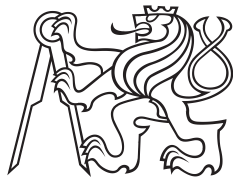


Master's Thesis



**Czech
Technical
University
in Prague**

F3

**Faculty of Electrical Engineering
Department of Computer Graphics and Interaction**

Visualization of spatiotemporal events

Bc. Daniel Jiřík

Supervisor: Ing. Ladislav Čmolík, Ph.D.

Field of study: Open Informatics

Subfield: Human-Computer Interaction

January 2025

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jiřík** Jméno: **Daniel** Osobní číslo: **492393**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávací katedra/ústav: **Katedra počítačové grafiky a interakce**
Studijní program: **Otevřená informatika**
Specializace: **Interakce člověka s počítačem**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Vizualizace časoprostorových událostí

Název diplomové práce anglicky:

Visualization of spatiotemporal events

Pokyny pro vypracování:

Analyzujte existující způsoby vizualizace časoprostorových událostí na mapě a na časové ose. Dále se seznámte s knihovnou D3.js. Na základě analýzy navrhnete a implementujete s pomocí knihovny D3.js prototyp vizualizace časoprostorových událostí, který bude schopen vizualizovat desítky až stovky časoprostorových událostí. Události mohou být různých typů (nominální atribut). Zaměřte se na přehlednou vizualizaci prostorové i časové souslednosti a umožněte filtrování událostí. Také se soustřeďte na modularitu prototypu, aby bylo např. možné snadno vyměnit jednu vizualizaci událostí na časové ose za jinou. Výsledný prototyp otestujte z hlediska rychlosti zobrazování v závislosti na množství zobrazovaných událostí. Dále výsledný prototyp otestujte pomocí testů použitelnosti alespoň s šesti účastníky.

Seznam doporučené literatury:

- [1] Tamara Munzner, Visualization Analysis and Design, A K Peters/CRC Press, 2015.
- [2] Matthew Huntington, D3.js Quick Start Guide: Create Amazing, Interactive Visualizations in the Browser with JavaScript, Packt Publishing, 2018.
- [3] Mike Kuniavsky, Observing the User Experience : A Practitioner's Guide to User Research, Elsevier, 2003.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Ladislav Čmolík, Ph.D. Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **10.09.2024**

Termín odevzdání diplomové práce: **07.01.2025**

Platnost zadání diplomové práce:

do konce letního semestru 2024/2025

Ing. Ladislav Čmolík, Ph.D.
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Podpis studenta

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Ing. Ladislav Čmolík, Ph.D., for his exceptional support, guidance, and dedication throughout my work. His willingness to assist and provide valuable insights, even during summer holidays, was truly inspiring and helped with the successful completion of this project. I am also thankful to Ing. Ivo Malý, Ph.D., and Ing. Václav Pavlovec for their contributions as part of the development team. Their input played a significant role in shaping this work. Lastly, I would like to thank my family and friends for their encouragement, helpful suggestions, and belief in me, which kept me motivated throughout this journey.

Declaration

I hereby declare I have written this Master's thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, 4. January 2025

Abstract

This Master's thesis focuses on making it easier to understand events that occur in both time and space. The work presents a tool that takes complex aviation data and shows it on an interactive map with supportive charts and table. This helps users quickly see patterns, filter information, and highlight important details. The application is built using D3.js, which allows for flexible and engaging visuals. Feedback from users guided the design, leading to a final solution that makes exploring data more intuitive. This work offers a new idea to support better decision-making in aviation and other fields where understanding when and where things happen is key.

Keywords: data visualization, spatiotemporal data, user-centered design, D3.js, aviation safety

Abstrakt

Tato diplomová práce se zaměřuje na zjednodušení porozumění událostem, které se zároveň odehrávají v čase i prostoru. V práci je představen nástroj, který složité data z letectví zobrazuje na interaktivní mapě doplněné o grafy a tabulku. Uživatelům to umožňuje rychle odhalit vzorce, filtrovat informace a zvýraznit důležité detaily. Aplikace byla vytvořena pomocí knihovny D3.js, jež umožňuje flexibilní a poutavé grafické zobrazení dat. Návrh byl průběžně konzultován s uživateli, díky čemuž vzniklo řešení, které usnadňuje prozkoumávání dat. Tato práce přináší nový nápad jak podpořit lepší rozhodování v oblasti letectví i v dalších oborech, kde je důležité pochopit, kdy a kde k událostem dochází.

Klíčová slova: vizualizace dat, časoprostorová data, user-centered design, D3.js, bezpečnost v letectví

Contents

1 Introduction	1
1.1 Background on Aviation Safety ..	2
1.2 Safety Management System (SMS)	3
1.3 Objectives of the Thesis	4
2 Safety Indicators in Aviation	5
2.0.1 Safety-I Indicators: Looking at Problems	5
2.0.2 Safety-II Indicators: Looking at Success	5
2.0.3 Combining Safety-I and Safety-II	5
2.0.4 Challenges in Defining Aviation Safety Indicators	6
2.0.5 Using Maps to Understand Safety	7
2.1 Relevance to Václav Havel Airport	7
3 Data Visualization Concepts	9
3.1 Attribute Types	9
3.1.1 Categorical	9
3.1.2 Ordered	9
3.2 Visual Channels and Marks	10
3.2.1 Channels	10
3.2.2 Marks	11
3.2.3 Practical Considerations for spatiotemporal aviation data application	12
3.2.4 Marks in Aviation Safety Context	12
3.3 Geospatial data	13
3.4 Visualization of Geospatial Data	13
3.4.1 Point Maps	13
3.4.2 Heat Maps	14
3.4.3 Hexagonal Binning	14
3.4.4 Choropleth Maps	15
3.5 Spatiotemporal Data	16
3.5.1 Challenges in Spatiotemporal Data Representation	16
3.6 Visualization of Spatiotemporal Data	17
3.6.1 Space-Time Cube (STC)	17
3.6.2 Animated Maps	17
3.6.3 Motion Charts	18
3.6.4 Space-Time Paths	19
4 D3.js	21
4.1 Data Binding in D3.js	21
4.1.1 Example: Binding Data to Circle Elements	21
4.2 DOM Manipulation	22
4.2.1 Example: Adding Text Labels	23
4.2.2 The General Update Pattern	23
4.2.3 Example of Enter, Update and Exit	23
4.3 Special DOM Manipulations in D3.js	24
4.4 Interactivity in D3.js	25
4.5 Performance of D3.js	25
4.5.1 SVG, Canvas and WebGL ..	25
4.6 Discussion of D3.js Components in the Aviation Data Visualization Application	26
5 User-Centered Design	27
5.1 Key Activities in User-Centered Design	27
5.2 Usability Methods in User-Centered Design	28
5.2.1 Categories of Usability Methods	28
5.2.2 Structure of Usability Methods	28
5.3 Selecting Appropriate UCD Methods	29
5.3.1 Selection Criteria	29
5.3.2 Using UCD in Our Application	29
6 Application Development	31
6.1 Objectives of the Application...	31
6.2 Prototyping process	31
6.2.1 Overview of Prototyping Approaches	32
6.2.2 Fidelity in Prototyping	32
6.3 Prototype for Aviation Data Visualization Application	33
6.3.1 Overview and Initial Design Process	33
6.3.2 Iterative Development Process	33
6.3.3 Evolution of the Application.	34
7 Application Implementation Approach	39
7.1 Overview	39
7.1.1 Project Structure	39
7.2 Filtering and Data State	42

7.3 Shared Domains and Reactive Updates	42
7.4 Independent Logic and Visual Layers	42
7.5 Extensibility	42
8 Final Results	43
8.1 Overview of the Application....	43
8.2 Core Features	44
8.2.1 Filter Component	44
8.2.2 Data Graph	45
8.2.3 Data Table	48
8.2.4 Map Visualization.....	49
8.3 Performance Testing	54
9 User Testing	57
9.1 Early Testing	57
9.2 Improved user testing	58
9.3 Results and Discussion	59
9.3.1 Initial Impressions and Core Features	59
9.3.2 Map Interactions.....	60
9.3.3 Graph Interactions and Data Interpretation	60
9.3.4 Overall Experience and Suggested Improvements	60
9.3.5 Questionnaire Results	60
9.4 Testing summary	63
10 Conclusions	65
A Used Software	67
B Content of Electronic Appendix	69
Bibliography	71

Figures

1.1	A symbol map showing customer preferences by region. Larger circles represent more customers who liked the product [1].	1
1.2	Graph showcasing the ongoing improvement of safety.	3
2.1	Comparison between lagging and leading safety indicators. Lagging indicators measure incidents after they occur, while leading indicators focus on preventing incidents before they happen [2].	6
2.2	Safety comic strip illustrating operational and safety challenges in the handling of passengers at Václav Havel Airport [3].	8
3.1	Examples of visual channels on maps [4].	11
3.2	Examples of marks on maps [4].	11
3.3	Example of a Point Map [5]. . .	13
3.4	Example of a Heat Map [6]. . .	14
3.5	Example of Hexagonal Binning [7].	15
3.6	Example of a Choropleth Map [8].	16
3.7	Example of a Space-Time Cube [9].	17
3.8	Example of an Animated Map [10].	18
3.9	Example of a Motion Chart [11].	19
3.10	Example of a Space-Time Path [12].	20
6.1	First prototype.	34
6.2	Second prototype.	35
6.3	Third prototype.	35
6.4	Fourth prototype.	36
6.5	Fifth prototype.	36
6.6	Final version.	37
8.1	Final integrated view of the application with filters, graph, map, and table components. . .	43
8.2	Filter interface allowing users to narrow down the event dataset by time and type.	44
8.3	Events visualized by type, allowing users to distinguish categories more easily.	45
8.4	Events visualized by time, allowing users to distinguish data times more easily.	45
8.5	The initial graph layout before showing sub-graphs	45
8.6	An unpacked bin graph representation, showing events grouped by time intervals.	46
8.7	Highlighted section of the bin graph reflected on the map component, showcasing interactive linking.	46
8.8	Highlighted selection of points in the map propagating to main graph and sub-graphs.	47
8.9	Tooltip details for a particular graph selection revealing selected time range	47
8.10	Sub-graphs enable better analysis of different event types.	48
8.11	Sorting part is highlighted in yellow rectangle, users can sort using drag and drop	48
8.12	A single highlighted event in the data table sorted by type, which also reflects on the map and graph.	49
8.13	Map visualization with events plotted according to their actual airport location.	50
8.14	Type tooltip shown when hovered over a point. Visible for selected visualization: "By time".	51
8.15	Time tooltip shown when hovered over a point. Visible for selected visualization: "By type".	51
8.16	Using rectangular selection to highlight a specific data on the map.	52
8.17	Highlighted points on the map.	53

8.18 Type based visualization of the map and graph. Changes graph color to gray 54

8.19 Time based visualization of the map and graph. Changes graph color to interpolated color representing time. 54

9.1 Ratings for the clarity of the user interface. (1 = Strongly Disagree, 5 = Strongly Agree) 61

9.2 Ratings for the effectiveness of using filters. (1 = Strongly Disagree, 5 = Strongly Agree) . 61

9.3 Ratings for the clarity of the visualizations. (1 = Strongly Disagree, 5 = Strongly Agree) . 62

9.4 Ratings for usefulness in analyzing data and identifying clusters. (1 = Strongly Disagree, 5 = Strongly Agree) 62

9.5 Ratings for the speed of finding information. (1 = Strongly Disagree, 5 = Strongly Agree) . 62

9.6 Ratings for user confidence. (1 = Strongly Disagree, 5 = Strongly Agree) 63

9.7 Ratings for overall satisfaction. (1 = Strongly Disagree, 5 = Strongly Agree) 63

Chapter 1

Introduction

Today, the amount of data being created is growing faster than ever. This brings many challenges because it can be hard to understand and use all this information. One of the best ways to make sense of data is through visualization. Data visualization is a way of turning numbers and other complex information into pictures or graphs that are easier to understand. It helps people see patterns, trends, and connections that they might miss otherwise. This is why it is used in many areas like science, business, and safety [13]. An example of big data visualization can be seen in the Figure 1.1

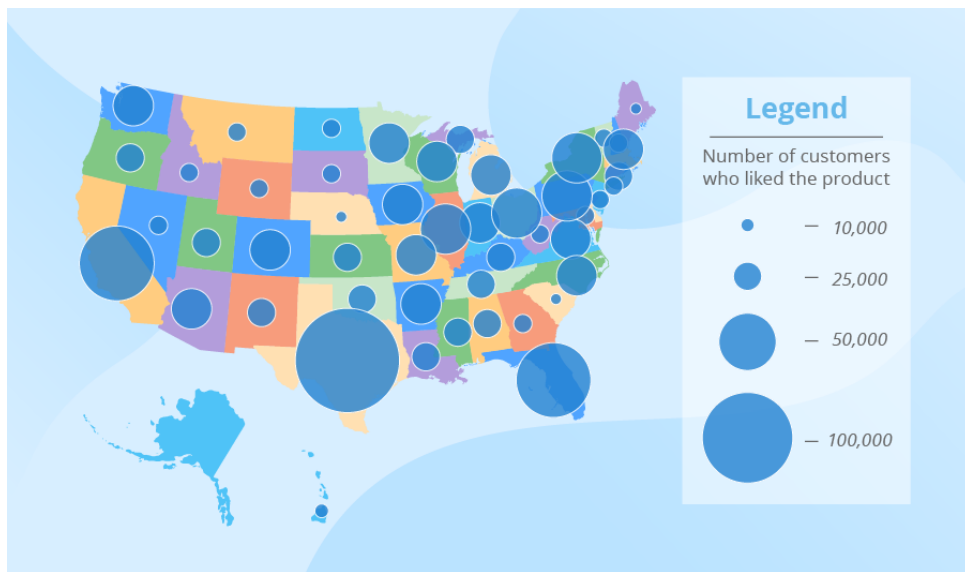


Figure 1.1: A symbol map showing customer preferences by region. Larger circles represent more customers who liked the product [1].

Humans are naturally better at understanding visuals than abstract numbers or text. A graph, a map, or a simple chart can tell a story much faster and more clearly than pages of data ever could. This makes visualization a kind of universal language—one that feels intuitive for all sorts of people.

The challenges of today's data-driven world can also be seen in complex environments like airports, where huge amounts of data are generated daily

to ensure safety and efficiency. Airports are dynamic systems with constant movement, where understanding and managing safety risks is crucial. Visualization tools have the potential to make these data more accessible, helping to support decision-making and improve safety practices. By exploring visualization techniques, this thesis highlights how data can be transformed into understandable form, even in environments as complex as airports.

This thesis focuses on the development of a data visualization tool specifically tailored for Václav Havel Airport. The tool aims to assist airport staff in understanding safety-related data by presenting it in an intuitive and interactive format.

1.1 Background on Aviation Safety

Aviation safety has come a long way over the years, shifting from only reacting to incidents to actively working to prevent them. This progress is thanks to lessons learned, technological advancements, and teamwork among industry players. Even with the complexities of modern air travel, the industry remains committed to safety, always trying to improve and reduce risks.

In the 1960s and 1970s, the aviation industry began adopting concepts from quality management systems to make safety better. These early efforts started the structured safety practices [14].

By the early 1990s, the idea of Safety Management Systems (SMS) began to form in the airline industry. This approach shifted from reacting to incidents and focused on proactively identifying and managing potential hazards [15].

In March 2006, implementing SMS became mandatory worldwide for certain airlines, which was a significant step in global aviation safety [15].

The 2010s saw further advances in technology, pilot training, and operational procedures, which caused a noticeable drop in accident rates [16]. However, challenges remained, especially during takeoff and landing phases, often due to human factors, bad weather, or technical issues. These experiences led to better teamwork and clearer rules, helping flight crews manage difficult situations more effectively.

By 2022, the impact of these efforts was clear [17]. Over 32 million flights took place, showing a strong recovery from the pandemic years, and the fatal accident rate continued to decrease. This showed the aviation industry's strength in maintaining safety as a priority during a busy period. Still, incidents like a runway collision in Lima, which sadly caused deaths, highlighted the need for further vigilance. Issues like weather challenges, communication breakdowns, and not sticking to procedures remained the main areas for improvement.

In 2023, the aviation industry reached a remarkable milestone: for the first time, there were no fatal accidents or hull losses involving jet aircraft [18]. This achievement proved the success of aviation industry's dedication to safety. However, a fatal accident in Nepal involving a turboprop aircraft reminded everyone that risks still exist, especially related to human error and tough conditions.

From the early efforts in the 1960s to the significant achievements by 2023, aviation safety has shown continuous progress which you can also see on the Figure 1.2. The industry has made it clear that it is committed to learning from past incidents, turning those lessons into effective strategies. The collective efforts of everyone involved ensure that flying is still one of the safest ways to travel.

Fatal airliner accidents per million commercial flights globally

Commercial airliners (passenger-only and cargo) with a capacity for more than 14 passengers.

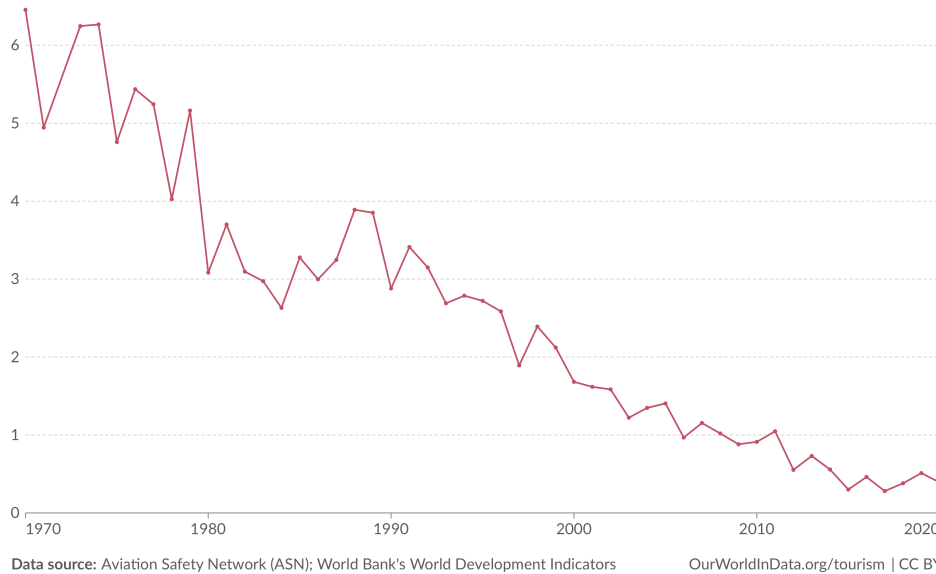


Figure 1.2: Graph showcasing the ongoing improvement of safety.

1.2 Safety Management System (SMS)

The introduction of Safety Management Systems (SMS) brought a big change in how the aviation industry approaches safety. Unlike the old reactive approach of "fly-crash-fix-fly [19]," SMS represents a proactive system designed to spot and reduce risks before they lead to incidents. SMS focuses on making safety an integral part of every aspect of aviation operations.

SMS combines structured processes, risk management practices, and organizational culture to ensure an in-depth approach to safety. SMS not only focuses on reducing errors but also works to prevent them by identifying hazards and dealing with risks effectively. It makes sure that safety is a shared responsibility among all participants.

The roots of SMS are based in quality management principles, similar to principles in other high-stakes industries like construction and chemical production. These principles focus on continuous improvement, feedback, and safety policy [19].

International use of SMS, as discussed in the aviation background on safety chapter, began in the early 21st century, with regulatory bodies like the

International Civil Aviation Organization (ICAO) requiring its implementation for member states in 2006. Today, SMS is a key part of aviation safety, encouraging consistent rules and teamwork across countries. This global approach not only ensures everyone follows the rules but more importantly helps the industry reduce risks [19].

With systems like SMS, the aviation industry continues to improve safety by combining technical skill with proactive and preventative actions.

1.3 Objectives of the Thesis

The aim of this thesis is to develop a tool specifically designed to visualize aviation-related events occurring at particular locations and times. These events will be presented on an interactive map, graph and table. The tool will enable users to interact with the data, such as filtering events to focus on specific criteria. This thesis will also talk about common challenges in visualizing spatiotemporal data.

The first part of the thesis will talk about the data being used, which comes from two aviation safety perspectives and their indicators. These indicators are important for understanding risks and making traveling safer. They will help decide how to design the tool so that the data is clear and useful for the people using it.

The tool will be created using the D3.js library, which is a framework for building visualizations. This part of the thesis will explain how D3.js works and why it is a good choice for this project.

The User-Centered Design (UCD) approach will help with the design of the tool. The tool will be developed step by step, starting with a basic version of the tool that can be tested and improved. The design will be modular, meaning that parts of the tool can be changed or replaced without rebuilding the whole application. This will make the tool flexible and ready for different modules in the future.

The tool will have several connected key features. It will include a map to show where events happen, a timeline graph to show when they happen, and a data table to explain data attributes. Users will be able to filter events by type or time.

Finally, the thesis will test the tool to see how well it works. Usability testing will involve seven participants trying the tool and giving feedback on how easy it is to use. The results will help improve the tool and make it more practical. The strengths and weaknesses of the tool will be discussed and new ideas for future improvements based on results.

This thesis will show how to create a tool that is simple and helpful. Using D3.js, it will focus on user needs to present clear and interactive ways to understand events in time and place. The tool aims to help people make better decisions and improve safety in aviation and other areas.

Chapter 2

Safety Indicators in Aviation

Safety is the most important part of aviation. To keep flying safe, it is important to measure and understand how well safety is working. One way to do this is by using safety indicators. These indicators help track safety over time, find risks, and check if safety measures are working. By looking at the data from safety indicators, aviation experts can spot problems, understand their causes, and take action to improve safety [20].

In a Safety Management System (SMS), there are two main approaches: Safety-I and Safety-II. In Safety-I we discuss Lagging Indicators (Reactive) and in Safety-II we discuss Leading Indicators (Proactive and Predictive). In Figure 2.1 you can see an image simply explaining these indicators. Keep in mind that these indicators are not only used in aviation but for example also in different places like construction or manufacturing industry.

■ 2.0.1 Safety-I Indicators: Looking at Problems

The traditional way, called Safety-I, focuses on what goes wrong. Safety-I (Lagging) indicators track things like mistakes, accidents, or near misses. These numbers help safety managers find problems and fix them so they don't happen again. For example, a Safety-I indicator might show how many bird strikes happen at an airport. This kind of data is good for finding weak spots and making plans to prevent future issues.

■ 2.0.2 Safety-II Indicators: Looking at Success

The newer way, called Safety-II, looks at what goes right. It focuses on how people and systems work well, even in tough situations. Safety-II (Leading) indicators measure things like how often air traffic controllers handle busy traffic successfully or how pilots adjust to bad weather for smooth landings. These indicators show what makes the system work well.

■ 2.0.3 Combining Safety-I and Safety-II

When both types of indicators are used, safety management becomes more complete. Safety-I shows what needs fixing, while Safety-II shows what is

working well (which does not necessarily mean it will work well forever). Together, they give a balanced view of safety in aviation.

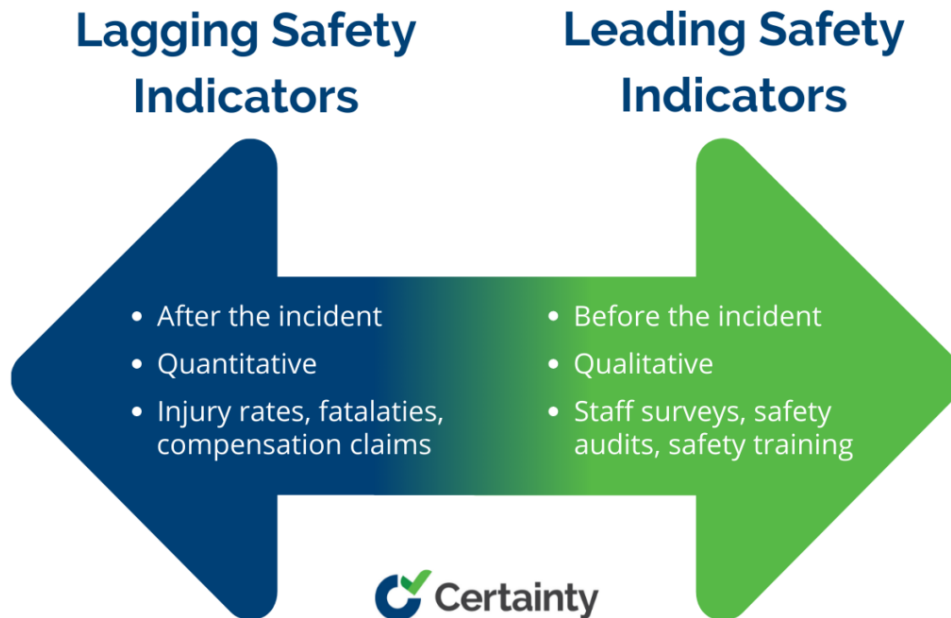


Figure 2.1: Comparison between lagging and leading safety indicators. Lagging indicators measure incidents after they occur, while leading indicators focus on preventing incidents before they happen [2].

■ 2.0.4 Challenges in Defining Aviation Safety Indicators

Defining aviation safety performance indicators is a complex task due to several challenges [21]. One significant issue is the inconsistency in metrics, as different organizations, such as air traffic service providers, airlines, and maintenance organizations, often develop their own indicators independently, leading to a lack of standardization. Additionally, making sure that the data are reliable and have good quality is challenging due to things like underreporting and variability in data collection methods. Another difficulty lies in balancing reactive indicators, which reflect past events, with proactive indicators that prevents future risks.

Safety-II indicators are harder to connect directly to preventing accidents. Combining Safety-II with other systems, like Occupational Health and Safety Management Systems (OHSMS) [21], can cause conflicts because these systems often have different goals. Composite indicators, which mix several safety factors into one measure, can sometimes hide risks because they rely on personal opinions to decide what is most important. Lastly, comparing safety performance between groups, like airlines and airports, can be tricky. This is because they work in different ways and measure safety differently.

2.0.5 Using Maps to Understand Safety

Showing safety data on maps makes it easier to understand and use. For example, Safety-I indicators, like where accidents or incidents happen, can show problem areas. Safety-II indicators, like places where workers handle challenges well, can show what is working and can give a hint why.

Maps make it simple to see patterns and trends, helping safety teams and airport managers decide where to focus—whether to solve problems that happened or improve what's working or improve what's getting worse. This goes well with Safety Management Systems (SMS) because it helps use resources wisely and find good solutions.

2.1 Relevance to Václav Havel Airport

Václav Havel Airport Prague uses a Safety Management System (SMS) to keep operations safe. This system combines three main approaches: predictive, proactive, and reactive. Predictive methods, like safety studies, help find potential risks before they happen. Proactive methods, such as regular audits and inspections, helps to spot and fix problems early. Reactive methods focus on investigating incidents to understand what went wrong and prevent it from happening again [22].

One of Václav Havel SMS component is the promotion of safety awareness among all personnel. This includes educational programs that share safety rules and good practices in a fun and easy-to-understand way. For example, Prague Airport has developed a comic series [23] addressing topics like Foreign Object Debris (FOD) hazards, airside movement protocols, and the importance of maintaining equipment. In Figure 2.2 you can see one official comic strip.

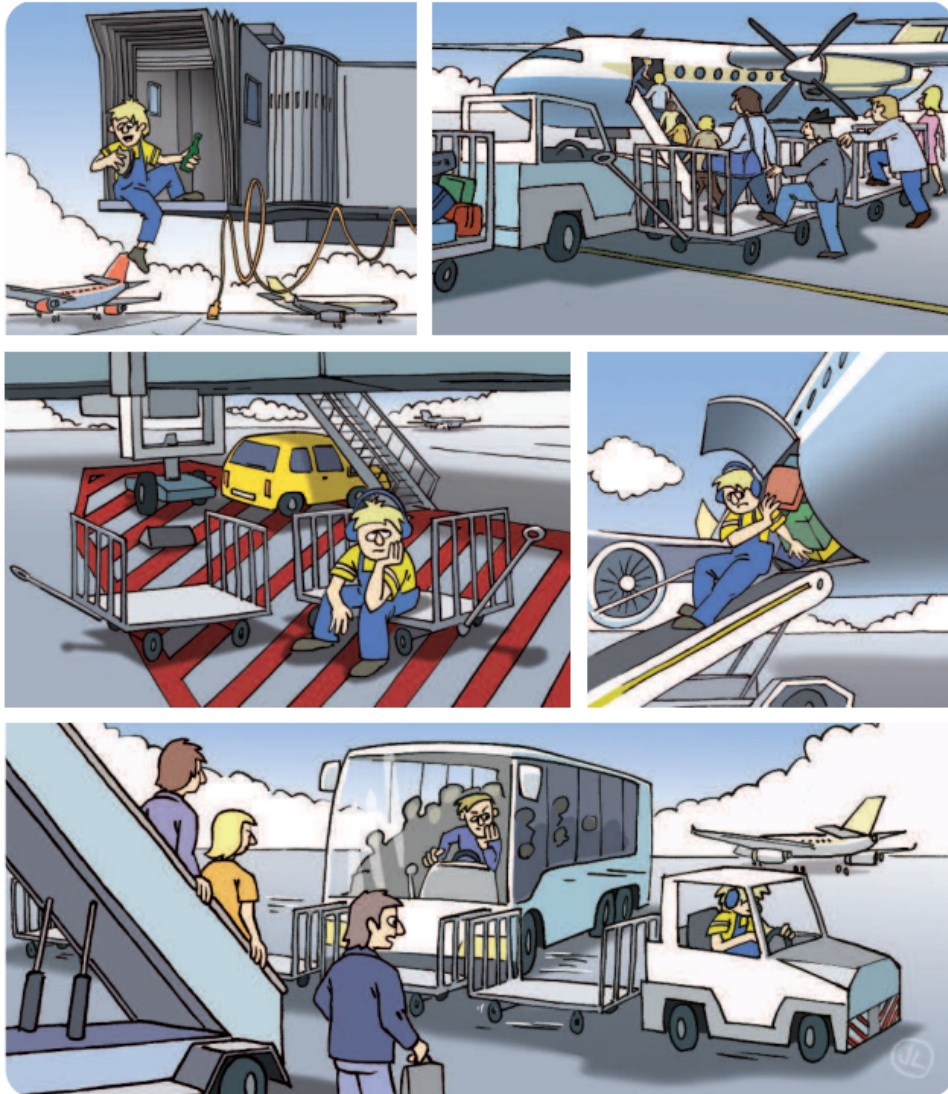


Figure 2.2: Safety comic strip illustrating operational and safety challenges in the handling of passengers at Václav Havel Airport [3].

Václav Havel Airport also does "Safety Briefs [24]." These briefs provide up-to-date information on operational safety, addressing topics such as construction activities, medical incidents, and runway incursions. By often doing these briefs, the airport ensures that all staff are informed about current safety concerns and best practices.

The airport also has safety campaigns to focus on specific safety issues. An example is the "Week of Operational Safety," which helps to support global efforts to raise awareness about safety in aviation. These campaigns help create a strong safety culture, encouraging everyone to stay alert and continue improving safety.

Chapter 3

Data Visualization Concepts

Data visualization is a powerful way to understand and make sense of complex information. It transforms raw data into simple pictures or graphs that are easier to interpret. This makes it possible to spot patterns, trends, and connections in the data quickly and effectively. In today's data-driven world, visualization has become an essential tool across industries, supporting better decision-making and problem-solving.

In this chapter, we explore the basic techniques of data visualization and data that changes over time and space. A key part of visualization is using visual channels, like color, size, and position, and marks, such as points, lines, and areas. These tools help create meaningful visuals created for the audience and the characteristics of the dataset.

While visualization is powerful, it comes with challenges. Managing large datasets, accurately representing relationships, and choosing the right techniques for different data types can be difficult. This chapter will address these issues and introduce techniques to deal with those challenges.

3.1 Attribute Types

We have different attribute types that are used in data analysis. Understanding these attribute types is essential for choosing appropriate visualization and analysis techniques.

3.1.1 Categorical

Categorical (nominal) attributes are types of data that can be divided into specific categories or groups. Each category is unique, and there is no order between them. Examples include types of fruits (apple, orange, banana) or car brands (Toyota, Ford, BMW) [25].

3.1.2 Ordered

Ordered attributes are characteristics of data that have a meaningful sequence or order. They can be divided into two subcategories: ordinal and quantitative.

Ordinal

Ordinal attributes are similar to categorical attributes, but with a clear order or ranking among the categories. The intervals between these categories are not necessarily equal. Examples include education levels (high school, bachelor's degree, master's degree) and satisfaction ratings (poor, fair, good, excellent) [26].

Quantitative

Quantitative attributes are numerical and can be measured. They can be further divided into:

- **Interval:** Data with meaningful intervals between measurements but no true zero point (e.g., temperature in Celsius).
- **Ratio:** Data with meaningful intervals and a true zero point (e.g., weight, height).

3.2 Visual Channels and Marks

Visual channels and marks are key parts of data visualization. They play different but complementary roles in showing information [4].

3.2.1 Channels

Visual channels are ways to represent data visually on maps. These include color, size, shape, position, and opacity. Each channel can be adjusted to show different values, making it easier for people to understand the information on a map.

Choosing the right visual channel is important because people notice some changes better than others. For example, humans are better at noticing differences in position or length compared to color or shape. On maps, position is a natural channel because it shows where things are located. For example, population density can be shown using a color gradient, or the number of people in cities can be represented by the size of circles.

When designing maps, it is important to match the data with the most effective channel. For example, categorical data (like natural disaster types) can be shown using different colors or shapes, while quantitative data (like rainfall) can be represented using size or color saturation.

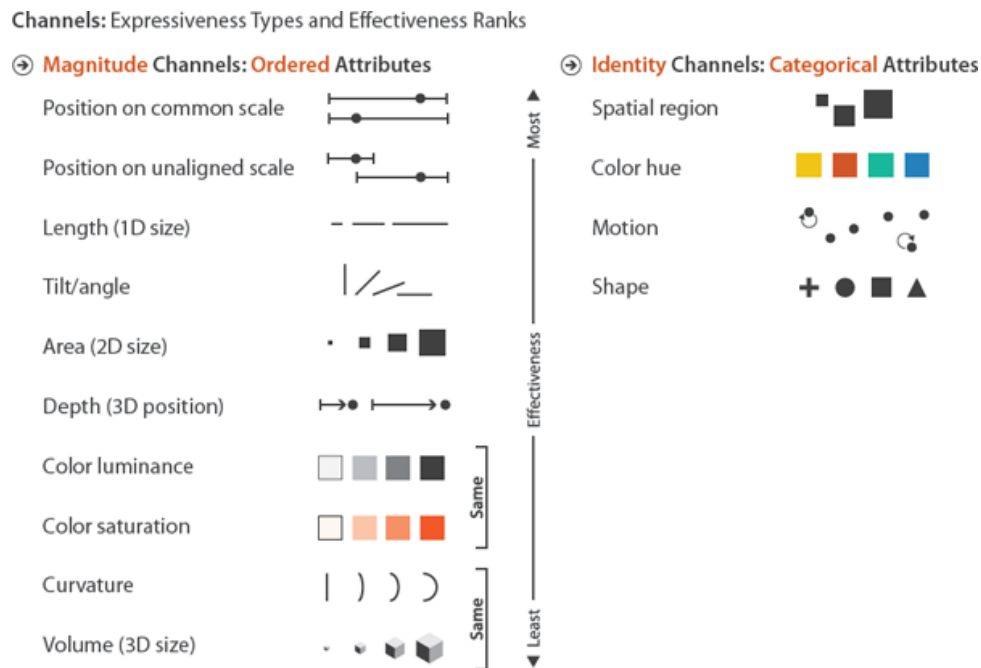


Figure 3.1: Examples of visual channels on maps [4].

3.2.2 Marks

Marks are the basic elements used to display data on maps. Common types of marks are points, lines, and areas. Each type is useful for different purposes:

- **Points:** Show specific locations like cities, weather stations, or events. Points can change in size, color, or shape to represent different values, such as population size or event type.
- **Lines:** Show connections or paths like roads, rivers, or flight routes. Line thickness or color can represent quantities such as traffic or river flow.
- **Areas:** Represent regions, such as countries, cities, and villages. Another example can be that the colors or patterns in areas can show variations, such as election results, vegetation types or population density.

Combining marks with visual channels helps make maps more effective. For example, a map could use colored areas to show temperature zones and points of different sizes to represent cities by population.

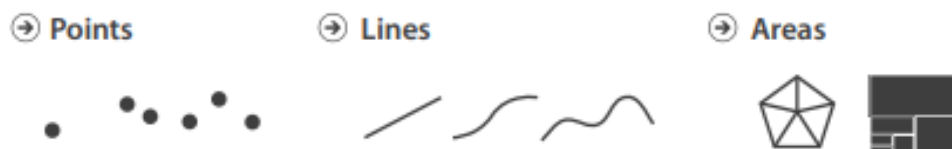


Figure 3.2: Examples of marks on maps [4].

3.2.3 Practical Considerations for spatiotemporal aviation data application

Aviation safety data often involve both *categorical* attributes (e.g., incident types such as bird strikes, runway incursions) and *ordered* attributes (e.g., timestamps). In this context of 2D application:

- **Effective Channels:**

- *Position on common scale* works very well to represent numeric values along a timeline.
- *Color hue* is effective for distinguishing incident *types* (categorical data).
- *Color saturation* or *luminance* can show varying levels of severity or frequency.
- *Area (2D size)* can highlight higher incident rates in regions on a map (e.g., larger circles for more frequent events).

- **Less Effective Channels:**

- *Tilt/angle and curvature* tend to be less intuitive and can confuse users, especially if the data does not inherently involve angular or curved pathways.
- *Volume (3D size)* is impractical for a 2D interface and can easily clutter a screen without adding clarity to safety-related data.
- *Depth (3D position)* is similarly irrelevant for a flat layout and can make spatial data harder to interpret if actual elevation/altitude is not a primary concern.
- *Motion* (e.g., animations) can be distracting if overused for static incident data; it is more suitable when depicting changes over time or progressive transformations.

3.2.4 Marks in Aviation Safety Context

- **Points** are particularly useful for plotting discrete occurrences like incidents at specific airport locations.
- **Lines** may be relevant if examining flight paths or trajectories, but are less critical for point-based incidents.
- **Areas** could be employed for partitioning the airport into zones (e.g., runway, taxiway, apron) and visually aggregating incident data per zone.

Overall, *position*, *color hue*, and *size* channels, in combination with *point* and *area* marks, form a strong foundation for aviation data visualization in 2D. Ensuring these visual encodings align with user expectations is crucial, since overly complex or less intuitive channels (like tilt, 3D depth, or excessive motion) may make worse clarity in the final application.

3.3 Geospatial data

Geospatial data [27] refers to information that includes location-based attributes on or near the Earth's surface. It can be categorized into two types: vector data, which represents discrete objects like points, lines, and polygons (e.g., roads, buildings), and raster data, which consists of grid cells or pixels (e.g., satellite imagery, elevation models). Geospatial data is important in places like urban planning, disaster management, and navigation.

3.4 Visualization of Geospatial Data

In this section we will discuss different techniques that are usually used for visualizing geospatial data [28]. These visualizations can be altered to represent spatiotemporal data. In the aviation project application point map technique which is discussed in 3.4.1 has been altered and used to fit the needs of the data time component. Different techniques that were more specifically designed for spatiotemporal data will be discussed in later sections 3.5 and 3.6.

3.4.1 Point Maps

A point map displays individual data points on a geographic map, primarily using **points** as marks. The **position** of each point shows its exact location on the map. Additionally, **color hue** or **shape** can be used to represent different categories or groups. Point maps are particularly effective for visualizing precise event locations or object positions.

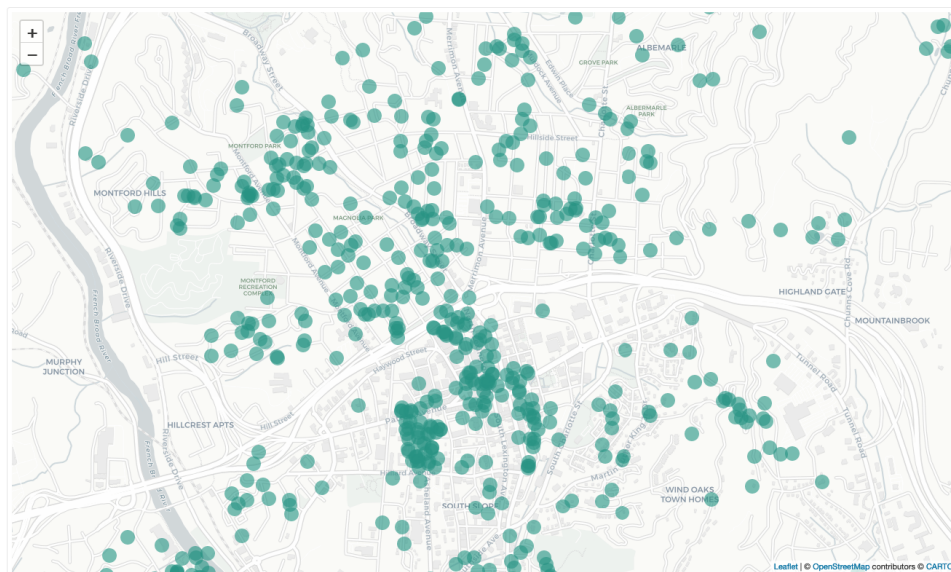


Figure 3.3: Example of a Point Map [5].

3.4.2 Heat Maps

A heat map uses **color luminance** or **color saturation** to show the intensity of data values across a geographic area. It uses **areas** on the map but encodes how “hot” or “dense” a region is with color. This is useful for displaying overall patterns, like population density or crime rates, because it makes high-intensity regions stand out.

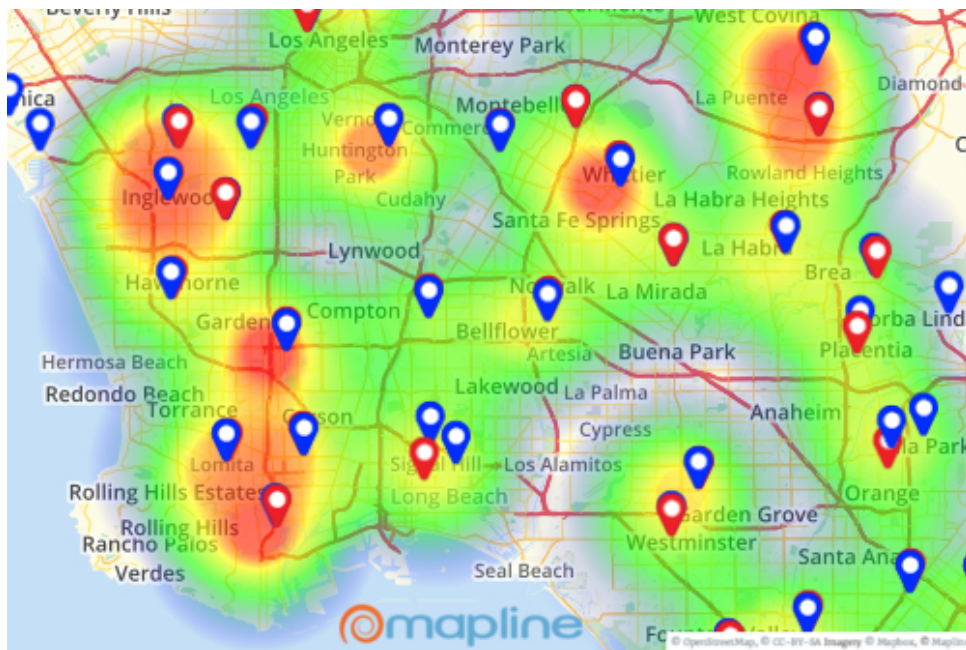


Figure 3.4: Example of a Heat Map [6].

3.4.3 Hexagonal Binning

Hexagonal binning divides a map into many small **hexagonal areas**, then uses color or shading (for example, **color luminance** or **color saturation**) to show how many data points fall into each cell. It is good for large datasets because overlapping points are combined into clearer regions, making patterns easier to see.

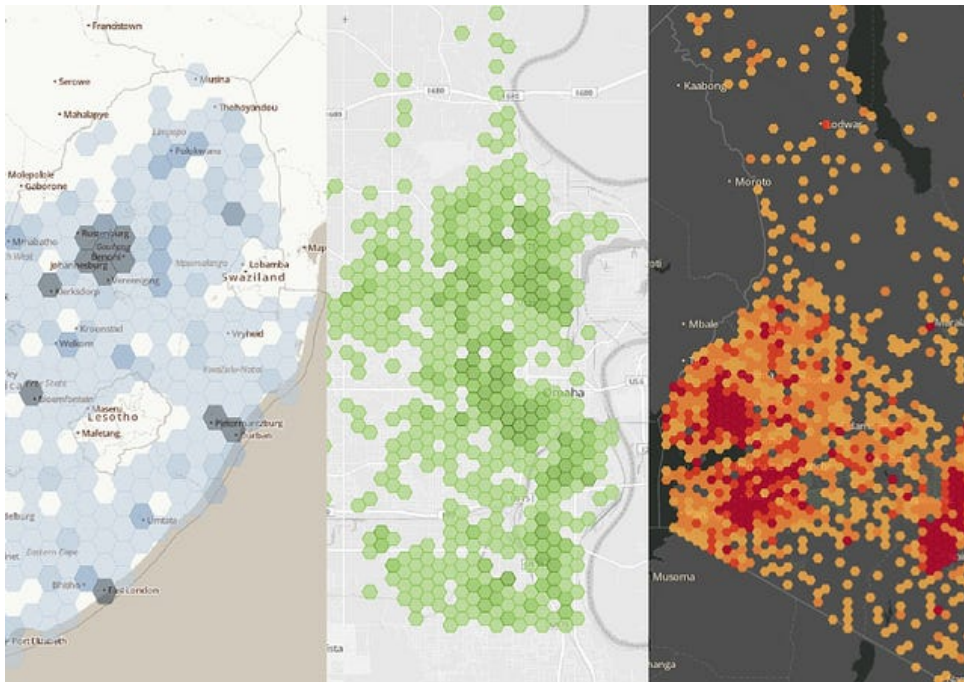


Figure 3.5: Example of Hexagonal Binning [7].

3.4.4 Choropleth Maps

A choropleth map shades or colors different **areas** based on a specific value, such as population or income. It mostly uses the **area** mark and the **color hue** or **color saturation** channel to show how the value changes from region to region. Choropleth maps are good for showing how a single metric varies across a broad area.

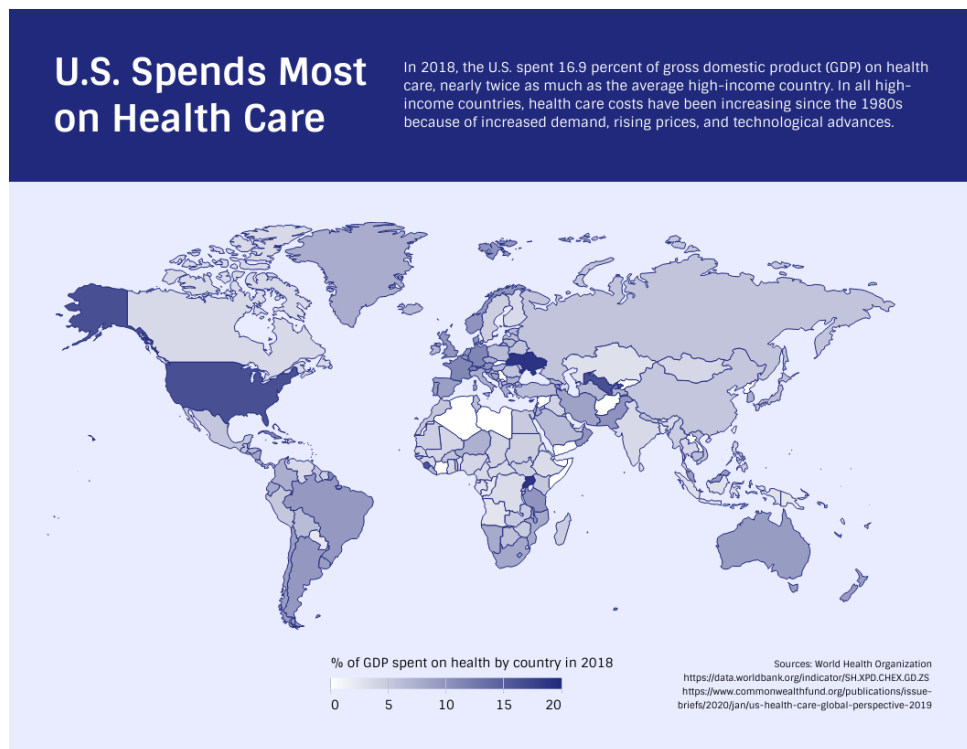


Figure 3.6: Example of a Choropleth Map [8].

3.5 Spatiotemporal Data

Spatiotemporal data is information that changes over time and across locations. It is important in fields like aviation, city planning, and public health. Visualizing this data helps us identify patterns and gain insights that are hard to see in raw numbers.

Spatiotemporal data can include both categorical and ordered information. Categorical data is grouped into categories, like weather conditions (rainy, sunny). Ordered data follows a meaningful sequence, such as hourly temperature readings or traffic volume during the day. Choosing the right visualization technique depends on understanding these data types.

3.5.1 Challenges in Spatiotemporal Data Representation

Representing spatiotemporal data comes with many challenges [29] due to its complexity and the fact that it changes across both space and time. One major issue is managing the complex structures needed to show the connections between location and time. Data quality can also be a problem, with issues like gaps in the data, errors from sensors, or uncertain measurements. Handling the huge amounts of data generated by things like satellites and sensor networks adds to the difficulty. Combining data from different sources and methods across various fields makes the process even more complicated. Creating effective visualizations of spatiotemporal patterns requires advanced

tools that can clearly show the relationships and trends. Solving these challenges is important for making accurate decisions in areas like monitoring the environment, planning cities, and improving public health.

3.6 Visualization of Spatiotemporal Data

Spatiotemporal data visualization shows how things change over time and across locations. Many techniques are similar to those used for purely geospatial data (e.g., heat maps, point maps), but they add the time dimension through appropriate visual channels and marks. For instance, a point map might use *color hue* (an identity channel) to distinguish time periods or *color saturation* (a magnitude channel) to indicate intensity changes over time. Below are some additional ways to visualize spatiotemporal data:

3.6.1 Space-Time Cube (STC)

The Space-Time Cube (STC) [9] is a 3D organizational approach by stacking data slices from different times into a cubic volume (using *volume* as a 3D size channel). The X and Y axes represent spatial location, while the Z axis encodes time. Each volumetric cell (a 3D mark) corresponds to a specific place and time, which helps to detect patterns through both the spatial (X, Y position) and temporal (Z position) dimensions.

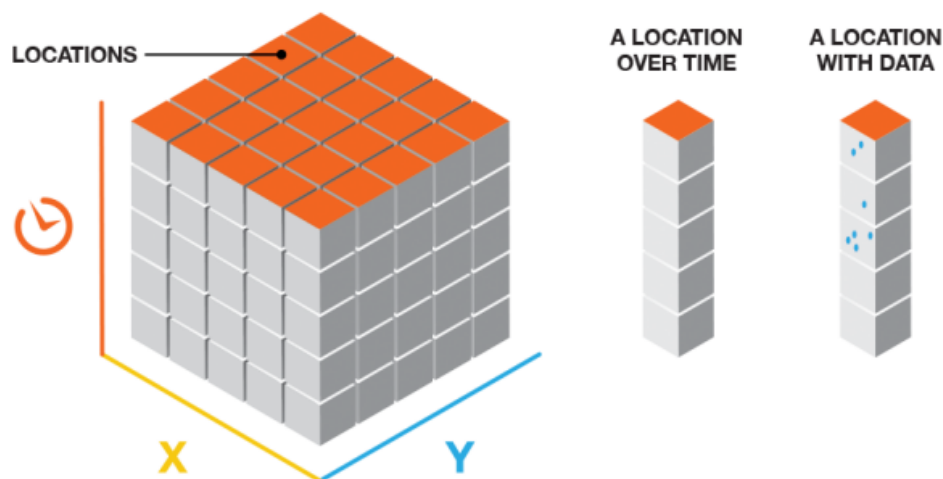


Figure 3.7: Example of a Space-Time Cube [9].

3.6.2 Animated Maps

Animated maps [10] show changes over time by playing a sequence of map frames (leveraging *motion* as an identity channel). They often use *spatial*

region to represent location and can apply *color hue* or *color saturation* to highlight variations in magnitude (such as different storm intensities). Points, lines, or areas (the fundamental mark types) may change in position or appearance across successive frames.

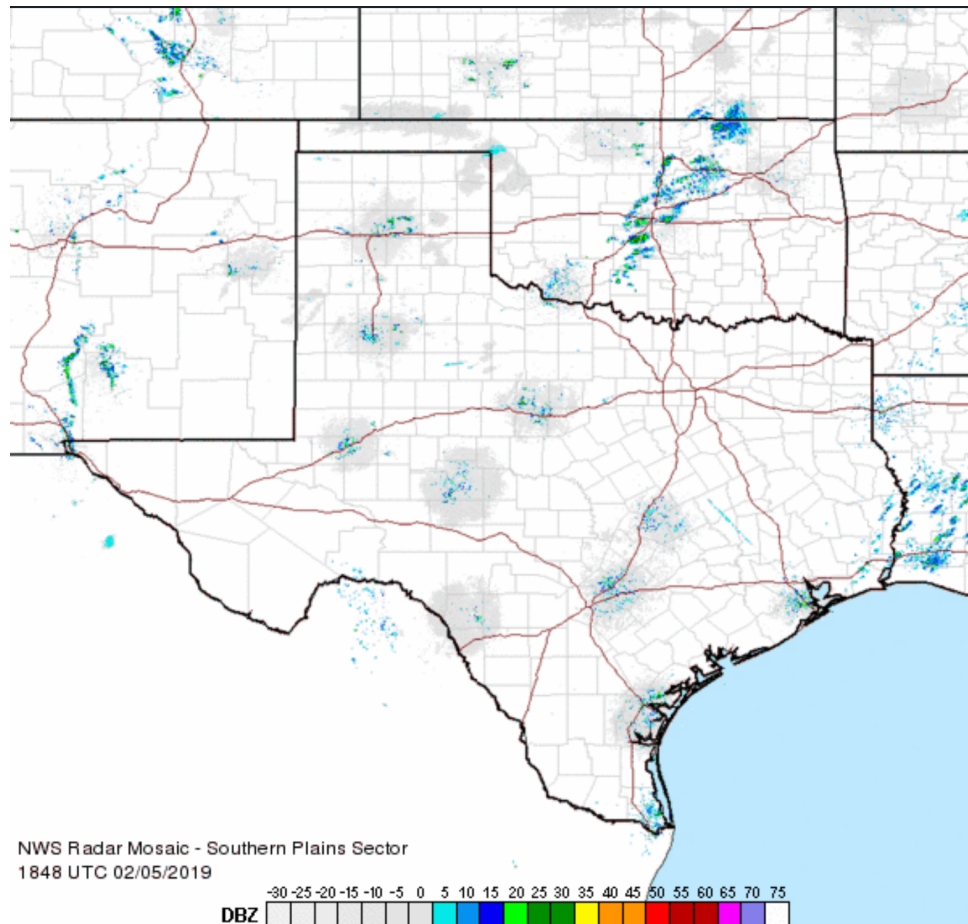


Figure 3.8: Example of an Animated Map [10].

3.6.3 Motion Charts

Motion charts [30] are interactive tools that show how data evolve over time. Bubbles (area marks) commonly use *position on common scales* (X, Y) to represent two quantitative variables, and *area* or *volume* (magnitude channels) to show an additional attribute, such as population. *Color hue* (an identity channel) often distinguishes categories, and *motion* shows temporal progression.

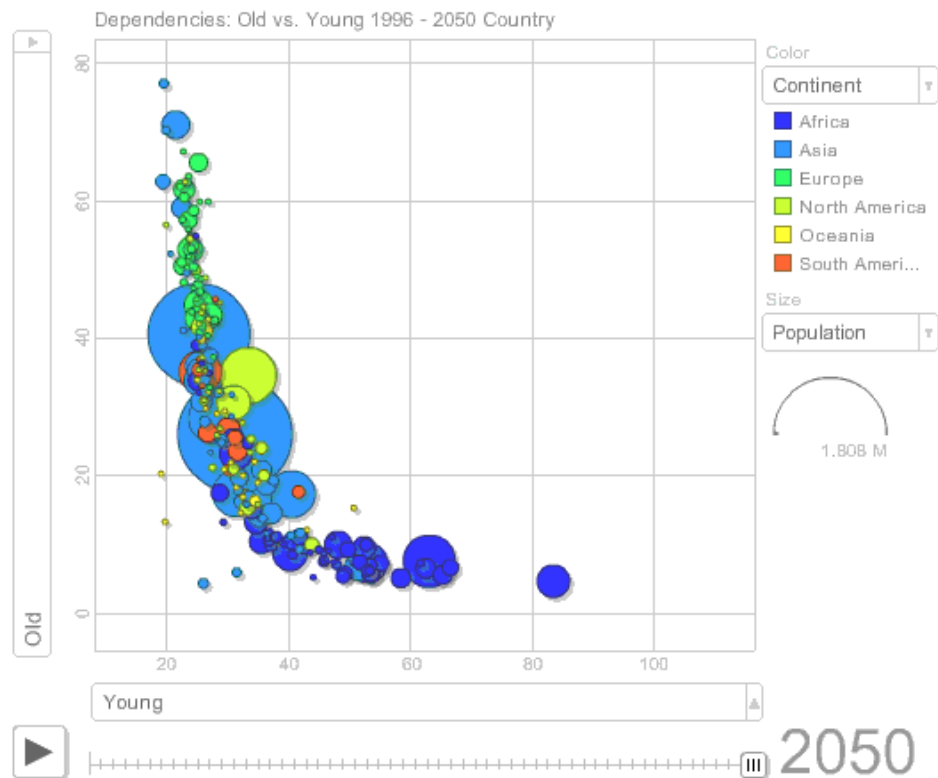


Figure 3.9: Example of a Motion Chart [11].

3.6.4 Space-Time Paths

Space-time paths track the movement or trajectory of an entity [12]. They typically employ *lines* (a mark type) to indicate the route taken. Position on the X and Y axes denotes spatial location, while the temporal dimension can be represented by either the path's vertical (Z) extension in a 3D scene (using *depth* as a magnitude channel) or by *color hue* variations along the path to distinguish different time segments.

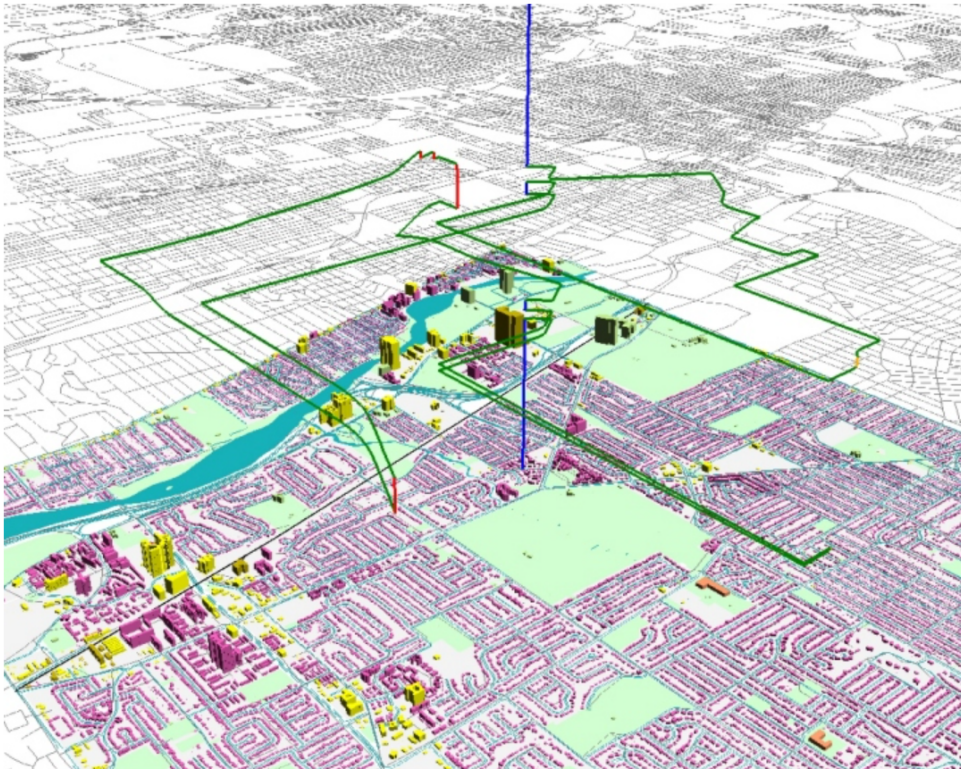


Figure 3.10: Example of a Space-Time Path [12].

Chapter 4

D3.js

D3.js, short for Data-Driven Documents, is a powerful JavaScript library used to create data visualizations [31]. It helps turn raw data into clear and interactive visuals like charts, graphs, and maps. Unlike many tools that come with pre-designed templates, D3.js gives developers complete control over how their visualizations look and behave, making it very flexible.

At its core, D3.js works by linking data to HTML, SVG, or Canvas elements on a webpage. Developers can use it to create visual elements like bars, lines, or shapes and make them respond to changes in the data. The library also provides many useful tools to build scales, axes, and layouts, which can be customized to match the needs of any project.

D3.js is especially helpful for creating a wide range of visuals. From basic bar and line charts to more complex layouts like tree diagrams and maps, D3.js supports many types of visualizations. For maps and spatiotemporal data, it works with formats like GeoJSON and TopoJSON.

Another key feature of D3.js is its ability to add interactivity to visualizations. Developers can include tooltips, animation, hover effects, zooming and panning, making it easier for users to explore and understand the data. This makes D3.js a great choice for projects that need both powerful visuals and user-friendly interactivity.

4.1 Data Binding in D3.js

Data binding in D3.js links your data (often an array) to DOM elements, enabling dynamic creation and updating of elements as your data changes. This is achieved using the `.data()` method. When you bind data to a selection, D3.js categorizes these DOM elements into different groups (e.g., `enter`, `update`, and `exit`), depending on how many elements are currently in the DOM versus how many data points you have.

4.1.1 Example: Binding Data to Circle Elements

Consider this example, which binds an array of numbers to SVG `<circle>` elements:

```
// Dataset
const data = [10, 20, 30];

// Select the SVG container and bind data
d3.select('svg')
  .selectAll('circle')
  .data(data) // Bind the dataset
  .enter() // Handle new data points
  .append('circle') // Append new circles
  .attr('r', d => d) // Set radius from data
  .attr('cx', (d, i) => (i + 1) * 50) // Calculate x-
    position
  .attr('cy', 50) // Set fixed y-position
  .attr('fill', 'blue'); // Apply initial style
```

Here:

- `.data(data)` binds the array `data` to the selection of `circle` elements.
- `.enter()` represents the data points for which there are no existing DOM elements (in this case, circles) and thus creates them.
- `.append('circle')` adds new `<circle>` elements to the DOM.
- Attributes like `r` and `cx` are set dynamically based on the data.

4.2 DOM Manipulation

Once the data is bound, D3.js provides straightforward methods to manipulate the DOM. Common methods include:

- `.attr()`: Sets or retrieves an element's attributes (e.g., size or position).
- `.style()`: Sets or retrieves an element's inline CSS styles.
- `.text()`: Sets or retrieves the text content of an element.

4.2.1 Example: Adding Text Labels

Suppose we want to add text labels beneath each circle to show its value:

```
// Add text labels to circles with styles
d3.select('svg')
  .selectAll('text')
  .data(data)
  .enter()
  .append('text')
  .attr('x', (d, i) => (i + 1) * 50) // Same x-position as
    the circles
  .attr('y', 100) // Place below circles
  .text(d => `Value: ${d}`) // Data label
  .attr('fill', 'black') // Text color
  .attr('text-anchor', 'middle') // Center alignment
  .style('font-family', 'Arial, sans-serif') // Font family
  .style('font-size', '14px') // Font size
  .style('font-weight', 'bold') // Font weight
  .style('opacity', 0.8); // Slight transparency
```

Here:

- `.attr(x)` and `.attr(y)` position the `<text>` elements based on the data index.
- `.text(d => `Value: ${d}`)` displays the numeric data in a string format.
- `.attr()` and `.style()` modify styling attributes and inline CSS.

4.2.2 The General Update Pattern

The **General Update Pattern** efficiently handles changes in data. It categorizes DOM elements into three groups:

- **Enter:** For new data points without corresponding DOM elements.
- **Update:** For existing elements that need modification.
- **Exit:** For elements no longer needed.

4.2.3 Example of Enter, Update and Exit

Below is an example that demonstrates each step:

```
// Dataset
const data = [40, 60, 80];

// Select and bind data
const rects = d3.select('svg')
  .selectAll('rect')
  .data(data);

// Enter: Create rectangles for new data
rects.enter()
  .append('rect')
  .attr('x', (d, i) => i * 60) // Dynamic x-position
  .attr('y', 0)                // Fixed y-position
  .attr('width', 50)           // Fixed width
  .attr('height', d => d)       // Dynamic height
  .attr('fill', 'green');      // Initial color

// Update: Modify existing rectangles
rects
  .attr('height', d => d) // Update height
  .attr('fill', 'blue'); // Change color

// Exit: Remove rectangles for missing data
rects.exit().remove();
```

Here:

- **Enter** (`.enter()`): For each data point that does not correspond to an existing rectangle, a new `<rect>` is created and appended to the SVG.
- **Update**: Existing rectangles are updated with new attributes (like height and color).
- **Exit** (`.exit()`): Rectangles that no longer have matching data points are removed from the DOM.

4.3 Special DOM Manipulations in D3.js

Beyond the previously mentioned methods, D3.js provides a number of other methods for DOM manipulation [32]:

- **insert()**: Inserts a new element at a specified position among the current element's children.
- **classed()**: Adds, removes, or toggles CSS classes on an element.
- **html()**: Sets or retrieves HTML content within an element.

These methods, combined with D3's powerful data binding, create interactive data-driven visualizations that automatically respond to changes in the data.

4.4 Interactivity in D3.js

Interactivity in D3.js makes data visualizations more engaging by letting users explore visualizations in dynamic ways. Key features include tooltips that show extra information when you hover over parts of the visualizations, brushing to highlight specific areas, and zooming and panning to view data at different levels of detail. Transitions add smooth animations to show changes, making them easier to follow. Buttons and sliders let users adjust visualizations settings. These interactive features work through event listeners (like `mouseover` or `click`) that trigger actions, creating visualizations that are more user-friendly and informative [33].

4.5 Performance of D3.js

The performance of D3.js depends on the rendering method used [34]. D3.js provides three primary rendering methods: SVG, Canvas, and WebGL, each suited for different levels of data complexity and amounts. While SVG works well for small datasets, Canvas and WebGL are better for handling larger datasets due to their improved rendering efficiency.

4.5.1 SVG, Canvas and WebGL

SVG (Scalable Vector Graphics) is one of the most commonly used rendering methods in D3.js, especially for small datasets. It works well with up to 1,000 data points because each visual element, such as a line or point, is represented as an individual DOM element. This structure makes SVG easy to style and interact with CSS and JavaScript. However, as the dataset grows, SVG's performance decreases significantly due to a large number of DOM elements. This limitation makes it unsuitable for rendering larger datasets [34].

Canvas, introduced in D3 v4, provides better performance for medium-sized datasets, handling up to about 10,000 data points efficiently. Unlike SVG, Canvas draws all visual elements on a single canvas element instead of creating individual DOM nodes for each one. This approach reduces overhead and significantly improves rendering speed. However, since Canvas lacks DOM elements for individual data points, adding interactivity and styling requires more manual coding, such as detecting mouse positions and recalculating visual effects dynamically. Despite these challenges, Canvas is a strong alternative when working with datasets that are too large for SVG but not large enough to require WebGL [34].

When working with very large datasets, WebGL is the best choice for rendering in D3.js. Unlike SVG and Canvas, WebGL uses the GPU (Graphics Processing Unit) to handle millions of data points efficiently. This makes it perfect for high-performance visualizations. WebGL processes data points directly on the GPU, avoiding the slower methods of updating the DOM or relying on the CPU. However, WebGL can be harder to use because it

requires knowledge of graphics programming and shaders. To make it easier, tools like D3FC provide ready-to-use WebGL components that work well with D3.js, allowing developers to use GPU power without writing complex code. WebGL is ideal for projects that need smooth interaction and fast rendering of very large datasets [34].

4.6 Discussion of D3.js Components in the Aviation Data Visualization Application

This application uses key D3.js techniques to create a synchronized map, chart, and table visualization.

Data Binding: The application employs `d3.selectAll(...).data(...)` to link raw data to visual elements, ensuring updates when the dataset changes.

Scales and Axes: It utilizes `d3.scaleTime()` for temporal data and `d3.axisBottom` or `d3.axisLeft` for labeled axes. `d3.scaleSequential()` is used for color coding to visually differentiate data points.

Interactive Filtering: The `d3.brushX()` tool enables users to select a range in line charts, triggering synchronized updates across the map, charts, and table. Animations using the `transition()` method make updates smooth and intuitive.

Geospatial Data Handling: The application uses `d3.geoMercator()` and `d3.geoPath()` to map geographic data. Background tiles are loaded with `d3.tile()`, and data points on the map are styled and filtered based on user interactions.

User Interactions: Event listeners like `.on("mouseover", ...)` and `.on("click", ...)` handle interactivity, enabling features like highlighting and tooltips.

The visualizations are rendered using **SVG** (Scalable Vector Graphics), which is ideal for this application due to its manageable dataset size and support for interactive features like brushing and highlighting.

Chapter 5

User-Centered Design

User-Centered Design (UCD) is a way of creating products that puts the people who use them at the center of the process. It follows ideas from standards like ISO 13407, which focus on how easy, efficient, and satisfying a product is for its users. To achieve this, UCD involves users throughout design and development. Designers ask for their feedback, test different versions, and improve the product based on what they learn.

By understanding user needs early and improving the design step by step, UCD helps avoid big, costly changes later on. This approach also helps catch and fix problems before they become harder to solve. People from different areas, such as design, engineering, and research, work together in UCD. This teamwork ensures the final product meets the needs of its users while also supporting the goals of the organization. [35]

5.1 Key Activities in User-Centered Design

In UCD, the main activities start with learning about the users — who they are, what they need to do, and where they will use the product. With this information, designers set clear goals, decide what is most important, and follow any rules or safety guidelines. As the project moves forward, these goals are checked and updated, since user needs can change over time.

Prototyping is an important part of UCD. Designers make simple models or early versions of the product to show ideas and get quick feedback. These can be anything from rough sketches to more detailed, interactive samples. By testing these prototypes early, designers can find and fix problems before too much time and money are spent. Over time, testing becomes more realistic, making sure the final product works well in real-life situations.

Keeping usability in mind at every step is essential. Designers and team members keep testing, documenting what changes are made, and staying in touch with everyone involved. This focus on users' needs, along with setting goals, building prototypes, testing, and improving, helps UCD produce products that users find useful, easy to use, and satisfying, while also meeting the organization's aims. [36]

- **Benefits and Limitations:** Points out the method's strengths and weaknesses.
- **Requirements:** States the time, skills, and resources needed.
- **Process:** Gives step-by-step instructions for how to prepare, run, and finish the activity.
- **Quality Assurance:** Describes how to judge the success of the method and avoid common mistakes.
- **Further Reading:** Offers more references for learning about the method.

■ 5.3 Selecting Appropriate UCD Methods

Choosing the right usability methods is key to getting good UCD results. Different stages in a project need different methods. Factors like the team's goals, the time and money available, and what kind of user feedback is needed all guide the choice of methods [38].

■ 5.3.1 Selection Criteria

Several factors influence which methods to pick:

- **Stage in the Lifecycle:** Early design might need quick tests like sketches, while later stages might need more detailed user testing.
- **Type of Information Required:** Depending on whether the team wants general opinions, exact numbers, or detailed user feedback, different methods are chosen.
- **Resources Required:** Time, budget, and the skills of team members influence which methods are possible.
- **Inherent Benefits and Limitations:** Each method works well for some tasks and not as well for others.
- **Cost-Benefit Analysis:** Methods that offer the most improvement for the least effort are often the best choice.

■ 5.3.2 Using UCD in Our Application

To ensure our visualization tool effectively meets user needs, we relied on specific User-Centered Design (UCD) techniques:

- **Early Feedback Sessions:** We presented simple middle-fidelity prototypes to potential users and noted their immediate reactions. Their feedback on basic layout and controls guided our initial changes before the tool became too complex.

- **Iterative Prototyping:** After each feedback round, we refined the design. New versions were tested again, allowing us to quickly fix issues—like making date filters clearer—and add features users said they needed, such as point highlighting on the map. More about this iterative desing process can be read in 6.3.2
- **Task-Based Usability Tests:** Once the tool was more complete, we gave users specific tasks (e.g., finding clusters of events). By watching how they performed these tasks, and encouraging them to think aloud, we could see where they got stuck, what worked well, and how to improve the interface further. You can read more about the testing in chapter 9.

This UCD approach helped us stay focused on real user goals, adjust design decisions based on actual feedback, and reduce wasted effort on unnecessary features. As a result, the final application is more intuitive.

Chapter 6

Application Development

This chapter explains how the airport data visualization application was developed, starting with its goals and moving through the steps taken to design and improve it. It covers the purpose of the application followed by an overview of the prototyping process. The chapter also looks at how a prototype for the application was created and why some earlier prototypes were skipped, and how the application improved over time.

6.1 Objectives of the Application

The main goal of the thesis was to create an application to help airport staff easily understand safety data by combining time and location information. It allows them to quickly spot problem areas, identify patterns, and improve safety. Designed for easy safety checks, the app makes it easy to manage large amounts of data. Its modular design ensures different parts can work together while being easily replaced or updated as needed. The app is also scalable compatible with various web platforms, making it easy for future improvements.

6.2 Prototyping process

Prototyping is the process of creating simple versions of a product or system to test and improve ideas before making the final version. These models, which can be physical or digital, are used to check how the product looks, works, or feels to use. Prototyping is important because it helps catch problems early, saving time and money. It allows teams to share ideas clearly, work together effectively, and make sure the design meets user needs. By testing with users, teams can gather feedback to improve the product's functionality, usability, and appearance, reducing the chances of costly mistakes or missed opportunities.

6.2.1 Overview of Prototyping Approaches

There are multiple approaches in prototyping. Two of them are explained in upcoming sections.

Throwaway/Rapid Prototyping

Throwaway or Rapid Prototyping [39] is a method where a simple model is quickly created to test specific ideas or features. This approach is helpful at the start of a project when the design or requirements are not fully clear. The goal is to gather feedback and make sure the concept works before moving forward. Once the prototype has served its purpose, it is discarded, and the feedback is used to improve the final design. For example, a team might create a quick sketch or basic screen design of a mobile app to see how the layout looks and get feedback from users.

Evolutionary Prototyping

Evolutionary Prototyping [39] is a method where the prototype starts as a simple version of the product and is improved step by step until it becomes the final version. This approach works well when the requirements are unclear or likely to change. Instead of throwing the prototype away, the team keeps improving it by adding features and making adjustments based on feedback. For example, a web app might begin as a very basic version and then grow into a complete product through several updates.

6.2.2 Fidelity in Prototyping

Fidelity in prototyping refers to how closely a prototype looks and works like the final product.

Low-Fidelity

Low-fidelity prototypes [40] are simple and basic models, like sketches, wireframes, or paper mockups. They are great for the early stages of design when teams are brainstorming and testing general ideas. These prototypes are quick and cheap to create, making it easy to try out different ideas and make changes based on feedback. For example, sketching a basic design of a mobile app or making a paper model of a website are low-fidelity prototypes.

High-Fidelity

High-fidelity prototypes [40] are detailed and interactive models that feel much more like the final product. These are usually created with advanced tools and are used in the later stages of design to test usability and refine the details. They allow users to get a realistic sense of how the final product will work. For instance, a clickable website prototype or an interactive app

design with animations can be considered high-fidelity prototypes. Both types of prototypes are important—low-fidelity ones are great for quick tests in the beginning, while high-fidelity prototypes are needed for testing the final design’s functionality and user experience.

6.3 Prototype for Aviation Data Visualization Application

Developing the Aviation data visualization application required a special approach to prototyping due to the project's complexity. Below is an explanation of the design process.

6.3.1 Overview and Initial Design Process

In many projects, low-fidelity prototypes are used as a method to explore ideas and gather feedback. However, for this project, we skipped this stage and went directly to middle/high-fidelity prototypes. This decision was based on several factors:

- **Functionality Complexity:** The desired functionalities, such as dynamically filtering points on a map and displaying graphs that react to selected filters, were difficult to replicate using paper prototypes or static low-fidelity methods.
- **Efficiency in Development:** Transitioning directly to a functional middle-fidelity prototype allowed us to rapidly test features, rather than spending time on static prototypes that wouldn't adequately represent the application's interactive nature.

While low-fidelity prototyping was skipped, some initial paper sketches were created to visualize the application's layout. These sketches, though limited in detail, served as a starting point for discussion between me and my supervisor, Ing. Ladislav Čmolík, Ph.D. These drafts helped with our vision about the application before moving into coding.

The decision to skip detailed low-fidelity prototyping proved effective, as it enabled us to focus on developing and iterating on functional prototypes, ultimately leading to a more robust application design.

6.3.2 Iterative Development Process

Consultation and Feedback: The project progressed in iterative cycles, with regular meetings. These regular meetings allowed us to:

- Review progress and identify areas for improvement.
- Incorporate feedback from both the supervisor and other members of the development team.

Incorporating User Testing: User testing was used in the development process. Testing was conducted in parallel with development, providing feedback about user needs and potential issues. These findings were:

- Analyzed to identify areas that needed an update.
- Implemented immediately into the ongoing development to ensure the application met user expectations.

6.3.3 Evolution of the Application

In the development process, the application evolved significantly, moving from a simple middle-fidelity prototype to an advanced high-fidelity product.

Version 1: Initial Prototype

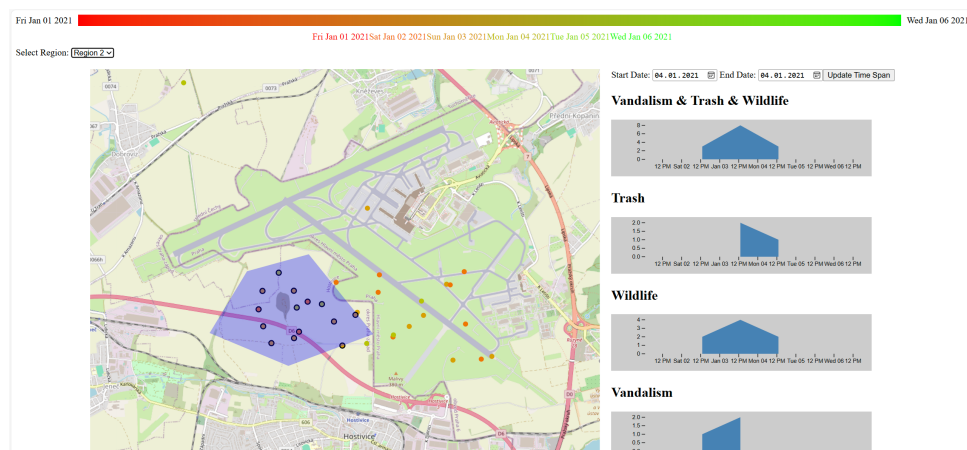


Figure 6.1: First prototype.

This version was a simple prototype to test the main ideas for the application. The user interface was very basic and not well-designed. The focus was on testing the map with points and allowing users to select them. The graphs could be zoomed in, but zooming did not filter the points on the map. The only interaction between map and graphs was that highlighting points on the map updated the graphs. There was a basic filter for selecting a time range, and points were colored based on time. The colors for certain time were visible on the top of the application. Users could hover over points to see a tooltip with details about the event type.

Version 2

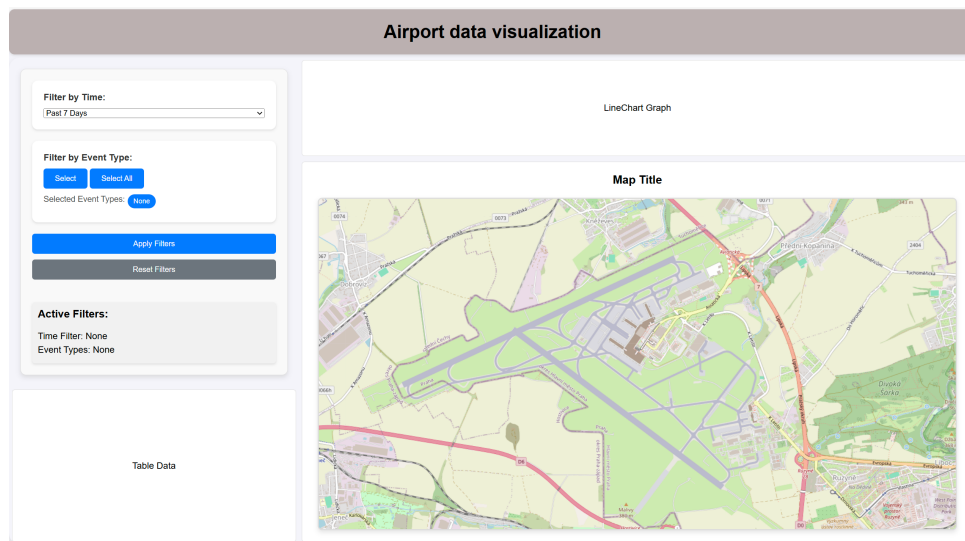


Figure 6.2: Second prototype.

The first prototype showed that features like the point map and line chart were important for the application. However, it also made clear that the design needed to change. We started fresh with this second prototype, which introduced a new layout featuring four main components: filters, data table, graph, and map. This layout proved to be user-friendly and became the foundation for future designs.

Version 3

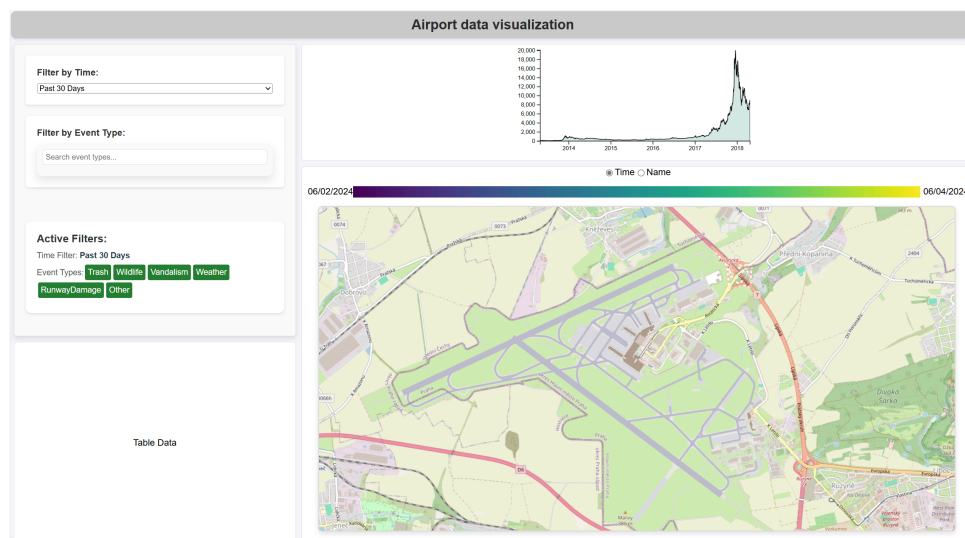


Figure 6.3: Third prototype.

This version added a line chart placeholder to the graph component, along with a color scale similar to the one in the first prototype. These

changes improved the application's ability to show time-based data in a clear and consistent way.

■ Fourth prototype

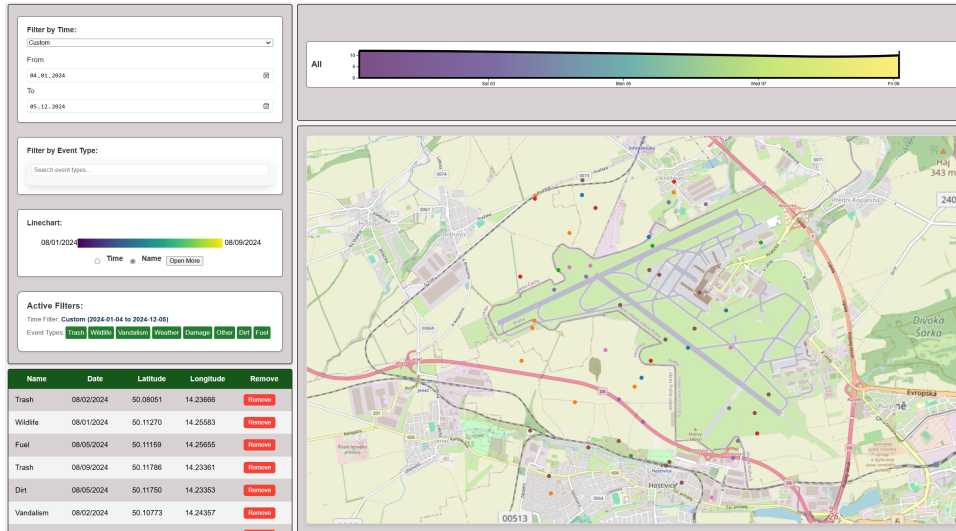


Figure 6.4: Fourth prototype.

In this version, the application started to feel more functional. Generated data was added, and users could interact with it in several ways. They could filter the data, highlight points on the map, highlight or zoom on the line chart graph and also table component was created for the data. This version made significant progress toward a usable tool.

■ Fifth prototype

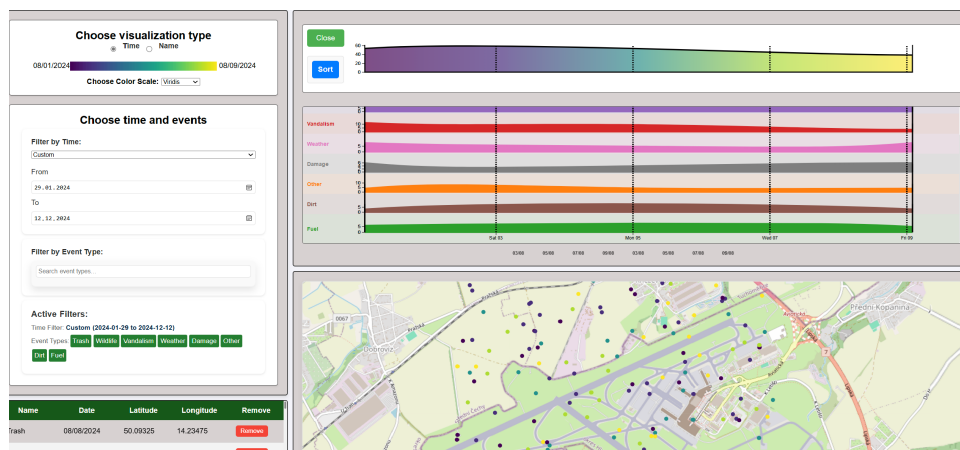


Figure 6.5: Fifth prototype.

This version improved the layout further, particularly within the filters component. Sub-charts were added for individual event types, making it

easier for users to analyze specific data sets. These changes added depth to the application's functionality.

■ Sixth prototype: Final version

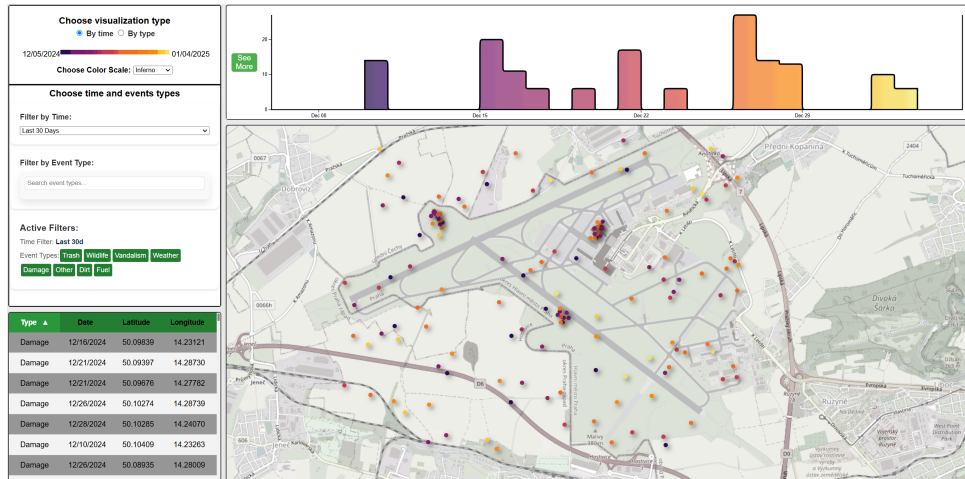


Figure 6.6: Final version.

This is the final version of the application. Previous versions struggled with the complexity of highlighting and zooming on the line chart at the same time. Based on user feedback, the line chart was changed into a bin graph. Zooming was removed, and brushing was used only for highlighting. This version focused on user-friendliness and attention to detail, including better colors, shades and better usability. It represents the final high-fidelity prototype.

Chapter 7

Application Implementation Approach

7.1 Overview

The application's code is split into clear parts, each handling a specific task.

The complete source code for this application is available on GitHub: <https://github.com/jirikdan/AviationDataVisualization>.

7.1.1 Project Structure

Below is a short overview of the main folders and files, describing what each does:

- **index.html**
Loads the scripts and CSS files, and sets up the main layout of the application.
- **css/** (folder)
 - **brush.css**
Manages styles for the D3 brush selection on charts (the semi-transparent selection rectangle, its handles, and overlays). It also takes care of pointer events to ensure the brushing and highlighting interact properly in various charts.
 - **colors.css**
Defines color-based themes across the application. This includes page background colors, alternate row colors in the table, highlighting styles for selected or hovered items, and special effects like drop shadows on map points or linechart bins.
 - **filters.css**
Styles the filtering panel, including the time and event-type filters. It handles scrollbar visibility, container layouts for filter sections, various button groups, and the dropdown interface for selecting multiple events at once.

- **linechart.css**

Contains specialized styling for main and sub line charts, such as hiding tick marks, customizing scrollbars for the “see more” section, and formatting the chart-area. Also styles labels, axes, and selection rectangles within the charts.
- **mapStyle.css**

Handles the appearance of the map container and any map elements (SVGs, tooltips, color-scale legends). It provides hover effects for the map, sets dimensions, and manages the layout for the color scale or legend elements.
- **styles.css**

Applies general layout rules for the entire page, including resets, flex containers, column layout, row distribution, and container sizing. It makes sure each major section (filters, tables, charts, map) fits neatly in the overall UI.
- **table.css**

Takes care about the table's appearance (scrolling, headers, sticky row positions) as well as the row and cell styling. It ensures consistent spacing, aligns text, and handles how columns are highlighted or sorted.
- **toggleButton.css**

Manages the look and behavior of the “See More” toggle button, including hover effects. It also defines the style for the event-selection area (checkbox containers, dragging and sorting controls) that appears when the toggle button is clicked.
- **js/** (folder)
 - **constants/** (subfolder)
 - **constants.js**

Defines global constants and utility values for the application. This includes the base map tile URL function (`url`), an initial date range (`dateSpan`), initial data, and color mappings for event names. It also initializes a few global variables (e.g. `lineChartWidth`, `loadedTileData`) needed throughout the code.
 - **dataFiltering/** (subfolder)
 - **dataFiltering.js**

Manages filtering logic and user interactions for date ranges and event selections. When users pick a new time filter or select/deselect event types, this file updates `selected` and `highlighted` states in the data, then triggers the charts and map to redraw with the new filters. It also includes helper functions for handling dropdowns, search bars, and custom time inputs.

- **dataHandling/** (subfolder)

- **data.js**
Defines a class that manages the global dataset, handling operations like counting selected/highlighted events, filling missing date gaps, and retrieving sub-sets of data for charts. It creates methods to get `SelectedEventCounts`, `HighlightedEventCounts`, or `EventTypeData`, so the visualizations can easily update.
- **dataArray.js**
Generates and stores the pseudo-random aviation-event data. It creates **Feature** objects (geoJSON-like) with random dates and coordinates, optionally clustering data in “hotspots,” using a seeded random for reproducibility.

- **linechart/** (subfolder)

- **linechart.js**
Implements the main line chart, including D3 brushing, time-based color gradients, and chart updates when filters or highlights change. It also propagates brush selections to smaller charts if needed.
- **seeMoreLinecharts.js**
Responsible for generating multiple sub-charts (one per event type), plus sorting and toggling them. It manages user interactions (like clicking the “See More” button) and updates chart layouts or orders based on event popularity or maximum Y-values.
- **subLineChart.js**
Handles drawing smaller, horizontally stacked line charts for individual event types. Uses a simplified version of the main chart’s logic, including its own brush, gridlines, and gradient area to visualize each event type’s distribution over time.

- **map/** (subfolder)

- **map.js**
Creates a zoomable map using D3’s projection and tile-based fetching. It updates the color of each location point by time or event name, and monitors user clicks for highlighting or showing tooltips. It also ties into the main chart with cross-selections.
- **selection.js**
Allows rectangle-based selection on the map by right-click and drag. Points within the selection rectangle become highlighted or unhighlighted, and the code then triggers chart/table updates for consistent filtering across the application.

- **table/** (subfolder)

- **table.js**
Builds and maintains the event data table (filtering rows to

only selected items). It supports column sorting and row highlighting, automatically synchronizing these visual changes with the map and charts.

7.2 Filtering and Data State

The filtering code updates the data points dynamically. Each point has `selected` (filtered data) and `highlighted` (selected filtered data), which change based on user actions. For instance, if a user picks a time range, selects event types, or draws a rectangle on the map, `dataFiltering.js` sets `selected` or `highlighted` accordingly for each point in the global `data` array.

Charts and maps then show only the filtered (selected/highlighted) points. They do not need to know how filtering works—they just re-render based on updated data.

7.3 Shared Domains and Reactive Updates

The main time-series chart and the smaller sub-bin charts share the global `dateSpan`. When filters modify `dateSpan` or the event data, the charts automatically adjust their axes and redraw. Similarly, `map.js` updates which points are shown based on their `selected` or `highlighted` properties.

This reactive design simplifies the code. The map and charts do not check conditions themselves. Instead, they use data binding of D3.js.

7.4 Independent Logic and Visual Layers

The filtering and data logic are separate from the drawing code. Files like `lineChart.js`, `subLineChart.js`, and `map.js` focus on rendering. Data files provide the processed dataset, while filtering ensures only the desired points appear.

This makes maintenance simpler. If the filtering changes, the charts and map see the new data automatically, without code changes on their side.

7.5 Extensibility

This setup scales easily. Adding new event types, filters, or time spans only requires updates to the data or filtering logic. The visualization code automatically adapts. Likewise, new components (like a different kind of chart) can be added by binding them to the filtered data, just as the existing elements do.

Chapter 8

Final Results

This chapter describes the final version of the application. It shows how different parts of the application—filters, graphs, tables, and maps—work together to form a single tool. The next sections explain its design and how it works.

8.1 Overview of the Application

The final application integrates time and location aviation data in a visual, interactive interface. When users open the application, they first see an overview of data filters, a dynamic map, a data graph and a data table that respond to user selections. Figure 8.1 provides a snapshot of the application’s main view, showing how these components are arranged together.

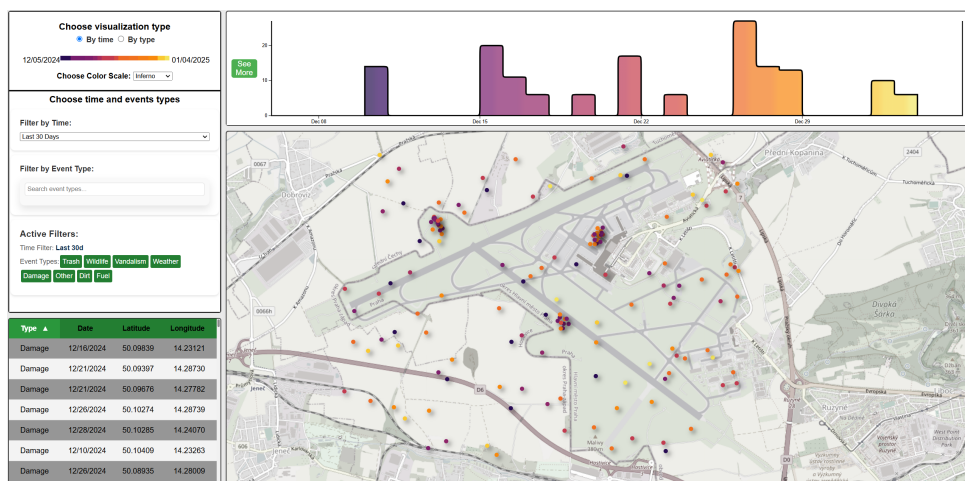


Figure 8.1: Final integrated view of the application with filters, graph, map, and table components.

8.2 Core Features

8.2.1 Filter Component

The filter component offers various tools to help users narrow down the data. They can choose specific time ranges and event types. As users adjust the filters, the visual elements update instantly to keep the displayed data clear and relevant.

Figure 8.2 shows an example of the filtering interface. Users can see their currently applied filters in the "active filters" section.

Choose time and events types

Filter by Time:

Last 30 Days

▼

Filter by Event Type:

Search event types...

Active Filters:

Time Filter: Last 30d

Event Types:

Trash

Wildlife

Vandalism

Weather

Damage

Other

Dirt

Fuel

Figure 8.2: Filter interface allowing users to narrow down the event dataset by time and type.

Users can adjust the filter's display settings to clearly view data by type or by time. Figures 8.3 and 8.4 show these options, offering different ways to analyze the same dataset.



Figure 8.3: Events visualized by type, allowing users to distinguish categories more easily.

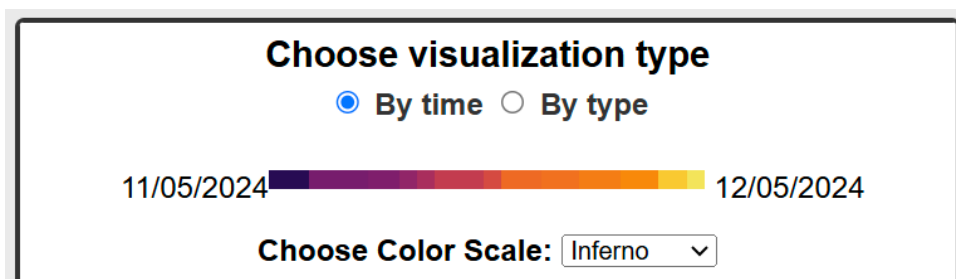


Figure 8.4: Events visualized by time, allowing users to distinguish data times more easily.

8.2.2 Data Graph

This component started as a line chart but was changed to a bin graph based on user feedback. The bin graph groups events into time bins, making it easier to see trends and patterns. On y axis is the number of all event occurrences and on X axis data are grouped by day. Figures 8.5 and 8.6 show the main graph and the smaller sub-graphs. The sub-graphs X and Y axis shows the same, however, only for the specific sub-graph event type of data.

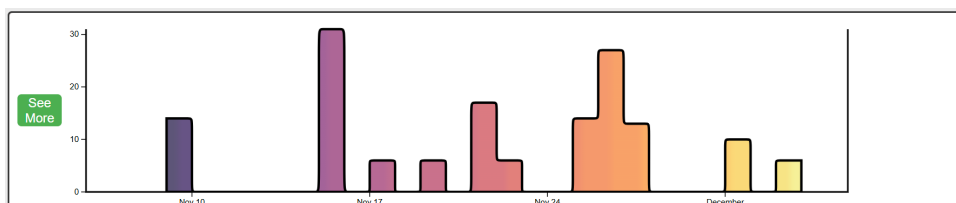


Figure 8.5: The initial graph layout before showing sub-graphs

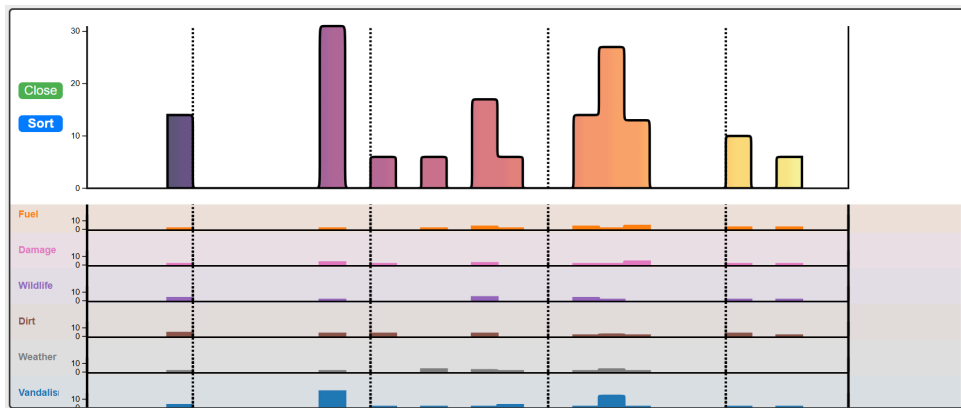


Figure 8.6: An unpacked bin graph representation, showing events grouped by time intervals.

Users can highlight portions of the main graph by brushing over it with the cursor. In Figure 8.7, a brush selection is used to highlight a specific time interval. At the same time this action highlights points on the map and in the data table. By removing the zoom functionality and using brushing to highlight data simplified the tool and improved its usability, as seen in Figure 8.7, where selected bins correlate with highlighted points on the map. Another example which also show highlighting propagation to sub-graphs can be seen here 8.8.

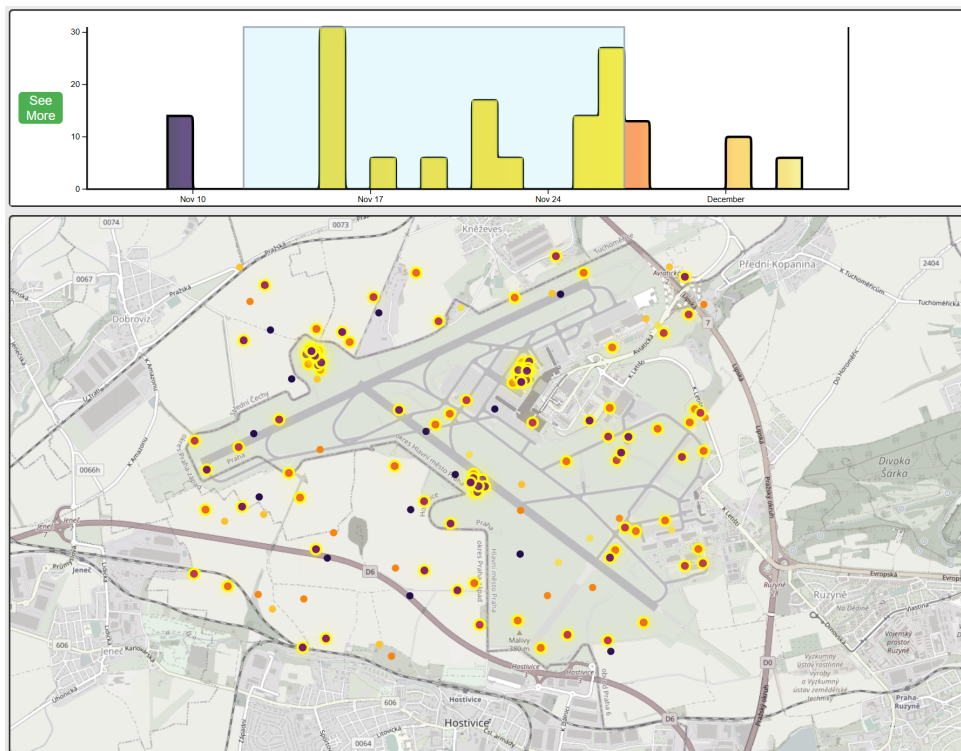


Figure 8.7: Highlighted section of the bin graph reflected on the map component, showcasing interactive linking.

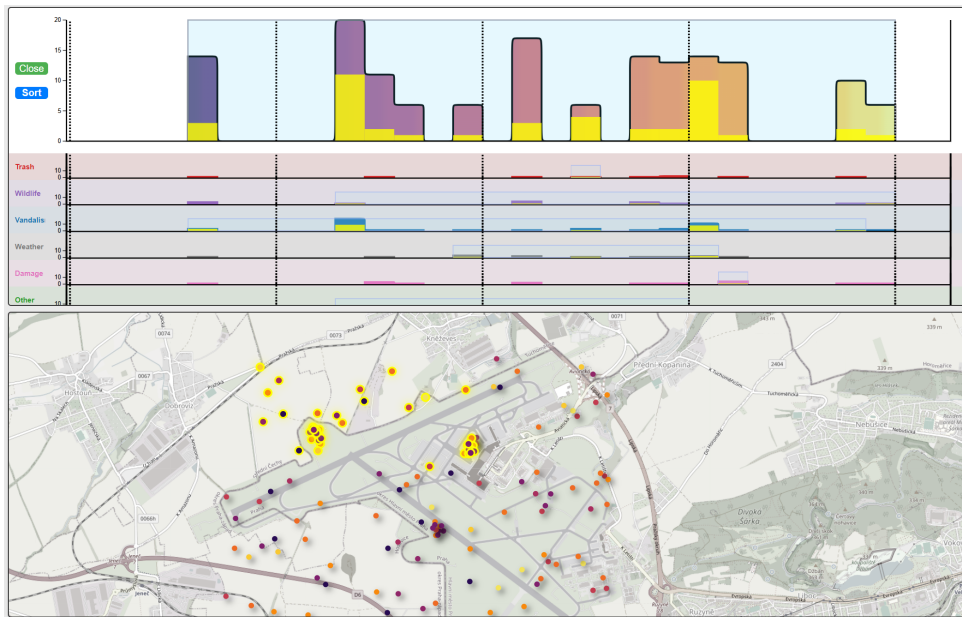


Figure 8.8: Highlighted selection of points in the map propagating to main graph and sub-graphs.

Next feature is the tooltips that show up when you hover over brushed selected bins, as shown in Figure 8.9. These tooltips provide quick details about the selected time range when hovering over graph boxes.

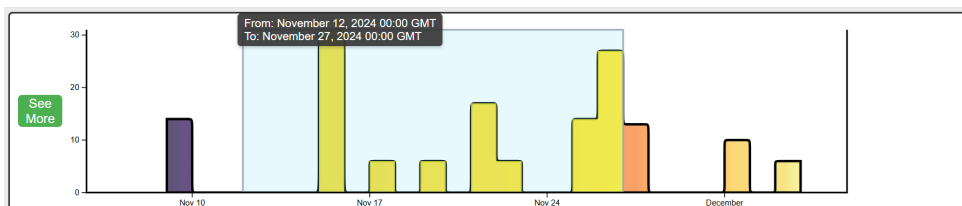


Figure 8.9: Tooltip details for a particular graph selection revealing selected time range

To help compare different groups of events, the graph includes sub-graph sorting features. Figures 8.10 and 8.11 shows how user can use this feature to uncover hidden patterns in data.

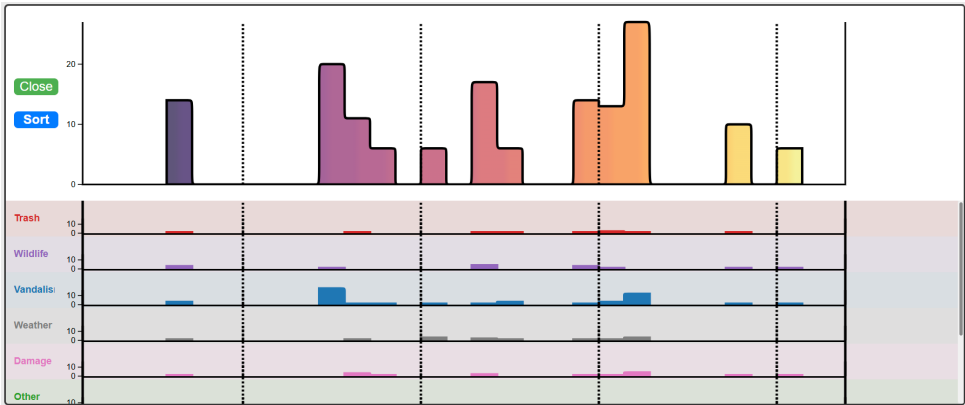


Figure 8.10: Sub-graphs enable better analysis of different event types.

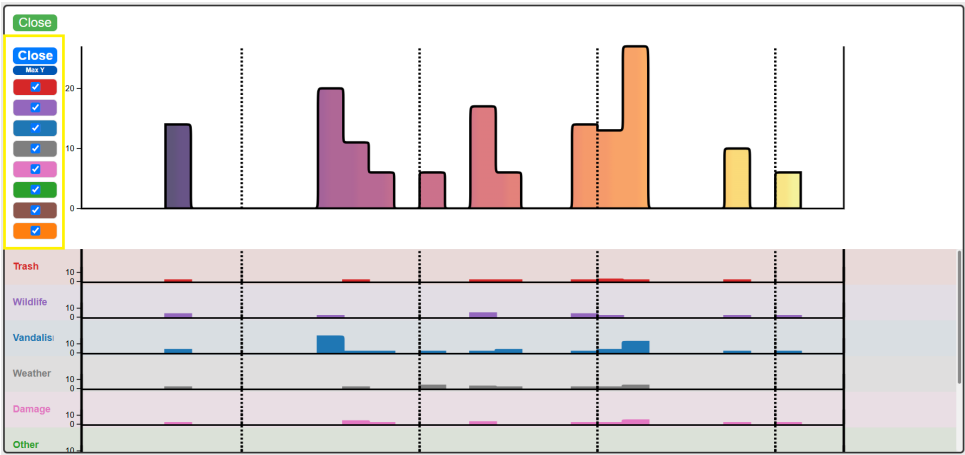


Figure 8.11: Sorting part is highlighted in yellow rectangle, users can sort using drag and drop

8.2.3 Data Table

The data table shows detailed information about the events that are currently selected. It works with the graphs and maps to provide specific details – event type, time and location. Figure 8.12 shows the table with one selected event. Clicking on an event in the table highlights its matching point on the map and the related bin in the graph, keeping everything connected. Users can also sort the table elements by clicking on the header elements.

Type ▲	Date	Latitude	Longitude
Damage	11/16/2024	50.09839	14.23121
Damage	11/21/2024	50.09397	14.28730
Damage	11/21/2024	50.09676	14.27782
Damage	11/25/2024	50.10274	14.28739
Damage	11/27/2024	50.10285	14.24070
Damage	11/09/2024	50.10409	14.23263
Damage	11/26/2024	50.08935	14.28009

Figure 8.12: A single highlighted event in the data table sorted by type, which also reflects on the map and graph.

8.2.4 Map Visualization

The map uses altered point map visualization and creates points at event locations directly on the map of the airport. As shown in Figure 8.13, colors are used to distinguish event types or times of the events (depends on selected visualization). Hovering over a point shows tooltip either as shown in Figures 8.14 and 8.15.

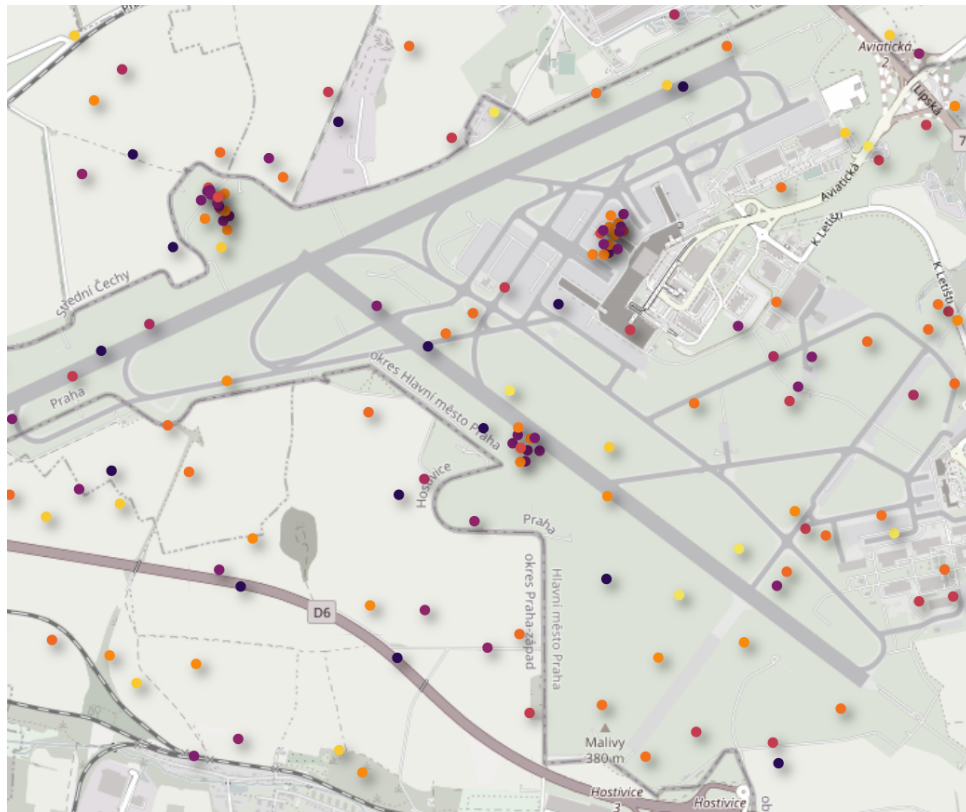


Figure 8.13: Map visualization with events plotted according to their actual airport location.

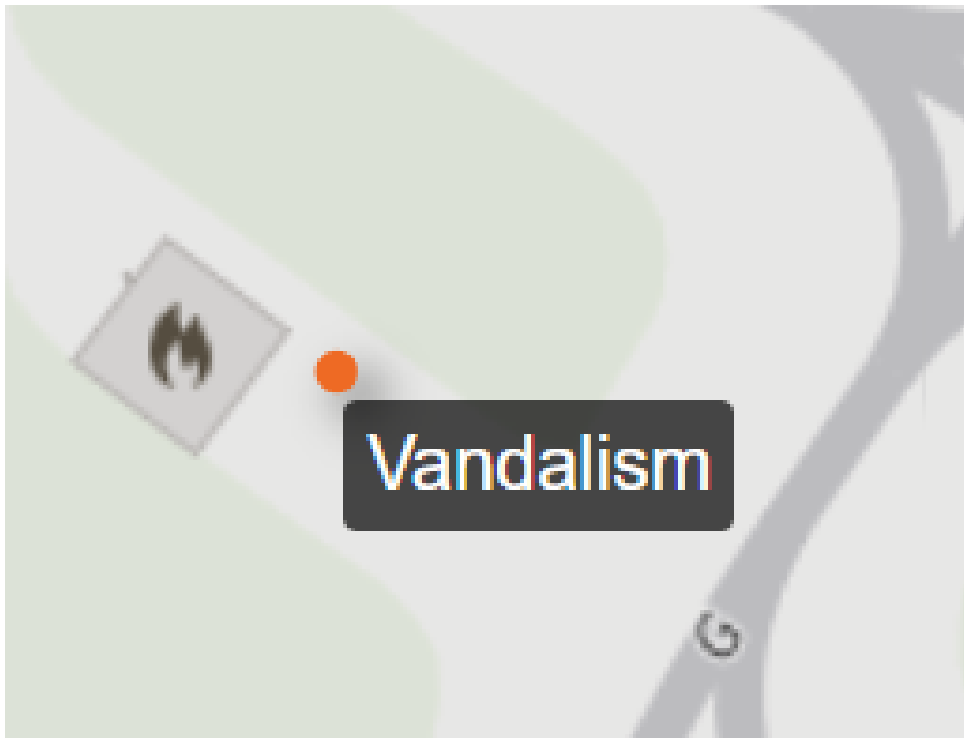


Figure 8.14: Type tooltip shown when hovered over a point. Visible for selected visualization: "By time".

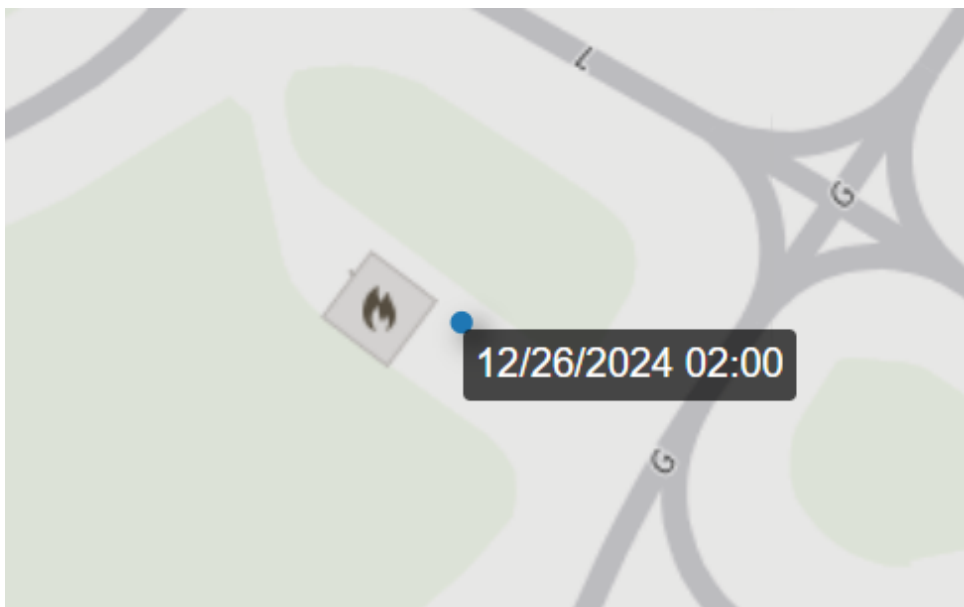


Figure 8.15: Time tooltip shown when hovered over a point. Visible for selected visualization: "By type".

Because the map is linked to other components, highlighting points on it, either by left clicking on separate points or by right click rectangular

selection, will highlight the corresponding points on the map. Figure 8.16 shows rectangular selection and Figure 8.17 shows highlighted points. When users want to unhighlight highlighted points they can either left click on them one by one (either on map or data table) or right click anywhere on the map to unhighlight all at the same time.

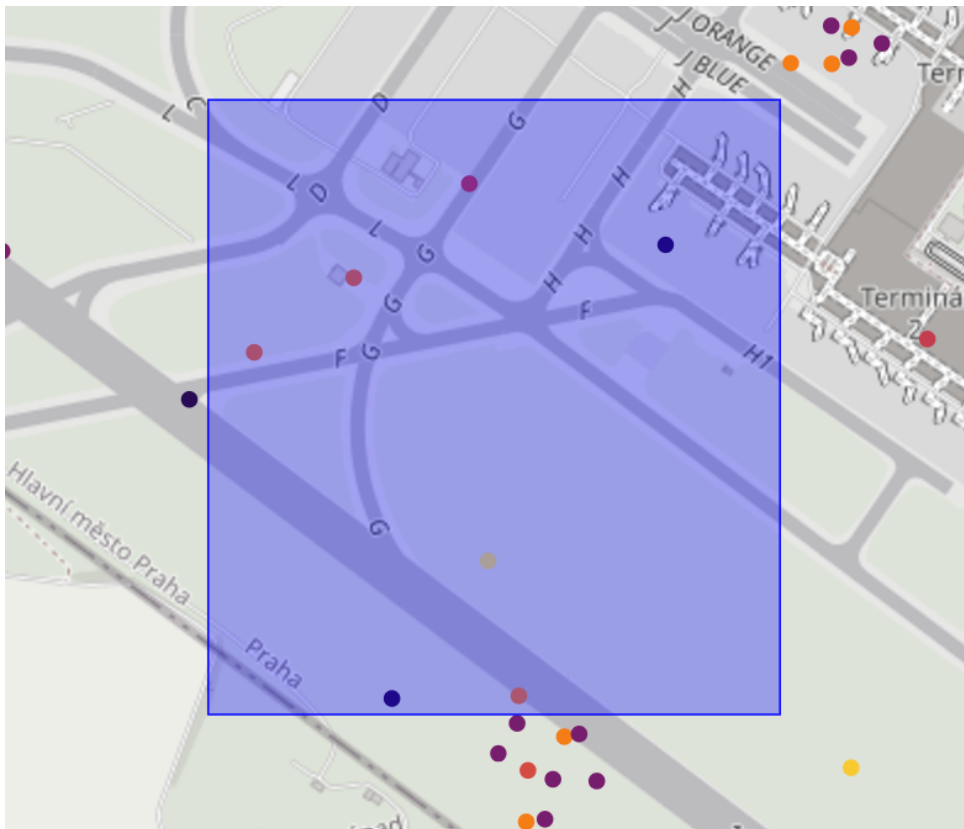


Figure 8.16: Using rectangular selection to highlight a specific data on the map.

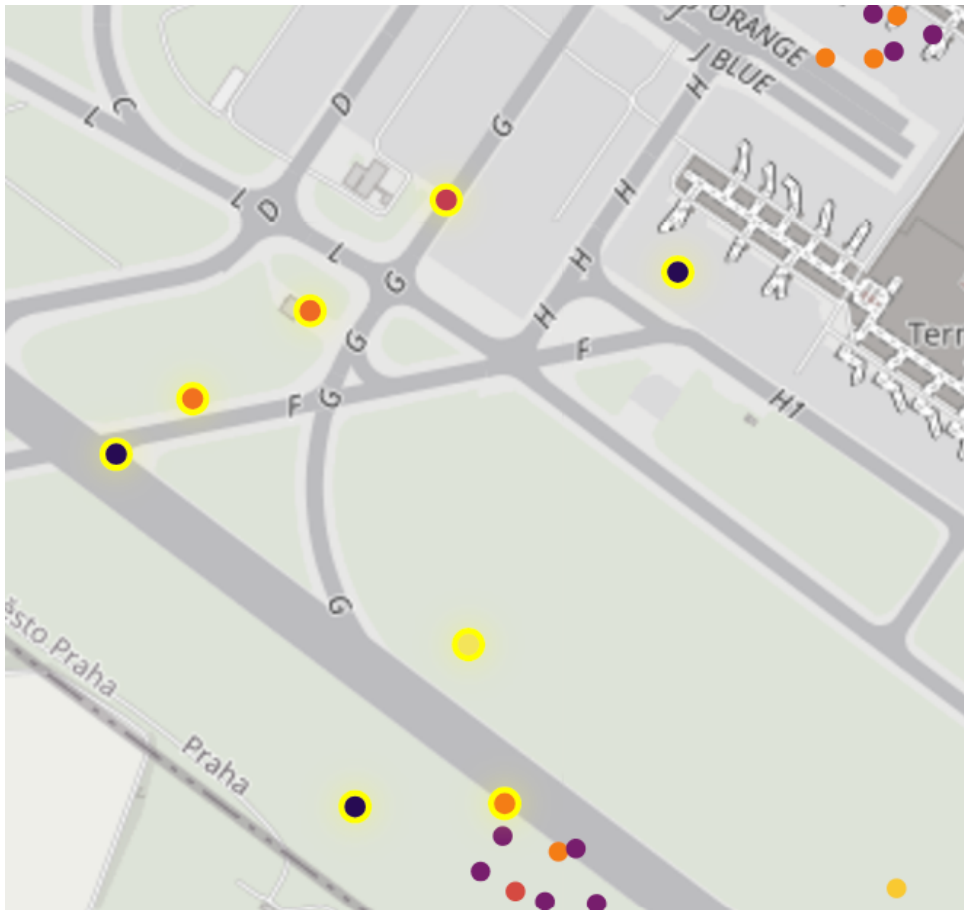


Figure 8.17: Highlighted points on the map.

When visualizing how events are distributed by type or time, the map can adapt to show these distinctions. Figures 8.18 and 8.19 illustrate how graph and map help users interpret patterns in either by time or type visualization.

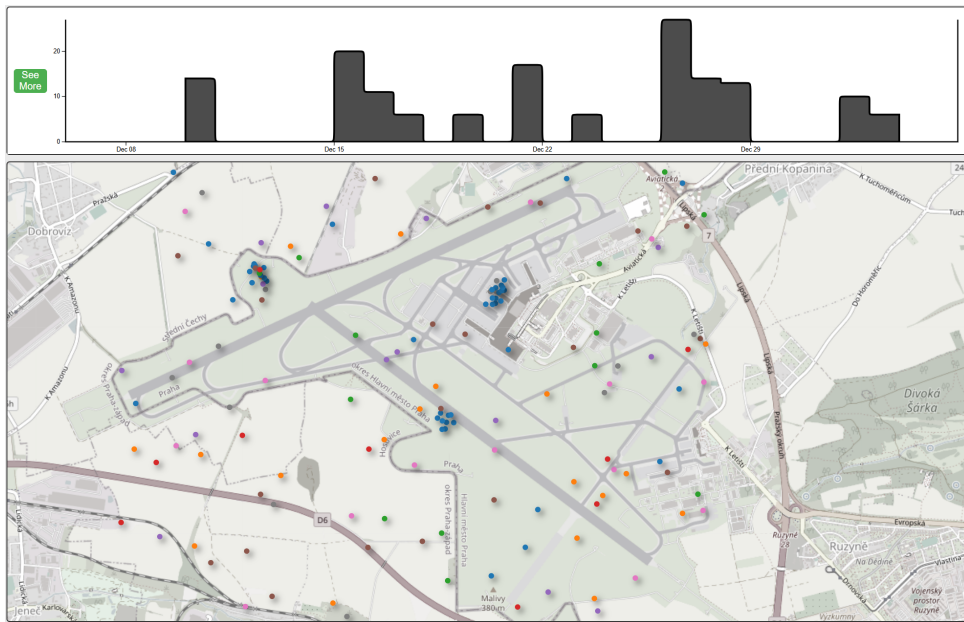


Figure 8.18: Type based visualization of the map and graph. Changes graph color to gray

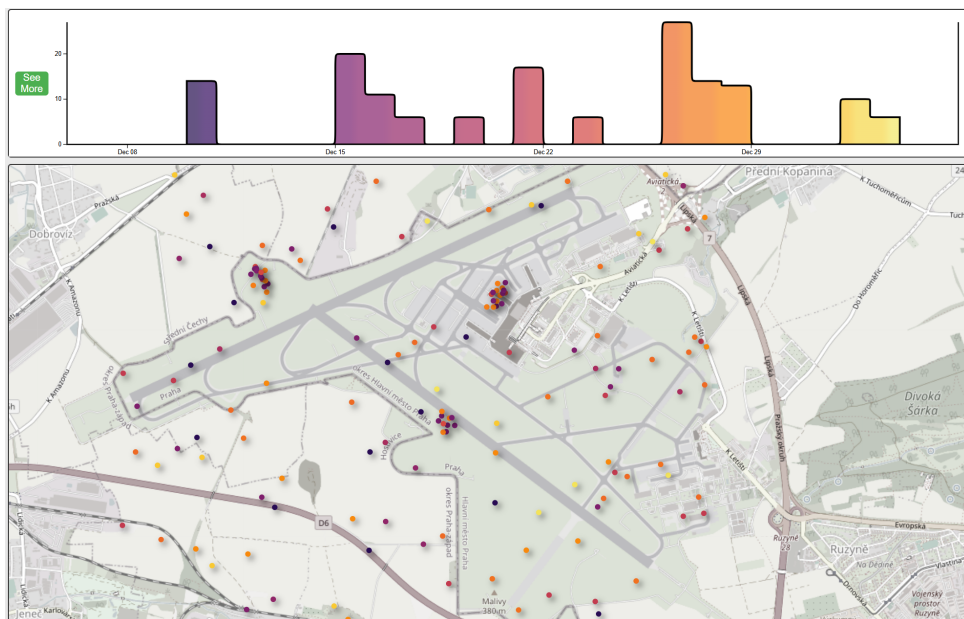


Figure 8.19: Time based visualization of the map and graph. Changes graph color to interpolated color representing time.

8.3 Performance Testing

Application was tested for performance both on Chrome and Edge browsers, the application was tried on a 13th Gen Intel(R) Core(TM) i7-13700HX laptop.

With up to around 150 data points, the application remained responsive; at 250 points, noticeable slowdowns began, and at around 500 points the interface was sluggish but still usable. Beyond 1000 points, performance approached the limits of SVG rendering, leading to stuttering.

On a desktop PC with similar specifications, overall performance was noticeably better. Edge also appeared to perform slightly faster than Chrome. In general, performance is sufficient for the expected dataset size (up to a few hundred data points). The biggest performance drop was caused by applying CSS shading through filters. However, the clarity these filter add outweighs the hardly noticeable performance drop

Chapter 9

User Testing

This chapter explains how the application was tested to see how easy and effective it is to use. It shows early informal tests that helped improve the design through changes and updates. It then describes the main testing phase, where seven participants were asked to complete specific tasks in different situations to evaluate the app in more detail. The results of these tests are shared to show how well the app worked. Feedback from participants is also reviewed.

9.1 Early Testing

Before formal testing began, the application went through several rounds of informal testing. These tests were done to find problems and improve the design based on user feedback. Over time, random participants tried the application in different ways, helping to identify what worked well and what needed fixing.

The testing revealed several issues:

- Users didn't find it easy to deselect active events.
- Sorting sub-charts mechanism was not clear.
- The color scale was not immediately clear what is its purpose.
- The line chart was hard to use, and users struggled to focus on important peaks.
- Filter highlights sometimes didn't work properly and were hard to distinguish from normal points.
- Greyed-out points still showed tooltips, which made no sense to users.
- Filter options, like resetting to a default order, were unclear or didn't work as expected.
- Text-based search did not work properly.

Each round of testing brought new updates to the application, which led to many improvements and fixes:

- A clear "unselect" button was added to make deselecting active events easier.
- The sorting mechanism for sub-charts was updated to be more intuitive with possibility of sorting by maximal values.
- The purpose of the color scale was clarified by adding labels and tooltips.
- Line charts were redesigned to bin charts to make peaks easier to identify and zooming was disabled.
- Highlighted points were redesigned to be clearly separable from normal points and worked reliably after filter changes.
- Greyed-out points were removed entirely.
- A reset sorting button was added to simplify filters.
- The text-based search feature was fixed.

By fixing these problems and improving the design with each round of feedback, the application became easier and more user-friendly. By the time formal testing began, many of the big issues had already been resolved.

9.2 Improved user testing

After making the improvements mentioned earlier, a detailed user testing scenario was created. Seven users were asked to pretend to be airport safety analysts reviewing and evaluating daily reports of incidents (such as vandalism, runway damage, or wildlife sightings).

The testing session took about 40 minutes on average. Although the time it took for individual participants differed significantly. Some took only about 20 minutes and some about an hour. The testing included several tasks that were trying to check specific parts of the application. Here are the tasks with their goals:

1. **Initial Impression:** Users were asked to look at the application and describe what they saw. They were asked to explain what the map, graph, table, and filter components represented.
Goal: Gain users' initial understanding of the application and how users interpret its various interface components.
2. **Wildlife Findings in the Last 7 Days:** Participants were instructed to filter events to show only wildlife (e.g., dead animals) from the last 7 days, identify how many were found, when they were discovered, and where they occurred.
Goal: Verify that users understand how to use filters by time and event type and can interpret details about events displayed on the map, graph, and table.

3. **Expanding the Timeframe:** Participants adjusted the timeframe to 30 days, restored all event types and then removed filter for wildlife.

Goal: Teach and users how to work with multiple filters and understand how they add or remove filters.

4. **Identifying and Analyzing Clusters:** Users looked for clusters of events (multiple incidents in the same location), explored which event types formed these clusters, and adjusted the visualization parameters to identify when the majority of clustered events were happening.

Goal: See how users analyze clusters of data in space and time using the map, graph, and table and what visualization parameters they change to analyze them.

5. **Custom Time Filter:** Users created a custom time filter for a specific 14-day window and examined the data.

Goal: Teach users to customize filters for specific needs and confirm their understanding of how filters function.

6. **Open Feedback:** Finally, users were asked to describe what they liked, what could be improved, and whether anything was confusing or missing.

Goal: Gather feedback on the application's usability and identify any confusing or missing elements to address potential issues in the future.

Participants were encouraged to “think aloud,” verbalizing their thought processes. Notes were taken on any difficulties, misunderstandings, or positive reactions.

9.3 Results and Discussion

This section discusses findings from testing the application with 7 participants of different ages, backgrounds, and computer skill levels. No features were explained beforehand so more complex features such as highlighting points on the map, selecting groups of points by right-clicking, or brushing the graph were tested for their intuitiveness.

9.3.1 Initial Impressions and Core Features

All participants quickly understood the basic purpose of each component (filters, table, map, and graph). The **Table** and **Filter** sections were particularly intuitive. Users easily scrolled through the table and used time-based filters (e.g., last 7 or 30 days) or set custom dates. Filtering by event type was also straightforward for everyone. The “Active Filters” area proved convenient for removing selected event types without reopening the event-type filter window.

The only aspect that caused hesitation in the filter component was the “Choose Visualization Type” option, which some users avoided due to uncertainty about its function.

9.3.2 Map Interactions

The **Map** component was well-received. Participants enjoyed the colors and found zooming and panning easy. Most users discovered how to highlight points by clicking on them, though only 2 out of 7 realized they could use right-clicking for “group selection” without being told. Removing highlighted points was also not intuitive until it was explained that a right-click on the map clears them. Despite these hidden features, the map became a primary tool for identifying event clusters, and after some time, participants understood the meaning of different point colors.

9.3.3 Graph Interactions and Data Interpretation

The **Graph** component was a bit harder to learn. Only 3 participants discovered that left-click dragging (brushing) on the graph highlights points. Similarly, the need to right-click on the map to remove the highlight felt unintuitive. One participant noted that the Y-axis lacked a clear explanation.

No one realized they could drag sub-graph to reorder them or that a “max y” button existed to automatically sort sub-graphs. However, after brief explanations, participants appreciated these advanced features.

9.3.4 Overall Experience and Suggested Improvements

Overall, participants agreed that after a short period of guidance, they fully understood the application. Based on their comments several improvements were proposed:

- **Tutorial:** Add a short tutorial (e.g., an overlay at first launch) to introduce key features.
- **Visibility of Controls:** Make sorting and visualization controls (like “Sort Y”) more noticeable.
- **Clearer Labels:** Add clearer axis descriptions and consider showing the number of currently selected or highlighted points.
- **Bug Fixes:** Correct the bug where sorting the table shows all data instead of only filtered data.

These changes would help users discover advanced features more easily, reduce confusion, and improve the overall user experience.

9.3.5 Questionnaire Results

After the hands-on testing, participants filled out a feedback form that rated different aspects of the application on a five-point scale (1 = Strongly Disagree, 5 = Strongly Agree). Unlike the open-ended comments given earlier, these ratings focused on specific features and gave a more structured view of how users felt. The following sections summarize these results and highlight key issues and successes.

Clarity of the User Interface

Overall, users found the interface fairly clear (see Figure 9.1), though not everyone gave the highest score. The main problem was understanding the purpose of the graph. While the general layout seemed fine, adding labels or short instructions for less intuitive parts of the graph could help users understand the whole interface better.

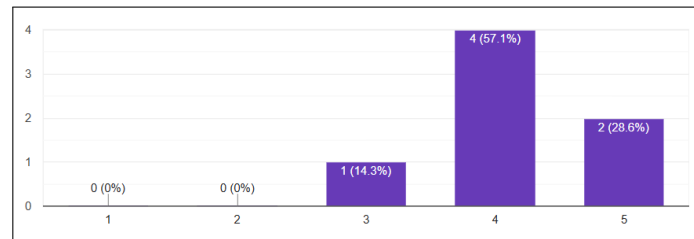


Figure 9.1: Ratings for the clarity of the user interface. (1 = Strongly Disagree, 5 = Strongly Agree)

Effectiveness of Working with Filters

As shown in Figure 9.2, most participants found using filters effective. However, two users rated it with a 3, saying they were confused by the date format and struggled to select all event types. Those who did not give a top rating also mentioned having trouble picking the right type of visualization. This suggests that clearer instructions for filters and better guidance in choosing visualizations would help.

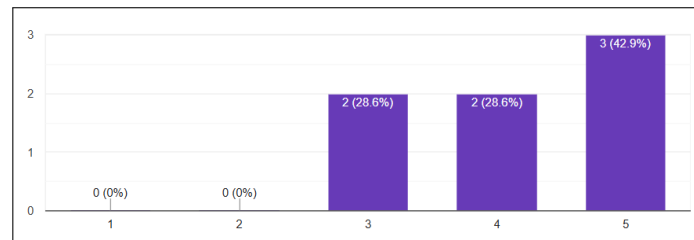


Figure 9.2: Ratings for the effectiveness of using filters. (1 = Strongly Disagree, 5 = Strongly Agree)

Clarity of Visualizations

Opinions on visualization clarity varied more than in other areas (see Figure 9.3). One user gave it a rating of 1 because they were confused by a mix of English and Czech labels. Others had a hard time understanding what the colors meant—whether they represented time-related information or event types—and struggled to interpret sub-graphs. Using one language consistently and adding clear legends or hints would likely improve understanding.

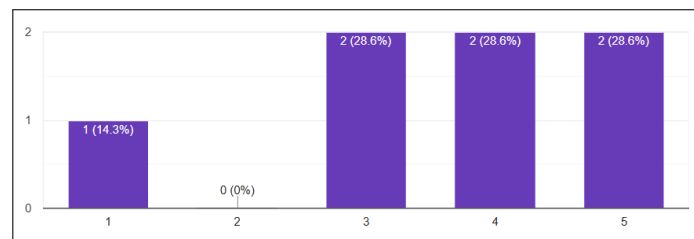


Figure 9.3: Ratings for the clarity of the visualizations. (1 = Strongly Disagree, 5 = Strongly Agree)

■ Usefulness for Data Analysis

Even though some users had trouble with certain parts of the interface, nearly all gave the highest rating for the application's usefulness in data analysis (see Figure 9.4). This shows that the main goal of the application—helping users analyze aviation data—was achieved. Users recognized its value despite some issues with clarity.

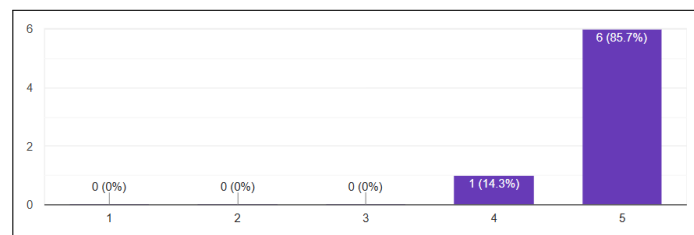


Figure 9.4: Ratings for usefulness in analyzing data and identifying clusters. (1 = Strongly Disagree, 5 = Strongly Agree)

■ Speed of Finding Information

As shown in Figure 9.5, most participants found it quick to locate the information they needed. One user, who rated it 3, had trouble finding sub-graphs easily. Overall, these scores suggest that most navigation elements worked well, though it may help to highlight certain features to make them easier to spot.

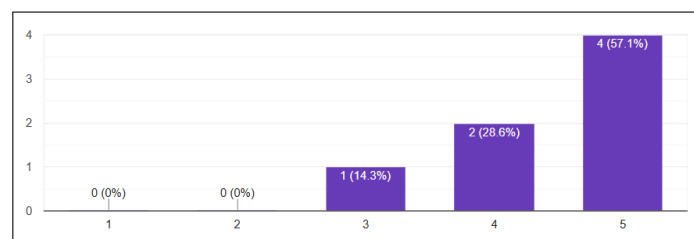


Figure 9.5: Ratings for the speed of finding information. (1 = Strongly Disagree, 5 = Strongly Agree)

User Level of Confidence

Figure 9.6 shows that no participants felt very confident using the application. This may be because there were intentionally no tutorials provided before the test. Offering simple instructions, tooltips, or a short tutorial could help users feel more confident in themselves while using the system.

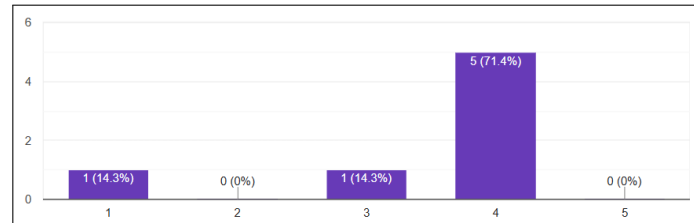


Figure 9.6: Ratings for user confidence. (1 = Strongly Disagree, 5 = Strongly Agree)

Overall Satisfaction

Despite some lower ratings in specific areas, overall satisfaction was high (see Figure 9.7). This suggests that users generally appreciated the application's core features. By improving instructions, labels, and other clarity issues, the user experience could be even better.

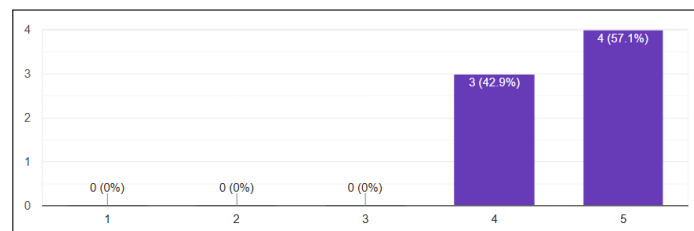


Figure 9.7: Ratings for overall satisfaction. (1 = Strongly Disagree, 5 = Strongly Agree)

9.4 Testing summary

The user testing showed that while participants found the application helpful for analyzing data, some parts were difficult to use without guidance. Many struggled to discover advanced features, like reordering graphs, highlighting points, or resetting filters, without being shown how. Adding a short tutorial could help users learn these features more easily. Improving labels on the axes, filters, and visualization options would also make the interface clearer.

Despite these issues, participants appreciated the application's ability to analyze data and found it useful overall. With a few improvements to make features easier to find and understand, the application could become much more user-friendly while still offering powerful tools for data analysis. These

changes would make it easier for both new and experienced users to work confidently and efficiently.



Chapter 10

Conclusions

This thesis focused on making an application that will make it easier to understand complex spatiotemporal aviation events by showing them on a map, along with related graphs and a table. We began by exploring different visualization methods and discussed which of them could be usable for our project. The D3.js library was chosen because it offers flexible and interactive ways to display data.

We used a user-centered design approach, meaning we involved real users throughout the development process. Through several rounds of feedback and prototypes, we created a tool that lets users filter data, highlight specific events, and see updates instantly across the map, charts, and table. Testing showed that the final application is clear, effective, and helpful for finding patterns and clusters in aviation data. While some features—like brushing the chart or selecting multiple points—were not obvious at first, users came to appreciate them with a bit of guidance.

In the future, this tool could be improved by adding a short tutorial, clarifying labels, and supporting more datasets. It can also be adapted for other fields that need to analyze events spread across time and place. Overall, this work shows that combining interactive maps, graphs, and tables—together with user feedback—can produce a practical resource for decision-makers in aviation and beyond.



Appendix A

Used Software

The following software tools were used in the creation and development of this thesis:

- **Writefull:** Used for grammar check and suggestions.
- **GitHub Copilot:** Used to assist in generating simple or very short code snippets to speed up the development process. These code snippets were manually modified to match the desired functionality.
- **ChatGPT (OpenAI):** Used for grammar checking and short rephrasing of text.

Appendix B

Content of Electronic Appendix

The electronic appendix includes the following directory structure. These files, together with link to application online, can also be accessed online via the GitHub repository at: <https://github.com/jirikdan/AviationDataVisualization>.

- `css/`
 - `brush.css`
 - `colors.css`
 - `filters.css`
 - `linechart.css`
 - `mapStyle.css`
 - `styles.css`
 - `table.css`
 - `toggleButton.css`
- `images/`
- `js/`
 - `constants/`
 - `constants.js`
 - `dataFiltering/`
 - `dataFiltering.js`
 - `dataHandling/`
 - `data.js`
 - `dataArray.js`
 - `linechart/`
 - `linechart.js`
 - `seeMoreLinecharts.js`
 - `subLineChart.js`
 - `map/`

- map.js
- selection.js
- table/
 - table.js
- index.html
- README.txt




Bibliography

- [1] ScienceSoft. Big data visualization: Value it brings and techniques it requires. Last accessed on 12/22/2024. Available online at: <https://www.scnsoft.com/data/big-data-visualization-techniques>.
- [2] Certainty Software. The difference between leading and lagging safety indicators. Last accessed on 12/22/2024. Available online at: <https://www.certaintysoftware.com/the-difference-between-leading-and-lagging-safety-indicators/>.
- [3] Václav Havel Airport Prague. Safety comic strip - episode 6: Handling of passengers. Last accessed on 12/22/2024. Available online at: <https://www.prg.aero/en/safety-promotion>.
- [4] Tamara Munzner. *Visualization analysis and design, Marks and Channels*, pages 95–115. CRC press, 2014.
- [5] Holistics Documentation. How to create a point map. Last accessed on 12/02/24. Available online at: <https://docs.holistics.io/guides/map/create-point-map>.
- [6] Mapline. Heat map. Last accessed on 12/02/24. Available online at: <https://mapline.com/demo/heat-maps/>.
- [7] Mapbox. Binning: An alternative to point maps. Last accessed on 12/02/24. Available online at: <https://blog.mapbox.com/binning-an-alternative-to-point-maps-2cfc7b01d2ed>.
- [8] Venngage. Choropleth map, 2022. Last accessed on 06/11/24. Available online at: <https://tinylink.net/GSI0j>.
- [9] Esri. How create space time cube works. Last accessed on 12/09/24. Available online at: <https://pro.arcgis.com/en/pro-app/latest/tool-reference/space-time-pattern-mining/learnmorecreatecube.htm>.
- [10] Pennsylvania State University. Animated maps. Last accessed on 12/09/24. Available online at: <https://www.e-education.psu.edu/geog486/node/720>.

- [illegible]

- [22] Václav Havel Airport Prague. Safety management system - václav havel airport prague. Last accessed on 10/22/24. Available online at: <https://www.prg.aero/safety-management-system>.
- [23] Václav Havel Airport Prague. Safety propagation - václav havel airport prague. Last accessed on 10/22/24. Available online at: <https://www.prg.aero/propagace-safety>.
- [24] Václav Havel Airport Prague. Safety briefs - václav havel airport prague. Last accessed on 10/22/24. Available online at: <https://www.prg.aero/safety-briefs>.
- [25] Fabio Villegas. Categorical data: Definition, types, features + examples. Last accessed on 12/02/2024. Available online at: <https://www.questionpro.com/blog/categorical-data/>.
- [26] Pritha Bhandari. Ordinal data: Definition, examples, data collection & analysis, 2023. Last accessed on 05/20/24. Available online at: <https://www.scribbr.com/statistics/ordinal-data/>.
- [27] IBM. Geospatial data. Last accessed on 12/09/24. Available online at: <https://www.ibm.com/topics/geospatial-data>.
- [28] SafeGraph. 12 methods for visualizing geospatial data on a map, 2024. Last accessed on 20 May 2024. Available online at: <https://www.safegraph.com/guides/visualizing-geospatial-data>.
- [29] Ali Hamdi, Khaled Shaban, Abdelkarim Erradi, Amr Mohamed, Shakila Khan Rumi, and Flora D. Salim. Spatiotemporal data mining: a survey on challenges and open problems. *Artificial Intelligence Review*, 55(2):1441–1488, 2022.
- [30] Mendi Benigni. Using a motion chart to visualize data. Last accessed on 12/09/24. Available online at: <https://blogs.charleston.edu/tlt/2023/01/31/using-a-motion-chart-to-visualize-data/>.
- [31] Mike Bostock. What is d3? Last accessed on 12/02/24. Available online at: <https://d3js.org/what-is-d3>.
- [32] TutorialsTeacher.com. Dom manipulation using d3.js. Last accessed on 12/02/24. Available online at: <https://www.tutorialsteacher.com/d3js/dom-manipulation-using-d3js>.
- [33] D3 Graph Gallery. Interactivity in d3.js. Last accessed on 12/02/24. Available online at: <https://d3-graph-gallery.com/interactivity.html>.
- [34] Colin Eberhardt. D3.js performance: Rendering one million points with d3, 2020. Last accessed on 12/02/24. Available online at: <https://blog.scottlogic.com/2020/05/01/rendering-one-million-points-with-d3.html>.

- 
- [35] Jurek Kirakowski and Nigel Bevan. *A user-centred approach to design and assesment*, pages 8–9. HFRG and NPL, 1998.
- [36] Jurek Kirakowski and Nigel Bevan. *Key Activities in User-Centered Design*, pages 11–17. HFRG and NPL, 1998.
- [37] Jurek Kirakowski and Nigel Bevan. *Introduction to usability methods*, pages 18–24. HFRG and NPL, 1998.
- [38] Jurek Kirakowski and Nigel Bevan. *Selecting an appropriate method*, pages 106–117. HFRG and NPL, 1998.
- [39] GeeksforGeeks. Prototyping approaches in software process, 2021. Last accessed on 2024-12-05. Available online at: <https://www.geeksforgeeks.org/prototyping-approaches-in-software-process/>.
- [40] Iulia Sorodoc. Low-fidelity vs. high-fidelity prototyping: Key differences explained, 2024. Last accessed on 2024-12-05. Available online at: <https://www.protopie.io/blog/low-fidelity-vs-high-fidelity-prototyping>.