Master's Thesis in Geographical Information Science nr 150

Analysing methods for visualizing time-series datasets in open-source web mapping

Viggo Lunde

2022
Department of
Physical Geography and Ecosystem Science
Centre for Geographical Information Systems
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Viggo Lunde (2022). Analysing methods for visualizing time-series datasets in open-source web mapping
Master's degree thesis, 30 credits in Master's in Geographical Information Science
Department of Physical Geography and Ecosystem Science, Lund University

Supervisor: Finn Hedefalk, Centre for Economic Demography and Department of Economic History, Lund University
Technical Supervisor: PhD. Mattia Natali, Senior Engineer, NIBIO (Norwegian Institute of Bioeconomy Research)
Examiners: Vaughan Phillips and Pengxiang Zhao

ABSTRACT

Geography is the study of knowledge in a given location, and history is the study of knowledge over time. The combination of changes over time has always been challenging to visualize in maps. The main aim of this thesis was to analyse differences between various methods used to visualize spatio-temporal data with open-source web mapping technology. The methodology of this thesis has been split into three main parts: (1) data collection and preparation; (2) application development; and (3) performance testing and statistical analysis.
Through a comparative study between four technology solutions/methods to visualize spatio-temporal data, this work has tried to document if there are differences in loading times and efficiency results evaluated based on performance tests. Four time series datasets ranging in sizes were utilized as the test data for the four different techniques, GeoJSON, WMS, D3 and Cesium. A large dataset contained all the farms in Norway over the last 20 years. A medium-large dataset contained the farms in the three northernmost counties of Norway. Two datasets, medium and small are from a radio collar that reported the position of the deer in Lærdal every hour with different time periods. The performance time results shows that the WMS technology was the fastest for data loading in terms of the display time and total loading time (including all JavaScript and Hypertext Markup Language for all four datasets). Cesium, GeoJSON and D3/TopoJSON were also usable for small datasets but failed the large data tasks. The results regarding animation efficiency were that WMS was the only technology that was able to display all four datasets. However, for the small datasets, the GeoJSON technology seemed to be the fastest, but the differences in time were so small that there were no significant differences among the animation efficiency measures.
The overall conclusion is that only one of the four technologies handled all datasets and that was the WMS application. WMS was the fastest method for data loading and had the smallest display times and total loading times, including all JavaScript and HTML, for all four datasets. And for efficiency and animation the conclusion was that WMS was the only technology that was able to display all datasets.
What of four open-source-based methods are most optimal for visualizing spatio-temporal data are the contribution of this work. There are also some suggestions to future work in the discussion section.

PREFACE

This thesis was written by Viggo Lunde and supervised by Finn Hedefalk at the Centre for Economic Demography and Department of Economic History at Lund University. Mattia Natali was a technical supervisor from the NIBIO. This thesis concerns web development in the geomatics section of the NIBIO. A lot of time has been spent in this work trying to find others who has made comparisons of the formats in OpenLayers as has been done in this thesis without success.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS DEFINED

2D: Two-dimensional mapping (north and east coordinates in a flat plane).
3D: Three-dimensional mapping (north, east and height, like virtual globes).
4D: 3D plus time.
ANOVA: Analysis of Variance.
Cesium: Open-source JavaScript library for creating 3D globes.
D3: Data-Driven Documents, a JavaScript library for producing dynamic, interactive data visualizations in web browsers.
Efficiency: Performance time of animation in browser.
GeoJSON: JSON for geographic data. Data format designed for representing simple geographical features, along with their non-spatial attributes.
HTML: Hypertext Markup Language.
JSON: JavaScript Object Notation, an open-standard human-readable file format and data interchange format.
Mashup: A term from two of my references (Wood, 2007) and (Kim, 2019).
NIBIO: Norwegian Institute of Bioeconomy Research.
OpenLayers: Open-source JavaScript library for displaying map data in web browsers as maps. It provides an API for building rich web-based geographic applications.
Performance time: Time measured in browser for total loading of web application, loading data to application and time to show first point in map.
Spatio-temporal: Spatial refers to space. Temporal refers to time. Spatiotemporal, or spatial temporal, is used in data analysis when data are collected across both space and time.
SVG: Scalable Vector Graphics, an extensible markup language (XML)-based vector image format for two-dimensional graphics with support for interactivity and animation.
TopoJSON: TopoJSON is an extension of GeoJSON that encodes topology.
WMS: Web Map Services, OGC standard.

1 INTRODUCTION

Spatial time series data have always been a challenge for cartographers to visualize in maps. (Han, 2018). According to Corbett (2012), one of the earliest examples is Minard's visualization of Napoleon's 1812 march to visualize data on time maps. This was a starting point of visualizing time-series datasets, and several GIS companies have also utilized the same data and concept with modern technology to show the time axis in a visual manner in a web environment. The combination of changes over time and the differences between open-source technologies are the main issues of this work. It combines geography as the study of knowledge in a location and history as the study of knowledge over time.

Modern computer technologies provide opportunities that are better than ever before for the storage, management, visualization, and analysis of dynamic spatial data over time. Web applications are also becoming more and more stable and larger amounts of data at an ever-increasing pace. Spatio-temporal visualization is a way to illustrate changes in an area over time on a map. The challenge of displaying time on interactive maps or web mappings is smaller compared printed maps. Unlike printed maps, these maps can be made with built-in animations that allow the user to see changes in an area over time. Some Internet maps have sliders that allow the viewer to see a snapshot of the exact point in time that they want to know about simply by sliding the slider to the appropriate time. Other maps are animated and allow the viewer to see a time lapse illustration that covers a set time period. A typical scenario for both types is to compare information in a given place over a period of time.

In the literature in this area of research, few studies with comparisons of open-source technologies for showing time series in web maps have been found. However, an increasing amount of time-series data from satellites and modern sensors in planes are produced(Gómez et al., 2016), and such methods are becoming increasingly in demand to show how different datasets change over time.

The present thesis has tried to find the differences between the technologies in the open-source landscape to provide a better way to choose which technologies to use in different scenarios. If a small dataset is given, then one technology might be better, but if larger datasets are studied, then another technology might be preferred.

Four different open-source technologies for visualizing time series data were compared, and the differences between them were measured. The four technologies developed and measured are as follows:

1. GeoJSON data in OpenLayers
2. TopoJSON data in D3 with OpenLayers,
3. WMS raster maps with OpenLayers and
4. GeoJSON with Cesium.

The data preparation time, loading time, and animation efficiency of each method were the main subjects in this work.

In the studies of Andrienko et al. (2003) one main issue was determining the types of technologies that are suitable for visualizing spatio-temporal data in web map applications. For the end-users of geo-visualization tools, it would be convenient if

this thesis could advise developers or users regarding which techniques to utilize in various situations. The corresponding knowledge base could be built from different techniques. My work will continue in terms of finding more knowledge about what technology is best to use in this field with datasets of different sizes. Different web applications were developed and measured with regard to data preparation time, loading time and animation efficiency.

Before starting the research, it was expected that WMS technology would be preferable in most cases, at least for large datasets.

In the context of the current study, the differences between various open-source methods regarding their ability to visualize spatiotemporal data were mainly of interest. The results should point developers in the right direction when choosing what technology to use.

1.1 Aim and research questions

The main aim of this thesis was to the analyse differences between various methods used to visualize spatio-temporal data with open-source web mapping technology.

Questions:
This thesis aimed to answer the following main research question: “What open-source-based methods are most optimal for visualizing spatio-temporal data with regard to preparation, loading time, and efficiency?”

Objectives:
This thesis has one main objective to answer the research question.
1. Statistically analyse the differences between the open-source techniques with regard to the following:

A) Loading time performance in the client
B) Efficiency of the animation that shows the time series data
C) Preparation time for the data to be visualized

1.2 Outline of the thesis

To analyse the differences between the methods used to visualize spatio-temporal map data with open-source web technologies this work will cover the following structure.

Chapter 1: Introduction
What of four open-source-based methods are most optimal for visualizing spatio-temporal data are the contribution of this work.
Chapter 2: Literature review
Previous researchers have done the same web performance measures as this work, but it has not been done on the same methods. The same measures have been used for other studies comparing for example javascript libraries.
Chapter 3: Materials and methods
Time measures through the window.performance object in the browser were used to compare the four methods. It is suitable for this study because no additional code had to be added to the javascript applications. And the numbers could easily be exported to a file after all the runs of the code.
Chapter 4: Results
Results for preparation time is that data preparation for the different technologies used does not have significant differences timewise. The performance time results shows that the WMS technology was the fastest for data loading in terms of the display time and total loading time.
Chapter 5: Discussion
The overall conclusion was that only one of the four technologies handled all datasets and that was the WMS application. WMS was the fastest method for data loading and had the smallest display times and total loading times, including all JavaScript and HTML, for all four datasets.

2 LITERATURE REVIEW

This section reviews the related studies and techniques that were applied within this thesis. The following three topics are reviewed: 1) spatio-temporal data visualization; 2) open-source techniques for visualization; and 3) performance testing for web sites.

2.1 Spatio-temporal data visualization

The following paragraphs summarize the current state of spatio-temporal data visualization. Different ways of displaying time series are described in this section. Wood et al. (2007) were able to design and combine specific encodings and interactions by interactively 'slicing and dicing' according to time, geography and attributes. Their client-side filtering approach by time was functionally useful and achieved impressive geo-visualization speeds; communicating temporal data through tag clouds in a map allowed for spatio-temporal visualization as well as spatial aggregations and selections. They also asked for further consideration in visualization case studies that integrate other technologies and alternative data, which is one of the goals within this thesis.

There are several approaches for displaying time series data, and Brunsdon et al. (2007) compared such methods. The techniques highlighted in their work include map animations, the co-mapping approach and the isosurface approach, all of which have potential advantages (and limitations) for researchers interested in the exploratory visualization of point-level data; their work related to crime events suggested that more evidence is needed to show how users benefit from geo-visualization tools. A comap is an extension of a coplot, or conditional plot, but with maps. These maps show different conditions side by side or arranged in a rectangular set of panels in time order.
Isosurfaces can be three dimensional plots with spatial data on two axes and time on the third axis. Brunsdon found them to be hard to identify in many cases. In the present work, one type of map animation was chosen. In the following study, the time series represented the attribute changes used.

The methods for illustrating changes in an area over time on a map were split into three main approaches by Dibiase et al. (1992). They stated that dynamic variables could be used to emphasize the location of a phenomenon, emphasize its attributes, or visualize changes in its spatial, temporal, and attribute dimensions. They were visualizing chronological changes through time series, visualizing attribute changes through re-expression, and visualizing spatial changes through fly-by map animation and how these methods could be used to visualize spatio-temporal data in both realistic (chronological) and abstract (reordered and paced) forms. In this early animation study, the authors highlighted the combination of static maps, graphs, diagrams and images and showed that animation enhances analysts' abilities to express data in a variety of complementary forms.

The following 4D studies were important because 3D mapping with time changes was a part of the present thesis, and the 4 dimensions in this case included north and east coordinates, height and time. Resch et al. (2013) described 4D cartography in 2013 as follows. The 4D representation (three spatial dimensions plus time) of time-varying

phenomena, i.e., through thematic 4D mapping, was still widely untouched. It was demonstrated that 4D cartographic representations could help to understand dynamic spatiotemporal phenomena. A central question suggested for future research was how to represent time in 4D visualizations. The efficient and effective 4D visualization of spatial data on the Internet was also a goal (Resch et al., 2014), and the authors solved this problem by displaying time-series diagrams of environmental parameters with WebGL. Future research questions involved the representation of the time dimension in 4D systems, optimized and generic temporal generalization, and the possibility of integrating and creating (pseudo)photorealistic illustrations in web-based geo-visualization systems. Other studies used virtual globes (3D) to show data effectively and to compare 2D and 3D background maps.

A systematic meteorological data visualization framework on a virtual globe, which extended the emerging virtual globe platform to include the simulation and visualization of meteorological data for climate studies, was presented by Liu et al. Most existing visualization technologies in 2015 could easily be transferred to the globe by redefining their stated coordinates (Liu et al., 2015). They also presented a virtual globe application of a tropical cyclone demonstrating that virtual globes could be an effective tool for meteorological data visualization and analysis.

In another presentation, three types of animated geo-visualizations of water depth based on tide information were included by the authors: (1) a 2D cartographic map (abstraction), (2) a 2D ortho-photorealistic image (photorealism) and (3) a 3D immersion (3Drealism). Their work showed three results at different tidal stages (low, medium and high levels). They also found that the third dimension provides more interaction possibilities (displacement in 3D) for users and increases the comprehension of the tide phenomenon in terms of producing more realistic sea animations (Masse and Christophe, 2016).

Murshed et al. developed a 4D canvas web application based on the open-source Cesium virtual globe, and the present work used Cesium as one of the evaluated technologies. The authors used their application to dynamically visualize multiple energy simulation results, such as the techno-economic photovoltaic potentials or energy needs of 3D buildings, in the context of different Asian and European cities. Analytical functionalities were also integrated, and a GUI was built to allow users to conduct explorative analyses of the results. This study demonstrated that with the wide availability of 3D datasets and technological advancements in virtual globe software frameworks, many different smart city applications can be integrated into the developed application (Murshed et al., 2018).

Expensive pre-processing steps can be offloaded to the associated server or cloud when the client machines do not have sufficient computational power to perform rendering at frame rates suitable for interactivity, and rendering in the client was usually performed earlier. However, this is now changing due to improvements in CPU and GPU technology on mobile devices and networking technology.
Hybrid visualization approaches are more promising because they exploit the entire spectrum of available computational resources (Mwalongo et al., 2016). Data access and filtering can be handled by a separate data service layer that hides the differences between data sources and presents a uniform data access interface to the visualization layer; this was a conclusion in the above work.

Some studies represented below reviewed big data and machine learning, which is beyond the scope of the present paper, but they also heavily examined web-based visualization, which is why they are mentioned here. Li et al. developed a cyberinfrastructure solution for visualizing big, multivariate and time-varying climate data in 2017. A major challenge was that collaborative scientific analysis has focused on how to handle the organization, transmission and rendering of these data over the web because their data were rather voluminous. They demonstrated the applicability of the video encoding technique in the PolarGlobe web-based visualization platform and suggested ideal choices for video compression techniques under various network environments (Li and Wang, 2017). Moreover, they suggested including machine learning techniques to automatically capture abnormalities in the climate system, such as extreme weather, cyclones, and their moving trajectories.

Padilla et al. proposed a dual-process cognitive framework that expanded on visualization comprehension theory (which was supported by empirical studies) to describe the decision making process with visualizations. They also offered practical recommendations for visualization design that take human decision-making processes into account (Padilla et al., 2018). In the present study, there are also guidelines on what technology to choose.

Moreover, Christophe (2020) experimented with a reconciliation between computer vision and map design, with the dual purpose of spatio-temporal analysis and the visualization of geospatial data, while combining knowledge and methods to obtain a more flexible approach for geo-visualization methods and tools. He also stated that visualization could facilitate the first steps of decoding, reasoning and learning. In particular, the issue of the interpretability of spatio-temporal data, from raw to simulated data, could help to visually analyse geophysical, climatical, sociodemographic, and historical phenomena and dynamics on earth. This line of research trended more toward machine learning than I went with spatio-temporal data in this thesis. However, they used style to display some data in a time window, as we did.

Park et al. demonstrated the nationwide spatial and temporal distributions of two major air pollutants, PM10 and NO2, over 14 years (from 2001 through 2014) in South Korea. They recommend a web-based visualization of scholarly data for active research and data sharing beyond academia by using an easy application without advanced technical skills, thereby allowing users to effectively understand scientific information. Such approaches not only lead to the proliferation of related studies but also improve public awareness and understanding (Park et al., 2020).

(Loechel and Schmid, 2013) argued that a fast response to a web request was a mandatory characteristic of Web Map Services. They proposed several caching techniques, such as tile caching and reverse proxy caching, for speeding up the responses of WMS.

Some main points and methods that were used in this thesis are summarized in this section. Dynamic data within interactive maps (obtained using the time slider) have been the focus in this thesis. Interactive maps often have two main components: background map layers and theme layers. Background maps change when you zoom

in and pan the map, and the theme layers are displayed on top of the background map to convey the message regarding the theme you want to highlight.
Differences in visualizing static and dynamic data in the theme layer were a major issue in this thesis. For static theme data in a web mapping application, the system loads and shows all data immediately when the application has finished loading in the browser. For spatio-temporal theme data, some data are shown first, and then more data are shown successively along a time axis. In this thesis, a time slider was chosen because the user could then move the slider to the desired time to see these differences. As the slider moves, new data are shown in the map either by loading all the data and only showing some of the data (GeoJSON) at a given time or by constantly loading new data (WMS)s.

In the context of the 4D data sharing on the Internet, (Hejmanowska et al., 2019) specify two kinds of problems. The first is the optimal presentation of 4D models, including making them available on the Internet, and the second, the integration of 4D models with GIS data on the Web GIS platform. Their conclusion was that results obtained for publications using X3D technology were not satisfactory, both in terms of the appearance of the 3D model and the functionality of the browser. Integration of 3D / 4D models is possible at the WebGIS level, with raster base maps shared as OpenLayers, cartographic layers and vector objects with attributes.

In this thesis, one of the three methods described in the study by (Dibiase et al., 1992), visualizing chronological changes through time series, has been the focus. As they stated, the depiction of changes over time in the positions or attributes of geographic phenomena from a constant viewpoint (changes in a chronological space) is the most obvious application of animation in both the social and physical sciences.

(Wood et al., 2007) asked for further consideration in visualization case studies that integrated other technologies and alternative data, as we have tried to do in this work. WMS is one of the five methods that were used in our work, as in that of (Loechel and Schmid, 2013). Fast response times are crucial to web applications today, and this work analysed some methods to find the abovementioned differences.

2.2 Open-source techniques for visualization

There are several open-source techniques for web visualization, and some are more suitable for visualizing spatio-temporal data than others. Openlayers were used in all of the tests in this work and were chosen because the literature highlights it as one of the most relevant and best web mapping libraries.

CampToCamp is one of the leading companies developing geospatial solutions, and in an article at dev.to, the developers highlighted what they thought were the three best open-source mapping libraries. Leaflet, Mapbox GL and OpenLayers are their favourites, and they considered OpenLayers to be the most complete library. Because OpenLayers has extensions such as ol-cesium that manage the bindings between OpenLayers and CesiumJs (a great virtual globe library written in WebGL), to switch from 2D to 3D maps, this thesis chose OpenLayers. The developers also mentioned that a massive crowdfunding campaign recently leveraged hundreds of thousands of dollars to help the community progress and enhance the library. Thus, it is safe to

build applications with OpenLayers and trust that the library will not stop developing and supporting them for many years to come.

Openbase.com has also made a list with the "12 Best Vanilla JavaScript Map Libraries", and the three best-rated map libraries according to the Openbase team and community were the same three as those in CampToCamp's article, with OpenLayers at the top. Vanilla JavaScript refers to using plain JavaScript without any additional libraries or frameworks such as React, jQuery or Angular. This, combined with our own experience with both OpenLayers and Leaflet, made us decide to use OpenLayers with different data formats in this master's thesis.

To display spatio-temporal data, it is a good choice to show different data in the same place in a sequence. OpenLayers is good at this, and studies looking into comparisons of such methods were reviewed. The aim of my study was to find an effective way to display time series data. A study looking at how to easily and quickly visualize a large amount of data via an Internet platform was deemed relevant. From the perspective of Netek et al. (2019), the main aim was to test the point data visualization possibilities of selected JavaScript mapping libraries to measure their performance and ability to cope with a large amount of data. The mapping libraries OpenLayers, Leaflet, MapBox and Supercluster were studied, and according to a result in this thesis, large data must be handled carefully to be loaded properly in a map application.

In another study Farkas (2017) identified the most capable libraries for acting as the basis of a web GIS client (Cesium, Leaflet, NASA Web World Wind, OpenLayers 2, and OpenLayers 3) and compared them. The libraries were compared by their GIS feature coverage and some quality metrics. OpenLayers 3 was identified as the most capable library due to its support for nearly 60% of the examined GIS features, its small size, and its moderate learning curve. OpenLayers seemed to be a recurring theme in these reviewed studies.

To summarize what open-source technologies to choose in the present study, OpenLayers with different data formats and technologies, Cesium, D3 and WMS were used from a GeoServer. OpenLayers was our choice because there was experience with it for many years in our organisation, and Farkas (2017) also meant OpenLayers was the best choice. Cesium shows data in 3D, which is important for understanding how data are placed in the terrain, and as discussed by (Bing et al., 2016), this was important to try out in our work. D3 and TopoJSON were described by (Sack et al., 2014) and inspired the D3 applications in the present study. The study of (Król, 2018) is an important study because they made prototype applications used for testing and conducted a comparative performance analysis like that done in the present work. One difference is that they compared only raster map viewers, and the present work tested and compared both vector and raster solutions. The questionnaire-based work of Ballatore et al. (2011) was done quite a few years ago but is still of some relevance because it dealt with both OpenLayers and GeoServer. This indicates that both are mature technologies that will continue to be relevant for many years to come.
Four different technologies were tested in the present thesis are described in the following sections. A map client was made with the open-source library OpenLayers. The client had background maps and showing time series data as points on the top

using these different data formats.

- **GeoJSON** – An application with OpenLayers and the GeoJSON data format was developed and tested. GeoJSON is an open-standard format designed for representing simple geographical features along with their non-spatial attributes. It is based on the JSON format. OpenLayers can read this format and make vector features to draw on the map.
- **WMS** – An application with OpenLayers and the WMS raster data format was developed and tested. A Web Map Service (WMS) is a standard protocol developed by the Open Geospatial Consortium in 1999 for serving georeferenced map images over the Internet. OpenLayers can display this format as images on a map. The images were served by a GeoServer.
- **D3** – An application with OpenLayers and D3 integration was made. OpenLayers loads TopoJSON geometries and uses D3 (d3.geo.path) to render these geometries to an SVG element on top of the OpenLayers background map.
- **Cesium** – An application with OpenLayers and Cesium integration was developed, and GeoJSON data were loaded and drawn on top of the Cesium background map.

Under is a subsection for each of these with references to the literature. For further description about the development of these methods, see the system integration section.

2.2.1 GeoJSON

GeoJSON is an open standard format designed for representing simple geographical features, along with their non-spatial attributes. In our case time was one of the attributes used to show the spatiotemporal data. A GeoJSON file was loaded as a map layer in the client.

Shang (2015) studied efficient vector mapping with vector tiles based on cloud server architecture. The study compared GeoJSON and TopoJSON with google protocol buffers and used D3.js to render the vector data. The results showed that overall performance of vector tile based vector maps can be improved by applying distributed memory caching implementation on the server side as compared to naïve architecture.

A questionnaire was designed by Ballatore et al. (2011) to obtain responses from the relevant online communities about a given set of characteristics. This article described the analysis of 14 open-source projects in the area of web and GIS technologies, providing first-hand information about open-source web and geospatial tools. Both GeoServer and OpenLayers were included and had high overall scores, even though Mapserver had a better score than GeoServer in their survey.

2.2.2 WMS

WMS is a raster image format and Geoserver was used to convert the data from shape to images. To filter out time attributes to show different the CQL_FILTER parameter was used when asking for the image from Geoserver.

Król (2018) conducted a comparative analysis of the performance of selected raster map viewers. The prototype applications were used to test the implementation of

selected design tools. Efficiency measurements were made using tools for automated testing. They stated that it is possible that the results may have been determined by the measurement mechanism rather than the actual performance of the applications. Therefore, it is advisable to complement automated tests with usability tests. The compromise between performance and usability was demonstrated by an application created on the basis of the ImageViewer plugin. It was this latter application that the authors pointed out as objectively the best choice for raster map presentation among all the tested methods.

2.2.3 D3 (TopoJSON)

Data was converted to TopoJSON and was read with the D3 javascript library and rendered to a map layer as a SVG element. Because TopoJSON is a more compact format it should be faster to load than for example GeoJSON.
In a study by Sack et al. (2014) a D3 tutorial that covered the finding and formatting of data, geographic data were converted into GeoJSON and TopoJSON formats, and the data were asynchronously loaded into a browser. Their work with D3 mapping is well documented, so we could implement our D3 application.

2.2.4 Cesium

To look at the world in 3D, one needs a library like Cesium that provides users with the opportunity to see the landscape as a virtual world. Cecium is a javascript library converting the OpenLayers client from 2D to 3D. The spatio-temporal data was added to the client as a vector layer from GeoJSON. A paper by (Bing et al., 2016) introduced a web-based visualization method based on Cesium and the 3D City Database to construct a three-dimensional panoramic electric power visualization system. All tools used in their paper are
open-source and could promote system development and upgrades, as in the present work. The extra Z-axis provided an extra dimension over two-dimensional maps for visualization.

A multidimensional web-mapping platform was developed for citizen participation purposes, where users could better assess and understand the evolution of their cities over time (Lafrance et al., 2019). The spatio-temporal mechanism devised in Lafrance et al.'s research, as well as its components (i.e., animations, charts, and timelines), are relevant for visualizing and analysing the impacts of the choices made during a consultation. Both 2D and 3D spatial dimensions need to be included in a citizen participation platform, given that each of them may be more useful or intuitive depending on the information provided by different scenarios. The authors also included a 4D representation for which Cesium 3D-tiles could be a potential solution to improve performance.

2.3 Website performance testing

The main aims in the present thesis were to test the preparation time, loading time and efficiency of animation for each of the datasets and technologies. Loading time and efficiency are the main two characteristics and are used in performance testing to measure the time required by different methods. In the simplest terms, the page load time is the average amount of time it takes for a page to show up on your screen. It is

calculated from initiation (when you click on a page link or type in a web address) to completion (when the page is fully loaded in the browser). In this thesis, the loading time was split into three different parts: the loading time of the theme data, the display time required by the site to present the first time data on screen, and the total loading time, described as extra page load time. The efficiency was measured after the page finished loading from the time animation and started; 100 time intervals (hours) were shown in the map for the deer data, and 20 time intervals (years) were shown for the production subsidies data.

2.3.1 Common approaches to performance testing

In regard to performance testing, some common metrics, such as load time, bandwidth and concurrent users, factors that are interesting when looking at such issues (Figure 1). In the present work, the scope has been narrowed to the loading time and efficiency of the animation that shows the time series data. There has been little focus on fine-tuning and re-testing in this work but re-tests were done this summer to ensure quality of the data. Later work in this area should have more focus on this aspect.

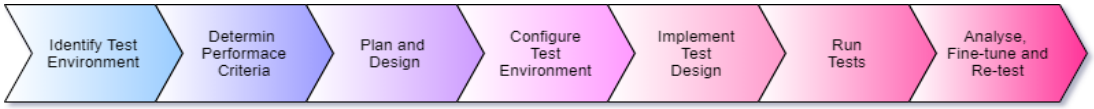

Figure 1: A generic process for conducting performance testing.

A study from 2001 about website testing involved page testing, hyperlink testing and all-path testing. The experimental analysis and testing techniques described by Ricca and Tonella (2001) were early thoughts regarding the assessment of website quality. They provided deep insight into the internal functioning of web applications with more than one page, highlighting the strengths and weaknesses found at that time. However, their study is not as relevant for single-page applications, such as those developed in this thesis.

An important measure of web page performance is how long it takes for a web page to be downloaded sufficiently for the web browser to start displaying the web page (or parts of it) to the user (Graham-Cumming et al., 2017). A web page typically includes the base HTML and potentially a large number of resources that must be loaded so that the web page can be displayed. These resources may include client-side scripts (e.g., JavaScript), cascading style sheets (CSS), and/or images that are required for the correct layout, appearance and functioning of the web page.
In our case, there were JavaScript libraries with CSS, theme data and images such as background map tiles.

Web site performance testing can be divided in two main categories. Single-page web applications and multipage web sites. GIS web applications is mainly single-page loading data from other sources as Geoserver or distributed servers from the mapping authorities. Here is the performance testing loading data from the sources. Other websites are often built of more pages and the performance testing is about loading new pages in the same site.

We focused on performance in terms of the loading times in the client. The tests of (Ricca and Tonella, 2001) were not applied in the present work because we were

testing single-page applications and not sites with many sub-pages. However, the description of Graham-Cumming et al. (2017) helped determine where to put the performance metrics used to measure the loading and animation results.

2.3.2 Related studies on performance testing

Performance testing for a single-page web application was the main issue of the present work. There are several tools or extra JavaScript libraries that could be used, such as selenium or jMeter, but we hoped to add as few JavaScript libraries that could affect the measured times as possible. Our goal was to measure preparation times, efficiency and loading times without needing to load extra objects and use extra time. This section concerns loading time performance testing and what has been done in this area.

2.3.2.1 Websites in general

One study tried to determine the metrics and evaluation methods that best suit a JavaScript framework comparison. A benchmark framework executed tasks to test the efficiency of three JavaScript frameworks (AngularJS, Aurelia, and Ember). The research showed the impact of the environment (CPU usage and network connectivity) on JavaScript frameworks. (Ferreira, 2018). The report used two functions to measure the execution times of JavaScript applications (Date.now() and Performance.now()).

A study comparing the DOM manipulation methodologies of vanilla JavaScript, Angular, React and Vue.js in terms of DOM performance was formed (Persson, 2020). Test applications were created, and these applications were used as a base for comparing application sizes and for comparison tests involving DOM performance-related metrics using Google Chrome and Firefox. The performance interface, which was implemented in both Google Chrome and Firefox, was used to measure the time durations of the test cases.

Another paper analysed page loading time trends, which is an important part of any website's user experience. Manhas (2013) stated "And many times, we'll let it slide to accommodate better aesthetic design, new nifty functionality or to add more content to web pages. Unfortunately, website visitors tend to care more about speed than all the bells and whistles we want to add to our websites." Additionally, page loading time is becoming an increasingly important factor in regard to search engine rankings. This study claimed that high-performance web sites lead to higher visitor engagement, retention and conversion rates. With JavaScript frameworks, they found that from 2008-2013, the size of the average web page had more than tripled, and the number of external objects had nearly doubled.

Nägele et al. (2015) described steps that were taken towards finding a method to measure the client-side performance of web applications. A definition of client-side performance within the context of web applications was formulated, and the properties by which to evaluate possible methods were defined. With these properties, an evaluation method was developed. Two profiling metrics were suggested, of which one metric was eventually implemented and improved in the next step. Finally, an integration method to apply the profiling metric to a project was implemented. The profiling metrics were accuracy, impact, usability and portability. The performance of

the application was affected significantly, as it decreased by approximately 45%. This thesis therefore found that this method was not very relevant.

A native Android application and a multi-platform web application to monitor solar radiation and the output power of a photovoltaic system were developed by Santana et al. (2016). The authors mostly looked into transfer times, and their research was therefore interesting to the work in this study. The response times of both applications were obtained, and the developed web application was very poor compared to the native Android application.

An article from Timanovskyi and Plechawska-Wójcik (2020) presented an analysis of the performance of selected tools when building a single-page application. A Chrome browser with the DevTools tool was used to evaluate the performance of the test application. The total number of tests was 112. As part of the study, a test application was created using different JavaScript frameworks - the angular framework and the Vue.js framework.

2.3.2.2 GIS web applications

A study looking at how to easily and quickly visualize a large amount of data via an Internet platform was conducted. From the perspective of the authors, the main aim was to test the point data visualization possibilities of selected JavaScript mapping libraries to measure their performance and ability to cope with a large amount of data. (Netek et al., 2019) The mapping libraries OpenLayers, Leaflet, MapBox and Supercluster were studied, and Netek et al. (2020) described an experiment to test both raster and vector tile methods. The concept behind raster tiles is based on pre-generating an original dataset
including a customized symbology and style. All tiles are generated according to a standardized scheme. Performance testing in terms of the loading time, data size, and the number of requests was performed. The authors measured different zooming and panning operations in the map and found durations in milliseconds with the Google Chrome console, as in this thesis.

The work of Zunino et al. (2020) assessed the advantages of representing geographical information as vectors over raster representations and was implemented using three popular JavaScript libraries. The presented case study application, namely, the LQI Web map, was built using HTML5 technology, which allowed it to run on different platforms. Their result was a clear indication that OpenLayers was the best library for raster maps on all devices.

Kim and Jang (2019) compared and analysed the dynamic mashup performance characteristics of various map platforms. To efficiently compare the performance, they defined and measured performance comparison metrics, including the data loading time, mashup time and user interaction time. In this thesis, different methods for visualizing time-series datasets and loading times was compared and analysed in a similar way.

To summarize this section, this thesis used the performance library to measure time because it was built inn and no additional code needs to be added. This method is also simple to use. The thesis of Ferreira (2018) also uses the performance library in a similar way as we do in this work. The present thesis also used performance metrics and the differences in these time metrics, as they did in the study of Persson (2020). As the page loading time is becoming a more important factor in regard to search engine rankings (Manhas, 2013), our measures for finding the fastest application have relevance in that area as well. Netek et al. (2020) dealt with a large amount of data, both raster and vector data, and we observed that the vector solutions were not able to effectively handle the largest dataset. Zunino et al. (2020) found that OpenLayers was the best for raster maps on all devices, and they also used vector tiles in addition to GeoJSON. Inspiration and experience from them made us choose OpenLayers as our test mapping library.

3 MATERIALS AND METHODS

To answer the research questions in this thesis, the main approach was to test different technologies for visualizing time series data in web map clients with 2D and 3D background maps. The methodology of this thesis has been split into three main parts: (1) data collection and preparation; (2) application development; and (3) performance testing and statistical analysis. Through a comparative study between four technology solutions/methods to visualize spatio-temporal data, this work has tried to document if there are differences in loading times and efficiency results evaluated based on performance tests.

3.1 Data and study area

Four time series datasets were utilized as the test data for the four different techniques, GeoJSON, WMS, D3 and Cesium.

3.1.1 Study area

All over Norway there are farm points in the large dataset. An area covering the three northernmost counties in Norway before 2020 (Nordland, Troms and Finnmark) is the coverage of the medium-large dataset that is a subset of the large dataset. See Figure 2. A small part of the western area of Norway close to Lærdal is where the medium and the small datasets are located. The datasets contain points from a radio-marked deer. See Figure 3.

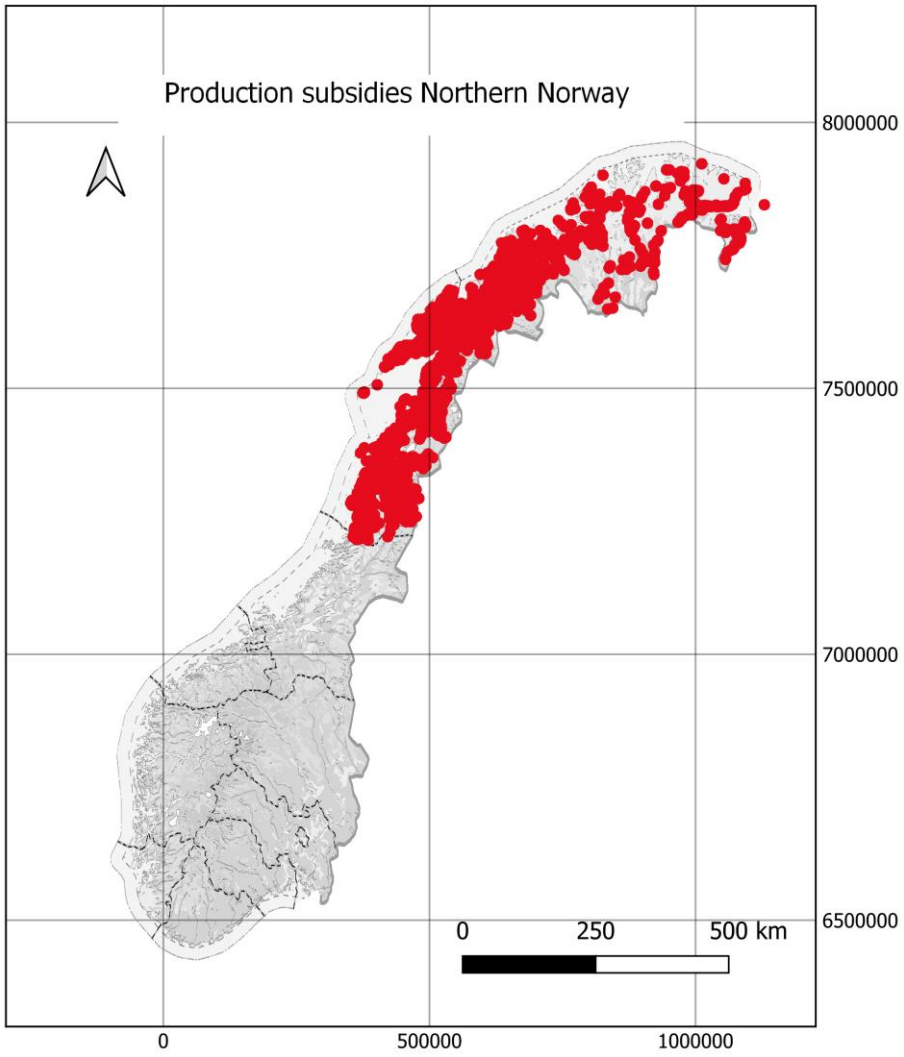

Figure 2: Map of the medium-large dataset study area in ETRS89/UTM zone 33N.

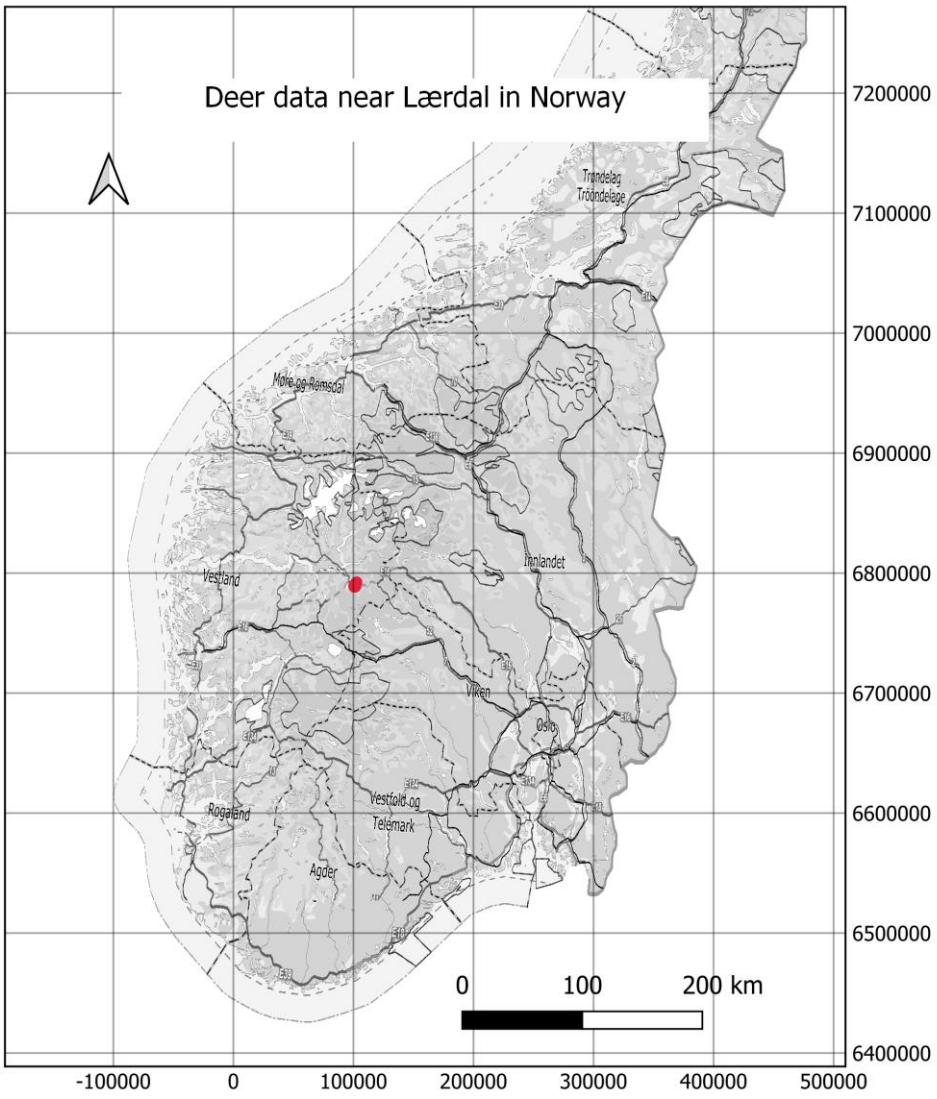

Figure 3: Study area for the small and medium datasets in ETRS89/UTM zone 33N.

3.1.2 Data

Four datasets ranging in sizes were used with the different technologies. These datasets are described in the following.

The **large dataset** contained the farms in Norway over the last 20 years (data from 1999 to 2019); farms were defined as properties that obtained production subsidies from the Norwegian authority. These data were obtained from the NIBIO (Norwegian Institute of Bioeconomy Research). The dataset contains points representing farms each year, decreasing from 66706 to 38983 over this time period. The total number of points in the dataset is 984 832.

The **medium-large dataset** contains the farms in the three northernmost counties of Norway over the last 20 years. The dataset is a subset of the large dataset that uses only the tree northernmost counties to obtain a smaller dataset to work with. The total number of points in the dataset is 145 449.

The **medium dataset** has points from a radio-marked deer.
The data are from a radio collar that reported the position of the deer every hour. The dataset covers a small part of the western area of Norway close to Lærdal. This is a point dataset with a time attribute. We stored the data in a PostGIS database and obtained the data in GeoJSON format for the client through a Java API. The total number of points in the dataset is 5 000.

The **small dataset** also contains points from a radio-marked deer but over a shorter time period. It is from the same part of Norway close to Lærdal. The total number of points in the dataset is 1 500. Table 1is describing the data.

Table 1: Datasets used in the analyses.

Dataset	Format	Reference system	Spatial coverage	Temporal coverage	Temporal detail	No. of objects	Provider
Norwegian farms (large)	Point shapefiles	EPSG 25833	National Norway	1999-2019	year	984 832	NIBIO
Norwegian farms (medium-large)	Point GeoJSON	EPSG 25833	Northern Norway	1999-2019	year	145 449	NIBIO
Radio collar deer (medium)	Point GeoJSON	EPSG 25833	Area near Lærdal	21.Sept to 31.Dec 2020	hour	5 000	NIBIO
Radio collar deer (small)	Point GeoJSON	EPSG 25833	Area near Lærdal	1-30. September 2020	hour	1 500	NIBIO

3.2 System implementation

JavaScript is a text-based programming language used both on the client side and server side that allows you to make web pages interactive. While HTML and CSS are languages that give structure and style to web pages, JavaScript gives web page interactive elements that engage a user. All the present methods were developed with different JavaScript libraries.

Performance tests were performed regarding both data preparation and the web client response and loading time. To do this, the *windows.performance* object in the browser was used. Google Chrome's debugger tools were used to find loading time differences. A small set of points was chosen, and the animation was measured. The web applications were loaded 100 times, and every parameter was stored in local storage in the browser. When the site was loaded the 100^{th} time, the data were retrieved from local storage and saved as a csv-file on the computer running the test.

Statistical analyses of the data obtained from the performance tests were used to find significant differences between the presentation methods. ANOVA tests were run to compare the performance results from the different applications. It was necessary to find the time intervals during which these data should be visualized, as well as to represent the data in a user-friendly way in Internet applications.

To obtain scientific results, excels statistical tools were used to compare the data from the presentation methods regarding speed and loading times.

At the NIBIO, we used GIT and continuous integration to develop web applications. One developmental branch was made first, and then twelve other branches were made for the different methods (4) and datasets (4). The applications were developed on a local computer, pushed to a GIT repository and deployed to two different servers: a development server inhouse (inside NIBIO's firewall) named **utvgeo01.int.nibio.no** and an external test server named **karttest.nibio.no**. The servers were set up with the Ubuntu operating system with Apache, Tomcat and GeoServer.

3.2.1 Open-source technologies

The following list presents the technologies used for development:

- Apache 2.2.15
- Tomcat 8.0.21
- GeoServer 2.18.1
- Parcel Bundler 2.0.0-beta.1
- jQuery 3.3.1
- OpenLayers 6.5.0
- ol-layerswitcher 3.6.0
- D3 5.9.2
- TopoJSON 3.0.2
- proj 4 2.5.0
- Lodash 4.17.20
- Moment.js 2.29.1
- JsPanel 2.6.3
- SweetAlert2 9.10.12
- FontAwesome 5.3.1

Further descriptions of the technologies are as follows:

Apache
Open-source HTTP server for modern operating systems including UNIX and Windows. The goal is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. The Apache HTTP Server ("httpd") was launched in 1995 and it has been a popular web server on the Internet since April 1996. The Apache HTTP Server is a project of The Apache Software Foundation.

Tomcat

Tomcat is the Apache Software Foundation's implementation of a Java "servlet container". Tomcat makes it possible to run Java-based web applications based on servlet or Java Server Pages standards. Open-source implementation of the Jakarta Servlet, Jakarta Server Pages, Jakarta Expression Language, Jakarta WebSocket, Jakarta Annotations and Jakarta Authentication specifications. These specifications are part of the Jakarta EE platform. The Jakarta EE platform is the evolution of the Java EE platform. GeoServer is a Java application and running in Tomcat.

Parcel

Parcel is a compiler for all code, regardless of the language or toolchain. Parcel takes all files and dependencies, transforms them, and merges them together into a smaller set of output files that can be used to run your code. Parcel supports many different languages and file types out of the box, from web technologies like HTML, CSS, and JavaScript, to assets like images, fonts, videos, and more. In this thesis JavaScript was used.

jQuery

jQuery is a JavaScript library that makes things like HTML document traversal and manipulation, event handling, animation, and Ajax simple with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way people write JavaScript.

GeoServer

Geoserver is a Java-based server that allows users to view and edit geospatial data. Using open standards set forth by the Open Geospatial Consortium (OGC), GeoServer allows for great flexibility in map creation and data sharing. GeoServer is free software and distributed under GNU General Public Licence Version 2.0. This significantly lowers the financial barrier to entry relative to proprietary GIS products. In addition, GeoServer is not only available free of charge but is also open-source. All datasets were served as WMS with a query filter from GeoServer version 2.18.1. The data from the GeoJSON files were opened in QGIS and converted to shape files. New styles were made, and the shapefiles were set up as layers in Geos\Server. All data were EPSG:25833.

OpenLayers

Openlayers is a JavaScript library that makes it easy to put a dynamic map on any web page. It can display map tiles, vector data and markers loaded from any source. OpenLayers has been developed to further the use of geographic information of all kinds. It is free, open-source JavaScript released under the 2-clause BSD Licence (also known as the FreeBSD). OpenLayers was used in all applications but has limitations regarding the number of points in vector data.

D3

Data-Driven Documents is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. The D3.js library uses prebuilt functions to select elements; create SVG objects; style them; or add transitions, dynamic effects or tooltips to them. These objects can also be styled using CSS. The data can be provided in various formats, such as JSON, comma-separated values (CSVs) or GeoJSON.

Cesium
Cesium is an open-source JavaScript library for creating 3D globes and maps with the best possible performance, precision, visual quality, and ease of use. A virtual globe such as that of Cesium is a three-dimensional (3D) software model or representation of the Earth or another world. A virtual globe provides the user with the ability to freely move around in the virtual environment by changing the viewing angle and position. Compared to a conventional globe, virtual globes have the additional capability of representing many different views on the surface of the Earth. These views may be of geographical features, man-made features such as roads and buildings, or abstract representations of demographic quantities such as population. In this thesis, points in a map were shown in a 3D globe.

TopoJSON
TopoJSON is an extension of GeoJSON that encodes topology. Rather than representing geometries discretely, geometries in TopoJSON files are stitched together from shared line segments called arcs. TopoJSON eliminates redundancy, allowing related geometries to be stored efficiently in the same file. Used with the D3 technology in this thesis. Installed through Node package manager.

Proj
Proj or Proj4js is a JavaScript library to transform point coordinates from one coordinate system to another, including datum transformations.

Lodash
Lodash is a modern JavaScript utility library delivering modularity, performance & extras. Used to sort on time property for the timeseries map data in this thesis.

Moment.js
Moment is a JavaScript library to parse, validate, manipulate, and display dates and times. Used to parse time property values in this project.

JsPanel
JsPanel is a JavaScript library to create highly configurable floating panels, modals, tooltips, hints/notifiers/alerts or context menus for use in backend solutions and other web applications.

QGIS
QGIS is a professional GIS application that is built on top of and proud to be itself Free and Open-source Software (FOSS). The program can view maps, edit and analyse geographic data and export in most common map file formats. The program has tools for both vector and raster maps.

3.2.2 Application development

A web application with common map functionalities, such as background maps, layer lists, and panels was made first. The following applications described under were based on this common web map application.

1. GeoJSON

The first application developed used OpenLayers to show points from GeoJSON (Figure 4). A GeoJSON file was made from QGIS form data in a PostGIS database and stored locally as a file to avoid the impact of network traffic on later measurements. The data were loaded from the file and read as OpenLayers features. The features were added to an OpenLayers VectorSource. The source was used to make a VectorLayer called hiddenLayer. Style was added to the layer to show the points on the map.

To reach the goal of displaying only one unit of time at once, the time slider was added with an interval of 100 time units to make it easier to measure a statistically significant number of times. To show a filtered subset of the features, another VectorLayer called visibleLayer with a different style was made. The features were ordered by the time attribute using the lodash library.

The animation was started when all points were loaded with the first index among the 100 that we wanted to show, and the source of the visibleLayer was filtered to show only the points for the time interval on the time slider. The filtered points were shown 500 milliseconds before the next index was sent to the filter function, the source of the layer was updated, hiding the current features and showing the features for the next time interval.

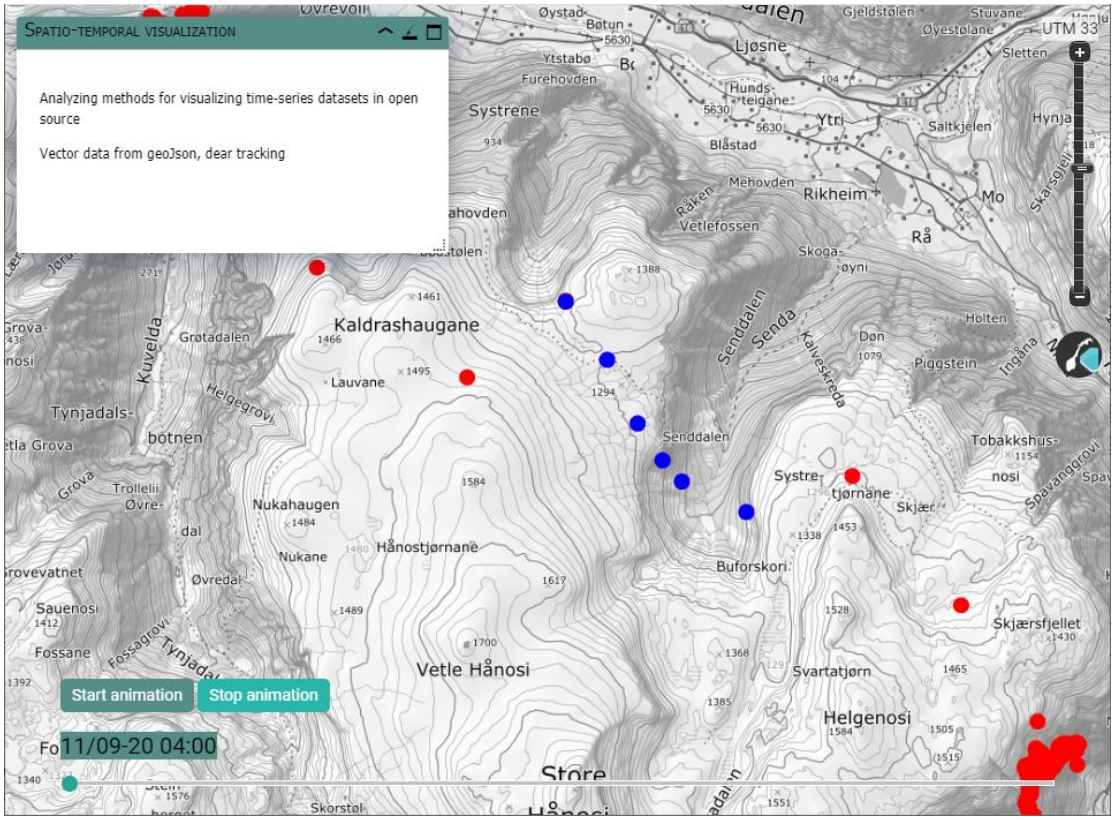

Figure 4: OpenLayers and GeoJSON application.

2. WMS

In this method, the data were saved as a shapefile in QGIS, and this file was moved to GeoServer. A new layer was made with the shapefile, and GeoServer provided many different output formats. The list of output formats supported by a GeoServer instance can be found with a WMS GetCapabilities request. Below is an example of a standard WMS request for one of the supported formats used in this study.

https://karttest.nibio.no/geoserver/master/wms?
service=WMS&
version=1.1.0&
request=GetMap&
layers=master%3Ahjort_1-30sept&
bbox=93749.41772619059%2C6783341.010973525%2C104093.4370224792%2C6791379.426336285&
width=768&
height=596&
srs=EPSG%3A25833&
styles=&
format=image%2Fpng

In the Web Map Services standard, it is possible to filter the data obtained from an input image. This can be done with an extra parameter called CQL_FILTER, and we can ask for a special time interval. An application was developed to show the filtered WMS images on the map. See Figure 5.

When the time slider moved, a new URL was sent with a new CQL_FILTER to obtain an image from GeoServer with the desired point in time.

The OpenLayers tutorial was set up with Parcel Bundler Version 1. However, to be able to ask for the WMS images from the local computer without cross-origin problems, Parcel Bundler Version 2 was used to proxy the requests to GeoServer from the local computer during development.

Example proxy settings.

```
{
  "/geoserver": {
    "target": "http://utvgeo01.ad.skogoglandskap.no/geoserver/",
    "pathRewrite": {
      "^/geoserver": ""
    }
  }
}
```

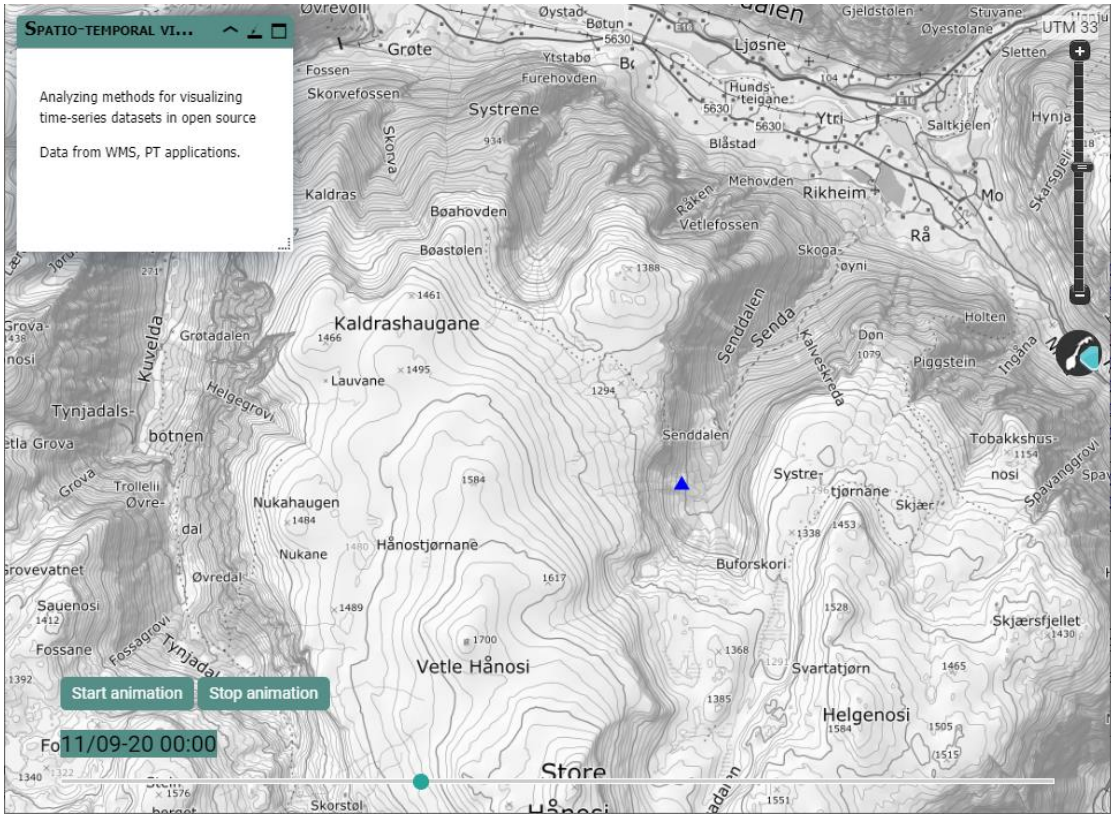

Figure 5: OpenLayers and WMS application with spatio-temporal data from deer collars.

3. D3

The D3 method was based on an example where TopoJSON was loaded and used the d3.geo.path from d3js to render geometries as an SVG element.
TopoJSON is an extension of GeoJSON that encodes topology. Rather than representing geometries discretely, geometries in TopoJSON files are stitched together from shared line segments called arcs. TopoJSON eliminates redundancy, offering much more compact representations of geometry than GeoJSON; typical TopoJSON files are 80% smaller than their GeoJSON equivalents. The loading of the data was expected to be faster when the size was smaller. First, the data had to be converted to TopoJSON, which was done with the geo2topo function. geo2topo is part of TopoJSON which you install as a global node package with “npm install -g topojson” and are used with the following command:

```
geo2topo pt=PTAll.geojson > pt.topojson
```

In the map application, a Canvas Layer was made, where the TopoJSON features were added. See Figure 6. To filter the points to show only the points for the actual time interval, the d3layerFilter was used. First, all points were set to opacities of 0. When the time slider moved, the points selected were set to opacities of 1:

```
d3layerFilter.svg.selectAll('path').filter(data => {
      return data.properties.Acquisition_time ===
ordered[sid].properties["Acquisition_time"];
    }).attr('fill', 'red').style('opacity', 1.0);
```

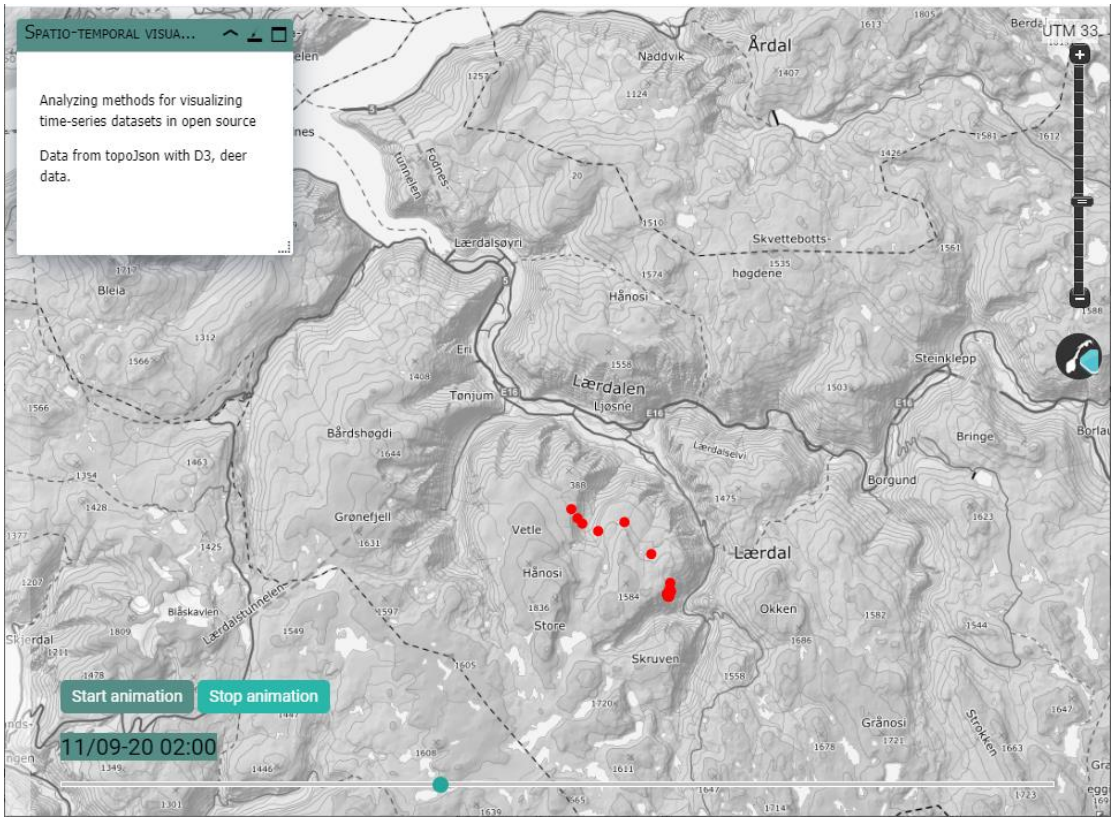

Figure 6: OpenLayers and D3 with spatio-temporal data from deer collars.

4. Cesium

Cesium is an open-source JavaScript library for creating 3D globes and maps with the good performance, precision and visual quality according to the Cesium website.

The data were loaded as GeoJSON, and the same filtering method as that in the GeoJSON technique was used. The features were read from the GeoJSON file in OpenLayers, and the features were added to a vectorSource and vectorLayer so they could appear in the map. See Figure 7.

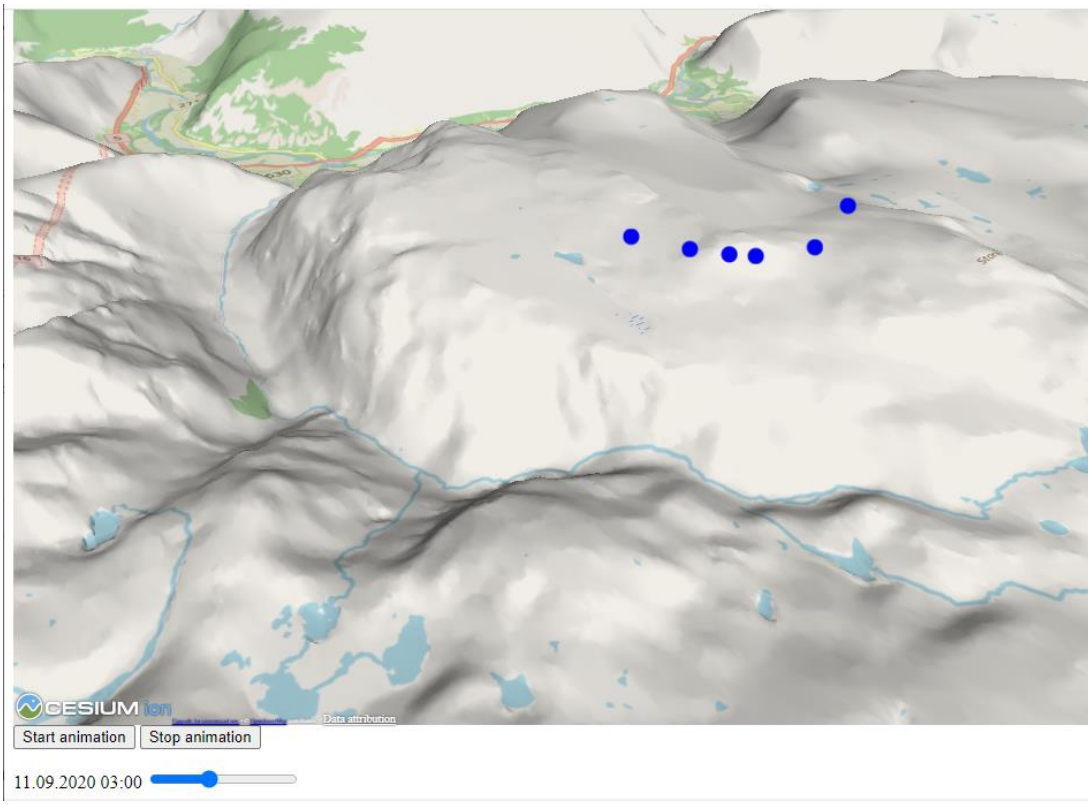

Figure 7: Cesium and spatio-temporal GeoJSON data from deer collars.

Time measuring

For all four technologies, it was important to measure the loading times and animation times to find differences. The window.performance object in the browser was used, and mark were set at the beginning of the html-file to start the timing and where we wanted to start and stop measuring.

An object with all the results was made each time the site was finished loading, and the objects looked like this:

```
  var newItem = {
      'Nr': loadNr,
      'mark-total': performance.measure('total', 'start',
'end').duration,
      'mark-load-data': performance.measure('load-data',
'load.data.start', load.data.end').duration,
      'draw-on map':performance.measure('draw-map', 'load.data.end',
'end').duration,
      'animation':performance.measure('animation', 'anim.start',
'anim.end').duration

  };
```

This object was added to an array in local storage. Then, a function with an input variable was made to decide how many times the site should be loaded before the array in local storage should be written to a csv-file. All methods were loaded 100 times each. All loading data for each method were downloaded as a CSV file to the

computer where the applications were run in the browser. The JavaScript function is called saveData:

```
var saveData = (function () {
var a = document.createElement("a");
document.body.appendChild(a);
a.style = "display: none";
return function (data, fileName) {
    var json = JSON.stringify(data),
        blob = new Blob([json], {type: 'octet/stream'}),
        url = window.URL.createObjectURL(blob);
    a.href = url;
    a.download = fileName;
    a.click();
    window.URL.revokeObjectURL(url);
};
}());
```

The CSV file had to be changed in TextPad so that Excel could open it. Commas had to be changed to semicolon, and /n had to be changed to line breaks.

The files for all four methods for a given dataset were added to a readable format, the data were copied to a spreadsheet, and an ANOVA test was run. See chapter 3.3.4. A result from the ANOVA was the average time required for all 100 measurements. From those average times, a chart was made showing the differences in the average times for each of the four metrics of the four methods and three datasets.

3.2.3 Data conversion

These datasets needed to be prepared in dissimilar ways for different presentation methods, and one task was to find an effective production trial to prepare the data for visualization. QGIS and geo2topo were used to convert the data between GeoJSON, Shape (ESRI's spatial data format) and TopJSON used in the different measurements.

3.3 Performance testing

By dividing the total time into the loading time and animation time, a distinction is made between what measures spatial data and spatio-temporal data. Performance tests were performed, and it was considered whether measurements should be made to compare static and dynamic data in other ways, but this idea was rejected as significant new information in relation to time use was provided. As mentioned earlier four datasets were used during the testing of the four different technologies.

The results were recorded using the Google Chrome console (Chrome DevTools). Caching was also switched off to ensure that all data were reloaded each time, and the "Preserve log" option was checked to record the data of all previous requests sent to the server during the interactions for each measurement. The console record was exported to the HAR (HTTP Archive) format after testing and then converted into a CSV file.

3.3.1 Data preparation time

Data was prepared for each of the technologies and the time used to prepare each dataset for each technology was measured manually using a normal clock. GeoJSON data was used both with the Cesium application and the GeoJSON application. QGIS was used to make GeoJSON from data in the PostGIS database. QGIS was also used to convert from PostGIS to shape files for WMS (delivered by GeoServer). TopoJSON was used in the D3 application. To convert GeoJSON to TopoJSON for D3, the geo2topo program was used in the following way.

```
geo2topo pt=PTAll.geojson > pt.topojson

geo2topo deer=orange_bukk_lærdal_jul_des.txt > deer_medium.topojson
```

The Cesium applications were executed using WGS 84/Pseudo-Mercator - Spherical Mercator EPSG:3857. The data had to be transformed to this format, and the export function in QGIS was used. This transformation was a part of the preparation time for all datasets for the Cesium map application. These measures can be considered as subjective and has therefore been discussed if it has any relevance.

3.3.2 Loading time

The loading time was split into four parts for each of the four applications developed. In the code there was placed timers (`javascript performance.marks`) for each part that was measured. Three different loading times were measured before the animation: (1) the time for loading the data, (2) the time required to display the first data on the map and (3) the total time for loading the whole application. The first part was from when the data started to load until the response from the server was delivered. The second part was from when the data were loaded to the point at which the first data were shown on the map. The third part was the total loading time of the site from the start of index.html until the start of the animation. The fourth was the time required to show 100 time intervals as an animation.
To obtain the loading times, the performance object in the browser window was used and the results was stored in the browsers local storage. After 100 loadings the data from local storage was written to a csv-file and downloaded to the local computer.

3.3.3 Efficiency

Efficiency was measured as 100 time intervals shown for the deer data, and for the production subsidies data, the large dataset did not provide any data for comparison since the browser crashed before the data was loaded. Two different browsers were tried but it crashed both in chrome and firefox.
During the animation, the map was zoomed in one level and then zoomed out two levels later in the animation. The animation times were also stored in the browsers local storage and written to a csv-file when finished with all the 100 time intervals. The performance test regarding efficiency was not expected to show large differences between the technologies because there was a javascript loop showing next point every 500 milliseconds to show next time interval.

3.3.4 Statistics

Analysis of variance (ANOVA), is a statistical method that separates observed variance data into different components to use for additional tests. One-way (or unidirectional) ANOVA is used for three or more groups of data to gain information about the relationships between the dependent and independent variables. The result of the ANOVA formula, the F-ratio, allows for the analysis of multiple groups of data to determine the variability between samples and within samples. If no real differences exist between the tested groups (the null hypothesis), the result of the ANOVA F-ratio statistics will be close to 1.
ANOVA tests on the data from the csv-files were performed in Excel to determine if there were significant differences between the technological methods. Four times were measured: the total loading time, the loading time for the data, the display time for the first time-interval, and the time required for the animation of 100 time-intervals. Graphs were made to compare the average loading times for each technology. See chapter 4 (Results) to see the results of the tests.

To see if there were significant differences between all methods the Bonferroni-corrected Multiple comparisons were used in all relevant results.

4 RESULTS

The results of this thesis are divided into three main sections: (1) data preparation time; (2) loading time; and (3) dynamic visualization efficiency. The results were gathered from the statistical analysis of the differences between the open-source techniques.

4.1 Loading time

All applications were loaded one hundred times, and three different loading times were measured: the time for loading the data, the time required to display the first data on the map and the total loading time for the whole application. The times were measured in milliseconds.

4.1.1 Deer data (small dataset)

The four technologies were compared via ANOVA single-factor analyses. Averages are preferable to totals because they represent the summarized data on the same scale as the individual values, which makes visual comparison much easier (Lindsay, 2011). All graphs are therefore based on average numbers from 100 runs.

Three different time measures were collected: load data times, display data time, and total load and display times. In addition, the animation efficiency was measured.

To interpret the single-factor ANOVA, the *P*-value is the probability that an *F* statistic would be more extreme (larger) than the *F*-crit shown in the table, assuming that the null hypothesis is true. When the *P-value* is larger than the significance level (.05), the null hypothesis is accepted; when it is smaller, it is rejected.

Table 2 shows the summary of the data loading times of the small dataset. The average column is showing.
Figure 8 (red bars in the graph) shows the average data loading times for the small dataset, where the values for Cesium and GeoJSON are quite similar, while those of D3 and WMS are much smaller.

Table 2: Summary for group of loading times for the small dataset in milliseconds.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	82.004	334.508
Geo	100	74.443	436.259
D3	100	4.773	1.948
WMS	100	9.466	9.433

Error! Reference source not found. shows the ANOVA results table for data loading with the small dataset. The *P*-value is smaller than .05, which means that there are significant differences between the methods.

Table 3: ANOVA result for loading times of small dataset. SS= sum of squares, df = Degrees of freedom, MS = mean squares, F=F-ratio, P-value = the area to the right of the F statistic and F-crit = alpha value.

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	509543.364	3	169847.788	868.620	9.803E-174	2.627
Within Groups	77432.824	396	195.537			
Total	586976.188	399				

Bonferroni-corrected Multiple comparisons were used (Table 4).

Table 4: Comparisons if there are significant differences in loading time in the small dataset

Compare	*P*-value	Alpha value:	Significant
Cesium vs Geo	.00704166	.008333	TRUE
Cesium vs D3	9.476E-101	.008333	TRUE
Cesium vs WMS	4.3233E-95	.008333	TRUE
Geo vs D3	4.8989E-83	.008333	TRUE
Geo vs WMS	2.1379E-77	.008333	TRUE
D3 vs WMS	3.949E-31	.008333	TRUE

As Table 4 shows there were statistically significant differences between all methods.

The times from the data being loaded to the first data point being displayed are shown in the map (Table 5). The ANOVA results are presented in Table 6. Figure 8 (green bars in the graph) shows the average display times compared. WMS was the fastest and GeoJSON was the slowest. The Cesium- and D3-methods had similar times, but the D3 method was slightly faster.

Table 5: Summary of display times for the small dataset in milliseconds.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	145	678.886
GeoJSON	100	200	1120.036
D3	100	130	326.696
WMS	100	75	161.232

Table 6: ANOVA table for the display times in the small dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	803742.127	3	267914.042	468.617	7.233E-130	2.627
Within Groups	226397.989	396	571.712			
Total	1030140.12	399				

Bonferroni-corrected Multiple comparisons were used (Table 7).

Table 7: Comparisons if there are significant differences in display time in the small dataset

Compare	*P*-value	Alpha value:	Significant
Cesium vs Geo	9.063E-20	.008333	TRUE
Cesium vs D3	2.088E-19	.008333	TRUE
Cesium vs WMS	4.025E-28	.008333	TRUE
Geo vs D3	3.609E-86	.008333	TRUE
Geo vs WMS	3.369E-103	.008333	TRUE
D3 vs WMS	2.475E-07	.008333	TRUE

As Table 7 shows there were statistically significant differences between all methods.

The total loading times for the small dataset were also measured, the summary of the times (Table 8) and the ANOVA results are presented in Table 9.

Table 8: Summary of total loading times for the small dataset in milliseconds.

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Cesium	100	53444	534	5073
GeoJSON	100	48402	484	3949
D3	100	26905	269	1855
WMS	100	10845	237	910

Based on the P-values, there are significant differences between the measures of the methods. Figure 8 (blue bars in the graph) shows the average total loading times and shows that the Cesium method used the most time to load the whole application, closely followed by the GeoJSON method. The D3 method was clearly faster than the previous two, and the WMS method had the fastest total loading time among the four methods for which time was measured.

Table 9: ANOVA table for the total loading times for the small dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	11687425.31	3	3895808.44	1322.155	7.207E-206	2.627
Within Groups	1166837.227	396	2946.55865			
Total	12854262.54	399				

Bonferroni-corrected Multiple comparisons were used (

Table 10). There are significant differences between all methods for the total loading times of the small dataset.

Table 10: Comparisons if there are significant differences in total time in the small dataset

Compare	*P*-value	Alpha value:	Significant
Cesium vs Geo	2.26444E-28	.008333	TRUE
Cesium vs D3	3.95801E-06	.008333	TRUE
Cesium vs WMS	1.41568E-61	.008333	TRUE
Geo vs D3	6.26281E-45	.008333	TRUE
Geo vs WMS	4.25658E-87	.008333	TRUE
D3 vs WMS	1.1113E-63	.008333	TRUE

As

Table 10 shows there were statistically significant differences between all methods.

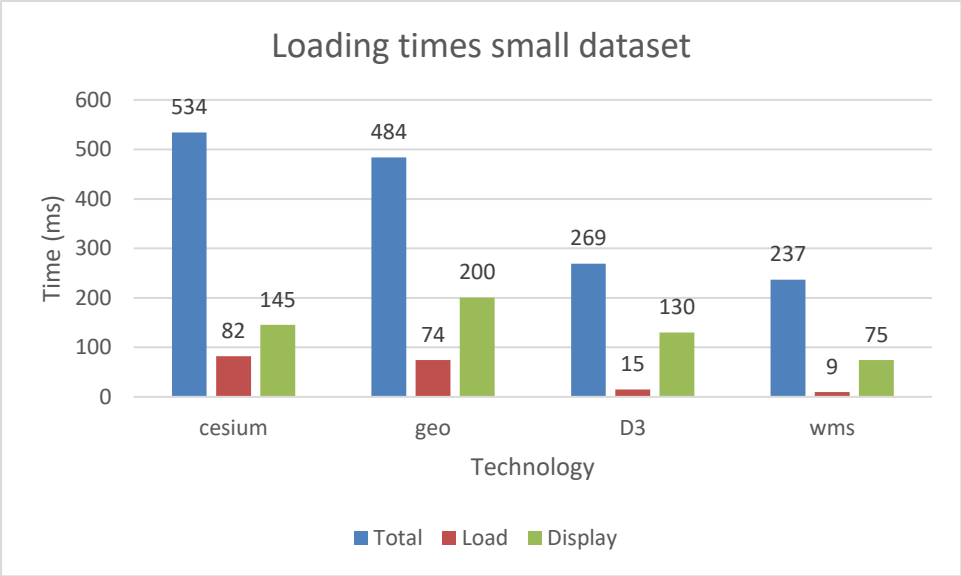

Figure 8: Comparing all average data for small dataset in milliseconds.

4.1.2 Deer data (medium dataset)

Table 11 presents the summary of the loading data for the medium dataset.

Table 11: Summary data loading times for the medium dataset in milliseconds.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	179	334
GeoJSON	100	174	381
D3	100	22	21
WMS	100	20	25

Table 12 presents the ANOVA results table for the data loading measures for the medium dataset. The null hypothesis was that all methods used the same amount of time. The tests show significant differences between the measures of the four methods

(P<0.01). Figure 9 (red bars in the graph) shows the average loading times for the methods tested. The Cesium and GeoJSON methods used more than 20 times the timespans of D3 and WMS.

Table 12: ANOVA table for loading times for the medium dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	2421601.27	3	807200.425	4245.041	1.303E-300	2.627
Within Groups	75299.9504	396	190.151			
Total	2496901.23	399				

Bonferroni-corrected Multiple comparisons were used (Table 13). There are not significant differences between Cesium and GeoJSON or D3 vs WMS, all other methods for the total loading times of the medium dataset had significant differences.

Table 13: Comparisons if there are significant differences in loading time in the medium dataset

Compare	*P*-value	Alpha value:	Significant
Cesium vs GeoJSON	0.055	0.008333	FALSE
Cesium vs D3	2.196E-156	0.008333	TRUE
Cesium vs WMS	9.852E-157	0.008333	TRUE
GeoJSON vs D3	1.929E-146	0.008333	TRUE
GeoJSON vs WMS	9.095E-149	0.008333	TRUE
D3 vs WMS	0.033	0.008333	FALSE

As Table 13 shows that the difference between Cesium and GeoJSON was not significant, the same between D3 and WMS, no significant difference. The rest showed statistically significant differences between the methods.

Table 14 presents the summary of the display times for the medium dataset.

Table 14: Summary regarding the display times after loading the medium dataset in milliseconds.

SUMMARY

Groups	Count	Average	Variance
Cesium	100	187	2996
Geo	100	250	726
D3	100	127	481
WMS	100	113	234

Table 15 presents the ANOVA results regarding the display times after loading the data. The *P*-values were well under 0.05, which shows that there are significant differences between the tested methods. Figure 9 (green bars in the graph) shows a graph with the average display times, and D3 and WMS had the best display times.

WMS was slightly better than the D3 method, with GeoJSON using approximately double the times of those two methods and Cesium between them.

Table 15: ANOVA table display times for the medium dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Grps	1177259	3	392420	353.636	1.31E-111	2.627
Within Grps	439429	396	1110			
Total	1616688	399				

Bonferroni-corrected Multiple comparisons were used (Table 16). There are significant differences between all methods for the display times of the medium dataset.

Table 16: Comparisons if there are significant differences in display time in the medium dataset

Compare	*P*-value	Alpha value:	Significant
Cesium vs Geo	9.063E-20	0.008333	TRUE
Cesium vs D3	2.088E-19	0.008333	TRUE
Cesium vs WMS	4.025E-28	0.008333	TRUE
Geo vs D3	3.609E-86	0.008333	TRUE
Geo vs WMS	3.369E-103	0.008333	TRUE
D3 vs WMS	2.475E-07	0.008333	TRUE

Table 17 presents the summary of the total loading times for the medium dataset.

Table 17: Average total loading times for the medium dataset in milliseconds.

SUMMARY

Groups	Count	Average	Variance
Cesium	100	680	15291
Geo	100	662	2304
D3	100	425	1991
WMS	100	330	3327

Table 18 shows the ANOVA results regarding the total load times for the medium dataset. The tests show significant differences between the measures of the four methods ($p<0.01$). Figure 9 (blue bars in the graph) shows the average total loading times for the medium dataset. Additionally, here, the WMS method was the fastest in terms of the metrics of the four approaches.

Table 18: ANOVA table total loading time for medium dataset

ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	9095174	3	3031725	529.233	3.953E-138	2.627
Within Groups	2268494	396	5729			
Total	11363668	399				

Bonferroni-corrected Multiple comparisons were used (Table 19). There are not significant differences between Cesium vs GeoJSON or D3 vs WMS, all other comparisons between methods for the total loading times of the medium dataset had significant differences.

Table 19: Comparisons if there are significant differences in total time in the medium dataset

Compare	*P-value*	Alpha value:	Different
Cesium vs Geo	0.059	0.008333	FALSE
Cesium vs D3	1.971E-154	0.008333	TRUE
Cesium vs WMS	8.242E-155	0.008333	TRUE
Geo vs D3	1.93E-146	0.008333	TRUE
Geo vs WMS	6.957E-147	0.008333	TRUE
D3 vs WMS	0.033	0.008333	FALSE

Table 19 shows that the difference between Cesium and GeoJSON was not significant, the same between D3 and WMS, no significant difference. The rest showed statistically significant differences between the methods.

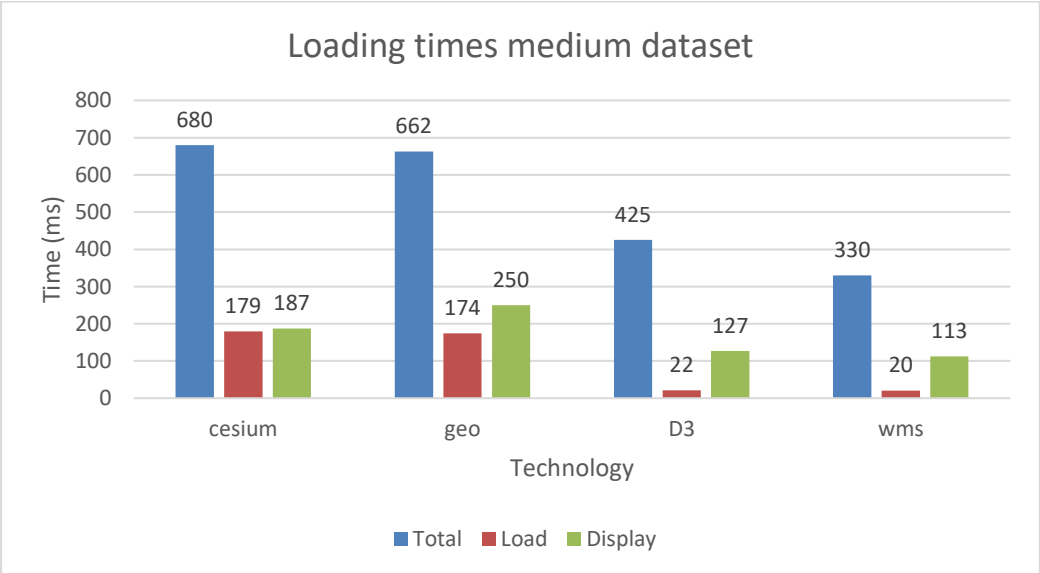

Figure 9: Comparing all average data for medium dataset in milliseconds.

4.1.3 Production subsidies data (medium-large dataset)

The SUMMARY results regarding the data loading times are presented in Table 20 . The tests show significant differences between the measures of the four methods ($P<0.01$). Figure 10 (red bars) shows the average load times for the four methods, and as expected, the numbers are much higher for this larger dataset. Additionally, the WMS came out as method with the shortest times. The Cesium method had the highest measures, with the two others in between.

Table 20: Summary of load times for the medium-large dataset in milliseconds.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	627	4086.713
GeoJSON	100	609	4670.512
D3	100	76	258.294
WMS	100	71	301.898

The ANOVA results regarding the data loading times are presented in Table 21.

Table 21: ANOVA table for load times for the medium-large dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	29664615.6	3	9888205.2	4245.041	1.303E-300	2.627
Within Groups	922424.392	396	2329.35453			
Total	30587040	399				

Bonferroni-corrected Multiple comparisons were used (Table 22). There are not significant differences between Cesium vs GeoJSON , all other comparisons between methods for the loading times of the medium-large dataset had significant differences.

Table 22: Comparisons if there are significant differences in total time in the medium-large dataset in milliseconds.

Compare	P-value	Alpha value:	Different
Cesium vs GeoJSON	0.178	0.008333	FALSE
Cesium vs D3	1.05888E-47	0.008333	TRUE
Cesium vs WMS	7.16855E-65	0.008333	TRUE
GeoJSON vs D3	2.61977E-89	0.008333	TRUE
GeoJSON vs WMS	1.1211E-104	0.008333	TRUE
D3 vs WMS	2.03726E-87	0.008333	TRUE

Summary of the display data for the medium large dataset is presented in Table 23.

Table 23: Summary of display times for the medium-large dataset in milliseconds.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	654	36703.1606
GeoJSON	100	874	8902.65304
D3	100	444	5899.13216
WMS	100	394	2868.83938

Table 24 presents the ANOVA display times after loading the data for the medium-large dataset. The P-values show that there were significant differences between the methods. Figure 10 (green bars) shows a graph with the average display times for the medium-large dataset. The WMS method displayed the data fastest, followed by D3, GeoJSON and Cesium.

Table 24: ANOVA results for display times for the medium large dataset.

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	14421422.3	3	4807140.76	353.636	1.31E-111	2.627
Within Groups	5383004.73	396	13593.44			
Total	19804427	399				

Summary of the total load data for the medium large dataset is presented in Table 25.

Table 25: Summary of total load times for the medium-large dataset in milliseconds.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	654	36703.160
GeoJSON	100	874	8902.653
D3	100	444	5899.132
WMS	100	394	2868.839

Table 26 presents the ANOVA results regarding the total load times for the medium-large dataset. The P-value of 0 shows that there were significant differences between the methods. Figure 10 (blue bars) shows a graph with the average load times for the medium-large dataset. The WMS method had the smallest loading time, followed by D3, GeoJSON and Cesium.

Table 26: ANOVA table for the total loading times for medium large dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	14421422.3	3	4807140.76	353.636	1.31E-111	2.627
Within Groups	5383004.73	396	13593.4463			
Total	19804427	399				

Bonferroni-corrected Multiple comparisons were used (Table 27). There are not significant differences between Cesium vs GeoJSON, all other comparisons between methods for the total loading times of the medium-large dataset had significant differences.

Table 27: Comparisons if there are significant differences in total loading time in the medium-large dataset

Compare	P-value	Alpha value:	Different
Cesium vs Geo	0.17808466	0.008333	FALSE
Cesium vs D3	1.0589E-47	0.008333	TRUE
Cesium vs WMS	7.1685E-65	0.008333	TRUE
Geo vs D3	2.6198E-89	0.008333	TRUE
Geo vs WMS	1.121E-104	0.008333	TRUE
D3 vs WMS	2.0373E-87	0.008333	TRUE

Figure 10 shows all the times for the medium large dataset in one graph.

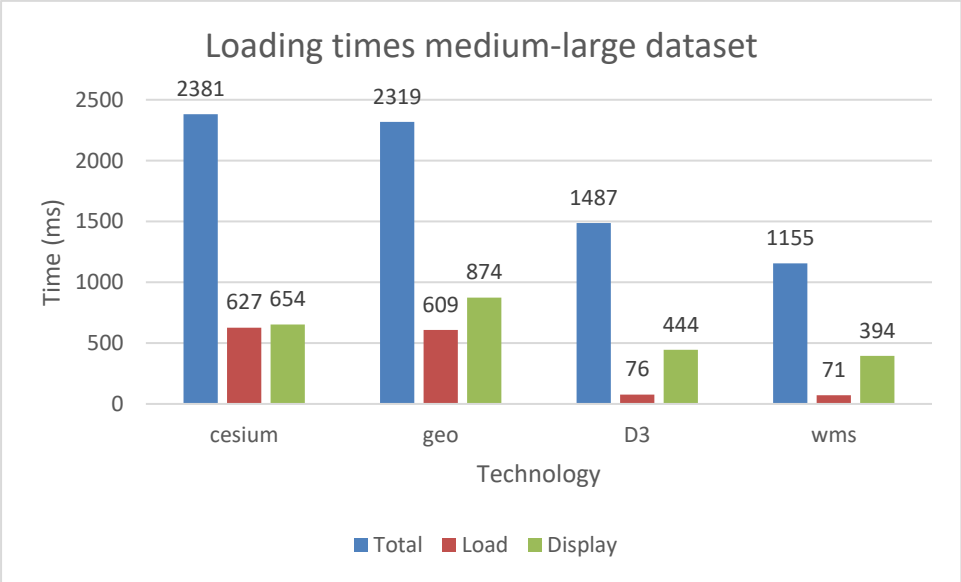

Figure 10: Comparing all average data for medium-large dataset in milliseconds.

4.1.4 Production subsidies (large dataset)

The large dataset was only sometimes displayed on the map, and we were not able to load it before the browser crashed. A value of 0 was used if the browser crashed as a measure denoting that there were problems with the viewing process.

Summary of the total load data for the large dataset is presented in Table 28.

Table 28: Average data loading times for the large dataset in milliseconds.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	0	0
GeoJSON	100	12544	1834889.65
D3	100	106	7.79
WMS	100	103	1.57

The ANOVA results regarding the data loading times for the large dataset are presented in Table 29. The tests show significant differences between the measures of the four methods ($P<0.01$). Figure 11 (red bars) shows the average loading times for the large dataset. WMS and D3 had very small average loading times, while the GeoJSON method used over 12 seconds, and the Cesium method crashed the browser.

Table 29: ANOVA table for loading times for the large dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	11672573516	3	3890857839	8481.901	0	2.627
Within Groups	181655002.8	396	458724.754			
Total	11854228518	399				

The display times for the large dataset are presented in Table 30. 0 means that the data could not be displayed. The WMS method was the only method that did not crash the browser. Figure 11 (green bars) shows the average display times for the large dataset.

Table 30: Summary of display times for the large dataset in milliseconds.

SUMMARY

Groups	Count	Average	Variance
Cesium	100	0	0
Geo	100	0	0
D3	100	0	0
WMS	100	43	121

Table 31 presents the ANOVA results regarding the display times for the large dataset. The *P*-values show that there were significant differences between the methods.

Table 31: ANOVA table for display times for the large dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	136838	3	45613	1507.641	3.442E-216	2.627
Within Groups	11981	396	30			
Total	148818	399				

Figure 11 (blue bars) shows the average total loading times for the large dataset. The WMS method had the smallest total loading time. D3 had slightly longer time requirement, GeoJSON struggled for over 13 seconds, and Cesium crashed the browser during the measurements.

Table 32 shows the Summary times for the total loading of the large dataset.

Table 32: Summary of total loading times for the large dataset in milliseconds.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	0	0
GeoJSON	100	13235	3365170.1
D3	100	328	3202.3
WMS	100	236	488.0

Table 33 shows the ANOVA results for the total loading of the large dataset.

Table 33: ANOVA table for total loading times for the large dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	12772082075	3	4257360692	5054.956	0	2.627
Within Groups	333517187.5	396	842215.12			
Total	13105599263	399				

Table 34: Comparisons if there are significant differences in total loading time in the large dataset

Compare	P-value	Alpha value	Different
Geo vs D3	5.211E-142	0.008333	TRUE
Geo vs WMS	1.249E-142	0.008333	TRUE
D3 vs WMS	7.380E-35	0.008333	TRUE

As Table 34 shows that the difference between the methods that managed loading. The differences were statistically significant between those methods.

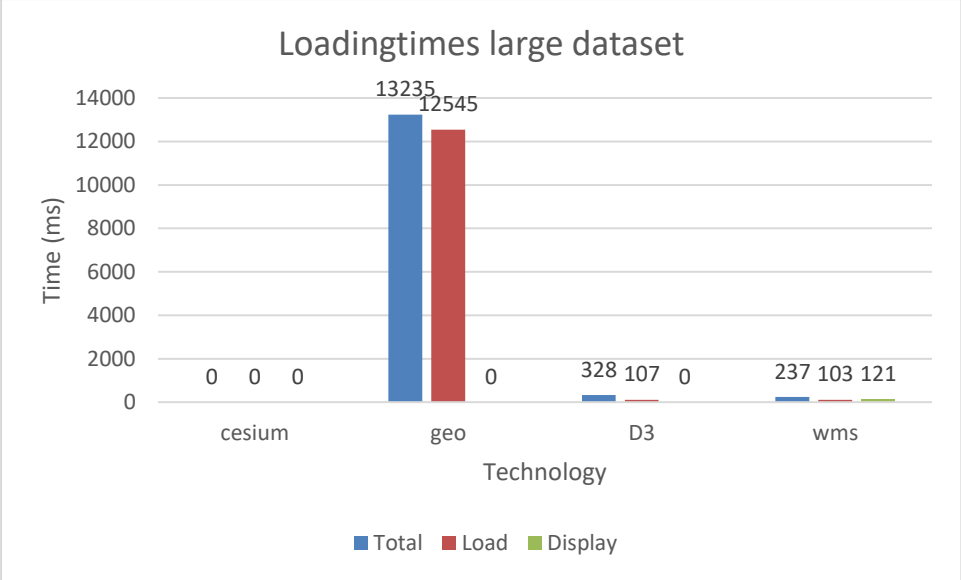

Figure 11: Comparing all average data for large dataset in milliseconds.

To sum up and visualize the results for total load time for the datasets and methods are showed in Table 35. Here the colour green and number 1 shows the fastest total loading the next were yellow and number 2 and so on. WMS was the best for all of the datasets and methods, D3 was the second best, GeoJSON was the third and Cesium used most time when it managed to load the data.

Table 35: Summary, 1 is best loading time and 4 is worst loading time. 0 is not loading.

Dataset	GeoJSON	D3	WMS	Cesium
Deer data (small)	3	2	1	4
Deer data (medium)	3	2	1	4
Prod. Subsidies (medium large)	3	2	1	4
Prod. subsidies (large)	3	2	1	0

4.2 Efficiency

After the site was loaded, an animation was started immediately, and the times it took to show 100 points of the small, medium and medium-large datasets were measured. Example URLs can be found in Table 36. The large dataset crashed when loading the data except when the WMS method was used.

Table 36: Example URLs of the different applications.

Technology	Dataset	URL
GeoJSON	Small	https://karttest.nibio.no/timemap/geoJson/
	Medium	https://karttest.nibio.no/timemap/geoJson_large/
WMS	Small	https://karttest.nibio.no/timemap/wms_deer/
	Medium	https://karttest.nibio.no/timemap/wms_deer_middle/
	Large	https://karttest.nibio.no/timemap/wms_PS/
D3/ TopoJSON	Small	https://karttest.nibio.no/timemap/md3/
	Medium	https://karttest.nibio.no/timemap/md3_medium/
Cesium	Small	https://karttest.nibio.no/timemap/cesium/
	Medium	https://karttest.nibio.no/timemap/cesium_medium/

The animation time results for the small dataset are presented in Table 37 and Table 38, the *P*-values here are greater than 0.05 (0.256), which tells us that there were no significant differences between the methods. Figure 12 is a graph with the average animation times that shows the small differences, with the GeoJSON method performing best.

Table 37: Summary of animation times for the small dataset.

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Cesium	100	4958205.7	49582	85712615
GeoJSON	100	4751981.6	47520	51.13439
D3	100	4953968.7	49540	96727731
WMS	100	4943869.2	49439	1.14E+08

Table 38: ANOVA table of animation times for the small dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	3.01E+08	3	100394567	1.355	0.256	2.627
Within Groups	2.93E+10	396	74098435			
Total	2.96E+10	399				

Since the *P-value* shows no significant differences (*P* =0,25) it is not required to do Bonferroni-corrected Multiple comparisons.

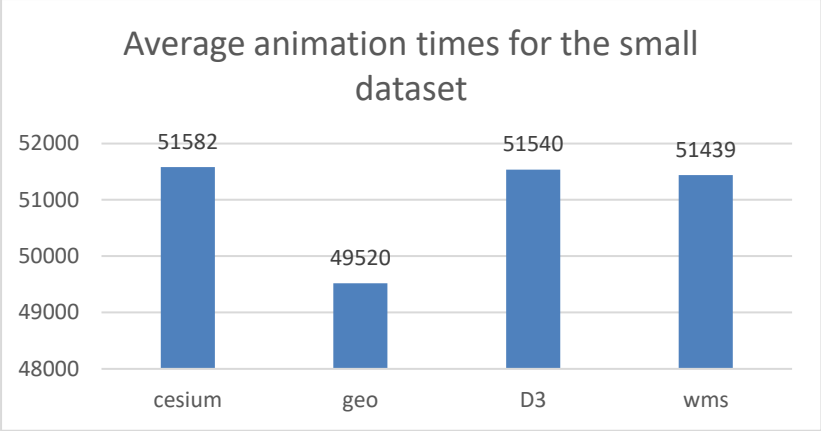

Figure 12: Average animation times for the small dataset.

The results regarding the animation times for the medium dataset are presented in Table 39 and Table 40. The P-values are greater than 0.05 (0.449), which tells us that there were no significant differences between the methods. Figure 13 is a graph showing the small differences, where GeoJSON and WMS have the smallest and second smallest metrics, respectively.

Table 39: Summary of animation times for the medium dataset.

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	50665	49420732
GeoJSON	100	49520	51
D3	100	50806	55554009
WMS	100	50141	49199297

Table 40: ANOVA table of animation times for the medium dataset.

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	102171574	3	34057191	0.883	0.449	2.627
Within Groups	15263234890	396	38543522			
Total	15365406464	399				

Since the *P-value* shows no significant differences (0,449) the Bonferroni-corrected Multiple comparisons are not requiered.

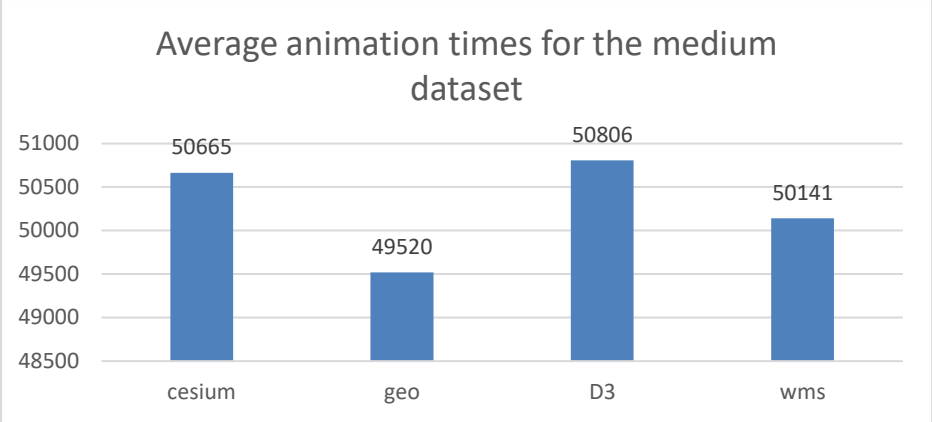

Figure 13: Average animation times for the medium dataset.

Medium large dataset

The results regarding the animation times for the medium-large dataset are presented in Table 41 and Table 42. The P-values are greater than 0.05 (0.449), which tells us that there were no significant differences between the methods. Figure 13 is a graph showing the small differences, where GeoJSON and WMS have the smallest and second smallest metrics, respectively.

Table 41: SUMMARY of animation times for medium large dataset

SUMMARY

Groups	*Count*	*Average*	*Variance*
Cesium	100	52665	49420732
GeoJSON	100	51520	51.13439
D3	100	52806	55554009
WMS	100	52141	49199297

Table 42: ANOVA table for the animation times for the medium large dataset

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.02E+08	3	34057191.2	0.883	0.449	2.627
Within Groups	1.53E+10	396	38543522.4			
Total	1.54E+10	399				

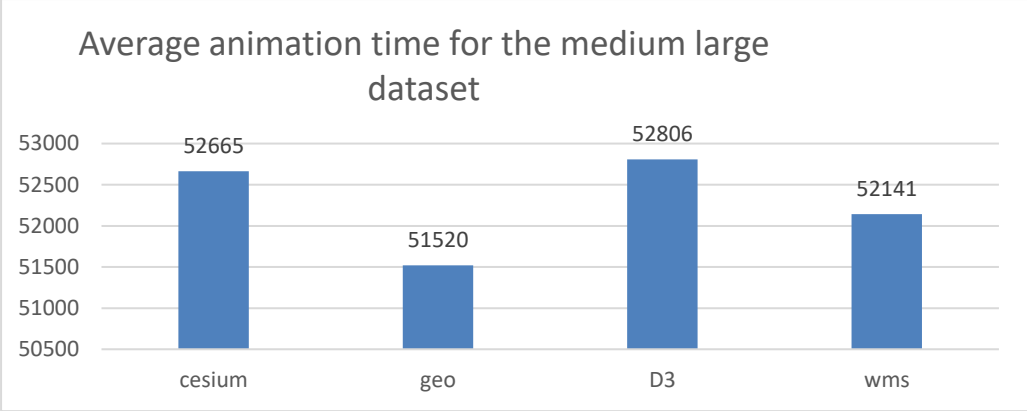

Figure 14: Average animation times for the medium large dataset.

Since the *P-value* shows no significant differences (0,449) are the Bonferroni-corrected Multiple comparisons not neededv.
For the large data set, only WMS was measured, and there was nothing to compare because the browser crashed when the other methods were used, as explained earlier.

4.3 Data preparation time

Table 43 presents the preparation times for the datasets. All measures are in units of minutes measured by the clock on the computer. There were in general no large differences in data preparation time between the methods.

Table 43: Preparation times for different datasets and methods (in minutes).

Dataset	GeoJSON	D3 (TopoJSON)	WMS (Shape)	Cesium (GeoJSON)
Deer data (small)	25	25	30	35
Deer data (medium)	30	30	35	40
Prod. Subsidies (medium-large)	45	45	50	50
Prod. subsidies (large)	60	65	75	60

The work of preparing the data was performed only once for each method and dataset, and the time did not vary much between the methods in terms of the development time, which was approximately three weeks for each method varying with the experience in each technique. There were some differences in preparation time, but the development time for each method was 2-3 weeks or 80-120 hours. The development time were not evaluated in this thesis. The average time was set to 100 hours multiplied by 60 to get the minutes getting 6000 minutes. The greatest difference was between GeoJSON and WMS and was only 15 minutes. 15 of 6000 is 0,0025 or 0,025%. The p-value in statistical significance is 5% or 0,05. A 15-minute difference of 6000 cannot be considered statistically significant. There should have been more objective and formal tests in preparation times if these results were to be emphasized.

5 DISCUSSION

This section discusses the results and the limitations, suggesting future work, and summarize the main conclusions. This thesis aims to answer the following main research question: “What open-source-based methods are most optimal for visualizing spatio-temporal data with regards to preparation, loading time, and efficiency?”

5.1 Main results

The results showing the greatest differences were the measures of the loading times. The loading times were split into the time required for loading data, the time for displaying the first data and the total loading time. All the figures showing average time measure graphs in the loading time section of the results chapter showed that the WMS method was the fastest. The ANOVA tests showed significant differences between the technological methods. For the small datasets, all technologies handled the visualizations without problems, but for the larger datasets, the browser had trouble keeping up with the vector solutions; however, for WMS, the browsers handled the heavy work of visualising the map data. There are large differences in performance between the open-source technologies tried in this thesis, for small datasets there is small differences in performance, but for larger datasets the WMS and the D3 technologies have better performance. It is crucial to select the right open-source technology in order to save time and increase the usability especially with the larger datasets. The results in this thesis indicate that WMS was the fastest for data loading and had the smallest display time and total loading time, including all JavaScript and HTML, for all four datasets. Cesium, GeoJSON and D3/TopoJSON were also usable for the small datasets but failed when used with the larger datasets. The following paragraphs discuss the results from each test in more detail.

Regarding the aim of measuring efficiency, the time intervals required for an animation were measured to determine if there were differences between the methods. No significant differences were found with regards to the animations for all datasets. The results in this thesis indicate that WMS was the only technology that was able to display all datasets. For the two small datasets, the GeoJSON technology had a bit faster times in the animation tests, but the time differences were so small that there were no significant differences in the animation efficiency measures.

Preparation time for the data to be visualized was one of the things that this work was having a brief look into. To figure out if there were one of the methods that required much more time than the others a clock was used to measure the time of the process. The preparation time measurement was subjective and not a formal test. The preparation time was manually measured and could be subjective. This is not given much weight.

In all loading time performance tests, there were statistically significant differences between the methods. The differences were tested with T-test with Bonferroni correction. WMS was the fastest method in all the measurements that were carried out. D3 followed as second best and was the best vector-based method. GeoJSON and Cesium made the browser crash with the large dataset, the memory usage was built up and caused problems for the browser.

Efficiency of the animation that shows the time series data was the third main aim. The two smallest datasets were measured, and the results showed there were no statistically significant differences. Efficiency was not measured for the medium-large dataset because no large differences were expected, and this dataset was added after the large dataset crashed to get loading times.
There has also been comparison of GeoJSON and TopoJSON in the study by Shang (2015), but their study discussed the vector tile based Web mapping system. In the present study, more unsimilar technologies have been compared, which can be more challenging in terms of, for example, background maps or for the purpose of comparing raster and vector data sources.
As described other studies evaluate other types of data/technologies, but no one have been found comparing the use of GeoJSON, WMS and TopoJSON / D3. The research work that is the basis for this thesis can be described as ground-breaking work that can form the basis for further work in this field. It should therefore be considered whether this work can be elucidated from different angles. The results are the contribution that has been made to bring research in this area further.

5.2 Limitations, chosen methods and critical analysis

The measurements based on loading time is valid if the methods in this thesis were reliable. The WMS technology measured the whole process of asking for new images and showing them in the map.

Cesium, as one of the tested methods, has 3D maps as background layers and it can be assumed that it takes longer with 3D than with 2D background maps. The background maps would only affect the total load time and not the data loading time and time to show the first map. The results indicates that the total load time are different from the other load times. In figures showing the times required to load the first map image are faster for Cesium than for GeoJSON for all the datasets. The data loading times were similar. This was expected since both Cesium and GeoJSON loaded the data as GeoJSON. Additionally, for the efficiency results, there were too small differences that they had significance, as for example Table 38 showed.

The choice of using OpenLayers as the mapping library was decided from a combination of NIBIO's experience with both OpenLayers and Leaflet, from the list of best javascript libraries, and they consider OpenLayers to be the most complete mapping library. There was therefore little doubt that OpenLayers should be used in this study.

Using the Apache/Tomcat /GeoServer framework was natural for this work since this is the choice my company NIBIO have made. The choice is also supported by Agrawal and Gupta (2014) where they did a comparison of Web GIS frameworks and it was clear that the framework based on Apache Tomcat was more preferable.

5.3 Future work

There are three things that should be considered to change in this thesis if it were to be repeated. The first is to use only one type of data to obtain more similar measures, for example, only deer data but with more points. This work choose to use two different datasets to acquire more variety but there is now an understanding that the tests would be more unambiguous with more similar data. The second aspect would be to find another way to take the measurements, especially for WMS, to ensure that the loading of the images is included in the measurements. Currently, it is a questionable regarding whether the measurement process was stopped after sending the call for the WMS image and not after the image was loaded. The third change would be to use the same measures for all data formats with Cesium as the background maps. The data would be split in two, with WMS as the background data in one half and with Cesium as the background data in the second half. The two partitions could test GeoJSON, D3 and WMS as the present thesis has done.

Cesium has now come out with a new version that supports time-dynamic visualization and 4D, and it would be interesting to try out this method and compare it with this work.

In future research, it also would be better to determine the limit regarding how large vector data OpenLayers can handle. Vector tiles may be another data format worth looking at in another study to see if they can solve some of the problems we had with the large datasets.
An improvement for the time series applications could be to show more than one deer in the solution and display how the animals move in relation to each other.

6 CONCLUSIONS

The study of the differences between various methods used to visualize spatio-temporal data with open-source web mapping technology provides many new insights in different fields in terms of the geographical aspects of preparing datasets and in the field of web application development and performance measurement of JavaScript single-page web applications.

The aim of this thesis was to analyse the differences between the methods used to visualize spatio-temporal data with open-source web mapping technology. This aim consisted of one main objective to statistically analyse the differences between the open-source techniques with regard to the following:

A) Loading time performance in the client
B) Efficiency of the animation that shows the time series data
C) Preparation time for the data to be visualized

The following main conclusions can be drawn from this thesis:

A) Performance time
   The conclusion is that WMS was the fastest method for data loading and had the smallest display times and total loading times, including all JavaScript and HTML, for all four datasets.
   Cesium, GeoJSON and D3/TopoJSON were also satisfactory with the small datasets but failed when tested on the larger datasets.
B) Efficiency:
   The conclusion is that WMS was the only technology that was able to display all datasets. For the two small datasets, the GeoJSON technology was the fastest, but the differences in time were so small that there were no significant differences in the animation efficiency measures.
C) Preparation time:
   The conclusion is that the preparation time does not have much significance regarding which technology is preferred since this time is such a small part of the whole development. The method was also highly subjective and should have been done in another way.

7 REFERENCES

AGRAWAL, S. & GUPTA, R. D. 2014. Development and comparison of open source based Web GIS Frameworks on WAMP and Apache Tomcat Web Servers. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences,* 40**,** 1.

ANDRIENKO, N., ANDRIENKO, G. & GATALSKY, P. 2003. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing,* 14**,** 503-541.

BALLATORE, A., TAHIR, A., MCARDLE, G. & BERTOLOTTO, M. 2011. A comparison of open source geospatial technologies for web mapping. *International Journal of Web Engineering and Technology,* 6**,** 354.

BING, H., WEN-XIONG, M., JIN-XING, H., GE, Y., GUO-JUN, L. & LIU, Y.-Q. Development of power grid Web3D GIS based on Cesium. 2016 2016. IEEE.

BRUNSDON, C., CORCORAN, J. & HIGGS, G. 2007. Visualising space and time in crime patterns: A comparison of methods. *Computers, Environment and Urban Systems,* 31**,** 52-75.

CHRISTOPHE, S. Geovisualization: Multidimensional Exploration of the Territory. VISIGRAPP (3: IVAPP), 2020. 325-332.

CORBETT, J. 2012. Charles Joseph Minard: Mapping Napoleon's March, 1861.

DIBIASE, D., MACEACHREN, A. M., KRYGIER, J. B. & REEVES, C. 1992. Animation and the Role of Map Design in Scientific Visualization. *Cartography and Geographic Information Systems,* 19**,** 201-214.

FARKAS, G. 2017. Applicability of open-source web mapping libraries for building massive Web GIS clients. *Journal of Geographical Systems,* 19**,** 273-295.

FERREIRA, J. 2018. A JavaScript Framework Comparison Based on Benchmarking Software Metrics and Environment Configuration.

GÓMEZ, C., WHITE, J. C. & WULDER, M. A. 2016. Optical remotely sensed time series data for land cover classification: A review. *ISPRS Journal of Photogrammetry and Remote Sensing,* 116**,** 55-72.

GRAHAM-CUMMING, J., GALLONI, A. & STOCK, T. 2017. Method and apparatus for reducing loading time of web pages. Google Patents.

HAN, K. 2018. *Online visualization of multi-dimensional spatio-temporal data.* Master Thesis at TU München.

HEJMANOWSKA, B., GLOWIENKA, E., MICHALOWSKA, K., MIKRUT, S., KRAMARCZYK, P. & OPALINSKI , P. 2019. The Comparison of the Web GIS Applications Relevant for 4D Models Sharing. *World Multidisciplinary Earth Sciences Symposium.* Prague.

JAGOSH, J., MACAULAY, A. C., PLUYE, P., SALSBERG, J., BUSH, P. L., HENDERSON, J., SIRETT, E., WONG, G., CARGO, M. & HERBERT, C. P. 2012. Uncovering the benefits of participatory research: implications of a realist review for health research and practice. *The Milbank Quarterly,* 90**,** 311-346.

KIM, M.-S. & JANG, I.-S. 2019. Dynamic Mashup Performance Comparison Using Open Application Programming Interface of Geoweb Map Platforms Available in Korea. *Sensors and Materials,* 31**,** 3229-3244.

KRÓL, K. 2018. COMPARATIVE ANALYSIS OF THE PERFORMANCE

OF SELECTED RASTER MAP VIEWERS. *Geomatics, Landmanagement and Landscape,* 2**,** 23-32.

LAFRANCE, F., DANIEL, S. & DRAGIĆEVIĆ, S. 2019. Multidimensional Web GIS Approach for Citizen Participation on Urban Evolution. *ISPRS International Journal of Geo-Information,* 8**,** 253.

LI, W. & WANG, S. 2017. PolarGlobe: A web-wide virtual globe system for visualizing multidimensional, time-varying, big climate data. *International Journal of Geographical Information Science,* 31**,** 1562-1582.
LINDSAY, D. 2011. *Scientific Writing = Thinking in Words*, CSIRO PUBLICHING.
LIU, P., GONG, J. & YU, M. 2015. Visualizing and analyzing dynamic meteorological data with virtual globes: A case study of tropical cyclones. 64**,** 80-93.
LOECHEL, A. & SCHMID, S. 2013. Comparison of different caching techniques for high-performance web map services. *International journal of spatial data infrastructures research,* 8**,** 43-73.
MANHAS, J. 2013. A study of factors affecting websites page loading speed for efficient web performance. *International Journal of Computer Sciences and Engineering,* 1**,** 32-35.
MASSE, A. & CHRISTOPHE, S. Geovisualisation of animated tides in coastal area with an OpenSource OpenGL platform. 2016.
MURSHED, S., AL-HYARI, A., WENDEL, J. & ANSART, L. 2018. Design and Implementation of a 4D Web Application for Analytical Visualization of Smart City Applications. *ISPRS International Journal of Geo-Information,* 7**,** 276.
MWALONGO, F., KRONE, M., REINA, G. & ERTL, T. 2016. State-of-the-Art Report in Web-based Visualization. *Computer Graphics Forum,* 35**,** 553-575.
NETEK, BRUS & TOMECKA 2019. Performance Testing on Marker Clustering and Heatmap Visualization Techniques: A Comparative Study on JavaScript Mapping Libraries. *ISPRS International Journal of Geo-Information,* 8**,** 348.
NETEK, R., MASOPUST, J., PAVLICEK, F. & PECHANEC, V. 2020. Performance Testing on Vector vs. Raster Map Tiles—Comparative Study on Load Metrics. *ISPRS International Journal of Geo-Information,* 9**,** 101.
NÄGELE, T., HOOMAN, J., ZIGTERMAN, R. & BRUL, M. 2015. Client-side performance profiling of JavaScript for web applications. *Universidad de Radbound*.
PADILLA, L. M., CREEM-REGEHR, S. H., HEGARTY, M. & STEFANUCCI, J. K. 2018. Decision making with visualizations: a cognitive framework across disciplines. *Cognitive Research: Principles and Implications,* 3.
PARK, Y., SONG, I., YI, J., YI, S.-J. & KIM, S.-Y. 2020. Web-Based Visualization of Scientific Research Findings: National-Scale Distribution of Air Pollution in South Korea. *International Journal of Environmental Research and Public Health,* 17**,** 2230.
PERSSON, M. 2020. JavaScript DOM Manipulation Performance: Comparing Vanilla JavaScript and Leading JavaScript Front-end Frameworks.
RESCH, B., HILLEN, F., REIMER, A. & SPITZER, W. 2013. Towards 4D Cartography – Four-dimensional Dynamic Maps for Understanding Spatio-temporal Correlations in Lightning Events. *The Cartographic Journal,* 50**,** 266-275.
RESCH, B., WOHLFAHRT, R. & WOSNIOK, C. 2014. Web-based 4D visualization of marine geo-data using WebGL. 41**,** 235-247.
RICCA, F. & TONELLA, P. Analysis and testing of Web applications. 2001. IEEE Comput. Soc.
SACK, C. M., DONOHUE, R. G. & ROTH, R. E. 2014. Interactive and multivariate choropleth maps with D3. *Cartographic Perspectives***,** 57-76.
SANTANA, S. D. S., ESTÉVEZ, J. I., MORA, S. B. S. & DELGADO, B. M. 2016. Comparison of performance between a native app and a mobile web application for monitoring a photovoltaic system. *Sistemas & Telemática,* 14**,** 29-40.
SHANG, X. 2015. A Study on Efficient Vector Mapping With Vector Tiles

Based on Cloud Server Architecture.
TIMANOVSKYI, Y. & PLECHAWSKA-WÓJCIK, M. 2020. Performance analysis of selected tools for building a Single Page Application. *Journal of Computer Sciences Institute,* 14**,** 82-87.

WOOD, J., DYKES, J., SLINGSBY, A. & CLARKE, K. 2007. Interactive Visual Exploration of a Large Spatio-temporal Dataset: Reflections on a Geovisualization Mashup. *IEEE Transactions on Visualization and Computer Graphics,* 13**,** 1176-1183.

ZUNINO, A., VELÁZQUEZ, G., CELEMÍN, J. P., MATEOS, C., HIRSCH, M. & RODRIGUEZ, J. M. 2020. Evaluating the Performance of Three Popular Web Mapping Libraries: A Case Study Using Argentina's Life Quality Index. *ISPRS International Journal of Geo-Information,* 9**,** 563.

8 APPENDIX 1 (Code example GeoJSON small dataset)

Package.json:

```
{
  "name": "project",
  "version": "1.0.0",
  "description": "Project IGEON master",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "parcel serve index.html panel.html",
    "lint": "jshint js/panel.js js/index.js",
    "build": "parcel build --public-url . index.html panel.html"
  },
  "keywords": [
    "NIBIO",
    "map",
    "master",
    "Viggo Lunde"
  ],
  "author": "Viggo Lunde",
  "license": "ISC",
  "dependencies": {
    "@fortawesome/fontawesome-free": "^5.3.1",
    "jquery": "^3.3.1",
    "jquery-ui-bundle": "^1.12.1-migrate",
    "jspanel": "^2.6.3",
    "lodash": "^4.17.20",
    "moment": "^2.29.1",
    "ol": "",
    "ol-layerswitcher": "^3.6.0",
    "proj4": "^2.5.0",
    "sweetalert2": "^9.10.12"
  },
  "devDependencies": {
    "jshint": "^2.9.6",
    "parcel": "2.0.0-beta.1"
  }
}
```

Index.html

```
<!DOCTYPE html>
<html>
<script>performance.mark('start');</script>
<head>
  <meta charset="utf-8">
  <link rel="shortcut icon" type="image/x-icon" href="img/NIBIO_emblem.ico" >
  <link rel="stylesheet" href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-
ui.css">
```

```
  <link rel="stylesheet" href="node_modules/ol-layerswitcher/src/ol-
layerswitcher.css">


  <title>GeoJson deer</title>
  <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
  <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

  <!-- Compiled and minified CSS -->
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.min.cs
s">

  <!-- Compiled and minified JavaScript -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js">
</script>
  <script src="node_modules/sweetalert2/dist/sweetalert2.all.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/promise-polyfill"></script>
  <link href="node_modules/@fortawesome/fontawesome-free/css/all.css"
rel="stylesheet">
  <link href="node_modules/jspanel/source/jquery.jspanel.min.css" rel="stylesheet">
  <script src="node_modules/jspanel/source/jquery.jspanel.min.js"></script>
</head>

<body>
  <button id="open_info_button" class="btn-primary" type="button">Master
menu</button>
  <div id="map"></div>
  <div id="mouse-position"></div>

  <form id="timeRange" action="#">
     <button id="start_button" class="btn-primary" type="button">Start
animation</button>
     <button id="stop_button" class="btn-primary" type="button">Stop
animation</button>
     <p class="range-field">
       <label id="timeLabel" for="timeInput">Timeseries change</label>
       <input type="range" id="timeInput" min="200" max="300" step="1" value="0"
/>
     </p>
   </form>

  <script src="js/index.js"></script>
</body>

</html>
```

Index.js:

```
/*jshint esversion: 6 */

import {
  initPanel
} from './panel.js';

import $ from "jquery";
import '../node_modules/ol/ol.css';
//import '../node_modules/ol-layerswitcher/src/ol-layerswitcher.css';
import '../css/main.css';

import moment from 'moment';
import _ from 'lodash';

import {
  Map,
  View
} from 'ol';
import {
  getWidth,
  getTopLeft
} from 'ol/extent.js';
import TileLayer from 'ol/layer/Tile';
import TileWMS from 'ol/source/TileWMS';
import OSM from 'ol/source/OSM';
import {
  Draw,
  Modify,
  Snap
} from 'ol/interaction.js';
import {
  Vector as VectorSource
} from 'ol/source.js';
import {
  Circle as CircleStyle,
  Fill,
  Stroke,
  Style
} from 'ol/style.js';
import {
  get as getProjection
} from 'ol/proj.js';
import {
  register
} from 'ol/proj/proj4.js';
import proj4 from 'proj4';
import WMTS from 'ol/source/WMTS.js';
import WMTSTileGrid from 'ol/tilegrid/WMTS.js';
import {
```

```
  defaults as defaultControls
} from 'ol/control.js';
import MousePosition from 'ol/control/MousePosition.js';
import {
  Zoom,
  ZoomSlider,
  ZoomToExtent,
  Attribution
} from 'ol/control.js';
import {
  createStringXY
} from 'ol/coordinate.js';
import {
  Image as ImageLayer,
  Vector as VectorLayer
} from 'ol/layer.js';
import ImageWMS from 'ol/source/ImageWMS.js';
import GeoJSON from 'ol/format/GeoJSON';
import Observable from 'ol/Observable';
import LayerGroup from 'ol/layer/Group';
import LayerSwitcher from 'ol-layerswitcher';
import Swal from 'sweetalert2';

var timeInterval;

var key;
var oldItems, loadNr;

var backgroundUrl = "https://opencache.statkart.no/gatekeeper/gk/gk.open_wmts";
// OPEN WMTS first
var backgroundUrlNib =
"https://opencache.statkart.no/gatekeeper/gk/gk.open_nib_utm33_wmts_v2";
// OPEN WMTS first

  $.ajax({
  url: '/map/token.jsp',
  contentType: "application/json",
  dataType: "text",
  success: function (data, textStatus, jqXHR) {
    var result = jQuery.parseJSON( data );
    if (result.key) {
      key = result.key;
      backgroundUrl = "https://gatekeeper{1-
3}.geonorge.no/BaatGatekeeper/gk/gk.cache_wmts?gkt="+key;
      backgroundUrlNib = "https://gatekeeper(Jagosh et
al.).geonorge.no/BaatGatekeeper/gk/gk.nib_utm33_wmts_v2?gkt="+key;
    }
    function createNewSource(layerName) {
      var newSource = new WMTS({
        url: backgroundUrl,
```

```
        crossOrigin: 'Anonymous',
        layer: layerName,
        matrixSet: 'EPSG:25833',
        format: 'image/png',
        projection: projection,
        tileGrid: new WMTSTileGrid({
          origin: getTopLeft(mapExtent),
          resolutions: resolutions,
          matrixIds: matrixIds
        }),
        style: 'default',
        extent: mapExtent,
      });
      return newSource;
    }
    var newSourceNib = new WMTS({
      url: backgroundUrlNib,
      crossOrigin: 'Anonymous',
      layer: 'Nibcache_UTM33_EUREF89_v2',
      matrixSet: 'default028mm',
      format: 'image/png',
      projection: projection,
      tileGrid: new WMTSTileGrid({
        origin: getTopLeft(mapExtent),
        resolutions: resolutions,
        matrixIds: matrixIds_nib
      }),
      style: 'default',
      extent: mapExtent,
    });
    graatone.setSource(createNewSource('topo4graatone'));
    farger.setSource(createNewSource('topo4'));
    raster.setSource(createNewSource('toporaster3'));
    norgeibilder.setSource(newSourceNib);
   },
   error: function(jqXHR, textStatus, errorThrown) {
     //Error handling code
     console.log('Kunne ikke hente token fra kartverket, bruker opencache');
   }

});


  proj4.defs("EPSG:25833", "+proj=utm +zone=33 +ellps=GRS80
+towgs84=0,0,0,0,0,0,0 +units=m +no_defs");
  register(proj4);
  //var projection = ol.proj.get('EPSG:32633');
  var projection = getProjection('EPSG:25833');
  var mapExtent = [-2500000.0, 3500000.0, 3045984.0, 9045984.0];
  var resolutions = [21664,
```

```
  10832,
  5416,
  2708,
  1354,
  677,
  338.5,
  169.25,
  84.625,
  42.3125,
  21.15625,
  10.578125,
  5.2890625,
  2.64453125,
  1.322265625,
  0.6611328125,
  0.33056640625,
  0.165283203125
];
var matrixIds = new Array(resolutions.length);
var matrixIds_nib = new Array(resolutions.length);
for (var z = 0; z < resolutions.length; ++z) {
  matrixIds[z] = 'EPSG:25833:' + z;
  matrixIds_nib[z] = z;
}



// Background raster
var graatone = new TileLayer({
  title: 'Gråtone',
  id: 'GRAATONE',
  type: 'base',
  source: new WMTS({
    // url: "http://opencache.statkart.no/gatekeeper/gk/gk.open_wmts",
    url: backgroundUrl,
    crossOrigin: 'Anonymous',
    layer: 'topo4graatone',
    matrixSet: 'EPSG:25833',
    format: 'image/png',
    projection: projection,
    tileGrid: new WMTSTileGrid({
      origin: getTopLeft(mapExtent),
      resolutions: resolutions,
      matrixIds: matrixIds
    }),
    style: 'default',
    extent: mapExtent,
  })
});
```

```
var farger = new TileLayer({
  title: 'Farger',
  id: 'FARGER',
  type: 'base',
  source: new WMTS({
    // url: "http://opencache.statkart.no/gatekeeper/gk/gk.open_wmts",
    url: backgroundUrl,
    crossOrigin: 'Anonymous',
    layer: 'topo4',
    matrixSet: 'EPSG:25833',
    format: 'image/png',
    projection: projection,
    tileGrid: new WMTSTileGrid({
      origin: getTopLeft(mapExtent),
      resolutions: resolutions,
      matrixIds: matrixIds
    }),
    style: 'default',
    extent: mapExtent,
  })
});


var norgeibilder = new TileLayer({
  title: 'Norge i bilder',
  id: 'flybilder',
  type: 'base',
  source: new WMTS({
    url: backgroundUrlNib,
    crossOrigin: 'Anonymous',
    layer: 'Nibcache_UTM33_EUREF89_v2',
    matrixSet: "default028mm",
    format: 'image/jpeg',
    projection: projection,
    tileGrid: new WMTSTileGrid({
      origin: getTopLeft(mapExtent),
      resolutions: resolutions,
      matrixIds: matrixIds_nib
    }),
    style: 'default',
    extent: mapExtent,
  })
});
var raster = new TileLayer({
  title: 'Raster',
  id: 'RASTER',
  type: 'base',
  source: new WMTS({
    // url: "http://opencache.statkart.no/gatekeeper/gk/gk.open_wmts",
    url: backgroundUrl,
```

```
      crossOrigin: 'Anonymous',
      layer: 'toporaster3',
      matrixSet: 'EPSG:25833',
      format: 'image/png',
      projection: projection,
      tileGrid: new WMTSTileGrid({
        origin: getTopLeft(mapExtent),
        resolutions: resolutions,
        matrixIds: matrixIds
      }),
      style: 'default',
      extent: mapExtent,
    })
  });


var mousePositionControl = new MousePosition({
  coordinateFormat: createStringXY(0),
  projection: 'EPSG:25833',
  // comment the following two lines to have the mouse position
  // be placed within the map.
  className: 'custom-mouse-position',
  target: document.getElementById('mouse-position'),
  undefinedHTML: 'UTM 33'
});

var zoomOptions = {
  className: 'map-zoom',
  zoomInLabel: '',
  zoomOutLabel: '',
  zoomInTipLabel: 'Zoom in',
  zoomOutTipLabel: 'Zoom out',
};

var zoomSliderOptions = {
  className: 'ol-zoomslider'
};
var extentOptions = {
  extent: [-2175810,5823784,2933018,8628631],
  tipLabel: 'Zoom til hele Norge',
  label: '',
  className: 'mapExtent'
};

var source = new VectorSource();
var features;

performance.mark('load.data.start');
$.ajax({
    type: "GET",
```

```
    url: "data/json1-30sept.txt",

    dataType: 'json',
    async: false,
    success: function (response) {
       features = new GeoJSON().readFeatures(response);
       source.addFeatures(features) ;

    }
});


var hiddenLayer = new VectorLayer({
  title: 'All points',
  source: source,
  visible: true,
  //type: base,
  style: new Style({
    fill: new Fill({
      color: 'rgba(255, 0, 0, 0.2)'
    }),
    stroke: new Stroke({
      color: '#ff0000',
      width: 2
    }),
    image: new CircleStyle({
      radius: 7,
      fill: new Fill({
        color: '#ff0000'
      })
    })
  })
});
var visibleSource = new VectorSource();
var visibleLayer = new VectorLayer({
    title: 'Filtered points',
    source: visibleSource,
    //visible: true,
    style: new Style({
      fill: new Fill({
        color: 'rgba( 0, 0, 255, 0.2)'
      }),
      stroke: new Stroke({
        color: '#0000ff',
        width: 2
      }),
      image: new CircleStyle({
        radius: 7,
        fill: new Fill({
          color: '#0000ff'
```

```
      })
    })
  })
});


const map = new Map({
  controls: [
    new Zoom(zoomOptions),
    new ZoomSlider(zoomSliderOptions),
    new ZoomToExtent(extentOptions),
    new Attribution({
      collapsible: false,
      collapsed: false
    }),
    mousePositionControl
  ],
  target: 'map',

  layers: [
        new LayerGroup({
            'title': 'Backgroud',
            layers: [
                new TileLayer({
                    title: 'OSM',
                    type: 'base',
                    visible: true,
                    source: new OSM()
                }),
                raster,
                norgeibilder,
                farger,
                graatone
            ]
        }),
        new LayerGroup({
            title: 'Time Layers',
            layers: [
              hiddenLayer,
              visibleLayer
            ]
        })

    ],
  view: new View({
    projection: projection,
    center: [378604, 7226208],
    zoom: 6,
    maxZoom: 20,
    minZoom: 6
```

```
  })
});

performance.mark('load.data.end');

var layerSwitcher = new LayerSwitcher();
  map.addControl(layerSwitcher);

//Change size on map
setMapSize();

function setMapSize() {

  var mapWidth = window.innerWidth;
  var mapHeight = window.innerHeight;

  $('#map').css({
    width: mapWidth + 'px',
    height: mapHeight + 'px'
  }); // change OpenLayers map *container* size
  map.setSize([mapWidth, mapHeight]); // adjust the map's size
  map.updateSize();

}


window.updateLayer = function updateLayer(sid) {
  console.log('sid'+sid);

        // Clear the source before adding data
        visibleSource.clear();
        var features = source.getFeatures();
        let ordered = _.orderBy(features, o => o.get('Acquisition_time'), ['asc']);

    document.getElementById('timeLabel').innerHTML =
ordered[sid].get('Acquisition_time');
    visibleSource.addFeatures([ordered[sid], ordered[sid-1], ordered[sid-2],
ordered[sid-3], ordered[sid-4], ordered[sid-5]]);

    //console.log(ordered[sid].get('id_position'));
  }

  var elem = document.getElementById('timeInput');

  var rangeValue = function(){
    updateLayer(elem.value);
  }

  elem.addEventListener("input", rangeValue);
```

```
export function startAnimation() {
  performance.mark('anim.start');
  console.log('startAnimation');
  var counter = 200;
  //var counter = 10;
  timeInterval = window.setInterval(function(){
    if (counter < features.length && counter < 300){
    //console.log('visibleSource.getState(): ' + visibleSource.getState())
    //if (counter < features.length && counter < 300 && visibleSource.getState() ===
'ready'){
      updateLayer(counter);
      document.getElementById('timeInput').value =
      counter ++;
      if (counter === 230){
        map.getView().setZoom(map.getView().getZoom() + 1);
      }
      if (counter === 260){
        map.getView().setZoom(map.getView().getZoom() -2);
      }
      if (counter === 299){
         performance.mark('anim.end');
         loadTimes(100);
       }
    }
  }, 500);
}

export function stopAnimation(){
  window.clearInterval(timeInterval);
}

var startButton = document.getElementById('start_button');
startButton.addEventListener('click', startAnimation, false);

var stopButton = document.getElementById('stop_button');
stopButton.addEventListener('click', stopAnimation, false);

//visibleSource.on('featuresloadend', startAnimation());



window.onresize = function(event) {
  setMapSize();
};

var saveData = (function () {
var a = document.createElement("a");
document.body.appendChild(a);
a.style = "display: none";
return function (data, fileName) {
```

```
    var json = JSON.stringify(data),
        blob = new Blob([json], {type: "octet/stream"}),
        url = window.URL.createObjectURL(blob);
    a.href = url;
    a.download = fileName;
    a.click();
    window.URL.revokeObjectURL(url);
};
}());




$(document).ready(function() {

  $('#open_info_button').click(function() {
    initPanel();
  });

  initPanel();

  var hiddenExtent = hiddenLayer.getSource().getExtent();
  console.log(hiddenExtent);
  if (hiddenExtent) {
     map.getView().fit(hiddenExtent);
     map.getView().setZoom(map.getView().getZoom()-1);
  }

  performance.mark('end');
  //loadTimes(100);
});

var saveData = (function () {
var a = document.createElement("a");
document.body.appendChild(a);
a.style = "display: none";
return function (data, fileName) {
    var json = JSON.stringify(data),
        blob = new Blob([json], {type: "text/csv"}),
        url = window.URL.createObjectURL(blob);
    a.href = url;
    a.download = fileName;
    a.click();
    window.URL.revokeObjectURL(url);
};
}());

function convertToCSV(arr) {
  const array = [Object.keys(arr[0])].concat(arr)
```

```
  return array.map(it => {
    return Object.values(it).toString()
  }).join('\n')
}



function loadTimes(times) {
  oldItems = JSON.parse(localStorage.getItem('itemsArray')) || [];
  loadNr = oldItems.length;

  var newItem = {
      'Nr': loadNr,
      'mark-total': performance.measure('total', 'start', 'end').duration,
      'mark-load-data': performance.measure('load-data', 'load.data.start',
'load.data.end').duration,
      'draw-on map':performance.measure('draw-map', 'load.data.end', 'end').duration/*,
      'animation':performance.measure('animation', 'anim.start', 'anim.end').duration*/

  };


  oldItems.push(newItem);
  console.log('stored : ',oldItems);

  //console.table('stored times: '+ Object.entries(oldItems));
  localStorage.setItem('itemsArray', JSON.stringify(oldItems));

  if (loadNr <= times) {
    console.log('load'+loadNr);
    window.location.reload(true);
  } else {
    var fileName = "timeData_geoJson.csv";
    var csvFormat = convertToCSV(oldItems)
    saveData(csvFormat, fileName);
  }
}
```

Series from Lund University

Department of Physical Geography and Ecosystem Science

Master Thesis in Geographical Information Science

1. *Anthony Lawther:* The application of GIS-based binary logistic regression for slope failure susceptibility mapping in the Western Grampian Mountains, Scotland (2008).
2. *Rickard Hansen:* Daily mobility in Grenoble Metropolitan Region, France. Applied GIS methods in time geographical research (2008).
3. *Emil Bayramov:* Environmental monitoring of bio-restoration activities using GIS and Remote Sensing (2009).
4. *Rafael Villarreal Pacheco:* Applications of Geographic Information Systems as an analytical and visualization tool for mass real estate valuation: a case study of Fontibon District, Bogota, Columbia (2009).
5. *Siri Oestreich Waage:* a case study of route solving for oversized transport: The use of GIS functionalities in transport of transformers, as part of maintaining a reliable power infrastructure (2010).
6. *Edgar Pimiento:* Shallow landslide susceptibility – Modelling and validation (2010).
7. *Martina Schäfer:* Near real-time mapping of floodwater mosquito breeding sites using aerial photographs (2010).
8. *August Pieter van Waarden-Nagel:* Land use evaluation to assess the outcome of the programme of rehabilitation measures for the river Rhine in the Netherlands (2010).
9. *Samira Muhammad:* Development and implementation of air quality data mart for Ontario, Canada: A case study of air quality in Ontario using OLAP tool. (2010).
10. *Fredros Oketch Okumu*: Using remotely sensed data to explore spatial and temporal relationships between photosynthetic productivity of vegetation and malaria transmission intensities in selected parts of Africa (2011).
11. *Svajunas Plunge:* Advanced decision support methods for solving diffuse water pollution problems (2011).
12. *Jonathan Higgins:* Monitoring urban growth in greater Lagos: A case study using GIS to monitor the urban growth of Lagos 1990 - 2008 and produce future growth prospects for the city (2011).
13. *Mårten Karlberg:* Mobile Map Client API: Design and Implementation for Android (2011).
14. *Jeanette McBride:* Mapping Chicago area urban tree canopy using color infrared imagery (2011).
15. *Andrew Farina:* Exploring the relationship between land surface temperature and vegetation abundance for urban heat island mitigation in Seville, Spain (2011).
16. *David Kanyari*: Nairobi City Journey Planner: An online and a Mobile Application (2011).
17. *Laura V. Drews:* Multi-criteria GIS analysis for siting of small wind power plants - A case study from Berlin (2012).
18. *Qaisar Nadeem:* Best living neighborhood in the city - A GIS based multi criteria evaluation of ArRiyadh City (2012).
19. *Ahmed Mohamed El Saeid Mustafa:* Development of a photo voltaic building rooftop integration analysis tool for GIS for Dokki District, Cairo, Egypt (2012).

20. *Daniel Patrick Taylor*: Eastern Oyster Aquaculture: Estuarine Remediation via Site Suitability and Spatially Explicit Carrying Capacity Modeling in Virginia's Chesapeake Bay (2013).
21. *Angeleta Oveta Wilson:* A Participatory GIS approach to *unearthing* Manchester's Cultural Heritage '*gold mine'* (2013).
22. *Ola Svensson:* Visibility and Tholos Tombs in the Messenian Landscape: A Comparative Case Study of the Pylian Hinterlands and the Soulima Valley (2013).
23. *Monika Ogden:* Land use impact on water quality in two river systems in South Africa (2013).
24. *Stefan Rova:* A GIS based approach assessing phosphorus load impact on Lake Flaten in Salem, Sweden (2013).
25. *Yann Buhot:* Analysis of the history of landscape changes over a period of 200 years. How can we predict past landscape pattern scenario and the impact on habitat diversity? (2013).
26. *Christina Fotiou:* Evaluating habitat suitability and spectral heterogeneity models to predict weed species presence (2014).
27. *Inese Linuza:* Accuracy Assessment in Glacier Change Analysis (2014).
28. *Agnieszka Griffin:* Domestic energy consumption and social living standards: a GIS analysis within the Greater London Authority area (2014).
29. *Brynja Guðmundsdóttir:* Detection of potential arable land with remote sensing and GIS - A Case Study for Kjósarhreppur (2014).
30. *Oleksandr Nekrasov:* Processing of MODIS Vegetation Indices for analysis of agricultural droughts in the southern Ukraine between the years 2000-2012 (2014).
31. *Sarah Tressel:* Recommendations for a polar Earth science portal in the context of Arctic Spatial Data Infrastructure (2014).
32. *Caroline Gevaert:* Combining Hyperspectral UAV and Multispectral Formosat-2 Imagery for Precision Agriculture Applications (2014).
33. *Salem Jamal-Uddeen:* Using GeoTools to implement the multi-criteria evaluation analysis - weighted linear combination model (2014).
34. *Samanah Seyedi-Shandiz:* Schematic representation of geographical railway network at the Swedish Transport Administration (2014).
35. *Kazi Masel Ullah:* Urban Land-use planning using Geographical Information System and analytical hierarchy process: case study Dhaka City (2014).
36. *Alexia Chang-Wailing Spitteler:* Development of a web application based on MCDA and GIS for the decision support of river and floodplain rehabilitation projects (2014).
37. *Alessandro De Martino:* Geographic accessibility analysis and evaluation of potential changes to the public transportation system in the City of Milan (2014).
38. *Alireza Mollasalehi:* GIS Based Modelling for Fuel Reduction Using Controlled Burn in Australia. Case Study: Logan City, QLD (2015).
39. *Negin A. Sanati:* Chronic Kidney Disease Mortality in Costa Rica; Geographical Distribution, Spatial Analysis and Non-traditional Risk Factors (2015).
40. *Karen McIntyre:* Benthic mapping of the Bluefields Bay fish sanctuary, Jamaica (2015).
41. *Kees van Duijvendijk:* Feasibility of a low-cost weather sensor network for agricultural purposes: A preliminary assessment (2015).

42. *Sebastian Andersson Hylander:* Evaluation of cultural ecosystem services using GIS (2015).
43. *Deborah Bowyer:* Measuring Urban Growth, Urban Form and Accessibility as Indicators of Urban Sprawl in Hamilton, New Zealand (2015).
44. *Stefan Arvidsson:* Relationship between tree species composition and phenology extracted from satellite data in Swedish forests (2015).
45. *Damián Giménez Cruz*: GIS-based optimal localisation of beekeeping in rural Kenya (2016).
46. *Alejandra Narváez Vallejo:* Can the introduction of the topographic indices in LPJ-GUESS improve the spatial representation of environmental variables? (2016).
47. *Anna Lundgren:* Development of a method for mapping the highest coastline in Sweden using breaklines extracted from high resolution digital elevation models (2016).
48. *Oluwatomi Esther Adejoro:* Does location also matter? A spatial analysis of social achievements of young South Australians (2016).
49. *Hristo Dobrev Tomov:* Automated temporal NDVI analysis over the Middle East for the period 1982 - 2010 (2016).
50. *Vincent Muller:* Impact of Security Context on Mobile Clinic Activities A GIS Multi Criteria Evaluation based on an MSF Humanitarian Mission in Cameroon (2016).
51. *Gezahagn Negash Seboka:* Spatial Assessment of NDVI as an Indicator of Desertification in Ethiopia using Remote Sensing and GIS (2016).
52. *Holly Buhler:* Evaluation of Interfacility Medical Transport Journey Times in Southeastern British Columbia. (2016).
53. *Lars Ole Grottenberg*: Assessing the ability to share spatial data between emergency management organisations in the High North (2016).
54. *Sean Grant:* The Right Tree in the Right Place: Using GIS to Maximize the Net Benefits from Urban Forests (2016).
55. *Irshad Jamal:* Multi-Criteria GIS Analysis for School Site Selection in Gorno-Badakhshan Autonomous Oblast, Tajikistan (2016).
56. *Fulgencio Sanmartín:* Wisdom-volkano: A novel tool based on open GIS and time-series visualization to analyse and share volcanic data (2016).
57. *Nezha Acil:* Remote sensing-based monitoring of snow cover dynamics and its influence on vegetation growth in the Middle Atlas Mountains (2016).
58. *Julia Hjalmarsson:* A Weighty Issue: Estimation of Fire Size with Geographically Weighted Logistic Regression (2016).
59. *Mathewos Tamiru Amato:* Using multi-criteria evaluation and GIS for chronic food and nutrition insecurity indicators analysis in Ethiopia (2016).
60. *Karim Alaa El Din Mohamed Soliman El Attar:* Bicycling Suitability in Downtown, Cairo, Egypt (2016).
61. *Gilbert Akol Echelai:* Asset Management: Integrating GIS as a Decision Support Tool in Meter Management in National Water and Sewerage Corporation (2016).
62. *Terje Slinning:* Analytic comparison of multibeam echo soundings (2016).
63. *Gréta Hlín Sveinsdóttir:* GIS-based MCDA for decision support: A framework for wind farm siting in Iceland (2017).
64. *Jonas Sjögren:* Consequences of a flood in Kristianstad, Sweden: A GIS-based analysis of impacts on important societal functions (2017).

65. *Nadine Raska:* 3D geologic subsurface modelling within the Mackenzie Plain, Northwest Territories, Canada (2017).
66. *Panagiotis Symeonidis*: Study of spatial and temporal variation of atmospheric optical parameters and their relation with PM 2.5 concentration over Europe using GIS technologies (2017).
67. *Michaela Bobeck:* A GIS-based Multi-Criteria Decision Analysis of Wind Farm Site Suitability in New South Wales, Australia, from a Sustainable Development Perspective (2017).
68. *Raghdaa Eissa*: Developing a GIS Model for the Assessment of Outdoor Recreational Facilities in New Cities Case Study: Tenth of Ramadan City, Egypt (2017).
69. *Zahra Khais Shahid*: Biofuel plantations and isoprene emissions in Svea and Götaland (2017).
70. *Mirza Amir Liaquat Baig*: Using geographical information systems in epidemiology: Mapping and analyzing occurrence of diarrhea in urban - residential area of Islamabad, Pakistan (2017).
71. *Joakim Jörwall*: Quantitative model of Present and Future well-being in the EU-28: A spatial Multi-Criteria Evaluation of socioeconomic and climatic comfort factors (2017).
72. *Elin Haettner*: Energy Poverty in the Dublin Region: Modelling Geographies of Risk (2017).
73. *Harry Eriksson*: Geochemistry of stream plants and its statistical relations to soil- and bedrock geology, slope directions and till geochemistry. A GIS-analysis of small catchments in northern Sweden (2017).
74. *Daniel Gardevärn:* PPGIS and Public meetings – An evaluation of public participation methods for urban planning (2017).
75. *Kim Friberg:* Sensitivity Analysis and Calibration of Multi Energy Balance Land Surface Model Parameters (2017).
76. *Viktor Svanerud:* Taking the bus to the park? A study of accessibility to green areas in Gothenburg through different modes of transport (2017).
77. *Lisa-Gaye Greene*: Deadly Designs: The Impact of Road Design on Road Crash Patterns along Jamaica's North Coast Highway (2017).
78. *Katarina Jemec Parker*: Spatial and temporal analysis of fecal indicator bacteria concentrations in beach water in San Diego, California (2017).
79. *Angela Kabiru*: An Exploratory Study of Middle Stone Age and Later Stone Age Site Locations in Kenya's Central Rift Valley Using Landscape Analysis: A GIS Approach (2017).
80. *Kristean Björkmann*: Subjective Well-Being and Environment: A GIS-Based Analysis (2018).
81. *Williams Erhunmonmen Ojo*: Measuring spatial accessibility to healthcare for people living with HIV-AIDS in southern Nigeria (2018).
82. *Daniel Assefa*: Developing Data Extraction and Dynamic Data Visualization (Styling) Modules for Web GIS Risk Assessment System (WGRAS). (2018).
83. *Adela Nistora*: Inundation scenarios in a changing climate: assessing potential impacts of sea-level rise on the coast of South-East England (2018).
84. *Marc Seliger*: Thirsty landscapes - Investigating growing irrigation water consumption and potential conservation measures within Utah's largest master-planned community: Daybreak (2018).
85. *Luka Jovičić*: Spatial Data Harmonisation in Regional Context in Accordance with INSPIRE Implementing Rules (2018).

86. *Christina Kourdounouli*: Analysis of Urban Ecosystem Condition Indicators for the Large Urban Zones and City Cores in EU (2018).
87. *Jeremy Azzopardi*: Effect of distance measures and feature representations on distance-based accessibility measures (2018).
88. *Patrick Kabatha*: An open source web GIS tool for analysis and visualization of elephant GPS telemetry data, alongside environmental and anthropogenic variables (2018).
89. *Richard Alphonce Giliba*: Effects of Climate Change on Potential Geographical Distribution of Prunus africana (African cherry) in the Eastern Arc Mountain Forests of Tanzania (2018).
90. *Eiður Kristinn Eiðsson*: Transformation and linking of authoritative multi-scale geodata for the Semantic Web: A case study of Swedish national building data sets (2018).
91. *Niamh Harty*: HOP!: a PGIS and citizen science approach to monitoring the condition of upland paths (2018).
92. *José Estuardo Jara Alvear*: Solar photovoltaic potential to complement hydropower in Ecuador: A GIS-based framework of analysis (2018).
93. *Brendan O'Neill*: Multicriteria Site Suitability for Algal Biofuel Production Facilities (2018).
94. *Roman Spataru*: Spatial-temporal GIS analysis in public health – a case study of polio disease (2018).
95. *Alicja Miodońska*: Assessing evolution of ice caps in Suðurland, Iceland, in years 1986 - 2014, using multispectral satellite imagery (2019).
96. *Dennis Lindell Schettini*: A Spatial Analysis of Homicide Crime's Distribution and Association with Deprivation in Stockholm Between 2010-2017 (2019).
97. *Damiano Vesentini*: The Po Delta Biosphere Reserve: Management challenges and priorities deriving from anthropogenic pressure and sea level rise (2019).
98. *Emilie Arnesten*: Impacts of future sea level rise and high water on roads, railways and environmental objects: a GIS analysis of the potential effects of increasing sea levels and highest projected high water in Scania, Sweden (2019).
99. *Syed Muhammad Amir Raza*: Comparison of geospatial support in RDF stores: Evaluation for ICOS Carbon Portal metadata (2019).
100. *Hemin Tofiq*: Investigating the accuracy of Digital Elevation Models from UAV images in areas with low contrast: A sandy beach as a case study (2019).
101. *Evangelos Vafeiadis*: Exploring the distribution of accessibility by public transport using spatial analysis. A case study for retail concentrations and public hospitals in Athens (2019).
102. *Milan Sekulic*: Multi-Criteria GIS modelling for optimal alignment of roadway by-passes in the Tlokweng Planning Area, Botswana (2019).
103. *Ingrid Piirisaar*: A multi-criteria GIS analysis for siting of utility-scale photovoltaic solar plants in county Kilkenny, Ireland (2019).
104. *Nigel Fox*: Plant phenology and climate change: possible effect on the onset of various wild plant species' first flowering day in the UK (2019).
105. *Gunnar Hesch*: Linking conflict events and cropland development in Afghanistan, 2001 to 2011, using MODIS land cover data and Uppsala Conflict Data Programme (2019).
106. *Elijah Njoku*: Analysis of spatial-temporal pattern of Land Surface Temperature (LST) due to NDVI and elevation in Ilorin, Nigeria (2019).

107. *Katalin Bunyevácz*: Development of a GIS methodology to evaluate informal urban green areas for inclusion in a community governance program (2019).
108. *Paul dos Santos*: Automating synthetic trip data generation for an agent-based simulation of urban mobility (2019).
109. *Robert O' Dwyer*: Land cover changes in Southern Sweden from the mid-Holocene to present day: Insights for ecosystem service assessments (2019).
110. *Daniel Klingmyr*: Global scale patterns and trends in tropospheric NO2 concentrations (2019).
111. *Marwa Farouk Elkabbany*: Sea Level Rise Vulnerability Assessment for Abu Dhabi, United Arab Emirates (2019).
112. *Jip Jan van Zoonen*: Aspects of Error Quantification and Evaluation in Digital Elevation Models for Glacier Surfaces (2020).
113. *Georgios Efthymiou*: The use of bicycles in a mid-sized city – benefits and obstacles identified using a questionnaire and GIS (2020).
114. *Haruna Olayiwola Jimoh*: Assessment of Urban Sprawl in MOWE/IBAFO Axis of Ogun State using GIS Capabilities (2020).
115. *Nikolaos Barmpas Zachariadis*: Development of an iOS, Augmented Reality for disaster management (2020).
116. *Ida Storm*: ICOS Atmospheric Stations: Spatial Characterization of CO2 Footprint Areas and Evaluating the Uncertainties of Modelled CO2 Concentrations (2020).
117. *Alon Zuta*: Evaluation of water stress mapping methods in vineyards using airborne thermal imaging (2020).
118. *Marcus Eriksson*: Evaluating structural landscape development in the municipality Upplands-Bro, using landscape metrics indices (2020).
119. *Ane Rahbek Vierø*: Connectivity for Cyclists? A Network Analysis of Copenhagen's Bike Lanes (2020).
120. *Cecilia Baggini*: Changes in habitat suitability for three declining Anatidae species in saltmarshes on the Mersey estuary, North-West England (2020).
121. *Bakrad Balabanian*: Transportation and Its Effect on Student Performance (2020).
122. *Ali Al Farid*: Knowledge and Data Driven Approaches for Hydrocarbon Microseepage Characterizations: An Application of Satellite Remote Sensing (2020).
123. *Bartlomiej Kolodziejczyk*: Distribution Modelling of Gene Drive-Modified Mosquitoes and Their Effects on Wild Populations (2020).
124. *Alexis Cazorla*: Decreasing organic nitrogen concentrations in European water bodies - links to organic carbon trends and land cover (2020).
125. *Kharid Mwakoba*: Remote sensing analysis of land cover/use conditions of community-based wildlife conservation areas in Tanzania (2021).
126. *Chinatsu Endo*: Remote Sensing Based Pre-Season Yellow Rust Early Warning in Oromia, Ethiopia (2021).
127. *Berit Mohr*: Using remote sensing and land abandonment as a proxy for long-term human out-migration. A Case Study: Al-Hassakeh Governorate, Syria (2021).
128. *Kanchana Nirmali Bandaranayake*: Considering future precipitation in delineation locations for water storage systems - Case study Sri Lanka (2021).

129. *Emma Bylund*: Dynamics of net primary production and food availability in the aftermath of the 2004 and 2007 desert locust outbreaks in Niger and Yemen (2021).
130. *Shawn Pace*: Urban infrastructure inundation risk from permanent sea-level rise scenarios in London (UK), Bangkok (Thailand) and Mumbai (India): A comparative analysis (2021).
131. *Oskar Evert Johansson*: The hydrodynamic impacts of Estuarine Oyster reefs, and the application of drone technology to this study (2021).
132. *Pritam Kumarsingh*: A Case Study to develop and test GIS/SDSS methods to assess the production capacity of a Cocoa Site in Trinidad and Tobago (2021).
133. *Muhammad Imran Khan*: Property Tax Mapping and Assessment using GIS (2021).
134. *Domna Kanari*: Mining geosocial data from Flickr to explore tourism patterns: The case study of Athens (2021).
135. *Mona Tykesson Klubien*: Livestock-MRSA in Danish pig farms (2021).
136. *Ove Njøten*: Comparing radar satellites. Use of Sentinel-1 leads to an increase in oil spill alerts in Norwegian waters (2021).
137. *Panagiotis Patrinos*: Change of heating fuel consumption patterns produced by the economic crisis in Greece (2021).
138. *Lukasz Langowski*: Assessing the suitability of using Sentinel-1A SAR multi-temporal imagery to detect fallow periods between rice crops (2021).
139. *Jonas Tillman*: Perception accuracy and user acceptance of legend designs for opacity data mapping in GIS (2022).
140. *Gabriela Olekszyk*: ALS (Airborne LIDAR) accuracy: Can potential low data quality of ground points be modelled/detected? Case study of 2016 LIDAR capture over Auckland, New Zealand (2022).
141. *Luke Aspland*: Weights of Evidence Predictive Modelling in Archaeology (2022).
142. *Luís Fareleira Gomes*: The influence of climate, population density, tree species and land cover on fire pattern in mainland Portugal (2022).
143. *Andreas Eriksson*: Mapping Fire Salamander (Salamandra salamandra) Habitat Suitability in Baden-Württemberg with Multi-Temporal Sentinel-1 and Sentinel-2 Imagery (2022).
144. *Lisbet Hougaard Baklid*: Geographical expansion rate of a brown bear population in Fennoscandia and the factors explaining the directional variations (2022).
145. *Victoria Persson*: Mussels in deep water with climate change: Spatial distribution of mussel (Mytilus galloprovincialis) growth offshore in the French Mediterranean with respect to climate change scenario RCP 8.5 Long Term and Integrated Multi-Trophic Aquaculture (IMTA) using Dynamic Energy Budget (DEB) modelling (2022).
146. *Benjamin Bernard Fabien Gérard Borgeais*: Implementing a multi-criteria GIS analysis and predictive modelling to locate Upper Palaeolithic decorated caves in the Périgord noir, France (2022).
147. *Bernat Dorado-Guerrero*: Assessing the impact of post-fire restoration interventions using spectral vegetation indices: A case study in El Bruc, Spain (2022).

148. *Ignatius Gabriel Aloysius Maria Perera*: The Influence of Natural Radon Occurrence on the Severity of the COVID-19 Pandemic in Germany: A Spatial Analysis (2022).
149. *Mark Overton*: An Analysis of Spatially-enabled Mobile Decision Support Systems in a Collaborative Decision-Making Environment (2022).
150. *Viggo Lunde*: Analysing methods for visualizing time-series datasets in open-source web mapping (2022).