Just-in-Time Annotation of Clusters, Outliers, and Trends in Point-based Data Visualizations

Eser Kandogan*

IBM Center for Advanced Visualization, IBM Research

Figure 1: *Just-in-Time Descriptive Analytics* can automatically identify and annotate visual features (e.g. clusters, outliers, and trends) describing their semantics as users interact with visualizations by performing statistical computation at run-time. Shown above are about 400 cars, shown as dots, representing dimensions such as mpg, weight, horsepower, and origin. Five clusters are detected and annotated automatically with descriptive labels (e.g. "European, 4 cyl.") distinguishing their semantics from others. Within each cluster outliers (encircled in red) are also detected and labeled automatically (e.g. "3 cyl"). Looking at these visualizations users can easily understand structure of the data, i.e. what each cluster represents semantically and how data is distributed across several clusters, and build a qualitative mental model.

ABSTRACT

We introduce the concept of *just-in-time descriptive analytics* as a novel application of computational and statistical techniques performed at interaction-time to help users easily understand the structure of data as seen in visualizations. Fundamental to just-in-time descriptive analytics is (a) identifying visual features, such as clusters, outliers, and trends, user might observe in visualizations automatically, (b) determining the semantics of such features by performing statistical analysis as the user is interacting, and (c) enriching visualizations with annotations that not only describe semantics of visual features but also facilitate interaction to support high-level understanding of data. In this paper, we demonstrate just-in-time descriptive analytics applied to a point-based multi-dimensional visualization technique to identify and describe clusters, outliers, and trends. We argue that it provides a novel user experience of computational techniques working alongside of users allowing them to build faster qualitative mental models of data by demonstrating its application on a few use-cases. Techniques used to facilitate just-in-time descriptive analytics are described in detail along with their runtime performance characteristics. We believe this is just a starting point and much remains to be researched, as we discuss open issues and opportunities in improving accessibility and collaboration.

* eser@us.ibm.com

IEEE Conference on Visual Analytics Science and Technology 2012
October 14 - 19, Seattle, WA, USA
978-1-4673-4753-2/12/$31.00 ©2012 IEEE

Keywords: Just-in-time descriptive analytics, feature identification and characterization, point-based visualizations.

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces; I.5.5 [Pattern Recognition]: Interactive Systems

1 INTRODUCTION

A good visualization reveals structure and patterns in data, and facilitates exploration of relationships between variables. The challenge is that as the data gets more complex (e.g. multiple dimensions, multiple datasets) inevitably representation and interaction becomes more complex. For example, for high-dimensional data, representation may exhibit clutter and interactive exploration may become tedious [1]. To effectively support exploratory activities, techniques should support (1) qualitative understanding of high-level structure of data, (2) development of hypotheses for deep analysis of relationships between variables, and (3) provenance and collaboration on qualitative insight (see also [2][3]). Our focus in this paper is (1).

Our position is that visualization techniques should be more tightly-coupled with computational analytics techniques to deal with increasing complexity, particularly in the exploratory phase of data analysis to help users build qualitative mental models of data (see also [4][5][6][7]). To that end we introduce the concept of *just-in-time descriptive analytics* as a novel application of computational and statistical techniques during interaction time to help them easily understand the structure of data. The goal is not to compete with the human perceptual abilities to detect and identify patterns in visualizations but rather to potentially decrease

the cognitive load on users by automatically explaining structure in real-time as they interact, thus facilitating quick development of qualitative models of data, creating a novel user experience.

Fundamental to interactive just-in-time descriptive analytics is (a) detecting visual features such as cluster, outliers, and trends that users might observe in visualizations automatically, (b) identifying the semantics of such features by performing statistical analysis as the user is interacting with data and creating annotations which uniquely describe visual features, and (c) enriching features in visualizations with annotations that not only describe their semantics (e.g. a cluster labelled distinctly as "European, 4cyl") but also allows the user to interact with them, using for example brushing, to help development of high-level understanding of data.

We argue that interactive just-in-time descriptive analytics allows the user to build faster qualitative mental models of data by demonstrating its application on a few use-cases, such as analyzing telecommunications churn. While we used the Star Coordinates technique (see [8]) to demonstrate the concept of just-in-time descriptive analytics, techniques developed are directly applicable to other point-based visualization techniques, such as 2-d and 3-d scatter-plots and multi-dimensional scaling, and conceptually applicable to other visualizations, such as line charts, though the features and as such algorithms would differ. In this paper we describe techniques used to facilitate just-in-time descriptive analytics for point-based visualizations in detail and present their runtime performance characteristics.

The primary contribution of the paper is the exploration of statistical and computational techniques *at interaction time* to support visual exploration by *explaining semantics* of clusters, outliers, and trends users see on visualizations and thereby *creating a novel user experience* in interaction with data. To that end, our specific contributions are: (a) simple yet fast algorithms for detecting visual features such as cluster, outlier, and trends in point-based visualizations, (b) identifying semantic characterization of such features, (c) ranking of derived feature semantics for presentation, and (d) designing interaction techniques with feature annotations.

2 Related Work

Related works falls into two fields: (1) data mining and (2) information visualization. First, we review computational and statistical techniques in the field of data mining for cluster detection. Next, we review techniques in the field of information visualization, particularly exploratory visualizations that leverage computational techniques to support users in visual data mining activities.

2.1 Cluster Detection in Data Mining

In data mining, clustering is considered as an unsupervised classification problem with the fundamental objective of identifying structure in data, particularly detecting and distinguishing groupings of data elements that are similar to each other and distinct from others based on some criteria.

There are many cluster detection algorithms that are used in different applications (see [9][10] for comprehensive review). These algorithms differ in the way they model the data-space and in the way they choose similarity criteria. K-means is the most common clustering technique and works by iteratively refining k clusters to improve the quality of clustering based on some distance function in high-dimensional data-space [11]. Among all the techniques grid-based approaches are fairly flexible in their ability to find arbitrarily shaped clusters efficiently and consistently across different runs but choice of the grid-size is user-driven and requires some experimentation [12]. Our cluster detection algorithm is also a grid-based approach though it operates on the view-space, i.e. visualization, and as such assumes data is already reduced to 2d through transformations defined by a visualization such as multi-dimensional scaling.

Fundamentally, clustering algorithms do not seek to provide an explanation or interpretation. As such our goal differs in that we seek to derive some semantics to explain the grouping and use such semantics to support interactive discovery process. Another important difference in our approach is that pattern detection is applied in the view-space as opposed to in the data-space. We seek to detect interesting patterns in the visualizations of data often mapped to lower-dimensional space but leverage full data-space to derive semantics for patterns observed in the view-space. Finally, we strive for simplicity in our algorithms, perhaps at the expense of quality, to achieve at least linear algorithmic complexity so that we can perform feature detection and semantics characterization as the user is interacting with data, without penalizing much the overall interactive performance. This is acceptable as our goal is to explain some of the features users might observe in visualizations for further exploration not necessarily to determine the optimal clustering of data. This is important particularly since high-dimensionality of data presents a problem and traditional clustering algorithms often do not perform well particularly in terms of efficiency [13]. Given the high-computational cost and essential requirement of human judgment, there is growing interest in involving people in the cluster detection process. In the next section we will review visual data mining techniques.

2.2 Visual Data Mining and Information Visualization

Visualizations are powerful tools to help users explore and make sense of data, intuitively revealing trends, outliers, and clusters, though they have their limitations particularly for large and complex datasets [14]. Visual data mining aims to address this limitation by involving users directly in the data mining process using interactive visual techniques working together with data mining algorithms, such as decision-tree classifiers [15] [16][17].

Several approaches exist to explore complex high-dimensional datasets, such as multi-dimensional scaling [18], self-organizing maps [19], principal component analysis [20], and [21][22][23] by reducing the complexity, i.e. dimensions of data. A common problem in many of these techniques is that the projections that transform high-dimensional data into lower dimensions are not intuitive. Furthermore, multi-dimensional data has an inherent clutter problem, which may obscure structure present in the data and may make it difficult for users to find patterns and relationships. Though some of these problems can be addressed by selecting a subset of representative dimensions from the original set [24], still, there is an inherent challenge of traversing the high-dimensional data-space to find and examine interesting relationships. For example, scatter plot matrices offer an intuitive way to examine binary relationships in multi-dimensional data yet they significantly suffer from dimensionality curse. To reduce the burden on the user, the grand tour technique uses a series of 2D projections in an animated fashion to traverse all different perspectives of data [25]. Projection Pursuit technique similarly uses statistical techniques to reduce the search space and find the most interesting possible projections [26][27]. While compelling these techniques may not be intuitive to the user and have high computational demands.

To improve the quality of visualizations, metrics can be applied in several stages of the visualization pipeline [28], in data transformation [29][30], visual mapping [31][32], and view transformation [30]. For example, rearranging dimensions by their

similarity may improve the effectiveness of exploration of multi-dimensional data [33]. Likewise, allowing users to rank (pairs of) dimensions to explore by a criterion, such as correlation coefficient, can help incorporate user's interests in the interactive exploratory analysis process [2]. Similarly, scagnostics technique uses graph-theoretic approaches to define measures of shape, trend, and density in 2D scatter plots and creates a scatter plot matrix of these measures that users can use as a pointer to access original scatter plots [34]. In essence many of these metrics model human pattern recognition to some extent [28]. Studies such as [35][36] are critical in bringing validity to these approaches.

In just-in-time descriptive analytics we avoid defining a complete path of exploration, often computed offline, but rather aim to explain what user might find interesting in the current projection, at interaction time, and have the user drive the exploration. Also, we avoid translating user's interest to the metrics space, since it can be potentially challenging for the user to define their interest in statistical terms such as correlation coefficients. Our focus is on explaining what the user might be seeing in the visualizations through annotations, which can help them understand the data quickly. We are not only talking about identifying interesting visual features but also about identifying the semantics of these features, unlike prior research.

A recent technique, click2annotate allows semi-automatic annotation of interesting patterns such as clusters and outliers [37]. In this approach users select a set of points and select a pattern type (e.g. cluster), then a textual summary reporting a set of statistics, such as min/max values, cluster radius, is generated based on a template for the pattern type. In our approach patterns are detected and labeled automatically. Moreover our annotations focus on identifying distinguishing semantics of patterns from others in the visualization, as opposed to reporting the same set of statistics for each pattern. So, annotations in our approach are not merely statistical summaries but rather concise descriptive labels. Unlike click2annotate goal is not to document discoveries (as in [38]) but to explain structure of data to guide exploration. Therefore, our approach needs to compare and rank what is "interesting" based on some heuristics of features and we perform such calculations at interaction-time for good user experience.

3 Just-in-Time Descriptive Analytics

The basic steps of just-in-time descriptive analytics are (1) visual feature detection, (2) feature ranking and annotation, and (3) annotation interaction to help support understanding of the structure of data (Figure 2). The goal is to accomplish all these computations just-in-time as users are interacting with data to provide a good user experience.

Note that we use the term "feature" in a broad sense referring to anything that might be perceptually observed by the user revealing insight about data, such as clusters, outliers, and trends. The goal is not to substitute or compete with the human perceptual abilities but rather to potentially decrease the cognitive load on users by automatically identifying and describing semantics of features they might see in visualizations thus facilitating quick development of qualitative models of data.

We will describe each of these steps in detail in the following sections by applying them on multi-dimensional data using Star Coordinates, to detect and describe features such as clusters, outliers, and trends. Star Coordinates provides an intuitive representation of multi-dimensional data by creating meaningful projections based on linear transformations as in 2- and 3-d scatter plots [8] and as such it is very suitable for our purposes.

Just-in-time descriptive analytics can also be applied to multiple-datasets. This is possible if one can connect datasets through a primary key or use time as the basis for establishing associations. Even in 2-dimensional visualizations it is often the case that features in the visualization are due to intrinsic relationships between attributes not represented in the visualization. For example, a plot of product rating and price can reveal clusters of products whose semantics is explainable by attributes in other datasets such as product type and manufacturer.

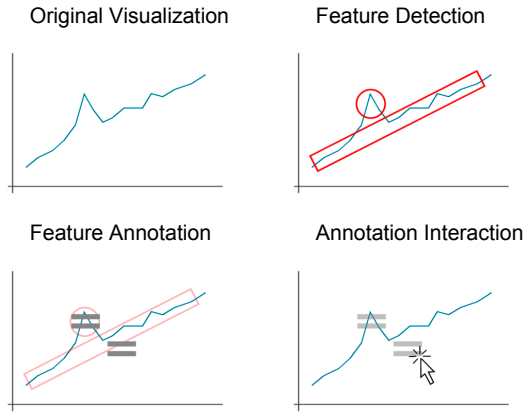


Figure 2: *Just-in-time descriptive analytics*: (a) Original visualization, (b) Feature detection, (c) Feature ranking and annotation, and (d) annotation interaction.

3.1 Cluster Detection

3.1.1 Definition

We consider clusters from a visual perception point of view since our goal for cluster detection is to identify what users might observe as a grouping of data points on a visualization. Perceptual grouping, as defined by Gestalt psychologists, refers to the human visual ability to extract relations using proximity, similarity, continuation, closure, and symmetry of lower-level primitive features and group them to obtain meaningful higher-level structure [39]. As such we define a cluster as a set of points in the view-space, connected to each other by proximity, and collectively perceived as a single arbitrarily-shaped object (as in [4]). Thus, our focus in terms of detection is on the view-space, eventual mapping of data to 2d, not on the high-dimensional data-space.

In essence, we separate concerns of computing a mapping from data-space to view-space (responsibility of the visualization technique) and identifying and describing interesting features in the view-space (just-in-time descriptive analytics) by deriving semantics from data-space. This has obvious performance benefits, which is important for achieving an interaction-time experience, since we are only dealing with a two-dimensional space for the purpose of detection, irrespective of the original dimensions of the data. It is important to note that the goal here is not to detect the clusters with the optimal quality but to identify visible clusters, describe and suggest them to the user for further exploration. The idea is to involve the user in the data mining process by driving their interest into potential hotspots of insight.

Another requirement for our purpose is the ability to detect arbitrary shapes as they may mean a particular kind of relationship between dimensions of data. For example, an elongated shape on a visualization might suggest a linear relationship or trend between two dimensions. Once such shapes are detected just-in-time descriptive analtyics techniques can be utilized to explain that particular shape (see Trend Detection).

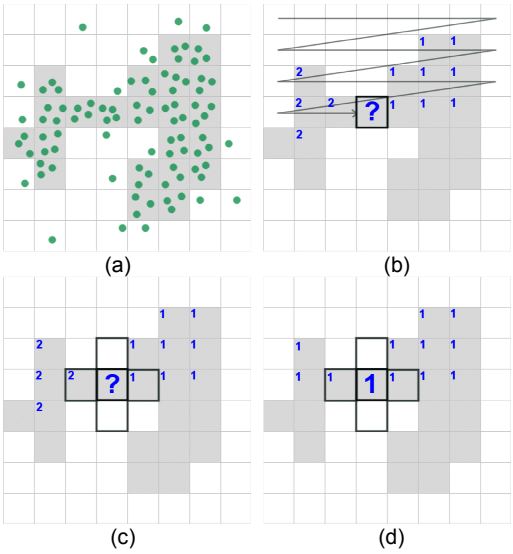


Figure 3: Steps of a 4-neighbor grid-based cluster detection algorithm: (a) assignment of points to grid cells, (b) traversal of grids in a row-major fashion, (c) assignment of a cluster id based on 4-neighboring grid cells, and (d) merging of connected clusters.

3.1.2 Approach

We took a grid-density based approach to detect clusters in the view-space, since they are able to detect arbitrarily shaped clusters, connecting contiguous high-density grid cells. Grid-based approaches also scale well thus they suit our interaction-time performance requirements.

We assume a set of k-dimensional data records, $D = \{d_1, d_2, \ldots, d_i, \ldots, d_n\}$, with a cardinality of n, already mapped to view-space through some sort of transformation (τ) by a visualization technique, to obtain a set of points, $P = \{ p_1, p_2, \ldots, p_i, \ldots, p_n\}$, where $p_i = \tau(d_i)$ and $p_i \in R^2$. The detection algorithm starts by assigning each point p_i to a grid cell $G_{(r,c)}$, based on the coordinates of the p_i, i.e. p_i^x and p_i^y, and grid width. We then count the number of points in each grid cell, $|G_{(r,c)}|$, and calculate average for all occupied cells, μ. A grid cell is considered an eligible cluster cell if $|G_{(r,c)}| > \mu \cdot f$, where f is a fixed factor, determined by the user. The algorithm then proceeds by traversing each eligible cluster grid cell in a row-major order, left to right and top to bottom, making cluster assignments to grid cells. The assignment of cluster ids is done considering either 4-neighbors (only sharing an edge) or 8-neighbors (sharing edge or corner) of a given cell. If any of these neighbors are already assigned to a cluster, then the current eligible cell and other cells among the eligible neighbors are assigned to that cluster, merging the two clusters. Otherwise, a new cluster is created and current cell is assigned to that cluster (Figure 3). Computational complexity of the algorithm is O(n). Since the algorithm works in the view-space complexity is independent of the original dimensions.

3.1.3 Limitations and Improvements

While our basic grid-density based algorithm works for most practical purposes there are several issues with it. It is important to note that the goal is not necessarily to identify all clusters with their optimum boundaries but enough of the clusters and annotate them with their semantics to help the user.

One issue is the choice of the grid cell size though unlike data-space clustering algorithms the issue may not be as severe. Because the algorithm works on the view-space, dimensions of the visualization are often scaled either automatically or by the user

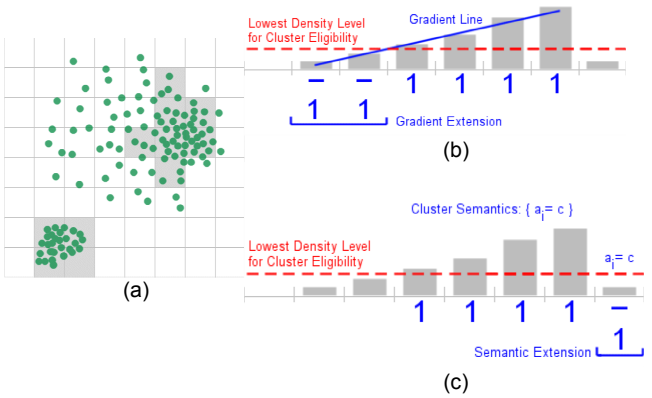


Figure 4: (a) Using the same density level to detect clusters in the whole view-space may result in suboptimal clusters for clusters of varying densities, which can be addressed using (b) gradient- or (c) semantics-based extension of clusters.

such that data fits the view and is readable. A more important issue is the use of a global average cell density, μ, to determine eligibility for cluster detection as it is possible that different clusters might have different densities (Figure 4.a). Another issue is the potential jitter as the user is interacting with the visualization changing the projection parameters. When users interactively change the projections the cluster centroids and cluster assignments might change due to changing grid assignments and cause undesired jitter when annotations are displayed at their centroids. The proposed algorithm also doesn't handle nested clusters, where large clusters may contain denser clusters within.

There are several ways to improve on the basic algorithm to address these issues without sacrificing performance much. The idea is to extend existing clusters in a post-processing step so that they are more inclusive of nearby cells if they are close in terms of change in density or in terms of semantics of the data points. Techniques such as hierarchical grids and nested means may also address these issues though with higher performance costs [3][40].

Density change can be incorporated by calculating the gradient along x- and y-dimension for each cell. Gradients can be calculated by calculating the change in the number of data points from its neighbor cells horizontally and vertically, i.e. $\nabla^y_{(r,c)} = |G_{(r,c)}| - |G_{(r-1,c)}|$ and $\nabla^x_{(r,c)} = |G_{(r,c)}| - |G_{(r,c-1)}|$. A grid cell would now be considered eligible if its gradients along either dimension are within a defined limit (Figure 4.b). Essentially the algorithm would extend clusters by smoothing out fuzzy boundaries based on grid cell densities. Semantics can be incorporated likewise in a post-processing step once the characteristics of each cluster are calculated. In this case cells with similar characteristics (e.g. cars with 4 cylinders) in neighboring cells are merged to the existing cluster (Figure 4.c).

User involvement in this process is equally applicable to resolve these issues. Appropriate user interaction techniques can be provided to allow users to expand or contract clusters or define completely new clusters to support supervised interactive cluster detection and verification.

3.2 Outlier Detection

3.2.1 Definition

As with clusters, we also take a perceptual perspective in thinking about outliers. There are several types of outlier points in a visualization: (a) *false-positives*, points that lie within a visual feature such as a cluster but semantically they don't belong there, (b) *close* points that are near a visual feature but perhaps differ in

just a few dimensions that pushed them out of the feature visually, and finally, and (c) *far* points that are distant from any visual feature but they are themselves not significantly dense enough to be identified as a cluster (Figure 5.)

3.2.2 Approach

We basically identify false-positives by going through all the data points in each cluster separately and comparing their values in each dimension to average values within the current cluster. If value is either below or above the mean by a factor (default 3, but could be adjusted by user) of standard deviation within cluster, i.e. $| d_i^k - \mu^k| > 3\ \sigma^k$, we mark such data points (d_i) as outliers and record such dimensions (k) along with the value (d_i^k) to inform the user. We save the number of outliers and total number of points in a cluster as indicators of the strength of the outliers to determine whether or not to show a dimension with outliers to the user. Alternatively, the difference between mean and actual value of outliers could also be taken into account for determining strength of outliers. The algorithmic complexity is O(kn) where n is the number of data points and k is the number of dimensions.

In our current implementation we only detect false-positives however detection of close outliers can be performed similarly. All data points in nearby grid cells of a cluster can go through a similar process of checking their values against cluster means. Note that only data points that are not members of the cluster as a result of semantic extension should be processed. As for far outliers one can think processing cells that don't belong to any cluster and checking their values against overall means and standard deviations. It is also possible to modify this process by adding a minimum distance to a cluster boundary requirement.

3.2.3 Limitations

Our outlier detection algorithms focus on a single dimension value. It is possible that only combinations of dimension values may result in outliers. In this paper we didn't focus on such outliers. Another limitation of the above algorithm is that for dimensions with categorical values it assumes values are mapped to a numeric value, and as such detection is at best suboptimal.

A final remark is that outliers are not only useful for reporting such cases to the user but also for improving cluster semantics as such data points are removed in a pre-processing step in cluster refinement to improve the precision of the semantics derived.

3.3 Trend Detection

3.3.1 Definition

We define trends as visual depictions of the form, direction, and strength of changes in data values. There are several types of trends, for example, the shape of a cluster can have an elongated shape suggesting that there is linear relationship between two or more dimensions, or it might reflect a dominant dimension exhibiting a directional trend within the cluster. Shape can be circular suggesting that perhaps there is a radial trend (Figure 6).

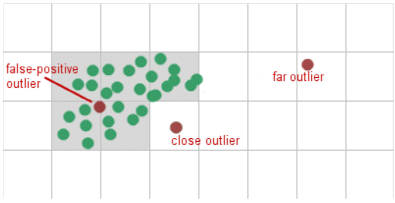


Figure 5: Types of outliers: (a) false-positive outliers are visually part of a feature they shouldn't be, (b) close outliers are similar to a close-by feature but they aren't part of, and (c) far outliers are far off from any other feature in the view space.

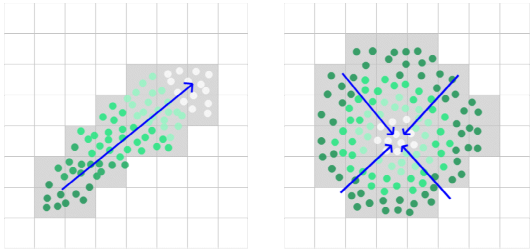


Figure 6: A few trend types: (a) Directional trend corresponding to dimension values increasing along the direction of the elongated cluster (b) Concentric trend corresponding to a dimension values increasing radially from the center of a cluster.

What is common in essence about these several types of trends is that their shapes reveal the characteristics of the distribution of one more dimensions of the data elements in a cluster. The high-level approach to detecting trends requires: (1) recognition of the shape of a cluster, (2) association of the shape to a trend-type, (3) identifying dimensions exhibiting that trend, and (4) determining the strength of trend.

3.3.2 Approach

In our current implementation, we only detect directional trends thus we only focus on steps 3 and 4. The algorithm starts off by calculating the average data values ($\mu^k_{row=i}$, $\mu^k_{col=j}$, for all (i,j), where $G_{(i,j)} \in c_l$) for each dimension (k) for each row (i) and column (j) in a cluster (c_l) (Figure 7.) Then, we perform simple linear regression to fit a line to the average values vertically ($\widehat{\mu^k_{row=i}} = \alpha^k_{row} + \beta^k_{row} \cdot r_i$, where $r_{min} \le r_i \le r_{max}$) and horizontally ($\widehat{\mu^k_{col=j}} = \alpha^k_{col} + \beta^k_{col} \cdot c_j$, where $c_{min} \le c_i \le c_{max}$)) to determine if the average values increase or decrease consistently row after row, or column after column, within a cluster. Using the actual average values ($\mu^k_{row=i}$, $\mu^k_{col=j}$) and estimated values ($\widehat{\mu^k_{row=i}}$, $\widehat{\mu^k_{col=j}}$) we calculate the standard errors for row and column average values (σ^k_{row}, σ^k_{col}) compared to the fitted line. If standard error is less than a some fraction of error ($\sigma^k_{row} < \varepsilon$) and magnitude of slope is higher than a threshold (β^k_{row} and/or $\beta^k_{col} > \Theta$) we claim that there is a directional trend. We save the standard error and slope as indicators of strength and direction of the trend to be used later to determine if trend is worthy of presenting to the user, in final analysis. The algorithmic complexity is O(kn) since we calculate statistics on each row/column in a cluster in each dimension.

3.4 Feature Ranking and Annotation

3.4.1 Definition

Annotation of features is about deriving the semantics that describe the data points in a feature, e.g. cluster. When deriving such semantics for a cluster it is important to focus on its distinguishing semantics compared to other clusters. The goal is not to provide a complete statistical summary for each feature detected but rather to highlight the unique semantics of features and use that to explain high-level structure through annotations on visualizations. Such automatic annotation also provides an opportunity for cluster validation by the user.

3.4.2 Approach

To derive such semantics we use a number of metrics, including density of data values in a cluster, overlap with other clusters, number of outliers in a cluster, and strength of trends that might exist in a cluster.

Cluster density is derived from the standard deviation of data values for each dimension. We essentially calculate a density

score ($density_i^j$) between 0 and 1, based on a step function of the standard deviation with steps at 0.01 (very dense), 0.05 (moderately dense), 0.1 (somewhat dense), and 0.25 (sparse), corresponding to scores 1, 0.8, 0.5, and 0.1, respectively.

Overlap score is determined by calculating the degree of overlap in data value ranges across each cluster. Instead of using the actual data values, e.g. min or max, to determine overlap, we calculate a range based on the standard deviation and means of data values as ($\mu^j - \sigma^j$, $\mu^j + \sigma^j$) to arrive at a more consistent range not deviated by extreme minimum and maximum values in clusters. For each cluster we calculate average degree of overlap with every other cluster to derive an overlap score, $overlap_i^j$, between 0 and 1, complete overlap and no overlap, respectively. Thus, the higher the degree of overlap the less interesting becomes the dimension for uniquely describing the cluster.

Outlier score is derived from the outlier ratio for each cluster by dividing the number of outliers to the total number of data points in each cluster. Again we employed a step function of outlier ratio to arrive at a score ($outlier_i^j$) between 0 and 1, with steps at 5%, 1%, 0.5%, and so on, corresponding to scores 1, .8, .6, and so on.

Trend score is calculated from the standard error of the line fit, as discussed in previous section. If the magnitude of the slope of the line is above a threshold it is considered to exhibit a trend with a trend score ($trend_i^j$) between 0 and 1, based on step function of the standard error. This score is calculated both for row and column trends.

The high-level algorithm basically goes through every cluster (c_i) in the visualization and for each dimension in the data (j) we calculate density, overlap, outlier, and trend scores, as discussed above. Then, we calculate a weighted sum of scores for each dimension ($score_i^j = w_{density} \cdot density_i^j + w_{overlap} \cdot overlap_i^j + w_{outlier} \cdot outlier_i^j + w_{rowtrend} \cdot rowtrend_i^j + w_{coltrend} \cdot coltrend_i^j$) and for each cluster in the visualization. In our current implementation density score is given the highest weight, with decreasing weights given to overlap, trend, and outlier scores, in that order. Adding scores for a dimension from all clusters we calculate total scores for each dimension and divide that by the number of clusters to arrive at an average score for each dimension (μ_{score}^j). Finally, we sort these average scores by dimension and determine a set of important dimensions (ID $\subseteq \{1,2,..,n\}$) based average scores being above a threshold. Thus, we arrive at a set of characteristics, such as whether a cluster is dense, has trends and outliers, its min, max, and mean values, for each cluster for each important dimension. We use these characteristics to render annotations (e.g. cylinders=4 for c_i, row trend on weight [1200..1600] for c_k) overlaid on the visualization to help users explore the data. The algorithmic complexity depends on the number of clusters (m) detected and on the number of dimensions (k), i.e. O(mk).

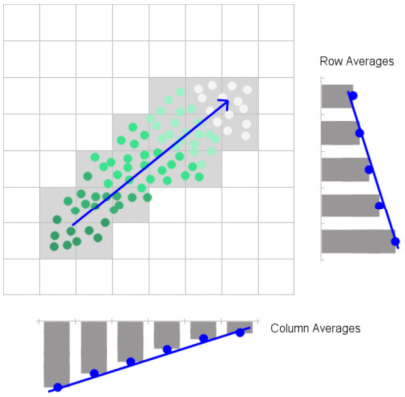


Figure 7: Directional trend algorithm examines vertical and horizontal average values for each dimension in each row and column of a cluster, and tries to fit them to a line. If the standard error between actual and estimated values is low and slope is above a threshold it is considered that there is a directional trend.

3.5 Annotation Interaction

3.5.1 Definition

Annotations describe the semantic characteristics of the clusters, trends, and outliers. They are overlaid on visual feature in a visualization to explain such features and facilitate interaction. There are basically four different types of annotations corresponding to different types of features they represent: (a) clusters with a nominal value (e.g. "6 cyl"), (b) clusters with a range value (e.g. "[2000..3000] pounds"), (c) outliers in a cluster, and (d) trends (e.g. "4 » 6 » 8 cyl") (Figure 8.) A cluster can have multiple such characteristics, e.g. a cluster of American cars, with 6 cylinders, weight ranging from 2000 to 3000 pounds, and as such multiple annotations.

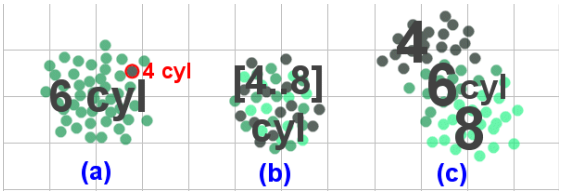


Figure 8: Annotations for (a) a cluster with a nominal value: 6 cyl. cars with an 4 cyl. outlier, (b) a cluster with a range value: 4 to 8 cyl. cars, and (c) a cluster with a directional trend where cars are distributed linearly from 4 cyl. to 6 and 8 cyl. within the cluster.

3.5.2 Approach

Given the characteristics of the feature, identified in the previous step, we use templates to produce the text of the annotations. Given a value or a range of values, templates define how to construct a short text phrase that can be displayed. Templates may include units (e.g. "cu in"), attribute name (e.g. "engine size"), value (singular and range) transformations (e.g. 1: "American"), and how these should be combined to produce a phrase. Each template specifies how a single value and a range of values should be rendered. There are default templates for short and long phrases, with reasonably good defaults, but having users specify their templates can further improve readability and recognition. Templates are specified as part of the schema definition. Templates are specified in JavaScript, thus a range of expressions are possible:

Below are a couple of template specifications, illustrating:

- data transformation, converting a two-digit year to a complete year representation: format:{single: "'19'+value", range: "'19'+min+'..'+'19'+max"}
- units, format:{unit: "cu in"}

Once texts of annotations are produced they are overlaid on the features they are associated with. For clusters, annotations are rendered at the cluster centroid. For outliers, annotations are rendered next to the data point. For trends, annotations are rendered centered over every other row or column within the cluster, thus there are often several annotations rendered along the shape of a cluster. The font size for the features are determined by the size of the features detected, we essentially used a fixed ratio to the radius of the feature size.

In our current implementation, since there can be multiple annotations for each feature, we automatically cycle through each

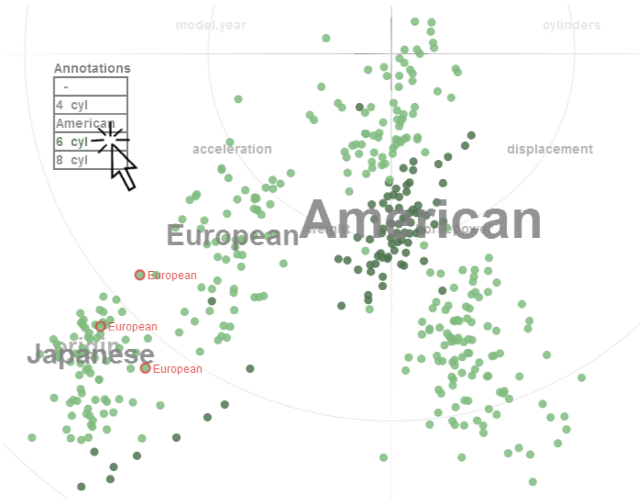


Figure 9: Users can make a list of annotations and using brushing technique can highlight annotated data points in another projection, or they can also revert back to the projection at the time the annotation was captured.

annotation at a frequency. When doing so we synchronize rendering of annotations such that at each cycle all clusters show annotations of the same dimension to support easy comparison across clusters. We also opted to display two annotations simultaneously, corresponding to different dimensions (e.g. "American, 6 cyl") so that associations between characteristics can be made easily (Figure 1). We took care of not displaying a trend annotation with other annotations at the same time as that may cause overlaps. When user is not interacting with the visualization we automatically cycle through each important dimension identified. When the user moves the mouse or interacts in another way we freeze the cycling for a brief period to allow the user to select an annotation if they desire so.

Currently, we support a few ways of interacting with annotations (Figure 9.) Users can brush over the annotations to highlight data points with the same semantics (e.g. "6 cyl"). This helps users in understanding and verifying clusters and how data points are distributed to each cluster. Users can also click to save an annotation and create a collection of annotations to use them later in their exploration. For example, they can brush over annotations in their collection and explore where the data points with those semantics lie at a later point in their exploration. Clicking over an annotation in the collection would bring back the projection to the point in time when that annotation was captured, supporting easier recall. While it is not yet implemented it would be possible to combine multiple annotations and support brushing so that hierarchical structures could be more easily explored.

3.5.3 Limitations

Currently, we don't support users creating or expanding automatically detected clusters. Likewise allowing users to annotate features with their own tags would also be useful for capturing insight. These would involve users in the data-mining process even more, and support supervised cluster detection.

We are also exploring several different design options for rendering annotations (such as small histograms) and features (such as enclosing borders to present clusters). While current annotations are easy to read, visual representation can further improve readability.

4 Evaluation

Below we present a quantitative and qualitative evaluation of just-in-time descriptive analytics techniques discussed. First, we examine runtime performance and feature detection quality. Then, we present our analysis of two different datasets.

4.1 Performance

4.1.1 Setup and Datasets

We performed runtime analysis on a Lenovo ThinkPad T410, with 6GB RAM, dual core 2.66Ghz Intel i7-620M, on 64bit Windows 7 Professional. Tests were performed on a web application using HTML5 Canvas and Dojo running on Firefox browser (ver. 10). Measurements were taken using console.time.

The datasets used in the analysis were based on the original car dataset with about 400 9-dimensional data points (See Analysis: Cars Data). Larger sizes of the same dataset was created by adding small noise (randomizing +/-) to original values, where each data point would appear multiple times with similar characteristics, so that we would preserve the high-level structure and patterns in the data but with larger number of data points. The point here is to measure how well it scales with higher number of data points while preserving the same structure of data so that comparisons across different number of data points are fair. Higher-dimensional datasets were created similarly by adding new dimensions based on original dimensional values, again randomized to some degree. We measured performance in 150 or so runs of the algorithm in a sequence of projections such that we observed a mixed set of clusters, outliers, and trends and reported average running times.

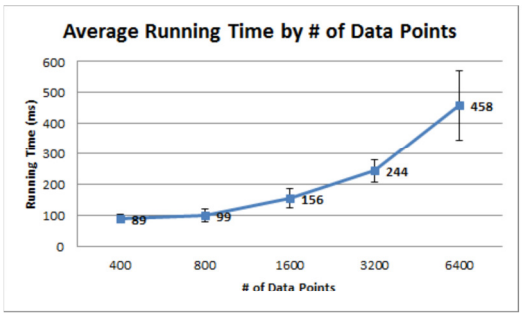
4.1.2 Runtime Performance

Our results show near linear runtime performance as we tested the techniques on datasets of increasing number of data points, confirming our algorithmic complexity analysis findings (Figure 10.a) At about 3200 data points we noticed slightly noticeable lag in interactive performance as total runtime was nearing 300ms, suggesting above that number of data points interaction quality would begin to suffer. Increasing the number of dimensions resulted in a similar near-linear performance curve (Figure 10.b)

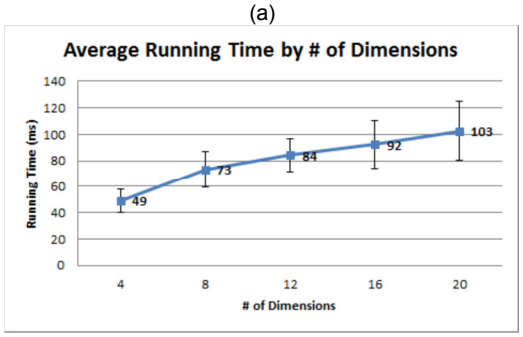
We also conducted another detailed run of the tests to examine the breakdown by the various parts of the technique (Figure 10.c) Visual cluster detection has about constant runtime performance as it leverages grid-based approach, it increased only slightly with the number of data points. Feature ranking also has about constant runtime performance as it depends on the number detected features and the number of dimensions, which in this case was the same in each dataset independent of size. Outlier detection is heavily dependent on the number of data points because it relies on statistic calculation on each data point so was cluster refinement, which involved outlier removal and recalculation of cluster statistics. While trend detection was faster, its performance was also linear in terms of the number of data points since it also involves going through each row/column and calculating statistics.

4.1.3 Quality Evaluation

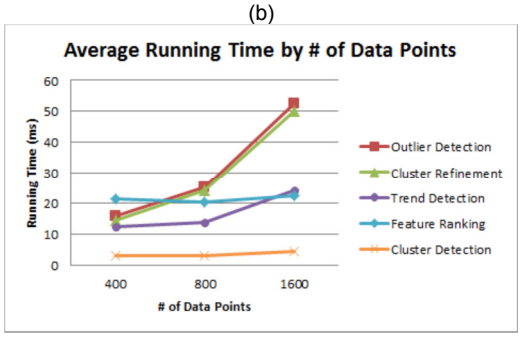
Quality of clusters is perceptual of nature, depends on task, and requires human-judgment and as such it is difficult to evaluate the quality of clustering techniques. In just-in-time descriptive analytics the goal is not to identify all clusters in absolute terms but to use potential features to guide further exploration. Nonetheless we evaluated the quality of our cluster detection and classification by perfoming simple statistics. Essentially for each classification for a cluster we calculated precision and recall, i.e. what percentage of data items within the cluster are appropriately



(a)



(b)



(c)

Figure 10: Performance: (a) Average total running times by number of data points on 9-dimensional data (b) Average total running times by number of dimensions on 400 data points, and (c) Breakdown of running times by analytics algorithm and by number of data points.

labeled with that classification (precision), and what percentage of data items total with that classification are within that cluster (recall). As in performance analysis, we measured precision and recall in 1000 or so runs of the algorithm in a sequence of projections and report averages scores.

It is important to note that this evaluation is very dependent on the mapping of the particular visualization technique and may not reflect the quality of our cluster detection and classification algorithm fairly. Just-in-time descriptive analytics can only be as good as dimensionality reduction/mapping to view-space of the visualization technique. Nonetheless we found fairly good average precision (.994) and reasonably good recall (.716) with an F1 score of .828 for categorical dimensions. For numerical dimensions these scores are .993, .795, and .880, respectively. It is not surprising that precision is so high; after all, the classification is determined based on these values. Recall is lower but as suggested earlier using density gradient and semantic expansion and potentially other techniques recall will likely increase.

4.2 Analysis: Cars Data

We analyzed the cars dataset, which contained 406 car specs on the following 9 variables: model name, model year, miles per gallon (mpg), number of cylinders, engine displacement (cu. inches), horsepower (hp), weight (lbs.), time to accelerate from 0 to 60 mph (sec.), and origin of car (1. American, 2. European, 3. Japanese). Dataset has missing data in several columns [41].

We started our analysis by turning off the name and model year variables to explore cars purely from their technical specs. As a result, we obtained a projection that clearly showed three clusters. Clusters were automatically labeled as (A) "American..Japanese, 3..5 cyl, 68..183 cu in"; (B) "American, 6 cyl, 156..262 cu in"; and (C) "American, 8 cyl, 302..400 cu in". The labeling appropriately showed the main distinguishing characteristics of each cluster, e.g. cluster A containing cars 3 to 5 cylinders, small engine-size, from each origin. Cluster A also contained several 6 cylinder cars identified as outliers and one 200 cu in. engine car as an outlier. Upon examination of detailed values that car turned out to have a missing value for horsepower. The other outlier cars however were interesting in that they have very similar engine specs such as weight, etc. to the cluster they were presented in except the 6 cylinders. None of the other clusters had outliers.

To examine the largest cluster (cluster A) even more, we scaled origin variable and as a result we split this cluster into three sub-clusters, one for each origin (Figure 1): (A1) "Japanese, 4 cyl."; (A2) "European, 4 cyl."; and (A3) "American 4 cyl." Cluster A1 had one European car as an outlier, 1971 buick-century, and had specs similar to Japanese cars. There were also a couple of 3 cyl. cars, identified as outliers. Cluster A2 had two 5 cyl. cars, and cluster A3 had one 6 cyl. car as outlier.

This was a rather quick analysis aimed at gaining a high-level understanding of the structure of the data. Our analysis quickly revealed the hierarchical clusters around origin and cylinders, which correlated significantly with other dimensions of the car, such as weight and displacement such that even turning off the cylinders the clusters were described by their number of cylinders.

4.3 Analysis: Telecommunications Churn Data

We also analyzed a telecommunications churn dataset that consists of 5000 customer records including information about 21 attributes such as state, area code, phone number, voice plan membership, number of voice messages, number of customer service calls, international plan membership, day, evening, night, and international minutes, calls, and charges, and churn indicating whether the customer left for a competitor or not [41].

We started our analysis by coloring churn and turning it off subsequently to examine how churned customers are distributed to different clusters. We also turned off columns such as state, area code, and phone number, we thought were not related to churn. As a result two main clusters emerged, labeled as (A) "no voice plan, 0..12 voice msgs", and (B) "voice plan, 8..50 voice msgs". To make clusters more definitive we turned off potentially correlated dimensions such as minutes and calls, but kept charges for different periods of time (i.e. day, evening, and night). We also turned off number of voice messages and slowly more clusters emerged, we also scaled voice mail plan and international plan to separate the clusters more clearly (Figure 11.) As a result 4 clusters emerged representing all combinations of international and voice plan membership, e.g. {intl. plan | no intl. plan} x {voice plan | no voice plan. These clusters were also labeled with number of voice messages, not shown in figure, as we only show two dimensions at a time. A quick examination of the distribution of churned data points, colored green, revealed that (1) customers

with international plan churned much more than those without; and (2) customers without an international plan but with voice mail plan are less likely to churn compared to those that do not have voice mail plan, as indicated by the number of green dots.

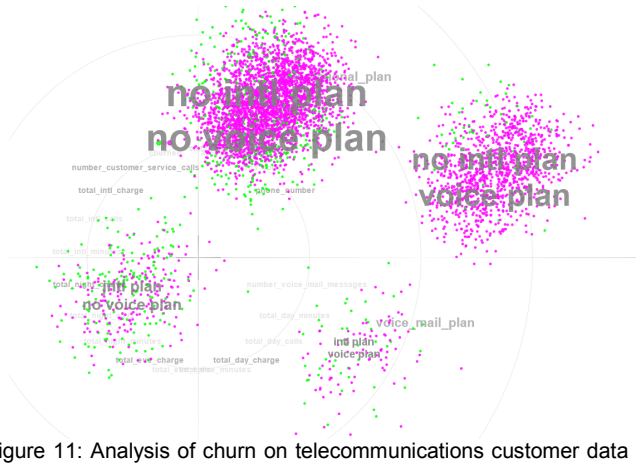


Figure 11: Analysis of churn on telecommunications customer data by examining distribution of (churned/green) customers in four clusters (by international plan and voice plan) shows that customers with international plan churn significantly more than others.

5 Discussion

Based on our evaluation we see that just-in-time descriptive analytics can provide quick insight into datasets, particularly those that have a mix of categorical and numerical dimensions, such as the two datasets we presented here. This is not surprising since in our default ranking we gave density score the highest weight. Categorical dimensions have the advantage that their values (e.g. American vs. Japanese) are meaningful to the user, as opposed to numerical dimensions. To apply techniques developed here more effectively to numerical dimensions we think that applying binning and qualitative labeling of the bins, such cars with weights above 2500 as "heavy", would prove very useful.

Furthermore, both of these datasets were dense datasets where data was strongly clustered. We believe to apply just-in-time descriptive analytics techniques effectively to more sparse datasets we need to improve on cluster detection and annotation. Particularly when there is a mix of dense and sparse clusters, fuzzy and overlapping boundaries, and potentially nested clusters we need to enhance our grid-based clustering algorithm. As for outliers, it would certainly be important to identify close and far outliers as they would certainly be informative as well. There are also several patterns that we are not addressing such as trends corresponding to contours of varying data values, typical in geographic datasets.

In terms of user experience, current implementation suffers from lack of coherent cluster assignment across interactions as such we experience some jitter. There are several consequences of this. One is that the annotation position shifts as the centroid of the cluster changes due to changing cluster membership. Second, cluster characterization might be affected by this and as such the annotation labels/values might change. Lastly, this might even cause change in the relative ranking of the dimensions and as such change the presentation order. Currently, we are addressing this problem by sorting important dimensions, above the threshold for presentation, alphabetically so as users interact with the dataset incrementally, such as scaling a dimension, annotation labels don't change abruptly.

Also important for user experience, we believe that for different datasets users should be able to adjust feature ranking weights to yield more or less of the desired features. For example, conditions for identifying points as outliers might be more or less stringent depending on the dataset and task. We also believe that involving users more in the identification and characterization of features would be help user experience significantly.

Our performance evaluations based on a client-side implementation show reasonable interactive performance until couple of thousand data points. To scale beyond this we would need to implement a client-server architecture where the server would mimic the visual mappings based on specific projection parameters it receives from the client. The server would then detect, rank, and identify features and annotations and would send the calculated annotations along with their position and size back to the client. This would be very reasonable implementation for scaling up the techniques described here, as the message sizes between client and server would be minimal.

Lastly, we need to evaluate whether annotations generated are useful, or whether they would mislead the user perception by conducting user studies. While we doubt that they will mislead the user as the annotations are precise in terms of their descriptions they may take attention away from other features that users might look at. User studies, particularly longitudinal studies, will need to be conducted to assess the validity of the proposed approach.

6 Conclusion

In this paper we explored just-in-time descriptive analytics techniques to help support understanding high-level structure of data. To that end we developed simple algorithms that would detect and describe clusters, outliers, and trends at interaction time as users are exploring the data. We believe that it provides a novel user experience of computational techniques working alongside of users allowing them to build faster qualitative mental models of data and we demonstrated its application on a few use-cases. These algorithms were specific to point-based visualizations however we think concepts developed in this paper are applicable to other visualizations.

Beyond supporting qualitative understanding of high-level structure of data, techniques developed can also be applied in improving accessibility of visualizations as fundamentally we are proposing not only a way to describe features in natural language but also interact with such features. Annotations display accessible textual descriptions of purely visual attributes for parts of a visualization and even support some interaction with them. We also think there is potential application of these techniques in improving collaboration among analysts, by capturing and sharing automatically identified annotations along with the specifics of the projections that resulted in particular views. Just-in-time descriptive analytics can also be useful in identifying and suggesting visual and data operations, for example view transformations such as scaling, rotation that users could do to gain new insight and as such improve quality and quantity of observations. This would be possible as just-in-time descriptive analytics computes how many features (e.g. # of clusters, trends) and what semantics (e.g. a cluster of European cars) are associated with each feature, at least as far as detectable features are concerned, and how that would change as a result of an operation.

We believe this is just the starting point of tighter integration of computational and visual techniques at interaction time. Much remains to be researched in terms of new visual features such as temporal trends, and algorithms to detect and describe them at interaction time.

REFERENCES

[1] Peng, W., Ward, M. O., Rundensteiner, E. A. Clutter Reduction in Multi-dimensional Data Visualization using Dimension Reordering. Proc. of the IEEE Information Visualization, pp. 89–96, 2004.

[2] Seo, J., Shneiderman, B. A Rank-by-feature Framework for Unsupervised Multidimensional Data Exploration Using Low Dimensional Projections. Proc. of the IEEE Information Visualization, pp. 65–72, 2004.

[3] Guo, D., Gahegan M., Peuquet D., MacEachren, A. Breaking Down Dimensionality: An Effective Feature Selection Method for High Dimensional Clustering. Workshop on Clustering High Dimensional Data and its Applications, Proc. 3rd SIAM International Conference on Data Mining, 2003.

[4] Guo, D. Coordinating Computational and Visual Approaches for Interactive Feature Selection and Multivariate Clustering. Information Visualization 2, pp. 232–246, 2003.

[5] Jong, H., Rip, A. The Computer Revolution in Science: Steps Toward the Realization of Computer-supported Discovery Environments. Artificial Intelligence 91, pp. 225–256, 1997.

[6] Wong, P.C. Visual Data Mining. IEEE Computer Graphics & Applications19, pp. 20–31, 1999.

[7] Ankerst, M., Ester, M., Kriegel, H.-P. Towards an Effective Cooperation of the User and the Computer for Classification. Proc. 6th International Conf. on Knowledge Discovery and Data Mining (KDD '00), pp.179–188, 2000.

[8] Kandogan, E. Visualizing Multi-dimensional Clusters, Trends, and Outliers using Star Coordinates. Proc. of the seventh International Conference on Knowledge Discovery and Data Mining (KDD '01), pp.107–116, 2001.

[9] Jain, A., Murty, M. N., Flynn, P. J. Data Clustering: A Review. ACM Computing Surveys 31(3), pp. 264–323, 1999.

[10] Han, J., Kamber, M., Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2001.

[11] MacQueen, J. B. Some Methods for Classification and Analysis of Multivariate Observations. Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability, pp.281–297, 1967.

[12] Sheikholeslami, G., Chatterjee, S., Zhang, A., Wavecluster: A Multi-resolution Clustering Approach for Very Large Spatial Databases. Proc. of Very Large Databases Conference, pp.428–439, 1998.

[13] Abul, A. L., Alhajj, R., Polat, F., Barker K. Cluster Validity Analysis Using Subsampling. Proc. of IEEE International Conference on Systems, Man, and Cybernetics 2, pp. 1435–1440, 2003.

[14] Shneiderman, B. Inventing Discovery Tools: Combining Information Visualization with Data Mining. Information Visualization 1(1), pp. 5–12, 2002

[15] Keim, D. A. 2002. Information Visualization and Visual Data Mining. IEEE Transactions on Visualization and Computer Graphics 8(1), pp. 1–8, January 2002.

[16] Ankerst, M., Elsen, C., Ester, M., Kriegel, H-P. Visual Classification: An Interactive Approach to Decision Tree Construction. Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining (KDD '99), pp. 392–396, 1999.

[17] Teoh, S. T., Ma, K-L. Starclass: Interactive Visual Classification using Star Coordinates. Proc. of 3rd SIAM International Conference on Data Mining, pp. 178–185, 2003.

[18] Borg, I., Groenen, P. Modern Multidimensional Scaling: Theory and Applications (2nd ed.). New York: Springer-Verlag, 2005.

[19] Kohonen, T. Self-Organizing Maps (2nd ed.). Berlin: Springer, 1997.

[20] Jolliffe, T. I. Principal Component Analysis. Springer Press, 2002.

[21] Paulovich, F. V., Nonato, L. G., Minghim, R., Levkowitz, H. Least Square Projection: A Fast High-precision Multidimensional Projection Technique and its Application to Document Mapping. IEEE Trans. Vis. Comput. Graph. 14(3), pp. 564–575, 2008.

[22] Joia, P., Paulovich, F., Coimbra, D., Cuminato, J. A., Nonato, L.G. Local Affine Multidimensional Projection. IEEE Transactions on Visualization and Computer Graphics 17, pp. 2563–2571, 2011.

[23] Ingram, S., Munzner, T., Glimmer, O. M. Multilevel MDS on the GPU. IEEE Transactions on Visualization and Computer Graphics 15, pp. 249–261, 2009.

[24] Yang, J., Ward, M. O., Rundensteiner, E. A., Huang, S. Visual Hierarchical Dimension Reduction for Exploration of High Dimensional Datasets. Proc. of the Joint Eurographics - IEEE TCVG Symposium on Visualization, pp. 19–28, 2003.

[25] Asimov, D. The Grand Tour: A Tool for Viewing Multidimensional Data. SIAM Journal of Scientific and Statistical Computing 6(1), pp. 128–143, 1985.

[26] Friedman, J., Tukey, J. W. A Projection Pursuit Algorithm for Exploratory Data Analysis. IEEE Transactions on Computers 23, pp. 881–890, 1974.

[27] Cook, D., Buja, A., Cabrera, J., Hurley, C. Grand Tour and Projection Pursuit. Journal of Computational and Graphical Statistics 23, pp.155–172, 1995.

[28] Bertini, E., Tatu, A., Keim, D. Quality Metrics in High-Dimensional Data Visualization: An Overview and Systematization. IEEE Transactions on Visualization and Computer Graphics 17(12), pp. 2203–2212, 2011.

[29] Johansson , S., Johansson, J. Interactive Dimensionality Reduction Through User-defined Combinations of Quality Metrics. IEEE Trans. On Visualization and Computer Graphics 15(6), pp. 993–1000, 2009.

[30] Yang, J., Peng, W., Ward, M. O., Rundensteiner, E. A. Interactive Hierarchical Dimension Ordering, Spacing and Filtering for Exploration of High Dimensional Datasets. IEEE Symposium on Information Visualization 2003 (InfoVis '03), pp. 105–112, 2003.

[31] Schneidewind, J., Sips, M., Keim, D. A. Pixnostics: Towards Measuring the Value of Visualization. Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST '06), pp. 199–206, 2006.

[32] Tatu, A., Albuquerque, G., Eisemann, M., Schneidewind, J., Theisel, H., Magnor, M., Keim, D. A. Combining Automated Analysis and Visualization Techniques for Effective Exploration of High-dimensional Data. Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST '09), pp. 59–66, 2009.

[33] Ankerst, M., Berchtold, S., Keim, D.A. Similarity Clustering of Dimensions for an Enhanced Visualization of Multidimensional Data. Proc. IEEE Symp. on Information Visualization, pp. 52–60, 1998.

[34] Wilkinson, L., Anand, A., Grossman, R. Graph-Theoretic Scagnostics. Proc. of IEEE Symposium on Information Visualization, pp. 157–164, 2005.

[35] Tatu, A., Bak, P., Bertini, E., Keim, D. A., Schneidewind, J. Visual Quality Metrics and Human Perception: An Initial Study on 2D Projections of Large Multidimensional Data. Proceedings of the International Conference on Advanced Visual Interfaces *(*AVI '10*),* pp. 49–56, 2010.

[36] Rensink, R. A., Baldridge, G. The Perception of Correlation in Scatterplots. Computer Graphics Forum (EuroVis '10) 29(3), pp. 1203–1210, 2010.

[37] Chen, Y., Barlowe, S., Yang, J. Click2Annotate: Automated Insight Externalization with Rich Semantics. Proc. of the IEEE Conference on Visual Analytics Science and Technology (VAST '10), pp. 155–162, 2010.

[38] Yang, D., Xie, Z., Rundensteiner, E.A., Ward, M. O. Managing Discoveries in the Visual Analytic Process. SIGKDD Explorations 9(2), pp. 22–29, 2007.

[39] Lowe, D. G. Perceptual Organization and Visual Recognition. Kluwer Academic Publishers, 1985.

[40] Jain, A.K., Dubes, R.C. Algorithms for Clustering Data. Prentice-Hall, 1988.

[41] Frank, A., Asuncion, A. UCI Machine Learning Repository, 2010. Available at http://archive.ics.uci.edu/ml.