Data Visualization (CIS 468)

Focus+Context

Dr. David Koop
Overview

Reducing Items and Attributes

- **Filter**
  - Items
  - Attributes

- **Aggregate**
  - Items
  - Attributes

Reduce

- **Filter**
- **Aggregate**
- **Embed**

[Munzner (ill. Maguire), 2014]
Restaurant locations are derived from the New York City Department of Health and Mental Hygiene database. Due to the limitations of the Health Department’s database, some restaurants could not be placed.

By JEREMY WHITE

Source: New York City Department of Health and Mental Hygiene

© 2013 The New York Times Company

New York Health Department Restaurant Ratings Map

The New York City Department of Health and Mental Hygiene performs unannounced sanitary inspections of every restaurant at least once per year. Violation points result in a letter grade, which can be explored in the map below, along with violation descriptions. The information on this map will be updated every two weeks. For menus and reviews by New York Times critics, visit our restaurants guide.

Related Article »

Gracie’s Cafe
Grade Grade pending
Violation points 27
Click for details

Gracie’s Cafe
Grade Grade pending
Violation points 27
Click for details

Improve chemicals
14+ points

Find a Restaurant
Find a Location
Filter

[J. White, New York Times]
Scented Widgets

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hue</td>
<td>Varies the hue of the widget (or of a visualization embedded in it)</td>
<td><img src="image" alt="Option A" /> <img src="image" alt="Option B" /></td>
</tr>
<tr>
<td>Saturation</td>
<td>Varies the saturation of the widget (or of a visualization embedded in it)</td>
<td><img src="image" alt="Option A" /> <img src="image" alt="Option B" /></td>
</tr>
<tr>
<td>Opacity</td>
<td>Varies the saturation of the widget (or of a visualization embedded in it)</td>
<td><img src="image" alt="Option A" /> <img src="image" alt="Option B" /></td>
</tr>
<tr>
<td>Text</td>
<td>Inserts one or more small text figures into the widget</td>
<td><img src="image" alt="Option A" /> <img src="image" alt="Option B" /></td>
</tr>
<tr>
<td>Icon</td>
<td>Inserts one or more small icons into the widget</td>
<td><img src="image" alt="Option A" /> <img src="image" alt="Option B" /></td>
</tr>
<tr>
<td>Bar Chart</td>
<td>Inserts one or more small bar chart visualizations into the widget</td>
<td><img src="image" alt="Option A" /> <img src="image" alt="Option B" /></td>
</tr>
<tr>
<td>Line Chart</td>
<td>Inserts one or more small line chart visualizations into the widget</td>
<td><img src="image" alt="Option A" /> <img src="image" alt="Option B" /></td>
</tr>
</tbody>
</table>

We begin by considering a basic language of visual encodings for navigation. Interactive visualization applications such as sense.us [12] capture a number of social activity metrics that are useful views.
Attribute Filtering on Star Plots

(a) [Yang et al., 2003]
Aggregation: Histograms

Observed ranks of posts by subreddit

- personalfinance
- Showerthoughts
- nosleep
- WritingPrompts
- mildlyinteresting
- explainlikeimfive
- Fitness
- movies
- TwoXChromosomes
- Documentaries
- Art
- philosophy

["The reddit Front Page is Not a Meritocracy", T. W. Schneider]
Common Distributions

Uniform ← Bernoulli

Hypergeometric ← Binomial

Poisson ← Geometric

Normal (Gaussian) ← Exponential

Log Normal ← Chi-Squared

Student’s t ← Gamma

Weibull ← Beta

Negative Binomial
Aggregation in 2D

- Hexagonal bins are more circular
- Distance to the edge is not as variable
- More efficient aggregation around the center of the bin
Modifiable Areal Unit Problem

In cartography, changing the boundaries of the regions used to analyze data can yield dramatically different results.
Modifiable Areal Unit Problem

In cartography, changing the boundaries of the regions used to analyze data can yield dramatically different results.
Modifiable Areal Unit Problem

In cartography, changing the boundaries of the regions used to analyze data can yield dramatically different results.
Aggregation: Boxplots

**How to read**

- 10th percentile
- 25th percentile
- Median annual earnings
- 75th percentile
- 90th percentile

$0  $20k  $40k  $60k  $80k  $100k  $120k  $140k  $160k  $180k  $200k  $220k  $240k

Harvard

UPenn

Princeton

Columbia

Cornell

Dartmouth

Yale

Brown

[Washington Post, 2015]
Four Distributions, Same Boxplot...

[Normal]

[Skewed]

[Bimodal]

[Peaked]

[Box plot]

[C. Choonpradub and D. McNeil, 2005]
Hierarchical Parallel Coordinates

Figure 4: This image sequence shows a Fatal Accident data set of 230,000 data elements at different levels of details. The first image shows a cut across the root node. The last image shows the cut chaining all the leaf nodes. The rest of the images show intermediate cuts at varying levels of detail. (See Color Plates).

Figure 6: Left image shows Iris data set without proximity-based coloring. Right image shows Iris data set with proximity-based coloring revealing three distinct clusters depicted by concentrations of blue, green and pink lines. (See Color Plates).

[Fua et al., 1999]
Assignment 4

• Link
• Interaction, Network, and Multiple Views
• Due before Thanksgiving, but remember Quiz 2!
Quiz 2

• Next Tuesday
• Similar Format to Quiz 1:
  - Multiple Choice and Free Response
• First half of the class period
• Focus on material from Midterm through this Thursday
Attribute Aggregation

• Remember reducing attributes—use statistics: either one variable matches another or doesn't change!
• We can also use similar criteria for aggregating attributes
• **Cluster** similar attributes together
  - How?
K-Means

[C. Polis, 2014]
K-Means Issues

Shape

Number of Clusters

[D. Robinson, 2015]
Dimensionality Reduction

- **Attribute Aggregation:** Use fewer attributes (dimensions) to represent items
- **Combine attributes** in a way that is more instructive than examining each individual attribute
- **Example:** Understanding the language in a collection of books
  - Count the occurrence of each non-common word in each book
  - Huge set of features (attributes), want to represent each with an aggregate feature (e.g. high use of "cowboy", lower use of "city") that allows clustering (e.g. "western")
  - Don't want to have to manually determine such rules
- **Techniques:** Principle Component Analysis, Multidimensional Scaling family of techniques
Principle Component Analysis (PCA)
Principal Component Analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize.

First, consider a dataset in only two dimensions, like (height, weight). This dataset can be plotted as points in a plane. But if we want to tease out variation, PCA finds a new coordinate system in which every point has a new (x,y) value. The axes don't actually mean anything physical; they're combinations of height and weight called "principal components" that are chosen to give one axes lots of variation.

Drag the points around in the following visualization to see PC coordinate system adjusts.

### 17 dimensions to 2

<table>
<thead>
<tr>
<th></th>
<th>England</th>
<th>N Ireland</th>
<th>Scotland</th>
<th>Wales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alcoholic drinks</td>
<td>375</td>
<td>135</td>
<td>458</td>
<td>475</td>
</tr>
<tr>
<td>Beverages</td>
<td>57</td>
<td>47</td>
<td>53</td>
<td>73</td>
</tr>
<tr>
<td>Carcase meat</td>
<td>245</td>
<td>267</td>
<td>242</td>
<td>227</td>
</tr>
<tr>
<td>Cereals</td>
<td>1472</td>
<td>1494</td>
<td>1462</td>
<td>1582</td>
</tr>
<tr>
<td>Cheese</td>
<td>105</td>
<td>66</td>
<td>103</td>
<td>103</td>
</tr>
<tr>
<td>Confectionery</td>
<td>54</td>
<td>41</td>
<td>62</td>
<td>64</td>
</tr>
<tr>
<td>Fats and oils</td>
<td>193</td>
<td>209</td>
<td>184</td>
<td>235</td>
</tr>
<tr>
<td>Fish</td>
<td>147</td>
<td>93</td>
<td>122</td>
<td>160</td>
</tr>
<tr>
<td>Fresh fruit</td>
<td>1102</td>
<td>674</td>
<td>957</td>
<td>137</td>
</tr>
<tr>
<td>Fresh potatoes</td>
<td>720</td>
<td>1033</td>
<td>566</td>
<td>874</td>
</tr>
<tr>
<td>Fresh Veg</td>
<td>253</td>
<td>143</td>
<td>171</td>
<td>265</td>
</tr>
<tr>
<td>Other meat</td>
<td>685</td>
<td>586</td>
<td>750</td>
<td>803</td>
</tr>
<tr>
<td>Other Veg</td>
<td>488</td>
<td>355</td>
<td>418</td>
<td>570</td>
</tr>
<tr>
<td>Processed potatoes</td>
<td>198</td>
<td>187</td>
<td>220</td>
<td>203</td>
</tr>
<tr>
<td>Processed Veg</td>
<td>360</td>
<td>334</td>
<td>337</td>
<td>365</td>
</tr>
<tr>
<td>Soft drinks</td>
<td>1374</td>
<td>1504</td>
<td>1572</td>
<td>1256</td>
</tr>
<tr>
<td>Sugars</td>
<td>156</td>
<td>139</td>
<td>147</td>
<td>175</td>
</tr>
</tbody>
</table>

Non-linear Dimensionality Reduction

original data space $\mathcal{X}$

component space $\mathcal{Z}$

$\Phi_{gen} : \mathcal{Z} \rightarrow \mathcal{X}$

$\Phi_{extr} : \mathcal{X} \rightarrow \mathcal{Z}$
Dimensionality Reduction in Visualization

5.2.2 Layout Quality

Fig. 8 shows the visual quality, normalized stress, and timing of Glimmer, Hybrid, and PivotMDS layouts on four data sets with known structure. In the case of grid, the correct shape is known. In the other three cases, the correct partitions of the points into clusters are available with these benchmark data sets, so the extent to which the color coding matches the spatial grouping created by an algorithm is a measure of its accuracy.

Qualitatively, with cancer, the Glimmer and PivotMDS algorithms indicate these two color-coded groups clearly with spatial position. Quantitatively, the stress of Glimmer is an order of magnitude lower than that of PivotMDS. Hybrid does separate the two groups but produces misleading subclusters in the orange group.

With shuttle_big, Hybrid produces a readable layout separating the red cluster from the other two but is slower by several hundred percent. Glimmer and PivotMDS both produce useful and qualitatively comparable layouts separating the clusters. The PivotMDS layout is twice as fast but has noticeable occlusion and much higher stress than the Glimmer layout.

The 10,000-point grid is accurately embedded by Glimmer and PivotMDS in comparable times. Hybrid is again slower but nevertheless terminated too soon, suffering from very noticeable qualitative distortion and with a much higher quantitative stress metric compared to that of the other layouts.

The Glimmer layout of the docs data set is qualitatively better than the other three. It shows several spatially distinguishable clusters, color coded by blue, red, orange, and green. The green cluster is split into three parts. It took approximately 2 seconds with normalized stress of 0.157. Hybrid suffers from cluster occlusion. The stress is nearly twice as high as that of Glimmer, and the spatial embedding does not clearly separate any of the given clusters. PivotMDS is very fast but almost completely fails to show anything useful.

[Glimmer, Ingram et al., 2009]
Tasks in Understanding High-Dim. Data

Task 1

In HD data → Out 2D data

What?  
In High-dimensional data  
Out 2D data

Why?  
Produce  
Derive

Task 2

In 2D data → Out Scatterplot Clusters & points

What?  
In 2D data  
Out Scatterplot

Why?  
Discover

How?  
Encode

Task 3

In Scatterplot Clusters & points → Out Labels for clusters

What?  
In Scatterplot Clusters & points

Why?  
Produce  
Annotate

[Munzner (ill. Maguire), 2014]
Probing Projections

Probing Projections: Interaction Techniques for Interpreting Arrangements and Errors of Dimensionality Reductions

Julian Stahnke, Marian Dörk, Boris Müller, and Andreas Thom

Abstract
—We introduce a set of integrated interaction techniques to interpret and interrogate dimensionality-reduced data. Projection techniques generally aim to make a high-dimensional information space visible in form of a planar layout. However, the meaning of the resulting data projections can be hard to grasp. It is seldom clear why elements are placed far apart or close together and the inevitable approximation errors of any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. There has been only little research on how the visualization resulting from data projection is interacted with. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

Index Terms
—Information visualization, interactivity, dimensionality reduction, multidimensional scaling.

Introduction
A primary goal of information visualization is to find patterns and relationships in multivariate datasets. Many visualization techniques have been developed towards this goal such as multiple coordinated views \cite{2}, parallel coordinates \cite{14}, scatterplot matrices \cite{28}, and dimensionality reductions such as multidimensional scaling (MDS) \cite{5}. Dimensionality reductions are a particular class of techniques that synthesise high-dimensional data spaces onto projection spaces with much fewer dimensions, typically the two dimensions of the plane. While most visualization techniques juxtapose the different data dimensions as matrices or columns, dimensionality reductions integrate them into a planar canvas. The projection results in a so-called spatialisation (i.e., embedding) of data elements that approximately represents similarity as proximity and in turn dissimilarity as distance. Considering that the human perceptional system comprises a well-developed capacity for spatial reasoning, the assumption is that spatialisation would be a more natural way \cite{31} to analyse high-dimensional datasets since groupings, separations, and other patterns among data elements become immediately discernible.

However, there are two major caveats linked with dimensionality reduction: first, it can be challenging to interpret the positions of projected elements, and second, the errors that occur with any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. There has been only little research on how the visualization resulting from data projection is interacted with. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

1INTRODUCTION

A primary goal of information visualization is to find patterns and relationships in multivariate datasets. Many visualization techniques have been developed towards this goal such as multiple coordinated views \cite{2}, parallel coordinates \cite{14}, scatterplot matrices \cite{28}, and dimensionality reductions such as multidimensional scaling (MDS) \cite{5}. Dimensionality reductions are a particular class of techniques that synthesise high-dimensional data spaces onto projection spaces with much fewer dimensions, typically the two dimensions of the plane. While most visualization techniques juxtapose the different data dimensions as matrices or columns, dimensionality reductions integrate them into a planar canvas. The projection results in a so-called spatialisation (i.e., embedding) of data elements that approximately represents similarity as proximity and in turn dissimilarity as distance. Considering that the human perceptional system comprises a well-developed capacity for spatial reasoning, the assumption is that spatialisation would be a more natural way \cite{31} to analyse high-dimensional datasets since groupings, separations, and other patterns among data elements become immediately discernible.

However, there are two major caveats linked with dimensionality reduction: first, it can be challenging to interpret the positions of projected elements, and second, the errors that occur with any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. There has been only little research on how the visualization resulting from data projection is interacted with. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

Index Terms
—Information visualization, interactivity, dimensionality reduction, multidimensional scaling.

1INTRODUCTION

A primary goal of information visualization is to find patterns and relationships in multivariate datasets. Many visualization techniques have been developed towards this goal such as multiple coordinated views \cite{2}, parallel coordinates \cite{14}, scatterplot matrices \cite{28}, and dimensionality reductions such as multidimensional scaling (MDS) \cite{5}. Dimensionality reductions are a particular class of techniques that synthesise high-dimensional data spaces onto projection spaces with much fewer dimensions, typically the two dimensions of the plane. While most visualization techniques juxtapose the different data dimensions as matrices or columns, dimensionality reductions integrate them into a planar canvas. The projection results in a so-called spatialisation (i.e., embedding) of data elements that approximately represents similarity as proximity and in turn dissimilarity as distance. Considering that the human perceptional system comprises a well-developed capacity for spatial reasoning, the assumption is that spatialisation would be a more natural way \cite{31} to analyse high-dimensional datasets since groupings, separations, and other patterns among data elements become immediately discernible.

However, there are two major caveats linked with dimensionality reduction: first, it can be challenging to interpret the positions of projected elements, and second, the errors that occur with any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. There has been only little research on how the visualization resulting from data projection is interacted with. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

Index Terms
—Information visualization, interactivity, dimensionality reduction, multidimensional scaling.

1INTRODUCTION

A primary goal of information visualization is to find patterns and relationships in multivariate datasets. Many visualization techniques have been developed towards this goal such as multiple coordinated views \cite{2}, parallel coordinates \cite{14}, scatterplot matrices \cite{28}, and dimensionality reductions such as multidimensional scaling (MDS) \cite{5}. Dimensionality reductions are a particular class of techniques that synthesise high-dimensional data spaces onto projection spaces with much fewer dimensions, typically the two dimensions of the plane. While most visualization techniques juxtapose the different data dimensions as matrices or columns, dimensionality reductions integrate them into a planar canvas. The projection results in a so-called spatialisation (i.e., embedding) of data elements that approximately represents similarity as proximity and in turn dissimilarity as distance. Considering that the human perceptional system comprises a well-developed capacity for spatial reasoning, the assumption is that spatialisation would be a more natural way \cite{31} to analyse high-dimensional datasets since groupings, separations, and other patterns among data elements become immediately discernible.

However, there are two major caveats linked with dimensionality reduction: first, it can be challenging to interpret the positions of projected elements, and second, the errors that occur with any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. There has been only little research on how the visualization resulting from data projection is interacted with. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

Index Terms
—Information visualization, interactivity, dimensionality reduction, multidimensional scaling.

1INTRODUCTION

A primary goal of information visualization is to find patterns and relationships in multivariate datasets. Many visualization techniques have been developed towards this goal such as multiple coordinated views \cite{2}, parallel coordinates \cite{14}, scatterplot matrices \cite{28}, and dimensionality reductions such as multidimensional scaling (MDS) \cite{5}. Dimensionality reductions are a particular class of techniques that synthesise high-dimensional data spaces onto projection spaces with much fewer dimensions, typically the two dimensions of the plane. While most visualization techniques juxtapose the different data dimensions as matrices or columns, dimensionality reductions integrate them into a planar canvas. The projection results in a so-called spatialisation (i.e., embedding) of data elements that approximately represents similarity as proximity and in turn dissimilarity as distance. Considering that the human perceptional system comprises a well-developed capacity for spatial reasoning, the assumption is that spatialisation would be a more natural way \cite{31} to analyse high-dimensional datasets since groupings, separations, and other patterns among data elements become immediately discernible.

However, there are two major caveats linked with dimensionality reduction: first, it can be challenging to interpret the positions of projected elements, and second, the errors that occur with any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. There has been only little research on how the visualization resulting from data projection is interacted with. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

Index Terms
—Information visualization, interactivity, dimensionality reduction, multidimensional scaling.
Heatmap from Dimension Hover

Fig. 4. Hovering over a dimension in the sidebar displays its distribution as a heatmap in the projection on the left. The heatmap is a grid of cells each representing the value for a certain dimension at its position, with higher values being darker. Brightness is used to avoid confusion with the group colours. This allows to visually assess the value distribution for a given dimension, which can help in comparing elements to see how they compare to other elements.

Fig. 2. A tooltip displays the sample’s absolute values, standard deviations, and graphical representations for each dimension. The deviations are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations. They are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations.

The heatmap is a grid of cells each representing the value for a certain dimension at its position, with higher values being darker. Brightness is used to avoid confusion with the group colours. This allows to visually assess the value distribution for a given dimension, which can help in comparing elements to see how they compare to other elements.

The deviations are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations. They are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations.

The projection space can also be used to answer a more theoretical question: what values would a fictive sample have to have to be projected to a certain spot? Or, phrased differently: what are the deviations, and graphical representations for each dimension. The deviations are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations. They are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations.

Furthermore, hovering over a grouping’s thumbnail displays small points in the projection. Some additional information, such as the name or the number of samples, is displayed below the thumbnail.

Clusters can also be saved and named as selections. On the projection, these groupings are coded by colour, with the dots on the projection and drawing a convex hull around them.

All of these groupings are displayed as panels in the sidebar. Each cluster shows a tooltip that compares these groups (here selections).

After selecting one group of samples, hovering over another allows to visually assess the value distribution for a given dimension, which can help in comparing elements to see how they compare to other elements. The deviations are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations. They are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations.

Comparing elements allows to visually assess the value distribution for a given dimension, which can help in comparing elements to see how they compare to other elements. The deviations are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations. They are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations.

Projections created with most dimensionality-reduction techniques, such as MDS, have no meaningful axes, complicating spatial orientation because dimensional values are distributed nonlinearly. Yet, in figure 2, groupings can also be compared to each other, displaying density plots for each of them. The methods for comparing samples, density plots for them are shown in the list of dimensions, as well as a text-based preview.

Clusters can also be saved and named as selections. On the projection, these groupings are coded by colour, with the dots on the projection and drawing a convex hull around them.

All of these groupings are displayed as panels in the sidebar. Each cluster shows a tooltip that compares these groups (here selections).

After selecting one group of samples, hovering over another allows to visually assess the value distribution for a given dimension, which can help in comparing elements to see how they compare to other elements. The deviations are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations. They are displayed in text form for accuracy, as well as showing the values for the various dimensions, and their standard deviations.
Showing Projection Errors

White: higher levels of similarity
Gray: lower levels of similarity

[Fig. 6. Halos represent the cumulative error for the respective samples. White indicates that a majority of samples is more similar than indicated by their distance to the given sample; grey indicates the opposite. The paths travelled by the points are shown as lines, leading from the points' original positions in the projection to the new, corrected positions (see Figure 8). This connects them to their original positions in the projection, and displays the size of the distance error at the same time. Resembling the brightness encoding of the halos, the brightness of the lines indicates whether they've moved closer or farther away.

A problem with this solution is that it introduces new distortions in the spatial relationship between all other points. Only the distances directly between the selected point and the other points are reliable, whereas all the other distances are distorted, and the new positioning might lead to wrong assumptions about potential clusterings. To mitigate this problem, the correction paths are shown. Another solution would be to recompute the projection while preserving the distances from and to the selected point and being more generous with distance errors among the remaining points. This would somewhat reduce the introduced distortions. However, in a recomputed projection, the positions of the points might change significantly, most likely leading to completely different positions for all points, possibly confusing the observer even if an animation is used.]

[Fig. 7. Dendrograms mapped onto the projection. Left: projection with low projection error. Right: high projection error.

4.5.3 Dendrogram In addition to the visualization of errors and corrections, a dendrogram can visualize the samples with regard to their position in the clustering hierarchy. Such a dendrogram (using the same agglomerative algorithm as the clusters) overlaid onto the projection may also help to visualise high-dimensional distances on the projection space.

It graphically emphasises clusters by connecting close dots through dense lines. Interestingly, the dendrogram is a surprisingly good indicator of goodness of fit: if many thick, long lines intersect, it is likely that the projection is of low quality.

EXAMPLE: OECD COUNTRIES To illustrate the functionality of the interface we visualize the dataset of OECD countries in the prototype (see Figure 9). The dataset contains 8 dimensions for 36 countries. First, the viewer is drawn to the projection and notices Turkey that seems to be a clear outlier, far away from all other countries. To explore why this is, the viewer can examine this sample by hovering over it. A tooltip relating Turkey to the rest of the dataset appears, showing that it deviates strongly from the mean in nearly every dimension. This indicates the positioning as outlier is probably correct.

To test this assumption and build up trust in the visualization, the viewer selects 'correct distances', showing the high-dimensional distances between Turkey and the other countries. This reveals that Turkey should be even farther apart from several of the other countries. Having confirmed that Turkey is an outlier in this dataset, the viewer uses the built-in clustering to get a sense of how the countries are grouped. Playing around with the number of clusters, they notice that there seem to be seven clusters roughly corresponding to the geographical and geopolitical placement of the countries.

Taking a closer look at the positioning of the clustered countries, they realise that the arrangement seems to roughly correspond to geographic directions: Northern and Southern countries are roughly distributed along the vertical axes, East and West along the horizontal. To find out if or how this correlates with the dimensions, the viewer first compares the different clusters. Here the differences along the dimensions are very much pronounced. Interestingly though, life expectancy is lower in Latin America than Asia, while the self-reported health is higher for the former than the latter.

After a few more comparisons between the clusters, the viewer becomes interested in the dimension life satisfaction and turns towards the heatmaps. They notice that the values for life satisfaction and self-reported health seem to be higher in the Western countries, whereas the value for employees working very long hours seems to be especially high in the countries of the far East and the South.


[J. Stahnke et al., 2015]
Focus+Context

- Show everything at once but compress regions that are not the current focus
  - User shouldn't lose sight of the overall picture
  - May involve some aggregation in non-focused regions
  - "Nonliteral navigation" like semantic zooming
- Elision
- Superimposition: more directly tied than with layers
- Distortion
Focus+Content Overview

- **Embed**
  - Elide Data
  - Superimpose Layer
  - Distort Geometry

- **Reduce**
  - Filter
  - Aggregate
  - Embed

[Munzner (ill. Maguire), 2014]
Elision

• There are a number of examples of elision including in text, DOITrees, …

• Includes both filtering and aggregation but goal is to give overall view of the data

• In visualization, usually correlated with focus regions
Degree of Interest Function

- DOI = I(x) - D(x,y)
  - I: interest function
  - D: distance (semantic or spatial)
  - x: location of item
  - y: current focus point (could be more than one)
- Interactive: y changes
Elision: DOI Trees

- Example: 600,000 node tree
  - Multiple foci (from search results or via user selection)
  - Distance computed topologically (levels, not geometric)

[Heer and Card, 2004]
Superimposition

• Different from layers because this is restricted to a particular region
  - For Focus+Context, superimposition is not global
  - More like overloading

• Lens may occlude the layer below
Superimposition with Interactive Lenses

(a) Alteration             (b) Suppression

[ChronoLenses and Sampling Lens in Tominski et al., 2014]
Superimposition with Interactive

(c) Enrichment

[Extended Lens in Tominski et al., 2014]
Distortion

It can be difficult to observe micro and macro features simultaneously with complex graphs. If you zoom in for detail, the graph is too big to view in its entirety. If you zoom out to see the overall structure, small details are lost.

Focus + context techniques allow interactive exploration of an area.

Mouseover to distort the nodes.

[M. Bostock]
Distortion Choices

• How many focus regions?
  - One
  - Multiple

• Shape of the focus?
  - Radial
  - Rectangular
  - Other

• Extent of the focus
  - Constrained similar to magic lenses
  - Entire view changes

• Type of interaction:
  - Geometric, moveable lenses, rubber sheet
Overplotting
Cartesian Distortion
Cartesian Distortion
Stretch and Squish Navigation

**Figure 3.** LiveRAC shows a full day of system management time-series data using a reorderable matrix of area-aware charts. Over 4000 devices are shown in rows, with 11 columns representing groups of monitored parameters. (a): The user has sorted by the maximum value in the CPU column. The first several dozen rows have been stretched to show sparklines for the devices, with the top 13 enlarged enough to display text labels. The time period of business hours has been selected, showing the increase in the In pkts parameter for many devices. (b): The top three rows have been further enlarged to show fully detailed charts in the CPU column and partially detailed ones in Swap and two other columns. The time marker (vertical black line on each chart) indicates the start of anomalous activity in several of spire's parameters. Below the labeled rows, we see many blocks at the lowest semantic zoom level, and further below we see a compressed region of highly saturated blocks that aggregate information from many charts.

Principle: multiple views are most effective when coordinated through explicit linking.

The principle of linked views [15] is that explicit coordination between views enhances their value. In LiveRAC, as the user moves the cursor within a chart, the same point in time is marked in all charts with a vertical line. Similarly, selecting a time segment in one chart shows a mark in all of them. This technique allows direct comparison between parameter values at the same time on different charts. In addition, people can easily correlate times between large charts with detailed axis labels, and smaller, more concise charts.

Assertion: showing several levels of detail simultaneously provides useful high information density in context. Several technique choices are based on this assertion.

First, LiveRAC uses stretch and squish navigation, where expanding one or many regions compresses the rest of the view [11, 17]. The accompanying video shows the look and feel of this navigation technique. The stretching and squishing operates on rectangular regions, so expanding a single chart also magnifies the entire row for the device it represents, and the entire column for the parameters that it shows. The edges of the display are fixed so that all cells remain within the visible area, as opposed to conventional zooming where some regions are pushed off-screen. There are rapid navigation shortcuts to zoom a single cell, a column, an aggregated group of devices, the results of a search, or to zoom out to an overview. Users can also directly drag grid lines or resize freely drawn on-screen rectangles. Navigation shortcuts can also be created for any arbitrary grouping, whose cells do not need to be contiguous. This interaction mechanism affords multiple focus regions, supporting multiple levels of detail.

Second, charts in LiveRAC dynamically adapt to show visual representations adapted in each cell to the available screen space. This technique, called semantic zooming [13], allows a hierarchy of representations for a group of device-parameter time-series. In Figure 3, the largest charts have multiple overlaid curves and detailed axis and legend labels. Smaller charts show fewer curves and less labeling, and at smaller sizes only one curve is shown as a sparkline [24]. On each curve, the maximum value over the displayed time period is indicated with a red dot, the minimum with a blue dot, and the current value with a green one. All representation levels color code the background rectangle according to dynamically changeable thresholds of the minimum, maximum, or average values of the parameters within the current time window. The smallest view is a simple block, where this color coding is the only information shown.

Third, aggregation techniques achieve visual scalability by ensuring dense regions show meaningful visual representations. Given our target scale of dozens of parameters and thousands of devices, the size of the matrix could easily surpass 100,000 cells. Stretch and squish navigation allows users to quickly create a mosaic with cells of many different sizes. [McLachlan et al., 2008]
Distortion Concerns

- Distance and length judgments are **harder**
  - Example: Mac OS X Dock with Magnification
  - Spatial position of items changes as the focus changes
- Node-link diagrams not an issue… why?
- Users have to be made aware of distortion
  - Back to scatterplot with distortion example
  - Lenses or shading give clues to users
- **Object constancy**: understanding when two views/frames show the same object
  - What happens under distortion?
  - 3D Perspective is distortion… but we are well-trained for that
- Think about **what** is being shown (filtering) and method (fisheye)
H3 Layout

[T. Munzner, 1998]
H3 Layout

[T. Munzner, 1998]
Focus+Context in Network Exploration

(a) Moderately large graph drawn with straight line edges. The graph nodes correspond to the USA major cities; edges show migration flows. The graph contains 1715 nodes and 9778 edges. Nodes are laid out according to geographical positions of cities, producing a drawing with poor readability, where edges mix in a totally unordered way and where some nodes are close to unnoticeable.

(b) The same graph as in Fig. 1(a) now drawn using edge bundling with edges rendered as Bézier curves.

Figure 1: Illustration of edge bundling.

(a) The fish-eye distorts a small region of the graph for local inspection.
(b) The magnifying lens shows a zoom on a local region.

Figure 2: Fisheye and magnifying lens.

Magnifying Lens and Fish-eye – They magnify the lens [3] and geometrical fish-eye [7] were also added to the system as basic interactors. They allow to get local details on an area of the graph without having to zoom in (see Fig. 2(a) and Fig. 2(b)). These techniques allow to get an estimate of the degree of node or number of edges that have been bundled together, and an idea on the spatial organization of neighborhoods.

Neighborhood highlighting – After edges have been bundled, the graph gains in overall readability at the loss of more local information. For instance, connections between any two particular nodes cannot be easily recovered and isolated out of a bundle. When designing the system and deciding on the interactions to implement and combine, we focused on the recovery of these local information. By hovering the mouse over any node in the graph drawing, the user can highlight its neighborhood. This is accomplished by showing a translucent circle over the immediate where a node sits while clearly displaying the neighborhood of the node (top of Fig. 3(a)). The circle fades off nodes not belonging to the selected neighborhood, temporarily providing a clear view of it. The size of the translucent circle is fitted as to enclose all immediate neighbors of the node in the graph. Using the mouse wheel, the user can select neighbors sitting at a bounded distance from the node. The size of the translucent circle adjusts accordingly (bottom of Fig. 3(b)).

Bring & Go – Now, neighborhoods in the graph don’t always sit close. As a consequence, the translucent circle highlighting neighbors of a node can potentially be quite large. That is, the distance between nodes in the graph does not always match their Euclidean distance in the drawing – [Lambert et al., 2010]
Focus+Context in Network Exploration

(a) Neighborhood highlighting – selecting a node brings up its neighbors, fading away all other graph elements.

(b) Using the mouse wheel, the neighborhood is extended to nodes sitting further away.

Figure 3: Illustration of the Neighborhood highlighting in interaction

This indeed is the challenge posed to all layout algorithms. The Bring & Go technique introduced by Tominski et al. [18] solves this paradox. The Bring operation pulls neighbors of a node to near proximity, temporarily resolving a situation where the layout algorithm had failed. Fig. 4(a) and Fig. 4(b) illustrate this situation – the passage from step 1 to step 2 being smoothly animated. Once the neighbors have been repositioned close to the node, the Go operation lets the user decide of a new direction to move to by selecting a neighbor. After clicking a neighbor node, the visualization is panned until re-centered around the target neighbor. The transition is performed by smoothly animating the pan (see Fig. 3). A recent user-study of this interaction technique has been made by Moscovich et al. [15]. When bringing neighbors close to the selected node, the edges abandon their curve shapes and are morphed to straight lines. This is done by modifying the control points coordinates of each curve so that they are all aligned.

Our system thus comprises a comprehensive palette of interactions focusing on adjacency or accessibility tasks (we borrow this terminology from Lee et al.'s [14] task taxonomy, itself referring to the work of Amar et al. [1]). That is, tasks such as exploring neighbor nodes, or counting them, finding how many nodes can be accessed from any given one, etc., can be easily done through direct manipulation of the graph using zoom, pan, neighborhood highlight or Bring & Go, for instance. All the interactions techniques have been implemented as interactor plugins for the Tulip graph visualization software [2] and are available through its plugin server.

4 Maintaining fluid interaction

The challenge we were faced with is that curves generation have a relatively high computational cost when it comes to interacting with bundles. Indeed, although the curves can be drawn in reasonable time for static drawings using standard rendering techniques, the problem becomes tedious when one wants to interact on bundles using any of the techniques described in the previous section. The curves' shapes must be continually transformed as the user moves the mouse and pilots interaction (geometrical fish-eye or Bring & Go for instance).

Moreover, we did not want fluidity to impact on the quality of the curves and impose an upper bound on the number of control points used to compute the edge routes. Instead, we aimed at producing a system capable of dealing with an arbitrary number of control points. As a consequence, the computation of the points interpolating the curve itself puts a real burden on the system and calls for an extremely efficient approach. The solution we designed avoids performing computations on the CPU as far as possible, relying on the GPU for almost all curve related computations. The only computations that are potentially performed on the CPU are the original graph layout and the bundling part.

4.1 Introduction to spline rendering

Now, there are two major issues when rendering a parametric spline. Control points define the curve analytically described as a polynomial (see Eq. (1 for Bézier curves). Second, once the polynomial has been determined, it must be evaluated as many times as required in order to interpolate the curve itself. As a consequence, when interacting with the graph asking for local deformation of edges, bringing neighbors closer or following an edge, the curves must be re-computed on the fly.

A classical approach when rendering a curve is to compute the interpolation points on the CPU, then call appropriate graphics primitives and let the GPU render the curve.
Focus+Context in Network Exploration

(a) Bring (step 1) – Selecting a node fades out all graph elements but the node neighborhood. (b) Bring (step 2) – Neighbor nodes are pulled close to the selected node. (c) Go – After selecting a neighbor (the green node in Fig. 4(b)), a short animation brings the focus towards a new neighborhood.

[4.2 GPU-intensive spline rendering]

Our solution delegates the computation of curve points to the GPU which is perfectly well designed to perform vectorial computation and floating points operations. By using the OpenGL Graphics API, we can encapsulate those tasks in a shader program. This type of program, written in a C-like language called GLSL, allows to modify the default behavior of some processing units in the rendering pipeline – the vertex processing unit can be customized this way. The purpose of vertex processing stage is to transform each vertex’s 3D position in virtual space to the 2D coordinates at which it appears on the screen. By designing a vertex shader we can manipulate properties such as node position or color, with all computations executed on the GPU. Shaders offer tangible benefits since they are well suited for parallel processing as most modern GPUs have multiple shader pipelines.

The vertex shader we designed is activated each time we render a curve on screen. Before sending vertex coordinates to the GPU, the curve’s control points are transferred to the shader and stored in an array. The maximum size of that array is hardware dependent and determined at runtime. On recent GPU, more than one thousand control points can be handled efficiently. However, most of the OpenGL implementations have restrained the maximum authorized number of control points to eight. So to draw a B´ezier curve or a cubic B-spline with more than eight control points using evaluators, it has to be done piecewise by subdividing the curve to render into curves with fewer control points. Consequently, the performance to draw high order curves with this technique decreases as the number of control points grows. So even if evaluators work well to render curves with a small number of control points, they are not suitable to resolve our issue of drawing curves with several dozens of control points efficiently.

4.2 GPU-intensive spline rendering

Our solution delegates the computation of curve points to the GPU which is perfectly well designed to perform vectorial computation and floating points operations. By using the OpenGL Graphics API, we can encapsulate those tasks in a shader program. This type of program, written in a C-like language called GLSL, allows to modify the default behavior of some processing units in the rendering pipeline – the vertex processing unit can be customized this way. The purpose of vertex processing stage is to transform each vertex’s 3D position in virtual space to the 2D coordinates at which it appears on the screen. By designing a vertex shader we can manipulate properties such as node position or color, with all computations executed on the GPU. Shaders offer tangible benefits since they are well suited for parallel processing as most modern GPUs have multiple shader pipelines.

The vertex shader we designed is activated each time we render a curve on screen. Before sending vertex coordinates to the GPU, the curve’s control points are transferred to the shader and stored in an array. The maximum size of that array is hardware dependent and determined at runtime. On recent GPU, more than one thousand control points can be handled efficiently. However, most of the OpenGL implementations have restrained the maximum authorized number of control points to eight. So to draw a B´ezier curve or a cubic B-spline with more than eight control points using evaluators, it has to be done piecewise by subdividing the curve to render into curves with fewer control points. Consequently, the performance to draw high order curves with this technique decreases as the number of control points grows. So even if evaluators work well to render curves with a small number of control points, they are not suitable to resolve our issue of drawing curves with several dozens of control points efficiently.