between the global and local normalization.



(b) In the Bar Controls panel the user can change the cell size used for the aggregation, move the slicing panes and change the elevation scale of the visualization. The slicing planes are moved with the 'Exclude Longitude' and 'Exclude Latitude' sliders. The labels of these sliders change based on the attribute depicted on the respective axis.



(c) In the Attribute Controls panel the user can select the attributes to visualize. The colors, the attributes are depicted in, can be seen next to the attribute names.

Figure 4.4: Control Panels

a new normalization. If the normalization is global, the numbers on the z-axis show real values of the attributes. As mentioned in Section 4.3, the local normalization leads to the values of the attributes being mapped between zero to one. This means that the maximum of each normalized attribute is one. So, the values on the z-axis are relative and range from zero to number of attributes rendered on top of the map.

For the navigation, we utilize broadly used 3D navigation techniques. The map can be panned by holding down the left mouse button and moving the mouse. The map can be zoomed by turning the mouse wheel. The visualization can be rotated by holding down the right mouse button and moving the mouse. This allows the user to observe certain areas in the data in detail. If the pitch which is the angle between the normal of the map and the camera direction is higher than 60°, the map is not rendered. This can be seen in Figure 4.8. The bars are an important tool to compare the different values, but do not allow the user to know the exact values of the grid points. Therefore, the user can hover over one of the bars to show the values of the attribute depicted by this bar and

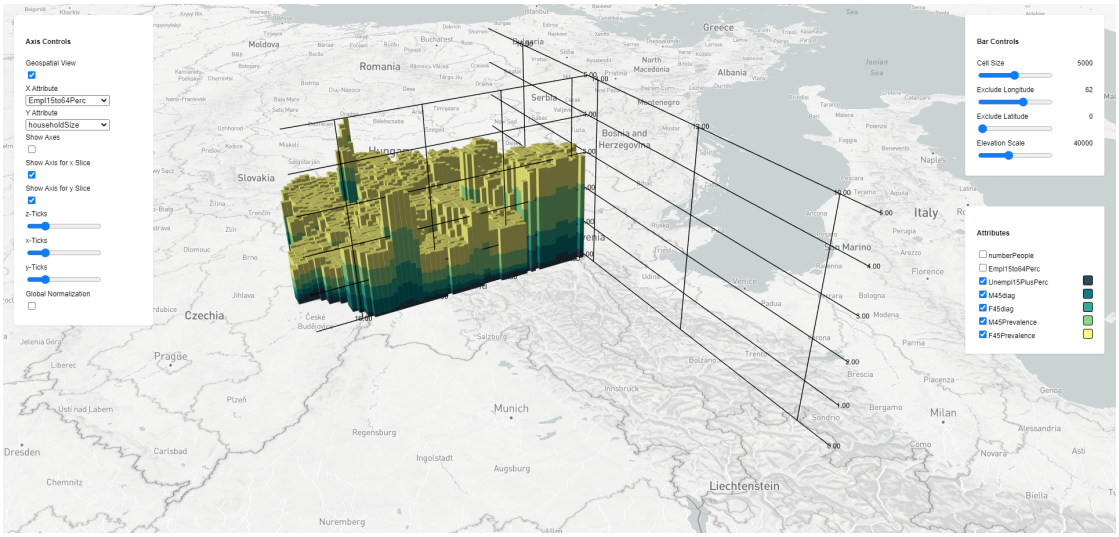


Figure 4.5: Geospatial view showing the same attributes as Figure 4.2. Additionally, slicing along the latitude axis is applied to show a cross-section of the data. Here we can identify peaks in the data in the region of Linz and in the south of Austria.

the location of the gridpoint as seen in Figure 4.5.

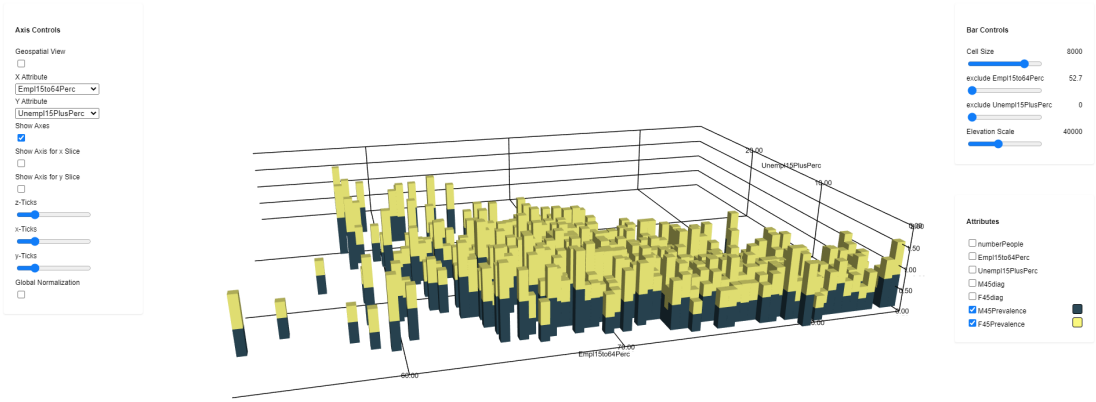


Figure 4.6: Abstract view showing the prevalence of diabetes for male people at the age between 45 and 59 (green) and the prevalence of diabetes for female people at the age between 45 and 59 (yellow). The x-axis shows the employment of people at the age of 15 to 64 years in percentage. The y-axis depicts the percentage of unemployment of people at the age of 15 or older. We can see a slight increase of the combined prevalence for low employment and high unemployment.

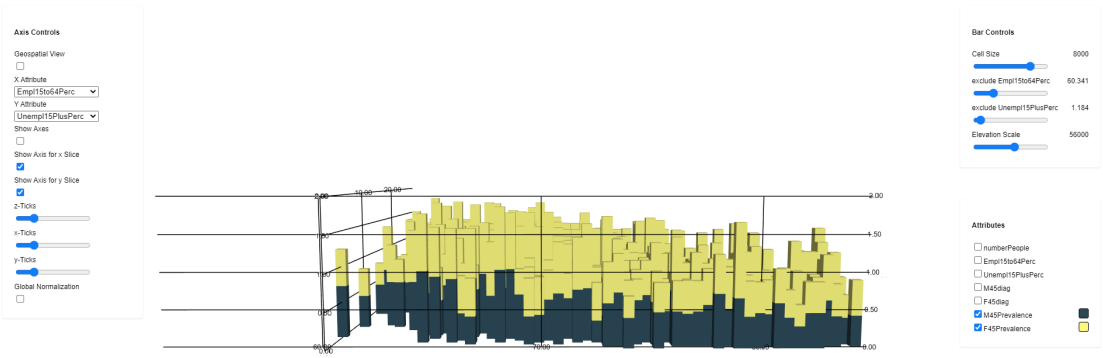


Figure 4.7: Abstract view showing the same attributes and the axes as in Figure 4.6. The data is sliced at an employment of 60.341 percent an an unemployment of 1.184 to reveal cross-section. Here we can analyse the data for this value of unemployment in detail. High employment is on the right side of the visualization. We can see an increase of the prevalence with decreasing employment.

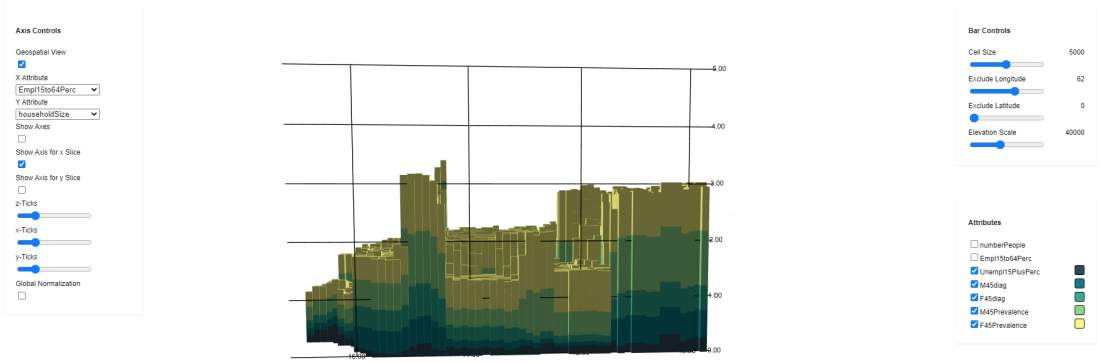


Figure 4.8: Geospatial view showing the same attributes as Figure 4.2. The data is sliced along the latitude axis to reveal the cross-section and the camera is rotated to a pitch bigger than 60°, hence the map is not rendered. This is the same slice as shown in Figure 4.5. We can see that the data has peaks in urban areas and in the south (right side).

CHAPTER 5

Implementation

The implementation is web-based and carried out in javascript using deck.gl [dec] for hardware accelerated map-based visualization. Deck.gl renders data visualization layers like scatterplots or heatmaps onto a map. The map is provided by mapbox [map] and rendered by react-map-gl [rea]. In Table 5.1, the utilized libraries with their respective usages are listed.

Library	**Usage**
deck.gl	Visualization layers
luma.gl	Shader programming
chroma-js	Color scaling
react-map-gl	Map Rendering
Papa Parse	Data loading
d3	Scales for Axes

Table 5.1: Libraries used in our implementation.

In Figure 5.1 a simple graphic of our program pipeline can be seen. First, data is loaded and simultaneously the maxima of the attributes are calculated. Afterwards, the data is normalized. Then, we sort the attributes by variance. In the next step, the data is sliced and the layers get rendered. User input can trigger new normalizing, sorting, or slicing of the data. In the next sections, we are going to take a closer look at these steps and how they are realized in our implementation.

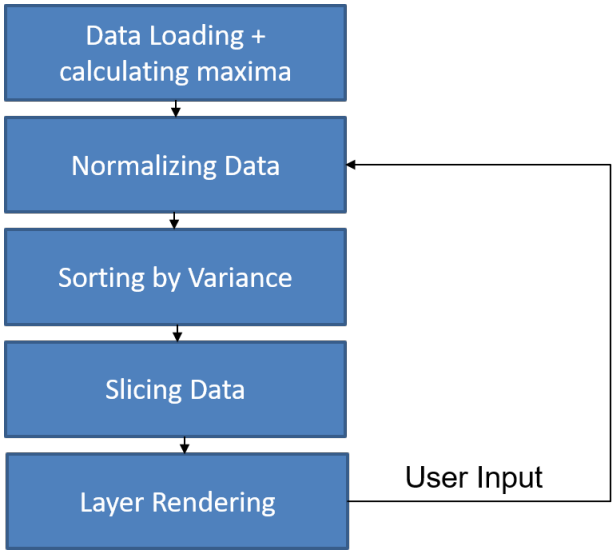


Figure 5.1: Program Pipeline: First the data is loaded and the maxima of the attributes are calculated. Then, the data is normalized. In the next step, the selectable attributes are sorted by variance. Afterwards, the data is sliced and then the layers are rendered. User input can trigger new normalizing, sorting, or slicing of the data.

5.1 Data Loading

As mentioned in Section 4.2 the data has to be provided as a grid of data samples, which contain latitude and longitude coordinates and an attribute vector. The test data we used consisted of data points scattered across Austria on a grid with 250 meter by 250 meter sample areas, each of them associated with 92 attributes. This data is loaded from a csv-file into an array of objects using Papa Parse [pap]. Each object represents a single data point. Since every row and every value has to be loaded into the array by looping through it, we use this loading loop to additionally calculate the maximum and minimum values of the attributes and also load them into an array of objects containing the maximum and minimum value of a single attribute. If handling big amounts of data this can take some time. We will discuss the performance later in Chapter 6.

5.2 Data Normalization

As mentioned in Section 4.3 two types of data normalizations are supported: global and local normalization. The "normalizeData" function in our program is carried out at the start of the program after the data loading and is triggered if the user changes between local and global normalization. To normalize the data, we use the attribute maxima that are calculated while loading the data. In Algorithm 5.1, the data normalization function is shown in pseudo code. The code shows a simplified version of our algorithm. Due to the data having a high number of attributes, in our case 92, we hard coded which

of the attributes are interesting and should be selectable in the visualization. To save computation time, only the attributes that are selectable are normalized. The other attributes are ignored. Longitude and latitude are not normalized. Algorithm 5.1 shows the basic logic behind the data normalization.

Algorithm 5.1: Data Normalization

Input: **isGlobal:** boolean true = global normalization / false = local normalization, **data:** array of data point objects each having a position and attribute fields, **maxValues:** object containing a field for every attribute's max value, **globalNormalizingMax:** maximum value across all attributes

Output: **normalizedPoints** dataPoints with normalized values

```
1  normalizedPoints ← [ ];
2  i ← 0 foreach dataPoint in data do
3  |   j ← 0;
4  |   foreach attributeValue in dataPoint do
5  |   |   if isGlobalNorm then
6  |   |   |   normalizedPoints[i][j] ← attributeValue / globalNormalizingMax;
7  |   |   else
8  |   |   |   normalizedPoints[i][j] ← attributeValue / maxValues[j];
9  |   |   end
10 |   |   j ← j + 1;
11 |   end
12 |   i ← i + 1;
13 end
14 return normalizedPoints;
```

5.3 Sorting

As mentioned in Section 4.4, to make the bars in the visualization as convenient to compare as possible, we need to sort the attributes by variance. The sorting function is called after the data normalization and every time the user changes the selected attributes. The function calculates the variance and sorts the attributes according to it. Equation 5.1 shows how to calculate the variance. s^2 is the variance, n is the number of data points, $\bar{x}$ is the arithmetic middle of the attribute across all data points, and x_i is the value of the attribute for the data point with the index i. Due to the number of data points being the same for every attribute we do not need to divide by n to order the attributes by variance. The Algorithm 5.2 shows our sorting in pseudo code. For the sorting, the javascript function sort is used. This javascript function uses a sorting algorithm provided by the used browser. For instance, Google Chrome uses Insertion Sort for arrays with the length of ten or less and Quick Sort for longer arrays [jav].

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{5.1}$$

5.4 Slicing

After the attributes are sorted, the data is sliced. This is done at the start of the program after the sorting and is triggered by different input events: Changing the slicing sliders causes slicing. Switching between the geospatial and the abstract view triggers slicing. Additionally, when the attributes for the x- or y-axis are changed, slicing is invoked. Before we added the aggregation due to performance issues, the slicing was done in the shader. So, the 3D visualization was sliced. Although, at one point we started to use an aggregator provided by deck.gl. To save computing time, this aggregator is programmed in a shader and therefore runs on the GPU. The input data we get for our shader is determined by the aggregator and hence cannot be customised. This leads to us not being able to pass the required data for the slicing to our custom shader. Thus, we had to find another way to slice our visualization. The slicing is now done beforehand by slicing the data. This leads to the aggregator having a different data set as input after the slicing, and triggering a new aggregation. This can lead to the bars changing their location and size due to different data points being aggregated together. A possible solution for this problem is to slice in steps of the grid size. We will discuss this solution in Chapter 7. As mentioned above, we slice our data based on the slider inputs and whether the user observes the geospatial view or the abstact view and which attributes are visualized. The sliders define minimum values for their respective attributes. Thus, every data point that has a lower value for the specific attribute is sliced from the data set. In the geospatial view these attributes are longitude and latitude. In the abstract view the attributes are the selected attributes for the x- and y-axis.

5.5 Layer Rendering

The processed data now needs to be visualized on top of the map. In deck.gl [dec] this is done through visualization layers. Multiple existing layers are provided by deck.gl but none of these were suitable for our task. Although, there is a histogram bar layer with aggregation, the bars can not be stacked, because the bottom of the bar is always on map level. Therefore, we extended the GPUGridLayer of deck.gl with a custom shader, which allows the bars to levitate at a certain height. This leads to every attribute having a separate layer. These layers are combined in our composite layer. The composite layer gets the processed data, the attribute order and the visualization settings as inputs. Additionally, an array of colors, one for every visualized attribute, gets passed to the layer. The colors are generated with chroma-js [chr]. The composite layer then creates a layer for every visualized attribute starting with the one at the bottom. As mentioned before, we aggregate our data points. This means that multiple data points are aggregated into

Algorithm 5.2: Attribute Sorting

Input: **attributes** array of names of the selectable attributes, **data** array of data points each having attributes that are in the same order as in **attributes**

Output: **sortedAttributes** array of the selectable attributes in the correct order

```
1  attributeQuantity ← attributes.size;
2  means ← [ ];
3  quadmeans ← [ ];
4  s^2 ← [ ];
5  index ← 0;
   /* Calculating mean and quadratic mean values for all
      attributes                                              */
6  foreach dataPoint in data do
7  |  j ← 0;
8  |  foreach attribute in dataPoint do
9  |  |  if index > 0 then
10 |  |  |  sums[j] ← means[j] + attribute;
   |  |  |   quadsums[j] ← quadmeans[j] + attribute^2;
11 |  |  else
12 |  |  |  means[j] ← attribute;
13 |  |  |  quadmeans[j] ← attribute^2;
14 |  |  end
15 |  |  j ← j + 1;
16 |  end
17 |  index ← index + 1;
18 end
   /* calculating variance                                    */
19 n ← data.length;
20 for i ← 0 to n − 1 do
21 |  s^2[i] ← [ ];
22 |  s^2[i][0] ← quadmeans[i]/n − (means[i]/n)^2;
23 |  s^2[i][1] ← attributes[i] ; // saving attributename into the array
24 end
   /* sorting                                                 */
25 s^2 ← javascript sort s^2 by s^2[n][0];
26 sortedAttributes ← [ ];
27 for i ← 0 to s^2.length do
28 |  sortedAttributes ← s^2[i][1];
29 end
30 return sortedAttributes;
```

one bar. This aggregation is provided by deck.gl with the GPUGridAggregator. In an article by Ravi Akkenapally [gpu] the method of this aggregation is discussed. As the name suggests, the aggregation is carried out on the GPU for better performance. The aggregation data is then directly passed to the shader. Every layer except the first one has values for the distance to the map plane, which are the sums of the values of the attributes in the layers below. These values need to be aggregated in the same way as the height of the bars. So, they are passed to the GPUGridAggregator as a second attribute to aggregate. The aggregated values are then used in the shader for the distance to the map plane of the bars. The GPUGridAggregator provides different aggregation operators like sum or mean. The aggregation leads to different numbers of data points per cell. Due to this and the fact that most of our attributes are provided in percent, we decided to use the mean aggregation. At the time our tool was implemented, the mean aggregation of the GPUGridAggregator was not working properly. It just summed the values and did not divide them by the number of data points, meaning that it did the same as the sum aggregation. The number of data points aggregated into the cell is passed to the custom shader by the aggregator. Therefore, we used the sum operator and divided the aggregated value by the number of data points in the cell. The result of this is the average.

In the abstract view we use the values of the x- and y-axis attributes as longitude and latitude coordinates. These coordinates are most likely not in the same area on the map as the coordinates used for our geospatial view. This would mean that we need to move the camera of our visualization. To avoid this, we need to convert the values of our x- and y-axis attributes into longitude and latitude coordinates that are in the same area as the coordinates for our geospatial visualization. Hence, the minimum value of our x-attribute is scaled to the minimum longitude and the minimum value of our y-attribute is scaled to the minimum latitude in our dataset.

For our axes, we used the axes layer of the plot layer from deck.gl and fitted it to our needs. We have three different layers for the axes. The first one represents the axes which are at the bottom, back and right of the visualization as seen in Figure 4.2. These range from the minimum longitude to the maximum longitude and from the minmum latitude to the maximum latitude. The second and third layer represent the axes that move with the slicing. The scales passed to these axis layers are created using d3 [d3].

CHAPTER 6

Results

In this chapter we take a look at the performance of the implementation, we discuss the feedback we received from the domain expert, and the advantages and disadvantages of our approach.

6.1 Performance

As we mentioned before, there are two areas in our implementation where the performance can be an issue. On the one hand, the data loading, maxima calculation, and data normalization can be time intensive and on the other hand, a large number of individual bars can lead to performance issues. We tested the performance in these two regards with the hardware configuration shown in Table 6.1 on a computer running Windows 10.

Hardware	**Type**
CPU	Intel Core i7-6700HQ
GPU	NVIDIA GeForce GTX 960M
RAM	16 GB DDR4

Table 6.1: Hardware configuration used for our performance evaluation.

We analysed the computation time to load the data including the maxima calculation and also the time it took to normalize the data. This data can be seen in Figure 6.1. The biggest suitable data set we had available consisted of 178,957 data points with each having a 92 dimensional attribute vector. The data loading with maxima calculation took on average 21.37 seconds and the data normalization with local normalization took 4.04 seconds on average. The data loading, which took the most time, is only done once when the tool is started. The normalization is done after the start of the program and can be triggerd again by changing the normalization method. After scaling down the data set to 100,000 data points while each still having a 92 dimensional attribute vector,

the time for the data loading took 11.77 seconds and the data normalization took 2.67 seconds on average. When testing with 50,000 data points the computation time was roughly cut in half. The average time for the data loading was 5.97 seconds and the data normalization took 1.32 seconds on average.

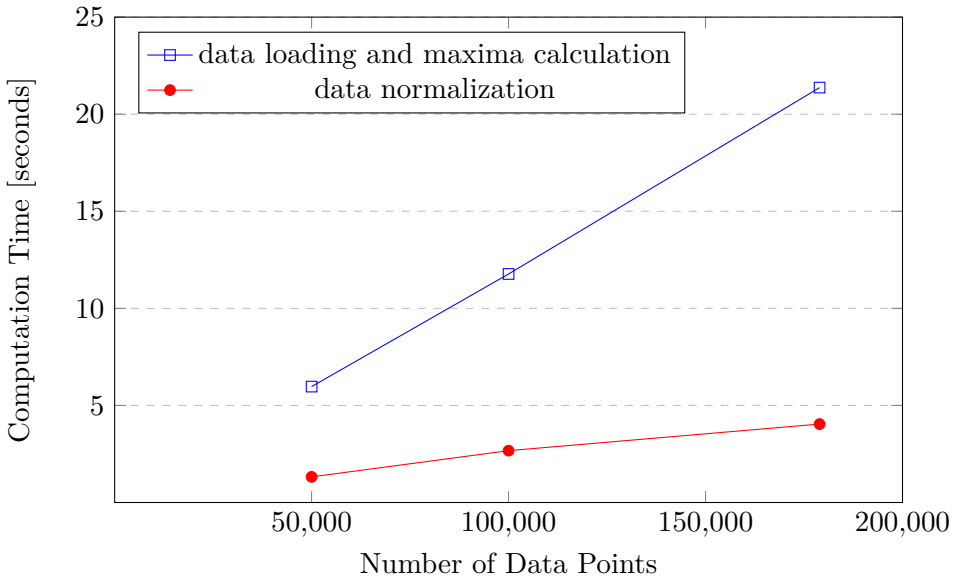


Figure 6.1: Plot of the computation time of the data loading with the maxima calculation, and the data normalization in relation to the number of data points visualized

The performance concerning the number of bars visualized is evaluated based on the frame rate. The frame rate of the implementation is dependant on the number of individual bars visualized. The number of these bars is determined by the size of the data set, the cell size for the aggregation and the number of attributes visualized. The larger the number of data points as well as attributes in the data set is, the more data has to be visualized and needs to be aggregated, and therefore the computational workload is bigger. The larger the cell size is, the more data points are aggregated into single bars and thus the visualization workload is lower. The bigger the number of visualized attributes is, the more bars need to be visualized due to each attribute being depicted in a separate layer. This means that the number of bars is based on the number of cells multiplied by the number of attributes. Our evaluation showed that our frame rate was consistently above 25 fps (frames per second) if the number of bars was lower than 30,000. But with a rising number of bars our frame rate dropped rapidly and we experienced lag. With roughly 40,000 bars the frame rate was consistently below 10 fps.

6.2 Feedback

This implementation should act as a tool for data scientists to locally compare the attributes and the quality of correlation models. So, to evaluate the implementation we

showcased it to a data scientist, who is analysing patient data, to identify the correlations between disease patterns and local environmental factors, like noise and air pollution. In his assessment he mentioned the issue with slicing, as it triggers new aggregation. This leads to different data points being combined in the histogram bars, which means that the positions and heights of the bars change due to the slicing. He mentioned that the slicing needs to be consistent and cut the data volume like a cake. Additionally, he said that if this issue is solved the implementation is a useful tool to analyse what is going on in subsets of the data and provides the ability to locally explore the data. He also said that the tool can be useful to identify possible correlation models and to analyse the formed models. The idea for our abstract view was based on feedback of a senior data scientist. He suggested that it would be good to be able to analyse the data not only in a geospatial context, but also in reference to non-spatial attribute ranges, therefore enabling the users to see if the local model quality indicators correlate with certain attribute ranges, for instance with high or low noise.

6.3 General Discussion

Our implementation provides the user with an overview of the whole data set with spatial awareness if zoomed out as seen in Figure 4.2. The user can identify areas in the data which need further analysis. The area of interest can then be observed by moving the slicing plane to it, exposing the cross-section and zooming in as shown in Figure 4.8. As mentioned before, the slicing triggers new aggregation for the histogram bars and therefore can lead to changed positions and heights of the displayed bars. This is due to our slicing being continuous, while the spatial aggregation is based on a fixed grid. There are situations where continuous slicing can be beneficial. For instance, if a subset of the data sliced at a specific value is of interest. But, it can also be a drawback, if for instance the variation of the attributes across the whole data is of interest while slicing. At this moment our implementation only supports continuous slicing. In Chapter 7, we will provide possible solutions to the drawbacks of the continuous slicing for future work on this implementation. Another drawback is that the computational workload due to the 3D visualization forces us to spatially aggregate the data for big data sets. Hence, we lose a certain degree of detail. This issue is specific to large data sets.

CHAPTER 7

Conclusion and Future Work

The aim of this thesis was to to design and implement a specialized visualization in the context of a geospatial map, which serves as a tool for data scientists to locally compare the attributes and the quality of correlation models. To accomplish this, we introduced the novel visualization technique called Geospatial Slicing and implemented a visualization tool based on this technique. Additionally, we wanted to enable the analysis of the data not only on a geospatial context, but also in reference to attribute ranges. To achieve this, we implemented a second non-geospatial view. We showed that the implementation provided can help get an overview of the data and analyse it in detail.

Concerning drawbacks, the slicing needs to be improved. Right now, the bars are newly aggregated each time the slicing is done. Therefore, the grid changes every time we slice. A possible solution to this issue is to do the slicing in steps that are the same as the width and length of the bars. This means that if we slice the data, we slice all the data points that are in a certain bar together. Another possible solution is to do the slicing after the aggregation in the shader like we did before we added aggregation to our implementation. This way not directly the data would be sliced, but the volume itself. As mentioned in Section 5.4 the aggregator provided by deck.gl prevents us from doing this, because the aggregator provides the input data for our shader.

It is planned to implement our tool into a model exploration tool of the Complexity Science Hub Vienna. Thereby, more user feedback can be collected. This can lead to more insight into how and if our tool is used by data scientists. Additionally, the user feedback could tell us which additions to our tool would be beneficial and what needs to be adjusted.

List of Figures

1.1 Illustrating the principle idea of geospatial slicing. 3

2.1 As speed increases, the speed-coupled blending lens smoothly fades into the context (from t1 to t3), and gradually fades back in when the target has been reached (t4 and t5). The inner circle fades in as the lens fades out; it delimits which region of the context gets magnified in the lens. The magnification factor remains constant [ACP10]. 6
2.2 After locking the focus of two tangible views to the same location by horizontally freezing, users can visually compare between the two views by lifting or lowering them simultaneously [STSD10]. 6
2.3 Multiscale navigation in US Census data, involving successive zooms into both space and color domains [EDF10]. 7
2.4 Element-based MoleView applied to color-mapped traffic speed image [HTE11]. 7
2.5 Four probes with focal point and radius are applied to a geospatial map. In the panes, a detailed view of the data in the area of the respective probe is shown. The multiple probes allow the user to compare the data of the different areas of interest [BDW+08]. 8
2.6 Choropleth map showing US unemployment rates of different regions from 2008, where a darker color means a higher unemployment rate [ZWC+16]. 9
2.7 Visualizing hotspots in a city using a heatmap. The locations with a large number of vehicles passing by are shown in red [ZWC+16]. 9
2.8 Example of point-based visualization of locations: Pickups (blue) and drop-offs (orange) of taxi trips in Manhattan from 7 to 8 am on May 1, 2011 are labeled by colored points [ZWC+16]. 10
2.9 The Community Health Map interface consists of a map (top), hinted double-sided sliders for map and table filtering (middle), a color coded table (bottom), and the selection panel (right). The selection panel allows users to click the link to render a layer on the map, or check boxes to display a table with the selected variables [SNK+12]. 11
2.10 Comparing statistics for four different levels of NUTS (Nomenclature of Territorial Units for Statistics) using attribute signatures with multi-bar charts [TSH+14]. 12

2.11 A stimulus containing ten bars as used in the experiment. Stimuli were represented and visualized in 2D (a) or visualized as a 3D slanted map with 3D bars (b). Imaginary lines between bar ends in sub-figure (b) are here indicated only for illustration purposes and they were not visible in the actual experiment [SC12]. 13

4.1 Sorting attributes by variance 22

4.2 Geospatial view showing percentage of unemployed people at the age of 15 years or older (dark blue), the number of male people at the age between 45 and 59 diagnosed with diabetes (blue), the number of female people at the age between 45 and 59 diagnosed with diabetes (light blue), the prevalence of diabetes for male people at the age between 45 and 59 (green) and the prevalence of diabetes for female people at the age between 45 and 59 (yellow). 23

4.3 Figure (a) to (e) showing choropleth maps with the same data as is depicted in Figure 4.2. The red colored regions have high values, while the blue colored regions have low values. For the black regions there are no data points. These maps were created with the kepler.gl [kep] demo. 24

4.4 Control Panels 25

4.5 Geospatial view showing the same attributes as Figure 4.2. Additionally, slicing along the latitude axis is applied to show a cross-section of the data. Here we can identify peaks in the data in the region of Linz and in the south of Austria. 26

4.6 Abstract view showing the prevalence of diabetes for male people at the age between 45 and 59 (green) and the prevalence of diabetes for female people at the age between 45 and 59 (yellow). The x-axis shows the employment of people at the age of 15 to 64 years in percentage. The y-axis depicts the percentage of unemployment of people at the age of 15 or older. We can see a slight increase of the combined prevalence for low employment and high unemployment. 27

4.7 Abstract view showing the same attributes and the axes as in Figure 4.6. The data is sliced at an employment of 60.341 percent an an unemployment of 1.184 to reveal cross-section. Here we can analyse the data for this value of unemployment in detail. 27

4.8 Geospatial view showing the same attributes as Figure 4.2. The data is sliced along the latitude axis to reveal the cross-section and the camera is rotated to a pitch bigger than 60°, hence the map is not rendered. 28

5.1 Program Pipeline: First the data is loaded and the maxima of the attributes are calculated. Then, the data is normalized. In the next step, the selectable attributes are sorted by variance. Afterwards, the data is sliced and then the layers are rendered. User input can trigger new normalizing, sorting, or slicing of the data. 30

6.1 Plot of the computation time of the data loading with the maxima calculation, and the data normalization in relation to the number of data points visualized 36

List of Tables

5.1 Libraries used in our implementation. 29

6.1 Hardware configuration used for our performance evaluation. 35

List of Algorithms

5.1 Data Normalization 31
5.2 Attribute Sorting 33

Bibliography

[ACP10] Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. High-precision magnification lenses. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 273–282, 2010.

[BDW+08] Thomas Butkiewicz, Wenwen Dou, Zachary Wartell, William Ribarsky, and Remco Chang. Multi-focused geospatial analysis using probes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1165–1172, 2008.

[chr] Chroma.js. https://gka.github.io/chroma.js/. Accessed: 28-05-2020.

[col] Chroma.js color palette helper. https://gka.github.io/palettes/. Accessed: 10-07-2020.

[d3] d3. https://d3js.org/. Accessed: 28-05-2020.

[dec] deck.gl. `https://deck.gl/`. Accessed: 14-05-2020.

[DPL+11] Stefan Diepenbrock, Jörg-Stefan Praßni, Florian Lindemann, Hans-Werner Bothe, and Timo Ropinski. Interactive visualization techniques for neurosurgery planning. In *Eurographics 2011, the 32th Annual Conference of the European Association for Computer Graphics, 11-15 April 2011, Llandudno, Wales, UK*, 2011.

[EDF10] Niklas Elmqvist, Pierre Dragicevic, and Jean-Daniel Fekete. Color lens: Adaptive color scale optimization for visual exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):795–807, 2010.

[GAW+11] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D Hansen, and Jonathan C Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[gpu] GPU accelerated aggregation in deck.gl. https://medium.com/vis-gl/gpu-accelerated-aggregation-in-deck-gl-7e2c7d701fb0. Accessed: 26-05-2020.

[HTE11] Christophe Hurter, Alexandru Telea, and Ozan Ersoy. MoleView: An attribute and structure-based semantic lens for large element-based plots. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2600–2609, 2011.

[jav] Time and space complexity of array.sort in v8. https://blog.shovonhasan.com/time-space-complexity-of-array-sort-in-v8/. Accessed: 25-10-2020.

[kep] kepler.gl demo. https://kepler.gl/demo. Accessed: 25-10-2020.

[KK04] Etien L Koua and Menno-Jan Kraak. Alternative visualization of large geospatial datasets. *The Cartographic Journal*, 41(3):217–228, 2004.

[map] mapbox. https://www.mapbox.com/. Accessed: 28-05-2020.

[pap] Papa Parse. https://www.papaparse.com/. Accessed: 28-05-2020.

[PM14] Chirag J Patel and Arjun K Manrai. Development of exposome correlation globes to map out environment-wide associations. In *Pacific Symposium on Biocomputing*, pages 231–242. World Scientific, 2014.

[rea] react-map-gl. https://github.com/visgl/react-map-gl. Accessed: 28-05-2020.

[RKK+13] Peter M Rabinowitz, Richard Kock, Malika Kachani, Rebekah Kunkel, Jason Thomas, Jeffrey Gilbert, Robert Wallace, Carina Blackmore, David Wong, William Karesh, et al. Toward proof of concept of a one health approach to disease prediction and control. *Emerging Infectious Diseases*, 19(12), 2013.

[SC12] Stefan Seipel and Leonor Carvalho. Solving combined geospatial tasks using 2d and 3d bar charts. In *2012 16th International Conference on Information Visualisation*, pages 157–163. IEEE, 2012.

[SNK+12] Awalin Sopan, Angela Song-Ie Noh, Sohit Karol, Paul Rosenfeld, Ginnah Lee, and Ben Shneiderman. Community health map: A geospatial and multivariate data visualization tool for public health datasets. *Government Information Quarterly*, 29(2):223–234, 2012.

[STSD10] Martin Spindler, Christian Tominski, Heidrun Schumann, and Raimund Dachselt. Tangible views for information visualization. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 157–166, 2010.

[TGK+17] Christian Tominski, Stefan Gladisch, Ulrike Kister, Raimund Dachselt, and Heidrun Schumann. Interactive lenses for visualization: An extended survey. In *Computer Graphics Forum*, volume 36, pages 173–200. Wiley Online Library, 2017.

[TMS+06] Christian Tietjen, Björn Meyer, Stefan Schlechtweg, Bernhard Preim, Ilka Hertel, and Gero Strauß. Enhancing slice-based visualizations of medical volume data. In *EuroVis*, volume 6, pages 123–130, 2006.

[TSH+14] Cagatay Turkay, Aidan Slingsby, Helwig Hauser, Jo Wood, and Jason Dykes. Attribute signatures: Dynamic visual summaries for analyzing multivariate geographical data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2033–2042, 2014.

[WLFY16] Shanshan Wei, Yang Lv, Baiin Fu, and Hiroshi Yoshino. The correlation study on the living environment and children's health problem in Dalian. *Procedia Eng*, 146:158–165, 2016.

[YKHKT18] Hui-Yi Yeh, Chen Kou-Huang, and Chen Kow-Tong. Environmental determinants of infectious disease transmission: A focus on one health concept. *International Journal of Environmental Research and Public Health*, 15(6):1183, 2018.

[YS09] Xin Yan and Xiaogang Su. *Linear regression analysis: theory and computing*. World Scientific, 2009.

[ZWC+16] Yixian Zheng, Wenchao Wu, Yuanzhe Chen, Huamin Qu, and Lionel M Ni. Visual analytics in urban computing: An overview. *IEEE Transactions on Big Data*, 2(3):276–296, 2016.