UiO : Department of Informatics
University of Oslo

VoVis: A Vocabulary-based Web Visualization Framework

Swati Sharma

Master Thesis

May 2015

VOVIS: A VOCABULARY-BASED WEB VISUALIZATION FRAMEWORK

SWATI SHARMA

MASTER THESIS

Institutt for informatikk
Universitetet i Oslo

May 2015

To my parents and my adorable son Shriyan,
for their Support and Inspiration.

ABSTRACT

The amount of data in today's digital world is growing day by day. Visualization is considered as the best form of communication because the human brain perceives it much faster than the text data that comprises with thousand of words.

Because of the enormous and continuously increasing data, the demand to visualize it is also increasing. Information visualization is a wide research area that covers a broad range of data fields. There are various visualization frameworks, tools and technologies are available in the market to present different data. A visualization framework is a complete package that contains several visualization processing stages i.e. data collection in different formats, data filtration, data mapping, data processing and data visualization. The important goal in visualization is to convey the information of data correctly. This can be achieved by improving the user-friendly data mapping mechanism. The state-of-art solutions lacks several features which are important in visualizing the data. This thesis has four important contributions. First, it enumerates these features by referring available standards and technologies. Second, the feature requirements for a general web Visualization framework are identified. Third, several existing frameworks are analyzed and evaluated based on these requirement set. Fourth, an optimal solution is stated, designed and developed as a vocabulary based web visualization (*VoVis*) framework.

In this framework, to improve mapping mechanism and to support a broad range of visualization types, the concept of vocabulary for visualization is introduced and designed. The developed *VoVis* framework has a vocabulary component (vocabulary and visualization libraries) which uses the standard JavaScript libraries for visualization.

The *VoVis* framework is built as a web-based application which is a modern and emerging technology and well known for low maintainability cost.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere thanks and appreciation to my supervisor, Dr. Arne-Jørgen Berre, and Dr. Dumitru Roman, who have provided invaluable guidance, expertise and support throughout the development of this thesis. Their mentoring has helped me to think from a technical perspective and enhanced the depth of my thesis.

I would like to extend my sincere thanks to my brother, Shashank for all the technical guidance and support.

Last but not the least my husband, Sudhir for continuous support and inspiration.

Swati Sharma

Oslo, May 2015.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

VoVis Vocabulary-based Visualization

RDF Resource Description Framework

DaPaaS Data-and-Platform-as-a-Service

PLUQI Personalized and Localized Urban Quality Index

OWL Web Ontology Language

XML Extensible Markup Language

JSON JavaScript Object Notation

HTML5 HyperText Markup Language 5

CSS Cascading Style Sheets

VA Visual Analytics

CSV Comma-Separated Values

InfoVis Information Visualization

AQI Air Quality Index

D3 Data Driven Documents

DOM Document Object Model

SVG Scalable Vector Graphics

API Application Programming Interface

W3C World Wide Web Consortium

1

INTRODUCTION

A Good Sketch is Better than a Long Speech.

– Napoleon Bonaparte

This chapter lays the foundation for this thesis. First, the problem this thesis seeks to address is presented, and the current research gaps in the field of data visualization are outlined. Second, the aims and tasks of this thesis are discussed. Finally, the research method used, as well as the structure of this thesis document, is described.

1.1 PROBLEM DEFINITION AND RESEARCH GAPS

Data visualization has become a vital part of our daily life, as it is a better way of representing the information. The data visualization is the fastest way to communicate the information to others. The visualization makes it easier for the people to understand the complicated data, so they can interpret the data in a better way. It is easy to spot the patterns even for the huge volume of data [12]. Data visualization is a tool that helps to easily understand the data patterns and the relationship between data through the visual presentations. People can quickly express their thoughts and ideas through the visualization and can share and communicate it with the others.

The most important aspect in a visualization framework is; how the visualization reflects the information one wants to convey through the visual presentation. The data collection and data analysis are the necessary steps in the process of communication of the data information.

The *first* step is to collect data from different sources in the different format. Visualization is an essential component of research presentation and communication as it can work with large amounts of data and visualize the data into an effective graphics [45]. Today the world is beyond the text data, the volume of data is enormous, so are the data formats to present it. The CSV format (tabular data) is the most commonly used format, that almost every visualization framework supports. RDF[1] is a model to represent enormous and continuous data, RDF is gaining popularity, a few frameworks support RDF format to visualize RDF data.

The *second* step is to understand data and to select the best way to represent it. The information that needs to communicate through the visual image

1 RDF: http://en.wikipedia.org/wiki/Resource_Description_Framework

should be clear. The output in any form of visualization should clearly describe the input data from the user and the developer perspective. To get the best results from the visualization in terms of information, it is imperative to select a correct visualization type. Many of the existing frameworks provide semi or fully auto selection of charts according to the data, but the information about each type of visualization is not considered. Figure 1 points a few *unanswered* questions in the field of data visualization, that are also mentioned in the following list.

1. How to collect data from different sources?
2. Who will consume the data?
3. What type of data format to select?
4. Which visualization type to choose for a particular data?
5. Which visualization tool to select for visualizing the data?
6. How to map the data with the selected visual type?
7. How to access the processed data and the visual image?

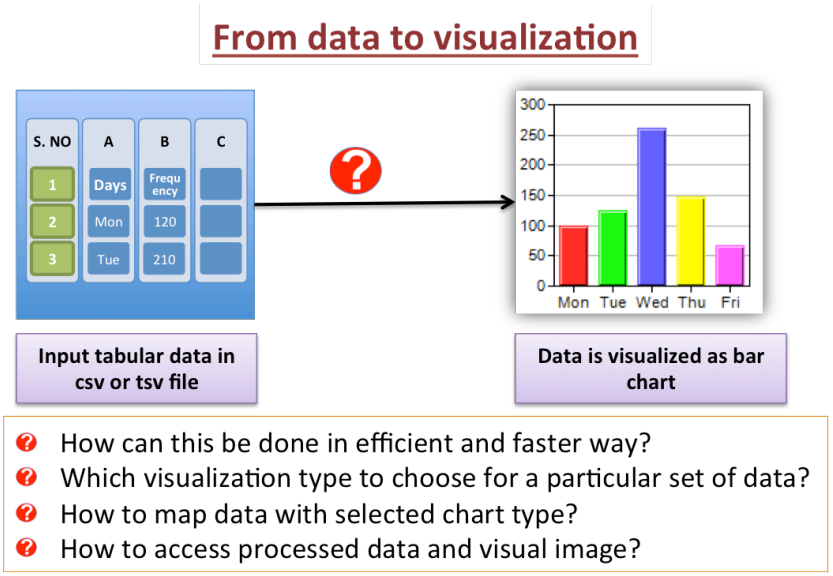


Figure 1: Queries in the Field of Data Visualization

There are many visualization frameworks, applications and tools available in the market to visualize the various types of data. However, only a few frameworks support a broad range of types of visualization and data formats and are very efficient in their work. These few frameworks can address most of the questions defined above, but not all the questions. Even if the existing visualization applications offer semi or fully auto selection of the visualization types according to the data, the information is not clear for each type of visualization. The choice of the visualization tools and frameworks mostly

depends on the user requirements. If the user requirements are not fulfilled, developers in the project end up developing a new framework for the data visualization instead of using the available frameworks, which is not desirable. There are some *research gaps* in the field of data visualization and web framework that are discussed below.

1. There is a broad range of data formats from CSV to RDF. Due to open data, big data, the RDF format is gaining popularity but at the same time the CSV format is used in almost every framework. The problem is that most of the existing frameworks support a few data formats. The RDF data is complex and requires a different mechanism for the filtering and processing than the text data, so there are few frameworks developed which focus only on the RDF data. It means there is a lack of a generic framework that can support many data formats.

2. A review of JavaScript based visualization libraries suggests that there are various libraries that can visualize any data to any visual form. Some of the libraries are best in their work which means using these libraries one can present the data in a desired form in less time. They are just tools, not a framework that the user can execute to generate visualization. The web framework that provides complete package right from uploading of the data to final visual output can use the standard visualization tools. Most of the frameworks develop their visual tools.

3. The most important part of a data visualization is the mapping of data to plot a visualization. If the data mapping is not done correctly, it can ruin the information one wants to convey through the visualization. Most of the frameworks do not provide a flexible and friendly data mapping process, which makes the mapping process time-consuming as the user first needs to understand the mapping process of a particular framework and then initiate the mapping. The information regarding the data mapping is also not clear in most of the frameworks. The data mapping process in the existing frameworks can be improved for better visualization results.

4. The number of visualization types a framework supports varies and there are very few frameworks that support a broad range of visualization types.

5. Not all existing web visualization frameworks support features like custom labels. Custom labels provide more meaningful information to the visual image. Only a few tools that support additional features over the visual image.

1.2 PILOT CASES

In the following sections two projects are introduced as pilot cases, Citi-Sense-MOB[2] and DaPaaS[3], for defining a set of requirements. These projects serve as examples to derive the use cases and subsequent requirements.

1.2.1 *Citi-Sense-MOB*

Air pollution and climate change are critical issues at present and are affecting the whole environment. The two EU projects: Citi-Sense[4] and Citi-Sense-MOB will contribute in raising the public awareness on the link between climate change and air pollution, and on the impact of air pollution on health.

The *primary* objective of the Citi-Sense-MOB project is to develop a new approach and services, mainly the mobile services to make the environment cleaner. The focus will be on Oslo region for a start. The task can be achieved by providing the citizens, and other stakeholders with the information related to the different air pollutant gases and the overall index as a air quality index. In order to provide this information, Citi-Sense-MOB will create and use an innovative technology, which will measure the levels of different air pollutant gases and other parameters. This technology will help to calculate air pollution at a particular location. Then the processed information will be presented through the visualization to the users, developers, and stakeholders, both on the web and mobile phone apps. The application must be designed to support cross-device and platform compatibility. The architecture of Citi-Sense-MOB[4] describes the overall process flow. The use case of Citi-Sense-MOB is further discussed in Section 4.1 to define a set of requirements.

1.2.2 *DaPaaS*

The goal of DaPaaS (data-and-platform-as-a-service) project is to deliver an environment, where the developers can both publish and host data-sets, data-intensive applications. These data can be accessed by the end-users and developers, through applications implemented in a cross-platform manner.

In the current scenario, the volume of data in every field is enormous, so large number of datasets have been publishing in the form of open data, readily available. At the same time, very few applications are there to utilize it, so this project focuses on open data publication and consumption. This project divides the work into layers to accomplish the goal.

The *data layer* (DaaS) will offer a publishing infrastructure by providing components for the data analysis and API's to access the data, and it will work mostly with the RDF data.

2 Citi-Sense-MOB: http://citi-sense-mob.eu/
3 Data-and-Platform-as-a-Service: http://project.dapaas.eu/
4 Citi-Sense: http://www.citi-sense.eu

The *DaPaaS UX layer* will interact with the users and developers through user-friendly interfaces. These interfaces will provide the means for the users, to access, navigate and explore the data both through the open data portals and mobile services. A cross platform prototype will be designed, which will create a user-friendly interface. The use case of DaPaaS is further discussed in Section 4.1 to define a set of requirements.

1.2.3 *Evaluation and Project Tasks*

The projects mentioned above focus on different applications and research areas. The collected data and its types also vary, but both the projects need a visualization component to complete their task. In order to identify the requirements for a framework, which should meet some of the needs of the projects introduced, some example tasks are defined; tasks are labeled as CSM for Citi-Sense-MOB and DP for DaPaaS. Table 1 explains the tasks for each of the projects.

Name	**Task**
CSM1	The system/project displays air pollution parameters and climate change through the different types of visualization
CSM2	A user interface is through the mobile app and supports cross-platform
DP1	The system works mostly on the RDF data analysis and visualization
DP2	The system works on a broad range of data, so the visualization varies
DP3	The system develops a cross-platform prototype for the user interaction

Table 1: Project Specific Tasks

From the above table, it is clear that the projects need a visualization framework, which will support a broad range of types of visualization. The framework should also support different data formats (RDF, CSV, JSON). These points will help to define the actual *problem analysis* of this thesis.

1.3 THE PURPOSE OF THIS THESIS

This thesis seeks to address the problems identified in the *previous* section. The *hypothesis* of this thesis is:

"*It is possible to have a generic visualization framework, that meets the need to support a broad range of charts with improved data mapping technique for cross-platform according to the identified requirements for data visualization, data service, web framework and usability.*"

The purpose of the thesis is to validate the hypothesis stated above, which will address the problem defined. The thesis work is further subdivided into six research tasks, which follows the standard technical methods. When the thesis will finish all the identified tasks, it is evaluated to check whether the hypothesis is valid or not.

1.4 RESEARCH METHOD

The method of work used for this thesis and the related development is based on the method for *technology research* [39]. Based on this method, the thesis will undergo all the following steps:

PROBLEM ANALYSIS This step describes that there is a potential need for a new or improved *artefact*. The problem defined in this thesis is that the current field of data visualization is lack of a genric web-based visualization framework that supports all popular types of visualization and data formats with a friendly mapping process. There is a need to develop a new design i.e. vocabulary design, to have a better mapping process, then using this vocabulary and other standard libraries to create a new framework (*VoVis*). A list of *requirements* for the *VoVis* framework will be identified based on the web framework, data, and visualization components and also from pilot projects of SINTEF.

INNOVATION The artefact that will be designed and implemented, is a vocabulary based visualization framework also known as *VoVis*. It will possess a vocabulary component, standard JavaScript libraries, a user interface written in HTML5 and JavaScript and a controller to execute the process flow. This artefact seeks to address the gaps in the data visualization research as discussed in Section 1.1 hence to validate the hypothesis stated in Section 1.3. *VoVis* will be developed and implemented based on the set of requirements identified in the background research study phase.

EVALUATION After the development and implementation of the *VoVis* framework, an experiment will be performed to evaluate whether the *VoVis* framework meets the set of requirements. If the results of the experiment will satisfy these requirements fully or partially, it will then be concluded that the *VoVis* framework has fulfilled its tasks, validated the hypothesis of this thesis, and closed few research gaps in the field of data visualization as identified in Section 1.1.

1.5 RESEARCH TASKS

The three major steps defined in the above section are further divided into six research tasks. In this section, the research tasks have been identified and discussed as a part of the process that this thesis will follow.

"Visualization[5] today has ever-expanding applications in science, education, engineering, interactive multimedia, medicine, etc."

The *first research* task is to have a conceptual knowledge of the data visualization and data analysis concepts in order to identify the types of visualization and data formats that the thesis will support. The *primary* task of the thesis is to find a better visualization technique, which requires a deep knowledge and research in the field of data visualization. The first task focuses on the identification of data sources with the different formats that need visualization.

The *second research* task aims to focus on the data visualization through the web application. The primary goal is to identify the components for the data visualization in a web/based framework.

The *third research* task is the identification of requirements based on the requirements of the use cases that discussed in Section 4.1 and the problem defined in Section 1.1. These requirements need to be fulfilled by the existing frameworks.

The *fourth research* task is to focus on the review and the evaluation of the existing frameworks and visualization tools, on the basis of the set of requirements defined. This review will help to know which framework is better by fulfilling all the requirements or part of it.

The *fifth research* task is concerned with the design and implementation of a better and improved visualization framework if existing frameworks do not fulfill all requirements. The visualization tools that are better can be used in the new framework.

The *sixth and last research* task is the evaluation of new framework through an experiment that is designed to evaluate whether the given framework fulfills the requirements set earlier in Chapter 4.

Summarizing the previous points, the research tasks for this thesis consist of the following steps:

A. Understanding the data visualization concepts.

B. Analyzing the notion of data visualization through the web application.

C. Listing a set of requirements for each component of a web visualization framework by referring the pilot projects.

D. Evaluating the few existing frameworks, techniques, and tools for visualization. If none of the existing frameworks can fulfill the requirements, then a new framework needs to be designed.

E. Designing and implementing a new visualization framework to satisfy the needs.

F. Evaluating the new framework with an experiment in order to verify that the new visualization framework can close the research gaps, and to validate the hypothesis stated in Section 1.3.

5 Visualization by Wikipedia: http://en.wikipedia.org/wiki/Visualization_(computer_graphics)

1.6 THESIS STRUCTURE

The structure of this thesis document is comprised of four parts that are broken down into nine chapters.

1.6.1 *Part I: Background Study*

Part I (chapters two to five) is concerned with the background research for the conceptual framework of this thesis. Chapter 2 addresses the concept of the data visualization and how it is classified. Chapter 3 discusses the components in a visualization framework. Chapter 4 is concerned with the identification of the requirements for the development of a visualization framework. Chapter 5 this chapter presents an assessment of the existing web-based visualization frameworks on the basis of the requirements from the previous chapter.

1.6.2 *Part II: The VoVis Framework*

The second part (Chapter six and seven) describes the development of a vocabulary based visualization(*VoVis*) framework and its implementation. Chapter 6 outlines the architecture of the *VoVis* framework and describes each of the component in detail. An essential component: vocabulary library is defined in this chapter. Chapter 7 discusses the implementation of the prototype of the *VoVis* framework both as a web and mobile app.

1.6.3 *Part III: Evaluation and Conclusion*

The third part (Chapters eight and nine) presents the results of the experiment carried out to evaluate the *VoVis* framework and discusses the contributions made by *VoVis*. Chapter 8 describes the experiment based on a case study from Citi-Sense-MOB project. Chapter 9 concludes the thesis by providing a summary of the thesis and discusses its contributions in the field of the data Visualization. A section on future work, explains the potential avenues for further research.

1.6.4 *Part IV: Appendix*

The last part of this thesis document contains - Appendix A: The *VoVis* vocabulary for types of visualization. Appendix B: A review of different JavaScript Libraries [59] and Appendix C: The *VoVis* Prototype Web Application manual.

Part I

BACKGROUND STUDY

2

DATA VISUALIZATION

This chapter addresses the concept of data visualization, data visualization types, and the process of visualization. The different types of data are discussed for the data visualization, various software techniques in the field of data visualization are explained. The visualization serves two major purposes, the first is data analysis [37] and the other is data presentation. The focus is on the data visualization that will present the data in different graphical forms.

2.1 WHAT IS DATA VISUALIZATION ?

"Data visualization refers to any graphic representation that can examine or communicate the data in any discipline" [19]. There are many ways to define the data visualization according to the different fields that use data visualization. Data visualization is the presentation of data in a pictorial or graphical format [47]. It is also viewed as a modern equivalent of the visual communication. Data visualization is both an art and science, it is the best form of communication as the human brain processes visual form much faster than the other forms. People can grasp the meaning of the data easier when they are displayed in some form of visualization instead of the data in text form. The main reason, why data visualization is booming in the market is the presence of enormous data volume that is increasing each day. The below statement taken from [11] provides some information about the amount of the data in the social network sites.

"Google receives more than two million search queries in a minute. In that same minute, more than 3,600 new photos are shared by users on Instagram and 684,478 content items are posted to Facebook", so it is important how these data can be presented.

Scientists and technical researchers work on the raw data as part of their research, but the rest of the world needs the data in some processed form in order to understand the massive data and its relationships. The one way to present the data is through the spreadsheets as rows and columns. The amount of data being produced is enormous in volume so it is not appropriate to present these data through the spreadsheets. It is hard to trace and understand the data in such a format, difficult to read as it consists of thousands of rows and columns. The best way is to visualize the data in an absolute form. The companies are using the data visualization to learn the business trends.

The students, developers, and researchers are also working with the data visualization. It is the age of visualization where each kind of data (from 2D to Big-data) is visualized in a broad range of possible visualizations.

Data visualization makes it easier for people to understand the complicated data. People can quickly express their thoughts and ideas through the visualization and can communicate with others. "Research from Massachusetts Institute of Technology and Harvard University suggests that people find faces and human-centric scenes to be easier to remember than landscapes" [11]. The color visualizations are better than the other visualizations, as they are more interactive and easy to remember [38]. It is also found that the visualization type like arc and tree diagrams are more memorable than the common graphs. The most important aspect of the data visualization is that the visualization should be accurate and easy to understand. The topic of data visualization has been explored and explained through a range of books, a few of them are "Handbook of data visualization" [9], "Interactive Data visualization" [44], "Designing tables and graphs to enlighten" [18]

Interactive visualization [47] are the visualization types that provide fine details of the data and are interactive in presenting the information, not like the static graphs and spreadsheets. A few examples of the interactive visualization are charts, tree map, geo maps and many more. Data visualization makes the interpretation easier and saves the time and the energy. A definition of the visualization from [46] states that every visualization should at least follow these three minimal criteria.

1. *Based on (non-visual) data* - The purpose of a visualization is to communicate the data, the data needs to be in a abstract form. The visualization can transform a organized or unorganized non-visible data to a meaningful and visible structure. The visualization conveys the useful information to the user in a desired form.

2. *Produce an image* - The most obvious outcome of the visualization is an image, but it is not always clear. The visual image should be the primary means of communication. If the image is only a small part of the process, it is not a visualization.

3. *The result must be readable and recognizable* - The visualization must provide the correct information that the user wants to communicate through the data. Sometimes the visualization leaves out the important information and deviates the result which makes it difficult for the user to understand the underlying data. the result. so there should be some relevant aspects of the data which can be read. The visualization must be clear, readable and usable.

Data visualization can further be divided into three categories according to the data each can process [41]. These categories are described in the following sections.

2.2 SCIENTIFIC VISUALIZATION

Scientific visualization presents the scientific data that are well integrated with the real-world objects having spatial properties. The data is available from many sources such as engineering, mathematics, medical and many more. This visualization focuses mainy on the realistic renderings of volumes and surfaces of the 3D data [21]. Scientific visualization mainly presents 3D volume data; an example is 3D volumes generated from the MRI and CT Scan. Multidimensional Multivariate visualization is an important sub-field of the scientific visualization [58]. Scientific visualization works with the real life data, and there are many challenges in this field, as it involves the complex graphical structures [24]. The various scientific fields often have very specific conventions for generating the types of visualizations [36].

2.3 INFORMATION VISUALIZATION

The field that studies the visual representation of various forms of raw and processed data (ranges from generic graph, tree structure, tabular data, text format and computer software) is known as information visualization. In a broad term, *infovis* covers mostly the statistical types of visualization. In recent years, there is a tremendous growth in the information visualization technology, both for the commercial and personal use. The users need tools to design and create their visualizations from the datasets [27]. Since the last decade the digital artifacts are growing in number, size and types also termed as "Big data" which acts as main catalyst for the growth of interest in field of the information visualization [41].

2.4 VISUAL ANALYTICS

In the field of data mining, there is a need for new discipline to focus on the processes and datasets that are either too large, or too complex. Visual analytics has emerged from the information visualization, scientific visualization, and data-mining communities [41]. The main work or goal of the visual analytics is to provide tools for the data mining and data analysis by means of the interactive visual interfaces.

Visual analytics is a multidisciplinary field that includes analytical reasoning techniques, visual representations and interaction techniques. These techniques help the users to understand deep insights, as the human mind can easily understand the complex information if received through the visual channels. Visual analytics is designed to facilitate the analytical reasoning process [42].

Figure 2 shows all the three categories of the visualization [41].

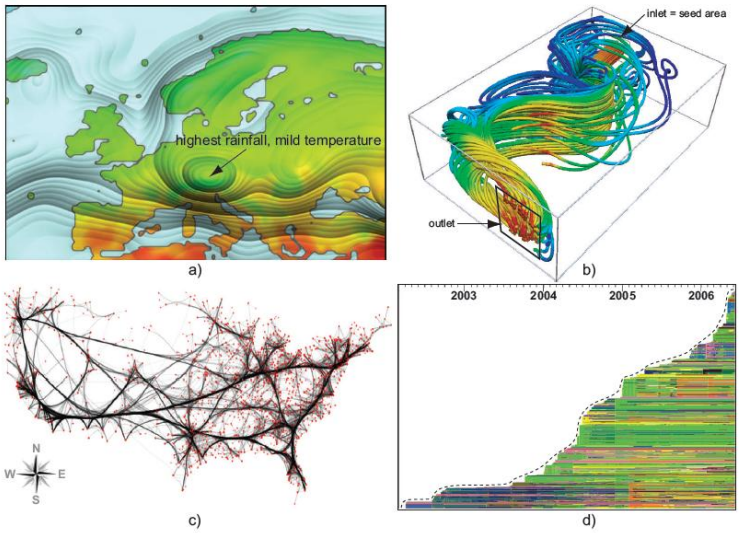


Figure 2: Visualization Categories based on the data. a)and b) An example of the scientific visualization. c)An example of the information visualization. d) An example of the visual analytics visualization.

2.5 DATA VISUALIZATION IN DAILY LIFE

Data visualization is rapidly increasing as the amount of information is vast and overwhelming. It is in every field of education [17], medical science [1], entertainment, and business [25]. Each type of data can be presented in a particular visual form. A few examples are outlined below [44]

- Some articles in a newspaper are discussed as a table.
- Weather chart shows the heavy rainfall information over an area.
- Train or Subway lines map helps people to track the way.
- In business the stock market is presented over different charts and graphs.
- In medical science, MRI and CT Scan are helpful for diagnosis of several diseases.
- In education system as mechanical design of some process or simulation of the complex process.

2.6 TYPES OF DATA

This section describes the various types of the data, for every data set, the data record carries its piece of information. These data records can be further classified in two different subgroups [44].

- **Ordinal** - This group covers the numeric set of data. These set of data are quantitative in nature. Any set of data which can be measured or counted are the ordinal data
    A. *Binary* - Binary data [53] is defined as the data with only two possible states. The binary numeral system and boolean algebra termed this states as 0 and +1.
    B. *Discrete* - This is a form of numeric data that can have only precise values (no fractional value). A simple example of this data is: Number of kids in a family 2 (we can not say 2.5 kid or 1.5 kid either we can say 2 or 3).
    C. *Continuous* - This is a form of numeric data where the values can change continuously, it is not possible to count the number of different values. It can be shown as fractions, decimals, and it can have many values between the two continuous numeric values. Example: the measure of height in a classroom - feet, inch, meters, centimeter, and millimeter
- **Nominal** - This group covers the non-numeric set of the data. Labels are used as a variable for scaling of the nominal data. The nominal data are mutually exclusive.
    A. *Categorical* - This form of data consist of at least 2 or more groups of data which can be easily divided using some labels. For example sex of student in a classroom: male -10, female -15.
    B. *Ranked* - This is another form of a categorical data, where the logical groups are used for arranging the data. For example size of shirts extra small, small, medium, large.
    C. *Arbitrary* - This form of data can have infinite range of values, it can not be logically arranged or grouped for instance.

2.7 DATA VISUALIZATION PROCESS

The data visualization process [41] is a complex interaction process that involves the user to provide the input data, and the visualization application to produce the visual images. The task of this process is to visualize the raw data through the following steps. Figure 3 outlines the process flow of a visualization.

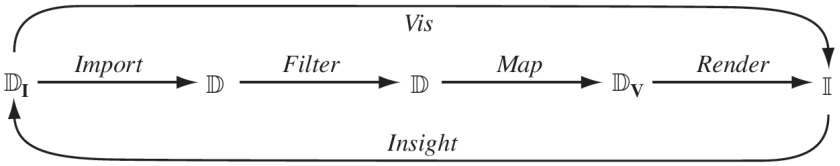


Figure 3: Steps Involved in the Process of Data Visualization

2.7.1 *Importing Data*

First, the input data needs to be imported into the visualization process. Input data has a specific format, so the types of format that the process supports must be defined. There is a wide range of data formats that can be supported by the application. The data varies as there are many types of data discussed in Section 2.6. If the application does not support particular data type or format, then before importing the data needs to be transformed to the supported type and format. Data import is a one-to-one mapping, for example uploading the input data from the external storage such as a file or a database. Sometimes the data needs to be translated or converted from a continuous to a discrete form.

2.7.2 *Filtering Data*

Once the data is imported the next step is to look into the important features in datasets that are interesting for the user. The raw data is filtered into a more appropriate form (filtered data). These filter data can be analyzed and visualized. This process of transforming the data to a filtered form is called filtering that helps to extract the relevant data from imported data. The two main reasons for performing filtration on the data are as follows.

1. *What is relevant-* It is important to know what data is appropriate as sometimes only a subset of the given data is relevant. In that case the whole data set is not required, so the data is filtered and only relevant information is considered for visualization.

2. *Large Data* - Sometimes the imported data is huge in volume so it is not possible to visualize the whole data. The visualization process can become complex if the data exceeds the size limit.

2.7.3 *Mapping Data*

Once the data is filtered the next step is mapping of the data with the visual domain to form a processed data that comprises of the visual information. These visual features can be axes, color, size, etc. The mapping is an important step in the process of visualization as it provides better information about visual features and is user-friendly. Following are the two reasons for supporting the mapping process.

1. *Purpose*: Mapping provides a clear picture about the visualization of the data. It helps the users and the developers to understand the concept of visualization and types of it. Mapping in a broader way converts the raw data into a informative data with visualization features.

2. *Modularity*: Mapping provides modularity in the process of visualization by separating the modules. The software can be reused by mapping steps.

2.7.4 *Rendering Data*

The last step of the visualization process is rendering of the data into a visual form. This step provides the final visualization. The result of the mapping step i.e. mapped data is presented using the visualization tools. Rendering means plotting of the data into a graphical form. The process can render from a simple bar chart to the complex maps. The final rendered image should be clear and simple for all users and developers.

Thus, this chapter discussed the data visualizations, types of data, followed by the discussion on the visualization process and its steps. Next Chapter 3 focuses on the different components of the visualization.

3

DATA VISUALIZATION TECHNIQUES AND FRAMEWORKS

This chapter focuses on the different visualization techniques and components, and guidelines for an effective visualization design. The Visualization components are discussed in detail, in order to define requirements for the evaluation of existing frameworks. This chapter is referred from the book [44].

3.1 DESIGNING EFFECTIVE DATA VISUALIZATIONS

An effective visualization can improve the communication within and across disciplines and conveys the information effectively [26]. The primary goal of any visualization technique or tool is to design a successful visualization. There are many parameters for a "successful visualization". The visualization should accurately conveys the information to the end users. There is a healthy competition among the different techniques and tools. The users selects those tools and technologies that provide valuable visualizations. A few important aspect in visualization are: how to map the data to the graphical form, which type of data and how much data to visualize, with some additional features and labels. There are many factors that need to be considered while designing a visualization technique. Some ways to develop an effective visualization [44] are as follows.

- *Intuitive Mappings* from the data to a visualization means the visualization should fulfill user expectations. In order to achieve that, it is important to consider the semantics of the data and context of the user. Intuitive mappings can reduce the translation time and, thus provide rapid interpretation.
- To provide the users with a broad range of visualizations for the given data.
- Views should be *clear, attractive, functional* and at the same time informative.
- The *information density* should always be balanced. Too much information should not be displayed.
- Additional features like *labels, keys, and color controls* can be included in the views, which helps the user to interact easily.

- The view should be focused and balanced. The display screen should be used effectively.

3.2 DATA VISUALIZATION FRAMEWORKS

The different fields of visualization and types of data are discussed in the Chapter 2. The visualization systems can be domain specific or data specific. For each of these categories there are various tools based on types of data. It is not possible to cover all the types of tools and is out of scope of this thesis. This thesis focuses on the information visualization.

A visualization software can also be classified as libraries, frameworks, and turnkey systems. The visualization libraries provide a set of functions for the data types and rules for mapping and visualizing that needs to be used with the third party tools to produce the visualization. Turnkey systems are systems designed for a particular task only, it can not be considered as framework for general visualization and have their user interface and specific rules. An application framework is a complete package. The framework has it own user interface, libraries, controllers, and other components. Framework can be extended by adding new components. The task of the framework is to generate visualization from the data uploaded by a user through the user interface.

There are many frameworks available in the field of information visualization. Each of the framework uses different technology, services, and tools. Most of the frameworks require a software to be installed on the client. The web-based frameworks are selected after reviewing many frameworks. The main reasons to choose web-based frameworks are as follows.

1. *Web-based applications* and tools are very popular in today's world. Every research field is using it, in some form.

2. Different web-based framework can choose the various tools and programming for the back end, but they all have common user interface through a web browser. The comparisons between such frameworks can be easily done.

3. The most *important advantage* of using web-based framework is, no software installation required, so the framework can be used from anywhere. The users can just access it through the browsers regardless of the operating system (Windows, Mac) and device (laptop, desktop, tablet), so the web-based framework saves time, effort and space.

3.3 COMPONENTS OF DATA VISUALIZATION TECHNIQUES

When the user wants to visualize the data, the first step is to select a visualization technique. The technique that provides the best result is selected, so the evaluation of techniques is necessary. A particular technique can be opted according to the task the user wants to accomplish, or the type of data. To compare the visualization techniques or to evaluate it, a few components

are defined in visualization techniques. These components are discussed in the following sections.

3.3.1 *User Component*

The user plays a crucial role in the visualization process. The success of a visualization process is highly associated with the user. The user is the one that provides data to be visualized through the user interface and collect the final visualization in a graphical form. The user are classified based on the knowledge they possess.

They are discussed below:

- *Familiarity with Data* - The knowledge of the user in the field of data is tested here, how familiar is a user with particular kind of data.
- *Familiarity with Domain* - How much knowledge the user has for the domain of data to be visualized. Is the user a domain expert or just new in the particular field.
- *Familiarity with Task* - This is about the task experience, how much experience a user has to perform a task.
- *Familiarity with the Visualization Technique* - How familiar is the user with the particular visualization technique. Is the user using this method for a long time or not.
- *Familiarity with the Visualization Environment* - There are various ways in which visualization technique can be employed, so the knowledge of visualization environment is considered.

3.3.2 *Data Component*

Data component plays a significant role in the evaluation of the visualization process. Data is the input that is visualized in a graphical form, and there are various features of data that need to be considered. They are discussed below.

- *Type* - Data can be of same kind, different kind or combination of both. The data is a mixture of various kind like numbers (ordinal type), names (nominal type). Data are also classified as continuous and discrete.
- *Size* - The data vary in a broad range of size from few records of dataset to thousand of records. In order to visualize the massive data more filtering and sampling is required, also the visualization requires some additional features like zoom in.
- *Dimensionality* - Data can be one dimensional, two or multidimensional. Different visualization patterns are selected based on the dimensions of data.

- *Number of parameters* - The number of parameters also varies in datasets from univariate to multivariate; all the parameter need to be considered for visualization.
- *Structure* - Structure of the data varies from the simple structure (tabular form) to complex (hierarchy, network form). The format of data depends on the structure of it. The simple tabular data can be represented as text (CSV) format, whereas the complex structure like tree hierarchy is represented as XML or JSON format. The RDF data is also gaining popularity due to open and big data.
- *Range* - Data can have a broad range of values.
- *Distribution* - Data can be distributed, either in uniform or nonuniform way.

3.3.3 *Visualization Component*

This is the an essential component of a visualization technique. All the logic, operations, and mappings required to generate a visualization along with the tools are part of this component. It takes all the decisions related to visualization. This component executes the visualization process once the task, user, and the data components are defined. The evaluation of user interfaces is accomplished by this component [20] to detect the design problems in layout. Following are the few important aspects of visualization components.

- *Computational performance* - The time a visualization process takes to generate the visual image affects the performance, so for the better performance visualization of the data should not take long time.
- *Data Limitations* - Sometimes the data is huge that need to be visualized, but there is a size limit for data in terms of presenting it on the web. The limits should be clearly defined and if the data exceeds this limit some alternative steps need to be taken.
- *Degree of complexity* - The user and the developer can interact easily and perform the task in efficient manner if the visualization process and tools are not complex. A user-friendly framework with less learning time makes the visualization efficient.
- *Degree of accuracy and usability* - Performance is an important factor but at the same time the visualization should give an accurate result, it should not deviate from the requirements. The user can perform the task successfully with this technique or not, will decide the degree of accuracy.

Thus, this chapter discussed the ways to design an effective visualization, components of visualization technique and different software tools and applications for visualizations are outlined. The web-based framework is selected for this thesis.

4

REQUIREMENTS FOR A DATA VISUALIZATION FRAMEWORK

The requirements for a web visualization framework need to be identified, in order to evaluate the existing solutions or to design a new framework. These requirements are derived from use cases of the pilot cases. The set of requirements are defined for each component of the visualization technique.

4.1 REQUIREMENTS ANALYSIS AND USE CASES

The two pilot projects defined in the Section 1.2 are discussed in detail for defining the use cases. DaPaaS and Citi-Sense-MOB have use cases that help to define the set of requirements for the framework.

4.1.1 *PLUQI : DaPaaS Use Case*

The DaPaaS report "Use case definition and requirements analysis" [29] describes the requirements for the DaPaaS project and defines the use case (PLUQI). The use case should demonstrate the concept of integrated DaaS and PaaS and the strength of the DaPaaS architecture.

Personalized and Localized Urban Quality Index (PLUQI) provides a customizable index model over the mobile/Web application. It can represent and visualize "the level of well-being and sustainability" for given cities based on individual preferences. PLQUI [2] is also an application deployed and hosted in the DaPaaS platform, and the end users can access it on the web and via smartphones. Figure 4 shows the concept of PLUQI use case. It has a visualization component that presents the indexed data from different sources. The requirements related to the visualization of PLUQI provides a base for requirement analysis of the framework.

The functional visualization requirements that are needed to visualize the PLUQI data are as follows.

- The visualization system should provide functionalities for viewing full datasets or previewing parts of datasets with the adapted visualizations.
- Visualization should provide support for the tabular forms, charts (line, plot, histograms etc. for displaying 2D data), time series, and plotting the data on a map for geo-spatial data.

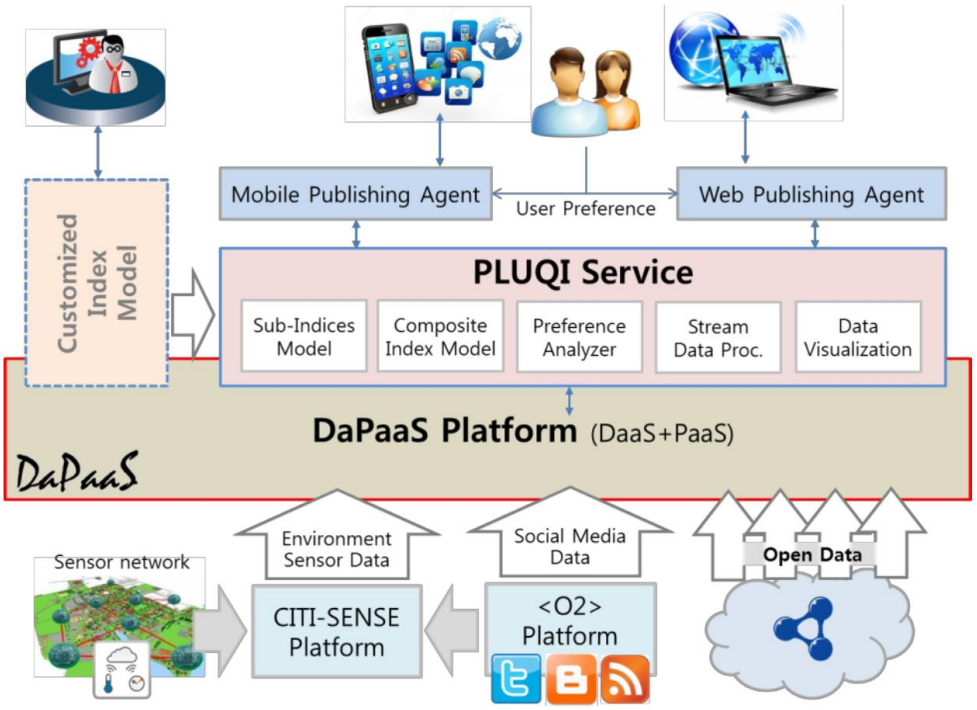


Figure 4: Conceptual Diagram of the DaPaaS Use Case (PLUQI)

The non Functional requirements for the PLUQI are as follows

- The visualization components should be characterized by a good response time.
- Web client shall be supported by major browsers.
- Mobile client shall be supported by major device OS.

In order to fulfill the above requirements a web application is implemented that visualizes the data of PLUQI. The PLUQI indices are presented in different types of visualization from a simple bar, table to a map. According to various indices different visualizations are plotted. Figure 5 shows the home page of PLUQI prototype taken from [2] that visualizes the data on a map with markers.

4.1.2 *Citi-Sense-MOB Use Case*

The Citi-Sense-MOB use case provide a mobile app for presenting the AQI index. It is an index for reporting air quality which determines the level of air pollution, how much the air is polluted in a locality and how it can affect the people surrounding it. This index helps to monitor air pollution in a locality. The aim of this use case is to visualize the AQI index and other air pollutants on the mobile and web. Some reference code like color code is used for visualizing different levels of air pollution. Citi-Sense-MOB has defined a color code according to which the air pollution levels is displayed (green=clean;

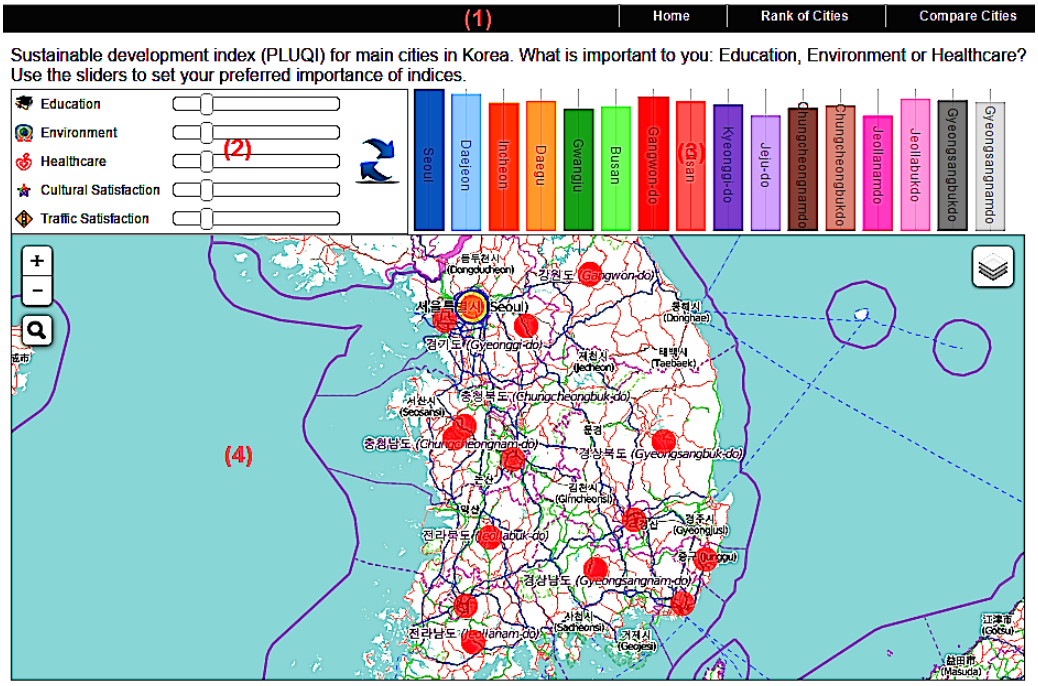


Figure 5: PLUQI Home Page Visualizing the Data in Different Types of Visualization

yellow=moderate; orange=unhealthy for sensitive groups; red=unhealthy for all).

Data can be gathered by two the approaches, first by sensors mounted on the bicycle which provides the location, time and levels of pollutants. This data helps to understand the levels of pollutants on a particular route at the particular time. The second way is to gather data from mobile phones and from fixed sensors mounted at different locations.

One way to present the data is to plot it on the map showing either the sensors at different locations or air pollutants level at each location. Figure 6 and Figure 7 are suggested types of visualization to present the data from the Citi-sense-MOB project on mobile phone and web [3]. Another way is to focus on individual air pollutant gases and other factors like temperature and noise level. Functions like comparisons of different gases or range of it over time. Figure 8 shows the comparison of NO gas from different sensor sources taken from source [3].

4.2 VISUALIZATION REQUIREMENTS

Visualization is the backbone for the design of visualization framework. In a framework, it is imperative that the user selects the correct type of chart for visualization and can also access the processed data. Following are the requirements in this domain.

- **R1** - *Possibility to Support All Popular Types of Visualization*: There is a huge range of visualization types, it can be simple 2D charts like the bar chart, line, pie or can be complex like treemap, arc diagram, and paral-

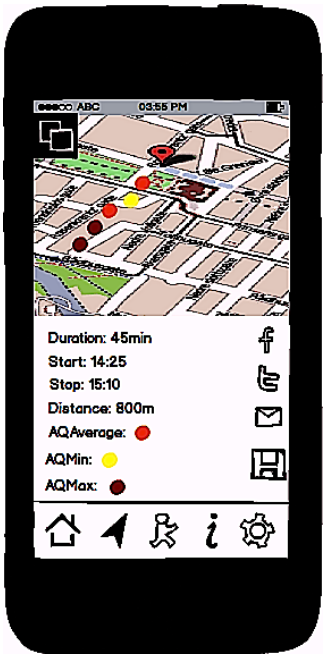


Figure 6: Citi-Sense-MOB App Visualizing the Air Quality Data through Map

lel sets. Geo charts and maps are also types of visualization. These are categorized according to the function they perform or by the type of data they present. The use cases discussed earlier need a broad range of visualization from 2D charts to Geo Charts, so the framework should support all popular types of visualization, or there should be a possibility to add new ones.

- **R2** - *Standard Third Party JavaScript Libraries*: Every visualization framework needs visualization tool to generate visualizations. Designing and developing such tool requires an enormous effort. Instead of creating such tool, existing tools can be implemented in the framework. There is a broad range of JavaScript visualization libraries available, some of it have excellent performance, reuse of such library reduces the effort and time. More than 30 JavaScript libraries are reviewed as a part of the evaluation and few of them are discussed in detail in Section 5.2. JavaScript libraries are flexible and independent, so it provide an opportunity to work on different platform and devices. They also offer a lot of custom features which can make the visualization very presentable and creative. The Citi-Sense-MOB project has requirements to plot their air quality data on the maps with few creative features, that can be fulfilled by using one of the JavaScript libraries.

- **R3** - *Local Access To Charts*: The framework should provide services to the end user and the developer. The user always wants access to the final visual image to reuse it. The developer may want to access visual results as data to work and process further, or just to forward to some other framework. Mobile device applications also need the final visual

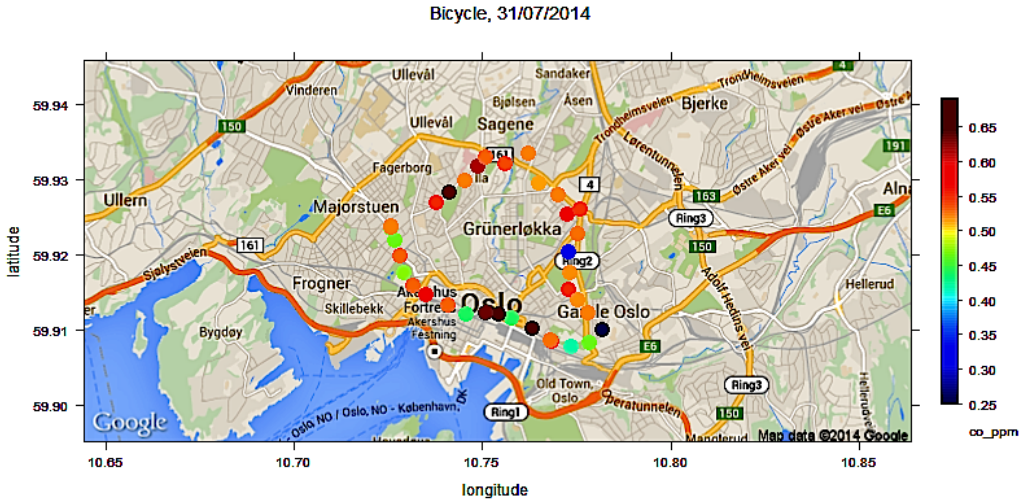


Figure 7: Map Visualization on Web Visualizing CO Values at Different Locations Tracked by a Bicycle

image as picture form(.png), so it is important to have local access to charts and graphs. Both the pilot cases require a mobile app version of their framework.

- **R4** - *Data Mapping using Generic Visual Vocabulary*: Data Mapping process can be executed automatically or semi-automatically by the framework or can be manual. The users and the developers prefer manual as it gives more customized output. The mapping process should be user-friendly, so that mapping can be done in less time. If the framework uses some general vocabulary for mapping, it is convenient for the users to understand and follow it. Every chart type should have its vocabulary, which makes it self-explanatory. DaPaaS and Citi-Sense-MOB need to present their data in different visualization types that require a lot of mapping process.

4.3 DATA SERVICE REQUIREMENTS

This defines the requirements related to the different data services such as types of a data format the framework should support. Data security in terms of data sharing and restrictions should be considered. Massive data should be uploaded quickly. The most important is to have processed data as output.

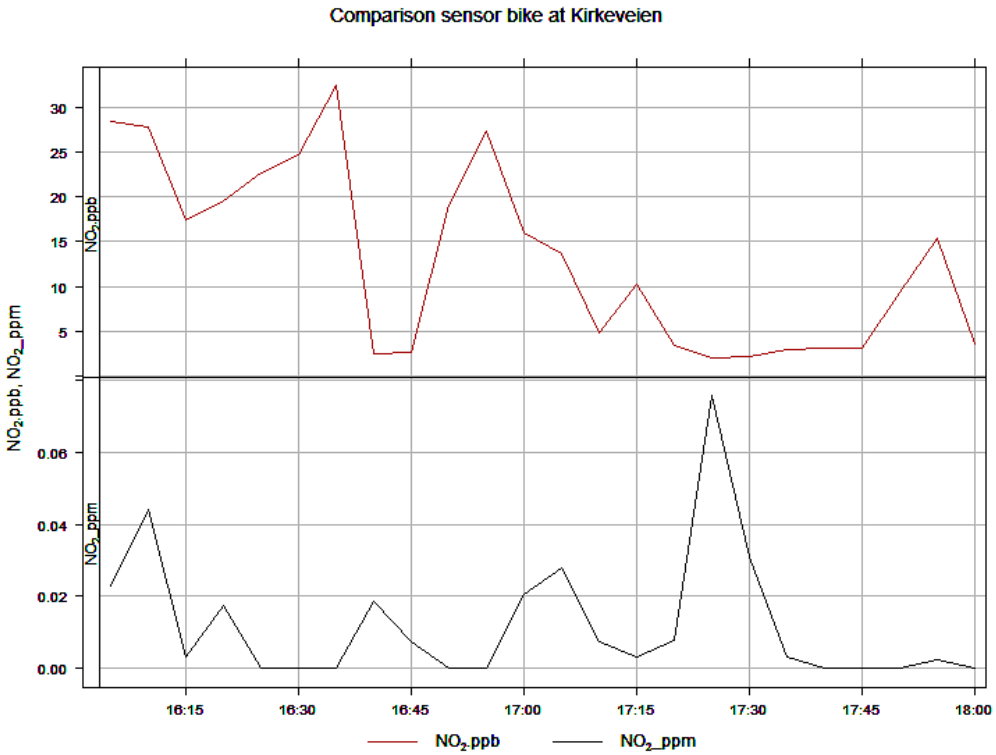


Figure 8: Comparison of NO2 Gas from two Sources through Line Graph Visualization

The transformed data should have the information about selected visualization type. The requirements are discussed below

- **R5** - *RDF and CSV Data Format as Input*: For any generic framework, it is important to support many formats for input data. According to a specific project needs, one or more format can be opted. But it is not feasible to support all. What is more essential, is to have correct types of formats. The most common format is CSV and considering Open data(Linked data), it is very beneficial to have RDF format. DaPaaS project mainly focus on RDF data so to support RDF format is a basic requirement in this project.

- **R6** - *Data Privacy*: Some times the user wants the data to be restricted not publically accessible. As the framework provides services to a wide range of customers from a lay user, developer to business clients. Every user may have different requirements in terms of data privacy, so the framework should offer data privacy.

- **R7** - *Easy to Upload Data*: It is easy to upload small data but most of the real and sensor data are enormous in volume. So the framework should also support uploading os large size files.

- **R8** - *Access to Processed Data in Standard Format*: The end user and the developer should be able to access the processed data in a standard

format such as JSON objects. Presenting the data as visualization is not only the target for all applications, some framework may need processed data for further processing so as input to another process. One of the requirements of PLUQI to have processed data as output.

- **R9** - *Processed Data Comprise of Visualization Information*: The developers work with the data and process it, so it is vital to have information of visualization type along with data.

4.4 WEB FRAMEWORK REQUIREMENTS

This section will cover all the basic requirements necessary for a web or desktop based framework. The language selected for framework always affects its performance. The web framework requirements are discussed below.

- **R10** - *Programming in JavaScript*: JavaScript is the most common language and has a smooth learning curve. The best part is there is a wide range of standard libraries in JavaScript which can be reused easily.
- **R11** - *Offline Mode*: To have internet access all the time, is not feasible. The work should not be hindered when the user wants to work offline.
- **R12** - *Cross Device Compatibility*: The framework should support cross platform devices. Most of the visualization libraries works well on the web browser, but the user wants a consistent experience across mobile devices as well. This is because of changing user habits in recent times. the Citi-Sense-MOB use case is primarily developed for mobile devices, so cross-device compatibility is one of the main requirement that needs to be fulfilled.
- **R13** - *Open Source*: Open source tools have numerous advantages over the closed tools. A few benefits are security, quality, Customizability, freedom, flexibility and many more. Open source provides better quality as thousands of developers work on it and provide new innovative ideas. There is a huge support group where actively problems are discussed and solved by the experienced developers. Bugs in the open source software also tend to get fixed immediately. The users and the developers have complete access to the tool(open code), so it is easy to modify or add new features. If the framework is open source, it can be referred by other open source frameworks.
- **R14** - *Cross Browser Compatibility (with no Flash Plugin)*: A strong web-based framework is one that supports all popular types of browsers and does not depend on an external tool like a flash player. During the creation of a web framework, the framework should support cross-browser compatibility. The framework should behave in the same manner for all popular browsers. The design should work properly without errors for every browser that user choose to work with. It is a very complicated aspect that needs to be considered while designing the web

framework. Now everyone is using a different browser, also depends on which operating system is used. A few popular types are like Firefox, Safari, Chrome and Internet Explorer.

- **R15** - *Framework Should be User-Friendly*: The learning curve should not be deep, it should be user-friendly otherwise there will be a drop in the use of the framework. If the framework is not friendly, users and developer will spend more time and effort that otherwise could be utilized to perform some better task. Usability directly affects the efficiency of the framework.

4.5 USABILITY REQUIREMENTS

This section outlines Usability requirements for web or desktop based framework. These requirements focus on overall performance and design of the framework and how it can be customize according to the need of the hour.

- **R16** - *Customizability*: The charting framework should be flexible enough to change or modify. There are many of things that user want to try, like adding custom shapes, attaching events (click, hover, key press), and applying themes, etc. To create an elegant design, it is good to have a library that is easily customizable and can be molded according the design or requirements of the user.

- **R17** - *Performance*: Performance is dependent on many factors like time to upload data, time to plot chart, the size of the library, memory usage while rendering, garbage collection and number of browser repaint cycles. The value of the performance for any framework is always high. Some place where memory size is not as important as proper rendering of the chart like a dashboard used on the desktop, but for mobile devices size is a major factor.

- **R18** - *Design and Interactivity*: The design is more than just look and feel of a chart. The visualization should not only look presentable (themes, color scheme), but it should have a meaningful interaction. For example, while building a pie-chart, clicking a pie should pull out some information by default. Clicking a icon in the multi-series area chart should toggle the details of related data.

Thus, a total of 18 requirements are defined and discussed based on the use cases and components of visualization, in order to evaluate the existing web visualization frameworks.

5

EVALUATION OF VISUALIZATION FRAMEWORKS AND TOOLS

This chapter focuses on the different visualization frameworks available and their comparisons and evaluations, followed by the evaluation of the different standard JavaScript visualization libraries.

5.1 VISUALIZATION FRAMEWORKS (WEB-BASED)

This section contains descriptions of four different web-based visualization frameworks: Many Eyes, Visualize Free, Data Wrangler, Tableau Public and Weave. All four frameworks are possible solutions for the current problem - either as they are or with a further development of the frameworks. To evaluate the strength and weakness of each system, all the four systems are described, compared and analyzed in this section. The evaluation is done considering the requirements identified in the Chapter 4.

5.1.1 *Many Eyes*

Many Eyes is a web visualization framework, by IBM, which offers easy visualizations. The user can either upload data or use the existing pubic data. Many Eyes offers many visualization types, such as Scatter plot, Matrix Chart, Network Diagram, Bar Chart, Block Histogram, Bubble Chart, Line Graph, Stack Graph and many more. Each data set uploaded in the database of Many Eyes framework is available to the public, so the data is not private. The uploading of large data takes longer time and also prone to error. The duplicate datasets are uploaded and stored in their database, the data is not secured.

The website of Many Eyes explains the framework as: "No programming or technical expertise is needed, so almost everyone has the power to create visualizations. Need to simply follow the three steps."

- Upload the public data set. Visualizations can be created only from the text file data.
- Select from a wide variety of visualizations or one recommended by Many Eyes.
- Unleash insight by sharing the visualization over the web. The user can share the result by embedding a visualization in the blog or by sharing it on a social network.

Figure 9 shows bubble chart visualization of the example data taken from the public database of the Many Eyes framework.

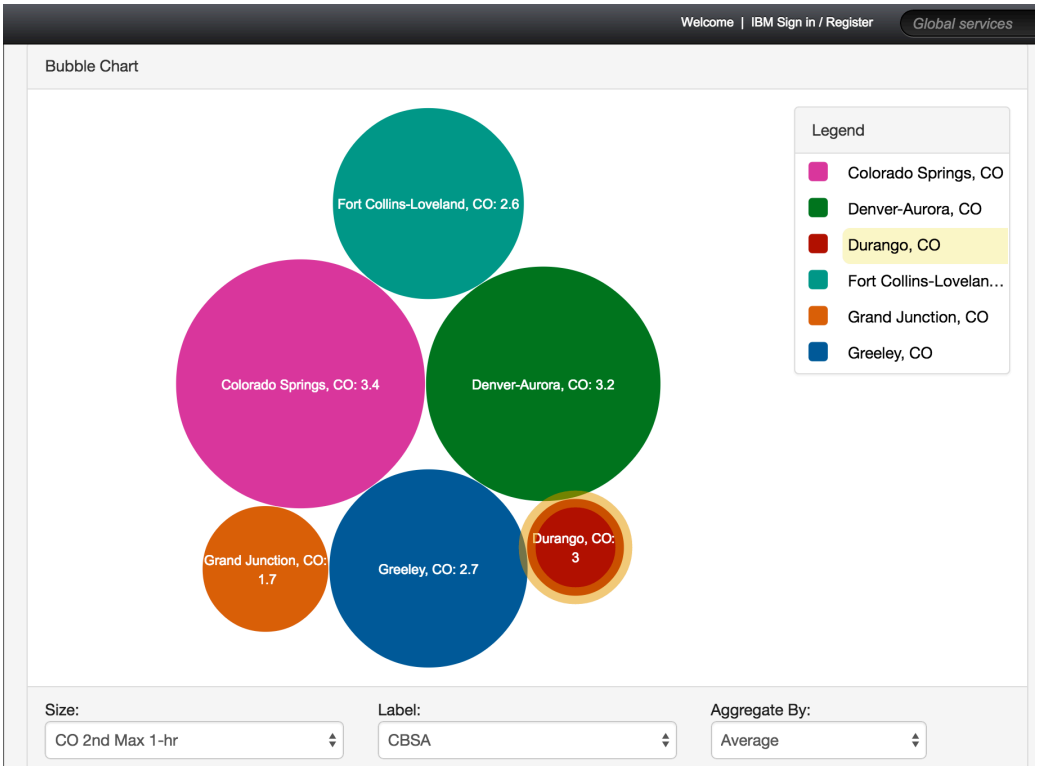


Figure 9: Bubble Chart Visualization of the Data through the Many Eyes Framework

5.1.2 *Visualize Free*

Visualize Free[1] is a free visualization tool that is based on visualization software developed by InetSoft[2]. It is the tool that provides access to the public data and also to upload the datasets. The data formats allowed for uploading the data are spreadsheet (Excel) and text (CSV) format. The uploaded data can be private or public by sharing the URL of the data on the server. It supports few types of visualization. Visualize Free is a friendly tool, so the user easily creates great visualizations. Visualize Free has a drag and drop feature to choose chart types and controls like filter lists,calendars. The data can be explored in a better way by use of such filters and brushing tools. Brushing highlights the related data across the multiple visualizations elements. It simply follows the three steps to complete the process of the visualization.

1. *Upload* - User registration is required to upload the data to the server which will create a user account. The data is private and can be made public by sharing.

1 Visualize Free Web Framework: https://visualizefree.com
2 InetSoft: https://www.inetsoft.com/

2. *Analyze* - Visualize Free can filter the data, analyze it and can share both the data and visualization.

3. *Create* - Visualize Free can create different visualizations with the help of the drag and drop service.

Figure 10 shows a dashboard with different types of visualization from the data available at the public database in the Visualize Free web framework.

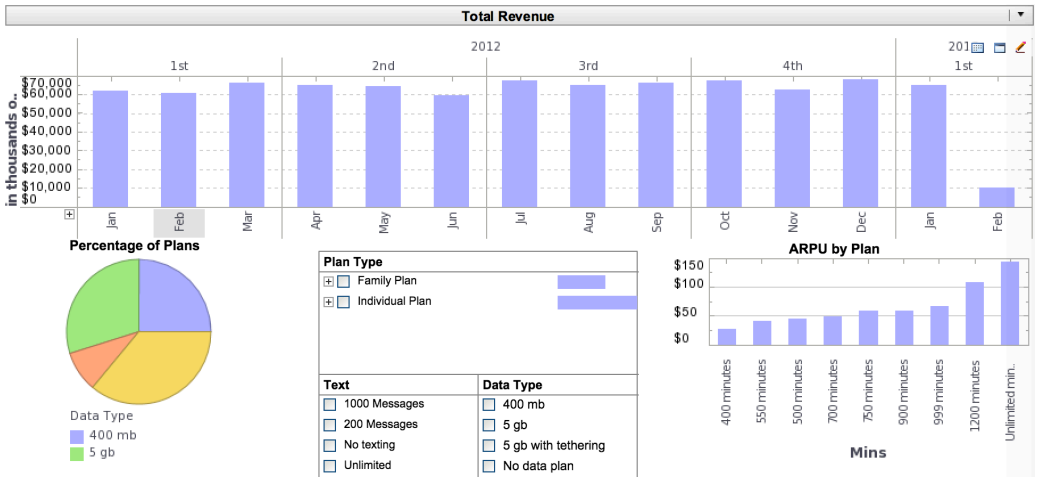


Figure 10: A Dashboard with Different Types of Visualization created through the Visualize Free framework

5.1.3 *Data Wrangler*

Data Wrangler[3] is a web-based service from the Stanford Universityś Visualization Group. This tool do not visualize the data, it provides service to visualization tools by cleaning and filtering the data. The main purpose of this tool is that the user should spent less time in formatting the data, to focus more on the visualization. This tool provides operations like split, extract, translate, drop, merge and transpose on the various data points. Data Wrangler is a free tool, but available in beta version with a deep learning curve.

5.1.4 *Tableau Public*

Tableau Public[4] is a web and desktop based framework for visualization that offers multiple visualizations based on the particular data. It provides sheet links through which views of framework can communicate with each other. It also provides a good filtering service so that complex visualization can be created. The few types of visualization are bar charts, line charts, scatter plots, bullet graphs, maps and many more. Tableau Public has robust tools

3 Data Wrangler: http://vis.stanford.edu/wrangler/
4 Tableau Public: https://public.tableau.com/s/

for filtering and selecting the data. Tableau Public makes the visualization easier for the users (non-technical) as programming is not needed to visualize the data. There is no access to real time data and not much support in terms of the custom views. The data mapping process is hard to understand as there is lack of a standard vocabulary for the visualization. The tool just uses some conventions like dimensions which are confusing at times. There is no local access to the data, only online access. Every data uploaded is public, so the data security is an issue.

Figure 11 shows distribution and ratio of girls education through the dashboard with a map and other types of visualization. This demo is created by Viz Candy[5] using Tableau Public.

Figure 11: A Dashboard Demo with a Map Visualization through the Tableau Public Framework

5.1.5 *Weave*

Weave[6] is defined as "a new web-based visualization platform designed to enable visualization of any available data by anyone for any purpose. Weave is an application development platform supporting multiple levels of users - novice to advanced, as well as the ability to integrate, analyze and visualize data at nested levels of geography, and to disseminate the results in a web page". Weave is a web and desktop based software tool for the visualization of data. This tool provides flexibility to visualize any kind of data anywhere. This tools supports a broad range of visualization types, which gives user more options to view the data. Weave was specially developed to accommodate huge data from the different sources and to geographically map that data using any shape files. The weave is a web-based tool. Weave provides services

5 Viz Candy: http://vizcandy.blogspot.ca/2013/05/educating-girls.html
6 Weave : https://www.oicweave.org/

to a wide range of users from beginner to expert. The weave is an open source system, having a high security, user-friendly and an efficient visualization.

Figure 12 is a demo taken from Weave demo examples[7] visualizes the data (obesity in US) through a map and a bar chart.

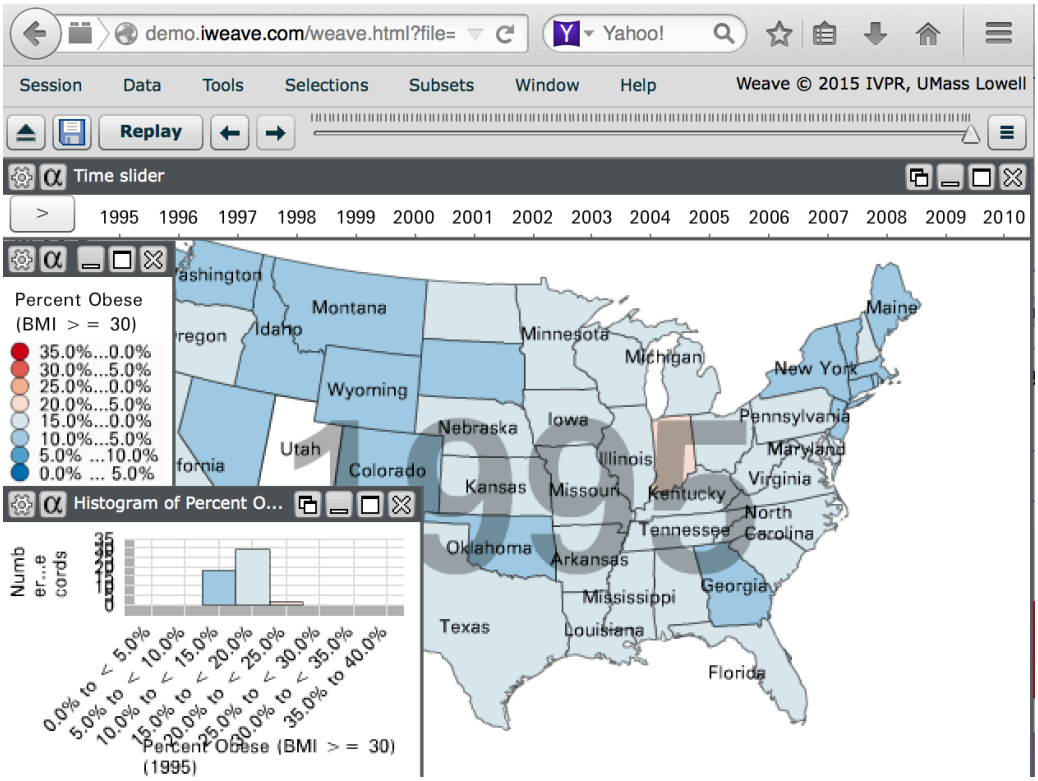


Figure 12: A Demo generates a Map and Bar Chart as visualization through the Weave Framework

5.1.6 *Evaluation of the Web-Based Visualization Frameworks*

All the introduced visualization frameworks - Many Eyes, Tableau Public, Weave and Visual Free need to be evaluated to identify if one of them meets the requirements identified in the Chapter 4. The evaluation is structured according to the identified requirements. To take their different importance into account, They are evaluated as ∅, + or ++.

- ∅ - the requirement is not fulfilled.
- + - modifications are necessary
- ++ - the requirement is fulfilled

In order to calculate the system's value, the above classification relates to a numerical system: ∅ stands for 0, + for 1 and ++ for 2. Once the frameworks are evaluated for each component requirements, then an overall framework

7 Weave Demo Visualization: http://demo.iweave.com/weave.html?file=demo_obesity.weave

evaluation is performed as shown in Table 6. This evaluation helps to determine which system is the closest to satisfy the requirements.

Table 2 shows the evaluation of the visualization requirements.

Visualization Requirements	Tableau Public	Weave	Visualize Free	Many Eyes
R1 - Possibility to Support All Popular Types of Visualization	+	+	+	+
R2 - Standard Third Party JavaScript Libraries	n.a	∅	∅	n.a
R3 - Local Access To Charts	∅	++	+	+
R4 - Data Mapping using Generic Visual Vocabulary	∅	+	∅	∅
Sum	**1**	**4**	**2**	**2**

Table 2: Evaluation of Visualization Reqirements

All the frameworks are evaluated as + for the requirement "Possibility to Support All Popular Types of Visualization" that means these frameworks do not support a broad range of types of visualization and hence are not generic in terms of visualization types. It is difficult for a framework to support all types of visualization but at the same time there should be a generic framework that covers the most popular types with a simple graphical interface, so the users can access it. There is a wide range of standard JavaScript libraries for the data visualization that offer creative modules and features for visualizing the data in an interactive way. The JavaScript libraries are platform independent. All the framework introduced, either use their library or are not open source. None of the frameworks except Weave, has a user friendly data mapping process, so the data information and labels are confusing and take time. Only the Weave framework provides local access to the charts, easy to save and edit it. All other frameworks do not allow access to the visual image locally, only allows to share the visualization on the web as a frame or a link. Table 3 shows the evaluation of the Data Service Requirements.

Data Service Requirements	Tableau Public	Weave	Visualize Free	Many Eyes
R5 - RDF and CSV Data Format as Input	+	++	+	+
R6 - Data Privacy	++	++	++	∅
R7 - Easy to Upload Data	++	++	++	++
R8 - Access to Processed Data in Standard Format	+	+	+	+
R9 - Processed Data Comprise of Visualization Information	+	∅	∅	∅
Sum	**7**	**7**	**6**	**4**

Table 3: Evaluation of Data Service Requirements

The Weave framework supports a broad range of data formats so it is evaluated as ++ and rest of the frameworks support only CSV (tabular) format.

Many Eyes works on the principle of public and open data, so all the data uploaded are public and anyone can access the data. All the frameworks provide the processed data in tabular or other form but not in a standard JSON Object form. The processed data generated by these frameworks has no information regarding the type of visualization selected, so Weave, Visualize Free, and Many eyes are all evaluated as ∅ for the requirement (R9). Table 4 shows the evaluation of the Web Framework Requirements.

Web Framework Requirements	**Tableau Public**	**Weave**	**Visualize Free**	**Many Eyes**
R10 - Programming in JavaScript	n.a	∅	n.a	n.a
R11 - Offline Mode	++	++	∅	∅
R12 - Cross Device Compatibility	∅	∅	++	∅
R13 - Open Source	∅	++	∅	∅
R14 - Cross Browser Compatibility (with no Flash Plugin)	++	+	∅	+
R15 - Framework should be User Friendly	+	+	++	+
Sum	5	6	4	2

Table 4: Evaluation of the Web Framework Requirements

Only Weave is an open source framework. The rest all frameworks are not open source so it is difficult to know their programming language. Weave is not a JavaScript based framework. The Visualize Free framework is a user-friendly tool while the other frameworks have a deep learning curve, so the user takes more time to understand and use these tools. All the frameworks need a flash enabled browser except Tableau Public, in order to visualize the data. "Cross Device Compatibility" requirement is important in today's world as the use of mobile devices with different platforms is increasing, smartphones are the most popular and highly used devices. Only the Visualize Free framework supports and fulfills this requirement. Table 5 shows the evaluation of the Usability Requirements.

Usability Requirements	**Tableau Public**	**Weave**	**Visualize Free**	**Many Eyes**
R16 - Customizability	+	+	+	+
R17 - Performance	++	++	+	+
R18 - Design and Interactivity	+	++	+	+
Sum	4	5	3	3

Table 5: Evaluation of Usability Requirements

All the frameworks provide some level of customizability but there is a scope of improvement, so these frameworks are evaluated as +. The Users and the developers should have more options to customize the visualization. The Tableau Public and Weave framework have a better performance than others in terms of time taken to upload the data and display the visual image

and also rendering of the charts. For the design and interactivity requirement all the frameworks need improvement except Weave.

Evaluation	Tableau Public	Weave	Visualize Free	Many Eyes
Data Service Requirements	1	4	2	2
Visualization Requirements	7	7	6	4
Web Framework Requirements	5	6	4	2
Usability Requirements	4	5	3	3
Sum	17	22	15	11

Table 6: Overall Evaluation

To figure out which of the introduced systems is the closest in fulfilling all the requirements, Table 6 gives an overview of the previous evaluations. Summarizing the strength of each system compared to the others, Weave leads in the visualization components and web-based capabilities as it is the only open source framework. The main strength of Tableau Public are ease of uploading the data and data privacy. The main advantage of the Visualize Free framework is the cross platform support as it provides the mobile app for both an Android and iOS platform. The weave framework is evaluated and granted 22 points as the total out of 36 points, this means the Weave framework fulfills 61.11% of the total requirements. The Tableau Public framework is evaluated and granted 17 points as the total out of 36 points, this means the Tableau Public framework fulfills 47.22% of the total requirements. Visualize Free is evaluated and granted 15 points as the total out of 36 points, this means the Visualize Free framework fulfills 41.66% of the total requirements. Many Eyes is evaluated and granted 11 points as the total out of 36 points, this means the Visualize Free framework fulfills 30.55% of the total requirements.

By the overall evaluation it is clear that no framework supports all the requirements that are defined, a few are good at some features but not for all the features. The weave framework is the closest to fulfill the requirements. Still there are some important requirements that need to be satisfied that Weave is not able to fulfill. The requirements for which Weave is evaluated as ∅ are as follows.

- R2 - Standard Third Party JavaScript Libraries
- R4 - Data Mapping using Generic Visual Vocabulary
- R9 - Processed Data Comprise of Visualization Information
- R10 - Programming in JavaScript
- R12 - Cross Device Compatibility

This clearly outlines that there is a need to develop a new framework that should focus on these requirements, so the concept of the vocabulary library and the *VoVis* framework evolves. The problems in the current frameworks

are: information regarding the visualization types are opaque and so is the mapping information. Also, the user has confusion regarding the labels and dimensions mentioned in the frameworks. To solve this problem a visualization vocabulary needs to be designed as a new component in the web framework and the JavaScript based visualization framework should be implemented using the JavaScript visualization libraries.

5.2 VISUALIZATION LIBRARIES (JAVASCRIPT-BASED)

More than thirty types of visualization libraries are reviewed that are listed in the Appendix B. The following sections contain descriptions of four different third party JavaScript visualization tools: D3, Google Charts, jqPlot and Flot. All of them are evaluated for their visualization capabilities and the ability to integrate with the *VoVis* framework. The selection process includes libraries that are open source, free, support desktop and mobile devices, with reasonable documentation, examples and support community, and are regularly maintained.

5.2.1 *Data Driven Document (D3)*

D3[8] is a free and open source JavaScript library for creating data visualizations, developed in 2011 by Mike Bostock (in collaboration with Heer and Ogievetsky). The book "Interactive data visualization for the Web"[34] explains D3:

" Data-Driven Documents (D3) where the data is provided by a user, and the documents are web-based documents, meaning anything that can be rendered by a web browser, such as HTML and SVG. D3 does the driving, in the sense that it connects the data to the documents".

This clearly explains that it is a library for manipulating documents based on the data. It is not a chart drawing library, as its primary focus is not drawing charts, so there is no built-in charts available. But the library has more than two hundred examples of existing, new and innovative charts that are created from scratch. The input data is converted in the graphic components by using the SVG as graphic technology and then grouped together in a general graphical representation. Thus to draw a simple bar chart, every component (bars, title, axes, labels, etc.) has to be implemented separately and then are assembled in a single chart. D3 has some unique features which encourage to use it to generate the SVG and HTML Markup. A few features are mentioned below.

- D3 can add, remove and modify multiple elements from an existing DOM.
- D3 can dynamically add attributes and styles according to a function.
- D3 can animate the attributes and styles.

8 Data Driven Documents: http://d3js.org/

- It can bind the data to the automatically added elements when needed.
- It provides benefits from a lot of helper functions like select, append, and bind event.

While this tool is extremely flexible, it has a deep learning curve, so it requires deep coding knowledge for the implementation of the visualizations. It is best used for the complex and non-standard data visualizations. Figure 13 shows D3 Gallery[9] with the different types of Visualization.

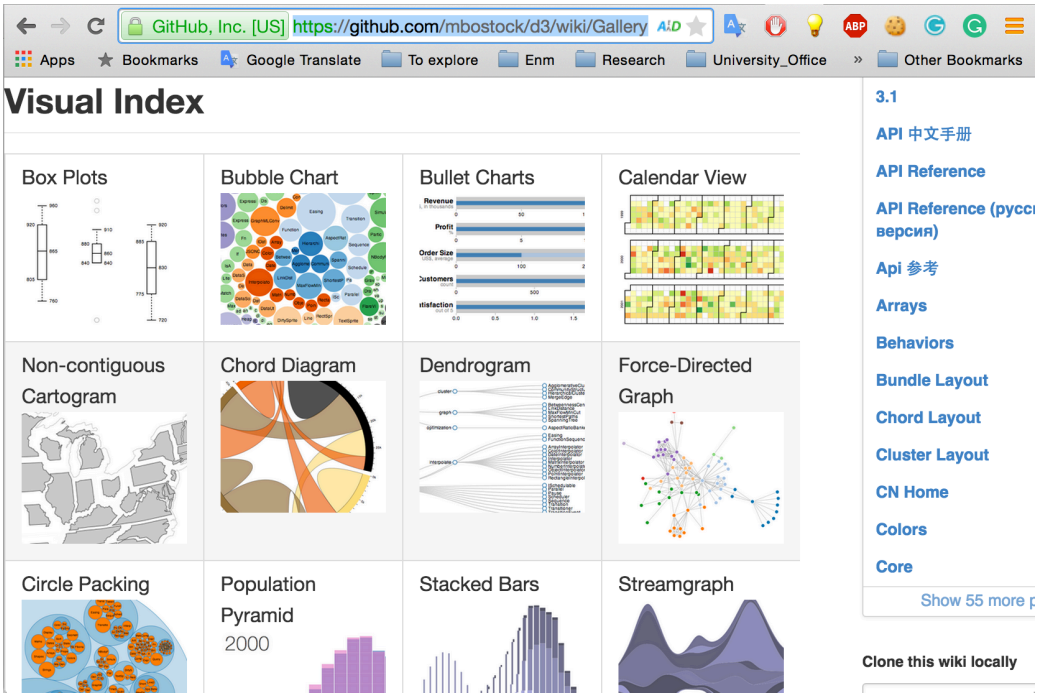


Figure 13: D3 Gallery showing Different types of Visualization

5.2.2 *Google Charts*

Google Charts[10] is a Google product for data visualization that owns various charts ranging from the simplest line graph to a complex hierarchical treemap. It is a "ready to go" and "easy to implement" kind of a tool with examples for all the charts available in Google Chart Gallery[11]. The charts are rendered in the HTML5 charts using the SVG and VML as a graphic technology. It provides the cross-browser compatibility (including VML for older IE versions) and the cross-platform portability to iPhones, iPads and Android devices. To draw a simple chart in Google Chart, all one needs is some simple JavaScript code in a web page. The steps to create a simple chart using Google Charts are as follows.

9 D3 Gallery: https://github.com/mbostock/d3/wiki/Gallery
10 Google Charts: https://developers.google.com/chart/
11 Chart Gallery: https://developers.google.com/chart/interactive/docs/gallery

1. First step is to load the Google Chart libraries.
2. List the data to be visualized.
3. Customize the chart with the options.
4. Create a chart object with an id.
5. Display the chart in the web page through a .

Listing 1 shows a piece of programming code in JavaScript to visualize the data through the Google Charts library.

Listing 1: JavaScript function to draw a chart using Google Charts

```
function drawChart() {

        // Create the data table.
        var data = new google.visualization.DataTable();
        data.addColumn('string', 'Topping');
        data.addColumn('number', 'Slices');
        data.addRows([
          ['Mushrooms', 3],
          ['Onions', 1],
          ['Olives', 1],
          ['Zucchini', 1],
          ['Pepperoni', 2]
        ]);

        // Set chart options
        var options = {'title':'How Much Pizza I Ate Last Night',
                       'width':400,
                       'height':300};

        // Instantiate and draw our chart, passing in some options.
        var chart = new google.visualization.PieChart(document.
            getElementById('chart_div'));
        chart.draw(data, options);
      }
```

All the chart types are produced with the data using the DataTable class, makes it easy to switch between the chart types. The DataTable provides options for sorting, modifying, and filtering data, and can be populated directly from the web page. Figure 14 shows the gallery[12] of Google Charts with the different types of visualization supported.

5.2.3 *jqPlot*

"A Versatile and Expandable jQuery Plotting Plugin".

12 Google Charts Gallery: Google Charts

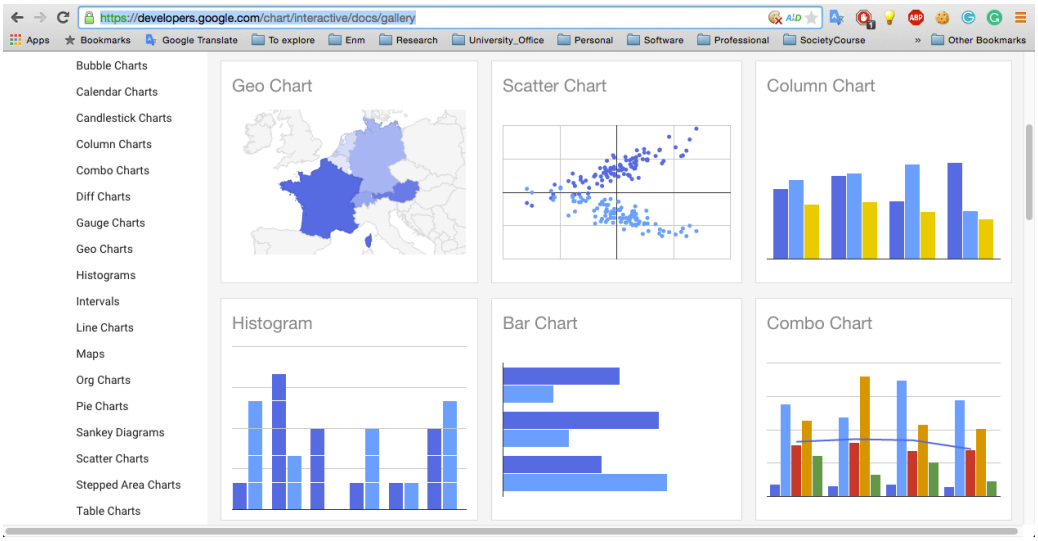


Figure 14: Gallery of Google Charts with different types of Visualization supported

jqPlot[13] is a free and open source JavaScript library for the generation of the charts. jqPlot has a profound standardized structure and, it is developed using a large number of plug-ins. The jqPlot library is an extension of jQuery, so jqPlot requires the jQuery plug-in to perform the operation of the visualization. Every object that the user draws can be a line, an axis, or the grid itself is managed by a plug-in. jqplot provides the customizable setting options, and every added plug-in can be extended. jqPlot is a modular and flexible library. The users with less programming expertise can create professional charts in easy steps without adding too many lines of code. It is possible to download this library from the official site, and registration is not required. The steps to visualize the data using jqPlot are as follows.

1. Add jQuery plug-in, the jqPlot plug-in, and a jqPlot CSS file.
2. Add a plot container.

```
<BODY>
...
<div id="myChart" style="height:400px; width:500px;"></div>
...
</BODY>
```

3. Create the plot inside the jQuery function. jqPlot is an extension of jQuery, so its methods need to be called inside the $(document).ready() inorder to execute the code.

```
$(document).ready(function(){
// Insert all jqPlot code here.
});
```

13 jqPlot Library: http://www.jqplot.com/

4. Then, to create the actual plot, needs to call the $.jqplot plug-in with the id of the target in which chart will be drawn. This call is executed by the following jQuery function:

```
$.jqplot(target, data, options);

$.jqplot ('myChart', [[100, 110, 140, 130, 80, 75, 120, 130, 100]])
    ;
```

   The jqplot() function has three arguments; target, which is the ID of the target element in which the plot is to be rendered. The data, consisting of an array for data series; and options, the main feature of jqPlot, with the options, the chart can be customized.

Figure 15 is a home page of jqPlot library, shows the different types of Visualization generated using jqPlot.

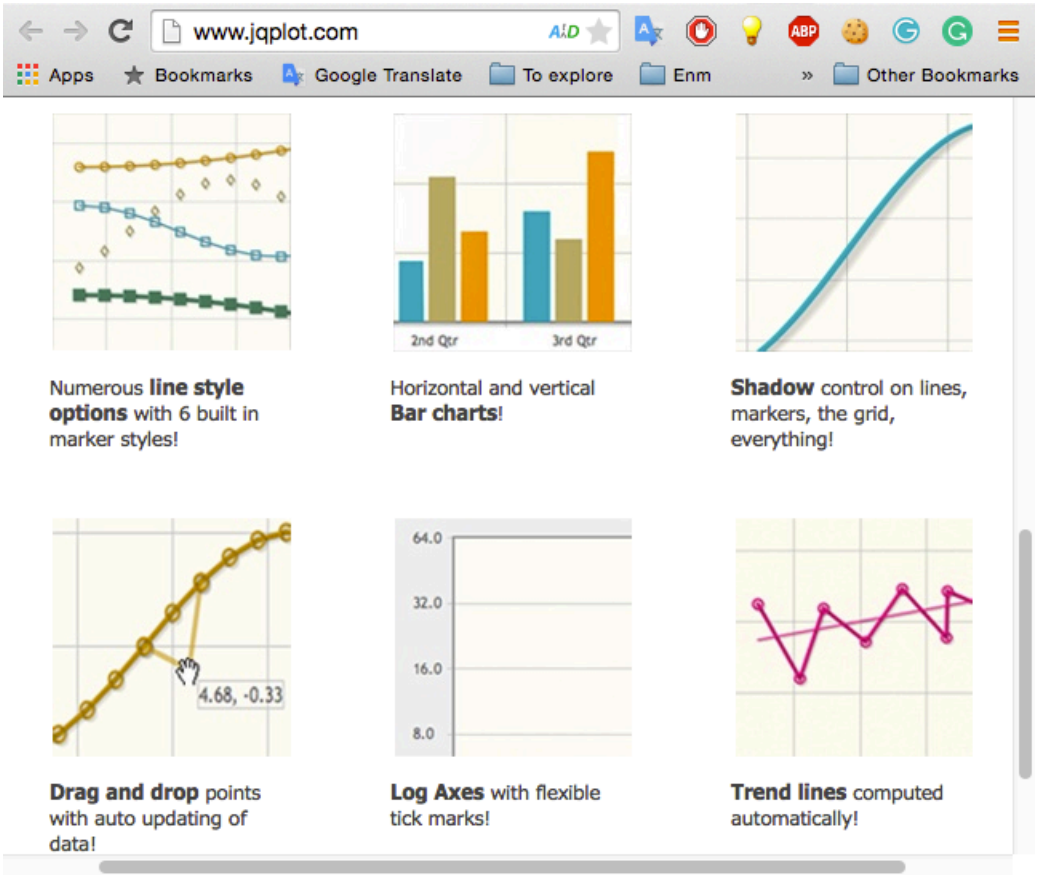


Figure 15: Different types of Visualization generated using jqPlot Library

5.2.4 *Flot*

"Attractive JavaScript plotting for jQuery"

Flot [14] is a pure JavaScript plotting library for jQuery, with a focus on simple usage, proper aesthetic and interactive features. Flot is easy to use, just a few lines of code, can create a simple line chart, it also gives a comprehensive API documentation which have examples, usage, and methods. The most important, Flot continues to release new versions, and each new version comes with new features.

The three things that are important to create an attractive plot are as follows. The first is to create a placeholder, make sure it has dimensions (so Flot knows what size to draw the plot). The second is to call the plot function with the JSON data. The axes are automatically scaled. The graphs are interactive and can be altered within the web-browser without the server communication. The third thing is to retrieve additional data from the server. It provides features like moving, shifts, zoom, retrieval of the values at precise points on the graph and select an area data. Figure 16 shows a Line Series Visualization example generated using the Flot Library.

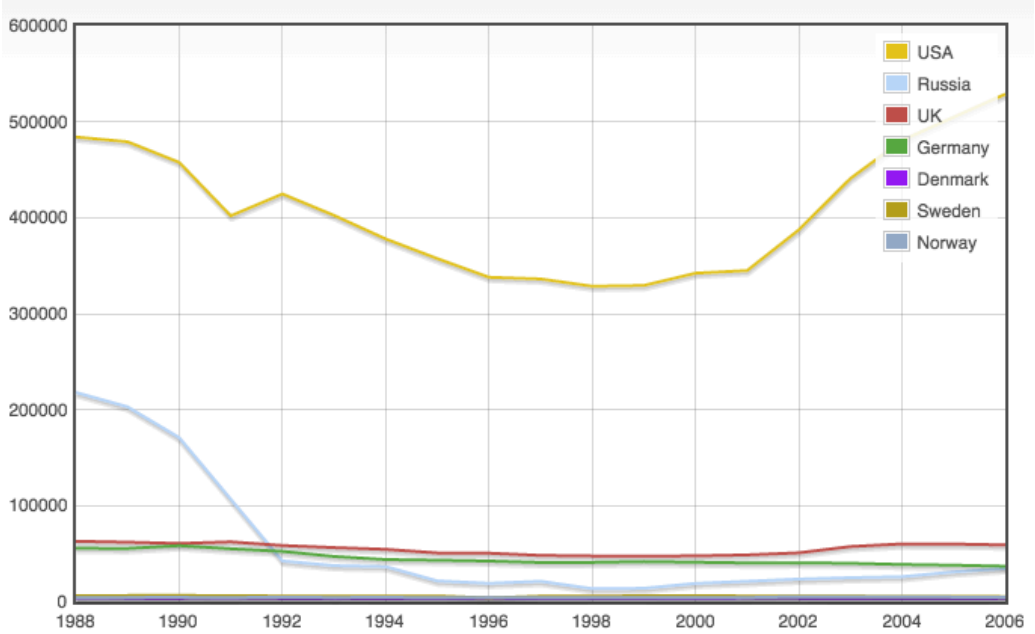


Figure 16: Line Series Visualization generated using Flot Library

5.2.5 *Evaluation of JavaScript Visualization Libraries*

The result of evaluation of the web-based visualization frameworks in the Section 5.1.6 shows that there is a need of a visualization component that can be integrated with the *VoVis* framework. The previously introduced JavaScript tools - D3, Google Charts, jqPlot and Flot need to be evaluated to identify if one of them meets the requirements related to JavaScript visualization libraries and to find the best solution out of these four visualization tools. Table 7 shows the evaluation of the JavaScript Visualization Requirements.

14 Flot Library: http://www.flotcharts.org/

JavaScript Visualization Libraries Requirements	D3	Flot	Google Charts	jqPlot
R2/1 - Open Source	++	++	∅	++
R2/2 - Offline Mode	++	++	∅	++
R2/3 - Range of Charts Supported	++	+	+	++
R2/4 - Supports Maps and Geo Charts	++	∅	++	∅
R2/5 - Cross Browser Compatibility	+	++	++	++
R2/6 - No Dependency on Other Tools	++	∅	++	∅
R2/7 - Steep Learning Curve	∅	+	++	++
R2/8 - SVG as Graphic Technology	++	∅	+	∅
R2/9 - Support for Additional Features like ToolTips, Number Formatting	++	∅	++	∅
Sum	**15**	**8**	**12**	**10**

Table 7: Evaluation of JavaScript Visualization Libraries

All the requirements mentioned above are part of a requirement "R2 - Third Party JavaScript Visualization library". Each of the sub-requirement is discussed below, and each JavaScript tool is evaluated for that requirement.

- **R2/1** - *Open Source*: Open source tools holds numerous advantages over the closed tools. A few important advantages are security, quality, Customizability, freedom, flexibility and many more. It provides better quality as thousands of developers work on it with many new innovative ideas. There is a huge support group where actively the problems are discussed and solved by the experienced developers. The bugs in the open source software also tend to get fixed immediately. The users and developers both have complete access to the tool(open code), so it is easy to modify or add new features. The users can use the software as they want. It is the decision of the developer when to update the tool or even can work with an older version of the software. It gives freedom from the proprietary package that costs both time and money which the customer can utilize in the actual product. The three of the above tools are open source except Google Charts(evaluated as ∅).

- **R2/2** - *Offline Mode*: In Google Charts the JavaScript files are loaded directly from the Google's servers. The application always need to be online to view the charts in Google Charts. D3, jqPlot and Flot can even work offline. So the application using these tools can plot the charts without the Internet as it is not feasible to be always online.

- **R2/3** - *Range of Charts Supported*: Google charts draws more than 15 types of 2D charts with the several custom options, still many types of visualization are not supported by it so evaluated as +. Similarly jqPlot offers more than 25 2D charts and are easy to plot so evaluated as ++ for this requirement. D3 does not come with the pre-built charts, but

a library with 200+ examples is available. Whereas Flot offers very few charts so evaluated as +.

- **R2/4** - *Supports Maps and Geo Charts*: Google Charts supports maps and Geo charts(with markers), so evaluated as ++. Through D3 any custom Geo chart or map can be created, but no inbuilt maps in D3. Both Flot and jqPlot do not support the maps and geocharts which is an essential visualization type for both the projects DaPaaS and Citi-Sense-MOB.

- **R2/5** - *Cross-Browser Compatibility*: Everyone uses different browsers, and operating system. A Few popular types of browsers are Firefox, Safari, Chrome and Internet Explorer. Google Charts supports all the modern web and mobile browsers including IE6+, jqPlot supports IE 7, IE 8, Firefox, Safari, chrome and Opera, D3 supports all the modern web and mobile browsers for IE - 9 and above, and Flot supports IE 6+, Chrome, Firefox 2+, Safari 3+ and Opera 9.5+.

- **R2/6** - *No Dependency on Other Tools*: D3 and Google Charts are the libraries that implement all the functionalities without using any external library so are evaluated as ++. Both jqPlot and Flot are an extension of the JQuery library, they are plugins that are depended on the JQuery library. The dependency on the other tools makes the tool decoupled and less flexible. The maintenance of the tools is also an issue in such cases.

- **R2/7** - *Steep Learning Curve*: The learning curve is defined as the time taken to learn and understand a tool. It is important that the tools should have a steep learning curve so that the developer can easily learn it in less time and implement the tool in the framework. But at the same time the tool should poses features which are essential with a cost of some learning. D3 requires a deep coding knowledge for the implementation of the visualizations so is evaluated as ∅, both Google Charts and jqPlot have a steep learning curve and evaluated as ++. Flot is a friendly tool but needs some learning so evaluated as +.

- **R2/8** - *SVG as Graphic Technology*: There are two graphic technologies the HTML Canvas and the SVG. These two do not compete against each other, but there is always a preferred tool for the particular job. Scalable Vector Graphics (SVG) offer a better alternative when the task is to plot the charts. SVG has better scaling feature which means the charts will scale accordingly. The SVG elements are accessible means the text will be in text form, and it is easy to understand the SVG code. The most important feature is it supports DOM, so it is easy to attach the event handlers and manipulate the elements. D3 draw elements using the SVG technology and is evaluated as ++, In Google Charts the charts are rendered in HTML5 charts using SVG and VML so evaluated as +. Both jqPlot and Flot use Canvas technology and are evaluated as ∅.

- **R2/9** - *Support for Additional Features* like ToolTips, Number Formatting: To customize the visualization as per the user requirement it is

essential that the JavaScript tools should support additional features. There is a wide range of features which a tool can have for better visualization like support for multiple axes, number formatting, and tool tips. Google charts and D3 support most of it so are evaluated as ++. But Flot and jqPlot do not support these features so are evaluated as ∅.

The evaluation shows that D3 with a total of 15 points out of the 18 points is better than all the other libraries. The only problem D3 has is a deep learning curve and no pre-built charts support. Google Charts is ranked second after D3, with a total of 12 points out of the 18 points, is a good option when open source is not a requirement, also this library needs online access. The jqPlot library is an excellent option for plotting user-friendly charts and graphs in less time but maps not supported. The fulfillment rate of all the JavaScript tools evaluated are D3 (83.33%), Google Charts (80%), Flot (53.33%)and jqPlot (66.66%).

Part II

THE VOVIS FRAMEWORK

6

VOVIS: CONCEPT AND DESIGN

The evaluation from the previous chapter concludes that none of the introduced web-based visualization frameworks meets all the requirements. Thus, there is a need for a new visualization framework that will support a wide range of data formats and types of visualization with a friendly data mapping technique. To meet these requirements the concept of a vocabulary based visualization is introduced, and a web-based framework is designed and developed using this vocabulary component - Vocabulary Based Visualization (*VoVis*) Framework.

In this chapter, a conceptual architecture for a web-based visualization framework is proposed in the Section 6.1, followed by the design of such a framework - *VoVis* in the Section 6.2. The concept and design of each component is described more precisely in the remaining sections.

6.1 CONCEPTUAL ARCHITECTURE OF A WEB-BASED FRAMEWORK

The pattern-based web applications have become more popular as it provide reusability and consistency. The design and development of a web application faces many challenges to fulfill the requirements along with the other factors like scalability and security. The design Patterns are chosen as building blocks by the architects and developers which are considered as extensions of object-oriented technologies [40]. In order to develop a better application, the design steps should follow some pattern. This can be achieved by separating the abstraction component from the implementation in a framework.

Most of the web frameworks are based on the model-view-controller (MVC) pattern [30]. The main motivation behind the selection of the MVC pattern are as follows.

- Parallel development of the different modules like view, controller and model is possible.
- Business logic can be reused across applications.
- No dependency between the user interface and business logic.

Each web application has three main layers: presentation (UI), application logic, and data management. MVC pattern breaks the presentation layer into controller and view. Following are the components of MVC pattern

- Model: The model provides business logic in an application. The model evaluates the data given by controller, executes the operation and produces the result.
- View: Defines how data should be displayed to users and presents the results generated by the model to the user. A view can be output representation of information, such as a bar chart or a map.
- Controller: Acts as glue between model and view. It contains logic that updates the model and view. The controller has event listeners that get notifications when events are triggered in the user interface. The controller maps these events into operation.

Figure 17 shows the collaboration of MVC components [55]. A general workflow of MVC web visualization application are as follows.

1. The user interacts through the user interface by uploading a file.
2. The controller handles the input event from the user interface.
3. The controller then call the model to execute the operation on data.
4. The model maps and process the data to generate visualization image and forward it to the view.
5. The view gets the visual image from the model and displays it in user the interface.

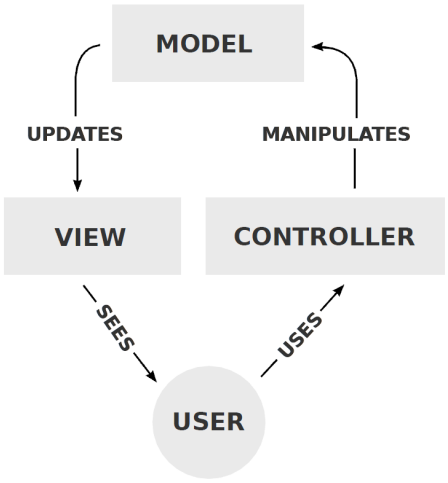


Figure 17: Model-View-Controller Components

Figure 18 shows a conceptual architecture of the web visualization application framework. A web application runs on a web browser and is created using the scripting language like JavaScript. This Web application is based on MVC pattern. The MVC pattern for the web application are implemented in several ways such as server-side MVC, client side MVC or combination

of these two. There are technologies already developed that can coordinate Client Side Scripting with Server Side technology like PHP. In the client side JavaScript MVC application, all three modules i.e. the controller, business logic and views are part of the client. The web application can be purely in JavaScript, or front-end with JavaScript and back-end with Java or PHP with web services. Client-side JavaScript application needs a server to host the application; the example would be simple interactive visualization application delivered over the web using a web browser as a user interface.

The user interface is a client web browser that has views, and they are written in HTML5, CSS, and JavaScript. When the model (process) perform business logic and renders the view; as a result, the view is shown to the user in a browser. Data is uploaded, options are selected in the user interface, both data and options are forwarded to the controller. Controller controls the flow of work. Following are the main layers of the web application for the visualization, also outlined in Figure 18.

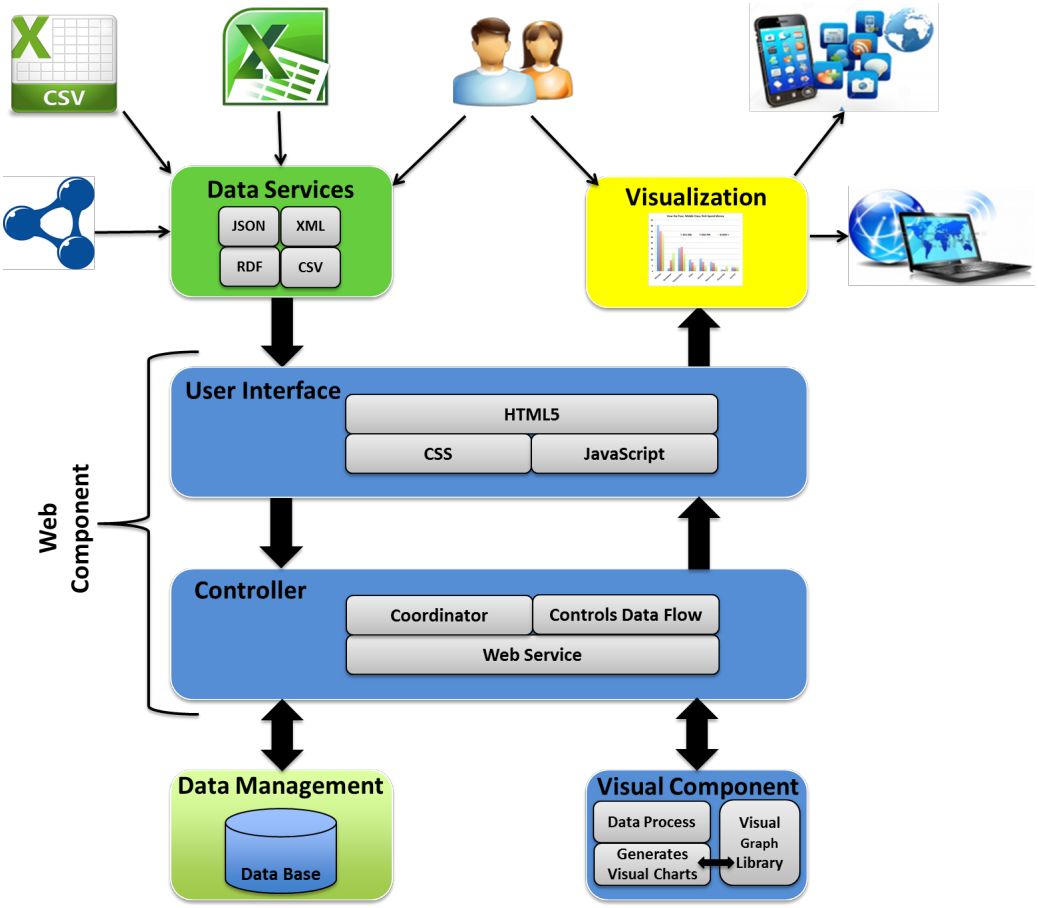


Figure 18: Concept and Design of a General Web Visualization Framework

- User Interface: user interface interacts with the end user through a web browser or app. Input files can be uploaded, or can be fetched from the server through a web browser. The user has the control to select actions (like select particular visualization for data). When model and

controller process the request of the user, the final visualization is displayed on the browser or mobile app page.

- Controller: Controller is the primary coordinator between user interface, database, and visual component. It manages the execution of all tasks. Controller can use a web service to get additional services.
- Visual Component: Provides the visualization logic to generate a particular type of visualization from data. It executes all the operations, right from data mapping, data processing to visualization generation.

6.2 THE VOVIS: DESIGN OVERVIEW

In order to fill the research gaps and fulfill requirements, the *VoVis* framework will try to accomplish the following tasks.

A. Design a visualization library that have information for most popular (many) types of visualization and are categorized by the functions they demonstrate.

B. Design a vocabulary for each type of visualization in the library. This supports the data in all formats. For this, two version of the library is designed i.e. an RDF and text vocabulary library.

C. Implement a process to map data with vocabulary into a standard format.

D. Implement a process to visualize the processed data using standard tool.

E. Make the framework cross-platform both for app and web.

Figure 19 shows the architecture of the *VoVis* framework. The *VoVis* framework has layers of components with vocabulary as the new component. This framework is designed to explain how data can be easily and efficiently visualized, if it follows the standard vocabulary and structure. The important task in field of visualization is to understand the requirements and plot the correct chart accordingly. If user or developer is willing to plot a chart, but the data available does not fulfill vocabulary requirements for a particular chart, then the chart can not to be plotted. Therefor some standard set of rules are required for visualization of data. This framework provides visualization along with processed data (JSON object), which has complete information about particular chart type.

6.3 THE VOVIS VOCABULARY COMPONENT

The proposed vocabulary component contains libraries required for the mapping and visualization of the data. Each library is discussed in detail in the following sections.

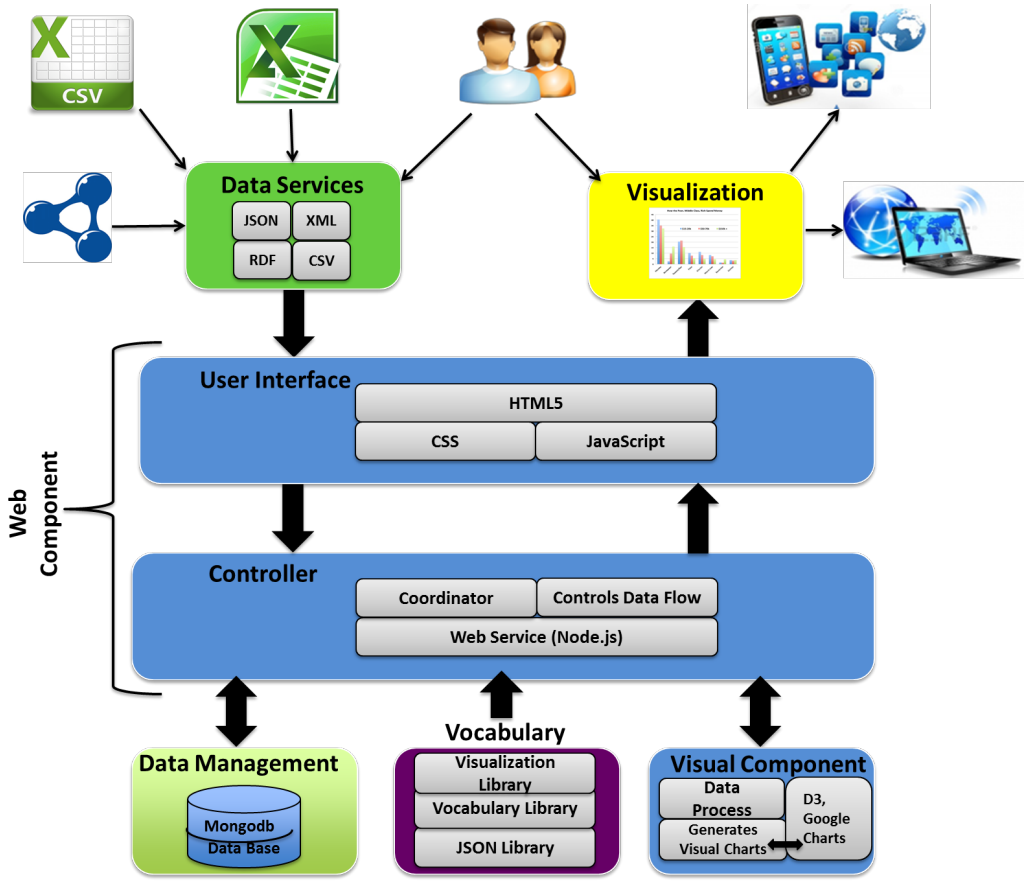


Figure 19: Concept and Design of *VoVis* Framework with Vocabulary Component

6.3.1 *Visualization Library*

One of the biggest challenge for non-technical and business users in producing the data visualizations is to select the appropriate type of visualization to represent the data accurately. Here the focus is more on the conventional static visualization types, which are more useful for the information visualization [31]. To make the selection process easier, the data should be grouped into a category and according to that category the correct visualization type can be selected. Below are the categories for the data and the visualization types to visualize it.

1. 1D/Linear - The linear data are textual documents, program source code, and alphabetical lists of names. It is normally not visualized, can only be arranged in some order.

2. 2D/planar - The 2D data consists of planar or map data with geographical features. These data can be easily located through the maps. Many features like markers, different colors can represent additional data on the map. Geospatial are the best examples for this category such as choropleth, cartogram and also heat maps.

3. 3D/volumetric - The 3D data are real-world objects such as molecules, the human body, and objects with volume. The scientific visualization uses these data to visualize it in 3D graphs and charts. These data are complex and requires a profound knowledge to visualize it.

4. Temporal- The data in this category are time stamped. They are useful when the user wants to arrange the data in chronological order. These data can be visualized in the form of calendars, timeline, arc diagram and many more.

5. Multidimensional - Data with the multiple attributes are multidimensional data. In order to understand multidimensional data and the relation between the attributes, the visualizations that can display the information correctly are used. By the number of dimensions a particular visual presentation can be selected. If data has three or four dimensions, bubble chart can be used to display all the dimensions. Multibar/line, box chart, pie chart are few types of presentation.

6. Tree/Hierarchical - This is collection of data items that form hierarchies/tree structure with each dataset link to the parent object. A tree has instances as parents, children, and siblings. This tree-like structures grow in depth with data and has a common root node. This type of hierarchy helps to understand the structure very well. Types of visualizations for these data are tree diagram, arc diagram, etc.

7. Network - These data are connected to other data with links other than parent-child relation. It can be presented using tree hierarchy. Network graphs are the structure to present this kind of data according to the association data have with each other.

The second category is based on the functions [5] that the types of visualization demonstrate. It helps those users who do not have in-depth knowledge about visualization types, because it is most important and challenging task to select correct type of visualization. If the right chart type is not selected, the information that the user wants to communicate will be misinterpreted. Following are the few functions that different types demonstrate.

1. Comparisons - This function allows all types of charts which can be plotted to compare the inputs on level of their similarities or differences. This could further be subdivided into
    a) Based on the position of axis - Bar, box, bubble, histogram, stacked area, and population pyramid.
    b) No axis - Donut, choropleth map, and venn diagram.

2. Proportions - This method can use size and area to show the similarities and differences between the data. Bubble chart, proportional area chart, and word cloud are few types to demonstrate this function.

3. Relationships - This function describes the relationship between the data, how are they related to each other. Arc diagram, tree diagram, and radial chart are few types to demonstrate this function.

4. Locations - This method shows data over a geographical region. Flow map and choropleth map are few types to describe this function.

5. Hierarchy - This method shows how the data or objects are ranked and ordered together in an organization or system. Tree diagram, tree map are few types to demonstrate this function.

6. Concepts - This method helps to explain and show ideas or concepts. Brainstorm, venn diagram are few types to demonstrate this function.

7. Part to Whole - This method shows the fraction of data to it's total, is often used to show the split,how the data is divided up. Example types are donut chart, pie chart.

8. Distribution - This visualization method displays the frequency of data, how the data is distributed over an interval or is grouped. Examples are bubble chart, dot matrix, population pyramid.

9. Movements and Flow - This method helps to explain the flow of information between data. Example types are flow map, sankey diagram, parallel sets.

10. Patterns- This method reveals the forms or patterns in the data to give it a better meaning. Bar chart, population pyramid, and scatterplot are few types to demonstrate this function.

11. Data over Time - This method shows the data over a period or show time itself. Example types are calendar, timeline, and timetable.

12. Range - This method shows the variations between upper and lower limits on a scale. Example types histogram, box plot

13. Process and Methods - This method helps to explain processes or methods that the data follows. Example types are sankey diagram, parallel sets.

By the use of these categories choosing a particular chart type is much easier and precise. There are few frameworks that work on above principle better known as chart chooser. They only provide a few categories, a few of them are as follows.

1. Chart chooser [7] - It is a small free tool that provides some information about categories based on the functions. The functions act as filters and by choosing a particular one helps to find the correct chart type. And once a chart type is selected it is possible to download a template in Excel or Power-point.

2. Slide Chooser [10] - It is a two slide layout that helps to identify which chart type to select according to the requirements.

3. Graphic cheatsheet [23] - It is just a sheet that creatively defines the different methods and how the different charts are organized in these methods.

The Data Visualisation Catalogue [1] is an online reference tool that helps to choose the right data visualization method. This web application provides details for each type of visualization and categorized it by the functions and list. This thesis collects the information about types of visualization from Data Visualisation Catalogue. The visualization library of the *VoVis* framework supports a total of 30 types of visualization. The user or the developer can choose any of the available types based on the functions a visualization can demonstrate. By selecting a particular type do not solve the problem, the selected chart type needs to map with the input data. Every chart has its unique requirements and features, there should be some standard rules or information about each type. This is the part missing in almost every existing framework. This makes the new framework special by introducing the concept of vocabulary Library.

6.3.2 *RDF Visualization Vocabulary (VisVo)*

Visualization vocabulary describes the visualizations in terms of properties they hold. Without a generic vocabulary for visualization features, mapping and plotting of the data becomes challenging. Most of the current visualization frameworks lack comprehensive vocabulary library. In some frameworks, either it is completely missing or they use only few conventions such as dimensions and measures. Existence of multiple dimensions in the data even makes the mapping harder. In this case, users need to understand those implicit rules and conventions. None of the frameworks has a standard visualization vocabulary for each visualization type that helps users to map the data with particular visualization. The two data formats (i.e. CSV and RDF) mostly covers all popular types of data including the research data. These data are either in the tabular form (CSV/Excel) or in RDF form (also known as linked data). If the data is in CSV format, the best option is to have text vocabulary so that it is easier to map this data and process it as a JSON object. When the data is already in the RDF format, or data needs to be mapped into RDF form, simple text vocabulary might not work. Therefore description of visualizations type for these data needs to be done semantically [14]. Without vocabulary visualization of the linked data needs to be done manually. This requires even high maintenance cost.

The design of the visualization vocabulary should be in the form of an OWL ontology [33]. For the semantic description of charts, it is possible to use the formal language, like XML. This language does not describe relations between different properties, for example, the chart has x axis, y axis and

1 The Data Visualisation Catalogue: http://www.datavizcatalogue.com/

has data values. With RDF format, it is easier to retrieve data in a standard way. Semantic description of visualizations using RDF is a novel research approach, and elementary work has been done in this area. There are few standard common vocabularies like RDF Data Cube. The problem is, all are very complex and cannot visualize directly, need a broad knowledge of vocabulary. The most significant and similar research is the Statistical Graph Ontology [16] that provides a new approach to annotate visualizations semantically. While the Statistical Graph Ontology provides information which is useful only to present statistical graphs. The drawback of this kind of vocabulary is that some of the essential features of visualization are missing like color, size also data type, etc. Some of the classes in this vocabulary are not human readable.

Vispedia [6] is other web-based visualization system to visualize Wikipedia[2] datasets. This tool is restricted to be used only for Wikipedia data; furthermore no binding of data is supported. Also, it does not provide automatic binding of heterogeneous data onto visualizations.

The CODE project[3] developed a Visual Analytics (VA) Vocabulary[4] which uses some standard vocabulary like RDF Data Cube [13]. This is a W3C Standard and has been developed to represent statistical data as RDF. Visual Analytics vocabulary is in the form of an OWL ontology. This vocabulary is an interface between the RDF Data Cube and visualization-specific technologies. Along with the RDF Data Cube Vocabulary, the VA vocabulary forms the basis for automating the visualization process. In the CODE project, the RDF Data Cube is used to define the meta-model in order to capture the evaluation results from publications. The work describes about VA vocabulary which is used to represent the information about visualizations. VA vocabulary describes the visualization axes and other visual channels, such as color, the size of visual symbols, which are used to represent visually the data [35]. This vocabulary has mapping component that maps the RDF Data Cube and the VA Vocabulary. Mapping is between dimensions, measures of the RDF Data Cube (i.e. cube components) and the corresponding axes and visual channels of the visualization. This explains that VA vocabulary cannot be used alone and always needs results from Data Cube. It does not support for most popular visualization types, and many specific features of visualization are missing.

Literature survey shows that no visualization vocabulary ontology exist, hence new Vocabulary design is required. The RDF visualization vocabulary (VisVo) is the vocabulary that is standard and provides information for most popular visualization types. The VisVo vocabulary has all the properties that a visualization type needs to present data in visual form. All the common features are clubbed together along with unique features, specific to each type. For example a chart can be a single or multi-chart, again it can be an the area, bar or line chart. A chart has standard features with others visual types like single charts have single x axis, y axis, labels, and values. The values

2 Wikipedia: https://en.wikipedia.org/wiki/Main_Page
3 CODE Visualization Wizard: http://code.know-center.tugraz.at/vis
4 VA Vocabulary: http://code-research.eu/ontology/visual-analytics

can be further subdivided into x values and y values with certain data types. All such information can be stored in a vocabulary. The VisVo vocabulary library will have different classes and properties, the idea is to write vocabulary for most popular types defined in new visualization library. Initially approximately 30 types is identified. For the proof of concept, the VisVo library has information for the single bar, area, line charts and also bubble and scatter-plot charts. There are total 23 classes and 18 properties for describing five types of visualizations. Figure 20 shows the VisVo ontology as graph diagram, with the classes and properties.

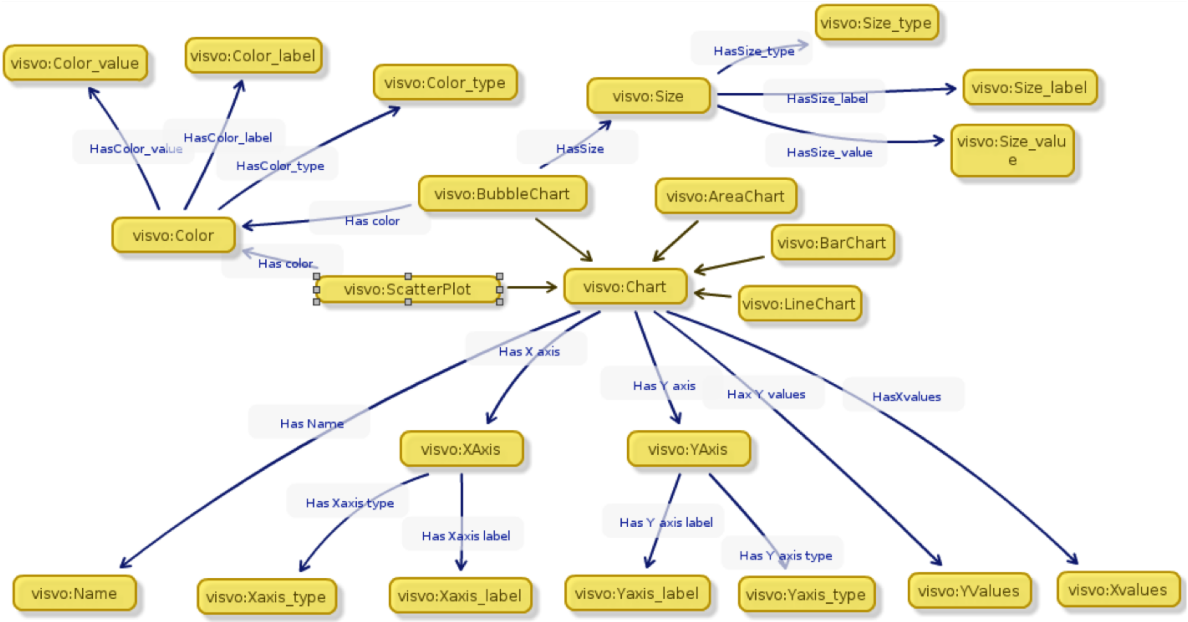


Figure 20: The VisVo vocabulary ontology diagram with 23 classes and 18 properties

6.3.3 *Types of Visualization*

This section describes the visualization types. Each visualization type requires to answer question such as; Which data points are important for plotting?, Is the given data sufficient enough for visualization?, Is it possible to map these data into a chart? Designing a standard vocabulary library leads to answer these queries by providing visualization information like name, labels and types etc. The *VoVis* Vocabulary contains 30 types of visualization which are listed in Appendix A. Following are the descriptions for some visualization types along with their vocabulary.

1. Area graph
   a) Description - Area Graph is a type of line chart that draws area with colors. To plot it, first lines are drawn with Cartesian coordinate points and then are filled with some colors. They are used to show or display quantitative data. There are two variations of this graph: Grouped and stacked Area graph.
   b) Vocabulary - To plot a simple area graph most important is the coordinate axes. Area graph works with 2D data, so vocabulary is simple with x and y axis.
      - name: name of chart.
      - xaxis_label: xaxis label.
      - yaxis_label: yaxis label.
      - xaxis_type: data type for x values.
      - yaxis_type: data type for y values.
      - xvalues: values for xaxis.
      - yvalues: values for y axis, one for each x value.
   c) Functions - Area graph can be used to demonstrate following functions - Relationships, Patterns and Data over time.

Figure 21 shows an D3 example of an area chart[5].

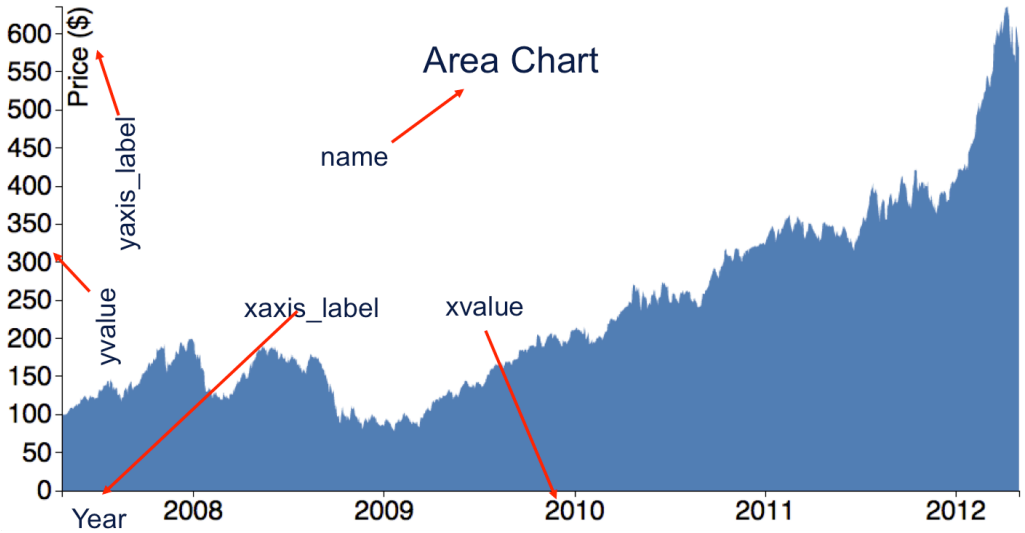


Figure 21: D3 example of Area Chart

2. Bar chart
   a) Description - A bar chart presents data as rectangular blocks, can be displayed as horizontal or vertical rectangular bars. They are used mainly for comparisons of data. Bar cart has two variations, Grouped and stacked bar charts, both work on multiple data. Bar

5 Area Chart: http://bl.ocks.org/mbostock/3883195

cart is defined as "chart that provides a visual presentation of categorical data" [52]. Categorical data is a grouping of data into qualitative groups, such as months of the year, age group, shoe sizes, and animals. These categories are usually qualitative.

b) Vocabulary - To plot a simple bar graph most important is the coordinate axes. Area graph works with 2D data, so vocabulary is simple with x and y axis.
   - name: name of chart.
   - xaxis_label: xaxis label.
   - yaxis_label: yaxis label.
   - xaxis_type: data type for x values.
   - yaxis_type: data type for y values.
   - xvalues: values for xaxis.
   - yvalues: values for yaxis, one for each x value.

c) Functions - Bar Charts can be used to demonstrate following functions - Comparisons, Relationships, and patterns.

Figure 22 shows an example of bar chart taken from Google Charts[6].

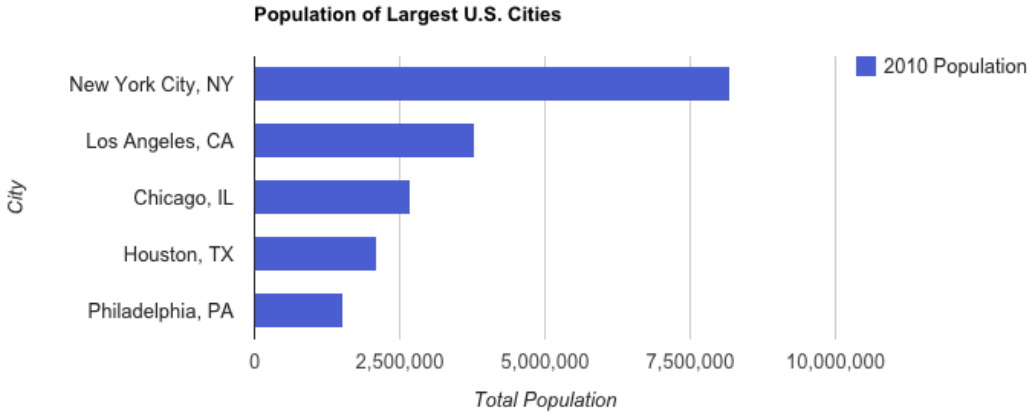


Figure 22: Bar Chart example from Google Charts shows populations of US cities

3. Box and Whisker Plot

a) Description - A box plot is a visualization type that displays numerical data in groups using the concept of quartiles. First quartiles need to be calculated from data and then shown as boxes and whiskers. Additional features can also be highlighted by using some parallel lines. Outliers are sometimes plotted as individual dots that inline with whiskers [54]. Box plots can be drawn either vertically or horizontally. Box plot can provide information about average value, values of outliers in data sets.

6 Bar Chart: https://developers.google.com/chart/interactive/docs/gallery/barchart

b) Vocabulary - To plot a simple boxplot, most important is the x axis and its values.

- *name: name of chart.*
- *xaxis_label: xaxis label.*
- yaxis_label: yaxis label.
- xaxis_type: data type for x values.
- xvalues: values of x.

c) Functions - Box chart can be used to demonstrate the following functions - distribution, Range. When box chart is grouped, it can demonstrate, comparisons and patterns.

Figure 23 shows an example of box plot[7].

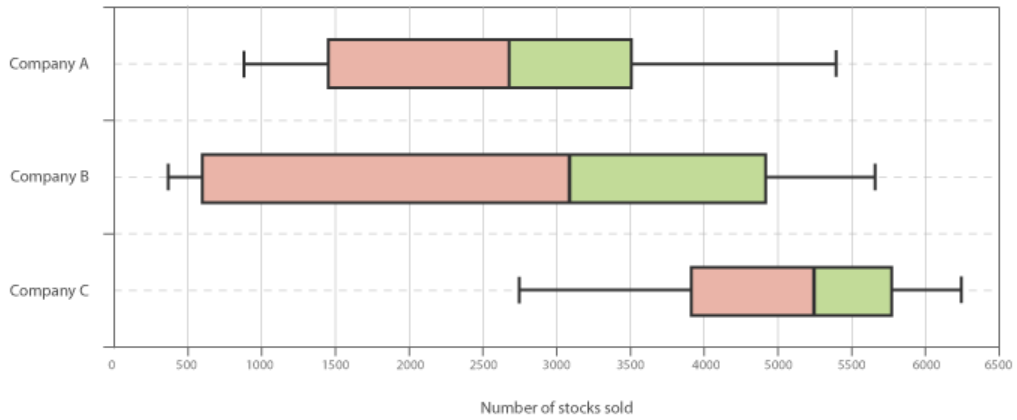


Figure 23: Box and Whisker Plot

4. Bubble chart

a) Description - A Bubble Chart can display up to four dimensions of data, bubble chart can sometimes resemble a combination of a Scatter plot and a Proportional Area Chart. Bubble Charts use a Cartesian coordinate system to plot bubbles together with separate x and y axis values. Each bubble represents a third variable, either by the area of it or by color. Colors can also be used to distinguish between categories or used to represent an additional data variable. Each entity with its triplet or quad-let (v1, v2, v3, v4) of associated data is plotted as a circle that expresses two of the vi values through the circle's xy location and the third through its size and fourth through its color.

b) Vocabulary - To plot a bubble chart, the data should be multidimensional.

- name: name of chart.
- xaxis_label: xaxis label.

7 Box Plot: http://www.datavizcatalogue.com/methods/box_plot.html#.VVUCY-mUc8o

- yaxis_label: yaxis label.
- xaxis_type: data type for x values.
- xvalues: x values.,
- yaxis_type: data type for y values.
- yvalues: y values for each x value.
- color_label: label which different colors will represent.
- color_type: data type for color values.
- color_values: values each for a color.
- radius_label: label which is shown by different circle radius.
- radius_type: data type for radius value.
- radius_values: values for radius for each circle.

c) Functions - Bubble Charts are typically used to compare and show the relationships between circles. The bubble chart can be used to demonstrate the following functions - Comparisons, Data over time, Distribution, Patterns, Proportions, Relationships.

Figure 24 shows an example of bubble chart taken from Google Charts examples[8].

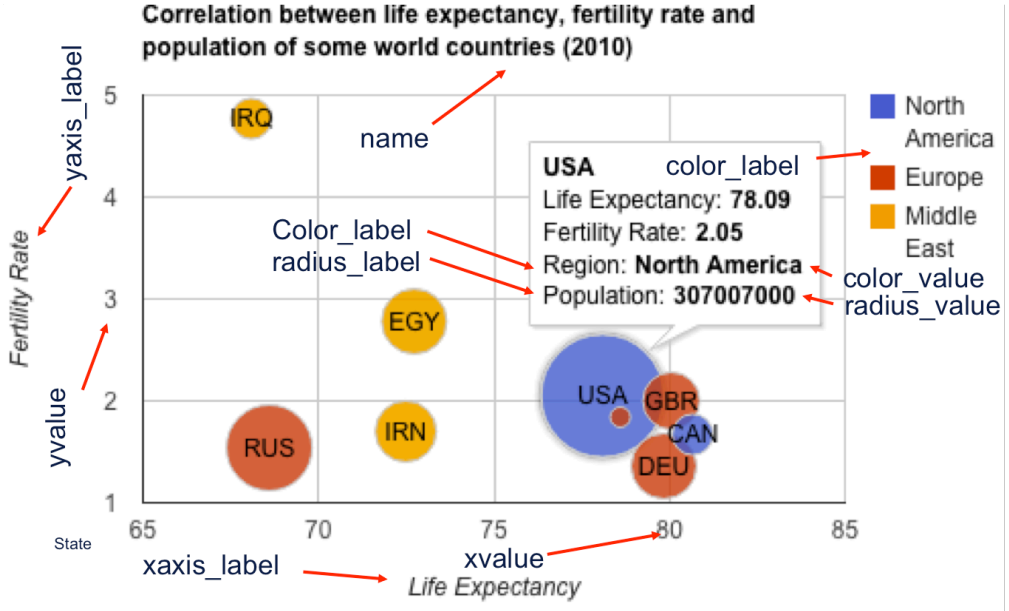


Figure 24: Bubble Chart from Google Charts Example

5. Histogram

a) Description - A histogram is a visualization type that displays data over time period or over continuous interval. Data is displayed as bars where each bar represents some data for each interval. They

8 Bubble Charts: https://developers.google.com/chart/interactive/docs/gallery/bubblechart

are used in the field of probability distribution. Histogram works with continuous data

b) Vocabulary - To plot a simple histogram chart, multiple data columns are required.
   - name: name of chart.
   - xaxis_label: name of xaxis.
   - xaxis_type: data type for x values.
   - xvalues: x values.
   - x_labels: labels for x values.

c) Functions - Histogram can be used to demonstrate the following functions - Comparisons, Data over Time, Distribution, Patterns, Probability, and Range.

Figure 25 shows an example of Histogram taken from Google Charts examples[9].

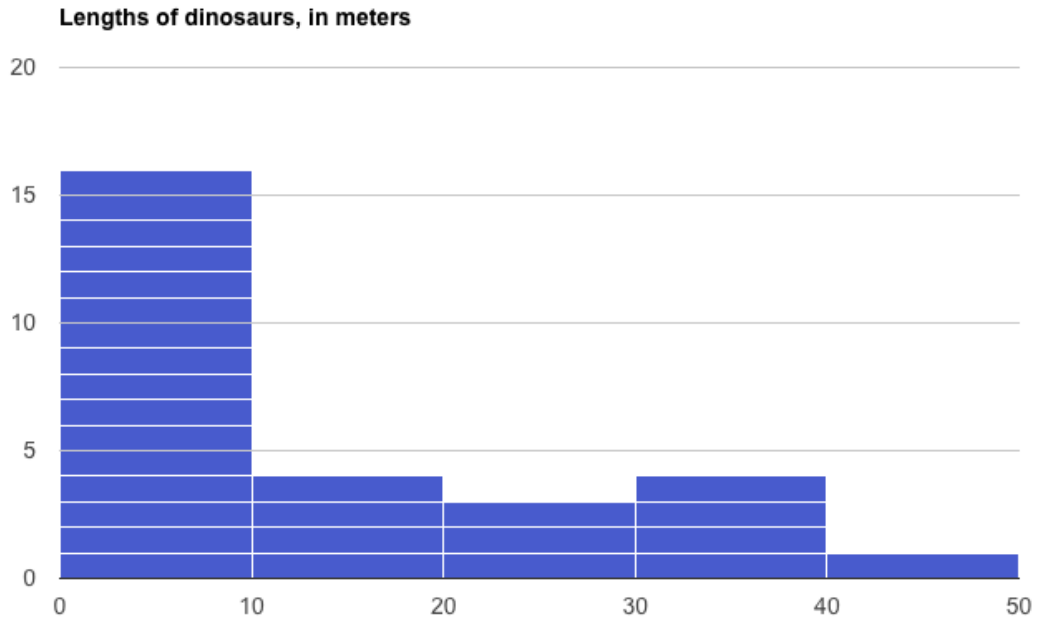


Figure 25: Histogram

6. Multi-set Bar chart

a) Description - Multi-set or Grouped Bar charts are used to display data which are grouped together under some categories, are plotted on same axis. Each group is presented as bar with a color, to distinguish it from other groups. Bunch of bars are spaced apart from each other based on categories. The use of multi-set bar charts is to compare grouped variables or categories.

9 Histogram: https://developers.google.com/chart/interactive/docs/gallery/histogram

b) Vocabulary - To plot a simple multi-set bar chart, one needs multiple data columns.
   - name: name of chart.
   - xaxis_label: xaxis label.
   - xaxis_type: data type for x values.
   - yaxis_label: yaxis label.
   - xvalues: values for xaxis.
   - y_names: title for y values.
   - yaxis_type: data type for y values.
   - yvalues: values for y for each x value.
   - y_labels: labels for each y_name

c) Functions - Grouped bar chart can be used to demonstrate the following functions - Comparisons, Distribution, Patterns, Relationships

Figure 26 shows an example of multi-set bar chart from D3 examples[10].

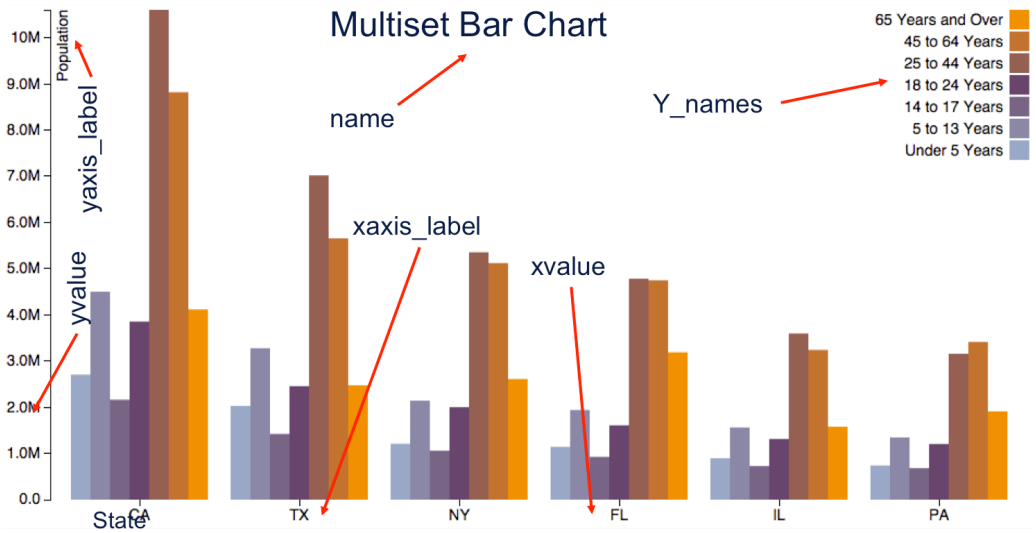


Figure 26: Multi-set Bar Chart from D3 example

7. Population pyramid

a) Description - A population pyramid is a pair of histograms that display population patterns. The Population pyramid can show distribution of the population of different groups, groups can be categorized as sex or age group. The shape of a population pyramid provides information, which helps to interpret a population. For example, "an pyramid with a very wide base and a narrow top section suggests a population with high fertility and death rates.

10 Multi-set Bar: http://bl.ocks.org/mbostock/3887051

Whereas, a pyramid with a wider top half and narrower base suggests an aging population with low fertility rates" [5]. Population pyramids are useful in field of Ecology and Sociology.

b) Vocabulary - For each x axis value there are two y variables (y1,y2) with two values.
   - name: name of chart.
   - xaxis_label: xaxis label.
   - yaxis_label: yaxis label.
   - xaxis_type: data type for x values.
   - xvalues: values for xaxis.
   - y1name: label for y1 values.
   - yaxis_type: data type for y values.
   - y1values: values for y1 one for each x values .
   - y2name: label for y2 values.
   - y2values: values for y2 one for each x values.

c) Functions - Population Pyramid can be used to demonstrate the following functions - Comparison, Distribution, Patterns. Figure 27 shows an example of Population pyramid taken from D3 examples[11].

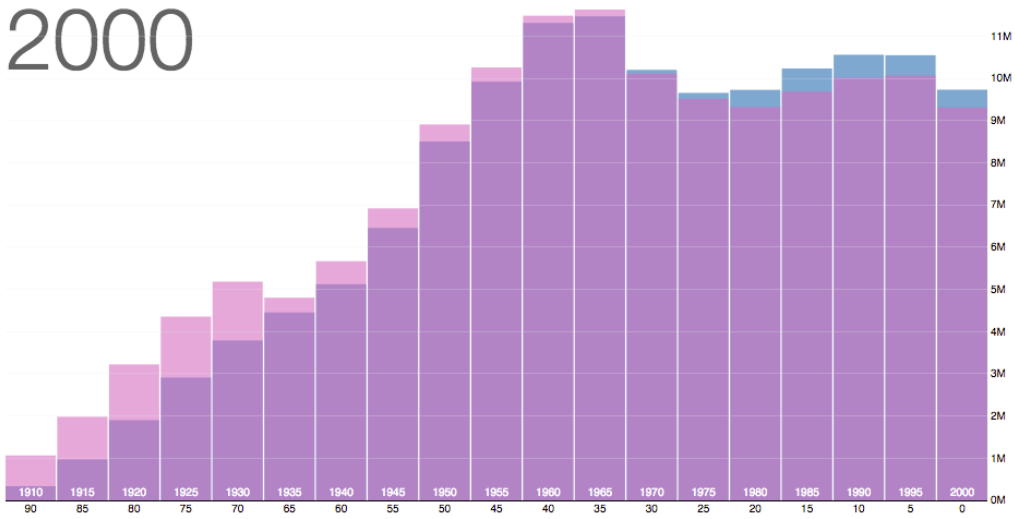


Figure 27: Population Pyramid example from D3

8. Radial Bar chart

a) Description - A Radial/Circular Bar chart is kind of a Bar chart but plotted on a polar coordinate system. The human can not interpret polar coordinates easily, as compared to straight lines, so

11 Population Pyramid: http://bl.ocks.org/mbostock/4062085

the ordinary user does not prefer this chart. Mainly used for aesthetic reasons.

b) Vocabulary - Vocabulary is based on the polar coordinates.
   - name: name of chart.
   - r_label: label for r axis.
   - dimension_label: dimension label.
   - r_type: data type for r values.
   - r_values: values for r.
   - dimension_labels: labels for dimensions.
   - dimension_type: data type for dimension values.
   - dimension_values: values for dimension.

c) Functions - Radial bar chart can be used to demonstrate the following functions - Comparison, Relationships.

Figure 28 shows an example of radial bar chart taken from D3 examples [8].

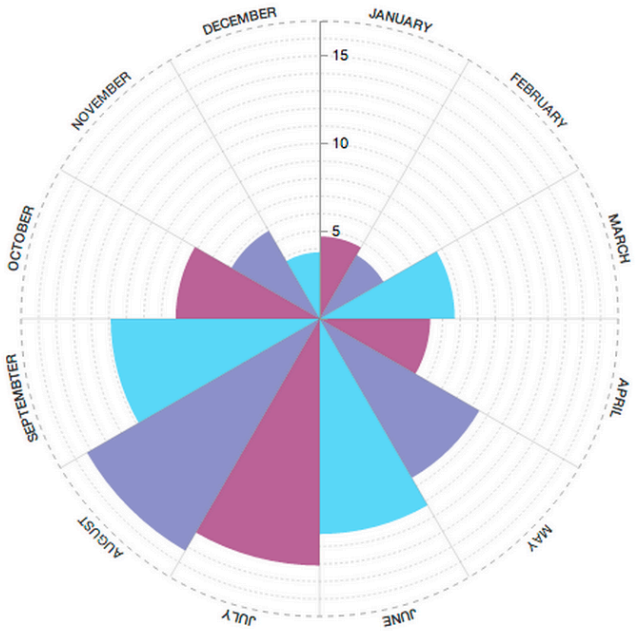


Figure 28: Radial Bar Chart example from D3

9. Scatter Plot

a) Description - This kind of plot is also known as scatter graph, point graph, X-Y plot, scatter chart or scattergram. Scatter plots use a collection of points placed in a graph using Cartesian coordinates. Each point represents two variables values w.r.t to the axis. Correlation between these two variables can be observed if it exist. Interpretation of the relationship between the points can be

done by seeing the pattern created by them in the plot. These are positive: variable values increase together, negative: one variable value decreases as the other increases, null (no correlation), linear, exponential and U-shaped.

b) Vocabulary -
   - name: name of chart.
   - xaxis_label: xaxis label.
   - yaxis_label: y axis label.
   - xvalues: x valuess.
   - yvalues: yvalues for each x values.
   - color_label: label which different colors will represent.
   - color_values: color values.

c) Functions - Scatter plot can be used to demonstrate the following functions - Patterns, Relationships.

Figure 29 shows an example of Scatter plot chart taken from D3 examples[12].

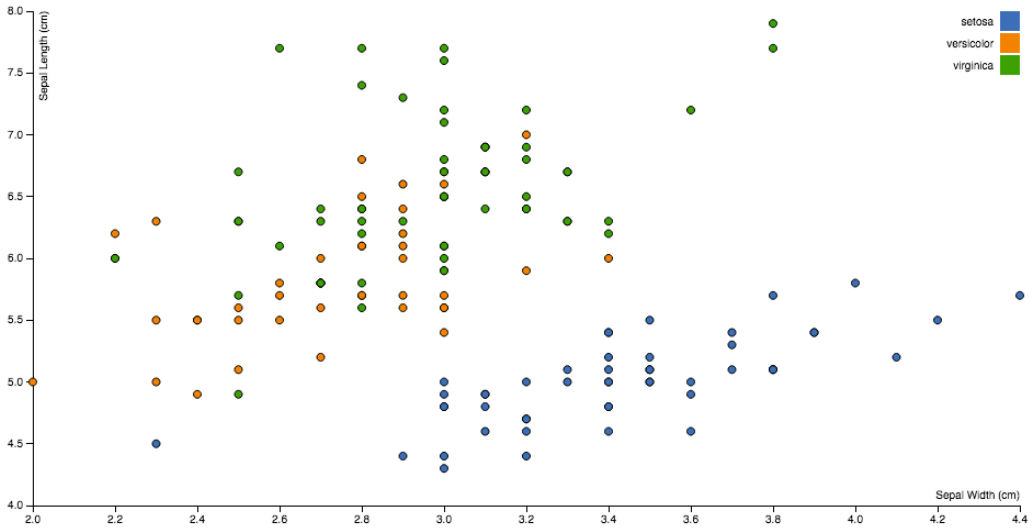


Figure 29: Scatter Plot example from D3

10. Span chart

a) Description - Span chart is a chart that displays different dataset ranges like time range or range between minimum and maximum value from data, for different categories. Span chart does not give information for all values in data; the focus is on the range of dataset. Span chart is also known as Range Bar/Column Graph.

b) Vocabulary - For each x axis value there are two y variables (y1,y2) with two values.

12 Scatter Plot: http://bl.ocks.org/mbostock/3887118

- name: name of chart.
- xaxis_label: x axis label.
- yaxis_label: y axis label.
- xvalues: different values for x axis.
- yname: same label for y1 and y2.
- y1values: values for y1 one for each x values.
- y2values: values for y2 one for each x values.

c) Functions - Span chart can be used to demonstrate the following functions - comparisons, and Range.

Figure 30 shows an example of Span chart[13].

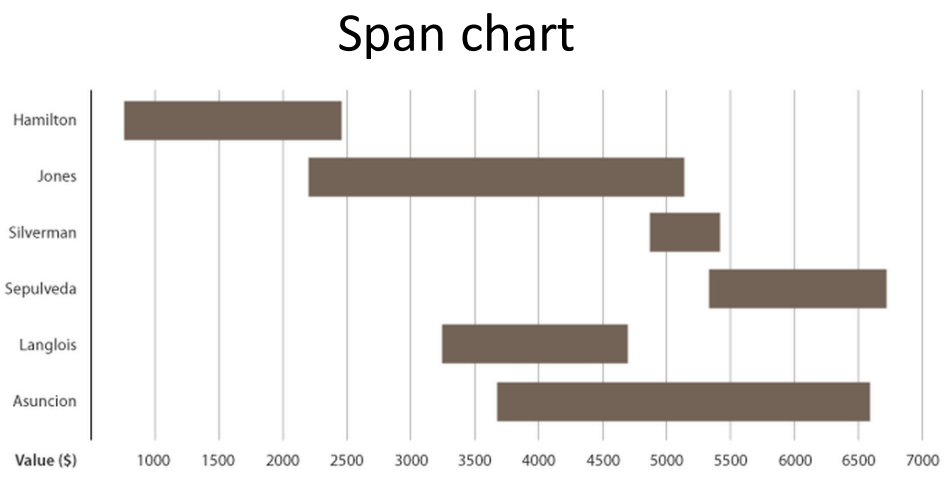


Figure 30: Span Chart example from The Data Catalogue

11. Arc diagram

a) Description - Arc Diagram is a style of graph drawing [51]. Arc diagram is a graph that represents 2D data as nodes and arcs between nodes, to show the flow between nodes. The frequency from the source node and target node is represented by the thickness of arc line.

b) Vocabulary - Vocabulary of Arc diagram is based on nodes and link concept of graph theory. It has two cases, with information about nodes or without it.

i. Case 1:

- node (name)
- link (source,target,value)

13 Span Chart: http://www.datavizcatalogue.com/methods/span_chart.html#.VVUUXumUc8o

ii. Case 2:

- node (name,info)
- link (source,target,value)

c) Functions - Arc Diagram can be used to demonstrate the following functions - patterns, relationships.

Figure 31 shows an example of arc diagram taken from D3 examples[14].

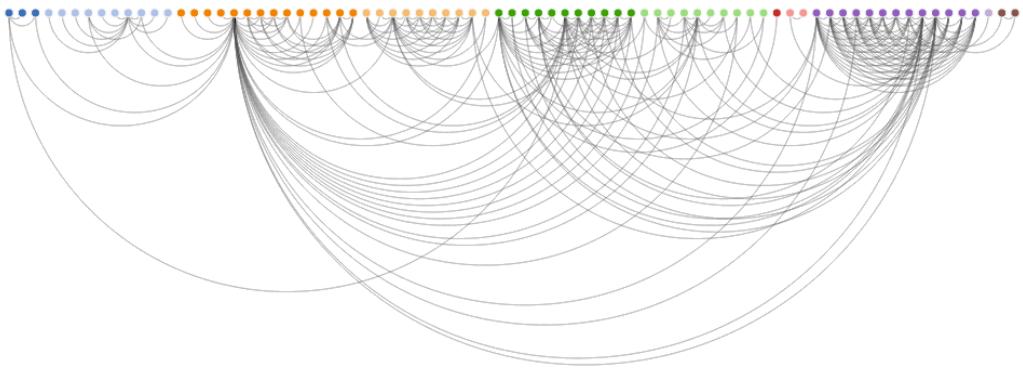


Figure 31: Arc Diagram from D3 example

12. Venn Diagram

a) Description - A Venn Diagram is a diagram that displays logical relations between a group of sets; each set is displayed as a circle. Venn diagram is used to show different functions over the sets, like intersection, union, etc. For intersection, different sets overlap with each other.

b) Vocabulary - A Circle has name and size value, with overlap function.

- circle (name,size)
- overlap (circle1,circle2,size)

c) Functions - Venn Diagram can be used to demonstrate the following functions - comparisons, concepts, probability, relationships.

Figure 32 shows an example of Venn diagram taken from D3 examples[15].

13. Pie Chart

a) Description - Pie chart is a circular graphic visualization used in offices. The pie or circle is divided into slices of arc length for each slice. Each arc length represents a proportion their category. This chart is not suitable for larger datasets. Pie charts take more space to display less information.

14 Arc Diagram: http://bl.ocks.org/sjengle/5431779
15 Venn Diagram: http://www.benfrederickson.com/venn-diagrams-with-d3.js/

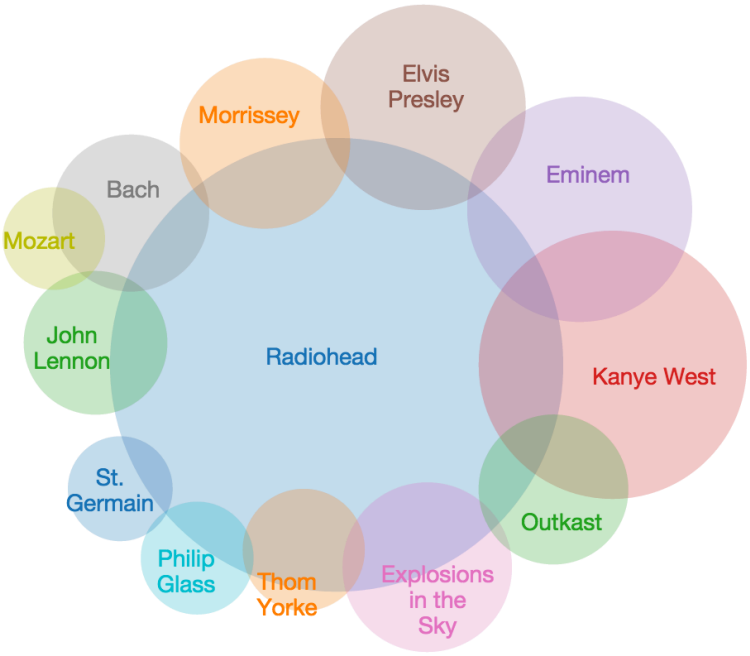


Figure 32: Venn Diagram

b) Vocabulary - Pie chart has radius and dimension values.
   - name: name of chart.
   - r_label: name for r axis.
   - dimension_label: dimension name.
   - r_type: data type for r values.
   - r_values: values for r.
   - dimension_type: data type for dimension values.
   - dimension_values: values for dimension.

c) Functions - Pie Chart can be used to demonstrate the following functions - comparisons, part-to-a-whole, proportions.

Figure 33 shows an example of a pie chart taken from Google Charts gallery[16].

16 Pie Chart: https://developers.google.com/chart/interactive/docs/gallery/piechart

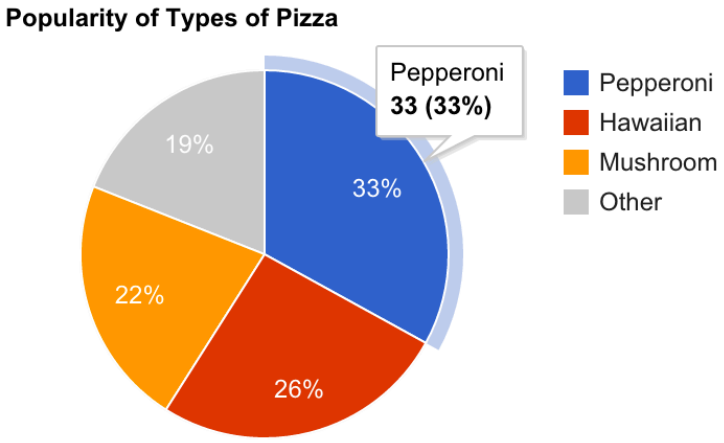


Figure 33: Pie Chart Example from Google Charts

14. Geo charts
    a) Description - Geo charts are map drawn with lines; they are not terrain map and cannot zoom or scale the maps. They can be subdivided as choropleth map and geo chart with markers. Choropleth map displays different regions with colors in relation to data values. Geo charts with markers need additional information to display markers. Most important inputs required are either region's name or longitude and latitude values.
    b) Vocabulary - To plot a geo chart longitude, latitude values are needed or the name of countries or cities
        - Longitude: data type for y values,
        - Latitude: yvalues for each x values,
        - color_data_label: label which different colors will represent,
        - color_data_type: data type for color values,
        - color_data_values:values each for a color,
        - size_data_label:label which is shown by different circle radius,
        - size_data_type: data type forradius values,
        - size_data_values:values for radius for each circle
    c) Functions - Geo Chart can be used to demonstrate the following functions - Comparisons, Data over time, Distribution, Patterns, Proportions, Relationships

Figure 34 shows an example of Geo chart taken from Google Charts[17]

The next layer is the JSON structure library. The vocabulary library provides the input to a mapping component whose function is to map input data with vocabulary. Input data together with mapped vocabulary is then converted into JSON structure. The overall conversion component is explained in the Chapter 7.

17 Geo Chart: https://developers.google.com/chart/interactive/docs/gallery/geochart

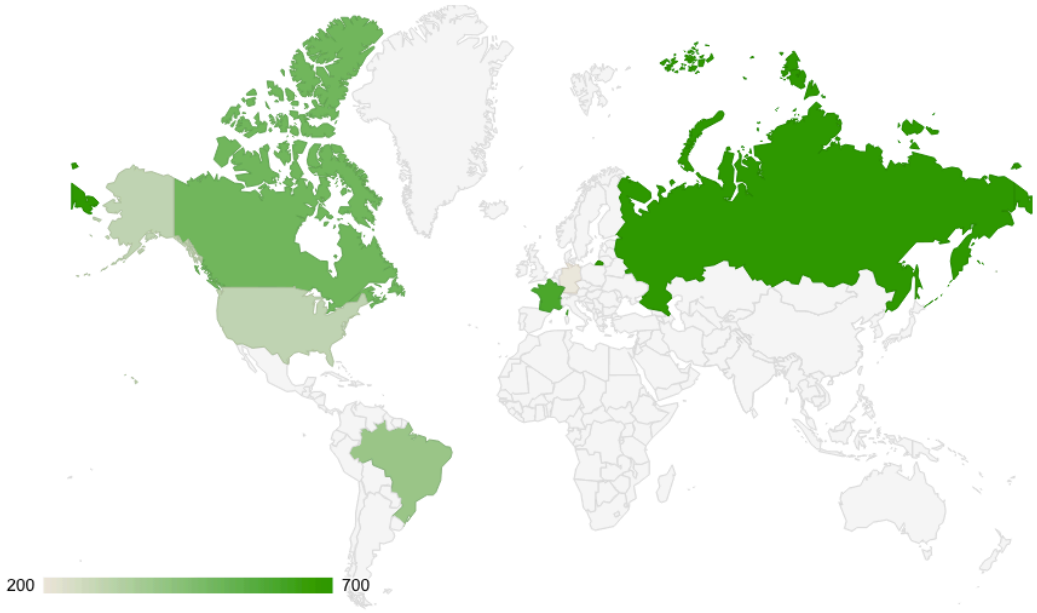


Figure 34: Geo Chart Example from Google Charts

6.3.4 *Vocabulary Storage Format*

Before explaining the details of the library, this section provides motivation behind creating this library. Most of existing visualization frameworks are rigid and accepts the input data in a specific format. Following are the motivation behind including this component in the framework.

1. The standard structure of the data provides the opportunity to the the users or the developer to understand it easily.

2. The idea of this framework is not only to provide visualization along with process data in standard format. It is important to have data in standard format for compatibility reason with other third party tools.

3. Once the input data is mapped with vocabulary it needs to be converted into some structure. Depending on vocabulary the mapped data can be simple or complex, so some standard format is required.

From above discussion it is clear that we require a standard structure. There are plenty of existing data formats that can be a potential candidates for our requirement. This includes CSV, XML, JSON etc. CSV or any similar tabular form is a not feasible option as they cannot hold all properties from the vocabulary. For example, some cases requires simple key value pair whereas a few may require array of it and so on. Structures such as XML format, JSON format has features to accommodate these requirements. For the purposed framework, the JSON structure is preferred over XML. The reason behind choosing the JSON structure is that it is simpler than XML, as JSON uses less grammar and maps directly to the data structures used in modern programming languages such as JavaScript. Note that our framework uses

JavaScript libraries hence JSON becomes more suitable candidate. JSON is also human readable therefore more suitable for the exchange format. Depending on the chart being selected from the visualization library, a vocabulary is selected and mapped with the input data. Further it get converts into JSON object that follows the JSON structure from the library. Following is the vocabulary and JSON structure for an Area chart.

Listing 2: JSON Structure for a single Area/Line/bar Chart

```
{  "values": [
   {"x":,"y":},
   {"x":,"y":},
  ] , "name":  , "xaxis_label":  ,
    "xaxis_type":   , "yaxis_label": ,
    "yaxis_type":
}
```

The task of conversion component is to convert the vocabulary data into the JSON object. The conversion function is written in JavaScript and the result is available to for the third party visualization tool.

In conclusion, this chapter describes the design and concept of the *VoVis* framework followed by the detail description of vocabulary components. Some of the visualization types along with their vocabulary.

7

PROTOTYPE IMPLEMENTATION OF THE VOVIS FRAMEWORK

The *VoVis* framework is introduced and designed along with the vocabulary library in the Chapter 6. The visualization vocabulary module can be developed as a separate component within the web-based framework, so this module can be easily used by the other visualization frameworks. Appendix A shows the visualization vocabulary in text form for different types of visualization along with the examples.

For proof of concept, a web-based visualization framework prototype has been implemented. The *VoVis* framework is implemented as a client-side MVC framework using JavaScript language. The main aim of this prototype is to show that vocabulary based visualization makes a difference in visualizing the data. Due to time constraints, the prototype has only five types of visualization in the library with vocabulary. For Visualization, D3[1] and Google charts[2] are selected as third party libraries according to the evaluation of visualization libraries in Section 5.2.5. The main reason to choose two completely different libraries is to demonstrate, how generic the *VoVis* framework is. Processed data in the *VoVis* framework can be visualized easily with any third party library. The complete framework is written in JavaScript language that makes it very flexible and platform independent [28]. The framework has both a web based version and a mobile app version. All modules are implemented according to the design of components and follow the architecture discussed in Chapter 6. The Figure 35 demonstrates the whole framework structure with all components.

7.1 THE CONTROLLER: GRAPHER

The controller of the framework is Grapher.js, It coordinates with all the components and the user interface. Every component interact with other only through Grapher. The primary task of this controller is to decide when and which process to execute and with what data. All the components depend on it for their input to perform a particular task they are assigned to. The following steps describe how the controller manages the whole workflow.

1. Grapher uploads the file that the user selects to upload through the user interface(Home page) of the framework.

1 Data Driven Documents : http://d3js.org/
2 Google Charts : https://developers.google.com/chart/

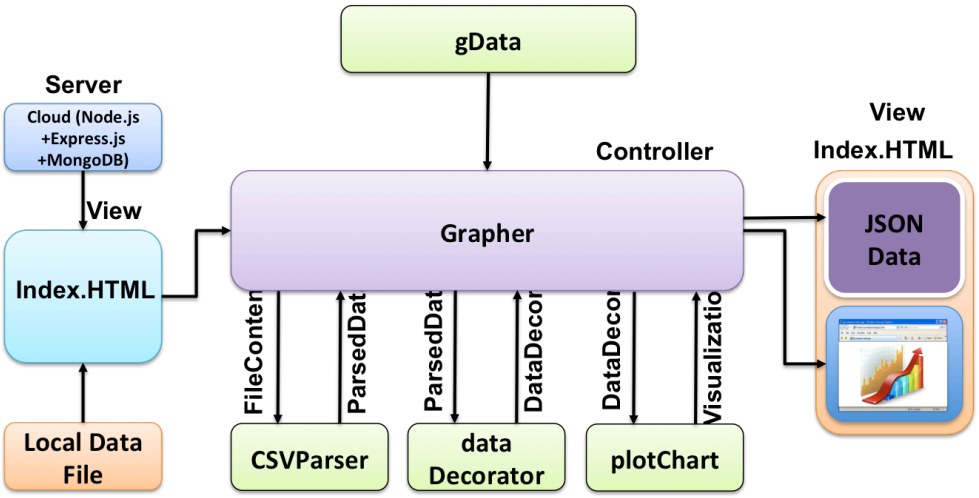


Figure 35: Process flow of the *VoVis* framework

2. Grapher assigns the visualization type that the user has selects, according to the "type" the Controller picks the correct vocabulary from the gData (config file).

3. Grapher passes the vocabulary to the user through the dataDrag module. Then the user maps the data with the vocabulary.

4. When the user is finished with mapping, next step is to process the data. The Controller transfers the vocabulary mapping information along with abstract data to the dataDecorator module.

5. When the dataDecorator finishes its task, it returns the processed data to Grapher. The Grapher displays the processed data and forwards to the plotGraph module.

6. When the plotGraph executes, the data is visualized on the user interface(Home Page) of the framework.

This way the controller plays a critical role in every step from beginning (file uploading) to the end (visualization of data).

7.2 THE VOCABULARY CONFIGURATION FILE: GDATA

The gData is like a configuration file; that contains vocabularies for types of visualization. For example, the bubble chart has a vocabulary which includes name label for chart name, x-axis_label, y-axis_label, color_label, and radius_label labels. Data type is not mentioned here because it is identified and assigned at the run time when the user uploads the data. Charts with multiple, x or y columns(multi-bar chart or box chart) need a field that defines the value of n, is configured in this file. The user will enter the value for n and according to the value columns will be available to drag and drop in Home page.

7.3 THE USER INTERFACE: HOME PAGE

The essential module in the web-based framework is Home page(index.html). This is the user interface, where final visualization is displayed. The user uploads a data file through it and chooses the type of visualization. This page provides mapping of data columns with vocabulary through drag-drop service. When the framework executes and creates the final visualization, it is presented in this page along with the processed data.

7.4 THE DATA ANALYZER: CSVPARSER

The work of CSVParser is to parse the raw data. This JavaScript is taken from Dr Converter [3] which is open source. This JavaScript processes the raw data from a file and converts them into arrays. As the name suggests it works on the data in CSV format, stores the data from a file into arrays and is together called the abstract data. This abstract data is saved in the gData file.

- dataArray - will store the data values from all columns in a 2D array.
- headersName - will store all the header names, i.e., the first row from CSV file.
- headerTypes - will store the data type of each data column. If the first column is "names of cities" then header type will be string for the first column, similarly it will be numbers if some column has "population of cities" in numbers. Figure 36 describes the flow of operation, the CSVParser executes on raw data.

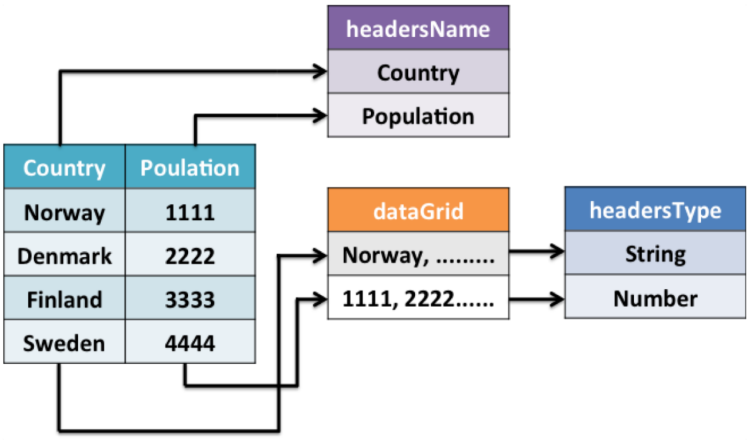


Figure 36: Data Analyzed by CSVParser

3 Dr Converter: http://shancarter.github.io/mr-data-converter/

7.5 THE VISUAL MAPPER: DATADECORATOR

The dataDecorator is visual mapper component of the framework. This component maps the vocabulary based data to a standard JSON object, which makes this framework unique as compared to other visualization frameworks. The dataDecorator has JSON structure library for all visualization vocabularies. When it receives vocabulary mapped data, it selects particular JSON structure and then converts the data into a JSON object. Any non-technical person will easily correlate this processed data to the visualization. Figure 37 describes the process of data mapping.

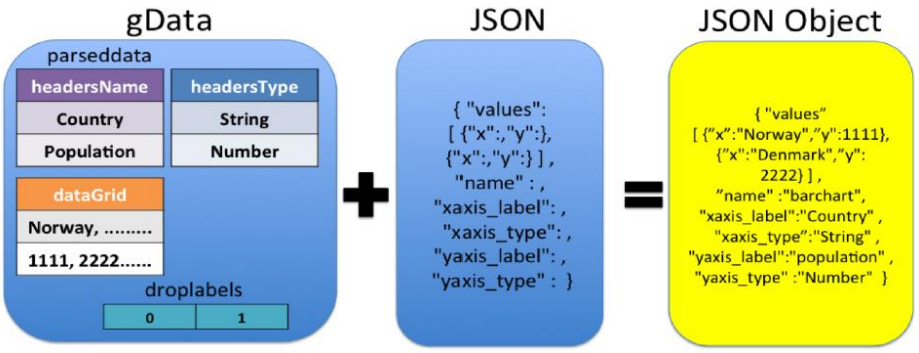


Figure 37: Data Mapped by DataDecorator

7.6 THE VISUAL DISPLAYER: PLOTCHART

The plotChart is the visual displayer component. The plotChart creates the visual chart with the help of third party visualization JavaScript libraries. The prototype plots few charts and maps using D3 and Google Charts libraries. The D3 library is complicated to use than the Google charts, as D3 do not offer predefined visual styles. In Google Charts, the chart components are already created, the properties can be changed using pre-defined option objects according to the needs. The D3 follows the opposite approach, where everything needs to be built from scratch. Using D3, it is possible to make virtually any type of visualization and can be customized to any extent. To learn D3.js, first step is to understand different D3 API which are available at the official API documentation[4], and the examples are listed on the D3 wiki[5]. The D3.js wiki is full of tutorials, blogs, and talks, also there are few good books that helped a lot in understanding D3 functions especially Interactive Data Visualization for the Web [34] and Getting Started with D3 [15]. The book D3 Tips and Trick [32] provided more practical tips on D3 library.

Like jQuery and other JavaScript frameworks D3 simplifies the selection of an element in the SVG DOM. The static method "d3.select()" takes CSS selector as an argument. The "d3.append()" method is similar but is used to add a

4 https://github.com/mbostock/d3/wiki/API-Reference
5 D3 wiki : https://github.com/mbostock/d3/wiki

new element as a child of the selected element. To append an SVG element, it can be prefixed by the SVG namespace like this: d3.append("circle").

D3 uses a declarative style of programming, and it needs to declare what is selected. The modifications are made in the DOM, CSS styles, HTML and SVG attributes, texts, animations, and events. D3 will loop through the selection set and apply the modifications and provides abstraction by hiding the DOM manipulation process. This leaves the user with the fun part and with full control on the process.

Binding events is a simple process, selects a DOM element and use the ".on()" method with two attributes. The first attribute is an event(as string), and a callback function as a second attribute.

Example: ".on" method is shown below with two attributes

```
.on("mouseover", function()
    {
        d3.select(this).style("fill", "white");
    })
```

Quite often, this type of anonymous function is seen as an argument in D3. Figure 38 shows the process of chart generation by plotChart.

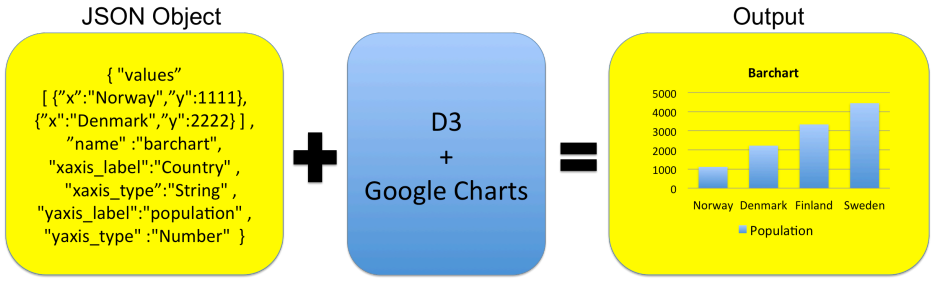


Figure 38: Chart Generated by plotChart

7.7 THE VOVIS: DATABASE

To store the data files that are uploaded by the user to visualize and also the processed data, MongoDB[6] as the database is selected. "MongoDB is one of many cross-platform document-oriented databases" [56]. It is a NoSQL database. MongoDB is a document-oriented database, it does not work like table-based relational database, instead uses documents with JSON-like structure and dynamic schemas (also called as BSON structure). Use of such format can make data integration easier and faster for applications. *VoVis* together with MongoDB provides the option to upload the data and download it. With this service, a user can analyze and process data anywhere, anytime. The uploaded data is stored in the cloud by MongoDB, so it is a data service provided by a web service. In order to access the MongoDB server is needed, which can connect the application to the database.

6 MongoDB: https://www.mongodb.org/

7.8 THE VOVIS: SERVER

A server is required to host the *VoVis* application for visualizing data, and a web service for accessing MongoDB. Node.js[7] is selected to fulfill above requirements. Node.js [43] provides http server to host the *VoVis* application and web services required to serve the web app and handle the back-end data storage/retrieval via MongoDB. "Node.js is an open source, a cross-platform runtime environment for server-side and networking applications. Node.js provides an event-driven architecture and a non-blocking I/O API that optimizes an application's throughput and scalability" [57]. Node.js applications are written in JavaScript. These technologies are commonly used for real-time web applications. In Rapid prototyping world, node.js environment has been proven one of the best full development stacks. With unified API approach, it is easy to deploy a complete Web-based solution in multiple platforms. Node.js combined with a browser, a document DB (MongoDB) and JSON offers a unified JavaScript development stack.

7.9 THE VOVIS: SOURCE CODE

The *VoVis* framework prototype is implemented as a client-side MVC web application for visualizing data with vocabulary based mapping. The *VoVis* prototype is an open source application, easy to download and run the application. The user can edit the source code of the *VoVis* application to modify the application according to the requirement. The source code of the *VoVis* application is available on GitHub [50]. The steps to download and run the *VoVis* framework are described in Appendix C.

Chapter 8 describes the test of the prototype to evaluate the *VoVis* framework.

7 Node.js : https://nodejs.org/

Part III

EVALUATION AND CONCLUSION

8

EVALUATION OF THE VOVIS FRAMEWORK

The functional prototype of the *VoVis* framework needs to be tested to determine whether all requirements listed in Chapter 4 are fulfilled. This prototype addresses the last research task of the thesis and discusses the experiment performed using the *VoVis* framework.

Finally, the *VoVis* framework is evaluated with respect to all requirements defined in Chapter 4. This evaluation will determine to what extent the *VoVis* framework meet the requirements.

8.1 THE VOVIS FRAMEWORK EXPERIMENTAL SETUP

In order to perform the test, certain datasets are taken from the use cases that are discussed earlier in Section 4.1. Then desired charts and maps are plotted from these datasets using the *VoVis* framework. The outcome of the experiment is reported to provide help in the assessment of the *VoVis* framework.

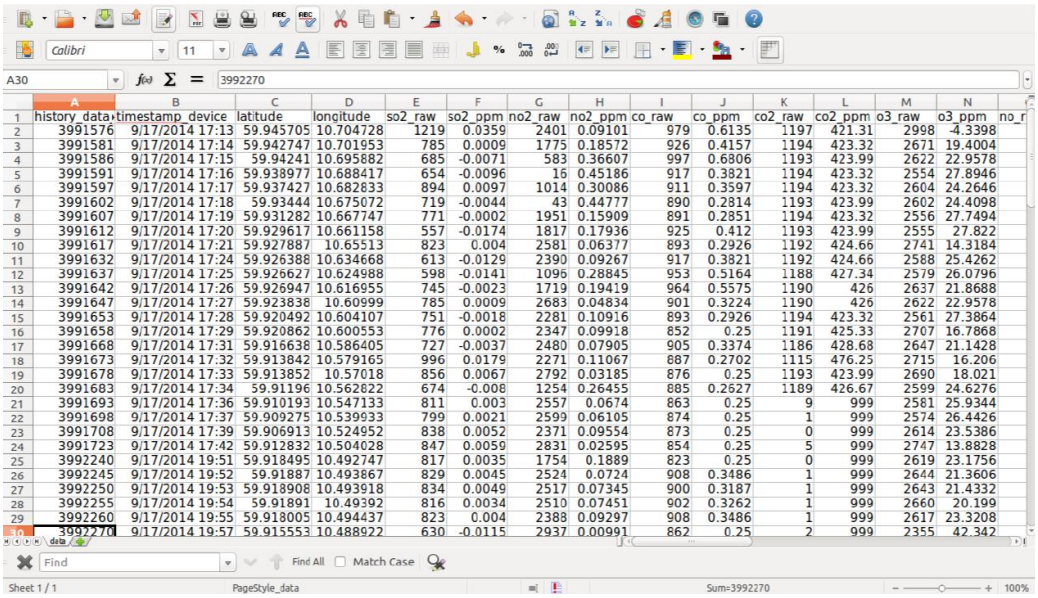
In order to test the *VoVis* framework, the first step is to get data which needs to be visualized. Out of two use cases from pilot projects, data from one of the project is taken to perform the experiment. Citi-Sense-MOB provides air quality data to present in the different types of visualization.

Air quality data consists of air pollutants in the air like NO2 gas, Co gas, etc., together with other measures like time, temperature, humidity, noise level, measured by different sensors. Data is measured using different units like raw, ppm, ppb, etc. The data is measured for Citi-Sense-MOB project by two ways.

1. Fixed sensors mounted at different locations in the Oslo city measures and records all the air pollutants continuously.

2. A mobile sensor collects the data, for example, a sensor is mounted on a bicycle. When a cyclist takes a ride from one location to the other, the data is measured by sensors throughout the ride, which provides the data for a particular track.

Figure 39 outlines the different air pollutants, measured for a track covered by the cyclist, where the sensor is mounted on a bike. The data shown below has many columns, so different types of visualization can be created to visualize the different column data.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	history_data	timestamp_device	latitude	longitude	so2_raw	so2_ppm	no2_raw	no2_ppm	co_raw	co_ppm	co2_raw	co2_ppm	o3_raw	o3_ppm
2	3991576	9/17/2014 17:13	59.945705	10.704728	1219	0.0359	2401	0.09101	979	0.6135	1197	421.31	2998	-4.3398
3	3991581	9/17/2014 17:14	59.942747	10.701953	785	0.0009	1775	0.18572	926	0.4157	1194	423.32	2671	19.4004
4	3991586	9/17/2014 17:15	59.94241	10.695882	685	-0.0071	583	0.36607	997	0.6806	1193	423.99	2622	22.9578
5	3991591	9/17/2014 17:16	59.938977	10.688417	654	-0.0096	16	0.45186	917	0.3821	1194	423.32	2554	27.8946
6	3991597	9/17/2014 17:17	59.937427	10.682833	894	0.0097	1014	0.30086	911	0.3597	1194	423.32	2604	24.2646
7	3991602	9/17/2014 17:18	59.93444	10.675072	719	-0.0044	43	0.44777	890	0.2814	1193	423.99	2602	24.4098
8	3991607	9/17/2014 17:19	59.931282	10.667747	771	-0.0002	1951	0.15909	891	0.2851	1194	423.32	2556	27.7494
9	3991612	9/17/2014 17:20	59.929617	10.661158	557	-0.0174	1817	0.17936	925	0.412	1193	423.99	2555	27.822
10	3991617	9/17/2014 17:21	59.927887	10.65513	823	0.004	2581	0.06377	893	0.2926	1192	424.66	2741	14.3184
11	3991632	9/17/2014 17:24	59.926388	10.634668	613	-0.0129	2390	0.09267	917	0.3821	1192	424.66	2588	25.4262
12	3991637	9/17/2014 17:25	59.926627	10.624988	598	-0.0141	1096	0.28845	953	0.5164	1188	427.34	2579	26.0796
13	3991642	9/17/2014 17:26	59.926947	10.616955	745	-0.0023	1719	0.19419	964	0.5575	1190	426	2637	21.8688
14	3991647	9/17/2014 17:27	59.923838	10.60999	785	0.0009	2683	0.04834	901	0.3224	1190	426	2622	22.9578
15	3991653	9/17/2014 17:28	59.920492	10.604107	751	-0.0018	2281	0.10916	893	0.2926	1194	423.32	2561	27.3864
16	3991658	9/17/2014 17:29	59.920862	10.600553	776	0.0002	2347	0.09918	852	0.25	1191	425.33	2707	16.7868
17	3991668	9/17/2014 17:31	59.916638	10.586405	727	-0.0037	2480	0.07905	905	0.3374	1186	428.68	2647	21.1428
18	3991673	9/17/2014 17:32	59.913842	10.579165	996	0.0179	2271	0.11067	887	0.2702	1115	476.25	2715	16.206
19	3991678	9/17/2014 17:33	59.913852	10.57018	856	0.0067	2792	0.03185	876	0.25	1193	423.99	2690	18.021
20	3991683	9/17/2014 17:34	59.91196	10.562822	674	-0.008	1254	0.26455	885	0.2627	1189	426.67	2599	24.6276
21	3991693	9/17/2014 17:36	59.910193	10.547133	811	0.003	2557	0.0674	863	0.25	9	999	2581	25.9344
22	3991698	9/17/2014 17:37	59.909275	10.539933	799	0.0021	2599	0.06105	874	0.25	1	999	2574	26.4426
23	3991708	9/17/2014 17:39	59.906913	10.524952	838	0.0052	2371	0.09554	873	0.25	0	999	2614	23.5386
24	3991723	9/17/2014 17:42	59.912832	10.504028	847	0.0059	2831	0.02595	854	0.25	5	999	2747	13.8828
25	3992240	9/17/2014 19:51	59.918495	10.492747	817	0.0035	1754	0.1889	823	0.25	0	999	2619	23.1756
26	3992245	9/17/2014 19:52	59.91887	10.493867	829	0.0045	2524	0.0724	908	0.3486	1	999	2644	21.3606
27	3992250	9/17/2014 19:53	59.918908	10.493918	834	0.0049	2517	0.07345	900	0.3187	1	999	2643	21.4332
28	3992255	9/17/2014 19:54	59.91891	10.49392	816	0.0034	2510	0.07451	902	0.3262	1	999	2660	20.199
29	3992260	9/17/2014 19:55	59.918005	10.494437	823	0.004	2388	0.09297	908	0.3486	1	999	2617	23.3208
30	3992270	9/17/2014 19:57	59.915553	10.488922	630	-0.0115	2937	0.00991	862	0.25	2	999	2355	42.342

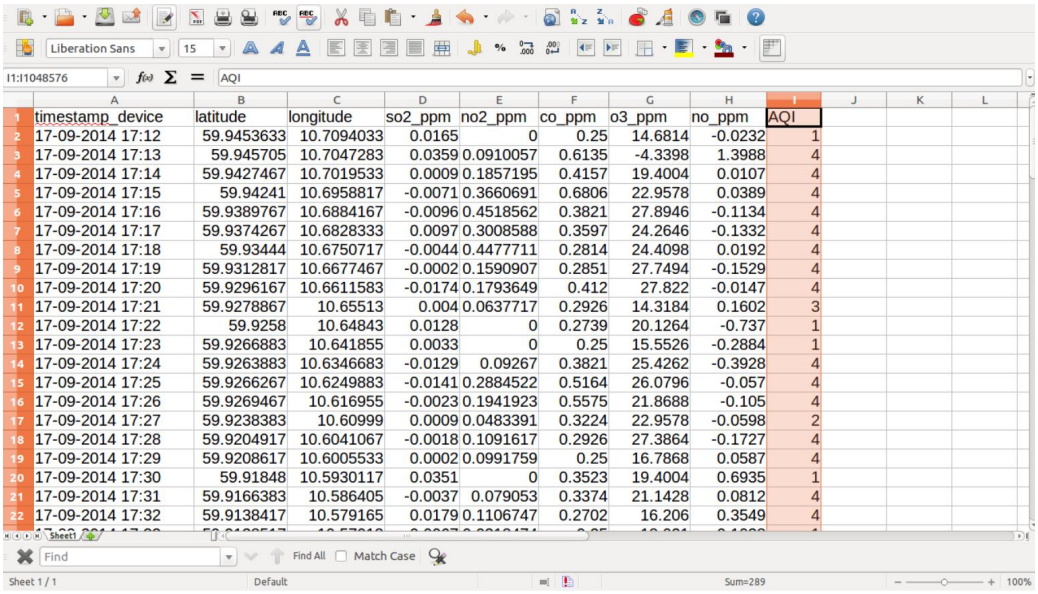
Figure 39: Data measured by a bike sensor for Citi-Sense-MOB use case, shown in tabular form

Based on this sensor data from a bike sensor, four test scenarios are defined. For each test, data is filtered and processed to generate final visualization through the *VoVis* framework.

1. *First test scenario* is to visualize AQI calculated from the data. AQI is an indicator of air quality based on air pollutants that have adverse effects on human health. In Citi-Sense-MOB, the AQI is used to display the data for the users. AQI can be calculated by the guidelines given by Norwegian luftkvalitet [1]. New dataset comprises of locations, pollutant gases from which AQI is calculated and AQI values. Figure 40 shows some pollutant gases and AQI calculated from the values of given gases.

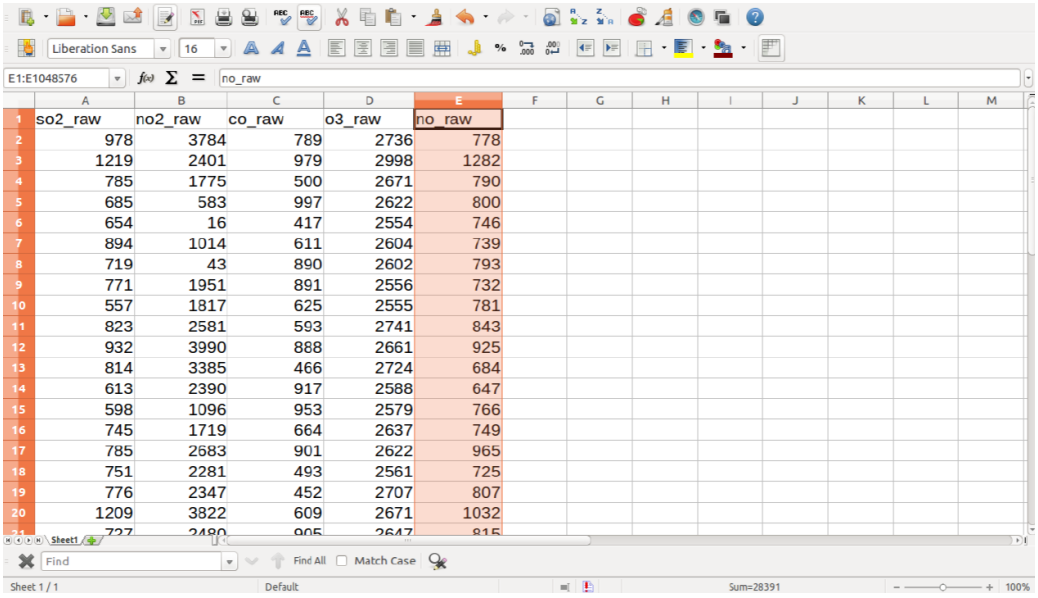
2. *Second Test scenario* focuses on individual air pollutant components like NO2, CO, etc. How the values of these air pollutants range over time, which type of visualization should be selected, the test is to answer the queries and to visualize these data correctly. To understand this scenario, the data is filtered into a subset of original data. Figure 41 shows different gases in raw form.

1 luftkvalitet : http://luftkvalitet.info/home.aspx



	timestamp_device	latitude	longitude	so2_ppm	no2_ppm	co_ppm	o3_ppm	no_ppm	AQI
2	17-09-2014 17:12	59.9453633	10.7094033	0.0165	0	0.25	14.6814	-0.0232	1
3	17-09-2014 17:13	59.945705	10.7047283	0.0359	0.0910057	0.6135	-4.3398	1.3988	4
4	17-09-2014 17:14	59.9427467	10.7019533	0.0009	0.1857195	0.4157	19.4004	0.0107	4
5	17-09-2014 17:15	59.94241	10.6958817	-0.0071	0.3660691	0.6806	22.9578	0.0389	4
6	17-09-2014 17:16	59.9389767	10.6884167	-0.0096	0.4518562	0.3821	27.8946	-0.1134	4
7	17-09-2014 17:17	59.9374267	10.6828333	0.0097	0.3008588	0.3597	24.2646	-0.1332	4
8	17-09-2014 17:18	59.93444	10.6750717	-0.0044	0.4477711	0.2814	24.4098	0.0192	4
9	17-09-2014 17:19	59.9312817	10.6677467	-0.0002	0.1590907	0.2851	27.7494	-0.1529	4
10	17-09-2014 17:20	59.9296167	10.6611583	-0.0174	0.1793649	0.412	27.822	-0.0147	4
11	17-09-2014 17:21	59.9278867	10.65513	0.004	0.0637717	0.2926	14.3184	0.1602	3
12	17-09-2014 17:22	59.9258	10.64843	0.0128	0	0.2739	20.1264	-0.737	1
13	17-09-2014 17:23	59.9266883	10.641855	0.0033	0	0.25	15.5526	-0.2884	1
14	17-09-2014 17:24	59.9263883	10.6346683	-0.0129	0.09267	0.3821	25.4262	-0.3928	4
15	17-09-2014 17:25	59.9266267	10.6249883	-0.0141	0.2884522	0.5164	26.0796	-0.057	4
16	17-09-2014 17:26	59.9269467	10.616955	-0.0023	0.1941923	0.5575	21.8688	-0.105	4
17	17-09-2014 17:27	59.9238383	10.60999	0.0009	0.0483391	0.3224	22.9578	-0.0598	2
18	17-09-2014 17:28	59.9204917	10.6041067	-0.0018	0.1091617	0.2926	27.3864	-0.1727	4
19	17-09-2014 17:29	59.9208617	10.6005533	0.0002	0.0991759	0.25	16.7868	0.0587	4
20	17-09-2014 17:30	59.91848	10.5930117	0.0351	0	0.3523	19.4004	0.6935	1
21	17-09-2014 17:31	59.9166383	10.586405	-0.0037	0.079053	0.3374	21.1428	0.0812	4
22	17-09-2014 17:32	59.9138417	10.579165	0.0179	0.1106747	0.2702	16.206	0.3549	4

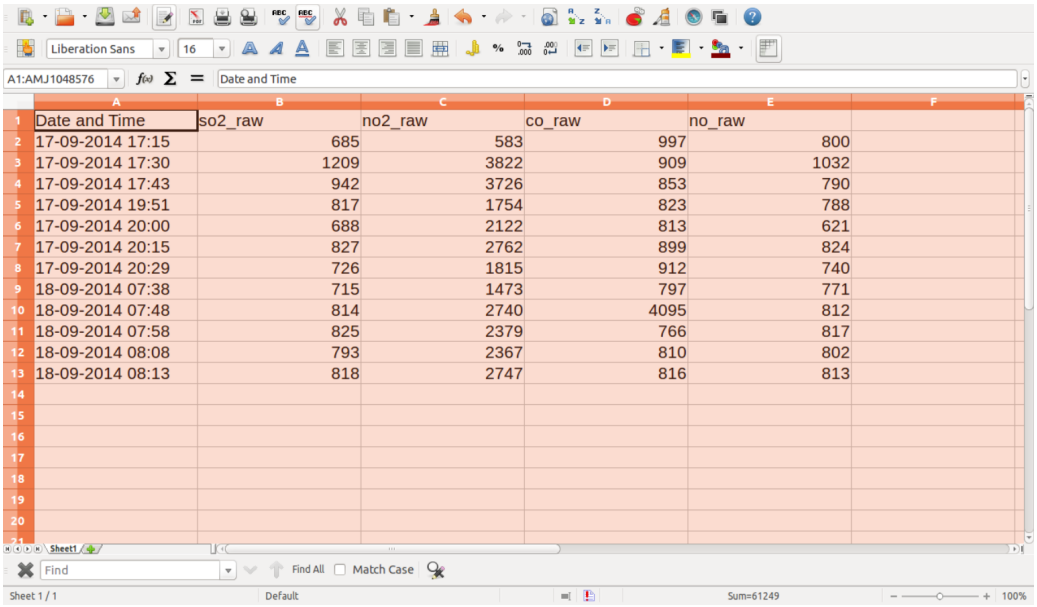
Figure 40: Filtered data with AQI calculated from pollutant gases shown in a tabular form



	so2_raw	no2_raw	co_raw	o3_raw	no_raw
2	978	3784	789	2736	778
3	1219	2401	979	2998	1282
4	785	1775	500	2671	790
5	685	583	997	2622	800
6	654	16	417	2554	746
7	894	1014	611	2604	739
8	719	43	890	2602	793
9	771	1951	891	2556	732
10	557	1817	625	2555	781
11	823	2581	593	2741	843
12	932	3990	888	2661	925
13	814	3385	466	2724	684
14	613	2390	917	2588	647
15	598	1096	953	2579	766
16	745	1719	664	2637	749
17	785	2683	901	2622	965
18	751	2281	493	2561	725
19	776	2347	452	2707	807
20	1209	3822	609	2671	1032

Figure 41: Filtered data with some air pollutant gases in tabular form

3. *Third Test scenario* focuses on comparisons of the air pollutant components over a period. Two features need to be visualized, first what are the values of pollutant gases at a certain period, and other is to compare few of the pollutant gases. Figure 42 shows some of the pollutant gases in raw form and date and time in CSV format.



	A	B	C	D	E	F
1	Date and Time	so2_raw	no2_raw	co_raw	no_raw	
2	17-09-2014 17:15	685	583	997	800	
3	17-09-2014 17:30	1209	3822	909	1032	
4	17-09-2014 17:43	942	3726	853	790	
5	17-09-2014 19:51	817	1754	823	788	
6	17-09-2014 20:00	688	2122	813	621	
7	17-09-2014 20:15	827	2762	899	824	
8	17-09-2014 20:29	726	1815	912	740	
9	18-09-2014 07:38	715	1473	797	771	
10	18-09-2014 07:48	814	2740	4095	812	
11	18-09-2014 07:58	825	2379	766	817	
12	18-09-2014 08:08	793	2367	810	802	
13	18-09-2014 08:13	818	2747	816	813	

Figure 42: Filtered data comprises of time and air pollutant gases in tabular form

4. *Fourth Test Scenario* is to show the visualization of sensor data through the mobile app, with cross-platform support.

Above mentioned are the four test scenario that will be tested in the *VoVis* framework.

8.2 RESULTS OF THE EXPERIMENT

The four test scenarios provides the data for visualization. Three types of visualization are created by the *VoVis* framework. Following are the steps for execution of the test by the *VoVis* framework.

- Upload the input file in CSV format.
- Select a type of visualization from the drop down menu. Once a type is selected information regarding the visual type is shown in the information box. This helps in mapping of data with the VisVo vocabulary.
- Drag and drop feature of the *VoVis* provides help for mapping of the data columns to the VisVo vocabulary and manually typing some input for labels as the name of the chart.
- Once mapping is done, and next is to process it by clicking the process button. Remapping is also possible by reset method that will reset the mapping.
- The processed data and the visualization is generated and can be viewed and accessed on the web page.

Figure 43 is the screenshot of the *VoVis* framework showing information about multi-bar-chart and the data file is uploaded.

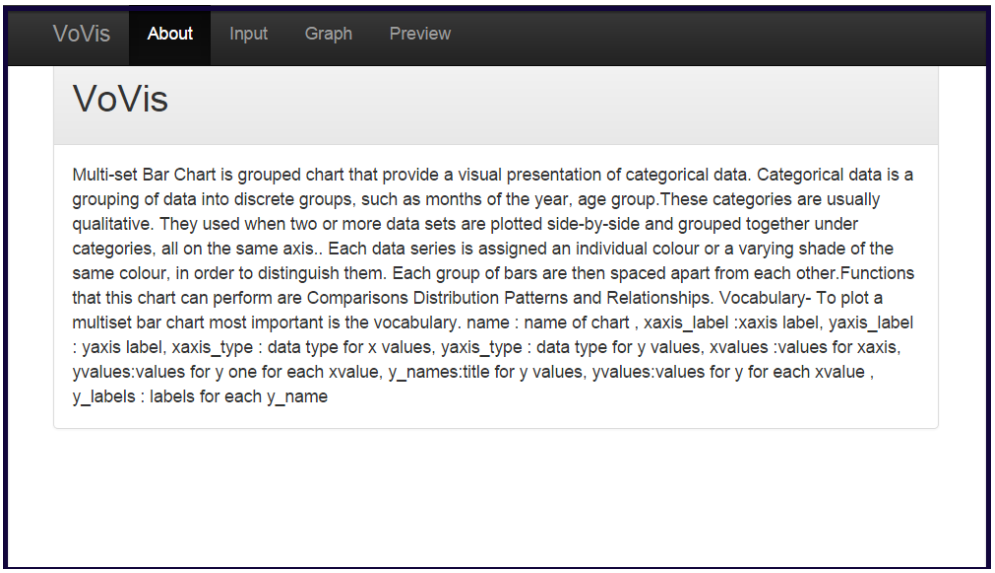


Figure 43: The *VoVis* framework shows the information of Multi-set-bar

8.2.1 *Result of Test Scenario 1*

The best way to visualize AQI calculated in test scenario is through maps, so it is a map visualization, with different color bubbles displayed over the map at different locations on a track. The latitude and the longitude are part of the input data so that it can be visualized through a map. The values of AQI ranges from 1 to 4, so four colors are used, each represents one value of AQI. This visualization helps to check, how is the air quality on a particular track for a given period. Figure 44 shows the result how the *VoVis* framework has visualized the sensor data.

8.2.2 *Result of Test Scenario 2*

The best way to visualize the second test scenario is through Box Chart. This chart describes statistically what is the highest, lowest and mean value for a pollutant gas in a day. By this chart, the range of each gas is easily visible and can show the fluctuations in any gas with different labels. X axis defines all the gases, and Y axis has all the values in raw for each gas.Figure 45 shows the visualization of sensor data as Box chart by the *VoVis* framework.

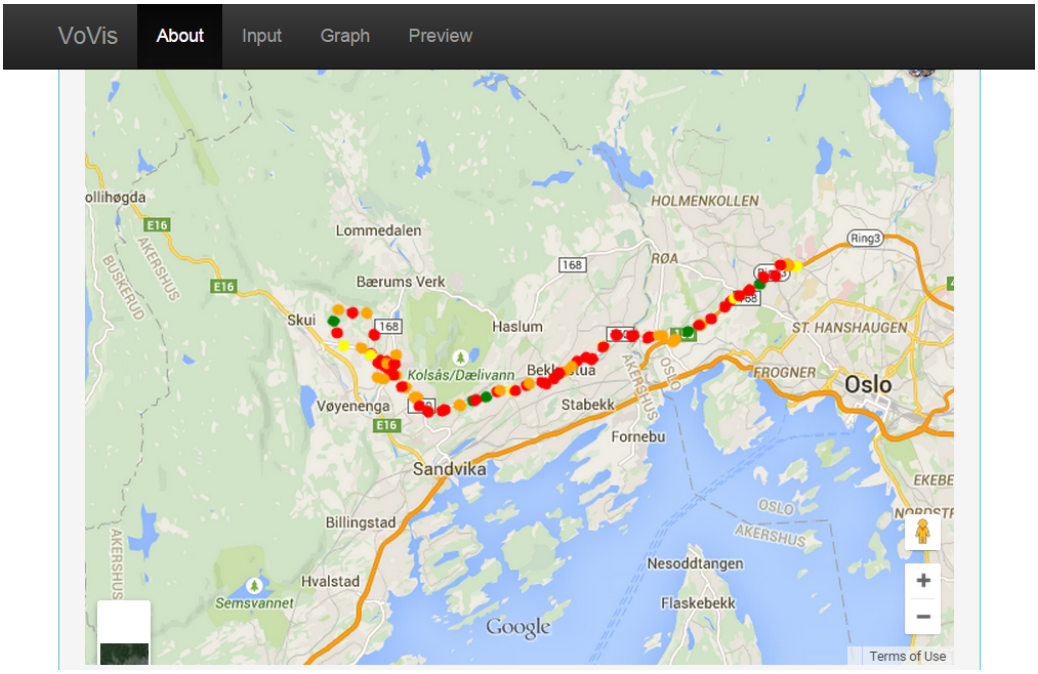


Figure 44: The *VoVis* framework Visualizing Sensor data as Map

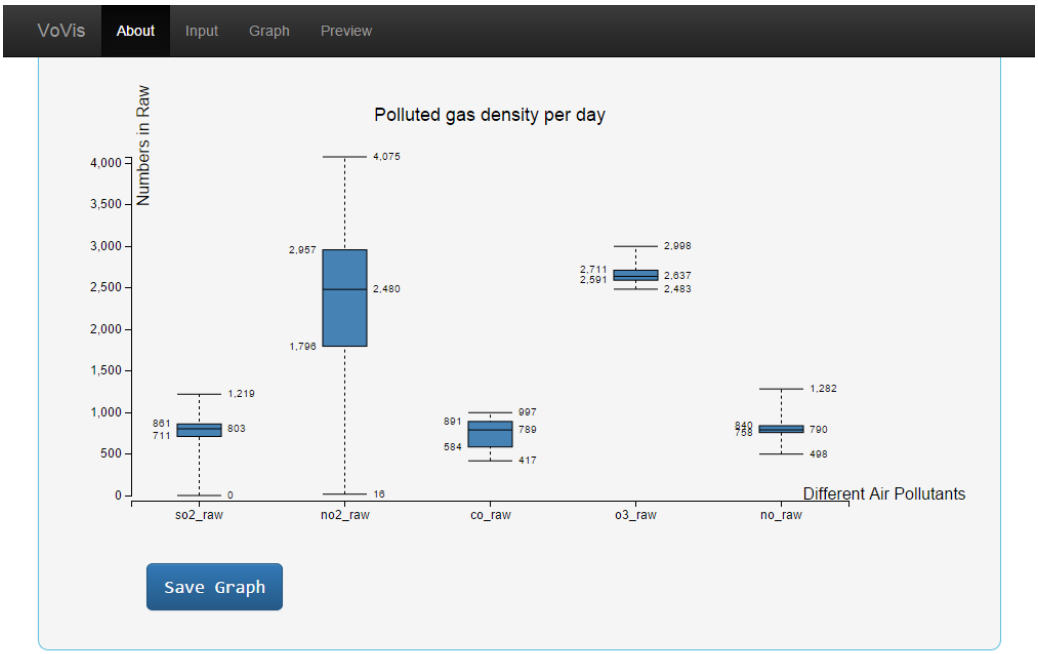


Figure 45: The *VoVis* framework Visualizing Sensor data as Box Chart

8.2.3 *Result of Test Scenario 3*

The best way to visualize the third test Scenario is through Multi-set Bar chart. As mentioned in the test scenario, the visualization should outline the value of each gas at the particular time, and also it should compare all gases.

The Visualization library of the *VoVis* framework suggests both Stacked Area chart and Multi-set Bar chart. The multi-set Bar chart is better than Stacked Area chart, as it is more human interactive. Figure 46 shows a comparison of four gases (No, Co, NO2, SO2) in raw form, over fifteen minutes time interval.

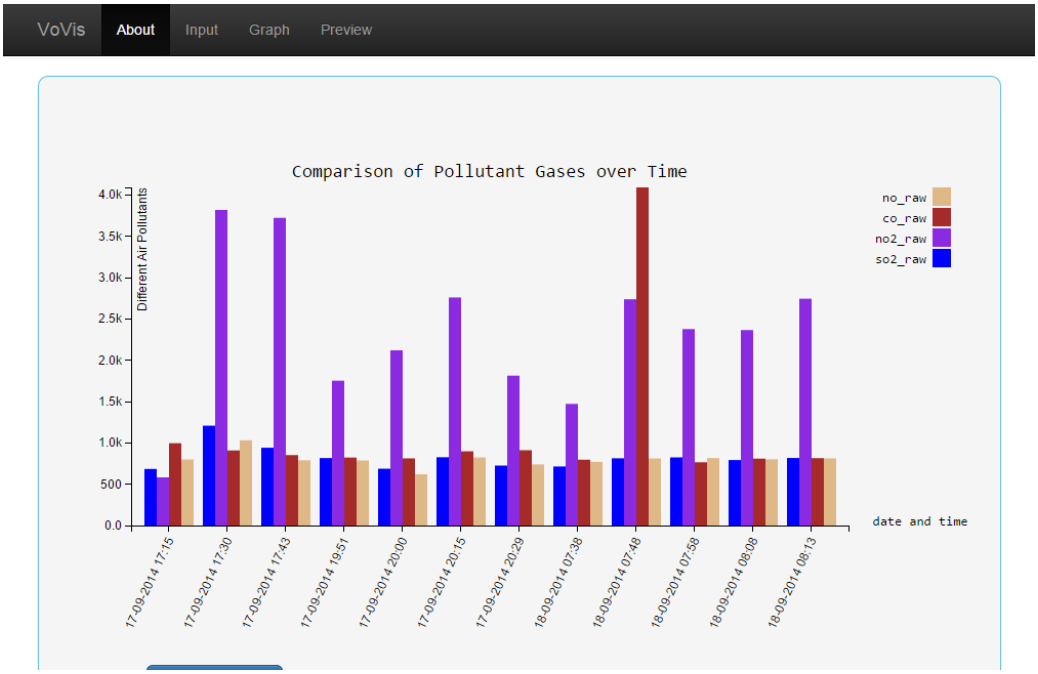


Figure 46: The *VoVis* framework Visualizing Sensor data as Multi-set Bar Chart

8.2.4 *Result of Test Scenario 4*

In order to execute the fourth Test Scenario, a mobile app version of the *VoVis* framework needs to develop with cross-device compatibility. The *VoVis* framework is a JavaScript application, so with the help of PhoneGap[2] framework, the *VoVis* mobile app is developed. With the help of PhoneGap framework, *VoVis* builds a mobile app for different platforms like iPhone/iPad, Android [3], etc. As a part of experiment only Android app is developed for *VoVis* framework. The *VoVis* framework is a JavaScript application, so with the help of PhoneGap[4] framework, the *VoVis* mobile app is developed [22] for the different platforms like iPhone/iPad, and Android[5]. As a part of experiment only an Android app is developed for the *VoVis* framework.

Thus, above are the results of the test scenarios. Based on the results of the experiment carried out using the *VoVis* Framework, the evaluation of *VoVis* is carried in next section.

2 Phonegap: http://phonegap.com/
3 Android: https://www.android.com/
4 Phonegap: http://phonegap.com/
5 Android: https://www.android.com/

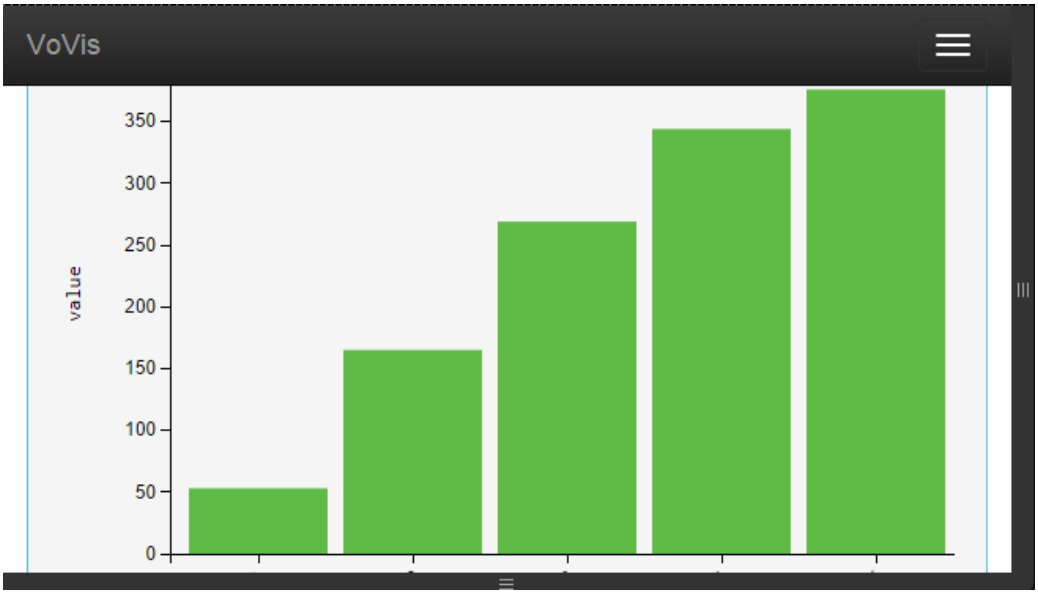


Figure 47: *VoVis* Framework as Mobile App shows a Bar Chart

8.3 VOVIS FRAMEWORK EVALUATION

An evaluation needs to be done, to determine whether the *VoVis* framework fulfills the requirements given in Section 4.2 Requirements for a web visualization Framework, or some parts that need to be improved in the *VoVis* framework. To be able to compare it with the previously evaluated frameworks Weave, Tableau Public, Visualize Free and Many Eyes, the evaluation schema is the same as the one which has been used for their evaluation in Section 5.1.6 Evaluation of visualization Frameworks: *VoVis* is classified as ∅, + or ++, depending on if the requirement is not fulfilled at all, if it needs some improvements or if it is fulfilled.

Visualization Requirements	**VoVis**
R1 - Possibility to Support all Popular Types of Visualization	+
R2 - Third Party Javascript Visualization Libraries	++
R3 - Local Access to Charts	++
R4 - Data Mapping using Generic Visual Vocabulary	++
Sum	**7**

Table 8: Evaluation of the VoVis framework for Visualization Requirements

Table 8 includes the evaluation for the first four requirements, related to the visualization. The *VoVis* visualization library supports thirty types of visualization, but the functional prototype of the *VoVis* framework currently visualizes only five types of visualization. Concept and design of *VoVis* supports a wide range of visualizations, but prototype only supports a few. The framework can be extended to support for more visualizations and therefore

evaluated as + for this requirement. The *VoVis* framework is standard and flexible, can use any JavaScript third party library for the visualization. Currently, it is using both D3 and Google Charts. *VoVis* is evaluated ++ for the requirement to support third party JavaScript Libraries. The *VoVis* framework provides full access to both data and visual image. Visualization can be saved as a picture for local access. The *VoVis* framework has a vocabulary component that provides the vocabulary for mapping of data. This component helps to map data easily, friendly to understand and process, so increases the efficiency of the framework. The user can have maximum benefits as there is no need to have pre-knowledge about the visualization types or process, very convenient to generate any visualization. Only *VoVis* has this new component; so is evaluated as ++ for this requirement of vocabulary mapping.

Data Service Requirements	**VoVis**
R5 - RDF and CSV Data Format as Input	+
R6 - Data Privacy	++
R7 - Easy to Upload Data	++
R8 - Access to Processed Data in Standard Format	++
R9 - Processed Data comprise Visualization Information	++
Sum	**9**

Table 9: Evaluation of the VoVis for Data Service Requirements

Table 9 includes the evaluation of the *Data Service Requirements*. When it comes to data service requirements, the *VoVis* prototype supports only CSV format. The reason is that the RDF vocabulary for visualization types is at experiment level. Only for proof of concept, the RDF visualization vocabulary ontology(VisVo) is designed for few chart types. A lot of things need to be done in the RDF field for visualization. Also, mapping of this vocabulary to the RDF data needs to be implemented. Because of time constraint this thesis could not focus much on this part, but it has a design that is shown in the design of *VoVis* framework in Section 6.3.2. For the data requirements(R5) it is evaluated as +. As the main focus of the *VoVis* framework is vocabulary mapping and visualizing through JavaScript libraries. Filtering of data is not implemented, so difficult to manage huge files, but it is very easy component to add as many JavaScript libraries provide functions for filtering of data according to the requirements. Crossfilter.js [6] is a JavaScript library that provides "Fast Multidimensional Filtering for Coordinated Views". Crossfilter along with the D3 can visualize even with datasets containing a million or more records. *VoVis* provides full access to processed data, and processed data comprises of information for the type of visualization. The processed data can be accessed by developers and with visualization information, developers can generate the visualizations in the more custom way or can use these data as an input for further research.

6 Crossfilter a JavaScript library : http://square.github.io/crossfilter/

Web Framework Requirements	**VoVis**
R10 - Programming in JavaScript	++
R11 - Offline Mode	+
R12 - Cross Device Compatibility	++
R13 - Open Source	++
R14 - Cross Browser Compatibility (with no Flash Plugin)	++
R15 - Framework should be User Friendly	++
Sum	**11**

Table 10: Evaluation of the VoVis for Web Framework Requirements

Table 10 includes the evaluation for the *Web Framework Requirements*. The *VoVis* framework is implemented as a web application, so it must fulfill all the web requirements. For requirement R10- Programming in JavaScript, *VoVis* is evaluated to ++, as *VoVis* is a purely JavaScript application that supports all platforms and browsers. Using a third party framework like PhoneGap, it is easy to develop mobile apps for different platforms only if the framework is in JavaScript. *VoVis* is an open source framework means everyone can access and contribute new and creative ideas. *VoVis* will have a big supportive forum where experienced developers can share their views. It can also easily be integrated to other open source frameworks as a visualization component. The *VoVis* framework can work offline there is no need to connect to internet always, but as it is using Google Charts as JavaScript visualization library for plotting Maps, it needs to be online. If only the D3 is used as a visualization tool, *VoVis* can work offline. For the requirement R11 - Offline Mode, *VoVis* is evaluated as +. The *VoVis* framework does not require any plugin to be installed in the browser; it supports all modern browsers, so it is evaluated ++ for this requirement.

Usability Requirements	**VoVis**
R16 - Customizability	++
R17 - Performance	+
R18 - Design and Interactivity	++
Sum	5

Table 11: Evaluation of the VoVis for Usability Reqirements

Table 11 includes the evaluation for the *Usability Requirements*. The first two requirements Customizability and Design and Interactivity are fulfilled by the *VoVis* framework, only because it is open source and uses third party JavaScript libraries. It provides better design and a lot of custom options to add on the visualizations by using standard tools. If developers are not satisfied, the visualization tools can be replaced by other standard tools. As the

VoVis framework is implemented as a prototype, there is a lot of scope for further development, which will enhance *VoVis* performance. For the performance requirement, the *VoVis* framework is evaluated as +.

Evaluation	VoVis	Tableau Public	Weave	Visual Free	Many Eyes
Data Service Requirements	9	1	4	2	2
Visualization Requirements	7	7	7	6	4
Web Framework Requirements	11	5	6	4	2
Usability Requirements	5	4	5	3	3
Sum	**32**	**17**	**22**	**15**	**11**

Table 12: Overall Evaluation of the VoVis framework

Table 12 outlines the *overall evaluation* of the *VoVis* framework for all requirements. *VoVis* is evaluated and granted 32 points as total out of 36 points that means *VoVis* fulfills **88%** of total requirements. This is the highest among all other four frameworks, evaluated earlier (look at Table 6 for an evaluation). The fulfillment rate of all other existing frameworks evaluated are **Many Eyes (30.55%)**, **Visualize Free (41.66%)**, **Tableau Public (47.22%)**and **Weave (61.11%)**. Thus, *VoVis* is better framework than existing frameworks that are assessed.

The *VoVis* framework meets all the requirements, satisfy the needs and also fills the research gaps. With above evaluation, it can be concluded that the prototype looks promising, it can later on be elaborated to a complete product. A functional prototype of the *VoVis* framework needs to be tested to determine whether all requirements listed in Chapter 4 are fulfilled. This will also address the last research task of the thesis and discusses the experiment performed using the *VoVis* framework.

9

CONTRIBUTIONS AND FUTURE WORK

This chapter provides a summary of the thesis as a whole and discusses its contributions to the field of data visualization. A section on future work is meant to explain potential avenues for further research.

9.1 MEETING THE RESEARCH TASKS

This section outlines how the *VoVis* framework meets all the research tasks that were defined in Section 1.5. All the research tasks are discussed and fulfilled as part of the thesis. The part one (Background Study) of the thesis discusses first four research tasks in different chapters. The last two tasks are fulfilled in the second and the third part respectively.

- The **first research task** was to provide a conceptual understanding of the visualization concepts and also to understand the data collection and analysis concepts. Chapter 2 outlines the concept of data and visualization, types of data, and different types of data visualizations. It describes that both data and visualizations are dynamic with the environment. There are various technologies available to process and visualize data. Detail study of data visualizations shows that the data volume is enormous and appropriate visualization techniques are required to capture them through visualizations. This chapter provides an overall picture why data visualization is important.
- The **second research task** is discussed in Chapter 3 which explains how to design a visualization effectively, different types of visualization frameworks are discussed, the importance of web visualization framework. The components of visualization techniques are discussed. These components helped to define the set of requirements for the *VoVis* framework that aims to focus on data visualization through a web application.
- The **third research task** was the identification of requirements and is discussed in Chapter 4. The pilot projects CITI-SENSE-MOB and DaPaaS served as examples to derive the subsequent requirements. Chapter 4 briefly described the use cases that are defined for the pilot projects. PLUQI is a use case from the DaPaaS project that provides a customizable index model and a mobile/web application accessed by the end users on the web and via smartphones. Citi-Sense-MOB also has AQI

index based mobile app as the use case. The set of visualization requirements from both the use cases formed the base to define requirements for the *VoVis* framework. Requirements for the *VoVis* framework are divided into three major sections according to the components defined in Chapter 3. Visualization requirements defined in Section 4.2 are mainly about the types of visualizations a framework should support, about mapping of data with the particular type of visualization and so on. A visualization tool is required for visualizing the data,which should be efficient. Many standard JavaScript visualization libraries can deliver good visualization as output.The framework can use these libraries. Data Requirements defined in Section 4.3 are concerned with data, like different types of data formats, access to data, uploading of data and data privacy. The main focus is on the processed data, the information it contained and access to the processed data. Web framework requirements defined in Section 4.4, focuses on the web related requirements such as a framework is open source, can work offline, cross browser and supports devices etc. Usability requirements defined in Section 4.5 discusses on factors like Customizability, Design and Interactivity of the framework and performance of the visualization framework. All the set of requirements are defined for a visualization framework.

- The **fourth research task** was to focus on review and evaluation of the existing frameworks and the visualization tools.These are discussed and fulfilled in Chapter 5, which first reviews the different existing web-based visualization frameworks. Four frameworks are selected for the evaluation based on the set of requirements. Many Eyes, Tableau Public, Visualize Free and Weave, all are different from each other in terms of tools, technologies and methods they use. All are evaluated for the requirements defined and by overall evaluation the frameworks are ranked. Weave framework found better than other frameworks but not the best to fulfill all the requirements. It is concluded that none of the frameworks, covers all the aspects of visualization, also no one focuses on mapping of the data with types of visualization(can be charts and maps). This limitation motivates for the development of a new framework and a vocabulary component. More than 30 JavaScript libraries are reviewed and four of them are discussed as a prominent tool for the *VoVis* framework. D3, Google Charts, jqPlot and Flot are evaluated on the basis of requirements for JavaScript libraries. D3 satisfied most of the requirements, next is Google charts. Both of them are selected as tools for the *VoVis* framework.

- The **fifth research task** was concerned with the designing of vocabulary library and implementation of the *VoVis* framework. The task is fulfilled in Chapter 6 and Chapter 7. First the design and concept of the *VoVis* framework and vocabulary component is discussed. Vocabulary component is designed by developing vocabulary library and visualization library. An RDF vocabulary is written as an ontology that

provides a proof of concept in RDF vocabulary research field. Visualization vocabulary defines the properties for each type of visualization. After the design, a full functional prototype of the *VoVis* framework is implemented as a web application that follows Client-side Scripting in JavaScript. Different components are discussed in detail.

- The **sixth and last research task** was the evaluation of the *VoVis* framework, which is performed in Chapter 8. To validate the hypothesis stated in Section 1.3 and to verify that the *VoVis* framework fulfilled all the requirements defined, the *VoVis* framework is evaluated with an experiment. The Experiment is performed based on four test scenarios. Each test scenario has an input data and type of visualization to generate, four tests are performed for each scenario. The results of the tests are reported and based on it, the *VoVis* framework is evaluated. The *VoVis* framework is evaluated for each set of requirements and found better than the other existing frameworks.

9.2 VALIDATION OF THE HYPOTHESIS

Implementation and evaluation of the artefact(*VoVis*) are done to validate or refute the hypothesis and also to close the gaps in the field of data visualization through web.

The hypothesis of this thesis is:

"*It is possible to have a generic visualization framework, that meets the need to support a wide range of charts with improved data mapping technique for cross-platform according to the identified requirements for data visualization, data service, web framework and usability.*".

The *VoVis* framework with vocabulary component has been tested. For this, four test scenarios from one of the pilot cases (Citi-Sense-MOB) have been used, described in Chapter 8. The evaluation shows that the *VoVis* framework fulfills almost all the tasks contained in the web framework (91%), data service (90%) and visualization requirements (87%). In terms of data format support and charts support, some extensions need to be added. The overall evaluation of the *VoVis* framework results in a fulfillment rate of 88% of the identified requirements. There are few improvements further discussed in Section 9.4, which can be easily implemented. The *VoVis* framework prototype represents a good basis for research and development in the field of data visualization, compared to the four other data visualization frameworks evaluated in this thesis in Section 5.1.6. The results of evaluation of existing frameworks shows the fulfillment rate of requirements by Many Eyes (30.55%), Visualize Free (41.66%), Tableau Public (47.22%)and Weave (61.11%). The Weave framework is better than the others but needs improvements in many aspects in terms of the requirements defined for a web-based visualization framework.

In terms of the pilot projects introduced in Chapter 1, the *VoVis* framework can be used for Citi-Sense-MOB, as shown with the four test scenarios in Chapter 8. *VoVis* can be used to visualize AQI calculated from the sensor

data of the Citi-Sense-MOB project, also individual air pollutant components like NO2, CO can be visualized as different types of visualization. The *VoVis* framework supports cross-platform and has a mobile app to visualize the sensor data over different platform smartphones, that is a core requirement for the Citi-Sense-MOB project. As an overall conclusion: *VoVis* meets almost all the identified requirements and is applicable to Citi-Sense-MOB project. In order to use the *VoVis* framework within the DaPaaS project, additional RDF support is necessary, but this can easily be added.

The problems identified in Chapter 1 and the research gaps clearly states that there is a need of improvement in data mapping techniques and a generic framework that supports a wide range of visualizations with user-friendly interface. The *VoVis* framework addressed most of the problems defined and closed the research gaps defined earlier. The *VoVis* framework is proved to be an improved solution for data visualization compared to other frameworks assessed. Thus, the stated hypothesis of this thesis has been validated.

9.3 THESIS CONTRIBUTIONS

The purpose of this thesis is to address the problems identified in the Section 1.1 and close the research gaps. To fulfill the purpose of thesis a new generic visualization vocabulary library is designed and a artefact i.e web visualization(*VoVis*) framework using vocabulary is implemented. Based on the result of the experiment and evaluation of the *VoVis* framework, it can be concluded that the *VoVis* framework with its vocabulary library concept has mostly fulfilled the set of requirements.

All the six research tasks of this thesis have been discussed and fulfilled by the *VoVis* framework along with the vocabulary component and the research gaps are closed. The contributions made by the *VoVis* framework are as follows.

- VoVis provides a visualization library that has information about thirty-one types of visualizations. Each chart or graph type has some unique features that differentiate it from others. This library not only explains about type of visualization but also the functions that they can perform on data like comparisons, relationships etc.
- VoVis provides a vocabulary library which is a new component in field of web visualization. The Vocabulary is designed and written both in text form and in RDF ontology. This new component helps in mapping of the data to get the final visual results. The processed data has both data and information of visualization type mapped together by the vocabulary mapping process.
- VoVis provides a means to support most popular visualization types and can support different data types with extension of the artefact. The artefact is validated and satisfied all the needs, so it can be extended.
- The VoVis framework can be integrated into other development processes or bigger framework where a visualization component is required

to visualize the results. Both use cases from the pilot projects can use this framework as a tool or as service.

- VoVis presented a way to use the standard JavaScript visualization libraries as tools in a visualization framework. Any standard library can be implemented as the tool that makes it a reusable library.

VoVis provides a good basis for further work. Some ideas and recommendations for further improvements are outlined in the following section.

9.4 FUTURE WORK

The *VoVis* framework is a general vocabulary based visualization web framework. *VoVis* can upload data in some format, can store it in a database, process the data and map it to particular visualization type, and finally visualize the data through the visualization tools. There is scope for few improvements and also extensions are possible. A set of improvements for the *VoVis* framework has been identified that can be part of future research. They are described below:

9.4.1 *Extension of Data and Visualization Types*

At the time of evaluation of the *VoVis* framework, it has been found that the *VoVis* framework prototype currently supports very few visualization types, but at the same time visualization library from vocabulary component have nearly thirty types of visualizations, and each has its own vocabulary. It means it is possible to extend this feature to support most popular types of visualization. The support for data format is currently CSV, as per design it should support RDF format. The reason for not supporting RDF currently is that, the RDF visualization vocabulary(VisVo) is in experiment phase as mapping process needs to be designed for mapping of RDF data to vocabulary. So this feature is dependent on other feature(RDF vocabulary) but can be extended to support RDF formats.

9.4.2 *RDF Vocabulary*

Currently the RDF Visualization vocabulary(Visvo) is designed for single Bar, line, area charts, Bubble charts and scatter-plot charts. This ontology can be extended to support all visualization types that are covered in the *VoVis* framework visualization library. The important feature to extend in future is RDF mapping process(mapping RDF data to RDF vocabulary(Visvo)). Once this process will be designed, the *VoVis* framework can support the RDF format.

9.4.3 *Extension of* VoVis

The *VoVis* framework can be extended to a software application in different forms and for both end users and developers. The *VoVis* framework is currently a client-side MVC web application and also available as an Android app (Prototype).

The *VoVis* framework can provide services in following extended version.

- **SaaS**: The *VoVis* can provide software as a service to the end users for data visualization. A cloud based solution where user can
  1. *Upload and Save* data in various formats - RDF, CSV, and JSON.
  2. *Decorate the data* as per Vocabulary.
  3. *Extend data* vocabulary.
  4. *Select different visualization* Engines Google Charts, D3, or any other.
  5. Visualize data in various types of visualization.
  6. Export the Graph/Maps.
- **Mobile App**: The *VoVis* can be installed as an app in Smartphones and Tablets. It can provide a handy interface to visualize and manage data in a quick way. If integrated with a cloud-based desktop solution, the app can be used to visualize the archive data with new parameters and visualization type.
- **Webservice APIs**: The *VoVis* can expose its APIs over HTTP for Developers. Developers can easily integrate the data processing and visualization features of *VoVis* into their existing software, website, and mobile apps. Developer can use this service to *decorate data* and to get *visualized data in image format*.
- **Framework Extension**: There are plenty of popular application frameworks in use. Example Wordpress(PHP), AngularJS(HTML5/JS), etc. The *VoVis* services can be written as these application framework extensions/modules to support the smooth integration of *VoVis* services in existing frameworks. Some popular modules/extensions are as follows.
  1. Angular Module
  2. Wordpress Plugin
  3. Node.js module
  4. JavaScript Library
  5. Java based API's
  6. Android API's
  7. iOS API's

Thus, the presented framework represents a good basis for further research and development. A set of improvements for the *VoVis* framework have been identified to drive future research. The *VoVis* framework itself or component of it like RDF vocabulary can be extended and improved, and better solutions can be identified by future research.

Part IV

APPENDICES

A

THE VOVIS VOCABULARY

This appendix contains the vocabularies for 30 types of visualization. Each type of visualization has the vocabulary, JSON structure and the functions it can demonstrate.

Appendix A -The VoVis Vocabulary

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Area Graph	name , xaxis_label , yaxis_label , xaxis_type, yaxis_type, xvalues, ,yvalues	name : name of chart , xaxis_label :xaxis label, yaxis_label : yaxis label, xvalues :values for xaxis, yvalues:values for y one for each xvalue e.g. ['Year', 'Sales',], ['2013', 1000], ['2014', 1170], ['2015', 660] prices over years. name:area chart ,xaxis_label:years , yaxis_label:prices over years , xvalues :2013,2014..., yvalues:1000,1170...	{ "values": [{"x":,"y":}, {"x":,"y":},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : }	Relationships, Patterns, Data over time
Bar Chart	name , xaxis_label , yaxis_label , xaxis_type, yaxis_type, xvalues,yvalues	name : name of chart , xaxis_label :xaxis label, yaxis_label: yaxis label, xvalues :values for xaxis, ,yvalues:values for y one for each xvalue e.g ['Year', 'Sales',], ['2013', 1000], ['2014', 1170], ['2015', 660] prices over years. name: bar chart ,xaxis_label:years ,yaxis_label:prices over years ,xvalues :2013,2014...,yvalues:1000,1170...	{ "values": [{"x":,"y":}, {"x":,"y":},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : }	Comparisons, Relationships, Patterns
Box and Whisker Plot	name ,xaxis_label, yaxis_label, xaxis_type, xvalues, x_names	name:name of chart, xaxis_label:x axis label, yaxis_label : yaxis label, xvalues : values of x E.g A candlestick/box chart is used to show an opening and closing values of stock every day in a week e.g ['Mon', 'Tue','Wed'], [20,30,40], [23, 25,45], [35, 50,55] name:Box chart showing stock values, xaxis_label:days of week, yaxis_label:stock values in number, x_names:Mon,Tues, xvalues:20,23,35	{ "values": [{"x":[ ]}, {"x":[ ]}] ,"name" :"box chart" ,"xaxis_label" : "","yaxis_label" : "", "xaxis_type" : " " , "x_names" : [ ] }	Distribution, Range, Comparisons, Patterns

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Bubble Chart	name , xaxis_label, xaxis_type , yaxis_label , yaxis_type , color_label , color_type , radius_label , radius_type, xvalues, yvalues, color_values, radius_values	name: name of chart, xaxis_label: xaxis label, yaxis_label:y axis label, xvalues: x values, yvalues: yvalues for each x values, color_label: label which different colors will represent, colora_values:values, radius_label:label which is shown by different circle radius, radius_values:values e.g. [Life Expectancy Fertility Rate Region, Population], [80.66, 1.67, 'North America', 33739900], [79.84, 1.36, 'Europe', 81902307], [78.6, 1.84, 'Europe', 5523095], [72.73, 2.78, 'Middle East', 79716203], [80.05, 2, 'Europe', 61801570], name: bubble chart, xaxis_label:life expectancy, yaxis_label:fertility rate, xvalues: 80.66 79.84.., yvalues: 1.67 1.36..., color_label:region, color_values: north America, radius_label:population, radius_values:33739900...	{ "values": [{"x":,"y": , "color" : , "radius" : }, {"x":,"y": , "color" : , "radius" :},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : , "color_label" : , "color_type" : , "radius_label" : , "radius_type": }	Comparisons, Data over time Distribution, Patterns, Proportions, Relationships
Histogram	name, xaxis_label, xaxis_type, xvalues, x_labels	name: name of chart, xaxis_label: name of xaxis, xvalues :x values , x_labels : labels for xvalues e.g. ['Dinosaur', 'Length'], ['Acrocanthosaurus (top-spined lizard)', 12.2], ['Albertosaurus (Alberta lizard)', 9.1], ['Allosaurus (other lizard)', 12.2] name: Histogram chart, xaxis_label: Length of dinosaures in meters, x_labels:['Acrocanthosaurus,'Albertosaurus...] , xvalues: 12.2,9.1,12.2...	{ "values": [{"x": }, {"x": },] , "name" : , "xaxis_label" : , "xaxis_type" : , "x_labels" : [] }	Comparisons, Data over Time, Distribution, Patterns, Probability, Range
Line Graph	name , xaxis_label , yaxis_label , xaxis_type, yaxis_type, xvalues, ,yvalues	name : name of chart , xaxis_label :xaxis label, yaxis_label : yaxis label, xvalues :values for xaxis, yvalues:values for y one for each xvalue e.g. ['Year', 'Sales',], ['2013', 1000], ['2014', 1170], ['2015', 660] prices over years. name:line chart ,xaxis_label:years , yaxis_label:prices over years , xvalues :2013,2014..., yvalues:1000,1170...	{ "values": [{"x":,"y":}, {"x":,"y":},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : }	Patterns, Data over time

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Multi-set Bar or line Chart	name , xaxis_label , yaxis_label , xaxis_type, yaxis_type, xvalues, yvalues, y_names, y_labels	name : name of chart , xaxis_label :xaxis label, yaxis_label: yaxis label, xvalues :values for xaxis, y_names:title for y values, yvalues:values for y for each xvalue , y_labels : labels for each y_name e.g. ['Year', 'Sales', 'Expenses'], ['2013', 1000, 400], ['2014', 1170,600], ['2015', 660,980] performance over years. name: multi line/bar/area chart , xaxis_label:years , yaxis_label: performance over years , xvalues :2013,2014..., y_name:[sales, expenses], yvalues:[1000,400],[1170,600], y_labels:[y1,y2]	{ "values": [{"x":,"y":[] }, {"x":,"y":[]},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : , "y_names" : [] ,"y_labels" : [] }	Comparisons, Distribution, Patterns, Relationships
Population Pyramid	name,xaxis_label,yaxis_label,xvalues,y1name,y1values,y2name,y2 values	name : name of chart , xaxis_label :xaxis label, yaxis_label: yaxis label, xvalues :values for xaxis, y1name:label for y1 values,y1values:valuess for y1 one for each xvalues , y2name:label for y2 values,y2values:valuess for y2 one for each xvalues e.g. population chart for year 2000 [Age Male Female] [0 147899 128975] [5 135678 134567] [10 122234 67899] name:population chart for year 200,xaxis_label:ages,yaxis_label:population ,xvalues:0,5,10..,y1name:male,y1values:147899,135678...,y2name:female,y2 values:128975,134567...	{ "values": [{"x":,"y1": ,"y2": }, {"x":,"y1": ,"y2":},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : , "y1_name" : , "y2_name" : }	Comparisons, Distribution, Patterns
Radial Bar Chart	polar coordinates name, radient_label, dimension_label, radient_values, d_labels, d_values	name:name of chart radient_label:name for r axis dimension_label:dimension name radient_values:values for r d_labels:labels for dimensions d_values:values for dimensions e.g. radial chart showing temperatures for different years for all months [Months 2000 2001] [Jan 5 4] [Feb 7 5] [Mar 10 6] name:radial bar chart , radient_label:Months, dimension_label:temperatures, radient_values:Jan,Feb,Mar, d_labels:[2000,2001] d_values:[5,4],[7,5].,	{ "values": [{"r":,"d":[]}, {"r":,"d":[]},] , "name" : , "radient_label" : , "r_type" : , "diemension_name" : , "dimension type" : , "d_labels":[] }	Comparisons, Relationships

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Scatterplot	name,xaxis,yaxis,xvalues,yvalues,color-label, color-values	name: name of chart, xaxis: xaxis label,yaxis:y axis label, xvalues: x valuess,yvalues: yvalues for each x values, color-label: label which different colors will represent,color-values:values e.g. scatterplot showing sepals and petals of various iris flowers [sepalLength, sepalWidth ,species] [5.1 3.5 setosa] [4.9 3.0 virginica] [4.7 3.2 versicolor] [4.6 3.1 setosa] name: scatterplot showing sepals and petals of various iris flowers,xaxis:sepal length in cm,yaxis:sepal width in cm,xvalues:5.1,4.9...,yvalues:3.5,3.0...,color-label:species,color-values:setosa,virginica...	{ "values": [{"x":,"y": , "color" : }, {"x":,"y": , "color" : },] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : , "color_label" : , "color_type" : }	Patterns, Relationships
Span Chart/diff/high-low	name ,xaxis ,yaxis ,xvalues,yname,y1values,y2values	name:name of chart ,xaxis:x axis label ,yaxis:y axis label ,xvalues:different values(labels) for x axis,yname:same label for y1 and y2 ,y1values: values for y1 one for each x values,y2values: values for y2 one for each x values e.g. Popularity of different people ['Name', 'Popularity','Popularity'], ['Cesar', 250,350], ['Rachel', 4200,8000], ['Patrick', 2900,4000], ['Eric', 8200,9000] name:Span/diff chart ,xaxis:Names ,yaxis:Popularity ,xvalues:Cesar,Rachel...label ,yname:popularity ,y1values: 250,4200,2900...,y2values: 350,8000,4000...	{ "values": [{"x":,"y1": ,"y2": }, {"x":,"y1": ,"y2":},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : , "y1_name" : , "y2_name" : }	Comparisons, Range
Stacked Area Graph	name , xaxis_label , yaxis_label , xaxis_type, yaxis_type, xvalues, yvalues, y_names, y_labels	name : name of chart , xaxis_label :xaxis label, yaxis_label: yaxis label, xvalues :values for xaxis, y_names:title for y values, yvalues:values for y for each xvalue , y_labels : labels for each y_name date IE Chrome Firefox Safari Opera 11-Oct-13 41.62 22.36 25.58 9.13 1.22 11-Oct-14 41.95 22.15 25.78 8.79 1.25 11-Oct-15 37.64 24.77 25.96 10.16 1.39 11-Oct-16 37.27 24.65 25.98 10.59 1.44 name:stacked area chart,x axis:Dates,y axis:Market share of browsers,xvalues:11-Oct-13,11-Oct-14...,y1name:IE,y1values:41.62,41.95,...,y2name:Chrome,y2values:22.36,22.15....	{ "values": [{"x":,"y":[] }, {"x":,"y":[]},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : , "y_names" : [] ,"y_labels" : [] }	Comparisons, Data over time, Patterns, Relationships
Stacked Bar Graph	name , xaxis_label , yaxis_label , xaxis_type, yaxis_type, xvalues, yvalues, y_names, y_labels	name : name of chart , xaxis_label :xaxis label, yaxis_label: yaxis label, xvalues :values for xaxis, y_names:title for y values, yvalues:values for y for each xvalue , y_labels : labels for each y_name State,Under 5 Years,5 to 13 Years,14 to 17 Years AL, 310504, 552339, 259034 AK, 52083, 85640, 42153 AZ, 515910, 828669, 362642 name:stacked bar chart,x axis:states,y axis:population in different ages,xvalues:AL,AK,AZ...,y1name:under 5 years,y1values:310504,52083,...,y2name:5 to 13 years,y2values:552339,85640,....	{ "values": [{"x":,"y":[] }, {"x":,"y":[]},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : , "y_names" : [] ,"y_labels" : [] }	Comparisons, Relationships, Patterns, Proportions

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Arc Diagram	Case 1: node(name) link(source,target,value) Case 2: node(name,info) link(source,target,value) Vocabulary name, node_label, node_name, Source, Target, value, value_type, value_label.	Case 1: source(Creditor):Britain..., target(debtor):France, Greece..., value(amount):22.4,0.55... creditor,debtor,amount Britain,France,22.4 Britain,Greece,0.55 Britain,Italy,26 Britain,Portugal,19.4 Case 2: "nodes":[{"name":"Myriel","group":1}, {"name":"Napoleon","group":1}, {"name":"Mlle.Baptistine","group":1}], "links":[{"source":Napoleon,"target":Myriel,"values":1}, {"source":2,"Mlle.Baptistine":Myriel,"values":8}, {"source":3,"Mooray":Myriel,"values":10}] name::Napoleon,Myriel... info(group):1,2... source(name):Napoleon,Mooray... target(name):Myrie.. value(values):8,10...	{ "nodes" : [{ "source" : , "target": , "value" : }, { "node_name" : , "parent": , "value" : },] , name : , "node_type" : , "value_type" : }	Patterns, Relationships
Brainstorm	node(name,value) link(node,parentnode)	['Location', 'Parent', 'Market trade volume (size)'] ['Global', null, 0] ['America', 'Global', 0] ['Europe', 'Global', 0] ['Asia', 'Global', 0] ['Australia', 'Global', 0] ['Africa', 'Global', 0] ['Brazil', 'America', 11]	{ "nodes" : [{ "node_name" : , "parent": , "value" : }, { "node_name" : , "parent": , "value" : },] , name : , "node_type" : , "value_type" : }	Concepts, Relationships
Sankey Diagram	Case 1: node(name) link(source,target,value) Case 2: node(name,info) link(source,target,value)	same as arc diagram	{ "nodes" : [{ "source" : , "target": , "value" : }, { "node_name" : , "parent": , "value" : },] , name : , "node_type" : , "value_type" : }	How things work, Flow, Process, Proportions
Timeline	name ,xaxis ,yaxis ,yvalues,x1values, x1name(start time),x2values, x2name(end time)	name:name of chart ,xaxis: x axis label ,yaxis:y axis label ,yvalues: values for y axis,x1values(start time):time values for each y values,x2values(end time):time values for each y values e .g American presidents served their terms President Start Date End Date Washington 1789 1797 Adams 1797 1801 Jefferson 1801 1809 name:Timline shows American Presidents service period ,xaxis:Time,yaxis :Presidents Name ,yvalues:Washington,Adams...,x1values(start time):1789,1797...,x2values(end time):1797,1801...	{ "values": [{"x1":,"x2": ,"y": }, {"x1":,"x2": ,"y":},] , "name" : , "xaxis_label" : , "xaxis_type" : , "yaxis_label" : , "yaxis_type" : , "x1_name" : , "x2_name" : }	Data over time

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Tree Diagram	node(name,value) link(node,parentnode) Vocabulary of Tree will be name, value_label, node_label, node_name, parent, value, value_type.	name: name of chart, value_label: label of value column, node_label: label of node, node_name: name of node, Parent: name of parent node, Value: value of node, value_type: data type of node. Example given ['Location', 'Parent', 'Market trade volume (size)'] ['Global', null, 0] ['America', 'Global', 34] ['Europe', 'Global', 20] ['Asia', 'Global', 10] ['Australia', 'Global', 50] ['Africa', 'Global', 5] ['Brazil', 'America', 11] Name: Tree diagram, value_label: Market trade volume (size), node_label: Location, node_name: Global, America,, Parent: Global, Value: 34, 10, 50...., value_type: numbers,	{ "values" : [{ "node_name" : , "parent": , "value" : }, { "node_name" : , "parent": , "value" : },] , name : , "value_type" : , ”value_label”:, “node_label”:, }	Hierarchy, Relationships
Venn Diagram	circle(name,size) overlap(circle1,circle2,size)	var sets = [{"label": "Radiohead", "size": 77348}, {"label": "Thom Yorke", "size": 5621}, {"label": "John Lennon", "size": 7773}, {"label": "Kanye West", "size": 27053},] overlaps = [{"sets": [0, 1], "size": 4832}, {"sets": [0, 2], "size": 2602}, {"sets": [0, 3], "size": 6141}, {"sets": [0, 4], "size": 2723}, {"sets": [0, 5], "size": 3177},]	{ "circles" : [{"label": "", "size": }, {"label": "", "size": }] , "overlaps" : [{"sets": ["label","label1"], "size": }, {"sets": [], "size": }] , "name" : }	Comparisons, Concepts, Probability, Relationships
Calendar	name,dates,values	name:name of chart,dates:dates in some format,values:value for each date e.g. Calendar showing temperature [Dates in seconds ,Value] [946702811", 15] ["946702812", 25] ["946702813", 10] name:Calander ,dates:946702811,946702812..., values:15,25,10...	{ "values" : [{ "date" : , "value" : }, { "date" : , "value" : },] , name : , "date_type" : , "value_type" : }	Time

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Donut Chart	name, radient_label, dimension_label, radient_values, dimension_values	name:name of chart radient_label:name for r axis dimension_label:dimension name radient_values:values for r dimension_values:values for dimension e.g. ['Task', 'Hours per Day'], ['Work', 11], ['Eat', 2], ['Commute', 2], ['Watch TV', 2] name:Donut chart showing daily activities, radient_label:task, dimension_label:hours per day, radient_values:work,eat...., dimension_values:11,2,2.....	{ "values": [{"r":,"d": }, {"r":,"d":},], "name" : , "radient_label" : , "r_type" : , "diemension_name" : , "dimension_type" : }	Comparisons, Proportions
Dot Matrix Chart	name,x1cellname,x1cell_ydata,x1cell_yvalues,xn (if n = 1 is single dot matrix)	name:name of chart,x1cellname:name of first dot matrix chart ,x1cell_ydata:data labels for first i.e. x1 dot matrix,x1cell_yvalues: values for each y data in x1,xn:xn dot matrix e.g. single dot matrix [ydata ,values] [normal ,4] [error, 3] [unknown,5] data expressed as person icon name: single dot matrix,x1cellname:data expressed as person icon,x1cell_ydata:normal,erroe,unknown...,x1cell_yvalues:4,3,5...	{ "name" : , "cells" : [{"cell_name" : , "data_type" : "data_name" : , "value_type" : , "values" : [{"data" : , "value" :}] }] }	Comparisons, Distribution, Patterns, Proportions
Parallel Sets	data(dimension1_key,dimension2_key,dimensionN_key) dimension(name,keys,keyvalues)	Class (0 = crew, 1 = first, 2 = second, 3 = third) Age (1 = adult, 0 = child) Sex (1 = male, 0 = female) Survived (1 = yes, 0 = no) [Class,Age,Sex,suvived] 1 1 0 0 1 1 0 0 1 1 0 0 1 0 1 1 1 0 1 1 1 0 1 1 data(1,1,0,0) dimension(class,[0,1,2,3],[crew,first,second,third]) dimension(age,[0,1],[child,adult])	{ "data" : [], "dimensions" : [{ "dimenson_name ": , "keys" : [{"key" : , "value" : }, {"key" : , "value": }] }] }	Comparisons, Distribution, Flow, Process, Proportions
Pictogram Chart	category(category-name,shape,shape-value,total-value)			Comparisons, Distribution
Pie Chart	name, radient_label, dimension_label, radient_values, dimension_values	name:name of chart radient_label:name for r axis dimension_label:dimension name radient_values:values for r dimension_values:values for dimension eg ['Task', 'Hours per Day'], ['Work', 11], ['Eat', 2], ['Commute', 2], ['Watch TV', 2] name:Pie chart showing daily activities, radient_label:task, dimension_label:hours per day, radient_values:work,eat...., dimension_values:11,2,2.....	{ "values": [{"r":,"d": }, {"r":,"d":},], "name" : , "radient_label" : , "r_type" : , "dimension_label" : , "dimension_type" : }	Comparisons, Proportions

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Proportional Area Chart	node(name,value) link(node,parentnode) Vocabulary of Tree will be name, value_label, node_label, node_name, parent, value, value_type.	name: name of chart, value_label: label of value column, node_label: label of node, node_name: name of node, Parent: name of parent node, Value: value of node, value_type: data type of node. Example given ['Location', 'Parent', 'Market trade volume (size)'] ['Global', null, 0] ['America', 'Global', 34] ['Europe', 'Global', 20] ['Asia', 'Global', 10] ['Australia', 'Global', 50] ['Africa', 'Global', 5] ['Brazil', 'America', 11] Name: Tree diagram, value_label: Market trade volume (size), node_label: Location, node_name: Global, America,, Parent: Global, Value: 34, 10, 50...., value_type: numbers,	{ "values" : [{ "node_name" : , "parent": , "value" : }, { "node_name" : , "parent": , "value" : },] , name : , "value_type" : , ”value_label”:, “node_label”:, }	Comparisons, Proportions
Word Cloud	name,word_name,word_size	name:name of chart word_name : name of words word_size:size of each word which shows frequency of word [word-name , word-size] [winter, 3] [snow, 1] [play, 1] [cold , 2] [day, 2] name:word cloud, word_name :{winter,play,snow}, word_size :{3,1,2}	{ "values" : [{ "word_name" : , "word_size" : }, { "word_name" : , "word_size" : },] , name : , "size_type" : }	Analyzing Text, Comparisons, Distribution/Frequency, Proportions

Type Of Chart	Vocabulary	Vocabulary Description	JSON structure	Functions
Choropleth Map(Geo Map)	1.region_code or region_name, data_label, data 2. region-name, longitude, latitude, color_data, color_data_label, color_data_type, size_data_label, size_data_type, size_data	region_code or region_name:name of region data : data for that region. data_label : label of data e.g. map showing courtiers with their popularity ['Country', 'Popularity'], ['Germany', 200], ['United States', 300], ['Brazil', 400], ['Canada', 500], ['France', 600] region_name:Germany,Brazil data_label : popularity, data : 200,300... 2. case with markers ['City', 'Longitude' 'Latitude' 'Population', 'Area'], ['Rome', 12.5113300 41.8919300, 2761477, 1285.31], ['Milan', 9.1895100 45.4642700, 1324110, 181.76], ['Naples', 14.2464100 40.8563100 ,959574, 117.27], region-name:Rome,Milan... Longitude:12.5113300,9.1895100... Latitude:41.8919300,45.4642700... color_data_label :Population, color_data ;2761477,1324110,... size_data_label : Area, size_data :1285.31,181.76....	1. { "values" : [{ "region_name" : , "longitude": , "latitude" : , "data" : }, { "region_name" : , "longitude": , "latitude" : , "data" : },] , "name" : , "data_label" : } 2. { "values" : [{ "region_name" : , "longitude": , "latitude" : , "color_data" : , "size_data": }, { "region_name" : , "longitude": , "latitude" : , "color_data" : , "size_data":: },] , "name" : , "color_data_label" : , "color_data_type" : ,"size_data_label" : , "size_data_type" : }	Comparisons, Location, Patterns

B

REVIEW OF DATA VISUALIZATION TOOLS

This appendix contains review of standard visualization libraries based on the types of charts supported, browsers supported and other additional features supported.

Appendix B: Review of Data Visualization Tools

1. Visualization Tools

		Open source	Latest version	Trial and Prices
amCharts	http://www.amcharts.com/	No	3.X	Free with watermark $99 (single website)
arcadiaCharts	http://www.arcadiacharts.com/	No	1.0.2	Free for non commercial use $ 89 (single website) - $ 899 (OEM)
CanvasJS Charts	http://canvasjs.com/	CC license 3.0	1	Free for non-commercial use $299+ for commercial license
D3.js	http://d3js.org/	BSD License	2.10.2003	Free under BSD
dhtmlxChart	http://www.dhtmlx.com/docs/products/dhtmlxChart/index.shtml	GNU GPL	2.6 Build 100928	Free under GNU GPL, $49
Dojo (dojox/charting)	http://dojotoolkit.org/	BSD, AFLv2	1.8	Free
Ejschart	http://www.ejschart.com/	No	2.3	Free / $100 / $250 / $1000
Elycharts	http://elycharts.com/	MIT License	2.1.2004	Free
Flot	http://www.flotcharts.org/	MIT License	0.8.1 (may 2013)	Free
flotr2	http://www.humblesoftware.com/flotr2/	MIT License		Free
Google Chart Tools	https://developers.google.com/chart/	No		Free
gRaphaël	http://g.raphaeljs.com/	MIT License	0.5.0	Free (you can donate)
Highcharts	http://www.highcharts.com/	CC by-nc 3.0	1	Free for non commercial use $ 80 (single website) - $ 2000 (10 developers license)
jqChart	http://www.jqchart.com/	No		$299
jqPlot	http://www.jqplot.com/	MIT, GPL v2	2013	Free
JSCharts	http://www.jscharts.com/	No	3	$ 39 - $ 149 Free with watermark
JSXGraph	http://jsxgraph.uni-bayreuth.de/wp/	LGPL	0.94	Free
KendoUI DataViz	http://www.telerik.com/kendo-ui-dataviz	No	Q1 2013	$ 399
Morris.js	http://www.oesmith.co.uk/morris.js/	BSD	0.4.1	Free
nvd3	http://nvd3.org/	Apache 2.0		Free depending on Apache 2.0

Protovis	http://mbostock.github.io/protovis/	BSD License	3.3.2001	Free
RGraph	http://www.rgraph.net/	**No**	2012	Free for non-commercial (CreativeCommons) License for commercial use.
Rickshaw	http://code.shutterstock.com/rickshaw/	**Yes**	2012	Free of charge with copyright attribution
Sencha Touch Charts	http://dev.sencha.com/deploy/touch-charts-rc/	**No**		Free under GPLv3 license; $999 commercial license
TeeChart	http://www.steema.com/teechart/html5	**No**	2012	Free for non commercial use. $129 commercial license (1developer + 1 server install + 1 year support subscription)
zingchart	http://www.zingchart.com/	**No**	2010	Free with watermark Single Domain Package : $249.00 Discounted Multi-Domain Package : $999.00 SaaS and OEM Pricing Available
Shield UI Charts	https://www.shieldui.com/products/chart	**No**	1.4.2002	$299
SVGware	http://www.svgware.com/	**No**	2.6 (July 2013)	Free
Reportivo.com	http://reportivo.com/			Free

2. Supported Chart Types

	Graphic technology	Area	Bar	Bubble	Line	Pie	Scatter	Spline	Sparklines	Candlestick	Donut	Node / Edge graph	Other charts
amCharts	SVG VML for old IE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Gauges, Radar, Polar, Step line, Step without Risers, OHLC, area range, Donut, Funnel, Waterfall, Bullet, XY, Maps are available as additional package
arcadiaCharts	Canvas	No	Yes	No	Yes	Yes		No		No	Yes		
CanvasJS Charts	CanvasJS	Yes	Yes	Yes	Yes	Yes	Yes	Yes		No	Yes		Bubble Chart
D3.js	SVG	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes	Yes	D3.js is a drawing library, and not just a chart library. You can make virtually any visualization or graph given time.
dhtmlxChart	Canvas	Yes	Yes	No	Yes	Yes	No	Yes	Yes	No	Yes		
Dojo (dojox/charting)	SVG Canvas VML Silverlight	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No		Bubble, Candlestick (OHLC), Spider. Other Dojo modules provide as well TreeMap, Gauges etc...
Ejschart	Canvas SVG VML	Yes	Yes	No	Yes	Yes	Yes	-	-	Yes	Yes		f(x), Gauge, Donut, alarm, candle stick, step series, over under, open high low, error series, stacked bar, trend, and floating bar
Elycharts	SVG	Yes	Yes	No	Yes	Yes	No	Yes	Yes	No	Yes		Funnel
Flot	Canvas	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes		Bubbles, Gantt, Pyramid, Spider
flotr2	Canvas	Yes	Yes	Yes	Yes	Yes	Yes	No		Yes	No		Bubble, radar
Google Chart Tools	SVG	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes	No		Geo chart, table, gauge, tree map
gRaphaël	SVG	Yes	Yes	No	Yes	Yes	No	No	No	No	No		
Highcharts	SVG	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes		
jqChart	Canvas	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	No		Bubble, radar, polar
jqPlot	Canvas	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		

JSCharts	Canvas	No	Yes	No	Yes	Yes	No	No		No	No		
JSXGraph	SVG	No	Yes	No	Yes	Yes	No	Yes		No	No		Math...
KendoUI DataViz	SVG	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes		Linear Gauge, Radial Gauge, Bubble, Bullet, Donut, Scatter, Stock
Morris.js	SVG	Yes	Yes	No	Yes	No	No	Yes	No	No	Yes		
nvd3	SVG	Yes	Yes	Yes	Yes	Yes	Yes	Yes		No	Yes		Bullet chart
Protovis	SVG	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		TreeMap, Node links
RGraph	Canvas	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes		Gauge, Funnel, Waterfall
Rickshaw	SVG	Yes	Yes	No	Yes	No	Yes	Yes		No	No		
Sencha Touch Charts	Canvas	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	No		
TeeChart	Canvas	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes		Horizbar, SmoothLine, Donut, HorizArea, Bubble, Candle (OHLC)
zingchart	Canvas SVG VML Flash	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Bullet, Chord, Funnel, Gauge, Grid, Maps, Pareto, Piano, Radar, Rankflow, Stock, Treemap, Venn, WordCloud
Shield UI Charts	SVG VML	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes		Range Bar/Area/SplineArea, Polar Bar/Area/Spline/Scatter, Stepline, Steparea
SVGware	SVG	Yes	Yes	Yes	Yes	No	Yes	No	No	No	No		heat map; error bars; linear and logarithmic scales;
Reportivo.com		Yes	Yes		Yes	Yes	Yes	Yes			Yes		Range Bar/Area/Spline Area, Polar Bar/Area/Spline/Scatter/Stepline/Steparea

3. Additional Chart Features

	Ability to zoom in and out of charts	Annotations on the chart	Combination of charts	Data labels	Date-time axis	Dynamic charts	Export files	External Data Loading	Interactive (responds to mouse hover/click)	Print	Text Rotation for Labels
amCharts	Yes	Yes	Yes	Yes	Yes	Yes	Yes PNG, JPG, SVG, PDF		Yes	Yes	Yes
arcadiaCharts			Yes			Yes			Yes		
CanvasJS Charts	Yes		Yes	Yes	Yes	Yes			Yes		Yes
D3.js	Yes		Yes	Yes	Yes	Yes			No		Yes
dhtmlxChart	No	Yes	No	Yes	No	Yes	No	Yes	No	No	Yes With CSS
Dojo (dojox/charting)	Yes	No	Yes	-	- Feasible with custom code	Yes	Yes SVG	Yes	No	Yes	Yes
Ejschart	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	Yes
Elycharts	No	No	No	Yes	No	Yes	No	No	Yes	Yes	Yes
Flot	Yes	Yes	Yes	Yes		Yes	No	Yes	Yes	No	
flotr2	Yes	Yes	Yes		Yes	Yes	Yes PNG, JPG	No	Yes	No	Yes
Google Chart Tools	No	No	Yes	No	Yes	No	No	No	Yes	No	No

gRaphaël	No	No	No	Yes	No	No	No	No	Yes	No	No
Highcharts	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
jqChart	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes
jqPlot	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes
JSCharts	No	No	Yes	Yes	No	No	No	No	No	No	No
JSXGraph	No		Yes	Yes	Yes	Yes		Yes	Yes		No
KendoUI DataViz	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Morris.js	No	No	No	No	Yes		No	No	Yes	No	No
nvd3	Yes	Yes	Yes	Yes		Yes			Yes		
Protovis	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No		Yes
RGraph	No	No	No	No	No	No	-	Yes	Yes		
Rickshaw	Yes			Yes					Yes		
Sencha Touch Charts									Yes		
TeeChart	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes
zingchart	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Shield UI Charts	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SVGware	Yes	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes	Yes	
Reportivo.com	Yes	Yes		Yes	Yes		Yes	Yes	Yes	Yes	Yes

4. Browser Support

	Firefox	IE	Chrome	Safari	Opera	iPhone	iPad
amCharts	Yes	Yes 6.0+	Yes	Yes	Yes	Yes	Yes
arcadiaCharts	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CanvasJS Charts	Yes	Yes 9+	Yes	Yes	Yes	Yes	Yes
D3.js	Yes	Yes 9+	Yes	Yes	Yes	Yes	Yes
dhtmlxChart	Yes 1.0+	Yes 6.0+	Yes	Yes 3.0+	Yes 9.0+	Yes	Yes
Dojo (dojox/charting)	Yes 3.6+	Yes 6+	Yes	Yes	-	Yes	Yes
Ejschart	Yes 1.5+	Yes 6.0+	Yes	Yes 3.1	Yes 9+	Yes 1+	Yes
Elycharts	Yes 3.0+	Yes 6.0+	Yes 5.0+	Yes 3.0+	Yes 9.5+	Yes	Yes
Flot	Yes	Yes 6.0+	Yes	Yes	Yes		Yes
flotr2	Yes	Yes 6.0+	Yes	Yes		Yes	Yes
Google Chart Tools	Yes	Yes	Yes	Yes	Yes	Yes	Yes
gRaphaël	Yes 3.0+	Yes 6.0+	Yes 5.0+	Yes 3.0+	Yes 9.5+		
Highcharts	Yes	Yes 6.0+	Yes	Yes	Yes	Yes	Yes
jqChart	Yes	Yes 6.0+	Yes	Yes	Yes	Yes	Yes

jqPlot	**Yes**	**Yes** 7.0+	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
JSCharts	**Yes** 1.5+	**Yes** 8.0+	**Yes** 10+	**Yes** 3.1+	**Yes** 9+	**Yes** 1.0+	
JSXGraph	**Yes** 2.0+	**Yes** 6.0+	**Yes**	**Yes** 3.0+	**Yes**	**Yes**	**Yes**
KendoUI DataViz	**Yes**	**Yes** 7+	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Morris.js	**Yes**	**Yes** 6.0+	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
nvd3	**Yes**	**-** IE 9, IE8 requires Aight	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Protovis	**Yes**	**No**	**Yes**	**Yes**		**Yes**	**Yes**
RGraph	**Yes**	**Yes** 9+	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Rickshaw	**Yes**	**Yes** 9+	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Sencha Touch Charts						**Yes**	**Yes**
TeeChart	**Yes**	**Yes** 9.0+	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
zingchart	**Yes**	**Yes** 6.0+	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Shield UI Charts	**Yes**	**Yes** 5.0+	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
SVGware	**Yes**	**Yes**	**Yes**		**Yes**	**Yes**	**Yes**
Reportivo.com	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

C

THE VOVIS PROTOTYPE WEB APPLICATION

This appendix contains the user manual to download, install and run the *VoVis* application on the web. The *VoVis* framework is client-side application, so a server is needed to host the application. MongoDB is used as database and Node.js as the server and the web service.

C.1 STEPS TO START AND LAUNCH THE VOVIS APPLICATION

To execute the *VoVis* web application, Node.js and MongoDB softwares need to be installed on a system which has a web browser, as the *VoVis* application is a web-based prototype.

- Installation and execution steps of MongoDB: The following steps will start the database service for the *VoVis* application.
  1. Download and Install the MongoDB from [48].
  2. Create a directory (/data/db).
  3. To start MongoDB service (daemon), run the command "mongod –dbpath /data/db" from the installed directory of MongoDB in the system, through the command line.
- Steps to install node.js and launch *VoVis* web application on the web.
  1. Download and Install the node.js from [49]
  2. Download the *VoVis* source code from [50]
  3. From the root directory, execute the following commands through command line - "npm install" and " npm start"
  4. To launch the *VoVis* framework on the web, type this url "http://localhost/" in the browser.
  5. The *VoVis* framework is ready for visualization of data.

C.2 STEPS TO VISUALIZE THE DATA BY THE VOVIS APPLICATION

The steps for visualizing the data by the *VoVis* framework is as follows.

- First step is to upload the input file in CSV format. Figure 48 is the Home page of The *VoVis* framework. The user can either upload the new data or can download data from the database to visualize it.

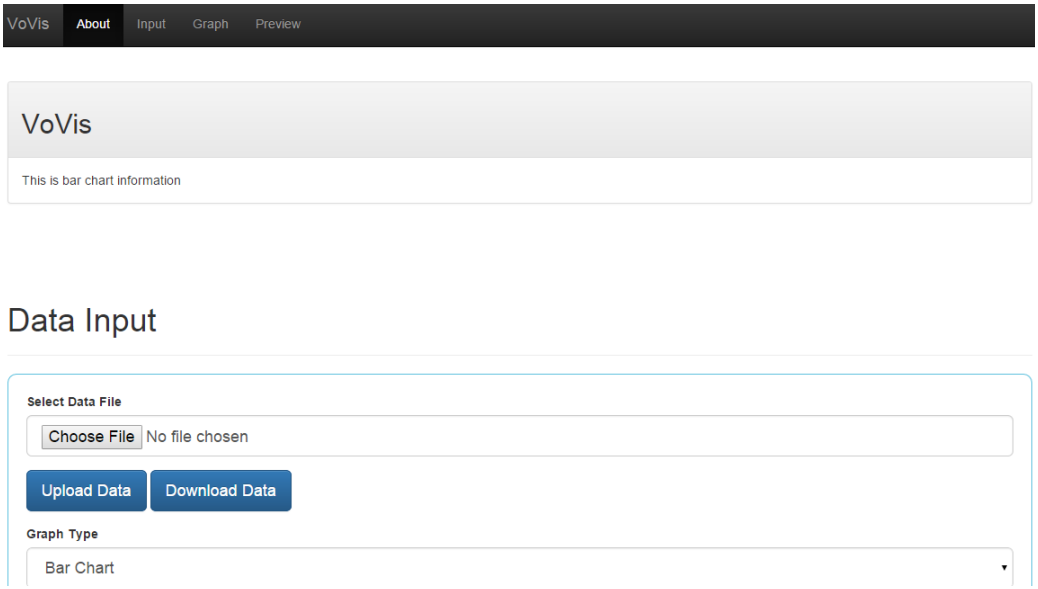


Figure 48: *VoVis* Home Page

- To select a type of visualization from the drop down menu. Once a type is selected information regarding the visual type is shown in the information box. This helps in mapping of data with the VisVo vocabulary.
- The drag and drop feature of *VoVis* provides help for mapping of the data columns to the VisVo vocabulary and manually typing some input for labels as the name of the chart. Figure 49 shows the mapping of data through drag and drop feature of the *VoVis* framework.

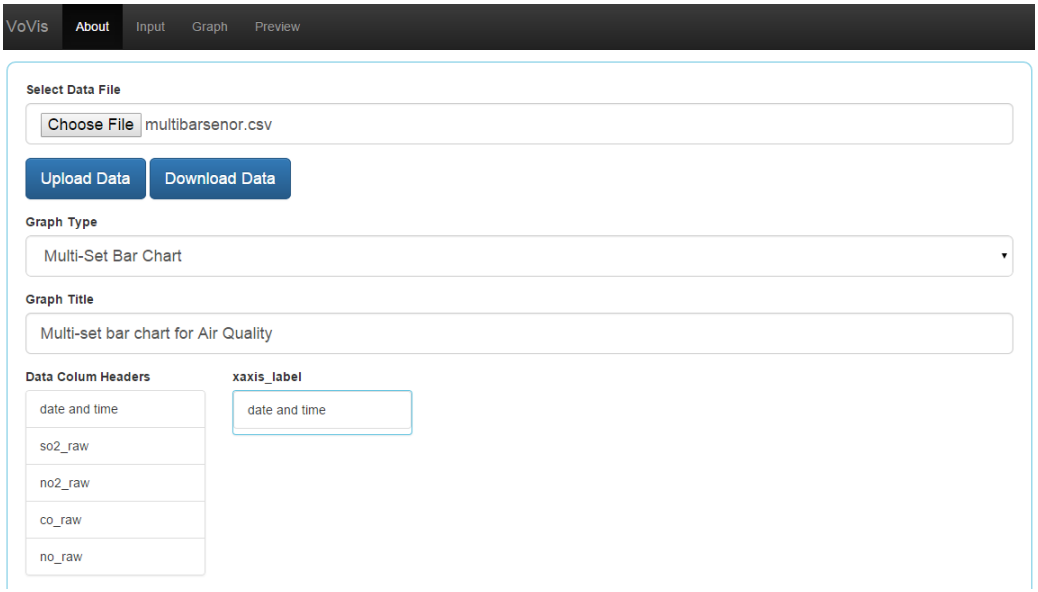


Figure 49: *VoVis* Input Data

- Once mapping is done, next is to process it by clicking the process button. Remapping is also possible by reset method that will reset the mapping.
- The processed data and the visualization is generated and can be viewed and accessed on the web page. Figure 50 shows the multi-set bar visualization and Figure 51 shows the processed data generated by the *VoVis* framework.

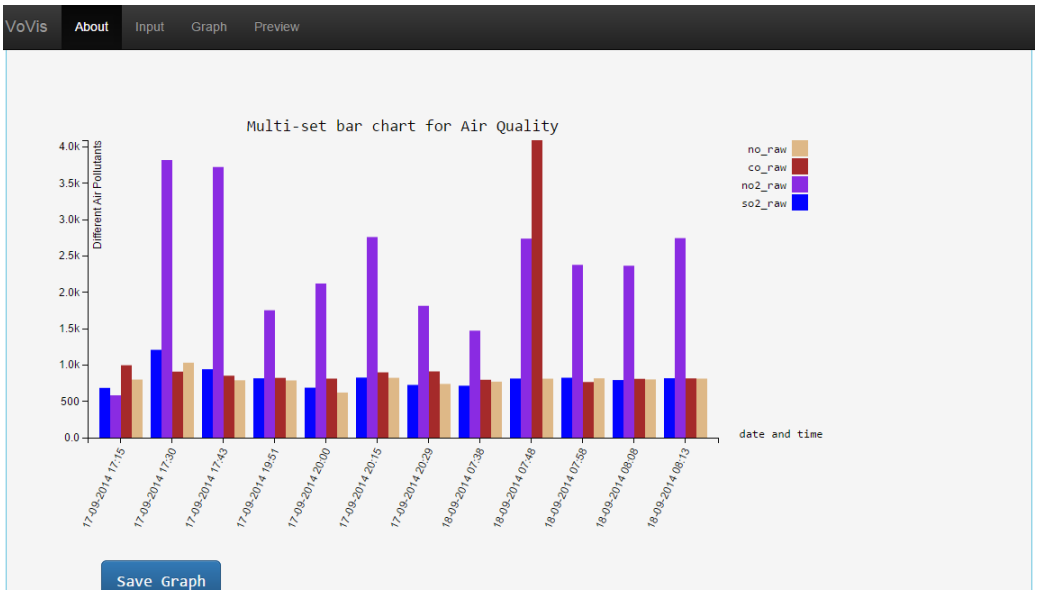


Figure 50: *VoVis* Visualization

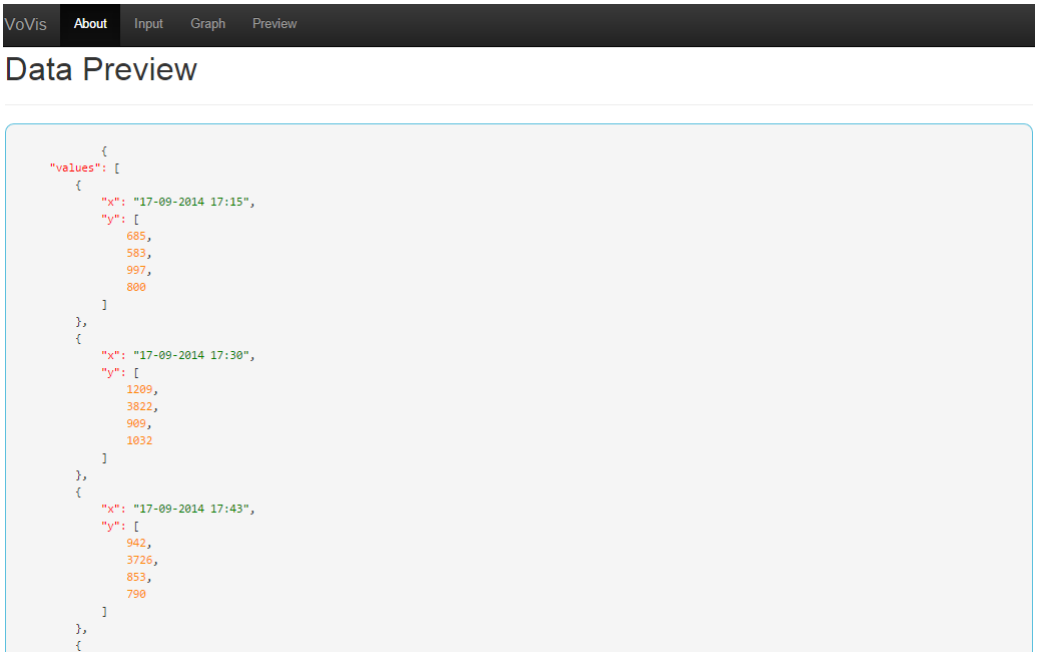


Figure 51: *VoVis* Processed Data

BIBLIOGRAPHY

[1] Allen EA, Erhardt EB, Calhoun VD (2012) Data visualization in the neurosciences: overcoming the curse of dimensionality. Neuron 74(4):603–608 (Cited on page 14.)

[2] Berlocher I, Kim S, Lee T (2014) Deliverable d5.2: Use case implementation. Tech. rep., DaPaaS Consortium 2013-2015, URL http://project.dapaas.eu/ (Cited on pages 23 and 24.)

[3] Castell N, Kobernus M, Liu HY, Schneider P, Lahoz W, Berre AJ, Noll J (2014) D.2.2: Evaluation of the performance. Tech. rep., Citi-Sense-MOB, URL http://citi-sense-mob.eu/ (Cited on page 25.)

[4] Castell N, Kobernus M, Liu HY, Schneider P, Lahoz W, Berre AJ, Noll J (2014) Mobile technologies and services for environmental monitoring: The citi-sense-mob approach. Citi-Sense-MOB URL http://citi-sense-mob.eu/ (Cited on page 4.)

[5] Catalogue DV (2014) The data visualisation catalogue. URL http://www.datavizcatalogue.com/search.html (Cited on pages 56 and 67.)

[6] Chan B, Wu L, Talbot J, Cammarano M, Hanrahan P (2008) Vispedia: Interactive visual exploration of wikipedia data via search-based integration. Visualization and Computer Graphics, IEEE Transactions on 14(6):1213–1220 (Cited on page 59.)

[7] Chartchooser (2014) Chartchooser. URL http://labs.juiceanalytics.com/chartchooser/index.html (Cited on page 57.)

[8] Charts DR (2015) Radial bar chart. URL http://prcweb.co.uk/radialbarchart/ (Cited on page 68.)

[9] Chen Ch, Härdle WK, Unwin A (2007) Handbook of data visualization. Springer Science & Business Media (Cited on page 12.)

[10] Chooser S (2015) Slide chooser. URL http://extremepresentation.typepad.com/blog/2015/01/announcing-the-slide-chooser.html (Cited on page 58.)

[11] Cooke R (2015) Data visualization. URL http://spinsucks.com/marketing/importance-data-visualization/ (Cited on pages 11 and 12.)

[12] Cukier K (2010) A special report on managing information. The Economist 394(8671):16 (Cited on page 1.)

[13] Cyganiak R, Reynolds D, Tennison J (2013) The rdf data cube vocabulary. W3C Recommendation (January 2014) (Cited on page 59.)

[14] Davies J, Fensel D, Van Harmelen F (2003) Towards the semantic web: ontology-driven knowledge management. John Wiley & Sons (Cited on page 58.)

[15] Dewar M (2012) Getting Started with D3. O'Reilly Media (Cited on page 80.)

[16] Dumontier M, Ferres L, Villanueva-Rosales N (2010) Modeling and querying graphical representations of statistical data. Web Semantics: Science, Services and Agents on the World Wide Web 8(2):241–254 (Cited on page 59.)

[17] Fayyad UM, Wierse A, Grinstein GG (2002) Information visualization in data mining and knowledge discovery. Morgan Kaufmann (Cited on page 14.)

[18] Few S (2004) Show me the numbers: Designing tables and graphs to enlighten, vol 1. Analytics Press Oakland, CA (Cited on page 12.)

[19] Few S (2009) Now you see it: simple visualization techniques for quantitative analysis. Analytics Press (Cited on page 11.)

[20] Freitas CM, Luzzardi PR, Cava RA, Winckler M, Pimenta MS, Nedel LP (2002) On evaluating information visualization techniques. In: Proceedings of the working conference on Advanced Visual Interfaces, ACM, pp 373–374 (Cited on page 22.)

[21] Friendly M, Denis DJ (2001) Milestones in the history of thematic cartography, statistical graphics, and data visualization. URL http://wwwdatavisca/milestones (Cited on page 13.)

[22] Ghatol R, Patel Y (2012) Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5. Apress (Cited on page 91.)

[23] GraphicsCheatSheet (n.d.) Graphicscheatsheet. URL `http://www.billiondollargraphics.com/GraphicsCheatSheet_GMG.pdf` (Cited on page 58.)

[24] Johnson C (2004) Top scientific visualization research problems. Computer graphics and applications, IEEE 24(4):13–17 (Cited on page 13.)

[25] Jou SF, Campbell D, Ballantyne I (2006) Interactive business data visualization system (Cited on page 14.)

[26] Kelleher C, Wagener T (2011) Ten guidelines for effective data visualization in scientific publications. Environmental Modelling & Software 26(6):822–827 (Cited on page 19.)

[27] Kerren A, Stasko J, Fekete JD (2008) Information visualization: human-centered issues and perspectives. Springer Science & Business Media (Cited on page 13.)

[28] Kessin Z (2011) Programming HTML5 applications: building powerful cross-platform environments in JavaScript. " O'Reilly Media, Inc." (Cited on page 77.)

[29] Lee T, Kim S, Berlocher I (2014) Deliverable d5.1: Use case definition and requirements analysis. Tech. rep., DaPaaS Consortium 2013-2015, URL http://project.dapaas.eu/ (Cited on page 23.)

[30] Leff A, Rayfield J (2001) Web-application development using the model/view/controller design pattern. In: Enterprise Distributed Object Computing Conference, 2001. EDOC '01. Proceedings. Fifth IEEE International, IEEE, pp 118–127 (Cited on page 51.)

[31] Library (2014) Datavis. URL http://guides.library.duke.edu/datavis (Cited on page 55.)

[32] Maclean M (2014) D3 Tips and Tricks Interactive Data Visualization in a Web Browser. Leanpub (Cited on page 80.)

[33] McGuinness DL, Van Harmelen F, et al (2004) Owl web ontology language overview. W3C recommendation 10(10):2004 (Cited on page 58.)

[34] Murray S (2013) Interactive Data Visualization for the Web. O'Reilly Media (Cited on pages 39 and 80.)

[35] Mutlu B, Hoefler P, Sabol V, Tschinkel G, Granitzer M (2013) Automated visualization support for linked research data. I-SEMANTICS (Posters & Demos) 1026:40–44 (Cited on page 59.)

[36] Puhan MA, ter Riet G, Eichler K, Steurer J, Bachmann LM (2006) More medical journals should inform their contributors about three key principles of graph construction. Journal of clinical epidemiology 59(10):1017–e1 (Cited on page 13.)

[37] Rebolj D, Sturm PJ (1999) A gis based component-oriented integrated system for estimation, visualization and analysis of road traffic air pollution. Environmental Modelling & Software 14(6):531–539 (Cited on page 11.)

[38] Silva S, Santos BS, Madeira J (2011) Using color in visualization: A survey. Computers & Graphics 35(2):320–333 (Cited on page 12.)

[39] Solheim I, Stølen K (2007) Technology Research Explained. Tech. rep., SINTEF ICT (Cited on page 6.)

[40] Sridaran R, Padmavathi G, Iyakutti K (2009) A survey of design pattern based web applications. Journal of object technology 8(2):61–70 (Cited on page 51.)

[41] Telea AC (2014) Data Visualization: Principles and Practice. A K Peters/CRC Press (Cited on pages 12, 13, and 15.)

[42] Thomas JJ (2005) Illuminating the path:[the research and development agenda for visual analytics]. IEEE Computer Society (Cited on page 13.)

[43] Tilkov S, Vinoski S (2010) Node. js: Using javascript to build high-performance network programs. IEEE Internet Computing 14(6):0080–83 (Cited on page 82.)

[44] Ward MO, Grinstein G, Keim D (2015) Interactive data visualization: foundations, techniques, and applications. CRC Press (Cited on pages 12, 14, and 19.)

[45] Ware C (2012) Information visualization: perception for design. Elsevier (Cited on page 1.)

[46] Website (2014) Pragmatic visualization. URL http://kosara.net/papers/2007/Kosara_IV_2007.pdf (Cited on page 12.)

[47] website (2014) Sas. URL http://www.sas.com/en_us/insights/big-data/data-visualization.html (Cited on pages 11 and 12.)

[48] Website (2015) Downloads-mongodb. URL https://www.mongodb.org/downloads (Cited on page 127.)

[49] Website (2015) Node.js. URL https://nodejs.org/download/ (Cited on page 127.)

[50] Website (2015) Vovis github. URL https://github.com/SwatCodes/VoVis (Cited on pages 82 and 127.)

[51] Wikipedia (2015) Arc diagram. URL http://en.wikipedia.org/wiki/Arc_diagram (Cited on page 70.)

[52] Wikipedia (2015) Bar chart. URL http://en.wikipedia.org/wiki/Bar_chart (Cited on page 62.)

[53] Wikipedia (2015) Binary data. URL http://en.wikipedia.org/wiki/Binary_data (Cited on page 15.)

[54] Wikipedia (2015) Box plot. URL http://en.wikipedia.org/wiki/Box_plot (Cited on page 62.)

[55] Wikipedia (2015) Model view controller. URL http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (Cited on page 52.)

[56] Wikipedia (2015) Mongodb. URL http://en.wikipedia.org/wiki/MongoDB (Cited on page 81.)

[57] Wikipedia (2015) Node.js. URL http://en.wikipedia.org/wiki/Node.js (Cited on page 82.)

[58] Wong PC, Bergeron RD (1994) 30 years of multidimensional multivariate visualization. In: Scientific Visualization, pp 3–33 (Cited on page 13.)

[59] Zarev M, Roman D, Elvesæter B (2014) Deliverable d3.1: Analysis, requirements, design & architecture specification for the data access framework. Tech. rep., DaPaaS Consortium 2013-2015, URL http://project.dapaas.eu/ (Cited on page 8.)