Browser Based Categorization of Data Towards Automated Visualization

Steven Diviney

May 22, 2013

Submitted to Graduate Faculty of
Department of Computer Science
Of the requirements for the degree of
Masters Computer Science (MCS)

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work. I agree to deposit this thesis in the Universitys open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signature

Date

Abstract

This paper presents a system to automatically generate suitable visualizations given arbitrary data. Data Visualization is an increasingly common technique used to reinforce human cognition. In many areas of human activity the volume of data being generated is increasing rapidly. New methods must be employed to assist in the comprehension of this data. There has been a good amount of research performed to assess what factors contribute to the creation of an effective visualization. Additionally many new and novel visualizations have been created. However, automatic generation of visualizations has received little attention.

Such a tool would help to combine these two areas of research. An understanding of what factors contribute to an effective visualization are encoded into the system presented. Given an arbitrary dataset the system attempts to select the most appropriate visualization. This paper also discusses to what extent this process is viable.

Using the limited amount of information contained in a raw dataset it is possible to select a comprehensible visualization. As the dataset becomes increasingly complex the effectiveness of such a system diminishes. A number of datasets are input to the system and an evaluation is undertaken. The results presented show that such a system can be used to assist users in the creation of suitable visualizations while avoiding the creation of inappropriate or even misleading visualizations.

Attestation

I understand the nature of plagiarism, and I am aware of the Universitys policy on this. I certify that this dissertation reports original work by me during my University project.

Signature

Date

Acknowledgements

Contents

Chapter 1

Introduction

This chapter introduces the dissertation topic and explains the motivation behind the work. A research question is purposed and a number of objectives are outlined in an attempt to satisfy it. This is followed by evaluation criteria and finally by a summary of the document structure.

1.1 On Visualization

There are three fields subfields of Data Visualization. The boundaries between them are not particularly distinct and they are referred to somewhat interchangeably in academic literature. There are many other types but these three are of particular interest as their primary concern is the visualization of large volumes of data almost always with the aid of a computer.

1.1.1 Information Visualization

Information visualization is perhaps the most broadly used and can be thought to encompass all of the fields of visualization. The term today is generally applied to the visual representation of large-scale collections of non-numerical information, such as text in a book or files on a hard-disk. The distinction between this definition and that of Data Visualization seems to be quite poor. The type of the data in question is used to distinguish the two. The terms "information" and "data" are not so easily distinguished. Information can be thought of as a level of abstraction above data. The mere fact that a dataset is non-numerical does not identify it as information. A set of ordinal labels or categories is just as meaningless as a set of

numbers. Information is created by organizing such data and presenting it with context.

Information visualization is concerned with representing more abstract topics. A good example is process visualization. Each element in a process visualization represents a complex topic.

1.1.2 Data Visualization

Data Visualization is the science of visual representation of data, defined as "facts and statistics collected together for reference or analysis" (oed31, 2013). As stated the distinction between data and Information Visualization is not very concrete. This distinction is generally not regarded as important but this paper is concerned with Data Visualization. The synthesis of new information through the creation of visual artifacts.

1.1.3 Scientific Visualization

Scientific Visualization is concerned primarily with the visualization of objects in three dimensional space with an emphasis on realistic rendering. This emphasis on realism is primarily what distinguishes it from other forms of visualization. This is not to say that the other forms may distort the data, rather that abstract data does not necessarily have a spatial dimension. How would one realistically visualize the lines of a book? Novel ways must be invented to accomplish this.

1.2 Motivation and Description

Data Visualization is defined as an internal construction in the mind. The field of Data Visualization is connected with creating visual artifacts in order to facilitate individuals in building an internal representation of a dataset (Spence, 2001). Data Visualization is not a new field but it has only become an established area of scientific research in recent years (Friendly & Denis, 2001). The volume of data a typical computer can generate and process far exceeds what a human can comprehend. Data Visualization is becoming an increasingly popular technique to aid this comprehension.

Information visualization is a growing area of research. Presently there is a good amount of discussion surrounding new and novel visualization

techniques and how to create effective visualizations. The articles presented in these journals typically focus on the creation of specific visualizations, new and interesting ways to graphically represent specific types of data. There is also a suitable body of literature concerning the process of creating a visualization. There have been a number of works published on the subject (Mazza, 2009), (Shneiderman, 1996), (Bertin & Barbut, 1973), (Haskell, 1919). These works outline the steps needed to visualize different types of data but do not attempt to automate the process.

The majority of new techniques exists in some degree of isolation. Individual implementations typically exist in near complete isolation, often with the dataset hard coded into the software. There are a number of products that attempt to create a complete end to end process for visualizing data sets. These products are either highly specialized or lack complex visualizations and require user input throughout the process. An evaluation of a number of such systems is presented in section 2.3

Highly specialized applications such as gretl can afford to make numerous assumptions about the input dataset. It is assumed they will be used as part of a specific suite of tools and as such are able to directly process the proprietary output of such tools. Such output is often rich with meta-data which is used to assist the visualization process.

General purpose tools such as Microsoft Excel contain a number of simple charts and graphs. They require a basic level of user training to create and offer no assistance in selecting the most appropriate chart for a given dataset. This often leads to unsatisfactory, confusing or even misleading results. Figure 1.1 shows a snippet from an example data set and two visualizations produces by excel. The pie-chart omits a variable as it is not intended to visualize bivariate data. The scatter plot has plotted each variable independently. It is possible to combine them but the user may require training to do so.

With a few exceptions these tools are proprietary and lack documentation on the techniques they use.

This paper aims to address these deficiencies by providing a fully automated end to end visualization tool for arbitrary datasets. There have been some notable projects that accomplish such a goal (Stolte, Tang, & Hanrahan, 2002), (Mackinlay, 1986). Emphasis has been placed on areas where previous works have relied on human actors to complete the process.

Name	Population 2006	Unemployed 2006
AIRPORT	1611	66
BALBRIGGAN RURAL	9615	353

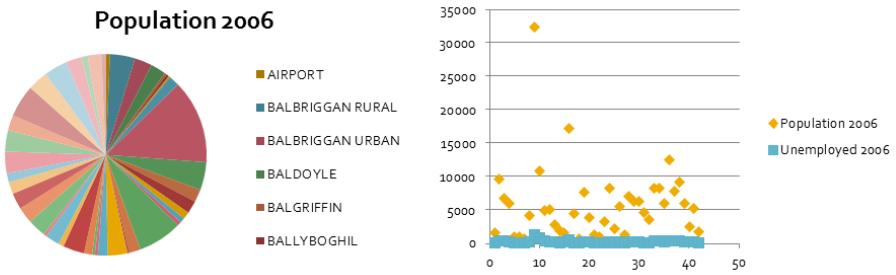

Figure 1.1: Example dataset and Excel visualization output.

1.3 Research Question

The objective of this project is to determine to what extent can suitable visualizations be dynamically and automatically generated using browser based technologies. The goals for the project are as follows:

- Design and develop software capable of accepting arbitrary data in a specific format and display it using suitable visualizations. This should be done without an intervention from the user.
- Access the level of benefit that the visualizations can bring to potential users from various fields.
- Investigate to what extent visualizations can be generated given only an input dataset.
- Elaborate on the potential of a more sophisticated version of the software using additional techniques to determine features of input datasets.

Key to this project is the notion of suitable visualizations. In order for a visualization to be considered useful it must meet several criteria. These criteria will be outlined in detail in section 3.3

Browser based technologies have been selected due to their wide-scale support. The goal of the system is to simplify the process of visualization creation. Browser based technologies require no setup or configuration and thus simplify the process. The use of browser based technologies also help to offload the vast majority of computation to the client. Furthermore client side technologies allow the bulk of computational effort to be offloaded from the server. These technologies have matured to a point where very elaborate applications can be run on a clients machine with the server merely providing data.

1.4 Evaluation

The system will be evaluated by inputting a number of datasets and assessing the output against the previously stated goals. A number of use cases have been drawn up to determine how beneficial such a tool is in aiding various users in visualizing data. These use cases entail typical visualization tasks a user from any number of different fields may encounter, i.e. visualizing expenditure and returns for different products.

The intention is not to create a system suitable for direct use by end users. It is the potential core of an easy to use visualization tool. For this reason an evaluation of the user experience is not needed. The goal of the system is to quickly produce effective visualizations that represent the data accurately.

Visualizations are composed of many individual elements with different attributes such as colour, size and spatial location. These elements have been the subject of previous research and a number of guidelines exists outlining their usage. However these guidelines are not hard rules and their exists no formal way to evaluate a complete visualization. Methods from the field of Human Computer Interaction are typically used, specifically user evaluations and trails. Visualizations that are well understood are used, thus eliminating the need for lengthy user evaluations concerning the effectiveness of the individual output visualizations. The guidelines stated above are used by the system to pick appropriate visualizations. These guidelines will be used to benchmark the effectiveness of visualizations produced by the system.

1.5 Overview

This chapter is followed by a survey of the State of the Art in the areas of automated visualization and the visualization process. chapter 3 examines the design of the visualization tool and gives an overview of the process of Information Visualization. chapter 4 gives a complete description of the projects implementation and any problems encountered. chapter 5 is an evaluation of the project and a discussion of its merits and failures. The extent to which such a system is viable is also discussed in this chapter. The paper concludes with a look at potential future work.

Chapter 2

State Of The Art

This chapter presents work currently being undertaken in the two main areas of research discussed in this paper; automated visualization and the visualization process. An analysis of these projects conclude this chapter.

2.1 Visualization Process

An understanding of how to create a visualization is key to this thesis. This process has been the subject of a good amount of discussion and seems to be fairly well understood but there are still discrepancies and differences of opinion. This thesis uses a model based off of Bertin's retinal variables (Bertin & Barbut, 1973). This is a cognitive model that ranks the effectives of various visual attributes, i.e. size, colour, at conveying various types of information, i.e. quantity, frequency etc. This section presents an overview of two works detailing this model. The model will be detailed in full in a later section.

2.1.1 Introduction to Information Visualization

Introduction to Information Visualization (Mazza, 2009) is one of the most recent texts providing a state of the art in the field of information visualization. As well as listing many examples of newer more novel visualization techniques in the later chapters it also contains a substantial amount of information about the process of information visualization. This set of first principles has influenced the thesis presented in this paper quite heavily. The author acknowledges that given an arbitrary set of data there is no way

to determine, automatically or otherwise, what visualization is a suitable representation of that data. There is in fact no agreed criteria to evaluate the effectiveness of a finished visualization.

However there are certain characteristics of data that can be used to help select an appropriate type of visualization. These steps are outlined as follows:

- Define the problem: Is the goal of the visualization to communicate some meaning in the data, allow the user to explore the data or confirm a hypothesis?
- Examine the nature of the data to represent: What type of data is being dealt with; ordinal, quantitative or categorical.
- Dimensionality of the data: The number of attributes of the dataset. In a univariate dataset one attribute varies with respect to another. Bivariate and trivariate extend this with multivariate meaning four or more dimensions.
- Structure of the data: This can be more easily understood by defining what types of data structures can be used to store the data; linear data structures such as vectors and tables, temporal data, spatial data hierarchies and networks.
- Type of interaction will the user have with the data: can the user transform the data by modifying it, can the modify parameters used to display the data or is the visualization static.

This process is summarized in Table 2.1 (Mazza, 2009). It will be revisited in later sections.

Problem	Data type	Dimension	Data Structures	Interaction
Communicative	Quantitative	Univariate	Linear	Static
Explore	Ordinal	Bivariate	Temporal	Transformable
Confirm	Categorical	Trivariate	Spatial	Manipulable
		Multivariate	Hierarchical	
			Network	

Table 2.1: Variables to consider when designing visual representations.

This process eliminates visualizations that are not suitable for a given data set. Deciding on an individual visualization to use is a decidedly harder

task. The effectiveness of a visualization is dependent on the perceptual capabilities of the viewer. As there is no empirically verified model of human perceptual abilities one is purposed. This model is based off the work of Jacques Bertin (Bertin & Barbut, 1973). Bertin was the first person to define a set of "retinal variables"; how different properties of graphics can be used to convey various types of information. His work has been extended since its original publication and the model presented here is perhaps one of the most up to date. A list of retinal variables is presented in order of effectiveness conveying different types of information. This idea will be revisited throughout this paper and a complete model will be detailed in a later chapter.

Mazza's work is a very complete introduction to information visualization and draws attention to all of the factors that need to be considered when creating a visualization. It is perhaps the most complete state of the art currently available and builds on a large amount of influential publications in the field. It has been very influential in the design of the system presented in this thesis.

2.1.2 The Structure of the Information Visualization Design Space

The Structure of the Information Visualization Design Space provides an overview of the field of data visualization as it was in 1997. An organization of information visualization literature is purposed and several examples are provided. The organization technique is then used as a framework for new designs. The paper builds on Bertins work (Bertin & Barbut, 1973) which has been expanded on by Mackinlay (Mackinlay, 1986). Visualization techniques are grouped based on similarities of their data to visualization mappings.

A series of steps that need to be preformed to create a visualization are given.

Data Data is defined as a set of values taken on my a set of variables. It is an obvious prerequisite of any visualization. Different types of data have their own characteristic operations and subcategories have yet more unique operations. For example patent text or a financial report have their own characteristics and unique operations. In order to do any useful processing

of data a more general model is used. Data is divided into three categories.

- Nominal, denoted by N
- Ordered, denoted by O
- Quantitative, denoted by Q

Data is further divided into the original dataset D and data D' that has been selected from this set and possibly transformed by some filter F.

Visualizations Visualizations are defined as being of composed marks, their graphical properties and elements requiring human controlled processing. Human visual processing works on two levels.

- Automatic Processing: Works on visual properties such as position and colour. These are Bertins retinal variables (Bertin & Barbut, 1973). This processing occurs at a subconscious level and as a result is very fast but limited in power.
- Controlled processing: This can be easily thought of as conscious thought. It is very powerful but limited in capacity. These limitations are fairly well understood (Miller, 1956).

Visual presentations consist of a set of marks (points, lines etc), a position in space (X and Y) and a set of retinal properties (colour and size). This distinction is a somewhat different grouping to the ones purposed by Bertin and Mackinlay. One can only assume the reason for this grouping is to limit the number of attributes or combination of attributes of a visualization. Connectedness and enclosure are also given as attributes.

These attributes are then arranged in a table to categorize various visualizations. An example is shown in Table 2.2. This example has been simplified to two dimensions.

Name	D	F	D'	X	Y	R	$--$ []	CP
Year	Q	$>$	Q	P				
Quality	Q	$>$	Q		P			
Type	N	$>$	N			C		

Table 2.2: Bar chart classifications. R, retinal properties. C, colour. $--$ Connection. [], Enclosure. CP, Control Processing.

Structure of the Information Visualization Design Space presents a good overview of the state of visualization research. The classification method is useful but is continuously amended to account for different types of visualizations.

2.2 Automated Visualization

The focus of this paper is automated visualization, taking a dataset and generating a visualization from it without any user input. There are a few projects that incorporate some kind of automated data visualizer. Generally such a visualizer is part of a larger project and requires user interaction to function. This chapter presents two such projects with varying degrees of automation.

2.2.1 Polaris

Polaris extends the well known pivot table interface to display information visually (Stolte et al., 2002). Multiple visualizations are displayed on a pivot table which the user can interact with by selecting or "brushing" datapoints to filter the displayed data. The visualizations act as interactive query builder that allow user to explore large datasets quite rapidly. First published in 2000 it has evolved into Tableau, a commercial visualizations product.

Polaris consists of two main components. A graphic generator and a database query generator. The database query generator will be omitted in this discussion as it is not relevant this thesis. The graphic generator makes several assumptions about the nature of the input data. Similar assumptions are made in this thesis.

- Data is characterized as either quantitative or ordinal.
- Nominal data is assigned an ordering and treated as ordinal.
- Nominal fields are dimensions, or independent variables.
- Quantitative fields are measures, or dependent variables.

Three different types of graphics can be generated based on the input data. User input is then used to refine this to a single generated graphic.

The user interacts with a relatively complex UI. The UI contains a number of "shelves" onto which the user places data-sources and data records. The user also selects what type of mark to associate with each record. The selected mark and nature of the data is used by the system to determine what type of visualization to render. For example, if the data-source contains an ordinal and quantitative field and the user selects a bar as the mark to use a bar chart is generated. If the user selects a dot as the mark a dot plot is generated. An understanding of Bertins retinal variables (Bertin & Barbut, 1973) is encoded into the system to ensure the visualizations are comprehensible. It is not clear how Polaris handles input that cannot be effectively displayed using these rules, e.g. when the input data set is too large to be displayed.

Polaris allows multiple series of data to be overlaid on the same visualization but it does not appear to allow more than two dimensions of data to be displayed at the same time. As it is a data exploration tool it requires user input too function. While it does contain an understanding of retinal variables these are used to automate the generation of mappings between data and graphical marks. It does not contain any rules governing how the various retinal variables are combined so it may be possible for a user to create ineffective or incomprehensible visualizations.

2.2.2 A Presentation Tool

A Presentation Tool, henceforth referred to as APT, is an application-independent presentation tool capable of automatically designing effective visualizations of relational information. It attempts to create a wide variety of designs and encode graphic design criteria in a useful form. It achieves this by defining graphical presentations as sentences in a graphical language. An expressiveness and effectiveness criteria are used to codify the graphic design criteria. The "sentences" produced by these two systems are then combined using artificial intelligence techniques to create designs.

APT is divided into three parts, expressiveness, effectiveness and composition. Expressiveness determines how the input information can be expressed in a graphical language. It is encoded into a formal language. This languages main purpose is to ensure that any graphic encodes all of the input data and only the input data. It is not used to select entire graphics, rather it is used to test individual graphical mappings, i.e. to a single axis

or type of mark.

Effectiveness is is used to determine which of the sentences generated by the expressiveness phase is most effective at displaying the data in a useful manner. It is dependent on the capabilities of the user. There is a difficulty in that no verified theory of human perceptual capabilities exists so one is purposed. It is based off Bertins retinal variables (Bertin & Barbut, 1973) which have been ranked in order of effectiveness by Cleveland and McGill (Cleveland & McGill, 1984). This ranking has been extended to include rankings of ordinal and nominal types of data although this extension has not been empirically verified (Card, Mackinlay, & Shneiderman, 1999).

APT generates multi-dimensional graphics by combining the effectively one dimensional or single data series sentences generated in the expressiveness and effectiveness stages. Designs are merged on common data. The types of data to merge on are ordered in terms of effectiveness.

- Mark composition.
- Double axis composition.
- Single axis composition.

APT combines these three stages using artificial intelligence techniques. The algorithm has three steps: partition, selection and composition. Each step contains various choices. If these choices do not lead to design backtracking is used to consider others.

- Partitioning: Each set of relations, or columns in the database, are partitioned to match the criteria of of one of the sentences defined in the expressiveness stage.
- Selection: A list of candidate designs for each partition is ordered in terms of effectiveness.
- Composition: The individual designs are composed into unified presentations of all the input data using the composition algebra.

Overall APT is a very robust and complex system. However it is perhaps somewhat dated. It is only capable of generating static graphics. It can also factor in the output media into the design choices, i.e. if the screen is monochrome colour is omitted for the effectiveness stage. The use of A.I.

techniques is a novel and somewhat unique approach. The parameters of APT are not made completely clear, it is not clear what the limits of its capabilities are.

2.3 Data Visualization Tools

Thus far a reasonably complete examination of various visualization systems and processes have been discussed. However in order to get a more complete sense of the state of the art of automated visualization a broader approach is needed. This section presents brief summaries of a number of visualization design tools. These tools are then used to create a feature matrix presented in Table 5.1.

Gretl Gretl is a software package for econometric analysis (Cottrell & Lucchetti, 2002). It provides a wide variety of financial calculations and ,notably , visualizations. It is a highly specialized application and generally expects data formatted in a certain way. The user must pick the type of chart to render. No further user input is needed. Admittedly very few intermediary processing steps are needed to generate a specific visualization given a pre-defined data format. However Gretl can handle some variation in this format so the process is not a straight mapping. The types of visualizations generated include line graphs, box blots, scatter plots and other graphics typical of econometric analysis. Gretl automatically ranges the dataset so the output visualizations are always comprehensible.

Microsoft Excel Microsoft Excel is a substantial data analysis tool that centers around a pivot table display (Inc., 2007). Users can visualize the data they are working with by selecting the relevant table entries and selecting a chart type. Excel then automatically generates the visualization. Some user training is required in order to produce effective and correct visualizations. It is very easy for a novice user to create nonsensical visualizations, or in the worst case misleading ones. An new version of the software attempts to remedy this by providing functionality to automatically recommend the most effective visualization. A number of visualizations are available but are generally limited to no more than three dimensions. Any exceptions expect data of a specific format, i.e. Candle stick graphs expect specific labels.

Google Charts API Google Charts API is a browser based charts library (Inc., 2012). It is an API aimed at developers and is not suitable for end users. User friendly interfaces do exist (Shevchuk, 2012) but it is debatable how easy they are to use without training. The API provides a wide number of visualizations. These visualizations also contain elements of interactivity, leveraging the features of modern browsers to do so. Although it is an API the software is not completely manual. Given an input dataset it is capable of automatically generating visual mappings. However it should again be noted that this process is greatly simplified by having the user select the output visualization to generate.

Tableau Tableau was born out of the research done by Mackinlay (Software, 2008), (Mackinlay, 1986). Like APT it appears to be a very robust and complex system. Interestingly it makes use of the visual shelving system presented by Polaris (Stolte et al., 2002). The user inputs a data source and then select the attributes to visualize. Again, as in Polaris, the data is divided into dimensions and measures. As the user selects attributes to visualize various types of applicable charts are made available. It is not made clear which one of the available visualizations is most effective so the user is left to decide. As it is a progression of the work presented by Mackinlay it is assumed an encoding retinal variables is present.

ManyEyes Many eyes is a web based collaborative visualization authoring tool (Viegas, Wattenberg, Van Ham, Kriss, & McKeon, 2007). Multiple users can work to create a visualization. It offers a wide variety of visualizations but leaves the task of selection completely to the user. If the system cannot generate the selected visualization it notifies the user. Aside from this the user can generate any visualization they see fit. Without an understanding of the various visualizations a user can easily generate ineffective graphics. While the focus of this tool is collaborative authoring it does contain features that automate various aspects of the visualization process. Data can be input in a variety of of formats and the system appears to do a good job of mapping them to appropriate visual features. The visualization engine is Java based.

OpenHeatMap OpenHeatMap is a web based geographic visualization tool (Warden, 2010). It allows users to overlay data across various maps. It is very easy to use. A dataset is uploaded and the visualization is presented. It is not clear what the formatting restrictions of the input data is but it appears to be relatively robust if occasionally fickle. The software is flash based and struggles noticeably with even moderately sized datasets. The user can zoom and pan the visualization interactively.

D3 D3 is a browser based charts library (Bostock, 2010). It offers much more control and flexibility then Google Charts API and as a result requires a substantially greater amount of developer effort. Whereas Google Charts offers complete visualizations D3 sits at a much lower level of abstraction. It is a tool set developers can use to create their own visualizations from scratch.

2.4 Analysis

Presently there is a good amount of literature surrounding individual visualization as demonstrated by (Mazza, 2009), and large degree of consensus on how to go about creating a visualization. However there have only been a few attempts to automate such a system. Such an application would not only allow users to quickly produce useful and accurate visualizations but would also server to validate the process that has become agreed upon. Such an evaluation is frequently omitted from these publications. Mazza outlines a set of criteria to evaluate individual visualizations but the focus of this thesis is testing if the entire process is valid.

Works such as Polaris and APT suggest that this process is indeed valid. However Polaris does rely on the user for some tasks. APT appears to be a very robust and complex system but is somewhat dated and cannot generate visualizations with any interactivity. Given the ubiquity of powerful computing platforms this seams to be quite a large deficiency. The project presented in this thesis aims to create an automated version of the visualization process. The interesting aspects and possible deficiencies of the various systems examined in this chapter are summarized in the table below. The terms used to evaluate each system are qualified below the table.

	Native Browser Based	Automatic Visualization	No User Training Required	Generic Application	Large Number of Visualizations	Interactive	Automatic Feature Determination	No User Input Required	Encoding of Retinal Variables
Gretl		x	x			x			
Excel				x	x		x		
Google Charts API	x			x	x	x	x		
Tableau		x	x	x	x	x	x		x
ManyEyes		x	x	x	x	x	x		
Open HeatMap		x	x			x		x	
D3	x			x	x	x			
Polaris		x	x	x	x	x	x		x
APT		x	x	x	x		x		x

Table 2.3: Summary of systems examined

Native Browser Based. The systems runs in a web-browser environment using native web technologies. These technologies, i.e. HTML5, SVG, EMCAScript etc, arguably compose the most widely adopted software platform in history. These technologies are described via specifications and implemented by multiple vendors. This approach has several benefits compared to older container based web-technologies such as Java or Flash.

Automatic Visualization. Given a dataset the system should be capable of automatically generating visualizations with no user input during the visualization process. Tasks such as selecting which entries to visualize or what visualization to output are not included as user tasks in this context, only the tasks involved in mapping data entries to graphical marks.

No User Training Required. This is somewhat hard to quantify meaningfully as it depends on the capabilities of each user. Low level technologies such as APIs are not intended for end users and require training to use. Complex applications with many features may also require user training. Any system that allows the user to generate incomprehensible visualization requires training to be used successfully.

Generic Application. A wide array of datasets can be input to generic systems. Such systems do not make assumptions about the structure of the input data, i.e. Each record must contain a data, certain number of variables etc.

Large Number Of Visualizations. The systems can output a wide array of visualizations. Of particular importance is the systems ability to output different types of visualization, i.e. Two dimensional, three dimensional etc.

Interactive. The system should provide some sort of interaction. This can be as simple as scroll over text or as complex as multiple nested visualizations than can be panned and enlarged.

Automatic Feature Determination. The system should be able to automatically determine the features of the dataset. These include attributes such as dimensionality and types of data. Systems with a limited number

of uses typically have this information hard coded in based on data labels and do not determine it at runtime.

No User Input Required. At no point should the user have input information into the system. This extends automatic visualization requirement by including such actions as selecting an output visualization.

Encoding of Retinal Variables. The work started by Bertin (Bertin & Barbut, 1973) has been incredibly influential. Most visualization systems and visualizations incorporate it to some degree. Furthermore Bertin's original work has evolved substantially and can be found under different names in many publications outlining the visualization process. For our purposes it is assumed that some knowledge of retinal variables are encoded in a system if there is some mention of them or if it is not possible for the user to generate visualizations that clearly violate the rules outlined by Bertin and others (Card et al., 1999), (Mazza, 2009), (Spence, 2001), (Cleveland & McGill, 1984).

This thesis will attempt to determine to what extent can suitable visualization be dynamically and automatically be generated from arbitrary data using browser based technologies. At no point should the user have to provide input. Browser based technologies have in recent years become very powerful platforms and are perhaps the most ubiquitous software platforms in existence.

2.5 Conclusion

This chapter introduced a state of the art of automated and manual visualization processes and how they can be used to simplify the creation of visualizations from arbitrary data. By making the process of creating a visualization easier more data can visualized and thus comprehended more readily. Many existing systems are limited by requiring human intervention. If the user is untrained they can introduce errors which may lead to confusing or even mis-representative visualizations.

Chapter 3

Design

This chapter introduces the design principles and core technologies used in this project. The architecture for the Automatic Classifier and Data Visualizer, henceforth referred to as AC↯DV, is presented. The visualization process is also described.

The project goals are taken detailed from the state of the art and are detailed in section 2.4. They are summarized below:

- Native browser based
- Automatic visualization
- No user training required
- Generic application
- Large number of visualizations
- Interactive
- Automatic feature determination
- No user input required
- Encoding of retinal variables

3.1 Technologies used

This section describes the technologies used in this project and the motivation for choosing said technologies. One of the main aims of the project was

to quickly generate visualizations. This extends to any system setup the user may have to perform before inputing data. Extendability was another key consideration. The system should allow new visualizations to be added with relative ease.

Browser based technologies have been selected for use in this project. The core technologies used are SVG and Javascript. Modern web-browsers allow for near seamless interoperability between these technologies. Such technologies offer many advantages over more traditional standalone executable binaries. If new visualizations need to be added they need only be appended to the existing software. They are then instantly available when the page is refreshed. The web browser is perhaps the most ubiquitous software platform in history making cross compatibility for applications almost a non-issue. The technologies used are freely available and exist as standards, not just implementations. This can lead to unexpected behavior in separate implementations but this is becoming less common as these standards and their implementations mature. As these technologies are not maintained by a single vendor they are far more resilient to an organization discontinuing their support. A short description of all the technologies used follows:

Javascript is a loosely typed interpreted programming language. It is defined in ECMA-262 standard. It was originally implemented as a client side scripting language. Javascript functions can be embedded or included in HTML pages and they can interact with the DOM. It is a dynamic language, a feature that was utilized extensively.

Document Object Model (DOM) is a convention used to represent and interact with objects in HTML and XML. The objects are stored in a tree structure that can be addressed and manipulated. This project creates graphics by generating and placing multiple SVG objects into the DOM using Javascript.

Scalable Vector Graphics is an XML based vector image standard created by the World Wide Web Consortium. SVG images support interaction and animation. The images and their behavior are are defined in XML which allows them to be manipulated and generated like any DOM object. The ability to generate interactive graphics at runtime makes them ideal for this

project.

HyperText Markup Language (HTML) is a markup language that allows the creation of web pages that can embed objects such as SVG images. AC↯DV uses HTML as a simple container to link together its various components. Google Chrome was used to render the HTML pages.

D3 is an open source Javascript library for manipulating DOM based on Data. It allows data to be bound to DOM elements and then apply data-driven transformations to those elements. The generation and manipulation of SVG elements is the focus of D3 but any element can be generated. A simple example would be generating a HTML table from an array of numbers. It greatly simplifies the modification of DOM compared to the W3C DOM API. The key difference is D3 uses a declarative approach instead of the W3C imperative approach. D3 was used extensively in the creation of AC↯DV.

Cascading Style Sheets (CSS) is a style sheet language that expresses the presentation of structured documents. The key benefit offered by CSS is the separation of the presentation and structure of DOM elements. It is not used extensively in AC↯DV.

Web Server. AC↯DV runs entirely on the clients computer. However most web browsers do not support cross-origin resource sharing from local files. Such requests must be sent using HTTP. This means that data stored in an external file can not be loaded into AC↯DV without the use of a web server. A very simple python server was used in this project but any HTTP server is sufficient.

One area of concern was performance. Standalone executables have greater access to a machines resources and can take advantages of optimization techniques such as multi-threading or specialized instruction sets such as Streaming SIMD Extensions (Inc., 2002). This is not usually a problem for browser-based applications but as AC↯DV may have to process thousands of records some effort has been made to evaluate the its performance and keep it within a reasonable bounds.

3.2 Overview of Data Visualization Process

A variety of different techniques have been proposed to created date visualizations. The technique proposed by Mazza (Mazza, 2009) is one of the few constructed as a sequence of complete steps. AC↯DV adopts this process, though does so using web-browser techniques. These steps do not vary based on the nature of the data or the desired output making the process one of the most suitable for automation. The process was outlined in Table 2.1 and will now be expanded on.

1. **Define the problem.** This step depends on the goal the user is trying to accomplish. It is also necessary to make note of the perceptual capabilities of the user as these can be used to inform decisions later in the process. There are three main problems data visualization can help to solve.

    - Communication: Visualizations can be used to quickly communicate information present in a data set. This is the most general task and can be thought of as presenting the entire dataset for easy comprehension by the user.
    - Exploration: Visualizations can be used to explore datasets. Exploratory analysis is often used to form new hypothesis that may not have been considered with more formal techniques (Tukey, 1977). Such visualizations summarize the main characteristics of the data set and then allow the user to filter the data and focus on subsets.
    - Confirmation: Here the data is being analyzed to confirm a specific hypothesis.

2. **Nature of the data.** The data must be classified into one of three types. This classification is later used to inform the visual mapping.

    - Quantitative or numeric data, e.g. real numbers.
    - Ordinal or non-numeric data which does have an intrinsic ordering, e.g. days of the week.
    - Categorical or non-numeric data with no ordering, e.g. names.

3. **Dimensionality.** The number of dimensions, or attributes, of a dataset determine what representation to use. These attributes can be dependent or independent. The dependent attributes are generally the ones of interest and are analyzed with respect to the independent attributes. The number of dependent attributes determine the dimensionality.

    - Univariate. One dependent attribute varies with respect to an independent variable. Example visualizations include Bar and pie charts.
    - Bivariate. Two dependent attributes vary with respect to an independent variable. Example visualizations include Histograms.
    - Trivariate. Three dependent attributes vary with respect to an independent variable. Example visualizations include Bubble charts and Tree Maps.
    - Multivariate. Four or more dependent attributes vary with respect to an independent variable. This is where visualization becomes much more difficult because people are not used to working in four or more dimensions. Example visualization include Parallel Coordinates (Inselberg & Dimsdale, 1991) and Scatter-plot Matrices (Elmqvist, Dragicevic, & Fekete, 2008).

    Generally multiple instances of the same type of visualization can be combined to create a visualization of greater dimensionality, i.e. Clustered bar charts can be used to show data with many dimensions. There are limits on the combination of visualizations. These limitations vary depending on the specific type of visualization and will be addressed in Section 3.3.

4. **Data structures.** These are used to contain the data and are derived from the nature of said data.

    - Linear.
    - Temporal.
    - Spatial.
    - Hierarchical.
    - Network.

This is somewhat difficult to automate completely and generally involves pattern matching the data against known templates. For example there are a myriad of ways to represent a valid date.

5. **Type of interaction.** This depends largely on the initial needs of the user but is also decided by factors such as the scale of the dataset (large set may necessitate interaction as they cannot be displayed completely) and the medium used. A visualization can be:

   - Static. Not modifiable.
   - Transformable. The user can modify and transform the data by filtering entries, choosing different visual mappings etc.
   - Manipulable. The user manipulate the view by zooming or rotating the image.

Each of the options described here help to point a specific visualization technique. AC↯DV uses a simplified version of this process.

3.3 Visual Mapping

Visual Mapping is the most important aspect of creating a visualization. If the elements of a dataset are mapped to visual elements effectively any patterns in that dataset are easily perceivable. A poor mapping results in these patterns becoming almost imperceptible. Creating an effective mapping is done by understanding and exploiting the way humans perceive what they see. Certain visual properties will more readily draw focus. There is a model detailing the effectiveness of various visual properties although it is not concrete. This model of human perception began with the work of Bertin (Bertin & Barbut, 1973) and has since been expanded (Card et al., 1999) (Few, 2004).

3.3.1 Memory

Human perception is composed of human vision and human memory. What we see is stored in memory and then processed. There are three basic types of memory:

- **Sensory Memory.** All input from sense organs is stored in memory for a very short time. Such memory is independent of conscious

control and is processed automatically. For this reason the processing that takes place is called *preattentive processing*. During preattentive processing only a limited set of basic features are considered. These are the *retinal variables* and includ attributes such as colour, size and position. An understanding of these attributes and how they can be used to convey different types of information is absolutely key to creating an effective visualization. Pre-attentive processing is extremely fast and allows a detailed mental picture of a scene to be built very quickly.

- **Short-term memory.** Short term memory is where some of the information from sensory memory is transfered. Information can persists here for up to a minute but the capacity is extremely limited compared to sensory memory. The general rule is that 5 to 9 items can be stored in short term memory although this depends on the type of information being stored (Miller, 1956).

- **Long-term memory.** Items in short-term memory can be committed to long term memory by repeated rehearsal.

3.3.2 Retinal Variables

A number of pre-attentive visual properties have been identified by research. These visual properties have been grouped in order of effectiveness at conveying different types of information although such groupings do not seem to have been completely verified (Mackinlay, 1986), (Mazza, 2009).

Ths visual properties can be grouped into four categories: *colour, form, movement and position.*

Colours can be expressed in different ways. Red, Green and Blue values are typically used when expressing the colours on a computer screen. For the purpose of data visualization it is more efficient to talk about colour in terms Hue, Saturation and Lightness as they map to preventative attributes far more readily. Hue is the aspect of colour we typically refer to by name and is usefully for labeling different sets. Saturation and Lightness describe the intensity of the Hue. A colour with a high intensity is more distinguishable from its surroundings so intensity is used to give certain elements a kind of visual precedence.

Motion is the most effective way for an attribute to gain focus. It should be used sparingly else the visualization quickly descends into a disorienting mass of flickering objects.

Position is the most accurate attribute for encoding quantitative data in graphics.

Form is the most comprehensive of the visual attributes. There are many attributes that make up the form of an object, all of which are capable of representing various types of data with various levels of effectiveness.

- Orientation
- Length
- Width
- Size
- Collinearity
- Curvature
- Spatial Grouping
- Added Marks
- Shape
- Numerosity

There are a number of different rules governing the use of these attributes. These rules are primarily concerned with the number of attributes to use and which attributes are the most suitable to use for different types of data.

The number of different pre-attentive attributes to use in a visualization is quite important. If it is too great an incomprehensible visualization is produced. If the number is too low then the full capabilities of the medium have not been exploited. The latter is the more desirable so a conservative approach is best adopted. Millers work on human information processing capacity (Miller, 1956) is the most widely known in this area but does not deal with specifically with preventative loading. Similar studies specifically

with visualization in mind have also been conducted (Few, 2004), (Ware, 2012). These studies have not come to the same conclusions so the limits they purpose are not hard rules. Ware suggests no more than ten distinct values for pre-attentive attributes, although there are several exceptions, while Few limits this number to four.

The combination of attributes is a far harder problem and has not been the subject of much research. However different attributes are better than others at conveying different types of information. This has been the subject of a relatively large amount of work. Cleveland and McGill purposed and verified a ranking of of attributes in terms of their ability to convey quantitative information. Mackinlay expanded, although he did not verify, this ranking to include ordinal and categorical information (Mackinlay, 1986). More recent research has shown that the issue is more complicated (Spence, 2001) so a definitive raking does not exists. Mazza extends the work done by Few (Few, 2004) to purpose an "indicative rule of thumb" for the three data types (Mazza, 2009). Mazzas work is a simplified ranking that groups the pre-attentive attributes into one of three categories for each data type: suitable, limited suitability and not suitable. It is shown in Table 3.1.

Attribute	Quantitative	Ordinal	Categorical
Hue	×	×	✓
Intensity	−	✓	×
Orientation	−	−	×
Length	✓	−	×
Width	−	−	×
Size	−	−	×
Collinearity	×	×	×
Curvature	−	−	×
Spatial Grouping	×	×	×
Added Marks	×	×	✓
Shape	×	×	✓
Numerosity	✓	✓	×
Position	✓	✓	−
Flicker	×	×	−
Motion	−	−	×

Table 3.1: Encoding the three data types with pre-attentive attributes (Mazza, 2009). ✓ indicates the attribute is suitable for the given data type. − indicates the attribute has limited suitability. × indicates the attribute is not suitable.

Short term memory and pre-attentive processing play an extremely important role in the design of effective visualizations. An understanding of these pre-attentive attributes can be used to make the most important information draw the focus of the viewer. If the rules surrounding these are ignored it can result in incomprehensible visualizations.

3.4 Architecture

AC↯DV consists of two main components. A chart API and a classifier. Data is input to the classifier which determines an appropriate visualization and displays it in the view. These two components are distinct. The chart API can be used completely independently. If a user does not wish to use AC↯DV as an automated tool they can use the visualizations directly. Similar chart APIs do exist (Zhu, 2012) but are relatively complex to use, usually requiring large amount of arguments or specific setup steps before they can be used. The goal of AC↯DV was to create a chart API that used a minimum of parameters. Simplifying the the interaction between the two main components in this way reduces development effort, reduces coupling and promotes re-usability.

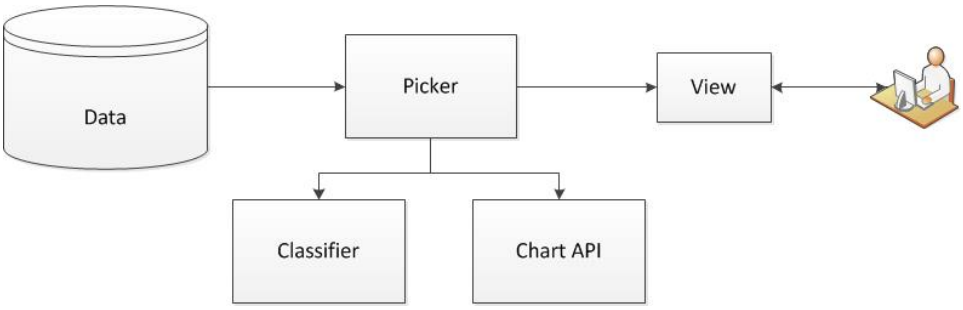

Figure 3.1: System Architecture

The system has been designed in this way to reflect the visualization process described above. The classifier encodes a paired down version of the process outlined by Mazza detailed in section 3.2. This approach also has the benefit of yielding a chart API that can be used by developers who wish to easily create a specific type of visualization.

3.4.1 Classifier

As mentioned the classifier based on the process outlined by Mazza. It is summarized in Table 3.2.

Problem	Data type	Dimension	Data Structures	Interaction
Communicative	Quantitative	Univariate	Linear	Static
Explore	Ordinal	Bivariate	Temporal	Transformable
Confirm	Categorical	Trivariate	Spatial	Manipulable
		Multivariate	Hierarchical	
			Network	

Table 3.2: Variables to consider when designing visual representations. Grayed out entries have been omitted from AC↯DV.

The first and last stages of the pipeline are fixed as they depend on the needs of the user. The most generic choices have instead been selected. A communicative visualization gives an overview of the dataset while a manipulable view allows the system to create an interactive visualization. AC↯DV is not a data processing tool so any transformations on the dataset must be done using other tools. The middle three stages of the pipeline have been the focus of the project.

AC↯DV reduces the categorization of data to ordinal and quantitative by assigning an implicit ordering to all nominal fields and subsequently treating them as ordinal. The assigned ordering is not important, indeed with many visualizations preserving the ordering of ordinal variables is not important for the final output. All nominal fields are treated as independent variables and quantitative fields treated as dependent variables. These are the same assumptions made by Polaris (Stolte et al., 2002).

The current implementation of AC↯DV is limited to producing three visualizations with three or less dimensions. There is no technical reason for this limitation. The classification engine is designed to be extended easily.

AC↯DV uses a single data structure for all processing. The system can test if a quantitative variable represents a date, hence the inclusion of temporal data structures. The design could be extended to detect the remaining types of data as such a task can be accomplished, to a high degree of reliability, with pattern matching, but this was out of scope.

The classifier determines these attributes at run time from the dataset. Each visualization in the API has a similar set of properties. The properties

are:

- Size: the number of entries in the dataset.
- Dimensionality: the number of dependent variables in the dataset.
- Keys: the variable names. These are used to label the data.
- Data types: The type of each element.

The system then attempts to match the characteristics of the dataset to the characteristics of a visualization. AC↯DV returns a confidence score along with the visualization to give an indication of how suitable the visualization is. This score is determined from how closely the two sets of characteristics match. Some characteristics must match for the visualization to be considered such as the dimensionality and types of information to be displayed, i.e. if no chart can display two independent variables then no chart will be returned.

The encoding of the visualization properties is a manual step that must be done by an expert. It is here that the retinal variables are considered. The mapping from data to marks is encoded into each visualization and it is up to the expert to decided how to do so. As mentioned there are no hard rules to go by but a conservative approach has been adopted. The chart API consists of a number of well known visualizations so there is no need for a detailed description of the effectiveness of each one. The number of retinal attributes used to encode information is always three or less. The retinal variables used have been selected using Table 3.1.

The system also contains a notion of "soft" and "hard" size for each visualization. These are the limits on how many records may be displayed in each visualization. Like the mapping to graphical marks and their associated retinal variables this notion of size is fixed. Deciding on the maximum number of entries for each visualization is quite tricky. It depends on

- The pre-attentive attributes used in the visualization.
- The number of items the user can commit to short term memory, which again is somewhat dependent on the pre-attentive attributes used (Miller, 1956).

- The task of the user. If they are investigating specific entries they will have to memorize them while the compare them to other entries. If they want an overview of an patterns in the data retaining specific entries is less important.

The last item is attempted with the notion of "soft" and "hard" size. If the dataset is within the soft size limit then every mark will be unique. If it is within the "hard" limit AC↯DV will modify the output visualization. An example is the bar chart. Many unique entries quickly become hard to distinguish but if there are sufficient entries (between the two size limits) the system considers them part of a single series. The colour for every bar is then set to the same value and the visualization resembles a time series plot. However this may not always produce the desired result so exceeding the soft size limit is reflected in the confidence score. The automatic classification of chart properties is a very hard problem and was abandoned for a simpler solution that would not compromise the end goal of this project.

3.4.2 API

The API consists of a collection of visualizations with shared declaration attributes. The design is based of off Mike Bostock's, the creator of D3, convention for reusable charts (Bostock, 2012). All visualization are designed as Javascript closures. Each chart only expects input data and a mapping of each data dimension to visualization "axis" (the notion of axis can be unique to each visualization) , which the classification engine can provide automatically. Attributes such as size and position can be adjusted after declaration if desired. Charts could be extended to included customizable colours and marks like similar libraries (Zhu, 2012). However such customization would allow interference with the decisions made in encoding the various retinal attributes.

3.4.3 Picker

The Picker is the glue logic used to combine the classification component and Chart API. It is responsible for creating DOM elements to bind the visualizations too, parsing input data and taking the results of the classification component and invoking the relevant visualizations in the Chart API.

3.5 Use Cases

Two use cases have been created to illustrate the operation of AC↯DV. They are introduced here briefly and described extensively in section 5.3. Their purpose is aid the evaluation of AC↯DV by providing a walk through of how the system may be used.

As with any system AC↯DV has stakeholders. The users input data and use the output visualizations for various tasks. Experts are responsible for adding new visualizations and encoding their properties. As it is an automated system the user does not need the help of an expert directly.

The first use case details the steps taken by an employee of a coffee shop chain tasked with cutting expenses. AC↯DV is used to quickly visualize data and investigate trends across multiple datasets. Here the visualizations are being used to form hypothesis and test them.

The second use case details how AC↯DV might be used in social planning. In this example the system is used to provide illustrations for statistics and clearly show the correlations between data.

3.6 Conclusion

This chapter has introduced and discussed aspects relating to the design of this project. The technologies used and the architecture of the system were presented and discussed. The visualization principles outlined were key to this discussion. The design of the system is broken into two sections.

Chapter 4

Implementation

The implementation of AC↯DV followed the design quite closely. For this reason the implementation of the classification system will be described as the respective components are triggered during an example execution flow. The implementation process and any issues encountered will be described in the course of this description. This is followed by a description of the Chart API.

4.1 Classification Component

Although the Chart API required the bulk of the implementation effort the classification component contains the application logic.

4.1.1 Data input

AC↯DV is begins with data input. Two data formats are supported, comma separated value files, or CSV (Shafranovich, 2005), and json (Crockford, 2006). Input is parsed using D3. An example call is shown below. The "pickChart" call executes AC↯DV. It requires no parameters other than the input data file. Such a small code footprint makes it relatively easy construct a user interface around AC↯DV. A simple example that allows users to load in and compare multiple datasets was created to illustrate the use cases presented in section 5.3.

```
d3.csv("data/edu.csv", pickChart)
```

The D3 parser returns a common data format for all types of input. These data object contain a number of name value pairs. Each object corresponds to a single record in the dataset. The object contains a number of name value pairs. Json files are organized in this way but for CSV files it is assumed that the first line is the header. This header is mandatory, not optional as defined in the CSV RFC (Shafranovich, 2005). An example entry from a trivariate dataset is shown below. The quoted entries are taken from the elements of the data set. The key values associated with them are taken from the first row in the case of CSV.

```
Third: "50"
Name: "CLONMETHAN"
Population 2006: "625"
Unemployed 2006: "19"
```

One small problem was presented by this data format. It is designed for deterministic access. The whole idea behind AC↯DV was to build a generic data visualizer so the data needs to be modified to allow non-deterministic, i.e. array style, access. The final implementation of the classification component simply extracts the keys at runtime and uses them to access the data like so: $data[keys[1]]$.

4.1.2 Classifier

The parsed data is then feed into the classification component. Classification is only performed on the first data object, or the first entry of the dataset. This is done to speed up system execution. It is assumed that all data entries share the same properties. This assumption is valid when working with relational data.

The first object is then classified and the properties returned are associated with the dataset. These properties are:

- Size: the number of entries in the dataset.
- Dimensionality: the number of variables in the dataset minus 1. It is assumed all input datasets have a single independent variable whether it be quantitative, ordinal or categorical.

- Keys: the variable names. These are used to label the data when it is displayed in the view. They are also used by the system to access the data during processing.

- Data types: The type of each element is determined.

Type determination involved a number of challenges. Numbers are classified as quantitative variables and all other input is classified as ordinal. Dates are an exception. While a date is typically represented by an alphanumeric character string it can be thought of as both an ordinal label or a quantity. The internal representation of a date is a number so dates are classified as quantitative variables by AC↯DV.

Determining if a string represents a number in Javascript is somewhat tricky. The language is loosely typed and generally seems to be somewhat sloppy when determining type information. An elegant solution was ultimately provided by a language expert (Salvad, 2009);

```
!isNaN(parseFloat(n)) && isFinite(n);
```

These properties are associated with the dataset and used to by the visualization selection stage.

4.1.3 Visualization Selection

The properties created by the classifier are then used to select a visualization. Each visualization in the Chart API has a set of properties associated with it. These properties are not stored inside the visualization code. While this results in more work to add a new visualization it reinforces the Chart API as a separate component. The Chart API is only exists to visualize input data. It has no knowledge of the classification system.

The same data structure is used to store data properties returned from the classifier and visualization properties. The visualization properties are extended to include a second "soft" size limit. The visualization properties are stored in a hashmap along with the function used to invoke them and indexed by their name.

The data properties and visualization properties are then compared to find the most suitable visualizations. The comparison consists of a number of tests:

- Dimensions; If the visualization and data are of different dimensionality the visualization is not suitable and the next set of visualization properties are tested.

- Data Types; The variable types supported by the visualization and present in the dataset are compared. If they do not match the next set of visualization properties are tested. The variables do not have to be in the same order. AC↯DV allows quantitative variables to be used as ordinal labels.

- Size; The size of the dataset is compared to the number of elements the visualization can display. Visualizations have a "hard" and "soft" size. Hard size is an absolute limit, if the data set contains more elements than the hard size the next set of visualization properties are tested. If a data set contains more entires than the soft size limit the visualization is of limited suitability. A visualization may change certain attributes of the elements displayed if the soft size limit is exceeded.

The suitable visualizations are returned along with a confidence score. Visualizations can be suitable, have limited suitability or not suitable.

Javascripts loose typing system was used extensively throughout the implementation. The properties associated with a visualization are stored along side a pointer to the visualization. Complete functions are returned from functions and used directly to continue execution flow. Javascripts loose typing and dynamic behavior can be at times tricky to work with and often produces unexpected behavior. If used effectively Javascript code can be quite concise and efficient.

4.1.4 Rendering

The selected visualizations are sorted in order of the confidence scores. A "div" element is created to bind the visualization to. If the classification engine determined an element in the dataset represents a date a date formatter is constructed. The dataset is then passed into each suitable visualization from the Chart API. Relevant entries are converted to dates. Every visualization in the Chart API requires each variable to be passed into it explicitly resulting in a different number of parameters depending on the

dimensionality of the visualization. The result is that univariate, bivariate and trivariate visualizations require separate sections of code to invoke them. This is a rather ugly compromise and is due to the design of the Chart API.

4.2 Chart API

The Chart API uses D3 extensively. The creator of D3 has purposed a set of guidelines for creating reusable visualizations (Bostock, 2012). These guidelines influenced the design of the Chart API heavily. They can be summarized as "implement visualizations as closures with getter-setter methods".

Closures are a tricky concept to grasp. They can be summarized as an expression, usually a function, that can have free variables together with an environment that those variables. There are several advantages and drawbacks to using them. AC↯DV uses them primarily for two reasons; they allow visualizations to be encapsulated easily and improve performance by reducing the cost associated with instantiating objects.

Every visualization is initialized the same way:

```
d3.select("#exampleChart")
  .datum(data)
  .call(example_chart
  .label(function(d) { return d.Area; } )
  .amount(function(d) { return +d.Amount; } ));
```

A "selection" in D3 is an array of DOM elements. The "div" argument is the DOM element to bind the visualization too. Data is bound to this element and a visualization is called. Every visualization expects an independent variable to be passed in as a label and a number of quantitative variables passed in as amounts.

This string of methods calls will look unusual to those not familiar declarative programming. Every call returns an object used by the next call. So "select" returns a selection, in this case a new div element identified by "exampleChart". The "datum" call is a setter for the selection object which then returns that selection object. The "call" function executes an arbitrary function, in this case "barchart" which expects a selection as an argument. Setter methods "label" and "amount" are then called. Functions are passed as argument which define how the data is to be accessed.

Every visualization in the Chart API shares a common structure and interface. A number of properties defining attributes such as size, margins, axis and any D3 layout functions are defined in the outer scope. This means this information does not have to be re-initialized every time the visualization is invoked. The visualization logic is contained in the inner scope as a "chart" method.

This "chart" method follows the same model in all visualizations.

- The data is transformed to allow non-deterministic access. This transformation is applied to all entries as they will all be accessed.
- An svg contain is created. Various attributes such as width and height are set.
- A group to contain the elements of the visualization is created. By grouping the elements they can be easily modified or searched as a complete set. It also simplifies the outputted DOM.
- The data is bound to the element group. D3 groups the data into two structures. The "enter" group contains data elements with no corresponding data elements. The "exit" group contains graphical elements with no corresponding data or data elements that have been filtered out.
- The elements of the enter group are assigned graphical properties.

```
var g = svg.selectAll(".dot")
    .data(data)
  .enter()
    .append("circle")
    .attr("class", "dot")
    .attr("id", function(d) { return d[0]; })
    .attr("r", 3.5)
    .attr("fill", function(d) { return fill(d[0]); })
    .attr("cx", function(d) { return xScale(d[1]); })
    .attr("cy", function(d) { return yScale(d[2]); });
```

- Axis and legends, if needed, are created and labeled.
- Mouse over behavior is defined.

By modifying only the attributes that actually change, D3 reduces overhead and allows greater graphical complexity at high frame rates. The Chart API contains the following visualizations:

- Pie-chart
- Bar-chart
- Time series plot
- Scatter plot
- Bubble plot

4.3 Conclusion

This section provided details of the implementation of AC↯DV. The implementation process and any issues encountered will were described in the course of this description. This was followed by a description of the Chart API.

Chapter 5

Evaluation and Discussion

This chapter describes the evaluation of AC↯DV. Visualizations are usually evaluated using empirical methods, e.g. user trials. AC↯DV is not an end user tool so such evaluations would not be relevant. Instead an analytic approach is adopted. The functionality of the AC↯DV is assessed with respect to the project goals and research question.

The objective of this project was to determine to what extent can suitable visualizations be dynamically and automatically be generated using browser based technologies. This question was posed with a number of goals for the project.

- Design and develop software capable of accepting arbitrary data in a specific format and display it using suitable visualizations. This should be done without an intervention from the user.
- Access the level of benefit that the visualizations can bring to potential users from various fields.
- Investigate to what extent visualizations can be generated given only an input dataset.
- Elaborate on the potential of a more sophisticated version of the software using additional techniques to determine features of input datasets.

The first two goals are examined at length below in Section **??**. The extent to which visualizations can be generated given only an input dataset

is addressed in section 5.4 along with the research question. The final goal is addressed in Section 6.

5.1 Evaluation of Functional Requirements

Dix et al. outline evaluation criteria for a system in HCI (Dix, 2004) which is elaborated by Mazza (Mazza, 2009). The process presented here is a modified version which aims to assess only the technical aspects of a system. The evaluation is divided into two parts.

- Functionality of the system. Does it achieve the intended goal.
- Effectiveness of the system. Does the system generate visualizations that provide potential users with better knowledge of the data.

5.2 Functionality

Overall AC↯DV meets the requirements set out in Table 5.1. A brief discussion of each requirement is given below:

Native Browser Based. AC↯DV runs entirely in a web browser. As noted in the state of the art the technologies used exist as standards, not implementations. While this offers many benefits it does make metering performance somewhat tricky. Older browsers may not support newer version of certain web technologies and different implementations will preform differently. This was not of much significance to the project. As long as standards are used they will be adopted by all platforms in time. AC↯DV does not use any particularly new or novel features of the technologies used. The system was developed using Chromium version 18. It was also tested in Firefox Version 11 as well as the latest version at the time of writing, version 20. Version 11 was released over a year ago. None of these platform had exhibited any problems and no errors were logged. Performance in Chromium was benchmarked ten times using a dataset of 5000 entries and the system returned a result, on average, in 4 ms. For comparison the time required by the browser to fetch the system code averaged 10 ms and when third party libraries are accounted for the time averaged at 55 ms. The times listed here will of course vary depending on different circumstances. It is the ratio

that is important. Although this was only a small test it shows quite clearly that web technologies in their current state are more than capable of handling large datasets. The system was used extensively during development and at no point was there any appreciable delays in either page loading or interaction.

Automatic Visualization. AC↯DV is completely automated. The only input required is the data file. As the system was not developed to be used off the shelf by an end user how the file is loaded is not of great importance. It is not sufficient to abandon the topic here though. If AC↯DV is to be used as part of a more complete system intended for an end user it should be easy to integrate into said system. To this end the system is successful. A developer need only provide an element for the chart to attach to and call a single function. This is absolute minimum needed to use AC↯DV.

```
<div id="chart_area"></div>
<script>
    d3.csv("data/edu.csv", pickChart)
</script>
```

If desired the Chart API can be used in isolation as shown in section 4.2. Such a small footprint allows more complex user interaction to be added with relative ease. It was envisaged that AC↯DV would be a component in a larger system. A small mock system was created that allowed users to compose a web page containing various visualizations from predetermined data. This was done with a minimum of effort and the results are quite useful.

No user training required. This objective is hard to qualify as it depends on the capabilities of the user. Some users may require no training where others may struggle without instruction. By any measure AC↯DV meets this requirement. As stated in the requirements this goal pertains to mapping of data entries to graphical marks. AC↯DV requires no user input at any stage in this process so no user training is required. The main benefit of this approach is it allows the retinal variables to be suitably encoded in every output visualization, thus ensuring comprehensible visualizations.

This is done at the cost of users customization. This may not be ideal for expert users who understand retinal variables and can ensure their correct usage. The guidelines for the usage of retinal variables are just that, guidelines. There are situations where these guidelines can be ignored without compromising, or even perhaps enhancing, the comprehensibility of the output.

These guidelines were drawn up with the most common cases in that mind so it follows that by using them the most common visualization tasks are catered for.

Generic Application. AC↯DV makes no assumptions about the data other than its basic organization in a file. Currently comma separated value files and json are supported. Support for different file formats was not a goal of the project but json was included as an exercise to determine the difficulty of including new formats. Some minor changes had to be made to accommodate jsons object oriented description of data but these changes were generic and do not interfere with record based processing. They were the result of flawed assumptions made about the output of D3s data parser. AC↯DV does not distinguish between the two data formats explicitly. D3 is used to parse input data files into a common format. It is this common data format that is passed into AC↯DV. The data format must be discriminated at this stage, not during AC↯DVs subsequent operation.

After the file has been parsed and input into AC↯DV no additional assumptions are made about the data. There are however limitations that were outlined in Table 3.2. Only three dimensional datasets or less or supported. This was not due to any technical limitations. The Chart API simply does not contain any multivariate visualizations as there was insufficient time to create any. It was observed during research for this project that multivariate visualizations typically accommodate a much smaller range of datasets than more traditional statistical graphics. The system is intended to visualize the widest range of possible datasets so effort was concentrated on visualizations with the greatest applicability.

As stated a single internal data structure is used to store and process data. Temporal data structures were included to illustrate how different types of data may be differentiated. This is done it two ways. The data is searched for any labels containing the string “date” and the first set of data

entires are pattern matched against known date formats. These methods are very simple and do not guarantee a date will always be correctly identified. They merely serve to illustrate that different data types may be distinguished through pattern matching. The creation of a robust pattern matcher was not a goal of this project.

There is only one visualization in the Chart API that explicitly expects a date entry, a time series plot. The remaining charts can display a date as an ordinal label. The same is true of any spatial or hierarchical data. The Chart API only contains visualizations, with the exception of the time series graph, that expect linear data. Had more specialized visualizations been included, i.e treemaps or geographical maps, a component to identify other data structures would be necessary. In the current implementation it is simply not required.

All numbers are classified as quantitative variables and treated as dependent variables by AC↯DV. All other types of data are assumed to be ordinal and assigned an ordering by the system and treated as independent variables. Polaris makes the same assumption (Stolte et al., 2002). The assumption is that the quantitative data is the subject of analysis and the ordinal data are labels.

These bounds on the input dataset are not particularly strict. As a result AC↯DV can accept a wide array of input, which yields generic applicability.

Large number of visualizations. The Chart API contains five completed and working visualizations. This is not a large amount but there are visualizations capable of displaying one, two and three dimensional data. A generic visualization system was one of the aims of this project so rather than providing many visualizations for one dimension focus was directed at creating a wider range. The design of the system allows new visualizations to be easily added to the Chart API. Properties for these visualizations must be encoded in the classifier. No further work is needed for them to be used by AC↯DV.

The classifier must be extended if visualizations of higher dimensions are to be added. This is due to technical limitations. Each visualization in the Chart API must have each dimension of data passed into in explicitly. A simplified example, illustrating a function call to create a scatterplot is shown below.

```
d3.select(div)
  .datum(data)
   .call(charts.scatterplot()
    .label(function(d) { return column_1) })
    .amount(function(d) { return column_2; })
    .amount2(function(d) { return column_3; }) )
```

As each variable in the dataset must be explicitly passed through to each chart the classification component of AC↯DV must have separate sections to invoke visualizations of each dimension. However the steps for adding a new dimension are not particularly difficult, merely requiring the addition of a parameter.

Interactive AC↯DV supports basic interaction. All charts allow for individual entries to be selected and inspected. When the mouse is moved over a graphical mark all of the information relating to that mark is displayed on screen.

It is assumed that the data input into the system has been filtered by external tools. For this reason AC↯DV does not support transformable interaction, it is not intended as a data filtering and manipulation tool.

The view manipulation supported by the system allows multiple records to be displayed on screen even when there is insufficient room to label them. This allows the user to observe overall trends in the data without significantly compromising their ability to further analyse these trends by investigating the individual entries. Miller states that a user can remember around seven entries at once in short term memory (Miller, 1956). Such a limited capacity would suggest that it is more beneficial to immediately access all of the information associated with record rather than having to refocus attention on various labels and axis to obtain this information. This is one of major benefits offered by interactive visualizations.

The visualizations included in the Chart API were originally created during the 19th century (Friendly & Denis, 2001). They were not created with interaction in mind. Beyond the inspection of individual elements there is not much scope for interaction with such visualizations. As each visualization is self-contained interaction is done on a case by case basis. While this means there is no standard framework to for interaction it allows for more advance forms of interaction to be added simply by including new

charts.

The ability to view individual details is however not to be underestimated. Shneiderman coined the visual information seeking mantra "Overview first, zoom and filter, then details-on-demand" (Shneiderman, 1996). This idea has taken root and is widely cited. Although AC↯DV does not support filtering, as it is not a data manipulation tool, the goal of the system is to provide and overview visualization. The interaction described provides the "details-on-demand". Shneirderman argues that this principle is key to creating an effective visualization. By emptying the screen of visual clutter the user is able to quickly build an internal representation of the data more easily. If they desire individual records can then be investigated after such a representation has been constructed. Investigating individual entires before this takes place is as tedious as viewing the raw dataset and attempting to extract meaning by reading each entry.

Automatic Feature Determination. AC↯DVs classification component is responsible for determining the features of a dataset. The features used to classify a dataset are taken from the visualization process defined by Mazza and outlined in Section 3.2. These features, or properties, are determined automatically by the classifier.

The variable names in a dataset can be used to inform the classification process. As stated above in the generic application requirement AC↯DV illustrates this by assuming variables labeled with the string "date" represent timestamps. However AC↯DV does not depend on this assumption and also pattern matches for dates. Systems that require certain information to be encoded within the input dataset are typically quite limited to specific applications. Omission of specific labels renders the unable to create visualizations. Using labels to inform the visualization process results in a more robust system. If no labels are found then pattern matching can be performed. If pattern matching fails then all variables will still be classified as quantitative or ordinal. This still provides the system with enough information to produce useful visualizations. All datasets are assigned a minimum number of features:

- Data types
- Dimensions

- Size

These properties are determined from analysis of the first record in a dataset, not the complete dataset. It is presumed that the first data record in a dataset is representative of the dataset as a whole. This assumption is perfectly valid when dealing with relational data.

Every variable in a dataset is classified as an ordinal or quantitative data type. If the entries are numeric they are classified as quantitative, otherwise they are classified as ordinal. Without additional information it is not possible to infer if a variable is intended as a data point or a label. Classifying all numbers as quantities, or data points, and all other strings as labels is the only reasonable way to infer type information from raw data. However, if a dataset does not contain any ordinal variables a quantitative variable can used instead. This may not be ideal in all cases but ensures a visualization will be generated. Furthermore data without any labels is meaningless. If one of the quantitative variables is not intended as a label there is no visualization that will help the user understand the data. It will merely be a meaningless table of numbers.

As Javascript is loosely typed determining if a string represents a number can be somewhat difficult. An elegant solution was provided by a language expert.

Dimensionality is determined by the number of records in a dataset. Dimensionality is defined as the number of dependent variables with respect to the number of independent variables as discussed in section 3.2. All visualizations in AC↯DV contain only one ordinal field. This was observed to be the case for most visualizations in general. This means dimensionality can be determined merely from the number of fields in a dataset.

AC↯DV extends Mazza's process somewhat by adding size as a feature. This is determined simply from the number of records in the dataset.

No user input required. This is an extension of the automatic visualization requirement. Automatic visualization only applies to the mapping of data entries to graphical marks. No user input applies to the complete end-to-end process. After data is input AC↯DV requires no additional input from the user. No user input was one of the main goals in designing AC↯DV. Not many systems are described as fully automated visualizations systems so they do not attempt to meet this requirement. In order for AC↯DV to

be part of an end-user system some user input would be required if only to select what data entries are to visualized. To this end a fully automated system is not possible. However, as acknowledged in the automatic visualization requirement, such a task is not part of the visualization process and crucially provides little opportunity for the introduction of user error.

Many systems allow the user to select the data to visualize and then automatically produce visualizations. Such systems typically require the user to also select the output visualization. This is where user error can be introduced. The robustness of the visualization component of a system is irrelevant if a user selects an inappropriate visualization.

AC↯DV appears to be somewhat unique in this respect. By automatically selecting the most appropriate type of visualization user error is eliminated. With further work it could be ensured that AC↯DV will always output appropriate visualizations. At present there are limitations on the range of input as discussed in above requirement evaluations.

Encoding of retinal variables. As described in the design chapter the retinal properties of the visualizations in the Chart API are not determined automatically. Rather they must be described by an expert user when encoding the properties of a visualizations. The visualizations in the Chart API follow the guideline purposed by Mazza (Mazza, 2009), Mackinlay (Mackinlay, 1986) among others.

One important point that needs to be re-iterated is the classification of categorical data types as ordinal. Ordinal data types are assigned an ordering by the system as a consequence of its design but communicating this ordering to the user is generally not important. Ordinal and categorical data types are largely interchangeable: they are used as labels.

All labels are distinguished with different colour hues. Aside from shape there are no other effective ways to encode such information. Quantitative values are primary encoded by length and position as these are the most effective pre-attentive attributes for conveying quantitative information. In the case of bubble plots the size of a mark is also used to encode a quantitative attribute. Size had a limited suitability for this task. Different sized marks are effective at communicating various magnitudes of a quantitative variable but are not effective for determining specific values. Interaction is used for this purpose. The data element encoded in this way is the last

variable in the input dataset and is presumed to be of least significance.

Pie-charts also encode quantitative variables as size but this is supplemented by encoding the same variable as the arc length.

AC↯DV uses no more than three retinal variables for any one graphical mark. Such a conservative approach is based off the work of Ware (Ware, 2012) who recommends no more than four attributes be used. This approach was used in order to better ensure comprehensible visualizations are returned even if the potential of the medium is not fully exploited.

5.3 Effectiveness.

In order to evaluate the effectiveness of AC↯DV two use cases have been created. These are used to perform a cognitive walk through, a common method used to evaluate HCI systems (Polson, Lewis, Rieman, & Wharton, 1992). A small user interface was constructed that allowed multiple datasets to be selected and visualized by the user. The first use case uses mocked data, the second uses data provided by Fingal County Councils Open Data initiative (Council, 2011).

5.3.1 Coffee Chain Expenses

Tom has been tasked with cutting expenses for his company, a coffee chain. He has several datasets detailing the various expenses and profits associated with each product. These datasets are stored as comma separated value files and have been collected by other tools. An example excerpt is shown below:

```
Coffee,Profit,Expense
Americano, 550, 220
Cappuccino, 435, 110
Chocolate Dalmatian, 200, 100
```

To get an initial understanding of the situation Tom inputs the broadest dataset into AC↯DV which has the total profit and expenses for every product. AC↯DV outputs a scatter plot, as shown in figure 5.1, which clearly shows Soy Lattes are not selling enough to justify their cost.

Tom decides to investigate further and inputs the expense information for the Soy Latte. AC↯DV outputs a pie chart, illustrated in figure 5.2 that shows every expense associated for that product. Most of the expenses look

	Native Browser Based	Automatic Visualization	No User Training Required	Generic Application	Large Number of Visualizations	Interactive	Automatic Feature Determination	No User Input Required	Encoding of Retinal Variables
Gretl		x	x			x			
Excel				x	x		x		
Google Charts API	x			x	x	x	x		
Tableau		x	x	x	x	x	x		x
ManyEyes		x	x	x	x	x	x		
Open HeatMap		x	x			x		x	
D3	x			x	x	x			
Polaris		x	x	x	x	x	x		x
APT		x	x	x	x		x		x
AC↯DV	x	x	x	x		x	x	x	x

Table 5.1: AC↯DV compared to systems evaluated.

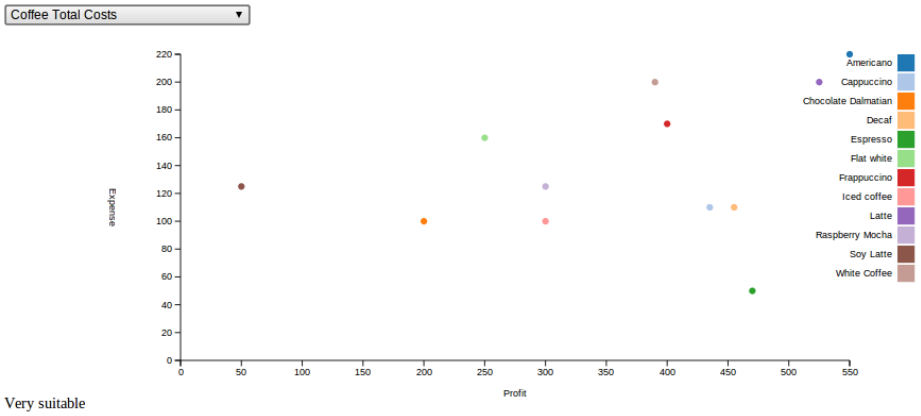

Figure 5.1: Example showing coffee type with associated profit and expenses.

normal except for marketing. Tom concludes that the aggressive marketing for the product is not working.

Tom shows this information to his superiors using the visualizations provided by AC↯DV to illustrate his point. Using this information they decided to drop the aggressive marketing for a trial period and see it affects sales.

5.3.2 Fingal County Council

Alice works for Fingal County Council. She is part of a committee tasked with drafting new proposals to help combat unemployment. These proposls will then be handed off to Ministers. Part of this work involves identifying the areas of highest unemployment within Fingal. She has access to the unemployment and population statistics for each region area within Fingal. She combines these datasets and inputs them into AC↯DV. The dataset, consisting of one categorical variable and two quantitative variables, is matched to a scatter plot as shown in figure 5.4. The scatter plot indicates what areas have the highest unemployment compared to population.

The scatter plot shows a fairly linear relationship between population and population. Blanchardstown is a clear exception. To further illustrate this point Alice expresses the unemployment as a percentage of population and inputs the data into AC↯DV. A bar chart is returned which shows quite clearly which areas have the highest percentage unemployment.

Using these illustrations as part of a body of evidence the committee are able to target the new proposals to have the greatest effect.

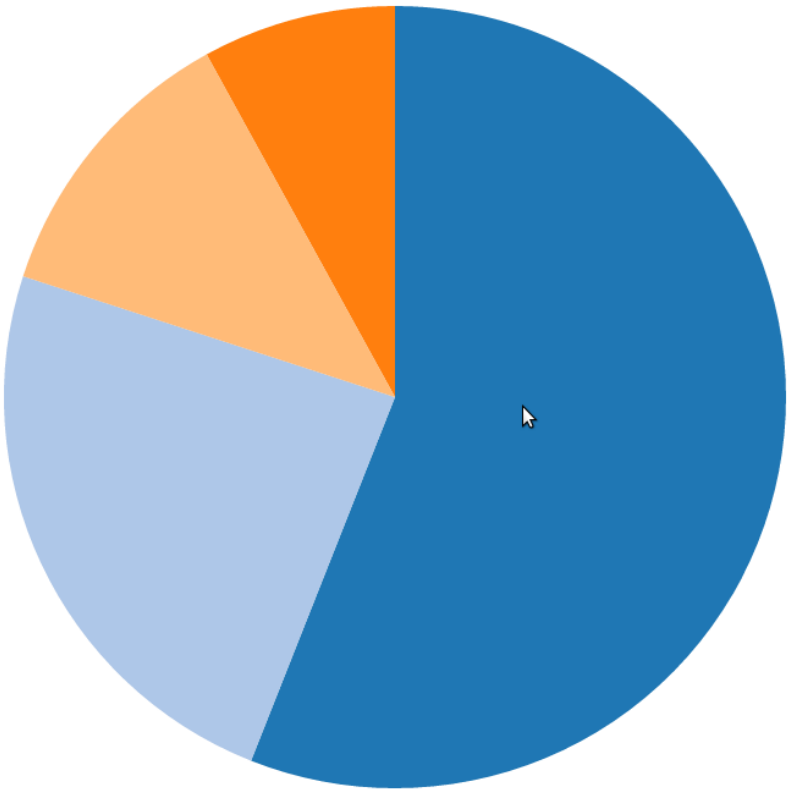

Figure 5.2: Example expenses for Soy Latte.

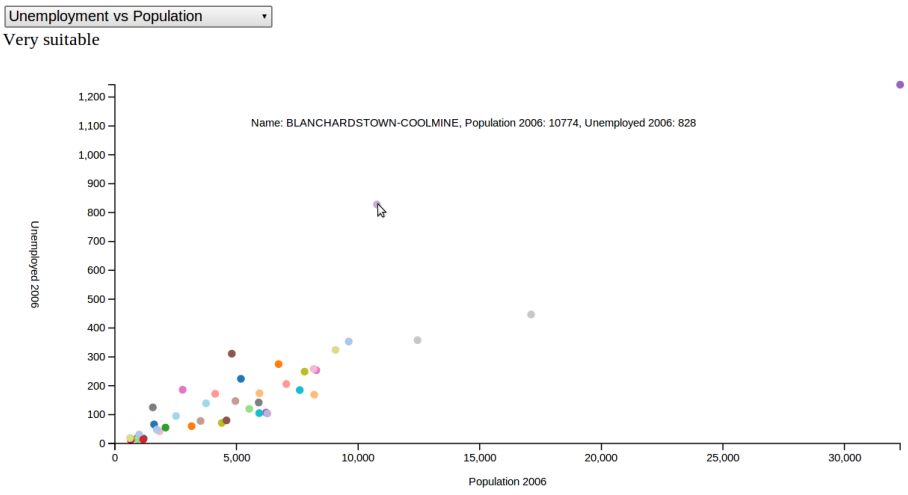

Figure 5.3: Unemployment and Population

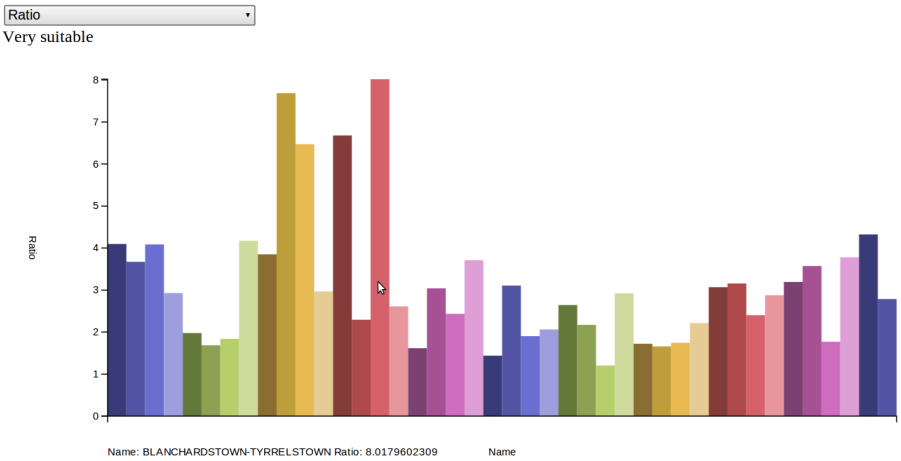

Figure 5.4: Unemployment as a percentage of population

5.4 Discussion

The purpose of this project was to determine to what extent can suitable visualizations by dynamically and automatically generated using browser based technologies. This statement is broken down into three key points to make discussion easier:

- Suitable visualizations
- Automatic generation
- Browser based technologies

Each point is addressed below.

5.4.1 Suitable Visualizations

The idea of suitable visualizations was key to this project. As noted in Section 2.1.1 the suitability of a visualization depends on the capabilities of the viewer. Since such capabilities are not sufficiently modeled a collection of related rules created from the results of human perceptual evaluations were used.

The visualizations are given a set of properties derived from the visualization process described by Mazza (Mazza, 2009). The same properties are

automatically determined for an entered dataset. The properties are then matched and the most suitable visualization is selected. This ensures that the visualization is capable of displaying the information effectively, guaranteeing suitability. The Chart API contains visualizations that are known to be effective as demonstrated by their widespread use.

A harder problem is determining the how suitable a given visualization is. The initial selection process provides a base level, ensuring all of the data will be visualized. As discussed a conservative approach was adopted. The number of data elements mapped to retinal attributes never exceeds three. While AC↯DV allows a large amount of individual elements to be displayed at once, far more than can be processed by sensory memory, the comprehension of of the overall visualization is assisted by interaction and labeling. The retinal attributes are encoded manually when the visualization is being created. While an automated solution would be more desirable the manual encoding is done using an informed process.

The lack of user selection is key to AC↯DV. If however this is desired the visualizations can be used without the classification component.

All of the chart allow details to be viewed on demand. This is one of the keys to creating effective visualizations (Shneiderman, 1996).

5.4.2 Automatic Generation

AC↯DV is completely automated and is designed to be extensible. No user intervention is needed at any stage as detailed above in section 5.2. Again this may be seen as limiting user choice by the visualizations can be used independently if desired. Some trade-offs were made in order to support this behavior. A more complex Chart API would allow more features. For example a less restrictive notion about the dimensionality of information a visualization can display might allow visualizations to display more data. A barchart could be used to display univariate information or could form groups of bars to display information of many dimensions. Currently AC↯DV would need a unique entry in the API for every dimension.

A single internal data structure is used to simplify the visualization process. The meta-data generated to guide the process is associated but separated from the raw data. The visualizations in the Chart API need only process the basic raw data. This allows the meta-data to be extended, i.e. the identification of dates, without effecting other parts of the system.

Some assumptions about dependent and independent variables have been made. These assumption appear to be reasonable and other systems have made similar assumptions (Stolte et al., 2002). They may limit the system from visualizing certain types of information but then searching for examples not a single instance could be found.

5.4.3 Browser Based Technologies

Web browsers are a suitable platform for data visualization. One of the motivations for using visualization is to assist users in the speedy comprehension of a dataset. Browsers are ideal for this as they are ubiquitous and generally do not require the user to preform any setup when running an application for the first time.

One of the main concerns was performance. AC↯DV demonstrates that modern web browsers are more than capable of processing and visualizing large amount of data. The limits of performance were not investigated but the browser appears to more than capable of intensive user tasks.

The addition of more meta-data would incur a performance overhead which may need to be considered if AC↯DV were to be extended. However as the execution time of the program is so small a significant amount of data would need to be added to be noticeable.

Chapter 6

Future Work and Conclusions

This dissertation presents an investigate into the feasibility of automated visualization using browser based technologies. In this final chapter, ideas for the future development of AC⚡DV are presented followed by some concluding remarks about the project.

6.1 Future Work

Data visualization is a complex topic. It is difficult to create visualizations because their effectiveness depends on the capabilities of the viewers. Perhaps the most important thing to come out of the evaluation of this dissertation was an understanding of the limitations and difficulties of an automated visualization system. Possible future work to address these issues is presented below.

6.1.1 Extension of Chart API

The strength of a generic visualization systems is dependent largely on the variety of visualizations it can produce. While AC⚡DV has a good range of visualization capable of displaying a broad range of data there is room for improvement.

Additional univariate, bivariate and trivariate visualizations would allow for the same dataset to be viewed in many different ways, perhaps assisting the viewers comprehension. Aside from creating the visualizations no effort

would be needed to do this other than adding their properties into the classification engine.

Multivariate visualizations are rather more complex and generally fall into two categories. Lower dimensional visualizations can generally be composed together to form multivariate visualizations, i.e. Grouped or stacked barcharts. Such visualizations can in theory be extended to many dimensions but in reality their are practical limits on their effectiveness. There does not seem to be much research evaluating the effectiveness and limits of these composed visualizations. Additionally an understanding of these limitations would also have to be encoded in AC↯DV. This would involve reworking the systems use of dimensionality as an attribute used during classification.

The second type of multivariate visualizations are specialized and intended to only visualize data of a specific dimensionality. These visualizations are often only capable of visualizing a relatively narrow range of data. Furthermore some greater insight about the data is needed to determine if they are appropriate.

6.1.2 Inclusion of Additional Meta-Data

AC↯DV generates meta data from the input dataset. This meta data is then used to select the most appropriate visualization. There is only a limited amount of information that can be gleaned from raw data. The inclusion of additional meta data could further inform and refine the selection process. This information could be supplied by an automated process or by user input.

Additional meta data could be supplied by extending the automated classification component of AC↯DV. One way to gain additional information about an input dataset would be to pattern match the entries against known data types. For example, if geographical coordinates were identified the data could be overlaid on a map. Creating a robust pattern recognition system is a hard problem and such systems are not completely accurate. However such a system would only have to correctly identify data, not extract it from a noisy medium, which would simplify the task.

User provided meta data has the capacity to provide a large amount of information about not just the types of data, but also to give the data context. Providing context and expert knowledge is a very powerful feature and

is currently the subject of a large amount of research. The Knowledge and Data Engineering Group (KDEG) at Trinity College Dublin have developed a system to allow for easy encoding of subject matter expertise. The system allows the SME to be reconciled with multiple data sources that share a common domain (Hampson, 2011). It is envisaged that a system like AC↯DV would be integrated into this framework to create complex visualizations that leverage the knowledge of domain experts.

6.1.3 Improved Understanding of Suitability

Improving the understanding of suitability and the limitations of various visualizations is a task for further research studies. The mapping of data elements to graphical marks is fairly well understood, if somewhat unverified and has been discussed at length throughout this paper. How combinations of these graphical marks interact and influence the viewer is notably less understood and does not appear to be the subject of much research.

An understanding of how to combine multiple elements to create effective visualizations appears to be key. AC↯DV attempts to solve this problem with the properties used to describe visualizations and datasets, namely the soft and hard sizes. This is something of a crude solution.

The choice of well known and understood visualizations included in AC↯DV was a deliberate attempt to further address this issue. It was thought that such visualizations would have had a body of research surrounding them detailing their limitations. This was not found to be the case.

Further work needs to be done to understand the limitations of data visualizations. A lot of existing work that purposes various rules lack any evaluation (Card et al., 1999) (Mazza, 2009). This would seem to be quite a serious omission for such a user dependent process. Any work done to understand the limitations of visualizations should included a complete evaluation.

One of the reasons AC↯DV only employees well known visualizations was to avoid user evaluation of novel visualization techniques as developing such techniques were not the goal of the project. User evaluation of the visualizations produced by AC↯DV should be considered for future work. However it would be both more efficient and beneficial to purpose and evaluate a model for combining elements in a visualization. Such an undertaking would would best be done as a separate project.

6.2 Conclusion

This thesis has shown how raw data from a range of applications can be transformed into a suitable visualization. The limitation of automatic visualization system were also investigated and discussed. By removing the need for user input the system is easier to use and better able to return an appropriate visualization. This comes at the cost of user choice. The notion of suitability is particularly hard to define and test empirically. Further research in this area is needed.

AC↯DV represents only the early stages of a complete solution to automatically visualize arbitrary data. While it is capable of visualizing a wide range of data there are some limitations. This work highlights any short comings of a fully automated approach and purposes ideas for future work. Additional meta data would provide greater context and allow for more suitable and nuanced visualizations. If such a tool was presented with a user interface it could provide a wide range of users with useful visualizations that enable them to better analyze their data and gain insight from it.

Overall the objectives set out have been met. Visualizations are created completely automatically entirely within the browser. AC↯DV has combined a number of features drawn form the state of the art and has shown the browser is an appropriate platform for native visualization, both in terms of performance as perceived by the user and it's ability to work with large amounts of data. In conclusion the work undertaken for this project has proved to be a worthwhile exercise and one that highlights some of the concerns the present themselves when creating an automatic visualization system.

References

Bertin, J., & Barbut, M. (1973). Sémiologie graphique. Mouton; Paris: Gauthier-Villars.

Bostock, M. (2010). D3. http://d3js.org/. (Accessed: 2013-03-23)

Bostock, M. (2012). Towards reusable charts. http://bost.ocks.org/mike/chart/. (Accessed: 2013-05-22)

Card, S., Mackinlay, J., & Shneiderman, B. (1999). Readings in information visualization: using vision to think. Morgan Kaufmann.

Cleveland, W., & McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. Journal of the American Statistical Association, 79(387), 531–554.

Cottrell, A., & Lucchetti, R. (2002). Gretl. http://gretl.sourceforge.net/. (Accessed: 2013-03-23)

Council, F. C. (2011). Fingal open data. http://data.fingal.ie/. (Accessed: 2013-05-22)

Crockford, D. (2006). The application/json media type for javascript object notation (json). http://www.ietf.org/rfc/rfc4627.

Dix, A. (2004). Human-computer interaction. Prentice hall.

Elmqvist, N., Dragicevic, P., & Fekete, J.-D. (2008). Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. Visualization and Computer Graphics, IEEE Transactions on, 14(6), 1539–1148.

Few, S. (2004). Show me the numbers: Designing tables and graphs to enlighten.

Friendly, M., & Denis, D. (2001). Milestones in the history of thematic cartography, statistical graphics, and data visualization. Accessed: March, 18, 2010.

Hampson, C. (2011). An expert supported approach to data exploration. Unpublished doctoral dissertation, Trinity College Dublin.

Haskell, A. (1919). How to make and use graphic charts. Codex book company inc.

Inc., G. (2012). gcharts. https://developers.google.com/chart/. (Accessed: 2013-03-23)

Inc., I. (2002). Sse. http://software.intel.com/en-us/articles/using-intel-streaming-simd-extensions-and-intel-integrated-performance-primiti (Accessed: 2013-04-14)

Inc., M. (2007). Excel. http://office.microsoft.com/en-ie/excel/. (Accessed: 2013-03-23)

Inselberg, A., & Dimsdale, B. (1991). Parallel coordinates. In Human-machine interactive systems (pp. 199–233). Springer.

Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. ACM Transactions on Graphics (TOG), 5(2), 110–141.

Mazza, R. (2009). Introduction to information visualization. Springer.

Miller, G. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. The Psychological Review, 63, 81-97.

Oxford English Dictionary Online, 2nd edition. (2013, July). http://www.oed.com/.

Polson, P. G., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. International Journal of man-machine studies, 36(5), 741–773.

Salvad, C. (2009). Validate numbers in javascript. http://stackoverflow.com/questions/18082/validate-numbers-in-javascript-isnumeric.

Shafranovich, Y. (2005). Common format and mime type for comma-separated values (csv) files. http://tools.ietf.org/html/rfc4180.

Shevchuk, A. (2012). Hohli charts. http://charts.hohli.com/. (Accessed: 2013-03-23)

Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations. In Proceedings of the ieee symposium on visual languages (p. 336-343).

Software, T. (2008). Tableau. http://www.tableausoftware.com/. (Accessed: 2013-03-23)

Spence, R. (2001). Information visualization. Addison-Wesley.

Stolte, C., Tang, D., & Hanrahan, P. (2002). Polaris: A system for query, analysis, and visualization of multidimensional relational databases. Visualization and Computer Graphics, IEEE Transactions on, 8(1), 52–65.

Tukey, J. W. (1977). Exploratory data analysis. Reading, MA, 231.

Viegas, F. B., Wattenberg, M., Van Ham, F., Kriss, J., & McKeon, M. (2007). Manyeyes: a site for visualization at internet scale. Visualization and Computer Graphics, IEEE Transactions on, 13(6), 1121–1128.

Warden, P. (2010). Open heat map. http://www.openheatmap.com/. (Accessed: 2013-03-23)

Ware, C. (2012). Information visualization: perception for design. Morgan Kaufmann Pub.

Zhu, N. (2012). Dimensional charting. http://nickqizhu.github.io/dc.js/. (Accessed: 2013-03-25)