Visual Analytics System for Large-scale Data Exploration

by
Glenn Allen Turner

A Thesis submitted to the Department of Computer Science,

College of Natural Sciences and Mathematics

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in Computer Science

Chair of Committee: Guoning Chen

Committee Member: Yunpeng Zhang

Committee Member: Weidong Shi

University of Houston
December 2019

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Guoning Chen, for his patience, guidance, kindness, support and instructions during my work on this thesis, especially for his help in constructing this text.

I am very grateful to Dr. Weidog (Larry) Shi, for serving on my thesis committee and for being my friend and offering me encouragement throughout my graduate studies.

I wish to thank Dr. Yunpeng (Jack) Zhang for serving on my committee and for guidance in the work with the traffic control data.

I also want to thank other students in Dr. Chen's group, Lieyu Shi, Kaoji Xu, and especially Duong Nguyen for early comments and help getting started with Qt and QCustomPlot.

ABSTRACT

Visualization aims to produce meaningful visual representation of data, which has been shown to be a powerful means to reveal information (e.g., structures and/or relations) hidden in large-scale and complex data where our knowledge of these data is sparse. From the visualization of these data sets, important details are often difficult to discern due to the occlusion caused by the excessive visual elements presented. In this thesis, I present a visual analytics system that offers a complete pipeline for processing and visualizing these large-scale, complex data, starting from data cleaning and analysis to visualization generation and user interactions. In particular, my system supports a level-of-detail exploration scheme, showing the overview of the data in the beginning with various visual representations, followed by the detail-on-demand exploration. This thesis provides detailed description on the design and implementation of my system. To demonstrate the utility and generality of my system, I customized and applied it for the discovery of abnormal patterns hidden in traffic control data. New data cleaning, analysis, and visualization techniques are developed in order to address these traffic control data. My system successful revealed the errors in the data, including missing and misplaced entries. It also helped identify different traffic situations at different street intersections.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Many scientific and business fields of study require collection and analysis of data for decision making. It is increasingly easy to collect and store data concerning all facets of life. However, depending on the applications, these data can be of large scale and reside in a high-dimensional and abstract space, making their comprehension challenging. Analysis and discovery tools are needed to extract useful information from these data. While many automated tools are now employed, human judgment and interaction is often needed. This is especially true for multidimensional, complex data. To address this challenge, visualization and visual analytics techniques can be employed to reveal certain relations hidden in these data in an intuitive way.

According to Khan [2], visualization is "the use of computer-supported, interactive, visual representation of data to amplify cognition". It represents the data into various visual forms to enable the comprehension of the data. Visualization has been shown to be a powerful means to reveal information (e.g., structures and relations) hidden in complex data and a de-facto tool in data analysis.

There are two different types of data that different applications collect and analyze, i.e. scientific data and information data. Scientific data is collected to describe certain physical phenomena in the physical world (e.g., fluid motion, temperature distribution, etc.), while information data measure and describe events and relations in abstract space (e.g., posts on social media, performance of the stock market, etc.). My thesis work focuses on the information data sets and their visualization. Different from scientific data where the spatial layout of the individual data points is provided (i.e. each data point is associated with a physical position), information data do not have the inherent spatial layout for their data points, making their intuitive representation challenging.

Often informational data, especially high-dimensional data, are stored in the form of tables, and are referred to as tabular data [1]. Specifically, each row of the table corresponds to a data entry and each column documents one attribute of this entry. One example of this tabular data format is students' records. Other popular types of information data are graphs and trees. Graphs are used to represent the relations among different subjects, which are usually stored as the list of graph nodes and the list of graph edges. Trees are used to represent hierarchical relations among subjects, which is a special type of graphs without loops. Trees can be similarly represented as a list of nodes and a list of edges. In my thesis work, I will focus on the tabular data and their effective visualization and exploration.

Simply visualizing all data entries is often insufficient to thoroughly explore the data. Often details are overlooked because of the density of objects on the screen, or objects are occluded. To address that, visual analytics is employed to marry machine power with human expertise to achieve an effective interpretation of the data. In particular, visual analytics supports an analytic reasoning enabled by effective interactive visual interfaces [2]. In this thesis work, I will employ the visual analytics paradigm to aid the comprehension of tabular data.

1.2 Contributions

I developed a visual analytics system that offers a complete pipeline for processing and visualizing large-scale, high-dimensional data stored in the form of tables, starting from data cleaning and analysis to visualization generation and user interactions. This system supports a level-of-detail exploration scheme, showing the overview of the data in the beginning with various visual representations, followed by the details-on-demand exploration. User interactions are supported at various stages in the visualization pipeline. In addition, I have added support showing the numerical values of user selected data to aid the user exploration. To additionally enhance the details-on-demand feature the preprocessor generates meta information that is utilized by the graphics program to produce annotation of the data on the display.

This system was not designed for the handling of a particular data set but rather with the flexibility to work with a variety of data. I demonstrate that the system can be adapted to a data set for which we have little prior knowledge. While the system had many features in its initial form, working with a test case suggested additional features specifically for the data being tested.

I apply this system to data from a traffic control system. In particular, I aim to identify certain abnormal patterns in the traffic lights of different street intersections documented that may suggest potential hacking activities. The existing visualization techniques have trouble revealing any meaningful patterns due to the frequent transitioning of traffic lights (i.e., Green to Yellow to Red) within a small time range. To address that, I enhanced the above general visual analytics system by including additional preprocessing to identify the individual traffic light cycles and to perform various calculations on the cycles, including accumulation of traffic light behaviors within a specific time range. With the additional preprocessing and new visualization mapping, we are able to identify errors in the raw data (including missing and misplaced entries) and different traffic light patterns at intersections that have different traffic volumes (i.e., Major roads vs. Minor roads).

1.3 Organization of this Thesis

The rest of the thesis is organized as follows. Chapter 2 reviews the literature of visual analytics and the relevant systems. Chapter 3 provides details of the system developed in this work. All of the plot types as well as the annotation and user interaction features are explained. An application of this system is described in Chapter 4 as a case study that shows the utility and effectiveness of the system for the discovery of abnormal patterns hidden in a traffic control data set. Finally, the conclusions drawn from this work are outlined followed by suggestions for future work.

Chapter 2

Background and Related Work

Visualization today involves many fields ranging from computer science, mathematics, physics to perception science [3]. It is an established area of study in academia and has become a part of business intelligence systems. Much current commercial visualization software available started in academic research [4].

Information visualization is used to either provide answers to questions about data or elucidate facts that were not known. In the case study presented later in this thesis we see both uses. When used to answer questions, the requirements of a visualization system are determined by the specific knowledge sought. Sometime these questions are specific, such as:

- Do the data show an evolving trend?
- Are different properties of the data related, can the behavior of one be predicted from another?
- Can groupings and clustering of data be identified?
- Do the data show abnormal changes as if there was tampering or errors?

For large and multi-variate data, visualization is often used to provide insight. Here abstraction and the use of a higher number of dimensions in the display can sometime reveal the unexpected.

As described later, it can also allow for the exploration of data through user interactions.

2.1 The Visual Analytics Pipeline

Visualization is an iterative process consisting of a number of steps. These steps are often referred to by different names by different authors [5]. The terms used here follow the terminology suggested by Telea [3].

- Importing raw data
- Filtering and enrichment
- Mapping to visual representation
- Rendering to image

In what follows, I will briefly review these steps and describe how I achieve these steps in my visual analytics system in Chapter 3.

Importing raw data. The data must be in a format the visualization program can read. The visualization system may be written for a specific data input, but more often it is written to read a standard data format, and data sources must be reformatted to fit that specification. The actual visualization program may require a custom format. Examples include Jason objects, XML, XLS (Excel) or more commonly a text file such as .csv, or even plain text.

Filtering and enrichment. The preprocessor may perform much more complex data processing than simple reformatting. The input data may not be a one-to-one match with the data to be viewed. Missing data may be flagged or interpolation may be used to fill missing parts. Summary calculations may be made to generate other data fields to be displayed. Clustering may be calculated to have this information conveyed on the display as well. Filtering may be done to remove data fields that are known to be not useful. Additional calculations may be performed for

the subsequent visualization tasks, including the calculation of statistical information of the data (e.g., mean, standard deviation, probability distribution, etc.) and other data abstraction computations. Sometimes outlier data are removed and dimensionality reduction may be performed. While this can be done in the visualization program it is often accomplished in the preprocessor. The preprocessor program is customized to the data source and the specific needs of the user.

Mapping. The data are mapped to graphics primitives such as lines, bars, circles, three dimensional mesh, etc. The choice of visual primitives is suggested by the type of data and the visual encoding is chosen by the user. The user can interactively switch between different visual encodings of the data before revealing interesting patterns and relations in the data.

Rendering. After the visual mapping is determined, the selected visual primitives representing the data will be drawn on to the (2D) screen of a computer. In general, graphics libraries, such as OpenGL, can be used to render the geometry and its associated optical properties onto the screen. For the rendering of 3D representation, the view port and view point need to be decided first before rendering. After rendering, user interactions can be performed to change view point and/or focus for the exploration of the data.

From this list we see that data visualization may be thought of as a series of steps, some may be in a preprocessing program, and some in the visualization program. In this case the preprocessor is an integral part of the visualization system. It should be noted here that user interaction may be part of more than one of these steps.

2.2 Visual Information Seeking – User Interaction

The initial visualization of the data may not put focus on the places or attributes that are interesting to the user. User interaction is then required to regenerate the visualization with more desired focus or representation [6][7]. Visual cues can be used to spot detail that would otherwise be overlooked, then the user may zoom in for more information. As we will see in the next chapter, a few points

of interest can even be displayed in numeric form.

The general paradigm for the visual exploration by the user is as follows [6]:

- Overview
- Zoom and filter
- Details-on-demand

In addition we might add the following:

- Link
- History
- Extract

Overview first. This is to get a quick assessment of the data. The user sees overall patterns and may spot areas of interest. One might imagine a very cluttered line graph. Some lines may stand out and warrant further exploration [8].

Zoom and filter. This is finding a portion of the data to display and examining it closely. This may be as simple as changing scale to enlarge a portion of the data, or using a "rubber band" selection to pick a portion of the plot to enlarge. It may also be selecting a line or a group of lines and have the others fade into the background. All of these methods have been implemented in the system described later in this thesis. As mentioned before, some filtering may be done in the preprocessor, so the user may set filter parameters after the initial look and reprocess the input.

Details-on-demand. This may be implemented as a pop-up window with numerical data for a selected line, or a portion of a line. In some systems this pop-up window has HTML text that links to other data [6]. In addition, meta information, that is other information about the displayed data, may be presented to the user.

Link. A system can be made to allow the user to select an item, then items with similar attributes would be highlighted. A pop-up text box may be used to name related items with similar attributes. In the implementation here, when a line plot and bar plot are shown together, I simply link from an item on one plot to select the related item in the other. For example, a user selection of a bar or line will also select the other presentation of the same data in the other plot.

The next two user interactions are not implemented here but we include the descriptions because (1) they are described in the literature and (2) we may consider them for future work.

History. The program may support a history of actions to allow "undo" to reverse a sequence of operations that does not produce the desired outcome. In addition, a series of steps may be saved and "replayed" on a series of data sets to reproduce a desired outcome.

Extract. The data that are a subset of the display may be extracted as data for use in another application. It may also be saved and re-imported for later comparisons to other data.

These steps are part of an iterative process. After details-on-demand the user may need to again see an overview. A visual analytics system should be able to allow the user to easily navigate these steps [8].

2.3 Related Work

Visual analytics is an active research field in data visualization. It is impossible to review all visual analytics systems and the relevant techniques. In this section, I will first review some of the popular commercial tools that only provide simple and standard plots for tabular data. Then, I will briefly review some recently developed visual analytics systems that incorporate customized visual representation and user interaction paradigms for specific data processing. Finally, I will describe the libraries that I use to aid the development and implementation of my visual analytics system.

2.3.1 Consumer Software

Often, the commonly available charting packages in consumer software, such as Microsoft Excel and Google Spreadsheets, can already satisfy the need of producing useful plots. They are widely used because they are easy to learn and generally available. Adobe Illustrator has been cited as another popular and widely available consumer tool for visualization. These consumer tools have some shortcomings in that the stock chart types may not be exactly what is wanted. In addition, there is little, if any, user interaction [9].

This availability of the above consumer tools is not without criticism. With the use of personal computers this software is on virtually every desktop, and some argue that the quality of the visual presentations is ignored [4].

2.3.2 Data Analysis and Visualization Platforms

Some visualization applications allow many types of plots and user interactions. Paraview [10], for scientific data, falls into that category. In addition, it is open source and therefore extensible. The extensibility requires considerable programming expertise.

More complex software packages allow much more complex displays and user interactions. We will call these Data Analysis Applications [11]. Some examples include Sisense [12], Google Data Studio [13], Tableau [14], R Studio based GGpplot2 [15] and others that also incorporate business intelligence.

All of these are extensible so that while they are useful on their own, with some programming expertise they can be customized. There is a broad grey line, it seems, separating extensible visualization packages and visualization toolkits.

Another class of applications, I will also call them platforms, is used for what is called “visual programming” [16]. In applications of this type the program specification is made graphically. The

user may use representations of program function as objects on the screen that can be manipulated. An example is GeoVista Studio [17] that allows a data analysis and display program to be assembled by connection components on the screen.

2.3.3 Platforms that Are Development Environments

A programming platform often used for data analysis and visualization is MATLAB. It has powerful plotting capabilities. It is usually used for mathematical simulation but sometime it is used for data analysis and plotting [18].

Similarly, R Studio may be called a programming platform, or programming environment. Users with minimal program development ability can generate custom presentations and analysis of data.

2.3.4 Frameworks and Toolkits

We make the slight distinction that frameworks are structured applications that are extensible, and toolkits are libraries of widgets. The framework calls user extensions, but a toolkit is called by a user program. This distinction is not maintained by all vendors, Qt for example markets itself as a framework but it is used as a library of widgets.

Visualization systems with specific target data domains are mentioned by Bostock [9] and others. Some of these are domain specific, such as for visualizing the organization of data within databases. McLendon [19] lists no less than eight toolkits and frameworks specifically for network data. They go on to develop their own application based on the VTK tookit.

Programming toolkits provide much more flexibility although requiring more programming knowledge and additional efforts to achieve the customized visualizations. More chart types are available and more user interaction can be provided. One popular and powerful programming toolkit is the Visualization Toolkit (VTK) by Kitware, although it is mostly used for scientific visualization. There are some lower level programming libraries, such as QT by the Qt group and

its add-on, QCustomPlot [20]. VTK and Qt have python wrappers, making them accessible beyond the programming community, but to people with some programming knowledge. It is worth noting that VTK and Qt have open source versions to allow users to customize the libraries. Another library for the development of web-browser based visualization system is D3.js [21].

With this we should include python libraries numpy, and matplotlib, for holding and plotting data. Python is becoming widely used even among non-programmers.

Sometimes there are open source additions to the vendor distributed libraries. This allows additional functionality to be used by a community associated with a toolkit. The visualization application for this thesis was developed using Qt with QCustomPlot. Qt has an open source offering, and QCustomPlot is distributed as source code.

Chapter 3

The Visual Analytics System

3.1 System Overview

This system is built as a collection of preprocessors and a graphics program. The roles of these programs are described in the sections below. The system follows the standard visualization pipeline discussed in Chapter 2. Various steps in this pipeline allow for user interaction as is indicated in Figure 3.1. The steps in this pipeline are defined for this system in this section and described in

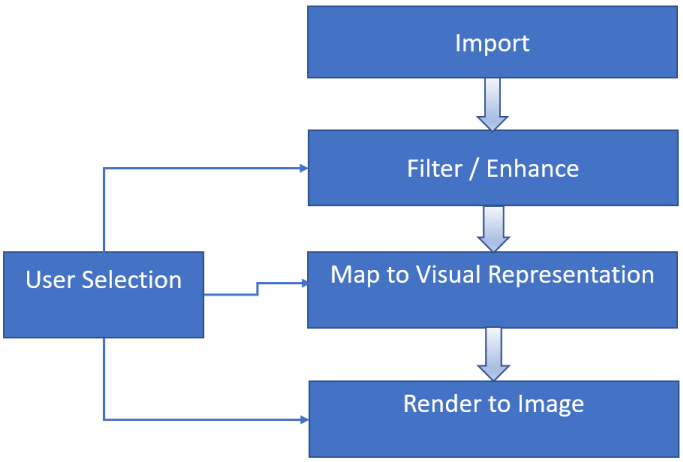


Figure 3.1: The classic visualization pipeline. User interaction may participate in several steps.

detail in the next section.

Import. The system is developed with the intention of handling different formats of the data files. However, since the graphics program reads a file with specific formats as the input, all data sources need to be converted into one of these formats, which can be achieved via a preprocessor. The preprocessors are an integral part of the system. Some preprocessors simply read the data in one format and write a file compatible to the graphics program. The formats required by the graphics program are described in Appendix A.

Filter and enhance. These actions are the basic validation of the loaded data to remove irrelevant or incorrect entries. Additional calculations may be performed for the subsequent visualization tasks, including intermediate summations, normalization, and other abstraction computations.

Some minimal filtering is done by the graphics program per user interaction. For example, complex plots of many variables may have several of these variables removed from consideration.

Map to visual representation. After filtering and enhancing, the next step determines the proper visual form given the properties of the data and the needs of the tasks. For instance, line plots can be used to visually represent time-series data. Scatter plots can be used for the study of the co-variance behavior of two independent variables (or attributes in this case). Histograms can be used to visualize the distribution of data entries based on a certain attribute. In short, the visualization mapping converts the data into a form that can be rendered using existing computer graphics techniques.

Render to image. This step determines the color for each pixel of the output image based on the selected visual representation and the value of the data. For instance, the axes of the plots and the curve representing a time-series data can be rendered (or drawn) on the screen using the line rasterization algorithm. In this project, we use the internal mechanism of QCustomPlot to achieve the rendering of the selected plots.

User interaction. User interaction can take place in all of these steps except the simple import. For example, in the Filter and Enhance step, user interaction may be applied to configure the preprocessing program to select or calculate the desired output. In the Map to Visual Representation step it can be used for selecting the desired plot type, choosing any subset of data to be plotted, and choosing graphical elements. In Render to Image it can be applied to choose the viewing parameter and to control the color.

3.2 Import - Filter and Enhance.

As indicated before, the components are a collection of preprocessors and a program that is both for visualization and exploration. This section describes the preprocessors. These small custom programs convert the data to the format that the subsequent visualization program can read.

3.2.1 Preprocessors.

Several preprocessors were written to reformat various data sets for this work. For example, the data from The University of California at Irving Machine Learning Repository Iris data [22], and Automobile data [23], each require a separate preprocessor because they are in different formats. In fact an additional preprocessor was written for the Iris data in order to produce the iris scatter plot, shown in the next section on plotting, because that plot required a different input data type. Data types will be described later in this section.

When the data are simply being reformatted there is no need for user interaction at this stage. There are times, however, when the reformatting eliminates variables and is also performing a filtering function. For instance the "Automobile" data set (UCI) [23] has 26 attributes; only a small subset of these are provided as input to the plotting program.

The preprocessors in this system that were developed for the case study described in Chapter 4 were written to not only reformat but also to filter and calculate summary values for display as

well. In the case study, the total times traffic signals spent in the various states, Red, Green, or Yellow, are calculated on an hourly basis and output for plotting. Ratios of Red/Green time are also calculated for each cycle. In addition, for other examinations, these data are normalized to emphasize different relationships. Another use of the preprocessor in the case study was to sort data on the fly to correct entries that are out of order.

In this system, for the case study, an additional preprocessor feature was needed. Missing data are noted by the preprocessor by measuring elapsed time, and special points are inserted into the data for the plotting program. Time series data are marked so that the plotting program will interrupt the line drawn to show that data is missing.

A novel addition made in this system is to have meta information generated by the preprocessor and added to data passed to the plotting program. This is used in three ways that will be described in the section on annotation.

3.2.2 Data Types.

The graphics program takes as input, tabular data, in comma separated value (.csv) files. These data may be arranged so that the rows are data records, which may not necessarily be identifiable as different variates, each with different sets of attributes. This is because for some data the columns are groups, and the rows are variables such as data for parallel coordinate plots, while for others the columns are variables, such as with a time series, like daily high temperature for a year. Each data file has a keyword in the file header that identifies the data type.

The data with the same set of attribute names may be grouped into classes. One such example is the Iris Data Set, where records describe various attributes of a flower, e.g., petal length and width. These records are classified by the type of flower they describe in this data set. Many variations on attributes are all classified as four different types of flower. Another example is the Automobile data set. Several sets of attributes are given for each make (class) of automobile. Data

from both of these data sets are used in plots produced by the graphics program.

Here I describe several data formats, and refer to them by names that are only used in this work, and have no meaning outside of this work. Each of these data types has a specific input format that is described in Appendix A. The files have a line of header text that identifies the data type to the graphics program via a keyword specified in Appendix A. Identification of the different data types is important because they have a set of plot types associated with each.

Series data. These are data where X is a variable and each Y is a value given at a specific X value. These data may be thought of as continuous data often expressed as $y = f(x)$. A simple example is temperature values of a place over many months. The program accepts many values of the Y variable for each value of X, that is, there may be many series (lines) with the same X but each with different Y value. The data may also be "X,Y" pairs, where there are unique X values for each series. Series data may be plotted as lines, lines with points, or points alone.

When series data are entered as pairs of points they may also include one of three types of metadata. Examples of how this meta information is used are shown later in this chapter. This meta information is only available for series data entered in pairs.

Categorical data. These groups may be for different attributes, and we call them categories. An example is the Iris data set: petal length and width, sepal length and width are all categories. Each data record has a set of values for each of these categories. This type of data can usually be visualized by a parallel coordinate plot. To study the co-variant behaviors between pairs of attributes, scatter plots or combined line plot can be used to show a subset of the variables.

We have a special kind of categorical data, that is data that can be grouped into classes. An example of this are data used for machine learning, or the result of a classification program.

Class data may be presented as a line plot or a scatter plot. In this case, a group of lines, or scatter points, that are members of the same class are drawn with the same color. An example of this is the scatter plot shown in the next section.

Data group series. These data have some characteristics of categorical and/or series data. It may be a data series that only exist at a few discrete points, or categories, for example, population data over a few years. Generally there is a collection of values at each discrete point along the X-axis. These collections of variables at each point are a group. This type of data may be represented by line (multi-line) plots, with points alone, or by bar charts. The cluster of bars at each point is the group. It is distinguished from series data because there are only a few points and so bar charts may be used in the display.

The independent variable here is not necessarily numeric. In fact the program considers the input as character strings. An example may be scores on a series of exams in a class of students, the data is the number of "A"s, number of "B"s and so on. It may be represented by a line plot, for instance, a line for the number of instances of each grade plotted with each exam on the X-axis It may also be represented as a bar chart with a histogram for each exam showing the number of occurrences of each letter grade.

Surface data. This is a data set of the form $z = (x, y)$. As implemented here, these data are roughly continuous. They are displayed as a 3D rectangular mesh. The program requires a regular grid to represent the XY plane in the implementation, and the user must choose the same number of samples along X and Y directions, respectively.

3.3 Visual Representation - Render to Image.

In this stage, the type of plot and the graphical elements on it are chosen. The program usually chooses the plot type based on the data type. The data type may be chosen by the user in preprocessing. Dimensions may be filtered out to allow multi-dimensional data to be displayed as a two dimensional plot. For one data type, i.e., "datagroupseries", the user may choose a line plot, bar plot, or both. Some aspects of rendering are controlled by user interaction also, for example, color. This is more fully explained in the Example Workflow section in this chapter.

3.3.1 Plot Types.

The following describes the plot types available in the system developed for this work. The intent was to keep the plotting simple to develop, and simple to use and understand. Even with those constraints, in the literature and in practice, there is an enormous universe of visual representations of data. Those implemented here were chosen to address the most likely types of data this system would encounter. We demonstrate the use of these plots via a few examples in this section. Additional examples are shown in the case study in Chapter 4. [1]

Line plot, Multi-line graph. Line plots are one of the most common and basic types of visual representations. They are commonly used to depict functional relationships among several lines displayed on a single axis pair thus showing common trends among data instances. Different instances of the same data type may be distinguished by color, or, lines belonging to a class are grouped by color. An example is shown in Figure 3.2.

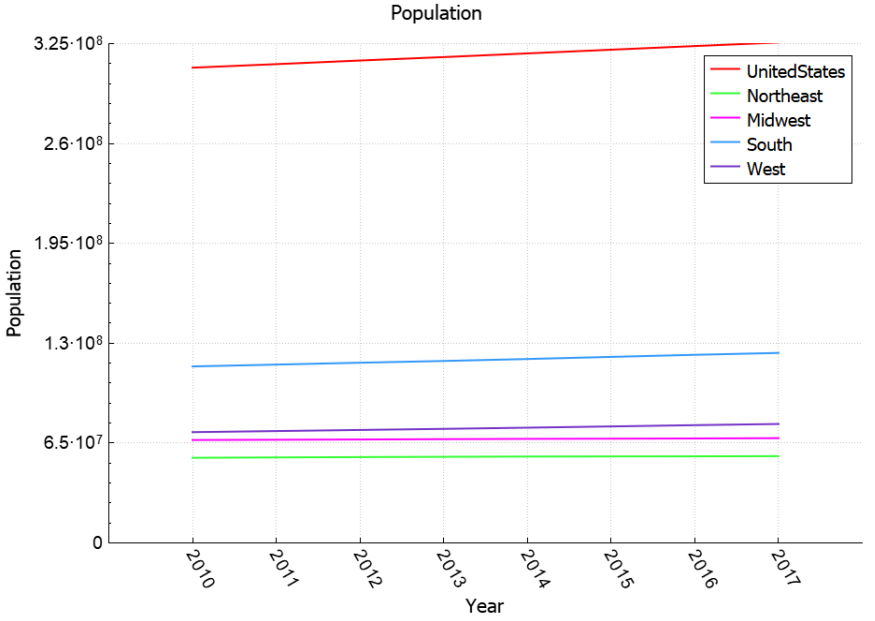


Figure 3.2: Line plot showing the population by region for a series of years.

[1] All plots shown in this thesis are generated by the application developed for this work

Bar plots. Bar graphs, or bar charts, display rectangles of varying heights to show values of variables in categorical data. The heights of the bars represent different values, and the bars are grouped into categories. This is commonly used for categorical data with only a few variables. Different variables are distinguished by color. Figure 3.3 is an example of a bar plot.

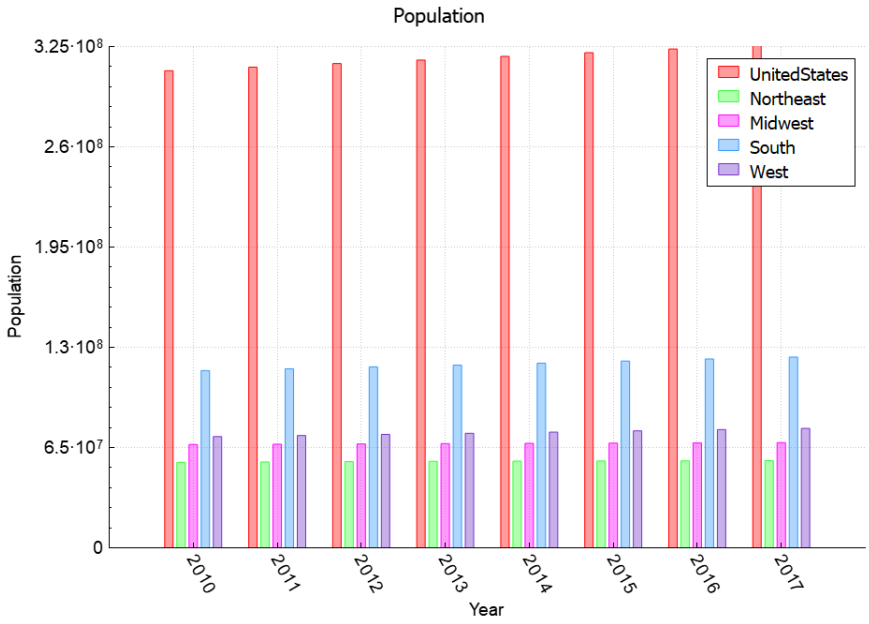


Figure 3.3: Bar plot showing the same data as Figure 3.2.

Scatter plots. Scatter plots show data at discrete points. Often data variables have no discernible relationship. Like line graphs, scatter plots show how one variable relates to another, but may be more useful when data points are not sequentially related and a connected line is not meaningful. In addition scatter plots are used to show clustering or natural grouping of data.

Another use for scatter plots is to show correlation. The application developed here can produce a matrix of scatter plots, showing every variable in a data set plotted against every other variable to see if there is a relationship. This was developed to allow highly correlated variables to be identified so that one or more can be removed to reduce clutter. See Figure 3.10 for an example of this. This is explained further in the User Interactions section.

Parallel coordinates plot. Parallel coordinate plots are used to visualize high dimensional

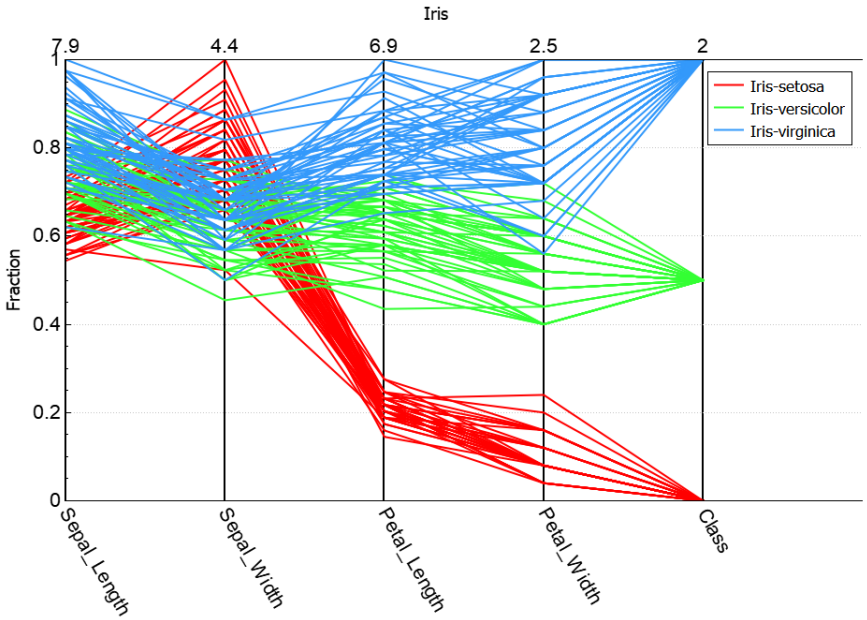


Figure 3.4: Parallel coordinates for the Iris data set. All four descriptive attributes (dimensions) of the data are shown. The colors only differentiate the types of flower.

data. These plots have multiple vertical lines that function as axes for each dimension. This allows for many data features to be examined at once. Each variable, represented by a line, will have a value for each of these dimensions. An input data type is defined for the program specifically for this type of plot.

In Figure 3.4, the Iris data set has 150 data points, each describes a flower in terms of four attributes. In addition, there is a classification of the type of flower. In Figure 3.5, color is used to describe this additional attribute.

3D Mesh. This can be used to visualize values of a variable Z with respect to both the values of X, and Y. While not implemented here, sometime surfaces are used to represent continuous values of Z. The surface can be enhanced with color, and a scatter plot with glyphs may be used to display additional information. The implementation here, shown in Figure 3.6, is only for numerical data and only displays a mesh.

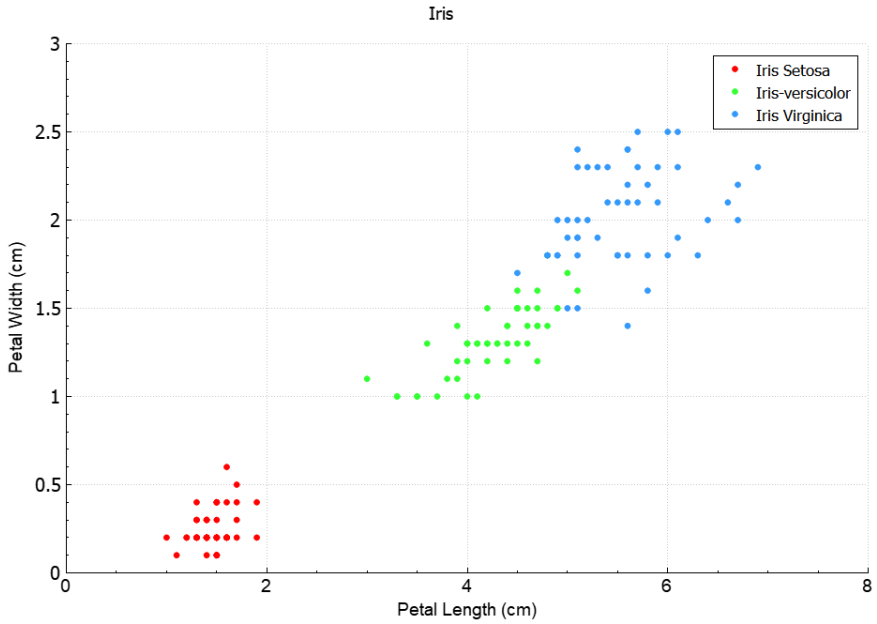


Figure 3.5: Scatter plot for the Iris data set. Petal length and width are the only dimensions considered. The color adds a dimension of type of flower.

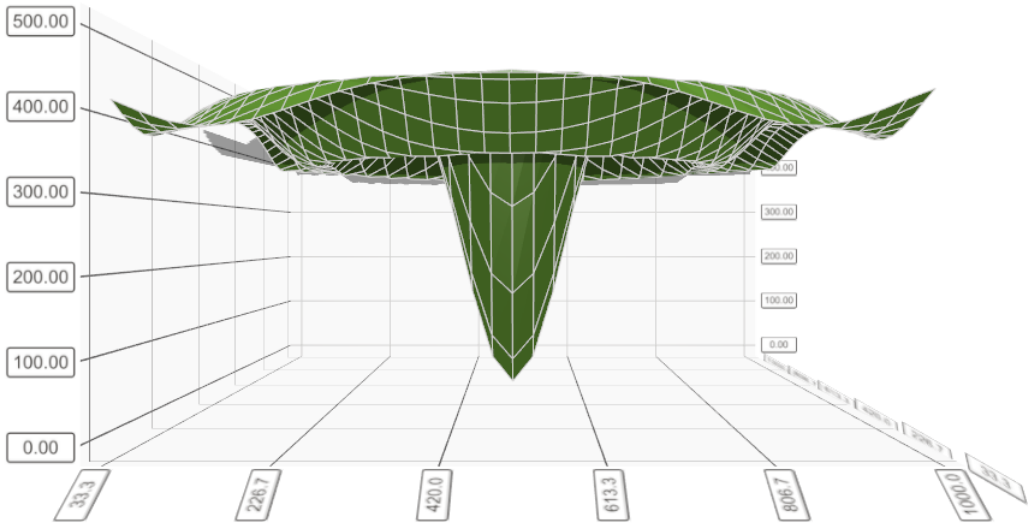


Figure 3.6: Perspective view of a 3D mesh

3.3.2 Rendering the Image

As mentioned earlier the basic drawing on the display is done by libraries Qt and the open source QCustomPlot. The API for QCustomPlot is a set of classes for basic charting. Some Qt Chart classes are also used directly for the 3D display. These libraries are responsible for the drawing of the axes and scaling the data points to screen coordinates within those axes, and the drawing lines and points.

3.4 User Interactions

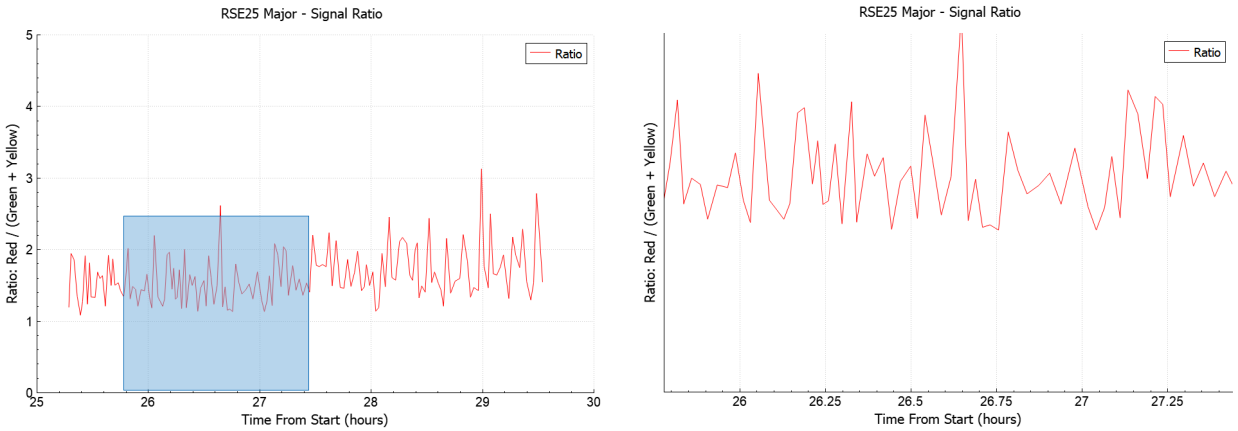
User interaction allow the visual exploration described in Chapter 2: overview, zoom and filter, details on demand, relate, and extract. Some user interaction is possible at the preprocessor stage by altering the code. In some cases this is just setting values of parameters used in program logic to generate meta information.

Additional user interactions allow the user to highlight some data or, exclude data, to allow graphic elements that would be occluded to be viewed. Some are simple, like highlighting a selected line or bar, or even selecting multiple lines. Some are more complex, such as removing data based on correlation. In the implementation here, none of the described user interactions is available for the 3D mesh plots beyond rotating and zooming.

Zooming. Zooming is to exclude part of the data and enlarge a selected portion to see detail. In the graphics program, zooming is accomplished in one of two ways. The first is by resetting the scale of the X and or Y axes. This is done by selecting the axis with the mouse, then a modal dialog appears to allow the user to set first value, delta values per tic, and number of tics. This also allows for "panning" to expose only a portion of the data. This dialog action is further explained later in this chapter.

Another form of zooming is to allow the user to select a rectangle by dragging the mouse, while

performing a right button click, on the plot. The X and Y axes are then scaled to show only that portion of the plot within the selection rectangle. An example of this action and its result is shown in Figure 3.7.



(a) Selected rectangle (b) Resulting plot

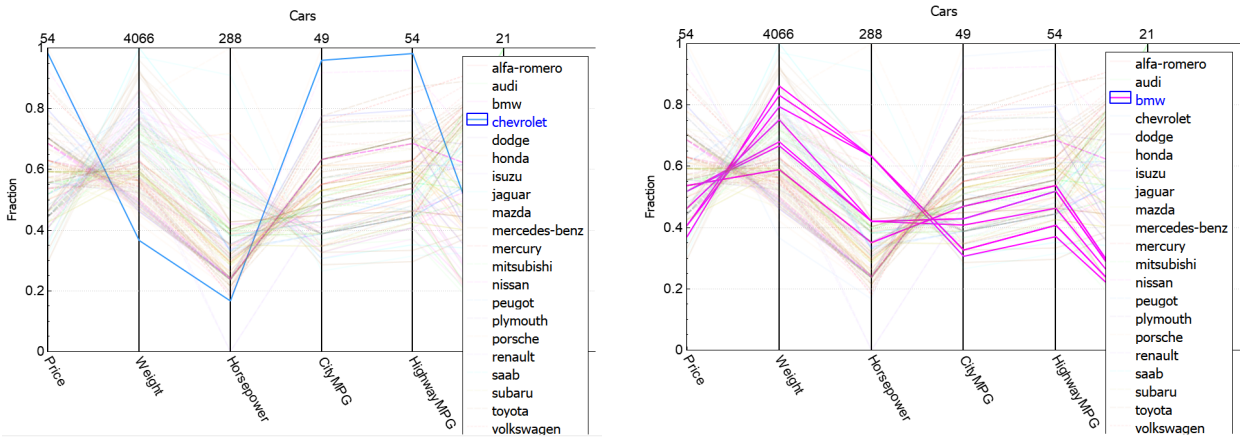
Figure 3.7: The effect of zooming to scale the axes values and enlarge the portion of data viewed.

It should be noted that this type of zooming is not available for the parallel coordinates plot. For those parallel coordinate plots, this "rubber banding" is reserved for selecting lines through brushing, used in this way it is a form of highlighting.

Highlighting. Highlighting is used to emphasize some data and to deemphasize data of less interest. A line in a complex plot may be selected with left mouse click. This causes all the other lines or bars to be displayed in a less saturated color so as to appear to be "greyed out", leaving only the selected line or bar in the original brightness. This is a way to selectively view some data. This is shown in Figure 3.8(a). This may be considered as an addition to *zooming* or *filtering*.

When the displayed data is divided into classes, the records are labeled with the class name rather than a unique variable name. In this case, all lines of a class may be selected by selecting the class name in the legend. An example of this, using the Automobile data set from U.C. Irving, is shown in Figure 3.8(b).

Brushing – Parallel coordinate plots only. Brushing is the process of interactively selecting



(a) Single line selected

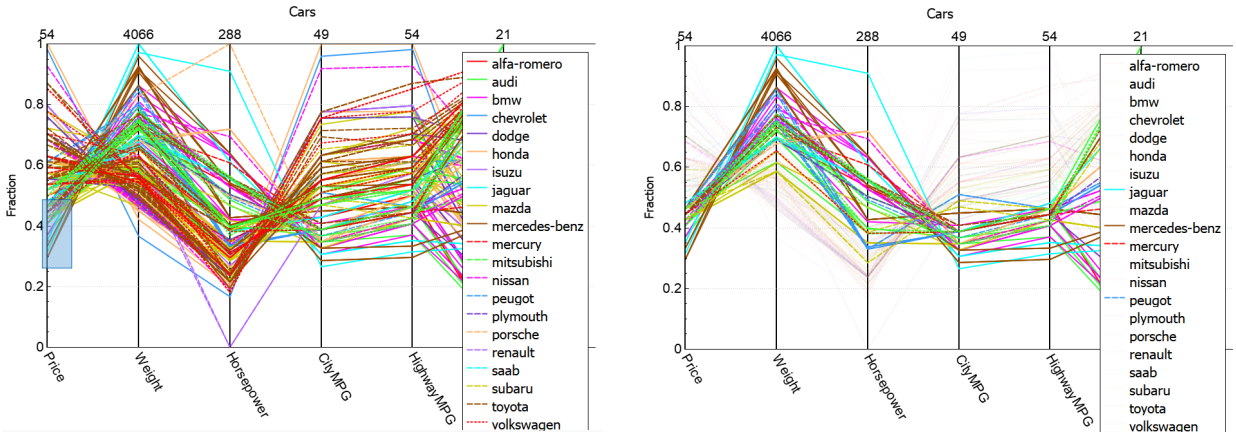
(b) All lines of a class selected

Figure 3.8: In (a) a single line is selected and the class of the line is shown highlighted in the legend, in (b) the legend has been selected and all lines of that class are highlighted.

data items from a visual representation. The original intention of brushing is to highlight brushed data items in different views of a visualization [24]. Brushing is used in various ways, for example, sometime in scatter plots to select a group of points [25]. We use it here with parallel coordinate plots to select a group of lines. This is done with a right mouse click and dragging a rectangle over a segment of one of the coordinate axes. The lines within the selected portion will be highlighted for the entire plot. This may be considered an operation of *zooming* or *filtering*. It may also *relate* items with similar values of an attribute.

In Figure 3.9(a) the lines are selected on the far left hand axis. In Figure 3.9(b) the selected lines are highlighted.

Removing variables. Plots with many lines may be more easily understood with some of the lines removed. A dialog is displayed that shows a list of variables and some may be selected for removal from the plot. For line data this is aided by a matrix of scatter plots showing each variable plotted against the others. This is shown in Figure 3.10. A correlation coefficient is calculated and the scatter plots are sorted by decreasing correlation coefficients. This is to aid in identifying variables that may be redundant. Variables may be selected by checking them in the list on the right and the variable is not plotted. This is an example of *Filtering*.



(a) Select low price cars (left hand axis)

(b) With highlighting we see other features of those cars selected.

Figure 3.9: Brushing: The rectangle on the far left axis left of (a) shows the area that needs to be zoomed to see the data of interest. We can then see the values of other attributes of those selected.

Viewing details as numeric values, viewing metadata. It is possible to link a view of the data in numeric form to a point on a plot. This capability is switched on via clicking a push button. A line is selected via a left mouse click, and the "X" and "Y" values for the nearest point, the point before in the list of plotted points, and the point after are also shown. If metadata were included in the input, the metadata are also displayed on this dialog. This is shown in Figure 3.11. In this particular display we see that frequency 110 has been selected, also shown are the values for the adjacent frequencies. The metadata in this example case are the same for every point and list the expected peak frequencies.

In this system the plotting library sorts the points according to increasing abscissa values before plotting. Therefore the point before and the point after are not necessarily the corresponding points in the original data.

3.4.1 Annotations, Adding Information to the Display

There are two other types of highlighting that may be initiated by the preprocessor and used to enhance the display by the graphics program. For the first type, a second metadata file is read

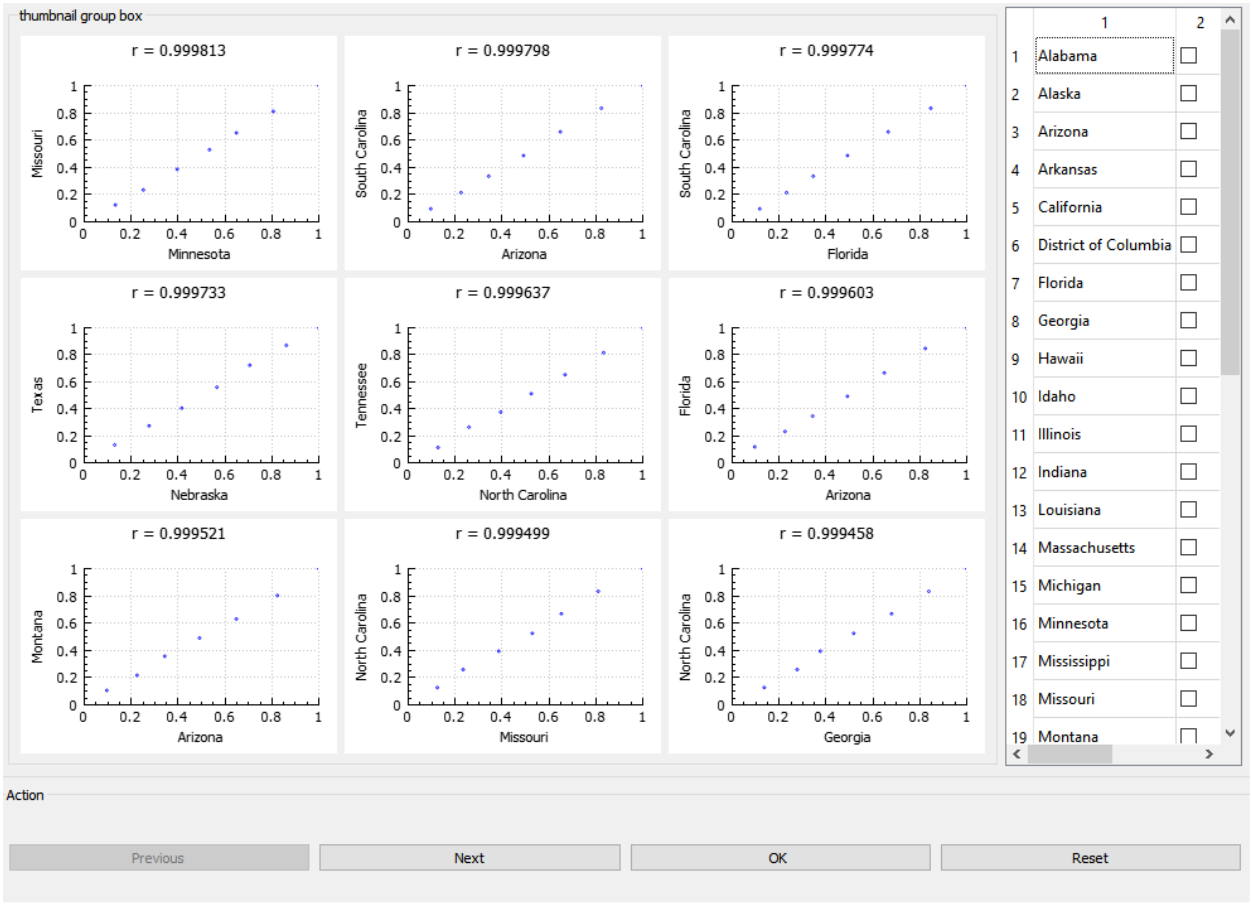


Figure 3.10: A dialog showing a matrix of scatter plots with correlations coefficients. Variables that may be removed are in the list of check boxes on the right side. Checking the box causes them not to be displayed.

that contains points that are displayed as glyphs. The details of this data input are described in Appendix A. In the implementation here, the preprocessor generates points to denote the location of data that exceed a certain maximum value and those that are below a specified minimum value for each variable.

In Figure 3.12, the highlighting cues us to the area of the data line that needs to be zoomed to have a detailed inspection. The marks are placed to show data that is below a certain value. After zooming we can easily see the problem, i.e., the data is below the minimum value.

The other type of highlighting is a change of intensity of points plotted. For data in pairs, a third column of metadata may be provided by the preprocessor to emphasize (or deemphasize) certain

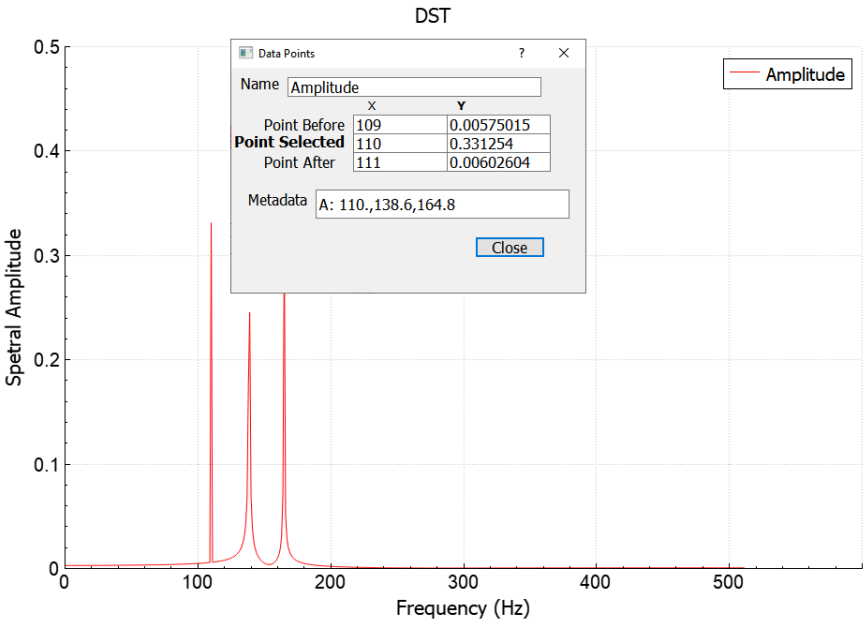


Figure 3.11: Dialog showing coordinates at a selected point, the point before, the point after and the metadata string for the selected point.

points. In Figure 3.13 we see points representing the fraction of time that a traffic signal spends in Red light to Green light as a fraction of the total cycle time (Red+Green+Yellow). The lighter colored dots were not generated during peak traffic times (morning and afternoon rush hour). The data were only recorded in the day time, so there are fewer data points recorded outside of the "rush hour" time than might be expected.

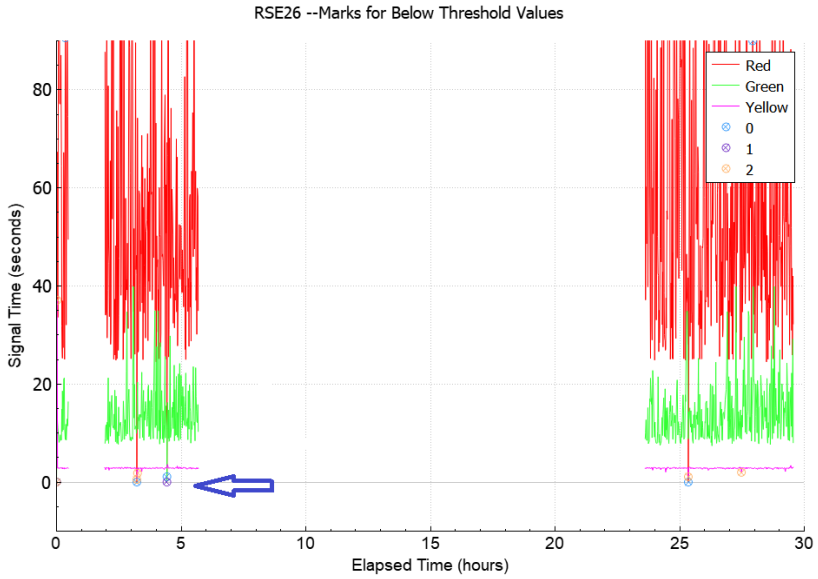


Figure 3.12: The marks (one is noted by the arrow) show the data that are below a certain threshold value. This can be zoomed to see the problematic data. The legend shows the marks for data series 0 to 2.

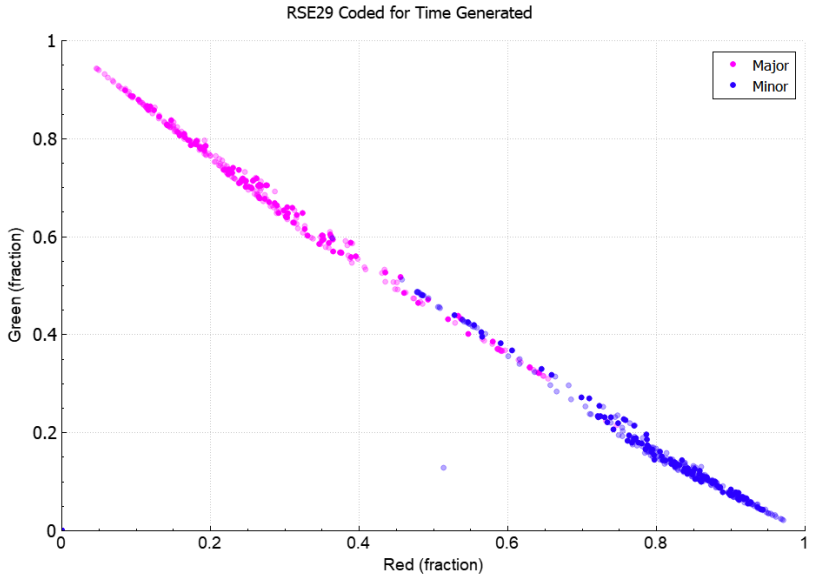


Figure 3.13: Plot using intensity to distinguish points. The ratio of time spent in Red light vs. Green light for Major and Minor road at an intersection. Darker colored points were generated during rush hour.

3.4.2 An Example Workflow

To illustrate the workflow of Overview, Zoom, Details-on-demand, and to illustrate the user interactions associated with this workflow, I present a short example below. This is using data from the case study described in Chapter 4, the background of these data will be explained in detail there. The data used here are from traffic lights at an intersection, one for the intersection Major street, and one for the other street. We have data that are a time series of states of the lights, the preprocessor assigns .5 when the light is Red, 1.0 when the light is Green, and 1.5 when the light is Yellow. So we expect to see the light change from 1.0 (Green), up to 1.5 (Yellow), and down to 0.5 (Red) and repeat. When the Major road signal turns Red we expect the Minor road to turn Green. The time series for the Major and Minor roads are plotted on the same figure.

In Figure 3.14 we see the main window of the program. In this figure a file has been opened, by the "File/Open" menu selection, and the filename is displayed, and the button marked "Plot" is the only one enabled. This graphics program makes available certain kinds of plots based on the input data type. This data type is series data so line plots will be the default.

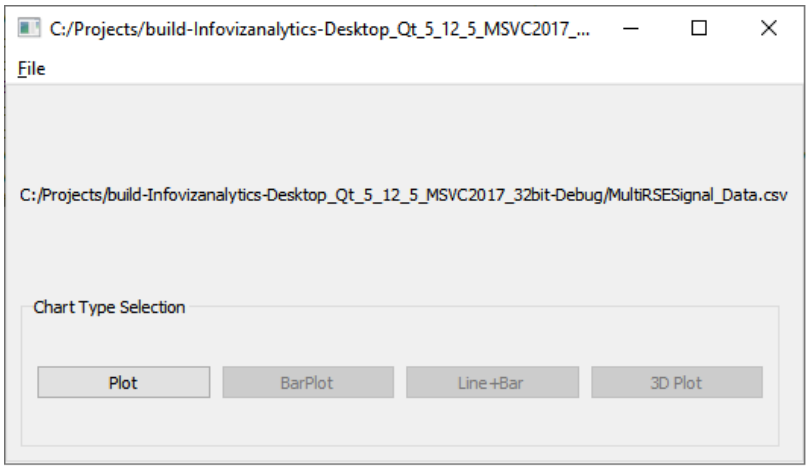
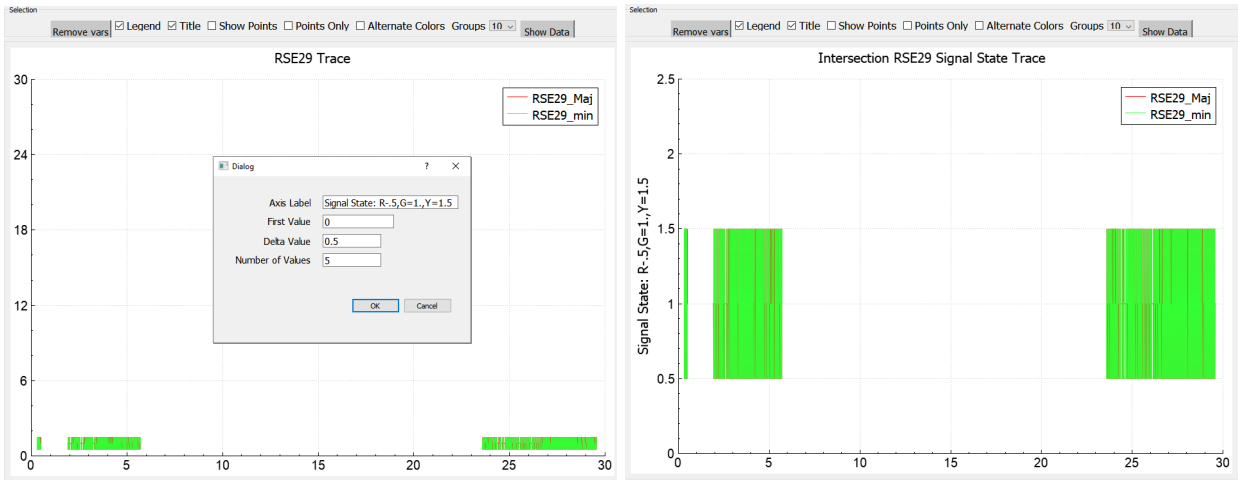


Figure 3.14: The program main window, the name of the opened file is displayed.

After pressing the Plot button, a plot with default settings is displayed. A dialog can be opened by clicking on the vertical axis to reset the scale. This is shown in Figure 3.15(a). After changing

the scale, and labeling the axis, we double click on the title to give more meaningful text. The result is shown in Figure 3.15(b). We are beginning to see some structure of the data.

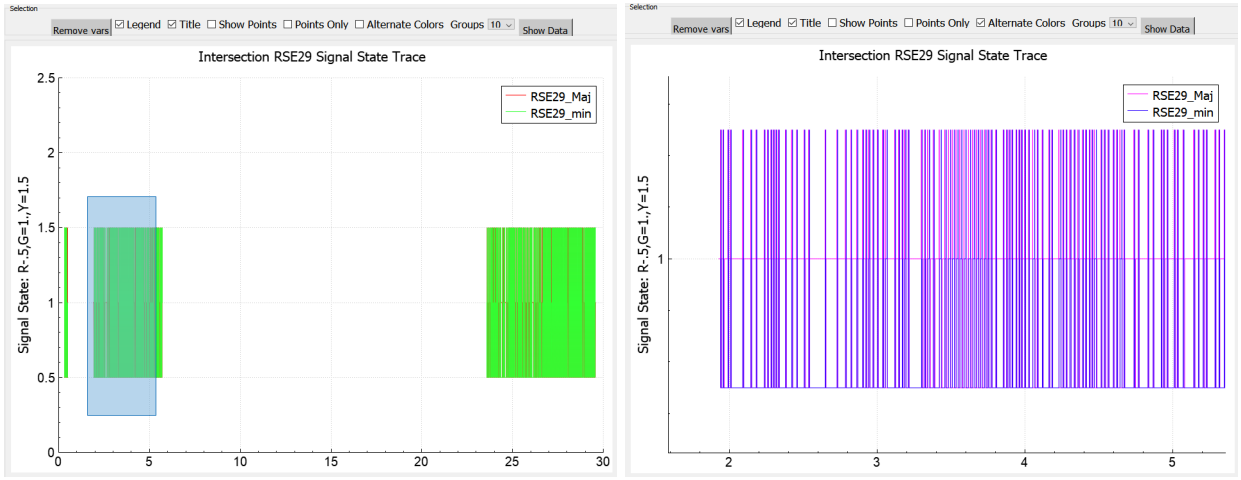


(a) The defaults, title, dialog to reset axis

(b) Vertical axis and a more meaningful title

Figure 3.15: In (a) a plot with the default settings is shown, and a dialog has been launched to set the label, first value, delta value, and number of tics for the Y-Axis to get a first look. The title is embellished. The result is shown in (b).

In Figure 3.16(a), we show a region that we wish to inspect more closely. The blue rectangle is the result of a right click and drag to "zoom in" on an area. In Figure 3.16(b) we see the result of that zoom and decide to change the color scheme, note that Alternate Colors box is checked.



(a) The default axes, colors, title, dialog to reset axis

(b) After zoom we see more structure

Figure 3.16: An example of zoom operation: zoom area shown in (a), the result with more structure in (b). The color scheme has been changed also.

We see an interesting area of data is between 2.5 and 3.0 hours along the X axis. With another use of the Axis dialog we can not only set the axis first value and delta value to further zoom in, we also provide an axis label.

In Figure 3.17 we see the result of turning on the display of the individual data points (not necessary but it shows a little more detail). In addition, "Show Data" has been selected via the button shown; the button is now in red. A point has been selected at the end of the long stretch of Green light for the Major road starting at about 2.55 hours and ending at about 2.65. The details displayed in the dialog show that the point is at time 2.6475 hours. If needed, the starting point could be sampled and the length of time spent in the Green light could then be computed.

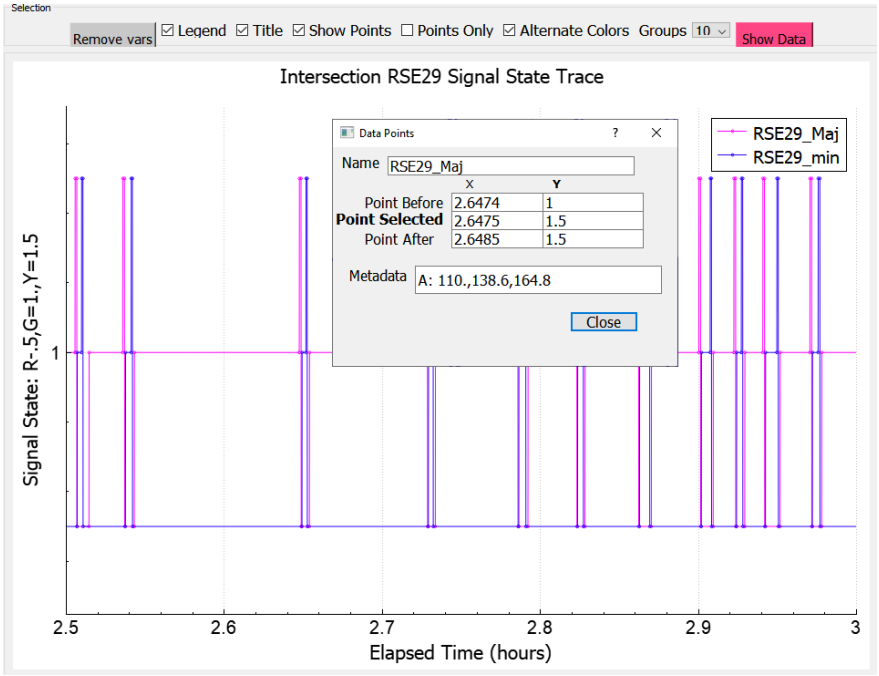


Figure 3.17: Plot with "Show Data" activated and dialog showing selected point.

In this brief demonstration I have shown how the program can be used to satisfy the: Overview first, Zoom, Details-on-demand workflow for data exploration.

The source code for the preprocessors, and the graphics program is stored on Github. A few sample data files as well as an archive of this thesis source may be found at:
https://github.com/glenntu15/ThesisArchive

Chapter 4

Visual Analytics for a Multi-Modal Intelligent Traffic Signal System

The previous chapter demonstrates the design of my visual analytics system and the application of its functionality using a number of simple example data sets. In this chapter, I apply the developed analytics system for the processing and interpretation of a multi-model traffic signal data to detect possible tampering. The data used are from the Multi-Modal Intelligent Transportation Signal Systems (MMITSS) study, described below. Timing data for traffic signals at six intersections is examined, but focus here is on one intersection as a representative example on the use of my system. The intent of the tampering may be to cause malfunctions of the equipment or may be to provide some advantageous signal timings.

With my visual analytics system, the overview of the data is shown by visualizing the summary information of the traffic light pattern within a given time period. Zooming is then performed to locate areas of interest in the data for a detailed inspection, and details-on-demand is used to find specific data values. With my system, I successfully identify a few issues in the raw data (including the missing entries and the mis-placed or mis-aligned entries). I am also able to reveal different traffic light patterns for intersections formed by Major and Minor roads.

4.1 Background of the Data

The data for the study is described in a Department of Transportation document and was produced as part of a study of autonomous vehicles. There are several data sets, but I specifically focus on the one described in a document section titled: "Detailed Description for Signal Plans for Roadside Equipment (RSE) Data", from the document "Multi-Modal Intelligent Traffic Signal Systems (MMITSS)-Sample Data, from Anthem Arizona". [1] These data are traffic signal data, and "Phase ring" data at a number of intersections. Each intersection is identified by an RSE (Road Side Equipment) number. The data were collected from March 3, 2015 to March 4, 2015.

The MMITSS document provides a map of the locations of the intersections where the data were collected. The test bed is described as six intersections extending from west to east. See Figure 4.1 for the map of the intersections (highlighted by circles). The six intersections are labeled as RSE25, RSE26,..., RSE30 from the bottom left to the top right, respectively. There is no signal data for the secondary road (or Minor road) at the intersections RSE27 and RSE28 (corresponding to the third and the fourth circles). Minimal examination is made for these two intersections.

4.2 Data Cleaning

The data were provided in a .csv (comma separated values) file containing 1.2 million rows and 23 columns. The last field of these data is an identifier that gives the name (RSE number) of the collection location. There are time stamps in the first column in epoch time (seconds since January 1, 1970). By examining the data we see multiple records with the same time stamp and the same RSE field. In addition, the records are not in strict chronological order. My first operation was to sort the records based on their time stamps. However, there were too many records to sort in Excel, which loads only up to a million rows, so a Python program was used to read the original

[1] The document describing these data, supplied by the investigators, is posted in the repository: https://github.com/glenntu15/ThesisArchive/tree/master

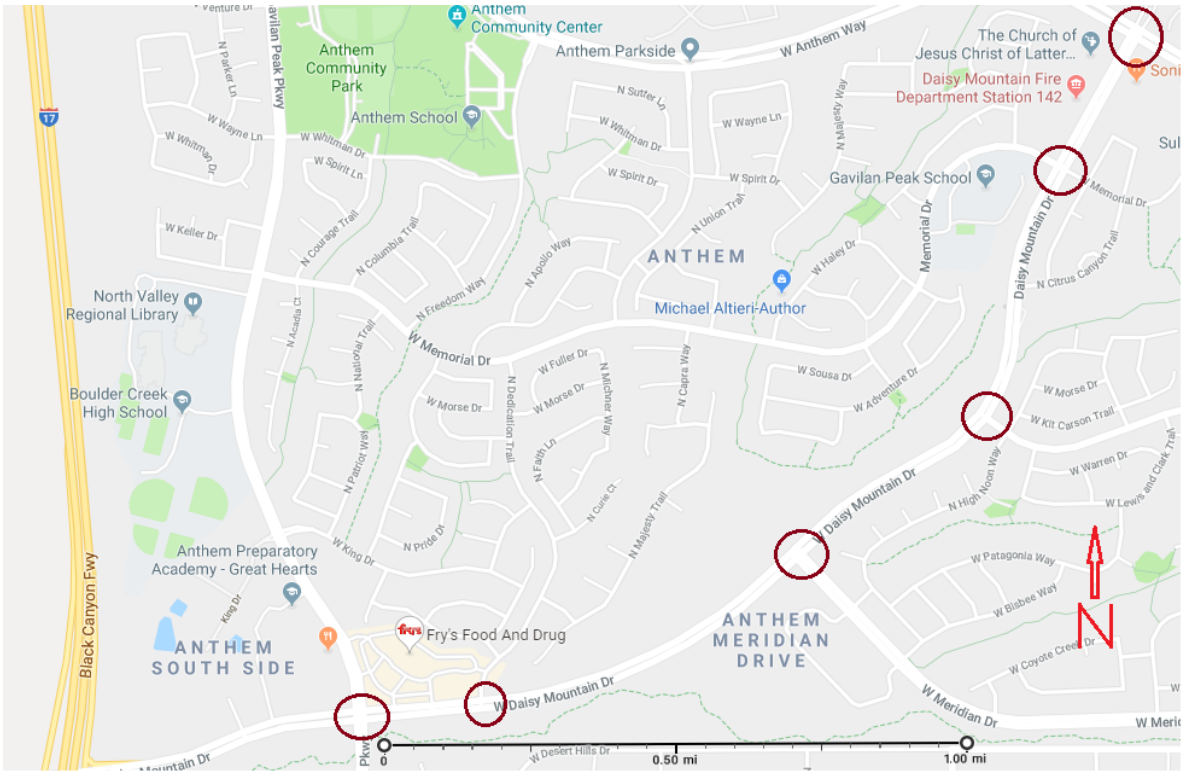


Figure 4.1: Map of a portion of Anthem Arizona showing the location of the test intersections based on map provided by MMITSS documentation which had been taken from Google 2015. The first intersection (bottom left) is at **latitude 33.8, longitude -112.1** and we see a scale of 1.0 miles, per Google Maps 2019.

data and split it into six files (in .csv format) one for each intersection. Each of these six .csv files was then sorted in Excel.

After sorting all the files, the smallest start time (epoch) was noted and set as a constant in the preprocessor program. This was subtracted from all times so the times for all locations were based on the same zero point.

In the data there are two columns for the traffic signal on the Major street, and two for the Minor street that are important. [2] The first column is the signal state, 1 = Red, 3 = Green, and 4 = Yellow. The other column is the number of seconds (given to two decimal places) that the signal has been in the current state. When the data are sorted in Excel to get the correct chronological

[2]There are actually more but they don't really provide additional information.

order, the records with the same time stamp are not necessarily in the correct order. We see a signal change state when the time stamp changes, then change back a few records later with the same time stamp. This may be because the sort in Excel is not stable. A stable sort keeps records in the same order when the key is the same. Another possibility is that records are just not in the right order. We know some time stamps are out of order by inspecting the original file.

To correct this, the preprocessor, when extracting the data, accumulates records with the same time stamp (at the same second) and sorts them to give a continuation of the signal seen in the previous second. A secondary sort on the time field (time in that state) is performed as well. When the data are sorted, it is easy to take the last time of a signal state when a signal changes to the next state. We can see the time spent with the signal showing Red, Green and Yellow. With some novel visual representation of the data, it was later discovered that this cleanup was not sufficient, which will be explained in a later section.

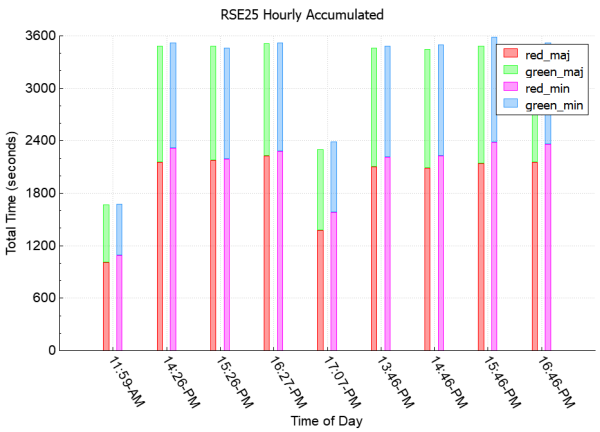
4.3 Approaches to Analysis of the Data

4.3.1 An Overview of the Data

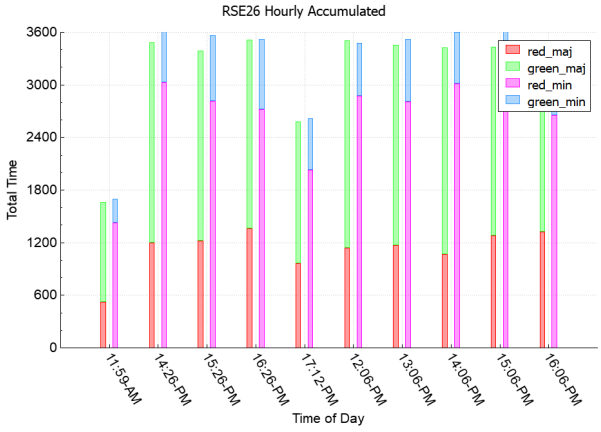
In this examination of the data I accumulate the time in each signal state for a "wall clock" hour. That is, when an hour (3600 seconds) has elapsed the preprocessor outputs the total time spent in each signal state. The epoch time is converted to Hour, and Minute in local (Arizon - Mountain) time. It is reported for the time at which the hour ends. The total times do not equal 3600 seconds because Yellow light times were not included. Even considering variation attributable to this, we see at some times the reported total signal time is smaller than the others. We later learned that this is due to some missing data and the way we define an hour in processing. This will be seen in the subsequent plots.

Figure 4.2 shows the accumulated times for the intersections RSE25 and RSE26. For RSE25 (Figure 4.2(a)), we see that the Red light time is longer than the Green light time for **both**

the Major and Minor streets. This is unexpected and cannot be explained without other information. For RSE26 the plot shows longer Red times in the Minor street than in the Major street, which is expected. Note that in these summary plots the signal times are reported at specific day and time. This is a first look to investigate time of day variations. Some additional bar plots



(a) RSE25 Major street (left bar) and Minor street (right bar)

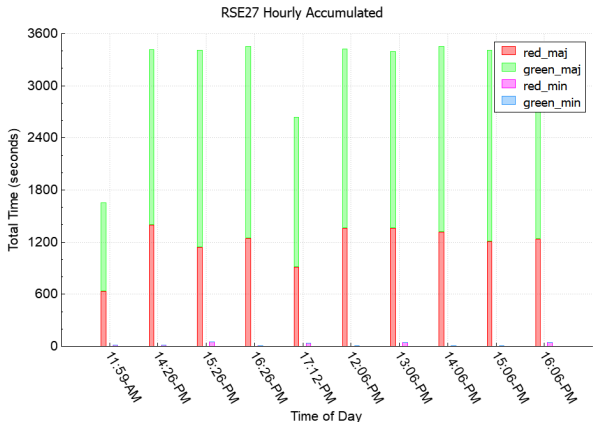


(b) RSE26 Major street (left bar) and Minor street (right bar)

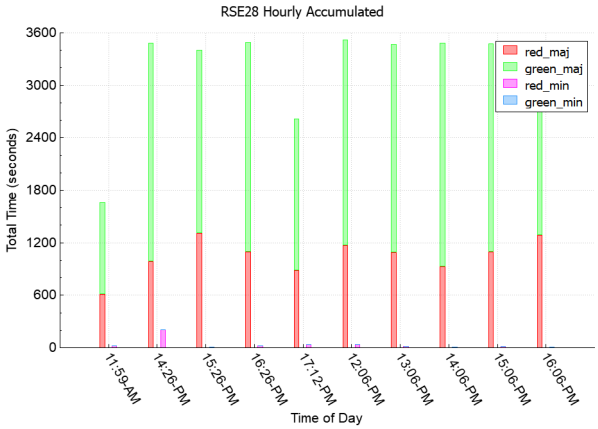
.

Figure 4.2: Hourly accumulated signal time for RSE25 (a) and RSE26 (b). Note that Red light times are longer than Green for both Major and Minor streets in RSE25, but only for the Minor street in RSE26.

are shown in Figure 4.3, RSE27 and RSE28 do not have data for the Minor street, but the Major street shows longer Green than Red times, which is expected.

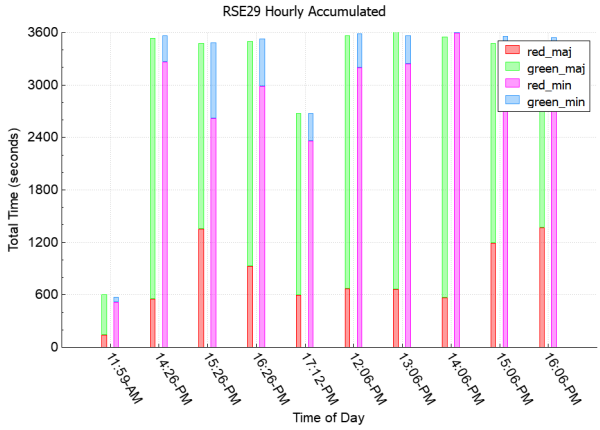


(a) RSE27 Major street – Minor street data is largely missing

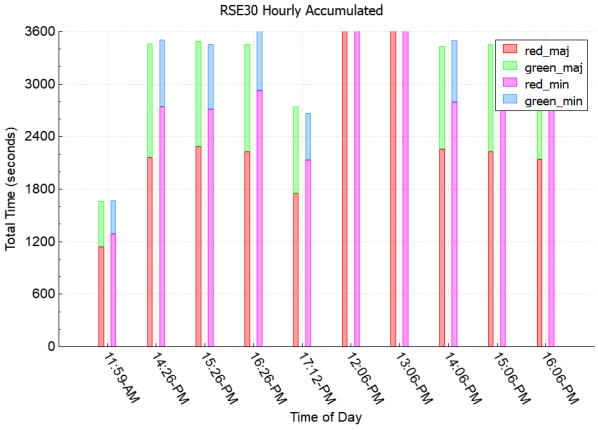


(b) RSE28 Major street – Minor street data is largely missing

Figure 4.3: Hourly accumulated signal time for RSE27 (a) and RSE28 (b). Note that the data for the Minor street is mostly missing, but shows up as small value histograms on this plot.



(a) RSE29 Major (left bar) and Minor (right(bar)



(b) RSE30 Major street (left bar) and Minor street (right(bar)

Figure 4.4: Hourly accumulated signal time for RSE29 and RSE30

What is not apparent on the RSE30 plot, is an obvious error in the data, i.e., two of the bars are going off scale. I expand the scale by a factor of 15 to see the actual size of the bars in Figure 4.5. It was determined by examining the data file that for a segment of the time series, data in two columns were swapped, which causes the error. This is rectified in subsequent data.

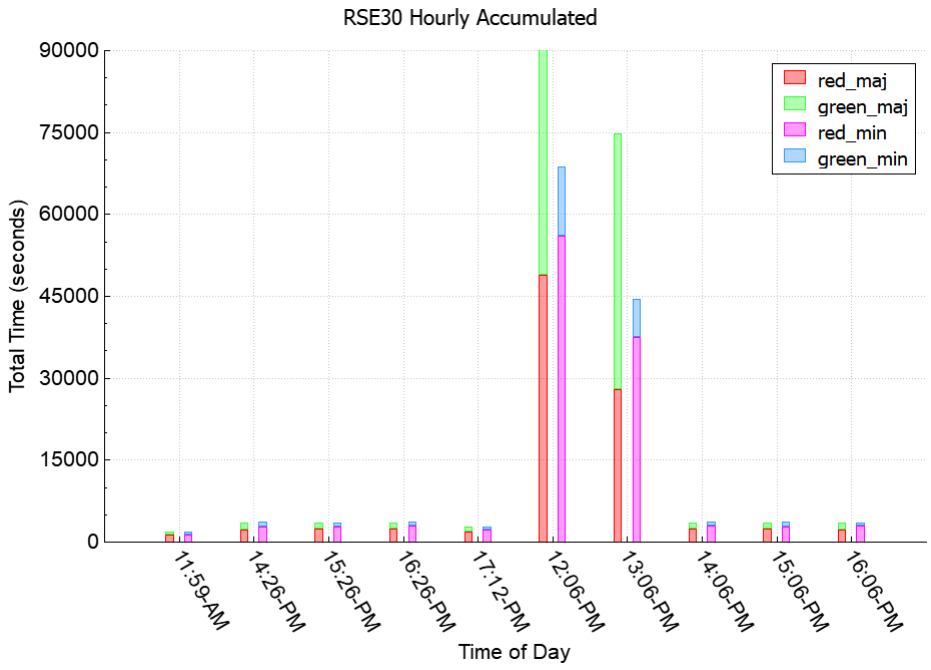


Figure 4.5: Greatly enlarged scale to show error in two hourly sums for RSE30.

4.3.2 Total Cycle Time

Next we look at the total cycle time. The purpose is to see if the total cycle time (Red+Green+Yellow) is a constant or with certain pattern over time, and if there are deviations from the pattern. Figure 4.6 shows the random nature of the observed cycle times. The other intersections have a similar looking plot. We don't see any discernible pattern, so tampering may well be hidden by the randomness of the cycle times.

4.3.3 Greater Detail

We next look at the data in a finer resolution. The preprocessor checks for a complete cycle of the traffic light, and then outputs the time spent in Red, Green, or Yellow states. It saves the "state time" (time in that state) at every state transition. This is reported at the time the cycle starts as hours since the start of data recording.

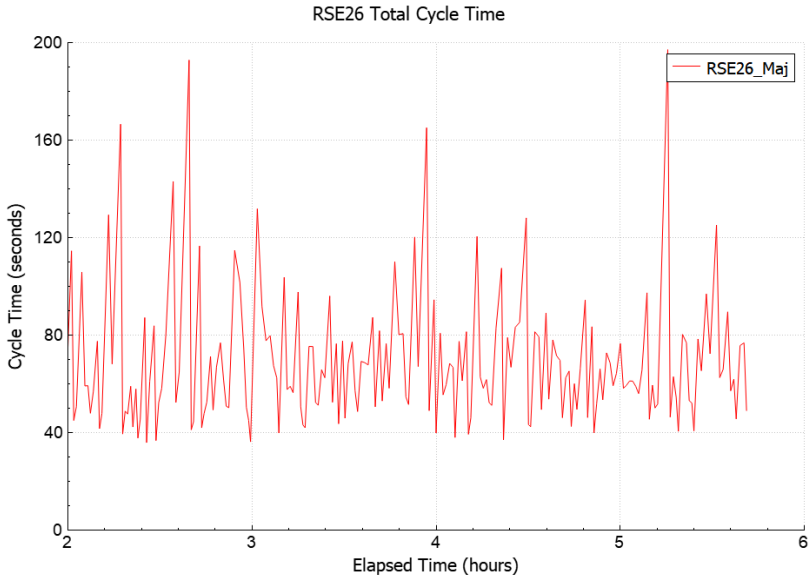
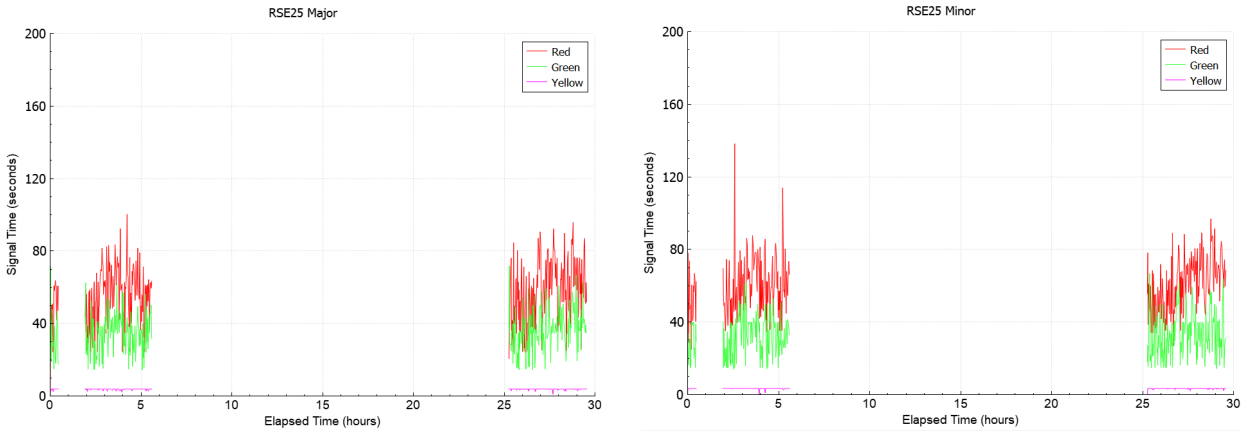


Figure 4.6: Plot of total cycle time (Green+Red+Yellow) as a function of elapsed time. This is a slightly zoomed view to show detail.

Figure 4.7 plots the times for Red, Green and Yellow lights for the Major road and the Minor road for the first intersection, RSE25. One thing we immediately notice is that there is a large amount of data missing, which is the case for the data of all intersections. There is no explanation in the document associated with the data on why this occurred. We note that the signal for the major street spends more time on Red than on Green, this is confirmed by Figure 4.2.

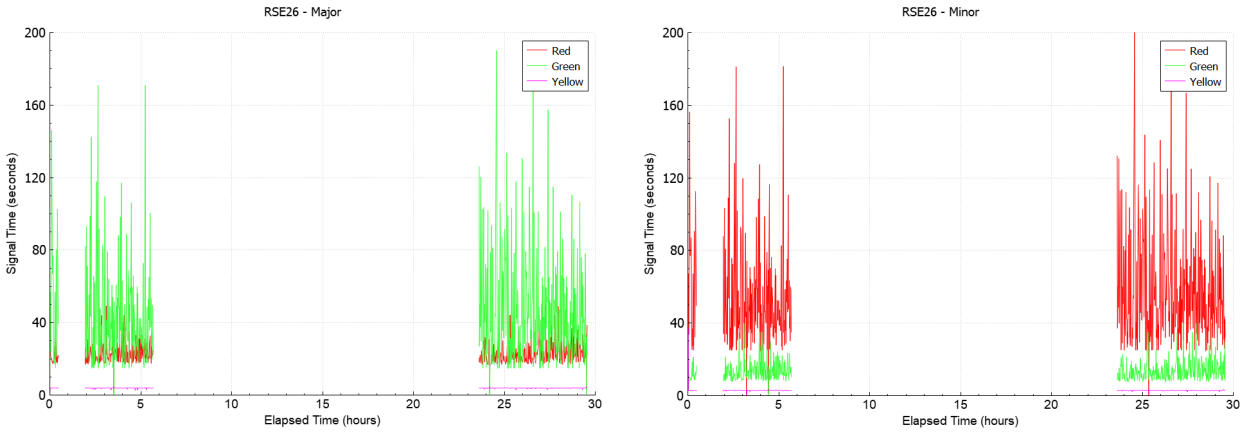


(a) Major Street

(b) Minor Street

Figure 4.7: Comparison of Major and Minor Street Signals for RSE25. The red line is time spent in Red state for each cycle, green line is time for the Green light, and purple line time for the Yellow light.

We then take a look at the data for RSE26, shown in Figure 4.8. In this figure we see that the signal for the Major street spends much longer time on Green than Red. The signal for the Minor street shows Red longer. This is to be expected.



(a) Major Street

(b) Minor Street

Figure 4.8: Comparison of Major and Minor street signals for RSE26. Note the different Red and Green times for Major and Minor roads.

There are some anomalies we can spot from the general visualization. We see a "blip" in Figure 4.9(a) for the Major street at a time of about 3.5 hours. A zoomed image of this is shown as an inset in this figure. Indeed, if we look at the data file produced by the preprocessor, the input file for this graphics program, at time 3.517 we see Red light last for 1.33 seconds, with zero Green light time, while the Yellow light last for 4.79 seconds. This is clearly not normal. We examine the raw data at this time and the findings are shown in Figure 4.10. We see that there is an error in the data in that the Red signal starts in epoch time 1425333723, then in the next epoch time the Yellow signal starts. Within that same epoch second the Red signal resumes.

The far right column of Figure 4.10 indicates the time the signal has been in the current state (Red in this case). We see from the circled values that the time the system has been in the Red state is resuming after being interrupted by a few records showing a Yellow state.

This is clearly an issue of data out of order that is not fixed by the sorting process described earlier. Since an easy correction for this issue is not available, the data for RSE26 is not used for

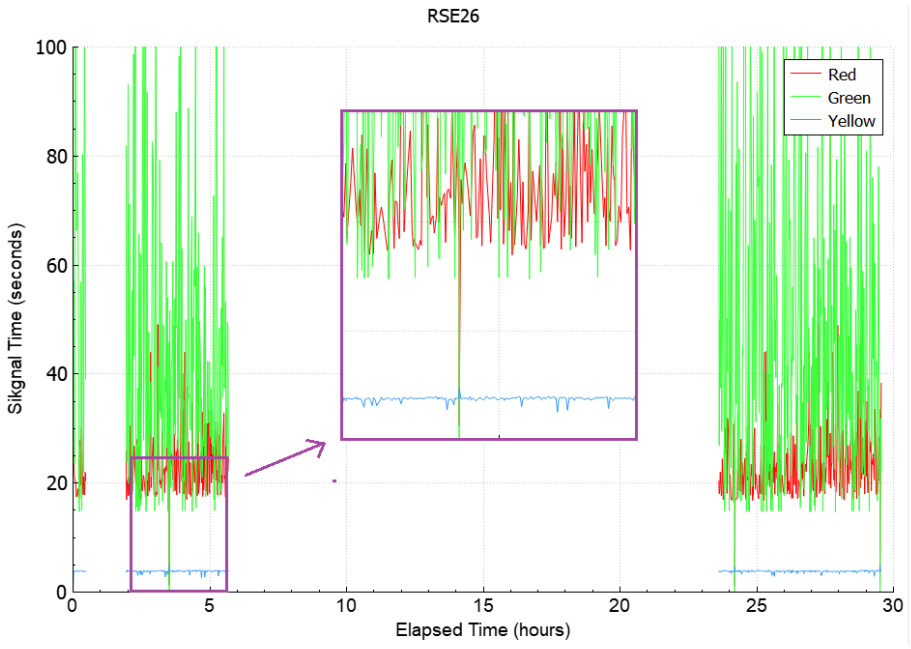


Figure 4.9: An anomaly, cycle time for Green of zero, at approximate time 3.5 hours shown in the inset.

the time being.

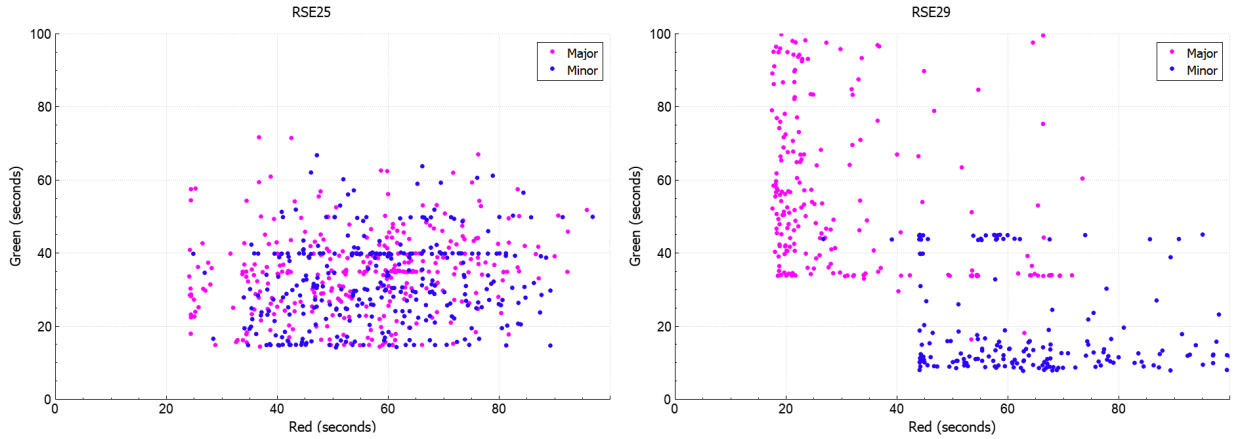
4.3.4 Examination of Signal State Times for Clustering

We plot the time spent in Green state vs the time spent in Red state as a scatter plot of the cycles. This is shown in Figure 4.11. It was thought that some clustering might reveal some behaviour of the signals not yet seen.

We see in Figure 4.11 that the behavior of the signals at two intersections is very different. If we look at Figures 4.2(a) and 4.4(a) in which the bar charts showing the total time per hour spent in Red and Green, we see confirmation of this difference. If we compare Figure 4.2(a) to Figure 4.4(a) we see that the time spent in Red and Green states for the Major and Minor roads is nearly equal. For other intersections, e.g., RSE26, and RSE29, we see a marked difference between the times spent in Red and Green states for the two roads.

1425333722	2	2.69	6	2.69	5748.33	4	2.69
1425333722	2	2.82	6	2.82	5748.46	4	2.82
1425333723	2	3.85	6	3.85	5749.48	4	3.85
1425333723	2	3.99	6	3.99	5749.62	4	3.99
1425333723	2	4.12	6	4.12	5749.75	4	4.12
1425333723	2	4.24	6	4.24	5749.88	4	4.24
1425333723	2	0	6	0	5749.56	1	0
1425333723	2	0.13	6	0.13	5749.69	1	0.13
1425333723	2	0.27	6	0.27	5749.83	1	0.27
1425333724	2	4.39	6	4.39	5750.02	4	4.39
1425333724	2	4.52	6	4.52	5750.15	4	4.52
1425333724	2	4.65	6	4.65	5750.28	4	4.65
1425333724	2	4.79	6	4.79	5750.43	4	4.79
1425333724	2	0.4	6	0.4	5749.96	1	0.4
1425333724	2	0.53	6	0.53	5750.09	1	0.53
1425333724	2	0.68	6	0.68	5750.24	1	0.68
1425333724	2	0.92	6	0.92	5750.48	1	0.92

Figure 4.10: The Yellow (state 4 in second from right column) signal appears interrupted then restarted. This is out of order data.



(a) RSE25 data

(b) RSE29 data

Figure 4.11: Scatter plots of Green vs Red actual times in seconds for RSE25 and RSE29. Note the difference in clustering for RSE25 and RSE29.

For example we do not see clear clustering in the times for Major and Minor roads for RSE25. However, we do see a minimum Red time and a minimum Green time, but they are not distinctly different for the Major road and the Minor road. For RSE29 we clearly see that the Major road has a larger minimum Green time and smaller minimum Red time than the Minor road. We also see that there is a general clustering of times different for the Major and Minor roads.

4.4 The Identification of a Problem

As noted before, the data were sorted before the analysis was done because the epoch times were not in order in the original data. After noting that sorting does not fix the error described above for RSE26, error checking (filtering) was added to the preprocessor to locate cycles that did not contain all three states. In Figure 4.12, we examine data in RSE29 and see a similar problem that was seen in RSE26. When we examine the data error described in Figure 4.10, we see this can be explained by the sorting of the data on epoch times.

In Figure 4.12(a), we see state 3 (Green) exists in epoch time 1425409870 (we will call this 870), then changes to state 4 (Yellow) in the same epoch second. In the next second, 871, we see state 3 again. We know this is a continuation of the previous state 3, because the time in that state (right hand column) is a continuation of the recorded times in the first two records shown. This gives a false cycle, and it is a cycle missing a state.

In Figure 4.12(b) we see the same data from a file that is not sorted. Here the states and times are in the expected order. The cause of the negative times is unknown, but they are not important, because the program uses the last value in this column before the state changes; these negative values are not used in the calculation. Because of this, I decided to use the **raw, unsorted** data for further exploration.

4.4.1 Normalization Shows Additional Features

In order to more effectively reveal abnormal patterns from these traffic signal data, I designed a new plot to visualize the normalized cycle times. In particular, instead of plotting based on the Red and Green light times for each cycle, we see the data based on the ratios of Red and Green lights in their respective full cycles (i.e., Red+Green+Yellow).

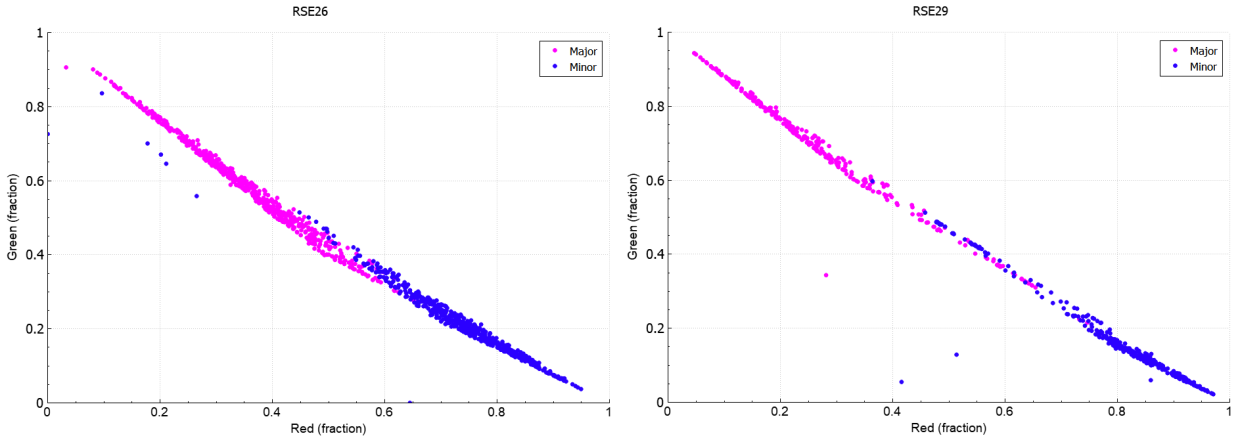
1425409870	3	34.36
1425409870	3	34.5
1425409870	4	-0.87
1425409870	4	-0.74
1425409870	4	-0.6
1425409870	4	-0.48
1425409871	3	34.63
1425409871	3	34.77
1425409871	4	0
1425409871	4	-0.35
1425409871	4	-0.2
1425409871	4	-0.07
1425409871	4	0.06
1425409871	4	0.2
1425409871	4	0.33
1425409871	4	0.46

(a) Sorted RSE 29 data

1425409870	3	34.36
1425409870	3	34.5
1425409871	3	34.63
1425409871	3	34.77
1425409871	4	0
1425409870	4	-0.87
1425409870	4	-0.74
1425409870	4	-0.6
1425409870	4	-0.48
1425409871	4	-0.35
1425409871	4	-0.2
1425409871	4	-0.07
1425409871	4	0.06
1425409871	4	0.2
1425409871	4	0.33
1425409871	4	0.46

(b) Unsorted RSE 29 data

Figure 4.12: Data as sorted and as unsorted files. Note that lack of sorting on Epoch times (left column) results in the states being continuous. In the sorted file the state is interrupted and it appears to be a cycle without a Red light state.



(a) RSE 26 data

(b) RSE 29 data

Figure 4.13: Normalized scatter plots of RSE26 and RSE29

Figure 4.13 shows the normalized plots of the individual traffic light cycles for RSE26 and RSE29. We see there are a few points located outside the cluster of the majority. These are investigated in the next section.

4.4.2 Investigation of Points Outside of the Usual Range

To understand what caused the outliers in the normalized plots, I attached, via the preprocessor, "metadata" to the data points, that include the epoch time the point was created, as well as the Red, Green, and Yellow times as a character string. The actual times that were used in the calculation were included in the metadata to identify in the raw data the cycle that made up the calculation. Using the details-on-demand pop-up box feature, those metadata were used to examine the raw data file and determine the cause of those spurious points.

Upon examination we find in almost all cases there are records out of order. We see this in Figure 4.14(a). The correct position of the data record can be determined by the epoch time, and the time the signal has been in the current state. This record was moved by editing the .csv file, so that the data are in the correct order. There are other instances of this type of data error that are found and fixed.

1425337328	1	2.18
1425337329	1	2.31
1425411430	3	0
1425337329	1	2.45
1425337329	1	2.58

1425411430	1	18.67
1425411430	1	18.8
1425411430	3	0
1425411430	3	0.13
1425411431	3	0.27

(a) Misplaced record

1425321083	1	52.81
1425322247	4	30.07
1425322248	4	31.08
1425322249	4	32.09
1425322250	1	0

(b) Missing Data

Figure 4.14: We see a data record is misplaced then corrected in (a) and in (b) we see that there is a gap in the data.

After these are fixed, only one point appears to be apart from the others. This is shown in Figure 4.15. The point selected is at (.51,.13). We examine this point using the details-on-demand feature for showing data, also shown in this figure. The metadata indicate the epoch time, and the

times in Red, Green, and Yellow states; the string is: **"1425322307:r:46.06:g:11.53:y:32.09"**. The light is in state 4, Yellow, for 32.09 seconds, then it transitions to state 1, Red, followed by state 3 Green. The time in red is 46.06 seconds, and in green is 11.53 seconds. The calculated Red/total is .51, Green/total is .13.

The Yellow state time of 32 seconds is much longer than other cycles and out of proportion to the Red and Green state times when compared with other cycles. It was determined that the cycle ends in the Green state. Other cycles examined have ended in the Red state. Tracing back up the data file until the cycle starts we see the situation shown in Figure 4.14(b). That is, there is a gap in the data which I hypothesize that the Yellow light may be thought of as part of another cycle for which we have no data for the Red and Green times; we see only three records for this Yellow state.

This explains, and mostly fixes, all points seen outside of the main cluster. This is interactive data cleaning, which will be difficult to do otherwise.

4.5 Results

I have shown that anomalies in the data, and a greater understanding of this traffic data may be found by using this visual analytic system. Various plotting and preprocessing techniques have uncovered problems with these data that would have been difficult to find by examining the text files or with conventional plotting of the raw data.

First, the **overview** seen in Figure 4.2 shows the longer Red lights for both the Major and Minor streets for RSE25. The more expected pattern of longer Red lights only for the Minor road in the other intersections is also noted.

Second, the **zoom** showing the detailed Red, Green, and Yellow light times for each cycle as a function of elapsed time can be used to identify where the data was missing. It also shows some anomalies that are the first indication of problems with the data.

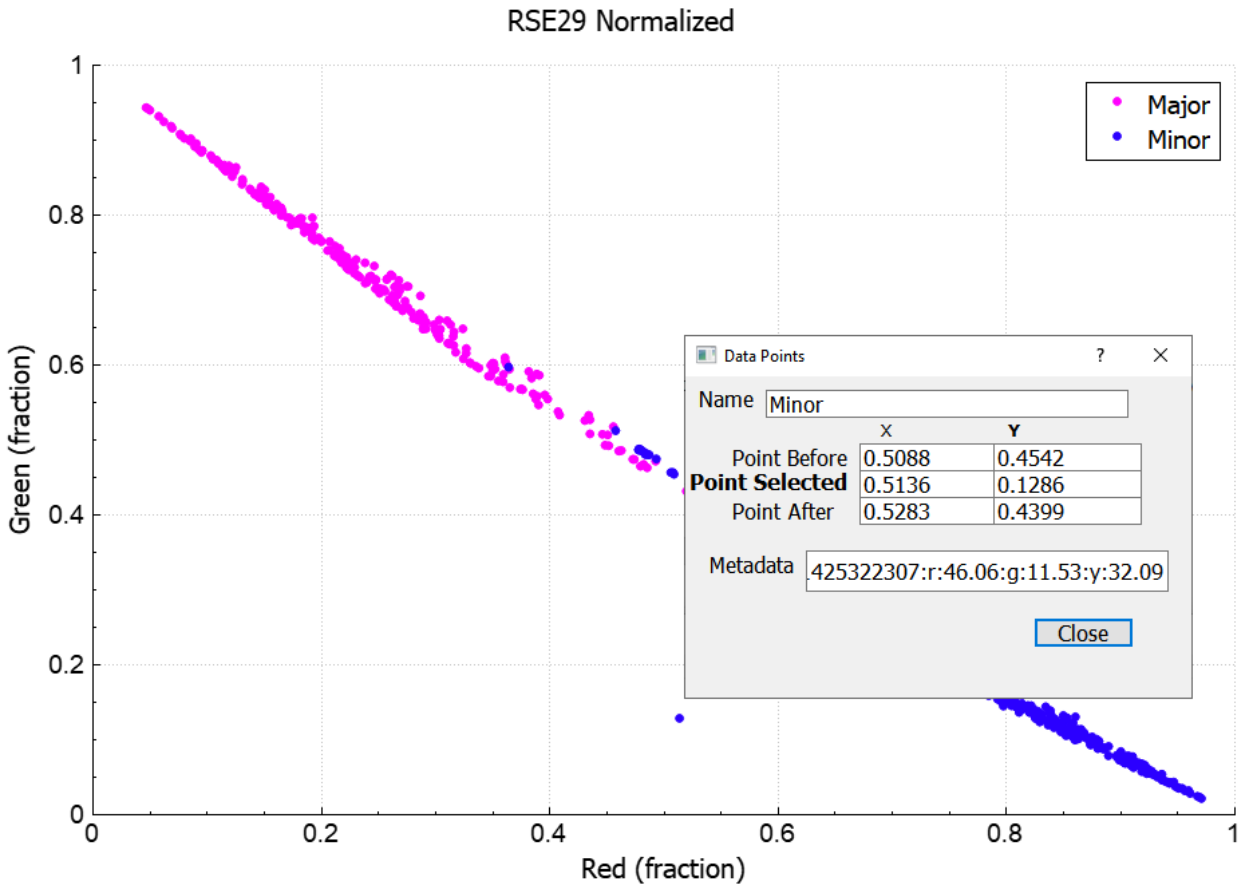


Figure 4.15: The metadata in the dialog indicate the epoch time for the generated point as well as Red, Greet and Yellow state times

Third, the preprocessing filtering gives a different view of the data and the attached meta information supports **details-on-demand**. These details allow the identification of two different types of data errors. The first is single misplaced records, of which there are several instances. The other is due to missing data, data from a partial cycle seem included as part of the next cycle.

Chapter 5

Conclusion

In this thesis work, I have developed a visual analytics system for the exploration of large-scale and complex data. The system design follows the principle of visual analytics, i.e., overview first, then zoom, and details-on-demand. To demonstrate its generality, I have customized and applied this system to the analysis and exploration of a traffic light data set. In this chapter, I will summarize my contributions of this thesis work (Section 5.1) and propose a number of future directions (Section 5.2).

5.1 Summary of Contributions

In this thesis I have presented a visual analytics system that I built to include all of the steps of the visualization pipeline, from cleaning data to final display. This system utilizes simple graphics techniques coupled with user interaction, to realize effective data exploration.

I developed a new suite of preprocessors for the visual analytics system that not only handles data from a variety of sources but also includes calculation of new values for more effective data presentation. Some data cleaning, selection of a subset of dimensions, sorting, and flagging missing data for the eventual display is also performed.

In addition, the preprocessor generates, and the graphics program processes, metadata, or information that is not inherently in the data but included to add to user information about the data. There are three types of metadata. These are: additional points to highlight a portion of the display, coding for intensity added to provide additional information about data points in a series, e.g., time of day they were generated, and a character string added to the data points to provide additional information to the user.

This novel combination of features was applied to traffic light data to examine them for possible tampering. These features were shown to be useful in analyzing that complex and multidimensional data set. In addition they revealed information that was not expected, for example, the minimum time limit for Red and Green signals, different on the Major and Minor roads.

5.2 Future Work

Future work may fall into three categories. The first is to further investigate data used in the case study in Chapter 4 and using this system on other similar data. The second is to enhance certain features of the system to improve the user experience. Specific examples are given below. The third is to incorporate additional capabilities into this system.

5.2.1 Additional Data Investigations

The results shown in Chapter 4 immediately suggest two investigations for the continuation of this work. One is to only consider cycles that start on a particular state. We see the possibility that missing data would cause erroneous calculations by including a "partial cycle". When calculating ratios of Red to Green light times it would be useful to explore at which state a cycle starts and how that affects the pairngs of Red and Green times in a cycle. This could be done by comparing the scatter plots discussed.

It would be useful to apply this system to explore other similar data sets, especially other traffic

control data. I believe the same data views would be useful but different anomalies may be seen.

5.2.2 Enhancements of Existing Features

In addition, we have seen places where additional features would enhance the utility of this system. The graphics program could be greatly expanded. There are several functional features that, if improved, would be more useful. For example, the display of numerical data currently allows the selection of a point on any line. It would be useful to select a line, have the others dimmed, then only select points on the highlighted line. This is because in very crowded displays it is difficult to select the desired point. In addition, the "show value" feature is not available for parallel coordinate plots or for bar plots. It could be worthwhile to be able to show numerical values on both the bar plots and the parallel coordinate plots. Furthermore, the brushing feature, to select multiple lines for the parallel coordinate plot, may be useful for selecting multiple lines on the standard line plots. To achieve that, the brushing feature could be switched on to distinguish it from the box zoom feature when clicking and dragging a mouse.

Another improvement is the way the intensity value for the plotted points is set. Currently the plotting program groups together points with a particular setting and then they are visualized using the same intensity value. For this reason the current program sorts these points into two groups and plots them using two intensity values. The plotting library used could be augmented to overload the plotting function to allow an array of colors, including intensity, to be passed to it. It could use this encoded information to draw many different intensities. The use of different colors could also be employed.

Two other improvements concern the added glyphs. The first one is to modify the plotting library to have additional shapes to use for glyphs. The current ones are not visible enough to attract the users attention. A solid diamond or colored arrow would be more visible. In addition, it should be possible to turn off the glyphs. Currently they are displayed if provided in the input. Second, it would be useful to have metadata added to these glyphs. This could be used for noting

events in a time series for instance.

5.2.3 Additional Features

There are places where additional features would enhance the utility of this system. A few are mentioned here.

The program architecture currently only allows for one data set to be loaded at a time. It may be desirable to have more than one data set loaded to compare them.

It would be useful to be able to save plot configurations. Currently when a plot is displayed the axes labels and axis scales need to be set to get a pleasing appearance. If all these settings could be saved and reloaded, it would greatly facilitate re-displaying a previously viewed plot. In addition, a "return to saved settings" feature would be useful. With this, the user can revert the changed setting to a previous one that is more satisfactory.

The parallel coordinate plots have several dimensions of data to be shown at once. There is often no relationship between two of them, but there may be occasionally. In this case it would be useful to select two variables (corresponding to two dimensions/axes in the parallel coordinate plots) and plot one with respect to another. An example of this can be seen in the parallel coordinate plot of Figure 3.4 with two of the variables shown in the scatter plot of Figure 3.5.

One challenge in developing this system was providing plots not offered by the QCustomPlot classes. QCustomPlot is open source and I compiled the source code as part of the program. This code was not modified, rather "work arounds" were found. For example, the parallel coordinates plot is not a featured chart type like line or bar plots. This plot is achieved by setting the vertical grid lines dark, normalizing the data for each variable/attribute so that the values fall between 0 and 1. On the resulting plot the vertical axis scale is labeled "Fraction". A label is placed at the top of each grid line to give the maximum value, so the plot is the fraction of that maximum value. In the future, new plot types could be defined and written for QCustomPlot.

Another feature that may be useful is to open a separate window to both zoom and pan simultaneously. This window would have a scroll bar to allow the view to cover different portions of the data in the display. This may be thought of as a "magnifying lens" to scan across the data.

Bibliography

[1] M. Hoffman. (1997) Visualizations for high dimensional data mining - table visualizations. Visited 07-17-2019. [Online]. Available: https://pdfs.semanticscholar.org/8a30/92ae419de3e6150e275b0ca782485d0ad75b.pdf

[2] K. Cook and J. Thomas. (2005) Illuminating the path: The research and development agenda for visual analytics. Visited 08-18-2019. [Online]. Available: https://www.hsdl.org/?abstract&did=485291

[3] A. Telea, *Data Visualization Principles and Practice.* A. K. Peters Ltd, Wellesley, Massachusetts, 2008.

[4] S. Few. (2007) Data visualization past, present, and future. Visited 9-12-2019. [Online]. Available: https://www.perceptualedge.com/articles/Whitepapers/Data_Visualization.pdf

[5] X.-M. Wang, T.-Y. Zhang, Y.-X. Ma, J. Xia, and W. Chen, "A survey of visual analytic pipelines," *Journal of Computer Science and Technology*, vol. 31, no. 4, pp. 787–804, 2016-07.

[6] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *Proceedings of IEEE Symposium on Visual Languages, September 3-6 1996*, Washington, DC, 1996, pp. 336–343.

[7] P. Godfrey, J. Gryz, and P. Lasek, "Interactive visualization of large data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 8, pp. 2142–2157, 2016.

[8] S. Few. (2006) The surest path to visual discovery. Visited 11-11-2019. [Online]. Available: https://www.perceptualedge.com/articles/b-eye/path_to_visual_discovery.pdf

[9] M. Bostock and J. Heer, "Protovis: A graphical toolkit for visualization," *IEEE Transactions onVisualization and Computer Graphics*, vol. 15, no. 6, pp. 1121–1128, 2009.

[10] Kitware. (2019) Welcome to paraview. Visited 11/1/2019. [Online]. Available: https://www.paraview.org

[11] S. Liu, W. Cui, Y. Wu, and M. Liu, "A survey on information visualization: Recent advances and challenges," *The Visual Computer: International Journal of Computer Graphics*, vol. 30, no. 12, 2014, visited 9-10-2019. [Online]. Available: https://link.springer.com/article/10.1007/s00371-013-0892-3

[12] Sisense. (2019) Power to the analytics builders. Visited 10-10-2019. [Online]. Available: https://www.sisense.com/

[13] J. Milnik, "How to build a google data studio dashboard," 2019, visited 11-11-2019. [Online]. Available: https://www.socialmediaexaminer.com/how-to-build-google-data-studio-dashboard

[14] T. Software. (2019) Changing the way you think about data. Visited 9-10-2019. [Online]. Available: https://www.tableau.com/

[15] J. Kaur. (2016) Data visualization with ggpplot2. Visited 11-25-2019. [Online]. Available: https://datascience899.wordpress.com/category/ggpplot2/

[16] S. Bentrad and D. Meslati, "Visual programming and program visualization – towards an ideal visual software engineering system," *ACEEE Int. J. On Information Technology*, vol. 1, no. 3, pp. 56–62, 2011.

[17] M. Takatsuka and M. Gahegan, "Geovista studio: a codeless visual programming environment for geoscientific data analysis and visualization," *Computers and Geosciences*, vol. 28, no. 10, pp. 1131,1144, 2002.

[18] MathWorks. (2019) Parallelplot. Visited 9-12-2019. [Online]. Available: https://www.mathworks.com/help/matlab/ref/parallelplot.html

[19] W. McLendon, T. Shead, A. Wilson, B. Wylie, and J. Baumes, "Network algorithms for information analysis using the Titan toolkit," in *44th Annual 2010 IEEE International Carnahan Conference on Security Technology, October 5-8, 2010*, San Jose, CA, 2010.

[20] E. Eichhammer. (2019) Qcustomplot. Visited 2019-04-30. [Online]. Available: https://www.qcustomplot.com/

[21] M. Bostock. (2019) Data-driven documents. Visited 10-31-2019. [Online]. Available: https://d3js.org/

[22] R. A. Fisher, "Machine learning repository/iris," 1988, visited 2019-04-30. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris

[23] J. Schlimmer, "Machine learning repository," 1987, visited 2019-04-30. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/automobile

[24] G. Andrienko and N. Andrienko, "Parallel coordinates for exploring properties of subsets," in *Proceedings. Second International Conference on Coordinated and Multiple Views in Exploratory Visualization 2004, July 13, 2004"*, London, UK, 2004.

[25] A. Buja, J. Mcdonald, J. Michalak, and W. Stuetzle, "Interactive data visualization using focusing and linking." in *Proceedings of IEEE Visualization (Vis91), October 22-25 1991*, San Diego, CA, 1991, pp. 156–163.

Appendix A

Data Types and Formats

Data is entered as a .csv (comma separated value) file. Each file must begin with a descriptor line. The format is DSN,*name*,DST,*type*. The value of *name* is placed as the default title of the plot produced. The value of *type* determines how the program will read the data. Then, the next line contains the column headers that are the variable or category names. Because this is a comma delimited file, all fields must **not** have a comma embedded in them. This includes metadata (described later). The keywords DST, DSN, MARKS and METADATA are **not case sensitive** and are used in mixed case here. The program ignores case in checking these keywords.

The next lines are the variable values. The order of them is determined by the data type. The format of these entries is dependent on the data type and described below. When data values are missing, a "?" should be put in the column. This causes the plotting routine to interrupt a plotted line and resume it again with the next data point.

DST, standard. This has the independent variable in the first column and all dependant variables in subsequent columns. This is seen in Figure A.1.

DSN	RSE25_major	DST	standard	
Elapsed time	Red	Green	Yellow	
0	0	35.37	0	
0.023333333	42.53	71.5	3.54	
0.040833333	27.92	31.42	3.8	
0.058055556	38.74	18.91	3.82	

Figure A.1: Format of a typical data file for **standard** data. Elapsed time is the single independent variable and Red, Green and Yellow are dependent variables.

DST, standardpairs. This is a similar format to **standard**, but the values of the independent variable may be unique to that X,Y pair This data format has columns in pairs, X_1, Y_1, X_2, Y_2, etc. The second column header of each pair is considered the variable name that is placed in the plot legend.

There are three types of meta information that may be entered to give the user additional information. The first two may be added to data types **standard** and **standardpairs** respectively. A third type is part of data type **augmentedpairs**, described later.

When the data type **standard** is used, an additional data file that contains points that are added in the plot as glyphs to show data points flagged by the preprocessor. This input is done by using the keyword "marks" and referencing the file name on the first line of the input file. It is referred to as "marks" data here. This is shown in Figure A.2. The file format for the "marks" data itself is described below.

DSN	RSE25	DST	standard	MARKS	MarksData.csv	
Elapsed ti	Red	Green	Yellow			
0	35.37	0	0			
0.022222	77.99	33.9	3.39			
0.04	41.37	18.46	3.51			
0.053611	28.45	16.42	3.47			

Figure A.2: Keyword and filename for MARKS data. The keyword MARKS is not case sensitive.

File for marks data. This is the file referenced in the **standard** file and has the file name given with the "marks" keyword. The file itself has no keywords. The data is organized by pairs of rows, one pair for each variable. They correspond to the variables in the **standard** file. The first

row is the "X" values, the next is values for "Y". The first column is only used as a placeholder. It must be included for the pair of rows in this file for each variable in the original data, even if that variable has no associated marks. This is the mechanism for synchronization between the marks and the variables in the data. A sample file is shown in Figure A.3. In this file we see the second variable has no MARKS data but the placeholders are there.

t	0.0997	0.3511	0.4242	2.0175	2.2192
one#1	156.29	90.6	112.57	103.23	109
t					
two#1					
t	0.0231	3.2219	3.2486	25.3469	27.4786
three#1	37.1	0.53	1.83	1.08	1.99

Figure A.3: Sample file for marks data. Notice that the second variable does not have marks data but the placeholders are there.

Character metadata. When the data type **standardpairs** is used, there can be an added meta information field as a short character string for each variable. This is indicated by the entry **metadata** on the first line of the file followed by the number of columns of metadata entered. When meta information is used, there can be no missing data points and no missing metadata for a data point. The program checks for these conditions, writes an error message if either are found, and when found the metadata is turned off. The meta information columns are all to the right of the data columns. That is, the data column pairs are read left to right, and the metadata columns follow, in the order of the data. An example is shown in Figure A.4.

DSN	RSE29	DST	standardp	metadata	2
Red_Maj	Major	Red_Min	Minor		
0.4822	0.4675	0.5136	0.1286	3/2/2015 11:51	3/2/2015 11:51
0.2217	0.7389	0.7816	0.1894	3/2/2015 11:53	3/2/2015 11:53
0.2158	0.7556	0.8969	0.0779	3/2/2015 11:55	3/2/2015 11:55

Figure A.4: Data file containing two columns of character string metadata.

DST, augmentedpairs. This is similar to **standardpairs** format but there is a third column of metadata for each data pair. Unlike the character string metadata described above, each column comes after the data pair with which it is associated. This third column of data is an integer.

The program currently uses this data to set the intensity of certain points. The program takes the midpoint between the highest value and the lowest value of this data item. Plotted points are grouped in to higher than, and lower than, this midpoint, and the groups are plotted separately. The points with higher than the midpoint value are plotted at full intensity and the others at a lower intensity.

DST, datagroupseries. This has a variable for each row and has a different value for each entry in the series in each column. The column headers are strings. They are used to label each group on a plot. An example is shown in Figure A.5.

DSN	Population		DST	datagroupseries	
	2010	2011	2012	2013	2014
UnitedStat	309338421	311644280	313993272	316234505	318622525
Northeast	55388349	55642659	55860261	56047732	56203078
Midwest	66973360	67141501	67318295	67534451	67720120
South	114869241	116060993	117291728	118422269	119699966

Figure A.5: Format of a data file for datagroupseries data.

DST, datagroupparallel. This data type is used to input data for parallel coordinate plots. It has a similar format to datagroupseries but the columns are different attributes.

DST, dataclassparallel. This is a special type of data for parallel coordinate plots. In this data type many variables may have the same name. That is what defines the class. An example is shown in Figure A.6.

DSN	Iris	DST	dataclassparallel		
	Sepal_Len	Sepal_Wi	Petal_Length	Petal_Width	Class
Iris-setosa	5.1	3.5	1.4	0.2	0
Iris-setosa	4.9	3	1.4	0.2	0
Iris-setosa	4.7	3.2	1.3	0.2	0
Iris-setosa	4.6	3.1	1.5	0.2	0

Figure A.6: Format of a data file for *dataclassparallel* data.

DST, surface. This is data in the X,Y,Z format. There is a triplet for each point on the mesh, so constant values of X, and Y are repeated. Missing data is not allowed for this data format.