ABSTRACT
The world is producing more data than ever before. Our ability to process this data relies on quality data visualisation tools that are tailored to larger datasets. The problem is that the individual disciplines involved in data visualisation are not collaborating, leading to a situation where data is not being visualised as well as required. This is particularly evident in software. By combining the expertise of each discipline into a suite of user controls, an improvement can be demonstrated. The results indicate a successful combination of expertise, leading to a visualisation that consumes larger datasets, maintains graphical refinement, and provides interactive features, which enables meaningful information to be derived. This contributes towards the need to redress the balance between data production and data processing.

Peter O'Connor - 100113380
University of Derby – School of Computing & Mathematics
6.5 Graduated symbol map.................................55
6.6 Heatmap..................................................57
6.7 Limitations..............................................58
6.8 Summary..................................................59
7 Conclusion and future work...............................60
8 References.................................................62
1 List of figures

Figure 1: Pre-attentive processing example. Source: (Bertin, 1983) ....................................................... 7
Figure 2: Pre-attentive feature types. Source: (Tidwell, 2011) ................................................................. 7
Figure 3: The use of colour as a pre-attentive feature. Source: (Tidwell, 2011) ........................................... 8
Figure 4: Chernoff Faces. Source: (Yau, 2011) ................................................................................................. 11
Figure 5: Flower metaphor star chart. Source: (OECD, 2013) ................................................................. 12
Figure 6: Parallel coordinates correlation types. Source: (Yau, 2013) ...................................................... 12
Figure 7: Novel visualisation techniques. .................................................................................................... 13
Figure 8: The basic data visualisation pipeline. ......................................................................................... 14
Figure 9: Comprehensive data visualisation pipeline. Source: (Fry, 2004) .................................................. 14
Figure 10: Stem and leaf plot. Source: (Tuft, 2001) .................................................................................... 15
Figure 11: The range frame. Source: (Tuft, 2001) ...................................................................................... 16
Figure 12: Example of an embellished visualisation. Source: (Tuft, 2001) ................................................ 17
Figure 13: Example of an infographic. Source: (McCandless, 2009) .......................................................... 18
Figure 14: Heatmap with a poor use of colour. Source: (Fry, 2004) ........................................................ 19
Figure 15: An unrefined Treemap. Source: (University of Maryland, 2003) ........................................... 19
Figure 16: A meaningless pie chart. Source: (Friendly, 2012) ................................................................. 20
Figure 17: Use of non-linear scaling. Source: (Weyenberg, 2011) ............................................................ 21
Figure 18: Visualisation by volume. Source: (Tuft, 2001) .......................................................................... 22
Figure 19: Misuse of perspective in 3D data visualisation. Source: (Jobs, 2008) ........................................... 23
Figure 20: Minard’s flow map of Napoleon’s march. Source: (Tuft, 2001) ............................................... 24
Figure 21: The development environment setup in Visual Studio. ............................................................. 29
Figure 22: Providing the dataset folder location to the controls ................................................................. 30
Figure 23: Visual Studio dataset file properties. ...................................................................................... 30
Figure 24: Visual Studio build error if a parameterless constructor doesn’t exist. ................................. 31
Figure 25: Basic architecture design. ........................................................................................................ 32
Figure 26: Basic UI design. ......................................................................................................................... 32
Figure 27: Example linear transformations. ............................................................................................... 34
Figure 28: Linear transformation implemented in code. .......................................................................... 35
Figure 29: Example parallel coordinates column headings. ................................................................. 36
Figure 30: Setting the parallel coordinates control axes property. .......................................................... 36
Figure 31: Parallel coordinate’s axes index array. ..................................................................................... 36
Figure 32: Windows coordinate system. Source: (functionX, 2010) ....................................................... 37
Figure 33: Modified ListView template for horizontal display ................................................................. 38
2 List of equations

Equation 1: Tufte's lie factor formula. Source: (Tufte, 2001) ................................................................. 17
Equation 2: Linear transformation equation to apply the colour schemes ............................................. 34
Equation 3: Standard normalisation equation ......................................................................................... 37
Equation 4: Reversed normalisation formula to account for the Windows coordinate system .......... 37
Equation 5: Graduated symbol map bubble area equation ................................................................. 41
Equation 6: Parallel coordinates line height percentage equation ......................................................... 49
3 Introduction

We are living in a data driven age where analysts in all areas of human knowledge are drowning in data (Heer & Hellerstein, 2009). The ability to make decisions based on this available data is crucial to a number of areas, particularly business, medicine, national security and disaster management (Keim, et al., 2013). This vast amount of data is also difficult to communicate to the general public (Cukier, 2011), because without any context, people cannot comprehend the complexity of the datasets (McCandless, 2010).

It is estimated that only 0.5% of the world’s data was analysed during 2012 (Gantz & Reinsel, 2012), demonstrating a huge potential of untapped analytical value. Investment in new technologies for sensing, simulating and communicating, as well as a decrease in storage costs, has resulted in a situation where our ability to collect data has outstripped our ability to process it (Heer & Hellerstein, 2009).

Data visualisation – the graphical representation of data (Ware, 2004), is widely accepted as one way to assist in this area (Keim, et al., 2013). Visualisation enables people to analyse large volumes of data (Lee, et al., 2003), and detect trends and patterns which would otherwise remain hidden in a vast table of numbers or text.

With this massive investment in data acquisition technology, data visualisation finds itself at the forefront of the data age, as such it can no longer be an afterthought. The general consensus amongst the experts is that there is room for improvement (Fry, 2004). This is no longer acceptable in the current data age. There is a pressing demand on the field to address this to enable meaningful knowledge to be derived from the data. This situation will be explored to verify this claim, and ways of improving it will be investigated.

As a field, data visualisation involves expertise from many different areas. These will be inspected to ascertain their contribution, and to establish potential areas for improvement. Latest developments in the field will also be investigated to gauge whether these can contribute towards improved visualisations.

Improvements will be demonstrated through the creation of a set of modern data visualisation tools that are suited to visualising the larger datasets that are now being created. This will enable accurate and meaningful data discoveries to be made, a vital benefit in today’s world of data overload. The tools will take the form of user controls, which implement specific visualisation techniques. This scalable approach allows the tools to be used in a wide array of visualisation software that can target more specific and comprehensive scenarios.

Section 4 introduces the field of data visualisation, and proceeds to discuss the current work on the topic. Section 5 details the design and development of the user controls. Section 6 investigates how the user controls meet the goals that are set out in this work, and section 7 concludes by examining the implications of this work and looks towards possible future research.
4 Technical background and research

This section begins by providing a background to the field of data visualisation as a lead in to a more detailed discussion about the subject. The limits of traditional visualisation techniques are then discussed. New techniques are investigated to see if the field is addressing current demands in terms of the large quantities of data that are being produced. Data visualisation as a field involves a lot of different skill-sets, these are investigated to try and find areas for improvement. The limits of contemporary work are then discussed, particularly in the context of software. The section finishes by investigating the impact of new technology on the field in an attempt to bring the discussion up to the cutting edge.

4.1 The power of data visualisation – how it works

The key to how visualisation works lies in the psychology of perception. Humans acquire more information through vision than through all of the other senses combined (Ware, 2004). Therefore presenting data in visual form can aid in the understanding of it.

Visualisation works via an encoding and decoding phase (Cleveland & McGill, 1985). Quantitative data is encoded into graphical form through means such as position, size, shape, symbols and colour. This is then decoded by humans through their visual system. The decoding phase is where understanding happens, so the success of this phase dictates the success of the visualisation technique being employed. The true power in being able to display huge amounts of data on a single graphic is that all of this happens pre-attentively, i.e. without effort on the part of humans. This instantaneous decoding of the geometric patterns and magnitudes of a graphic reduces the cognitive overload which is present when viewing a deluge of data (Keim, 2001). This is an important point to make because the ultimate purpose of data visualisation is to increase cognition (Fekete, et al., 2008).

4.1.1 Cognition

It is worth digressing slightly at this stage to define cognition. Cognition is a term that is widely used in visualisation literature. It is often cited that data visualisation improves cognition (Keim, 2001), (Vande Moere, et al., 2012), but adequate definitions of cognition are hard to come by. Colin Ware is an expert in the psychology of visualisation. He introduces cognition in terms of the human visual system perceiving objects to aid in comprehension, problem solving and decision making (Ware, 2004). Definitions vary depending upon the field, but this definition is relative to data visualisation, and as such, is the definition that will be applied in this work.

4.1.2 An example of pre-attentive processing

Figure 1 demonstrates pre-attentive processing by visualising a set of socio- graphic data from regions in France.
The left hand image requires the brain to store the numbers for comparison, whereas in the right hand image, magnitudes can easily be seen, something is higher in the north-west for example. This is conveyed without effort, it is pre-attentive (Cleveland & McGill, 1985).

4.2 Data representation

This work is focused on the visualisation of quantitative data. Quantitative data is abstract, it doesn’t have a physical form (Few, 2009), unlike physical data, which has a spatial context such as flow/volume data, geographic data (Aigner, et al., 2007), or weather imagery (Fekete & Plaisant, 2002). To visualise abstract data requires suitable visual encodings. The encodings chosen for the data are vitally important in showing contrast between elements to draw comparisons. This can be achieved through the use of pre-attentive features as previously demonstrated. User interface expert Jenifer Tidwell neatly summarises the possible pre-attentive features in Figure 2.
The prominence of the pre-attentive features can be seen without any conscious attention being required. In algorithmic terms, it would be expected that this works in linear time \(O(n)\), that is, the time to find the pre-attentive feature depends on how many elements there are in total. Amazingly this isn’t the case. These features operate at a primitive cognitive level, working in a “massively parallel” fashion (Tidwell, 2011). This is demonstrated in Figure 3 which uses colour as the pre-attentive feature.

![Figure 3: The use of colour as a pre-attentive feature. Source: (Tidwell, 2011).](image)

Pick out the blue objects in the left image and then do the same for the image on the right. Did it take the same amount of time? It doesn’t matter how many red objects there are, the time it takes is constant \(O(1)\).

These visual cues show contrast between elements, allowing people to see patterns, spot trends and identify outliers (data points that are distant from others). This is one of the aims of visualisation (Heer, et al., 2010), meaning these features should be utilised in any effective visualisation (Fekete & Plaisant, 2002).

### 4.3 Types of visualisation

There are many types of visualisations, the choice is based on what is the simplest possible form that conveys the most relevant aspects of the dataset (Fry, 2004). However, this doesn’t mean there is a one-to-one mapping between a data type and a representation, there is a certain level of ambiguity when choosing an appropriate visualisation type. This section examines certain visualisation methods and the data types that they are best suited to.

#### 4.3.1 Tables

Perhaps the most basic form, the table is the standard way to show data using rows and columns. Tables are useful to show all the data, but can quickly become cumbersome when viewing large datasets (Fry, 2004).

#### 4.3.2 Independent quantities

For data that is not connected to one another, or has missing values, a bar chart is the most appropriate visualisation. This was invented by William Playfair (Tuft, 2001) as the year-to-year data he wanted to visualise had missing values, and was a novel way of depicting values where there was no continuity.

#### 4.3.3 Continuous quantities

For data that is continuous, the prime example being time (called time-series data), line graphs and stacked area charts are commonly used.
Line graphs are a series of points connected by lines. The horizontal axis usually represents time which makes it a powerful method of visualising trends over time.

Stacked area charts are similar to line graphs but with filled areas that represent proportions of a whole. An example is a business visualising their total income, with each stacked area showing how each department has contributed to the total.

4.3.4 Proportions
For pure proportional data, pie charts are the most common visualisation. Donut charts are an alternative, they are pie charts with a hole cut out of the middle. Values are therefore judged by arc length instead of area (Yau, 2011), which many researchers believe is easier for humans (Cleveland & McGill, 1985).

4.3.5 Correlations
If data has two variables then these can be plotted on a grid. The most common way to do this is with the use of scatterplots or bubble charts.

Scatterplots are a cloud of points plotted with horizontal and vertical locations based on their values. This is useful to spot correlations between variables and to spot trends.

Bubble charts are similar to scatterplots but they can represent extra dimensions through bubble size, colour and texture.

4.3.6 Hierarchies
Tree diagrams are often used to show hierarchical data through connected lines or branches. Each item has a link to a single parent, except for the root item.

4.3.7 Networks
Similar to hierarchical data, network data can be connected arbitrarily, rather than a single parent-child relationship. Graph visualisations are the ideal way of visualising networked data. Rather than a pure hierarchy, it is a collection of nodes and branches that connect between them (Fry, 2004).

4.3.8 Cartographic
Data that is relevant to geographic location is best visualised using a map. An example is plotting latitude and longitude. The spatial relationship is what makes this visualisation technique so effective.

4.4 The scalability problem
The problem with these visualisation types is that they are extremely basic. They do not represent larger datasets very well. The larger the dataset, the tendency is for the visualisation to look cluttered and hard to distinguish. For example, pie charts should be limited to only a few categories before readability is an issue, and stacked area charts become practically useless when there are a lot of stacks, due to the layers becoming very thin (Yau, 2011). Hierarchical data can be hard to visualise all at once due to high and small fan-outs, which lead to broad and deep visualisations that waste space, to the detriment of readability (Shneiderman, 1996). Different visualisation techniques are available that are more appropriate for larger datasets.

4.4.1 Small multiples
Instead of stacking area charts, the areas can be split into their own charts and arranged next to each other for comparison. This technique is very versatile as it can be used for almost any type of visualisation (Heer, et al., 2010).
4.4.2 Treemaps
Treemaps were invented by Ben Shneiderman as a way of visualising hard disk directories (Shneiderman, 2013). Tree diagrams grew too large to be useful because of their intrinsic problem of wasted space, so a space-constrained layout was designed using rectangles. The name treemap comes from the notion of turning a tree into a planar space-filling map.

The area of the rectangle encodes the size of the file. Although many traditional data visualisation experts conclude that the use of two dimensions to show one dimensional data should be avoided (Tufte, 2001), they never tackle the unique problem of displaying hierarchical data in a space saving manner. Rectangle area is a single dimension, the length and width of the rectangles are obviously inherently linked, but they also depict the hierarchy of the data, which makes this an acceptable technique.

4.4.3 Choropleth Maps
Plotting latitudes and longitudes is limited in the amount of data dimensions it can represent. The amount of geographic data has increased significantly recently partly due to the rise in mobile location services (Yau, 2011). With such a vast amount of data to visualise, it is common to aggregate data into geographic areas such as regions, states and counties.

One approach to display this aggregated data is with a choropleth map. Here, geographic areas are colour encoded, with the varying shades depicting a change in value.

However, the underlying area of the shaded region can skew one’s view of the data. Larger regions appear more important as they cloud smaller regions. A potential solution is to change the encoding from colour to size, i.e. a graduated symbol map. Here, symbols are placed on the map and sized according to certain variables (Heer, et al., 2010). This method also allows for a finer granularity than colour shaded, aggregated data (Few, 2009). These symbols can be simple shapes, or more complex, such as pie charts, enabling them to encode further dimensions.

4.5 Multivariate data
Encoding numerous data dimensions (or variables, the terms are interchangeable) is a point worthy of further discussion. Modern data is increasingly multivariate (Heer, et al., 2010), meaning each data point encodes more than three variables. Plotting all of the dimensions on a single graph isn’t feasible due to readability issues, plus humans find it difficult to mentally picture data in more than three dimensions (Heer, et al., 2010). Proposed solutions include:

4.5.1 Matrices
This technique involves breaking down the multiple dimensions into pairwise relations and plotting them in a matrix to cover all possible combinations. This enables visual inspection of correlations between all pairs of dimensions (Heer, et al., 2010). The most common implementation is a scatter plot matrix, but this technique can also be used in a different guise for network data, where each row and column represents a node link.

Heatmaps are another variation whereas instead of showing individual plots in a matrix, numbers are encoded with colours (Yau, 2011). This results in a grid of the data where each cell represents a value and is coloured based on that value. This enables all of the data to be shown at once, which is one of the advantages of a matrix approach. This technique is dependent on choosing a good colour scheme, particularly for large datasets with a lot of variability.
4.5.2 Glyph based visualisations
There are only so many variables that can be encoded through position, colour and size. To be able to fully reflect the power of multivariate data requires a more abstract approach. Glyph based visualisations are a type of visualisation where variables are assigned to specific features of objects (Lee, et al., 2003). The overall appearance of the objects change as the values change, enabling comparisons.

4.5.3 Chernoff Faces
Chernoff faces are perhaps the most common glyph based visualisation. Data is displayed as cartoon faces by choosing different facial features to represent each variable (Lee, et al., 2003). The problem is that it is based on the rather large assumption that people have a heightened sensitivity to reading facial features and expressions. A lack of empirical knowledge on Chernoff faces means this isn’t an exact science. The change in facial feature can be very subtle and hard to decode, which often leads to the assumption that this is a gimmicky approach to visualisation. The few studies that have been carried out reported that responses were slower for Chernoff faces than other visualisation methods, and people reported a lack of confidence in using the technique (Lee, et al., 2003). However, it is an excellent method for spotting outliers, as the crime rate data in Figure 4 shows (see District of Columbia).

![Chernoff Faces](image.png)

**The Face of Crime in the United States**

4.5.4 Star charts
Star charts (also known as radar or spider charts) are based on the same principle as Chernoff faces. The difference is that they use a different object to visualise the data. Multiple axes (one for each variable) are drawn, emanating from a centre point and equally spaced in a circle arrangement. The ends of the axes represent the maximum value and the axes are joined with connecting lines to highlight differences between the variables. This creates a star shape which can be plotted on two axes like a scatterplot to encode two additional variables.
This method can be adapted in numerous ways. A beautiful example of a star chart was created by the Organisation for Economic Co-operation and Development (OECD). To visualise countries well-being data across eleven variables, they used a flower metaphor (Cukier, 2011). Each country was represented by a flower, and a petal on the flower represented one of the eleven variables. The petal length encoded the value, and the width was user controlled depending on their interest in that particular variable.

![Flower metaphor star chart](image)

*Figure 5: Flower metaphor star chart. Source: (OECD, 2013).*

### 4.5.5 Parallel Coordinates

Star charts and their equivalents can encode a fairly large number of variables (thirteen for the flower design when plotted in two-dimensional space), but can quickly become useless after this if more variables exist in the data (Yau, 2011). Chernoff faces and star charts also make it difficult to identify how variables are related across groups (Yau, 2011).

Parallel coordinates can aid here. Parallel axes are used for each dimension, and for each unit of data, a line is drawn from left to right, moving up or down depending on the values at each axis (Heer, et al., 2010). This technique can visualise a large dataset in a compact way, and can help indicate correlation, as shown in Figure 6.

![Parallel coordinates correlation types](image)

*Figure 6: Parallel coordinates correlation types. Source: (Yau, 2013).*

However it is reliant on interaction to be fully effective as the large amount of lines can quickly affect readability. Filtering and highlighting data points becomes a necessity (Yau, 2011), and reordering of the axes can aid in data exploration (Heer, et al., 2010).

### 4.5.6 Interaction is the key

The constant theme with these new methods is that they require interaction to be fully effective. The datasets being visualised are so large that interactive tooltips, filtering techniques and
reordering of axes are vital techniques in helping the user make sense of it all. This makes interaction a vital tool in data visualisation and will be discussed in depth later.

At this juncture it is worth clarifying that this work is focused on modern, larger datasets. It has been noted that there is a demand for such visualisation needs, where traditional methods reach their limitations. However, there is a practical limit to the amount of data that can be displayed. A pressing limitation being screen space (Yau, 2013). Visual clutter is often a consequence of a careless use of screen space, especially when visualising larger datasets (Choo & Park, 2013), (Fekete & Plaisant, 2002).

4.6 Novel visualisations
Most current visualisation research has focused on novel ways of representing data (Soo Yi, et al., 2007). Figure 7 highlights some examples, including flocking boids (bird objects in a flocking simulation) to visualise stock exchange data (top left) (Moere, 2004), ThemeRiver to visualise news articles (top right) (Havre, et al., 2002), MyLifeBits to visualise photograph collections (bottom left) (Gemmell, et al., 2006), and streamgraphs to visualise music and film data (bottom right) (Byron & Wattenberg, 2008).

![Figure 7: Novel visualisation techniques.](image)

The problem with these techniques is that they are bespoke, and quite often, specific to visualising data from a certain domain. Novel visualisations require creative design and a steep learning curve to understand (Dix & Ellis, 1998). This appears to be an issue in current visualisation research, there just aren’t enough generic tools available that utilise the many years of accumulated experience that is inherent in traditional visualisation techniques.
4.7 The visualisation pipeline

Data visualisation requires a diversity of skill sets (Tuftes, 2001). Visualisation taxonomy is an area of research that can help understand the overall process and the disciplines that are involved. Most taxonomies are data-centric (Shneiderman, 1996), which classify visualisation types based on the data that they consume (temporal, network, hierarchical, cartographic etc.). Other research has attempted a more fine-grained approach by not only applying a data-centric slant, but also applying the operations that take place at each stage of the pipeline (called the data state model) (Chi, 2000). This results in a matrix of data types and data operations for a more detailed classification.

Either way, the pipelines described in these taxonomies are basic. They start with a data acquisition stage (computer science), a filtering stage to get to the appropriate data (statistics), and finally, a representation stage, where visual encodings are chosen for viewing (graphic design):

![Data Acquisition → Data Filtering → Data Representation](image)

Figure 8: The basic data visualisation pipeline.

Ware discusses a four stage pipeline (Ware, 2004), but this just separates the representation stage into a computer phase (display on screen), and a human phase (what the eye sees), to further his discussion on cognition. In their introduction to data visualisation paper (Grinstein & Ward, 2002), a more comprehensive pipeline is devised. Two additional stages are added after the data representation stage: - image orientation/viewing operations and visualisation interactions. These stages form part of an interactive layer in the pipeline to pan, zoom or rotate the display and to directly manipulate the elements that are displayed.

The problem is that these pipelines are not comprehensive enough to fully take into account the diversity of skill sets that go into the field of data visualisation. More importantly, they fail to address any sort of aesthetic consideration, encodings are arbitrarily chosen and that is it.

In his computational information design thesis, (Fry, 2004) attempted to bring all of the disparate disciplines together. To do this he devised a more comprehensive pipeline, see Figure 9.

![Computer Science, Mathematics, Statistics, and Data Mining, Graphic Design, Infovis and HCI](image)

Figure 9: Comprehensive data visualisation pipeline. Source: (Fry, 2004).

Note the refine stage, this is the stage that research tends to overlook. Its importance will be discussed in the next section. Also worth noting is the inclusion of an interaction stage that Grinstein and Ward initially proposed, this will also be investigated.
This work is focused on the represent, refine and interact stages of the pipeline. As such it assumes that the data has been acquired, parsed and filtered in such a way that it is ready for meaningful visualisation.

4.7.1 The refine stage

As established in section 4.2, vision is the key to how data visualisation works. As such, the visual design of a graph is vital. This isn’t just for aesthetic reasons, a poorly designed graph can hinder the portrayal of a dataset. This can often occur due to a haphazard assignment of a pre-attentive feature to encode the data (Fry, 2004). The refinement stage is vital in preventing this.

Whilst discussing the advantages of data visualisation, Ware noted that one of the greatest benefits is the sheer quantity of information that can be rapidly interpreted if it is presented well (Ware, 2004). Consequently making the case for some sort of refinement consideration.

Edward Tufte is seen as the expert in this field. In his seminal text ‘The Visual Display of Quantitative Information’ he enforces this message by noting that:

“Graphical elegance is often found in simplicity of design and complexity of data.” (Tufte, 2001, p. 177).

To implement this axiom, many of the graphs in the text appear quite bare, but they effectively depict the data. This is a surprising contrast to the expectation that refinement should include extra features to make a graph aesthetically pleasing.

Perhaps the best example of this is the stem-and-leaf plot. Although invented by Tukey (Tukey, 1977), Tufte advocates its use in his work as it neatly encompasses the principles he tries to espouse. A full explanation of how to construct such a plot is out of the scope of this work, but by simply focusing in on the most relevant digits of a set of numbers, and ordering them correctly, the overall distribution can be visualised.

![Figure 10: Stem and leaf plot. Source: (Tufte, 2001).](image)

Figure 10 shows a stem and leaf plot that visualises the heights of a set of volcanoes. The dataset distribution is easily realised. What is interesting is how basic the design is, proving that the refinement stage isn’t linked to visual artistry, but making a graphic simple and as easy to understand as possible.
To achieve this graphical simplicity, Tufte advocates the use of a data-ink-ratio rule. This is the removal of non-data-ink, which is ink on a graphic that fails to depict statistical information (e.g. grid lines, shading etc.). He even takes this to the extreme by removing the horizontal line from a bar in a bar chart. Tufte could perhaps take a leaf out of the usability book by recognising the familiarity principle. The bar chart minus the horizontal line looks odd, as if it is trying to depict something else, whereas a bar chart is immediately recognisable.

Why stop at removing the horizontal line? As it is height that is the encoding feature, why not just have a vertical line? A study by (Levy, et al., 1996) put this ‘line chart’ as an option to a group of students who had to pick what they considered was the most suitable visualisation based on certain scenarios. They chose the line chart only 13 times out of 654, a success rate of just under 2%. In comparison, a traditional bar chart was the most popular choice at 20%.

There is often a need for non-data-ink. For example, Yau stresses the need for annotating visualisations in many of his examples to highlight important areas (Yau, 2011). Would this be axed according to Tufte’s principles? Tufte does retreat slightly by adding the caveat ‘within reason’, but such rules can prove dangerous when applied strictly.

4.7.2 Multi-functioning graphical elements
A good method of ensuring that the data-ink-ratio is being maximised is to use multi-functioning graphical elements. The general principle being that data-ink can perform a dual role by not only depicting the underlying data, but also acting as a design function that is usually performed by non-data-ink. The prime example being the range frame as shown in Figure 11. This is a frame that is cropped to the upper and lower bounds of the data so as to depict the actual minimum and maximum values. A non-data-ink element is therefore converted into data-ink.

![Figure 11: The range frame. Source: (Tufte, 2001).](image)

Again, this can be a dangerous principle when applied strictly. It can lead to a graphical puzzle where the encodings only make sense to the creator (Tufte, 2001). However, a sensible approach, using encodings that are familiar, can result in a reduction of non-data-ink.

4.7.3 Embellishment or minimalism
The counter argument against this minimalist approach is that a visualisation must be embellished to engage the reader’s interest. One of the leading exponents of this approach is graphic artist Nigel Holmes.

An experiment conducted by (Bateman, et al., 2010) put Holmes’ embellished charts and a minimalist equivalent to a set of users, and measured how well they were understood by asking them questions about the data. They found that people did prefer the embellished versions as they improved memorability. Memorability is a part of the cognitive skill set, but there are more important skills that visualisation should be focusing on before memorability, such as comprehension, problem solving and decision making. Memorability will only benefit the public domain who may be reading a graphic in the context of a larger news article or story.

Unsurprisingly the debate continues to rage on such a contentious issue. One of the most recent studies attempted to measure what makes a visualisation memorable by displaying a sequence of
visualisations to users for one second, and asking them to click a button if they had seen that visualisation before (Borkin, et al., 2013). Their argument was that memorability was linked to understanding, so if memorability could be measured, it would lead a step closer to understanding what makes a visualisation effective. Their approach however is flawed, as the data is completely ignored. Displaying a visualisation for only one second doesn’t enable the memorability to be measured, only what eye-catching features in a visualisation are easily recalled. Users have nowhere near enough time to even begin comprehending the data behind it, which is the whole reason for data visualisation.

Visual embellishment is a dangerous approach in the wrong hands. Tufte highlighted this by showing that embellishments often hide or obscure the data, which is in direct contradiction to the graphical integrity principle:

“Show data variation, not design variation.” (Tufte, 2001, p. 61).

Figure 12 shows a graphic from the New York Times which highlights this issue.

![Figure 12: Example of an embellished visualisation. Source: (Tufte, 2001).](image)

Here, an increase in design (width of the road) of 783% is used to represent an increase in the data of only 53%. One can only assume that this is to magnify the issue by embellishing the chart and drawing attention to it. Tufte devises a lie factor formula to prevent such violations:

\[
\text{Lie Factor} = \frac{\text{size of effect shown in graphic}}{\text{size of effect in data}}
\]


A lie factor between 0.95 and 1.05 is considered acceptable. This graphic has a lie factor of 14.8 (783 / 53). The use of embellishment is therefore not warranted. This outlook is consolidated by the fact that (Bateman, et al., 2010) concluded that visual embellishment cannot be recommended fully as a valid approach, partly blaming its contentious nature.

This highlights an interesting dichotomy between a more traditional approach of producing graphics, and a more entertaining, emotional approach, designed to grab people’s attention. This depends upon the target audience. A scientific audience isn’t concerned with attention grabbing.
embellishments because they will have a vested interest in that particular field already. Whereas for the general public, the story will likely be struggling for attention in a mass media publication.

This work is focused on the traditional approach of visualising quantitative data. This is opposed to the artier, embellished end of the visualisation spectrum, which conveys more emotional subjects that do not involve quantitative data (e.g. infographics).

![Figure 13: Example of an infographic. Source: (McCandless, 2009).](image)

4.7.4 Examples of the missing refinement stage

To highlight the importance of the refinement stage, this section will demonstrate examples of a lack of graphical refinement, and how that can damage the effective portrayal of the data. Tufte filled an entire book on this subject, but this section will focus on the particular effect that software has made.

Software has the massive benefit of providing numerous options to create data visualisations. Software packages such as Excel put this power in the hands of millions of users. This can be both a blessing and a curse. These options enable the user to break many of the refinements that Tufte recommends.

Colour is perhaps the apotheosis. In data visualisation, colour is not an aesthetic consideration, it is a visual encoding just like position, size and orientation. (Yau, 2011). Software enables a colour to be chosen from millions of possible options. The problem is that humans can only distinguish between and recall approximately five colours (Fry, 2004). Humans also have a built-in preference for naturally occurring colours (Fry, 2004). Software enables artificial colours to be created by setting red, green and blue (RGB) values, this produces a jarring effect that distracts the viewer (Fry, 2004). Figure 14 highlights the problem, it shows a heatmap that visualises a set of biological data from the Visual Genotype project. The use of primary colours doesn't immediately convey a message as to what is most important about this dataset. The red and yellow cells are the interesting part of this...
dataset (rare genotypes), whereas the blue cells are the common genotypes. Therefore they could do with being a lighter colour that forces them into the background, magnifying the interesting rare genotypes.

Figure 14: Heatmap with a poor use of colour. Source: (Fry, 2004).

Figure 15 shows treemap software that suffers from a lack of refinement, particularly in respect to its layout (Fry, 2004). Too much space is taken up with interactive sliders, causing the rectangles (the actual data) to squash together. Sliders are perhaps best positioned horizontally across the screen to minimise wasted space. The hierarchy labels also use up vital screen space. This distorts the size of the hierarchy as a whole, making comparisons more difficult (Gemignani, 2009).

Figure 15: An unrefined Treemap. Source: (University of Maryland, 2003).
This is something that (Fekete & Plaisant, 2002) also noted during research into the scalability of treemap software. They found current systems were limited in displaying a lot of items due to control panels, labels and margins wasting too many pixels.

Software doesn’t tend to enforce enough rules in terms of using the correct visualisation technique for the data at hand, or providing simple checks on a dataset. This is a complex area and a lot of research has gone into attempts at automation to make the optimum choice (Viégas, et al., 2007). However, simple checks can go a long way. Figure 16 shows a classic example from the Fox news network, who visualised results from an opinion poll (ostensibly produced using computer software).

![Figure 16: A meaningless pie chart. Source: (Friendly, 2012).](image)

People were allowed to choose more than one candidate which temporarily saves Fox’s embarrassment of the totals not adding to 100%, but this means there is overlap in the data. Pie charts by definition cannot visualise overlapping data as each slice must be mutually exclusive. This chart is therefore meaningless. A simple check in the software to make sure the totals add up to 100% could easily have prevented any confusion.

Another check software can make is to ensure values are normalised prior to being visualised. For example, stacked area charts that visualise raw values and not percentages need normalising to correctly visualise parts of the ‘whole’ (Yau, 2011). This principle also applies to the size of rectangles in treemaps, they need to be normalised according to the available screen space. Normalising values to colour schemes is also required, as in heatmaps or choropleth maps, to ensure a consistent colour shading is used across a data range (Heer, et al., 2010).

When encoding a variable using area, it is important to calculate the size correctly. A common software mistake is to link an object’s dimensions to the variable, instead of the object’s area (Yau, 2013). For example, using a rectangle to encode file size in a treemap. The greater the file size, the larger the area of the rectangle. If one file size is 50% greater than another, and the sides of the rectangle are increased by 50% and not the area, the rectangle is too large which makes it impossible to fairly compare values. This also applies to bubble charts where the data variable is incorrectly linked to the radius or diameter (Yau, 2011).

Graphics software has exacerbated this situation. Designers are now attempting to encode data by ever more complex shapes. Figure 17 shows a visualisation of the trustworthiness of accents in call centres. They use triangles to encode the data, but do not scale them correctly.
The non-linear scaling makes it appear that a native accent is roughly 50% more trustworthy than a mild accent (the slice is approximately twice as large). Whereas according to the data, it is only 10%. A lie factor of 5.

This example also has a baseline that doesn’t start at zero, which distorts the data even more, something which Tufte warned heavily against. This is fine when using a scatterplot or line graph, but when the encoding feature is length or height, the baseline must start at zero to prevent skewing of the relative differences. Data visualisation expert Stephen Few notes that in most cases this isn’t a deliberate ploy to lie about the data, it is just that software such as Excel has made it too easy to set the baseline to a value other than zero (Few, 2013).

4.7.5 Why a lack of refinement in software?

These examples highlight a surprising lack of graphical refinement in software. It is surprising in so much that there has been a lot of work in this field for static visualisations in print, but these principles appear to have been ignored in software.

This is partly down to data visualisation historically lying in the scientific domain (Vande Moere, et al., 2012). Therefore research has mainly focused on optimising data exploration and analysis tasks, to the neglect of graphical refinement.

The need for interactivity has meant that controls for filtering, aggregating and reformatting have clouded the user interface, to the detriment of the graphic, something which Tufte didn’t discuss. These ‘dynamic’ displays are not rooted in current wisdom (Shneiderman, 1996), therefore resulting in a situation where user interface designers are now trying to work well beyond the current state of the art knowledge.

Another reason could be that refinement is such a contentious area. As discussed in section 4.7.3, refinement is often linked to graphical artistry and embellishment, which can obscure the data. This has resulted in a situation where refinement has been wrongly linked to displaying data in an insincere manner. Research has therefore tended to be wary about the subject, often ignoring it in the hope that a lack of refinement somehow proves the data is being visualised correctly.
It could also be just that it is considered to be of lower importance. The visualisation pipeline discussion in section 4.7 highlighted the fact that most research simplifies the steps needed for data visualisation, and as such, results in a situation where encodings are arbitrarily chosen without acknowledging how the graphic will be perceived.

4.8 New developments

4.8.1 3D

The increase in multivariate data, coupled with an improvement in 3D technology, has resulted in increased interest in 3D data visualisations (Levy, et al., 1996). Theoretically this makes sense, more visual dimensions means more possible data dimensions can be encoded. However, research has found that 3D visualisations do not necessarily improve the situation (Tversky, et al., 2002).

It is often concluded that users need more time to evaluate charts with 3D depth (Dix & Ellis, 1998). Humans perceive the area of 2D regions more accurately than 3D volume (Cleveland & McGill, 1985). This is because humans are unsure what to interpret as the value, should it be the area that is visible? The entire volume? Or something in-between? This is perhaps best highlighted using Tufte’s giant chicken example, as shown in Figure 18.

It is very difficult to tell how much bigger the 33 tonne of ice is compared to the 4,000 lb. chicken for example.

A study conducted by (Levy, et al., 1996) attempted to measure if the added dimensionality was valid or whether it was purely gratuitous. They concluded that 3D visualisations were preferred in certain situations, such as when memorability was an issue, and when the visualisation needed to stand out to make an impression to an audience. These situations are cosmetic, the point of data visualisation is to accurately portray data in graphical form, there is no evidence in this experiment that 3D visualisations aid in this regard.

Other problems with 3D include disorientation, occlusion (Shneiderman, 1996), and accuracy issues. 3D objects require a projection transformation to convert the coordinates into 2D space for display on screen. This transformation can complicate the visualisation and obscure the accuracy of results (Lie, et al., 2009).
Perspective can also be misused to create a false impression. Figure 19 shows a Steve Jobs keynote speech at Macworld 2008, in which a 3D pie chart is used to visualise the US smartphone market share by brand.

![Figure 19: Misuse of perspective in 3D data visualisation. Source: (Jobs, 2008).]

The ‘Apple’ green segment of the pie is positioned at the bottom of the chart and is tilted towards the viewer, making it appear much larger than the ‘Other’ purple segment at the top (Andrews, 2013). This design also goes against empirical knowledge which states that segments in a pie chart should be arranged in descending order to keep the data organised (Yau, 2011). An ordered 2D equivalent would prevent such ‘skewing’ of the data.

Familiarity issues have also been raised. The Financial Viewpoints project by Lisa Strausfeld (Strausfeld, 1995) joined spreadsheets together in three-dimensional space to show where they were linked. Although this was a novel idea, the target audience found it diverged too far from traditional visualisation methods, which made it difficult to understand (Fry, 2004).

Tufte didn’t discuss 3D in detail, his general rule was that dimensions on a graph should equal dimensions in the data to prevent misperception. However, he did provide examples of 2D charts displaying multivariate data. The prime example being Minard’s graphic depicting Napoleon’s Russian campaign of 1812:
The width of the flow-line depicts the army’s size during the march into Moscow (tan coloured band), and during the retreat (black band). In all, six variables are encoded: - the size of the army, its location on a two-dimensional surface, direction of the army’s movement and temperature on various dates during the retreat from Moscow. Tufte singled out this graphic for specific praise:

“It may well be the best statistical graphic ever drawn.” (Tufte, 2001, p. 40).

Proof that 2D visualisations can depict multivariate data just as effectively, if not more effectively than 3D visualisations.

4.8.2 Interactivity

Tufte’s work on visualisation failed to cover interaction. Static visualisations clearly have analytical and expressive value, but their effectiveness becomes limited as the dataset grows and the number of data variables increases (Soo Yi, et al., 2007). Modern visualisation techniques rely on interaction to be effective, as established in section 4.5.6. As such, many experts in the field agree that interaction is the future of data visualisation (Grant, 2013).

One of the goals of data visualisation is to solve problems. The psychology of problem solving is linked to the ability to reformat the problem domain (Eysenck & Keane, 1992). Data visualisation can aid here as it is seen as a hypothesis generation cycle (Keim, 2001), (Ware, 2004), where data (the problem domain) is displayed in various different fashions (reformatted), to help provide a new view of the data in an attempt to find different relationships, which help test new hypotheses.

This cannot be achieved through static data visualisations. The key is ‘engagement’, interactive visualisations engage users in exploratory behaviour (Pohl, et al., 2010). This exploratory behaviour is described in Shneiderman’s ‘visual information seeking mantra’ (Shneiderman, 1996). In it he describes a process in which visualisations are interactively explored to seek out information:

1. Overview first – the user sees the initial view of the entire dataset.
2. Zoom and filter – users typically have an interest in a certain portion of the dataset which requires tools to enable them to zoom in on areas of interest. To do this users are given control of the display to eliminate unwanted items.
3. Details on demand – the filtered dataset is now easy to browse and the user can click on individual items to bring up further details.

This process neatly encompasses what an interactive data visualisation should cover. The ability to search, filter and zoom in on details is critical to the cognitive process (Tidwell, 2011). The user then becomes a participant, rather than a passive observer. It has therefore become the contemporary process in interactive data visualisation. However, it is quite open ended. Although Shneiderman provides examples of how each step may be used in certain visualisations, he doesn’t provide any specific interactive techniques in terms of which tools to use. This has perhaps resulted in a situation where the user is overloaded with sliders, buttons and text boxes, without really thinking about the negative impact this can have on graphical refinement (see Figure 15).

This is exacerbated by the lack of empirical research and user testing of interactive techniques, partly due to the fact that research has mainly focused on data representation rather than interaction (Soo Yi, et al., 2007). A surprising situation considering the symbiotic relationship between the two.

In one of the few studies into how users interact with data visualisations (Pohl, et al., 2010), it was discovered that users did not in fact use many of the wide array of interactive tools that were available. They tended to interact more with the data itself (hovering and highlighting), rather than using actual interactive elements. This makes the case for a judicious use of interactive tool that are specific to certain visualisation types.

4.8.3 Animation

Due to technological improvements in data acquisition, data is now a constantly moving target, changing all the time (i.e. live data). Tufte’s work fell short of a discussion on this type of data. Animation is often considered a part of interaction, but it is important to discuss them both separately as each bring different capabilities to the field (Aigner, et al., 2007).

Animation is a sequence of images over time, used to convey the illusion of movement (Robertson, et al., 2008). This inherent change over time property makes it a natural fit to visualising time-series data, particularly live data (Fry, 2008).

An excellent example of this is the gapminder project by Hans Rosling (gapminder.org, 2008). Here, a bubble chart is used to plot life expectancy against gross domestic product since 1800. The bubble sizes represent a country’s population. The bubbles are animated to show how countries have developed over the last 200 years. The animation is good for spotting outliers, e.g. Rwanda’s life expectancy in the 1990’s.

Animations are ephemeral and can be too fast for people to follow. Due to this it is often questioned whether animations aid or inhibit understanding. A study by (Robertson, et al., 2008) aimed to answer this by putting the gapminder visualisation up against two non-animated equivalents - small multiples and traces (bubbles which are shown at every one of their x and y coordinates over time, and connected with lines to clarify their sequence) to a group of subjects to test their suitability in evaluating trends.

They found that the non-animated visualisations were preferred for analysis tasks, but animations were preferred if there was a presenter describing the data as the animation progressed. Part of the reason was that subjects found it hard to track the animated bubbles in gapminder, something which is magnified for a larger dataset. Subjects also reported difficulty in following ‘noisy’ datasets, i.e. data that has no general trend, which when animated, just swarms around the screen in an
apparent random fashion. The traces and small multiples options in this study were not quite non-animated equivalents of gapminder which may have skewed the test unfairly. For example, the bubble size in the traces visualisation didn’t encode population, but direction of trace flow.

Deducing the findings of this study shows that animation is very data-dependent in terms of having a clean dataset that has a general trend. This prevents the animation from being too cluttered and makes it easier to follow. The dataset used in gapminder suits this because the trend of life expectancy and GDP has generally increased over the last 200 years.

The problems with animation revealed in this study can be alleviated with interactive features, which is why it is important to separate the two. In gapminder features such as playback control, tooltips, and filtering techniques help overall understanding of the animation. However, it is still important to have a dataset which suits animation, therefore making animation a specific tool, rather than a generic one.

4.9 Data
A discussion about data visualisation wouldn’t be complete without considering the actual data that is to be visualised. Data has obviously been a constant theme throughout this discussion, as a result most aspects have already been covered. What hasn’t been covered though are the formats that modern datasets take.

Although modern datasets tend to be unstructured, as opposed to traditional structured data, which is backed by a strict data schema that is unlikely to change (Dijcks, 2013), they still need to be exported in a format that can be read by data visualisation software.

Many tools that target particular environments expect data to be provided in a specific way. For example, commercial data visualisation controls expect classes to be created that contain collections of the data (Syncfusion Inc., 2013). Data binding is then used to link the data to the control. Although this is an effective method for that particular environment, it isn’t generic enough. It just isn’t feasible to expect a class to be created for a large dataset.

The web based visualisation systems are perhaps the beacon. Being a web based system, they inherently want to appeal to a large an audience as possible. One such system, Many Eyes (IBM, 2013), use a table as the core data model. This appeals to a wide audience as tables are simple and well understood by end users, possibly due to the pervasiveness of Microsoft Excel, and the fact that users want to exploit the vast amount of data stored in Excel spreadsheets (Viégas, et al., 2007). A factor in the recent explosion of interest in data visualisation is the opening up of government data, most of which is stored in tabular form (data.gov.uk, 2014).

Consequently, many similar services have converged on the table model (Viégas, et al., 2007). Most of the examples in Yau’s work (Yau, 2011) using the R programming language (The R Project, 2013) consume comma separated value (.csv) files exported from Excel. This is so common that R contains a function specifically for this (Yau, 2011). All of the visualisations in McCandless’ work (McCandless, 2009) also consume data in the form of tables (McCandless, 2013).

4.10 Summary
This discussion has uncovered a situation where disparate disciplines have been operating in apparent isolation, to the detriment of data visualisation as a field. This has resulted in a ‘gap’ in current work where the wisdom of each discipline hasn’t been fully utilised as a whole.
There is the graphical refinement work by Tufte that focuses on traditional static visualisation techniques. These techniques do not scale well to the demands of the more modern/multivariate dataset. Different visualisation techniques are available that can address this problem, but the refinement has been lost. These techniques have also tended to be specific to certain domains that require a steep learning curve to decode and a considerable amount of artistic skill to create.

Technological developments haven’t necessarily improved upon the current situation. Animation and 3D are more specific tools that should be utilised in a prudent manner depending upon the dataset. Interactivity is an area that many people believe is the future of data visualisation. However, as it has been introduced, user interface designers have neglected graphical refinement by overloading the users with interactive tools.

The danger of a lack of refinement has been demonstrated. It isn’t just an aesthetic consideration, a lack of refinement is strongly linked (inadvertently or otherwise) to a lack of graphical integrity that doesn’t portray the underlying dataset in a genuine manner.

There is therefore a requirement to bring all of these findings together into a modern, generic, data visualisation tool that supports interaction, and upholds the graphical refinement principles to create an accurate representation of a dataset.
5 Design and development

The literature review has highlighted a potential area of improvement in contemporary work. Therefore software will be created to demonstrate how the situation can be improved. This section details the design and development of this software. The literature review is referred to throughout to enforce the decisions that are made. References to code are in italic.

The software will implement a sample set of data visualisation techniques that were identified in the literature review as being suited to larger datasets, as these are techniques that were identified as requiring the most improvement. These are:

- Parallel coordinates
- Treemaps
- Heatmaps
- Graduated symbol maps

The software will take the form of user controls. User controls lend themselves nicely to data visualisation as they are self-contained modules of code that can implement separate visualisation techniques. The ‘user’ in user controls refers to the programmer, rather than the end user (Petzold, 2013). This enables the controls to be used as building blocks in more complex data visualisation software, which makes this method a more scalable and versatile approach.

5.1 The target environment

The controls will target the Windows runtime (WinRT) application programming interface (API) for Windows 8.1. This new version of Windows is suited to data visualisation as it incorporates a new design paradigm that focuses on full-screen programs, with an emphasis on content over ‘chrome’ (a program’s user interface elements such as menus, status bars, tool bars etc.) (Harris, 2011). As discussed in sections 4.5.6 and 4.7.1, the importance of high data-ink-ratios, and maximising available screen space are vital factors in data visualisation, which makes this environment a suitable fit. WinRT development commonly involves a combination of programming language and markup language to separate the workflow between programming and design (Petzold, 2013). The controls will be developed using C# as the programming language and XAML as the markup language.

5.2 The development environment

The development environment that will be used is Microsoft Visual Studio 2013. The controls will be created as a class library. This enables them to be easily reused by other programmers for different projects, and also prevents them from becoming entwined within the code of an actual application. To test the controls during development, two projects will be added to a single Visual Studio solution: - the class library that implements the controls, and an application that will consume them (see Figure 21). This setup has the additional benefit of Visual Studio building both of the projects if they are out of date, which enables a quicker write-test-debug cycle.
5.3 Data model

As discussed in section 4.9, many existing tools expect the data to be provided in specific ways. This limits the use of these tools to domain experts. Web based systems provide a much better abstraction to try and appeal to as wide an audience as possible. With this in mind, it is decided to put as little overhead on the user as possible. This leads to the ideal situation where a user can just select a visualisation type by choosing the appropriate control, provide a dataset, and specify the salient details to create the visualisation.

The first task is to investigate the data model that the controls will exhibit. Section 4.9 established the pervasiveness of .csv files as a supply of data, so a data model that closely resembles this structure should be chosen. A .csv file displays data in rows and columns, the closest data structure to this being a two-dimensional array.

However, arrays are not dynamic. It is highly likely that the datasets will need to be rearranged or modified in some way during visualisation, particularly during interaction, which doesn’t make this an ideal approach. A dynamic List<T> collection is an ideal data structure for this. The file input/output (I/O) methods read the data in as a string, line by line. This can be split using a comma to create a string array for each data entry, which can then be added to a List<string[]>.

5.3.1 Reading external files

As user controls are intended for developers, the datasets should be accessed through Visual Studio, as opposed to providing other file access mechanisms such as file pickers. This is a setup that will be familiar to developers, and is a slicker approach to development as no dialog boxes are required.
Due to the controls being created in a separate library, they will require the directory that contains the data source. The user can specify this with the following call:

```javascript
```

*Figure 22: Providing the dataset folder location to the controls.*

The class library assumes a folder named ‘Data’ exists at this location. All controls will have a mechanism to store a filename which will be used to access the actual dataset. The user then only needs to ensure that the data is in correct .csv format, and for Visual Studio development purposes, the ‘Build Action’ option is set to ‘Content’ and the ‘Copy to Output Directory’ option is set to ‘Copy always’ (see Figure 23). This ensures that the file will exist at the specified location.

*Figure 23: Visual Studio dataset file properties.*

The next requirement is to provide a mechanism for the user to work with this data structure. The user will need to specify the salient properties of the dataset to create the visualisation. For example, which data dimension should dictate the size of the rectangles in a treemap. This will of course differ for each visualisation type, but the method to supply these properties should remain constant throughout.

One approach would be to provide all relevant properties during the creation of the control, i.e. in the constructor. However, defining a parameterised constructor in a UserControl derivative is extremely unorthodox (Petzold, 2013). This is partly because a parameterless default constructor is required if the control is to be instantiated in XAML (see Figure 24), a feature that is desirable.
A much better way is to define properties (Petzold, 2013). Therefore, each control exposes properties that need to be set to create the visualisation. The only decision to be made here is what values the user specifies for the properties. As each data entry is an array, the initial idea was to let the user specify the data dimension by providing an index. However, this approach is fairly abstract, and it doesn’t help with code readability and maintenance. A more descriptive way is for the user to provide the dimension name. The user will be familiar with the dataset, so providing the names will be a more intuitive approach.

Consequently the controls expect the first line of the .csv file to contain the dataset headings. The program peeks ahead into the dataset and reads these into a separate array. This is then searched using the properties to ascertain the accompanying integer index.

5.3.2 Data types
As the file I/O methods read the dataset in as a string, numerical data will have to be parsed. The double type will be used to prevent any potential loss of precision.

5.4 Architecture
The controls will utilise the model-view-view model (MVVM) architectural pattern that is prevalent in WinRT development (Petzold, 2013). This approach will help separate code between presentation (view), logic (view model), and data handling (model). This will maximise code reusability and prevent any duplication.

A model class creates the data model from the external .csv file and exposes properties for the view model to access. As user controls are partial classes, the XAML page will be the view, and the code behind will be the view model. The view model interrogates the model for the data and combines this with the logic of the visualisation method it is implementing. The view then displays the visualisation and handles user input via the interactive controls. The view transfers user input to the view model, which (if necessary) updates the model, and consequently forces an update of the view. The architecture is shown in Figure 25.
5.5 User interface design

As discussed in section 4.5.6, screen space is a limiting factor in data visualisation and should be used judiciously. The user interface (UI) will require a drawing area to display the visualisation and an area to host the interactive controls. As section 4.7.4 established, a lot of screen space can be wasted through the use of interactive controls (see Figure 15). It is envisaged that sliders, combo boxes and buttons will be required. These are best placed horizontally across the screen so as to limit wasted space. To achieve this, the UI is divided into two rows, one for the drawing area, and one for the interactive controls. This layout is shown in Figure 26.

![Figure 25: Basic architecture design.](image)

![Figure 26: Basic UI design.](image)
The controls will assume that they are being used on a ‘blank page’, i.e. they work best when they are placed in an empty container. As such, the user doesn’t have to specify a width or height value when instantiating the controls. The controls are then free to use all of the available parent container space, which is ideally, the root element of the application XAML, which enables the full screen to be used for drawing.

5.5.1 The drawing area
Retrieving the drawing area dimensions is required to plot data values onto 2D space, as this requires normalisation of the data to the available drawing area.

The dimensions are retrieved in a *Loaded* event handler, as this is when all of the elements in the visual tree have been given actual dimensions (Petzold, 2013). As the user doesn’t specify a width or height, the actual size of the drawing area is retrieved from the *ActualWidth* and *ActualHeight* properties (Microsoft, 2014).

5.6 Colour
Figure 14 demonstrated that a careless use of colour can obscure a dataset and result in a visualisation that is hard to decode. For this reason it is decided to implement a colour scheme class that controls the possible colour schemes that can be applied. The class is based on the ColorBrewer project designed by cartographer Cynthia Brewer (Brewer, 2014). Although intended for cartographic purposes, the principles apply in general to any visualisation technique (Yau, 2011).

ColorBrewer uses colours that appear more natural, and encode data values through a good organisation of hue, lightness and saturation. This also limits the number of colours that can be used, which is an important factor in ensuring that the human eye can distinguish between them. The use of a well organised colour scheme means that the pre-attentive effectiveness of colour is ensured. Users who aren’t aware of these factors often produce a random assignment of colours, which results in a situation where colour degrades into a look-up table for the data (Brewer, 1994).

5.6.1 Colour scheme class
The colour scheme class is a singleton class as only one instance is required across the library. The ColorBrewer values are provided in JavaScript object notation form (JSON) which is parsed when the class is instantiated.

The choice of colour scheme depends upon two factors:-

- The number of data classes – i.e. how many colours should be used. Limited to twelve to ensure the human eye can distinguish between them.
- The colour scheme type, of which there are three options:-
  - Sequential – suited to values that range from low to high. Colours are incremented in lightness from light to dark to encode the values.
  - Diverging – suited to values that have a mid-point with good and bad extremes either side. The mid-point is given a light colour with the low and high values either side given gradual darker shades of contrasting hues to emphasise the difference.
  - Qualitative – suited to categorical data. A unique colour is given to each category.

These two options are then used to retrieve a list of possible colour schemes. Each scheme has a unique name that is used to obtain the actual colour values.
5.6.2 Binding to the user interface

Three user interface elements are therefore required, one for the number of data classes, one for
the colour scheme type, and one for the actual colour scheme. The ComboBox element is chosen for
this as it as a compact way of enabling selection via a drop-down list. Radio buttons consume too
much screen space, which as has been identified, is a precious commodity.

5.6.3 Qualitative or non-qualitative

The data values encoded by the colours are binary in nature, in terms of being either qualitative or
non-qualitative. If the values are textual, then applying a sequential or diverging colour scheme will
not make sense. Conversely, numerical data coloured as a qualitative colour scheme will not encode
the underlying values correctly.

Section 4.7.4 uncovered a surprising lack of software checks. It is therefore decided that the controls
should make an inference about this automatically without the need for user input. This removes
the overhead from the user and results in a visualisation that immediately makes sense, without the
user having to manually match a colour scheme to a dataset.

If the data can be parsed into numeric values, then the colour scheme can be either sequential or
diverging. If not, the only possible colour scheme is qualitative. It also makes sense to automatically
derive the number of categories from a qualitative dataset. This prevents the user having to
calculate this manually.

5.6.4 Applying the colours

The actual colour data is stored as a jagged array. The first array size is the number of data classes,
and the second is an array of three elements, which contains the RGB values ranging from 0-255.

5.6.4.1 Sequential and diverging colour schemes

To encode a numerical value into a colour, a linear transformation is applied. First the value is
normalised, this takes into account the maximum and minimum value of the data dimension, which
ensures the colours accurately portray the underlying range of data. This normalised value is then
converted into an index between the data classes.

\[
\frac{\text{value} - \text{min}}{\text{max} - \text{min}} \times (\text{colourMax} - \text{colourMin})
\]

*Equation 2: Linear transformation equation to apply the colour schemes.*

As an example, take three data values (82, 54, and 25) of a data dimension that has a maximum of
82 and a minimum of 25. These values are being applied to a colour scheme with nine data classes:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>82</td>
<td>81</td>
<td>80</td>
<td>79</td>
<td>78</td>
<td>77</td>
<td>76</td>
<td>75</td>
<td>74</td>
</tr>
</tbody>
</table>

\[
\frac{82 - 25}{82 - 25} \times (8 - 0) = 8 \quad \frac{25 - 25}{82 - 25} \times (8 - 0) = 0 \quad \frac{54 - 25}{82 - 25} \times (8 - 0) = 4.07
\]

*Figure 27: Example linear transformations.*

82 is the maximum, so is at the darkest end of the range, 25 is the minimum, so it is the lightest. 54
is halfway, which equals colour index 4 (the whole number being used as the resulting index).
In code this can be simplified because the data class indices will always begin at zero. All that is required is the jagged array length, subtracting one to account for the zero starting index.

```csharp
    double linearTransform = (cellValue - min) / (max - min) * (nDataClasses - 1);
    int colourIndex = (int)Math.Truncate(linearTransform);
    byte[] colourRGB = colourScheme[colourIndex];
```

**Figure 28: Linear transformation implemented in code.**

### 5.6.4.2 Qualitative colour schemes

If the data values are qualitative, a list of distinct categories is retrieved from the dataset using the `Distinct` method of the `List` class. The number of data classes is then inferred automatically using the size of this list. To assign the colours, the category string is used as a lookup into the distinct categories list, this returns an index which is used to retrieve the RGB array. This ensures that each category is assigned the same colour.

### 5.7 Elements for data representation

To represent the data visually, simple shapes are required such as a rectangle for a treemap or heatmap, a line for a parallel coordinates plot, or a circle for a graduated symbol map. Similar shapes already exist as built-in XAML elements. However, to provide full control over the visual appearance and behaviour, these visual elements are extended as user controls themselves.

This enables finer adjustment of the visual appearance, and through the use of event overriding, enables the behaviour of the elements to be specifically defined. The implementation details are discussed within each user control section.

### 5.8 Interaction

Section 4.8.2 established that the user can be bombarded with interactive features, of which, only a small number will be actually used. A judicious use of interactive technique is therefore warranted. The techniques will be specific to the visualisation method being employed, these will be discussed within their appropriate sections.

#### 5.8.1 Search

However, search is a technique that applies across all visualisation methods. Search satisfies the zoom stage of the interactive pipeline by enabling the user to focus in on entries of interest. Search functionality is provided by the `SearchBox` element. When a query is submitted, the control’s properties are searched to see if there is a match. If there is, the data entry is highlighted accordingly. The `IndexOf` string method is used to check the query, this way only a subset of the data entry information has to match. This provides a more useful search functionality than having to make an exact match.

### 5.9 Exception handling

Accessing the file system can result in a number of exceptions being thrown, particularly if the file doesn’t exist, or if the file is being modified. The file I/O methods are placed within a `try/catch` block to catch any exceptions that may occur.

Each control has a small number of mandatory properties that need to be set, otherwise the visualisation cannot be created. A separate method checks if these properties have been set before any further code is executed.
As the data is read in as strings, any numerical data will need to be parsed. TryParse will be used to initially check all numerical data dimensions. Any subsequent access to these fields can then just use Parse safe in the knowledge that the numbers have been checked, this simplifies the code.

There is a possibility of a dataset requiring more than twelve data classes for a qualitative colour scheme. ColorBrewer doesn’t support this. This will be checked when the colour ComboBox elements are initialised.

Whenever any of the checks above return an exception, an appropriate message is reported to the user using the MessageDialog class.

**5.10 Parallel coordinates**

**5.10.1 Properties**

The parallel coordinates control requires the axes that are to be plotted. This is exposed as a property that consumes a string array. The user can then specify the data dimensions in the order they wish them to be plotted.

The control then iterates through this array and generates an equivalent index array from the column headings. Figure 29 demonstrates this with the use of an education statistics dataset.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>reading</td>
<td>math</td>
<td>writing</td>
<td>percent_graduates_sat</td>
<td>pupil_staff_ratio</td>
<td>dropout_rate</td>
</tr>
</tbody>
</table>

*Figure 29: Example parallel coordinates column headings.*

The user then specifies which columns to plot using the Axes property.

```csharp
pc.Axes = new string[] { "writing", "reading", "math", "dropout_rate", "pupil_staff_ratio", "percent_graduates_sat" };
```

*Figure 30: Setting the parallel coordinates control axes property.*

Which results in an index array of:

<table>
<thead>
<tr>
<th>Array element -&gt;</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array value -&gt;</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

*Figure 31: Parallel coordinate’s axes index array.*

The maximum and minimum value of each axis are required to normalise the data values to the drawing area. A Dictionary is used to store these. The key is the axis name, and the value is an array of two elements, the maximum and minimum value. This Dictionary can then be queried using the axis title to return the equivalent maximum and minimum values.

**5.10.2 Plotting**

Graphs that tend toward the horizontal (greater in length than height) enable the user to spot deviations more easily, as the eye is naturally practiced in detecting deviations from the horizon (Tufte, 2001). Therefore, the parallel coordinates plot will be drawn from left to right with vertical axes.

The x coordinates will be incremented by the distance between each equally-spaced axis. The y coordinates are based on the data values themselves. The values are normalised between 0 and 1 to get a location along the axis, which is then multiplied by the drawing area height to get the actual
screen coordinate. The problem is that in the Windows coordinate system, x0y0 is at the top left corner of the window, with y increasing from top to bottom, as shown in Figure 32.

![Figure 32: Windows coordinate system. Source: (functionX, 2010).](image)

Using the standard normalisation formula shown in Equation 3 would result in the minimum values being at the top, and the maximum at the bottom.

\[
\frac{value - \min}{\max - \min}
\]

*Equation 3: Standard normalisation equation.*

As established in section 4.7.4, height as an encoding feature is irrevocably linked to increases in value as the height increases, meaning this approach isn’t acceptable. There are two options, either stick to the traditional normalisation formula and subtract the result from 1 to ‘invert’ the result, or modify the formula. Reversing the formula is a trivial task (see Equation 4).

\[
\frac{value - \max}{\min - \max}
\]

*Equation 4: Reversed normalisation formula to account for the Windows coordinate system.*

To draw the lines, the *Polyline* element is used. This ensures that each entry in the dataset can be represented by a single line, as opposed to a collection of basic *Line* types. A new user control called *ParallelCoordsPolyline* is created to provide this. The control exposes a property that takes a collection of points which are used to plot it. This makes it a simple case of passing the x, y coordinates as they are normalised.

### 5.10.3 Axis headings and reordering

One of the important interactive features of a parallel coordinates plot is the ability to reorder the axes. The crossing of lines between axes often indicates inverse correlation (Heer, et al., 2010), (Yau, 2013), as shown in Figure 6. The reordering of axes can help spot this correlation by enabling the user to look at line crossings between all possible combinations of axes.

Reordering the axis labels via drag and drop is an intuitive way of implementing this technique. This prevents the need for any additional controls cluttering up the screen. The *ListView* control being the ideal candidate.
The default stacking for a ListView is vertical, so this is modified to horizontal by changing the ItemsPanelTemplate to a horizontal StackPanel. The CanReorderItems and AllowDrop properties are set to true to enable drag and drop.

```xml
<ListView x:Name="axesRow" Grid.Row="1" CanReorderItems="True" AllowDrop="True">
    <ListView.ItemsPanel>
        <ItemsPanelTemplate>
            <StackPanel Orientation="Horizontal"></StackPanel>
        </ItemsPanelTemplate>
    </ListView.ItemsPanel>
</ListView>
```

Figure 33: Modified ListView template for horizontal display.

The children of the ListView are the axes headings. These are created using data binding, as opposed to creating them manually. To accomplish this, the ItemsSource property is set to a collection. Originally a collection of strings was considered, but this approach doesn't enable control over the positioning of the labels to ensure they are centred over the axes correctly.

The child element of a ListView is a ListViewItem, as this object inherits from FrameworkElement, control over positioning via Width, Margin and HorizontalAlignment properties is permitted. A collection of ListViewItems is created, and their Content property is set to a string to display the axis name. The first and last axes are set to half of the x distance because they are placed at the edges of the drawing area.

The collection is of type ObservableCollection. This is required because the reordering will modify the collection at runtime. An ObservableCollection is notified when a change occurs and a CollectionChanged event is raised.

This event is raised twice when a drag and drop operation occurs. The first occurrence captures the item that is being moved (classed as a remove action). The second occurrence captures the collection in the reordered state with a new item at the index at which it was ‘dropped’ (classed as an add action). A switch statement handles this. In the remove action, the axis name that is being ‘dragged’ is captured, then in the add action, this name is applied to the new item’s content. The axes indices are then updated accordingly and the graph is redrawn.

5.10.4 Tooltips and highlighting

The interactive pipeline established in section 4.8.2 identified a details-on-demand stage where a user can obtain further information about a data entry. A tooltip is an ideal way of providing this functionality. Displaying tooltips as the user hovers over the data makes the information immediately visible without the need for additional mouse clicks. The PointerEntered event is required to implement this. As the visual elements have been extended as new controls, it is a simple case of overriding the PointerEntered event inside ParallelCoordsPolyline to specify the required behaviour. The ToolTip class is used to display the information.

5.10.5 Line selection

As part of the zoom and filter stages of the interactive pipeline, lines should be able to be highlighted on selection. This provides a way for a user to zoom in on points of interest. Simple selection via mouse click is provided. The user can select or de-select multiple lines at a time. The lines are highlighted using a different colour to provide contrast which helps the user focus in on just
those data entries. The Tapped event of ParallelCoordsPolyline is overridden to implement this functionality.

5.10.6 Advanced filtering
A more advanced interactive technique is to filter the dataset based on actual data values. This enables the user to highlight all lines that lie within a certain range of interest at a particular axis, thus allowing them to ‘zoom in’ on the dataset. Instead of using additional controls, it is decided to implement this functionality directly onto the visualisation itself. The use of a slider at each axis enables this, as well as providing the additional benefit of applying a visual appearance for the axis.

The problem is that the default Slider control only enables a single value to be chosen, whereas two are required to select a range. A new control called RangeSlider that extends the default Slider functionality is created to solve this problem. Unfortunately, the code of the standard Slider cannot be examined to see how it is constructed. However, by looking at the styles in generic.xaml, the visual structure of the default controls can be investigated (Petzold, 2013).

At its core, a Slider is built up from Rectangle and Thumb elements. The Thumb element is what the user moves, and two Rectangle elements are drawn either side to give the appearance that the thumb is moving along a track.

![Figure 34: Anatomy of a standard Slider.]

The RangeSlider simply adds another Thumb element and an additional Rectangle for the centre.

![Figure 35: Anatomy of a RangeSlider.]

This structure is created using a Grid. The Grid rows are resized when the Thumb is moved to give the appearance of the thumb moving. This approach also prevents any overlap problems as the Grid elements will shrink until zero, but no further.

The Thumb DragDelta event is used to redraw the Grid whilst the Thumb is being moved. The event handler retrieves the upper and lower rectangle grid rows and updates their sizes based on how far the Thumb has been moved.

5.10.6.1 Value conversion
To represent the axis values, maximum and minimum properties are used to store the corresponding axis maximum and minimum values. The difference between these values is then divided by the height of the control to create a normalisation factor, which is used to ensure that the values are clamped correctly.

Properties are also required for the upper and lower Thumb values. These properties are created as dependency properties, as it would be useful if they could be targets for data binding (Petzold, 2013). The properties are updated when the Thumb has finished dragging, in a DragCompleted event, rather than during a DragDelta event, when the Thumb is still moving. This prevents too many updates being fired.
The \textit{DragCompleted} event handler gets the size of the upper or lower rectangle (depending on which \textit{Thumb} has been dragged), and converts this to a value using the normalisation factor calculated previously. The dependency property is then updated accordingly.

A notification mechanism is required when the values have changed. This enables a consumer of the control to do something useful when the slider has been updated. The standard \textit{Slider} object achieves this through a \textit{ValueChanged} event. Using this same principle, two events are created, one for the upper thumb and one for the lower. An event arguments class stores the new values to enable them to be retrieved in an event handler. The event is fired at the end of the \textit{DragCompleted} event, immediately after the dependency property is updated.

\subsection*{5.10.6.2 Incorporation into the parallel coordinates control}

The \textit{RangeSlider} objects are added to the parallel coordinates canvas at the axis locations. When a lower or upper value changed event occurs, the values of each data entry are checked to see if they are within range. If the values are outside the range, the opacity is modified to fade them into the background, which emphasises the data entries that are within range.

The event handlers check every \textit{RangeSlider}, so that only data entries that are within range at every axis are left highlighted. For this to work, a property is required in \textit{ParallelCoordsPolyline} which contains the raw data values for that particular data entry.

To provide visual feedback as to the actual values the user is filtering by, the \textit{Thumb} elements in \textit{RangeSlider} contain \textit{ToolTip}s that are bound to their equivalent dependency properties.

\begin{verbatim}
<Thumb x:Name="UpperThumb"
    Grid.Row="1"
    DataContext="{Binding UpperValue}"
    Height="24"
    DragDelta="UpperThumb_DragDelta"
    DragCompleted="UpperThumb_DragCompleted"
    ToolTipService.ToolTip="{Binding ElementName=rangeSliderRoot, Path=UpperValue}"
/>
\end{verbatim}

\textit{Figure 36: Data binding to the RangeSlider dependency properties.}

\section*{5.11 Graduated symbol map}

\subsection*{5.11.1 Properties}

The symbol used in this control is a bubble, this enables the encoding of two additional data dimensions through size and colour. Properties are therefore required for size and colour, as well as place name, latitude and longitude.

As the sizes of the bubbles are directly linked to the data values, the displayed size cannot be controlled. Bubble size legibility is therefore an issue. For this reason a property is created which enables the user to specify the largest possible bubble size. Setting this property ensures that the largest data value in the dataset will equate to this size.

\subsection*{5.11.2 Plotting}

The mapping functionality is provided by the ‘Bing Maps SDK for Windows 8.1 Store apps’ (Microsoft, 2014).

Modern datasets are likely to contain the latitude and longitude already, as discussed in section 4.4.3. However, in datasets where this information doesn’t exist, the place name is geocoded. This
extends the usefulness of the controls. The ‘Bing Maps Simple Object Access Protocol (SOAP) Services’ provides this geocoding service (Brundritt & McGovern, 2014).

The default element for plotting positions on a map is the Pushpin. However, this doesn’t provide functionality to control the size. Plotting shapes on the map is possible, but is limited to polygons rather than circles. There is no XAML circle shape, the closest element is an Ellipse. A separate user control called BubblePushpin is created to give full control over the appearance and size of the bubbles.

The size of the bubbles are an encoding feature, so it is important to correctly size them according to area, as opposed to radius or diameter (Yau, 2011), as identified in section 4.7.4.

\[ \text{area} = \pi r^2, \ r = \sqrt{\text{area}/\pi} \]

\[ \text{Equation 5: Graduated symbol map bubble area equation.} \]

Pi can be omitted however as it is a constant (Yau, 2011).

To ensure the largest bubble size equates correctly to the largest bubble size property, a scale factor is calculated by dividing the property value by the radius of the largest data value. This is applied to BubblePushpin as a static property. The resulting radius value from Equation 5 is then multiplied by this scale factor. The radius value is multiplied by two to get the diameter, which is set to both the Height and Width properties of the Ellipse to ensure a circle is drawn.

The BubblePushpins are added to a collection rather than directly onto the map. This enables any future reordering that may be required as a result of user interaction, and helps prevent occlusion via sorting of the collection in descending order. As BubblePushpin inherits from UserControl, which in turn inherits from DependencyObject, it is a valid element for the SetPosition method, enabling it to be plotted with a latitude and longitude.

5.11.3 Tooltips

To implement tooltips, a string property in BubblePushpin is created to store the relevant information to be displayed. The OnPointerEntered and OnPointerExited events are then overridden to display them.

5.11.4 Refinement

As the mapping functionality is provided by an external library, overall refinement control is lost. The Bing maps API does allow overriding of the default map tiles. MapCruncher (Elson, et al., 2007) enables new map tiles to be created by overlaying a new image onto the default Bing maps image. Corresponding landmark locations are ‘pinned’ to position this new image correctly. The software then creates new map tiles based on the quad key convention (Schwartz, 2014). The tiles can then be served by creating a new MapTileLayer and subscribing to the GetTileUri event. This event takes the tile coordinate and zoom level, and converts this into a quad key which corresponds to a tile image created by MapCruncher.
This approach was discarded due to inaccuracies in positioning. Also, the resulting tile images are of much lower quality compared to the default tiles, especially when zoomed, as Figure 37 demonstrates. Finally, the new tiles can only be served from a web server (MSDN, 2012), which complicates development and results in performance implications.

There are however certain map properties provided by the API that offer a certain degree of control over the appearance. The map logo and copyright are moved to the bottom of the screen to minimise their impact. Traffic flow and buildings are turned off for the same reason. The breadcrumb navigation bar is enabled to provide the user with a navigational cue. The view is also set to show a repeated globe, this results in a location agnostic visualisation.

5.11.5 Bubble resizing
A Slider control is provided to scale the bubbles interactively, whilst maintaining their proportions. This helps to create a clearer graph if bubbles are located in close proximity, and prevents the user having to empirically set the largest bubble size.

5.12 Treemap
5.12.1 Properties
A size property is required to define which data dimension controls the size of the rectangles. The size is linked to the area of the rectangle, as opposed to the length or width. Name and description properties are required to identify and provide a more detailed description of each data entry. A colour value property is required to store the value that will be used to colour the rectangles. A hierarchy property is also required to define which dimension controls the hierarchy level.

5.12.2 Layout algorithm
A treemap layout algorithm works by recursively subdividing an initial rectangle. The size of each sub-rectangle corresponds to the size of the data entry it is encoding. In the initial treemap paper (Johnson & Shneiderman, 1991), a slice and dice algorithm is proposed, in which the direction of subdivision alternates between horizontal and vertical per level. This layout can however lead to
long, thin rectangles that are hard to read (Bruls, et al., 2000). This is particularly evident in treemaps that visualise flat tree structures that only have one or two levels, Figure 15 being a prime example.

To overcome this refinement issue, the layout algorithm used for the treemap control is based on the ‘squarified’ algorithm created by (Bruls, et al., 2000). This layout takes into account the aspect ratios of the rectangles and attempts to keep it as close to one as possible (i.e. a square). The pseudo-code for this algorithm can be seen in Figure 38. This layout algorithm improves the overall look of the visualisation as squares are easier to detect and compare (Bruls, et al., 2000). The overall accuracy of the visualisation is therefore improved.

The squarified algorithm is a recursive function that takes a list of data values, and a list of the current row of rectangles as it builds them up. A decision is made whether to draw vertically or horizontally depending on the remaining width and height. The width and height of the current rectangle are then calculated from its area value. From this the aspect ratio can be calculated, if it is better than the previous aspect ratio (i.e. closer to one), then the rectangle is accepted into the current row and the algorithm continues using this current row. If the aspect ratio is not improved, the rectangle is rejected and the row is ‘locked’. The remaining area is then updated, the aspect ratio is reset, and the algorithm continues with a new empty row.

```c
function Squarify(list children, list row)
{
    row.Add(head(children));
    bool drawVert = DrawVertically(remainingWidth, remaningHeight);

    if (drawVert)
        real width = area of row / remainingHeight;
        real height = row[last] / width;
    else
        real height = area of row / remainingWidth;
        real width = row[last] / height;

    if (AspectRatio(height, width) <= previousAspectRatio)
    {
        Squarify(tail(children), row);
    }
    else
    {
        AddRowToLayout(row.Remove[last]);
        if (drawVert)
            remainingWidth -= width;
        else
            remaningHeight -= height;
        previousAspectRatio = 0;
        Squarify(children, row.Clear());
    }
}
```

*Figure 38: Squarified treemap algorithm pseudo-code. Elaborated from the pseudo-code in (Bruls, et al., 2000).*

The downside to this algorithm is that computation time is increased. Each rectangle needs to be checked to see if the aspect ratio is improved, resulting in more passes through the dataset.

The order of the dataset is also important. Unlike the slice and dice algorithm, the squarified algorithm requires a descending order to produce optimum results (Bruls, et al., 2000). This requires more computation, and the initial order of the data is lost. This isn’t too much of a problem as the
data hierarchy will ensure it remains in a certain order. The order in which the rectangles are drawn inside the hierarchies should not be of primary importance. Drawing from largest to smallest has the benefit of providing an immediate visual ‘ordering’ of the data values. The search functionality will also help obviate any difficulty in a lost ordering of the data.

5.12.3 Normalisation
Unfortunately, it is unlikely that the areas of the rectangles will equal the area of the initial rectangle, i.e. the drawing area. Therefore the areas will require normalising so that all of the drawing area is utilised. A normalisation ratio is calculated by dividing the drawing area by the total area dimensions of the dataset.

5.12.4 Hierarchy
The control in its current state works well as a ‘flat’ treemap, without a hierarchy. This visualisation has merit, as users may only have a flat data structure to visualise. It is therefore decided to create the hierarchical treemap as a separate control. Although the code will be mostly identical, and could be easily combined, providing a separate control enables the user to pick a visualisation technique that matches a dataset more closely.

The hierarchical treemap control converts the flat dataset into a hierarchy using a hierarchy property, and the language integrated query (LINQ) GroupBy method, as shown in Figure 39.

```csharp
hierarchicalData = hierarchicalData.GroupBy(x => x.Hierarchy).Select(
grouping => new TreemapNode {
    NodeName = grouping.Key,
    Children = grouping.Select(leafNode => new TreemapNode {
        NodeName = leafNode.Name,
        Description = leafNode.Description,
        Hierarchy = leafNode.Hierarchy,
        NormalisedArea = leafNode.NormalisedArea,
        Value = leafNode.Value,
        ColourValue = leafNode.ColourValue,
    }).ToList()
}).ToList();
```

Figure 39: Converting a flat data structure to a hierarchical data structure using LINQ.

The user control TreemapNode that is used to display the rectangles contains a property called Children, which is of type List<TreemapNode>. This enables the object to represent a hierarchy.

It was initially envisaged to recursively loop through the hierarchical dataset to calculate the rectangle sizes, this would ensure that all child elements are visited in one pass. Unfortunately this isn’t possible. The parent nodes need to have their sizes calculated first, so that the children know their container size, which means that all parent nodes at a particular level have to be calculated first, before any of their children.

A common problem with treemaps is how to visually denote the hierarchy levels. Many use labels that waste space and distort rectangle sizes (see Figure 15). To prevent this issue, a method in TreemapNode checks if it contains any children, if it has, the rectangle is given a different stroke colour and thickness to visually group its children.
5.12.5 Labels
Only the parent labels are shown initially. This prevents the visualisation being cluttered, as parent and child labels will overlap. If the rectangle size is below a certain threshold the label isn’t shown. This prevents labels being reduced to just a couple of characters and ellipses to denote it has been trimmed, a problem which is noted in empirical treemap research (Gemignani, 2009).

5.12.6 Tooltips
It is envisaged that the leaf nodes will be of primary importance as these encode the actual data values, parents are just containers whose size is dictated by its children. Therefore, tooltips are shown only on the child rectangles, parent rectangles are ignored by setting the IsHitTestVisible property to false.

5.12.7 Zoom
As part of the interactive pipeline zoom stage, the hierarchical treemap control allows the user to zoom in on a certain hierarchy level. This functionality is provided as a context menu that is shown when a user right taps (i.e. a long right click) on a rectangle, this prevents the need for additional controls. The parent’s children are then recalculated using the entire drawing area as the container size to produce a zoomed-in view.

A TextBlock in the interactive panel indicates which level the user is currently zoomed in to, this provides a navigational cue to prevent the user becoming lost. The user can zoom back out using the same context menu.

5.13 Heatmap
5.13.1 Properties
Properties to identify the rows and columns are required. The row identifier is a string that uniquely identifies each data entry. The columns property is a string array that contains the names of the columns that are to be plotted.

5.13.2 Plotting
Heatmaps have an intrinsic scrolling issue. Displaying a large dataset on screen requires a decision to be made on horizontal/vertical scrolling. The width of the columns do not encode a value, so it is decided that the width of the heatmap should fit the actual screen width. This has the benefit of showing all data dimensions on screen at once without the need for horizontal scrolling. For the rows, it is decided to allow vertical scrolling. The screen height simply cannot accommodate a satisfactory amount of data entries, so for datasets of substantial size, vertical scrolling is the only option. The downside of this approach is that the user has to scroll around the visualisation looking for areas of interest, and comparing data entries which are not visible on screen at the same time is difficult, if not impossible. However, these problems can be overcome through the use of interactive features such as filtering and reordering.

To enable these interactive features, a ListView is required to show the data. As described in section 5.10.3, the ListView supports dynamic runtime modification when bound to an ObservableCollection, and supports reordering via drag and drop. A ListView is also used for the column headings, this allows the columns to be rearranged, allowing further exploration of the dataset.

The default ListView Style consumes a lot of screen space through generous padding and margin values. This is desirable for normal applications to separate content, but for data visualisation, the rows need to be arranged together in a matrix to enable comparisons to be drawn. A new Style is created in XAML which overrides the default appearance. The margin and padding values are set to...
zero, which is achieved by applying a ControlTemplate to the ListViewItem, this enables the appearance to be completely customised (Petzold, 2013).

Each data entry is created as a ListViewItem which has a child Grid that contains the cells that are to be coloured. The first column is the name of the row. A TextBlock is added to display this, and a ToolTip is provided to handle any text trimming that may be required. The columns property is then used to generate the rest of the cells. The Grid is then set as the content property of the ListViewItem.

The data values are converted to colours using the linear transformation method described in Figure 27. A Rectangle is added to the grid column to fill the cell with the appropriate colour. The ListViewItems are added to an ObservableCollection which is then bound to the ListView, which enables the interactive reordering and filtering to work.

5.13.3 Reordering
Both the columns and rows can be reordered to aid in data exploration. The column reordering code is identical to the parallel coordinates reordering of axes, which is described in section 5.10.3. For the row reordering, a method swaps the data entries using the old and new indices.

5.13.4 Sorting
Another interactive feature is the ability to sort the data based on values from a particular column. This functionality is enabled by subscribing to the PointerPressed event on the column heading ListViewItems. The name of the column is retrieved in the event handler, this is then used to get the accompanying index. The dataset can then be sorted using this index and the LINQ Order method.

This technique means that additional controls are not required. The column that is selected to sort by is given a highlighted background via the Style of the ListView. This provides a visual appearance as to which column the data is sorted by.

5.13.5 Filtering
The rows of the heatmap can be selected (multiple at a time). A Button is then enabled which can be used to filter the selections, satisfying the zoom and filter stage of the interactive pipeline. This technique also solves the issue with vertical scrolling not being a good enabler for comparisons.

5.13.6 Colour
The only difference in the use of the colour class for the heatmap control is that the qualitative colour scheme type is not required. A heatmap encodes numeric data as colours, so a qualitative colour scheme will not make sense.

5.14 Summary
This section has established that the creation of software is required to meet the goals set out in the literature review. The design and development of the software has been described, with the literature review referenced throughout to justify the decisions that were made. Five user controls have been created which implement specific visualisation techniques that are able to scale well to larger datasets, and which have been demonstrated to have omitted refinement considerations in the past. These techniques also require interaction to be fully effective. The rationale behind the inclusion of interactivity has also been described.
6 Results and analysis

This section will examine how the controls that have been produced, meet the goals that were established in the literature review.

To begin, it is worth reiterating the goals of this work. This can then act as a framework to gauge how effectively the controls have been in meeting them:

1. Traditional data visualisation techniques do not scale well to the more modern, larger dataset. These traditional techniques have been built upon the expertise of data visualisation experts such as Tufte. New techniques have been developed to suit larger, multivariate data, but have neglected the accepted knowledge of the experts in terms of refinement.
2. This is particularly apparent in software, which has exacerbated the situation by providing too many options to users. Users who do not have the data visualisation expertise are therefore free to break accepted rules.
3. Interaction is vital to modern data visualisation due to the complexity of larger datasets. Refinement has subsequently suffered as the user interface has been overloaded with interactive controls.

The visualisation techniques implemented by the controls have been specifically chosen to meet the first goal, as they represent ways of visualising larger, multivariate datasets that traditional wisdom has tended to overlook.

6.1 Refinement

The refinement improvements of the controls will be measured to answer the first two goals. It is worth stressing again that this is not just an aesthetic consideration (if it all), it ensures the data is being visualised correctly without distortion. Tufte’s data-ink and lie factor rules will provide a test environment to be able to measure the refinement.

6.1.1 Data-ink-ratio

The use of non-data-ink is difficult to quantify. In Tufte’s examples, the non-data-ink usage is approximated rather than measured. This is partly because the actual definition of non-data-ink is difficult to pin-point. It was argued in section 4.7.1 that a strict removal of all non-data-ink is dangerous in terms of harming readability and familiarity. As such, only the obvious uses of non-data-ink will be investigated. These are:

- **Grid lines.** These should only exist to aid the creation of a visualisation. They should not exist in a final graphic.
- **Shading.** Specifically, shading that doesn’t encode any variables.
- **Gratuitous decoration.** Anything in a graphic that could be classed as embellishment, as discussed in section 4.7.3.
- **Bilateral symmetry.** Wasted ink which repeats the data.

The controls will be checked visually to ensure that they are not utilising any of these techniques.

6.1.2 Lie factor

As discussed in section 4.7.3, Tufte’s lie factor is calculated by measuring the change of a visual encoding, and dividing this by the actual change in the underlying data. A result of one indicates that the graphic is representing the numbers accurately. Tufte accepts a tolerance of ±0.05.
This provides a quantifiable metric that can be used to measure the controls. To accomplish this, the following test is proposed:

1. The visualisation is screen captured and pasted into a graphics program. Paint.NET will be used for the test (Paint.NET, 2013).
2. Paint.NET has a measurement plug-in available (ComSquare, 2013), which can be used to measure the size of the encodings. This can be seen in Figure 41.
3. Two random encodings are measured (in pixels) to provide the change in the graphic.
4. The underlying data values of these two encodings provide the data change.
5. These two values are used to calculate the lie factor.

The results of each control’s lie factor test will be shown within their respective sections.

6.2 Interaction

The use of interactivity will be assessed to answer the third goal. The UI design described in section 5.5 purposely limited the available space for interactive controls. This forced a judicious use of interactive feature. This results in a minimal impact on graphical refinement as the majority of the screen is utilised for visualisation. To gauge the efficacy of the interactive controls, a sample dataset will be manipulated to show how interesting findings can be derived from it. This serves to validate the features that have been included in terms of aiding the hypotheses generation cycle established in section 4.8.2. All of this will be done within the context of the interactive pipeline to ensure that all stages have been satisfied.

6.2.1 Generic interactive features

The following interactive features are common throughout the controls. The specific interactive features of the controls will be discussed within their own sections.

Each control displays the entire dataset initially, therefore satisfying the first stage of the pipeline – overview first.

All controls contain a search function which enable users to zoom in on areas of interest, forming part of the zoom and filter stage.

The details on demand pipeline stage is met through the provision of tooltips that provide detailed information about each data entry. Each control implements this through mouse rollover. This removes the need for any additional controls.

6.2.2 Multi-functioning interactive elements

This work takes the principle of Tufte’s multi-functioning graphical elements (see section 4.7.2) and applies it to the modern, interactive world of data visualisation, something that Tufte fell short of discussing. Where possible, the interactive elements have been created with multi-functioning in mind. They supply a design function that would otherwise be provided by non-data-ink, and also act as a means to interact with the dataset.

6.3 Parallel coordinates

6.3.1 Data-ink-ratio

The only potential use of non-data-ink are the axis range sliders. In this instance however, the sliders are multi-functioning interactive elements, which gives them immunity from this rule. The opacity of the range sliders are set to ensure that they do not overly distract from the data.
6.3.2 Lie factor test

The parallel coordinates lie factor test requires the height of where the line crosses the axis to be measured. The overall axis length (in pixels), the axis maximum and axis minimum data values will also have to be taken into account. The standard normalisation equation (Equation 3) is used as the heights are measured from the bottom of the graphic, not the top.

The US education statistics dataset (National Center for Education Statistics, 2012) is used for this test, using ‘Idaho’ (highlighted) and the ‘reading’ axis as the test subjects, see Figure 40.

To perform a measurement, a selection is made from the bottom of the axis to where the line crosses the axis. The measurement plug-in then displays the result in pixels, see Figure 41. This is then divided by the axis height, which in this case is 1000px, to get the size effect as shown in the graphic, see Equation 6.

$$\frac{\text{measuredHeight}}{\text{axisHeight}} \times 100$$

Equation 6: Parallel coordinates line height percentage equation.
Idaho reading score = 541. Reading axis maximum = 610. Reading axis minimum = 466.

Percentage change in the data:

$$\frac{541 - 466}{610 - 466} \times 100 = 52.1\%$$

Percentage change in the graphic:

$$\frac{522 \text{ px}}{1000 \text{ px}} \times 100 = 52.2\%$$

Which results in a lie factor of:

$$\frac{52.2}{52.1} = 1.002$$

6.3.3 Interaction

Range sliders are provided to satisfy the zoom and filter stage of the pipeline. This multi-functioning interactive element enables the user to focus in on specific values of interest, as well as providing a visual appearance for the axes. The filtered items are not eliminated from the display, their opacity is adjusted to fade them into the background to ensure that the overall context isn’t lost.

A parallel coordinates plot can be hard to read due to the quantity of lines that are drawn. Data entries can be selected to highlight them, which gives the user a clear visual on the route the line takes between axes. This also satisfies the zoom and filter stage of the pipeline by allowing a user to focus in on an area (or areas, multi-selection is also enabled) of interest.
Lines are highlighted on mouse rollover so the user knows which data entry the tooltip is referring to. This is invaluable in a busy parallel coordinates plot. Tooltips are also provided on the range sliders to give the user an indication as to the actual axis values.

### 6.3.4 Interactive elements in action

One of the benefits of parallel coordinates is the ability to spot correlation, as shown in Figure 6. This involves rearranging the axes to test all combinations to see if the lines cross or not (indicating positive or negative correlation). This functionality is provided by the multi-functioning axis labels. These can be dragged and dropped to rearrange the order the plot is drawn in.

The overview first stage of the parallel coordinates plot can be seen in Figure 40. There is a hint of negative correlation but it isn’t clear. If however the ‘dropout rate’ and ‘percentage of graduates sat’ axes are swapped (see Figure 42) something interesting occurs. There is a clear negative correlation between the writing, reading and math scores, and the percentage of graduates who took the SAT test, a discovery that would be of interest to an education expert.

![Rearranged parallel coordinates axes to highlight correlation.](image)

The range sliders also perform the useful function of allowing filtering by mean, median and quartiles. In Figure 42 the reading axis is filtered by the median, which is 523. This helps to further highlight the correlation, and to provide additional information about the dataset.

### 6.4 Treemaps

#### 6.4.1 Data-ink-ratio

Gratuitous decoration is minimised by providing thin borders for the rectangles. Hierarchies are denoted by thicker borders, but the fact that this overcomes the need for headings that waste screen space makes this an acceptable approach.

Shading is used as an encoding feature which makes it exempt from being classed as non-data-ink.
6.4.2 Lie factor test

The billion dollar-o-gram dataset (McCandless, 2013) is used for this test, using the ‘Money in offshore tax havens’ (1), and ‘Global financial crisis’ (2) nodes as test subjects, see Figure 43.

![Figure 43: Treemap lie factor test.](image)

As it is area that encodes the data values, this needs to be calculated from the measured size. This is then input into the lie factor equation.

1. Data value = 21000. Measured size = 716px x 607px. Area = 434,612px².
2. Data value = 15000. Measured size = 716px x 433px. Area = 310,028px².

Percentage change in the data:

\[
\frac{21000 - 15000}{15000} \times 100 = 40\%
\]

Percentage change in the graphic:

\[
\frac{434612 - 310028}{310028} \times 100 = 40.18\%
\]

Which results in a lie factor of:

\[
\frac{40.18}{40} = 1.0045
\]

6.4.3 Other refinements

Figure 44 juxtaposes the original slice and dice layout from Figure 15, with the new squarified layout to gauge the overall refinement.
The squarified layout improves the overall readability as it is less cluttered. The closely matched aspect ratios makes comparisons easier, which also improves the accuracy of the presentation. The squarified nodes are also easier to point at and select, which is important in an interactive environment.

6.4.4 Interaction
The hierarchies of the treemap can be zoomed into. The display is then updated to show only the children of that particular parent. This functionality is provided by a context menu which bypasses the need for any further interactive controls. This satisfies the zoom and filter stage of the pipeline. A label is updated inside the interactive panel to indicate which hierarchy the user is zoomed into. This provides context and helps prevent any navigational confusion. The same context menu is used to zoom back out to the overview stage.

6.4.5 Interactive elements in action

6.4.5.1 Flat treemap
Figure 45 shows a treemap of US federal spending in 2008 (Riedl, 2008). What is interesting about this dataset is that it includes the cost of the A.I.G. bailout (which the search function has highlighted). The real cost of the bailout can therefore be compared by placing it in context with other federal spending. The ordering of the algorithm makes it easy to see what the rescue fund could otherwise have been used for. The colour scheme emphasises this, as hues lighter than A.I.G. are all areas that cost less than the bailout.
6.4.5.2 Hierarchical treemap

Figure 46 shows a hierarchical treemap of the FTSE 100 (London Stock Exchange, 2014). Treemaps are well suited to visualising the stock market as the area can encode market capitalisation, and colour can encode performance. The hierarchical treemap organises the data by industry sector. Investors are likely to be familiar with certain industry groups, thus enabling them to look at overall capitalisation, and to visualise performance across that sector. The user can then zoom in to a specific sector they are interested in to examine the constituents. The red-yellow-green diverging colour scheme is a natural fit for encoding the performance. Sectors that have performed well (E.g. mining), and those that haven’t (E.g. oil and gas producers) are easily spotted.
6.5 Graduated symbol map

6.5.1 Data-ink-ratio

There is a possible use of non-data-ink in terms of redundancy through bilateral symmetry. This is due to the map showing a repeated globe as it is flattened onto a 2D display, see Figure 47. This is purposely set to give the user maximum control over where they want to focus the map.

Tufte advocates the use of redundancy in this instance as it provides continuity and an aesthetic balance.

It could be argued that the map detail is gratuitous decoration. As the user zooms into the map, further detail is shown such as place names, roads and rivers. This can distract from the data, especially if the dataset is only concerned with aggregated country or continent data. The lower level detail is therefore redundant to a certain extent. This is a difficult problem to overcome. It is impossible to anticipate the required level of geographic detail, and as the mapping functionality is provided by an external library, overall control over the appearance of the map is lost. The use of custom tiles is a difficult solution to implement, as section 5.11.4 established. The interactive resizing of the bubbles helps overcome any difficulty in viewing the data against a detailed background.

6.5.2 Lie factor test

The world adolescent fertility rate dataset (The World Bank, 2014) is used for this test, using the Central African Republic and South Africa as test subjects (see highlighted in Figure 48). The largest bubble size is set to 30, and the bubbles are scaled up by a factor of three due to pixelation causing measurement difficulty.

Figure 47: A repeated globe view.
Central African Republic fertility rate = 106.6. Measured diameter = 108px. Area = \( \pi \times 54^2 = 9160.884 \text{ px}^2 \).

South Africa fertility rate = 59.2. Measured diameter = 80px. Area = \( \pi \times 40^2 = 5026.548 \text{ px}^2 \)

Percentage change in the data:
\[
\frac{106.6 - 59.2}{59.2} \times 100 = 80.07\%
\]

Percentage change in the graphic:
\[
\frac{9160.884 - 5026.548}{5026.548} \times 100 = 82.25\%
\]

Which results in a lie factor of:
\[
\frac{82.25}{80.07} = 1.03
\]

6.5.3 Interaction
The use of Bing maps as a map provider means that interactive panning and zooming is already provided.

A bubble resizing slider is included to help control overall readability. This is especially useful if lots of bubbles reside in close proximity.

Breadcrumb navigation is enabled to provide a navigational prompt, and to allow the user to focus in on specific geographic locations.
6.5.4 Interactive elements in action
Figure 49 shows the worldwide alcohol consumption rate (World Health Organization, 2007). At the overview first stage the dataset confirms expectations, alcohol consumption being greater in the west for example. But if the map is zoomed into Africa, an interesting outlier is spotted. Uganda has the largest alcohol consumption in the world, a surprising find considering that all neighbouring countries consume relatively low amounts. The number of colour scheme classes has been set to emphasise this point.

![Graduated symbol map of the worldwide alcohol consumption rate.](image)

6.6 Heatmap

6.6.1 Data-ink-ratio

The only possible use of non-data-ink are the cell borders, but they provide a meaningful role in separating the cells, preventing the visualisation becoming a mass of colour. The borders are kept thin so as not to overly distract from the data.

6.6.2 Lie factor test

The heatmap control doesn’t use a measurable visual encoding, therefore the lie factor test cannot be performed on this control. The use of the linear transformation method (see Equation 2) to apply colour takes into account the full range of values, so therefore accurately portrays the underlying data. Figure 27 describes this in detail.

6.6.3 Other refinements

The use of a dedicated style for the ListView is vital in creating a matrix type arrangement. This ensures there are no gaps between cells which would otherwise hinder comparisons.

6.6.4 Interaction

Column labels are multi-functioning interactive elements that enable the order to be rearranged through drag and drop, and if selected, the data is sorted in descending order on that particular column. The label is highlighted so as to give a visual prompt as to which column the data is being sorted by.
Row labels also serve as multi-functioning elements. They can also be rearranged via drag and drop, and can be selected to highlight the row and to enable possible filtering.

Filtering is provided to meet the zoom and filter stage of the pipeline. This enables a user to get all areas of interest on the screen at once, which obviates the vertical scrolling issues established in section 5.13.2.

6.6.5 Interactive elements in action

To demonstrate the interactivity, the results of the national student survey from 2012 (The Guardian, 2012) will be examined from the point of view of a prospective student. The user can filter by entry tariff to remove the institutions for which they do not qualify. Another potential preference is for a reasonable staff to student ratio. This column is filtered for any ratios higher than 20. The user now has a manageable list to work with, see Figure 50. This can now be sorted by important factors such as student satisfaction and average teaching score to enable the user to find their ideal choice of university.

6.7 Limitations

The parallel coordinates range sliders are not 100% complete. There are slight inaccuracies in resetting the sliders which is partly the reason why a reset button is provided (although this is useful to reset the sliders all at once). This is acceptable to prove the interactivity, but for production, this would have to be addressed.

There exists a limit on the amount of data entries that a Canvas or Grid can display. In x86 mode a parallel coordinates plot with over 1000 data entries crashed when the axes were rearranged. This isn’t a problem in x64 mode. Virtualisation is a technique that can be employed to prevent this, but this assumes that the display is only showing a subset of the data, whereas to satisfy the overview first stage of a visualisation interactive pipeline, the entire dataset needs to be shown.

The hierarchical treemap is limited to a simple two level parent-child hierarchy. This is sufficient to prove the goals of this work in terms of refinement and interactivity. However, there may exist a
requirement to visualise arbitrary deep hierarchies. There is an inherent difficulty in converting a flat data structure to a hierarchical data structure when the number of levels isn’t known beforehand. The use of recursion and LINQ proved a difficult solution to implement.

The level of detail in the graduated symbol map is a concern. Although interactive elements have been provided to try and obviate the difficulty of visualising the bubbles against a detailed map background, it would be preferable for the map to be less cluttered with geographical information. This helps to minimise non-data-ink and to enable the user to focus on the data.

6.8 Summary
The results indicate a successful combination of the diverse skill sets that are involved in data visualisation. The traditional wisdom of Tufte has been brought up to date by applying it to the latest visualisation methods. The user interface expertise has been added in terms of interactivity, and graphical design principles have been incorporated through careful refinement.

This has been achieved chiefly by limiting the involvement of the user to just the bare minimum. Users who aren’t familiar with the principles of data visualisation are therefore able to produce an accurate visualisation without any data distortion or lack of graphical integrity.

The data-ink-ratio tests prove that the data, and only the data is shown. There is no gratuitous embellishment occurring which may distract from the story the visualisation is trying to depict. Graphical integrity therefore remains uncompromised. The use of multi-functioning interactive elements being a significant factor.

The lie factor results prove that there is no data distortion involved. The normalisation of data value to screen space means that the lie factor is always within accepted tolerance. The lie factor results should in fact equal one exactly as a result of the normalisation calculations. The slight variations in results can be attributed to the lack of measurement precision due to pixelation.

Testing the interactive elements to make interesting data discoveries grounds the controls in real implementations, as opposed to using theoretical, and potentially contrived scenarios. The fact that data discoveries were made proves that each stage of the interactive pipeline has been satisfied, and that a good balance between providing the optimum amount of interactive features, without overloading the user, has been struck.

A lot of refinement issues arise from a poorly designed colour scheme. The use of the dedicated colour scheme class overcomes the problem of users having to choose their own. Users who are not familiar with the nuances of colour in data visualisation have, until now, had too many options to choose from. This leads to poorly assigned colour schemes that distract from the data, as demonstrated in section 4.7.4.

The overall outcome of this work demonstrates that it is possible to bring the traditional wisdom of data visualisation into the modern, interactive realm, without breaking any of the accepted rules.
7 Conclusion and future work

The traditional wisdom of data visualisation experts such as Tufte is concentrated on static visualisations for the print medium, and is focused on visualisation techniques that do not scale well to the demands of larger datasets. Designers have proposed techniques that are suited to larger datasets, but these have tended to ignore the traditional expertise, particularly in software. This lack of graphical refinement leads to an incorrect portrayal of a dataset.

Interaction is a key component for visualisations that target larger datasets. User interface designers are not well versed in the unique demands data visualisation places on the interface. This has resulted in poor UI designs that overload the user with features, to the detriment of the graphic.

This work has addressed these issues by combining the expert knowledge of each discipline into a suite of data visualisation user controls that are suited to visualising larger, multivariate datasets. The traditional data visualisation wisdom has been applied to techniques that are better equipped to visualising modern datasets. The user interface has been specifically designed to maximise screen space, and to limit the amount of interactive features that are available to the user, thus minimising the impact on the graphic. The end product is a set of tools for programmers who may not be familiar with data visualisation principles. This is in contrast to the tools that currently exist which target the reverse, statisticians with little or no programming experience.

The results show that graphical refinement can be upheld even when dealing with larger datasets and modern visualisation techniques. The judicious use of interactivity hasn’t hindered the ability to make interesting data discoveries, and ensures that the impact on the visualisation is minimised. This was enforced by keeping the number of options available to the user down to a minimum. This wasn’t a deliberate ploy, but as the work progressed it became clear that software has been an enabler for bad data visualisation. An interesting area of future work would be user acceptance testing of the software to discover if users find this approach acceptable. This can be extended into a full experiment where sample datasets are put to test subjects, and then posed questions to see how accurate the visualisations are. The findings of this will serve as an interesting addition to the embellishment versus minimalism debate that continues to rage.

Another key finding during the discussion was a need for a generic set of tools, as opposed to the specific tools that research has tended to focus on that are particular to certain domains. The visualisation techniques implemented in this work have therefore been based on existing techniques that have proved useful in visualising large datasets. As these techniques are already in existence, they are widely accepted by the visualisation community, and have found uses in numerous domains. This opens up another potential area of future work in which the controls produced in this work can be used in more comprehensive data visualisation software. The generic usefulness of the controls can therefore be proved by using them in software that has specific goals, such as animations, matrices or small multiples.

The controls were designed so as not to overly leverage the target environment. For example, interactive controls could have been hidden in app bars to maximise screen space, or semantic zooming could have been used to implement hierarchical treemap zooming. These types of techniques are very specific to WinRT. People picking up this work with different target environments in mind might have difficulty in transferring these techniques across. Instead, only standard controls were used for interactivity and display, all of which are widely available in alternative environments. The target environment in this work is therefore largely irrelevant, there is no reason why the methodology described in this work couldn’t be applied to other environments.
As the data explosion continues unabated, there exists a pressing need to understand data across all fields of human knowledge. This work can be expanded to investigate such future demands. The consumption of live data from sensors, web feeds, or the cloud can be examined to place this work at the very cutting edge.

As Nathan Yau points out, data represents real life. As long as there exists people with a desire to understand what is going on around them, there will be a need for data visualisation. There is more data being produced now than ever before. The need to visualise it accurately to derive information is vital. This work has established the problems involved in achieving this desired outcome, and through the careful creation of data visualisation controls, has demonstrated how the problems can be overcome.
8 References


[Accessed 3 January 2014].


[Accessed 21 December 2013].


[Accessed 9 February 2014].


McCandless, D., 2013. *$Billions*. [Online] Available at: https://docs.google.com/spreadsheet/ccc?key=0Aqe2P9sYhZ2ndGxWNjVkQmNCVIR3Q0JkZmF4WmV4SEE#gid=13 [Accessed 2 December 2013].


Ware, C., 2004. *Information Visualization - Perception For Design*. 2nd ed. San Francisco: Morgan Kaufmann.


